МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

МОВЧАН Олексій Васильович

Метод оптимального вибору веб-фреймворків / Method for Web Frameworks Optimal Selection

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КНм-21
О.В. Мовчан
_____

Науковий керівник:
к.т.н., доцент П.Є. Биковий
_____

Кваліфікаційну роботу
допущено до захисту:
«___» _____ 20___ р.
Завідувач кафедри
_____ М.П. Комар

ТЕРНОПІЛЬ - 2023

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «магістр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Мовчан Олексій Васильович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Метод оптимального вибору веб-фреймворків / Method for Web Frameworks Optimal Selection
керівник роботи к.т.н., доцент П.Є. Биковий
затверджені наказом по університету від 8 грудня 2022 року № 491.

2. Строк подання студентом закінченої кваліфікаційної роботи 1 грудня 2023 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити
- огляд предметної області;
- аналіз архітектур та особливостей front-end веб-фреймворків;
- аналіз методів оцінки якості вибору веб-фреймворків;
- постановка задачі дослідження;
- побудова методу оцінювання якості вибору веб-фреймворків;
- вибір основних характеристик;
- запропонований метод оптимального вибору веб-фреймворків;

5. Перелік графічного матеріалу у роботі
- функціональна схема застосунку;
- загальний алгоритм функціонування застосунку.

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 8 грудня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Аналіз предметної області та постановка задачі дослідження	12.2022 р. – 03.2023 р.	
2	Метод оптимального вибору веб-фреймворків	03.2023 р. – 05.2023 р.	
3	Розробка системи оптимального вибору веб-фреймворків	05.2023 р. – 11.2023 р.	
4	Повне завершення та представлення кваліфікаційної роботи на кафедру	01.12.2023 р.	

Студент _____ О.В. Мовчан
підпис

Керівник роботи _____ к.т.н., доцент П.Є. Биковий
підпис

РЕЗЮМЕ

Кваліфікаційна робота на тему «Метод оптимального вибору веб-фреймворків» на здобуття освітнього ступеня «Магістр» зі спеціальності 122 «Комп’ютерні науки» освітньої програми «Комп’ютерні науки» написана обсягом в 76 сторінок і містить 10 ілюстрацій, 2 додатки та 44 використаних джерела.

Метою кваліфікаційної роботи є аналіз підходів і розробка методу оптимального вибору веб-фреймворків для розробки застосунку.

Методи досліджень: узагальнення інформації з літературних джерел та з технічної документації веб-фреймворків. методи вибору оптимальних рішень, методи проведення експериментальних досліджень, методи візуалізації та інтерпретації отриманих результатів.

Результати дослідження: Запропоновано метод оптимального вибору веб-фреймворків, який дає можливість враховувати ряд характеристик веб-фреймворків на представити найбільш оптимальних для заданого проєкту.

Результати роботи можуть успішно застосовуватися для використання на етапі проектування майбутнього веб продукту при виборі веб-фреймворків для розробки.

Ключові слова: ВЕБ-ФРЕЙМВОРК, ВЕБ РОЗРОБКА, ФРОНТ-ЕНД, JAVASCRIPT, ВЕБ ЗАСТОСУНОК.

ABSTRACT

Qualification work on the topic " Method for Web Frameworks Optimal Selection" for the degree in "Master" or Master's degree on speciality 122 «Computer Science» educational and professional program «Computer Science» is written on 76 pages and includes 10 illustrations, 2 annexes 44 sources.

The purpose of the qualification work is to analyze approaches and develop an optimal method for choosing web frameworks for application development.

Research methods: summarizing information from literature sources and technical documentation of web frameworks, methods for selecting optimal solutions, methods for conducting experimental studies, visualization methods, and interpretation of the obtained results.

Research results: An optimal web framework selection method is proposed, which allows considering a range of characteristics of web frameworks to present the most optimal ones for a given project.

The results of the work can be successfully applied during the design phase of a future web product when choosing web frameworks for development.

Keywords: WEB FRAMEWORK, WEB DEVELOPMENT, FRONT-END, JAVASCRIPT, WEB APPLICATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

BOM – Browser Object Model

CLI - Command-Line-Interface

CSS – Cascading Style Sheets

DOM – Document Object Model

HTML – Hyper Text Markup Language

JSX – JavaScript and XML

MVC – Model-View-Controller

MVVM – Model-View-ViewModel

MVW – Model-View-Whatever

SEO – Search Engine Optimization

XML – Extensible Markup Language

ВСТУП

Актуальність теми. Сучасний світ неможливо уявити без всесвітньої мережі Інтернет і багато людей навіть не бачили його таким. Зараз важко знайти сферу життєдіяльності людини де не було б залучено цифрові технології та електронне відображення інформації, переваги якого важко переоцінити. Напрям розвитку людства останніх десятиріч призвів до масового розповсюдження інформації і веб сторінки в мережі Інтернеті займають чи не найбільшу нішу. Варто також зазначити що з розвитком мобільних технологій зв'язку, стала можлива взаємодія користувача і веб сайту через мобільні платформи (смартфони планшети, тощо), що також висунуло свої вимоги до веб сайтів і веб додатків.

Глобальна цифровізація і міграція інформації та сфери послуг в "онлайн" призвела до стрімкого розвитку веб сайтів, збільшення їх різноманіття і розширення переліку задач які вони виконують. Зараз веб сайт це не лише статичний документ в мережі інтернет, а також відео архів, періодичне видання, блог, онлайн магазин, 3D мапа планети, соціальна мережа і багато іншого.

Ця багатофункціональність веб сайтів і мобільних застосунків, в свою чергу, відобразилося на структурі сайтів і їх функціональності, значно ускладнивши проектування і розробку для інженерів, що і стало основною передумовою для виникнення перших бібліотек та повноцінних інфраструктурних програмних рішень, що полегшують розробку складних веб систем.

Однак, треба відзначити що раніше цифрові технології дуже часто були обмежені ресурсними можливості пристроїв, але зараз, з огляду на стрімкий розвиток технологій у виробництві мікропроцесорної техніки, ця проблема поступово зникає, що також впливає на тенденції зміни архітектури веб-додатків. Через це постійно розширюються функціональні можливості існуючих фреймворків і також з'являються нові, спроектовані для більш оптимального вирішення актуальних проблем в розробці.

Кожен інженер-розробник веб застосунків стикається з проблемою вибору оптимального фреймворку для реалізації проекту і часто на це доволі складна задача яка потребує ретельного аналізу багатьох даних. Вхідні дані із технічного завдання клієнта є лише їх малою частиною, адже архітектуру майбутнього застосунку, швидкість та простоту реалізації, його підтримку та потенційні можливості подальшого розширення та масштабування обумовлює саме вибір веб-фреймворку. Тому розробка методики оптимального вибору веб-фреймворків є актуальною задачею.

Мета і завдання дослідження. Метою кваліфікаційної роботи є аналіз підходів і розробка методу оптимального вибору веб-фреймворків для розробки застосунку.

Для досягнення поставленої мети визначено ряд завдань:

- визначити основні вимоги, що висуваються до сучасних веб додатків;
- дослідити основні технології, які використовуються для розробки сучасних веб додатків;
- дослідити основні проблеми при розробці веб додатків і можливі методи їх вирішення за допомогою веб-фреймворків;
- розробити методи оцінювання вибору веб-фреймворку;

Об’єкт дослідження – веб-фреймворки для сучасної розробки веб додатків.

Предмет дослідження – процес оптимального вибору веб-фреймворків.

Методи дослідження. Для вирішення поставлених задач в роботі використовувалися наступні методи:

– узагальнення інформації з літературних джерел;

– узагальнення інформації з технічної документації веб-фреймворків.

– методи вибору оптимальних рішень;

– методи проведення експериментальних досліджень;

– методи візуалізації та інтерпретації отриманих результатів.

Наукова новизна одержаних результатів. Запропоновано метод оптимального вибору веб-фреймворків, який дає можливість враховувати ряд характеристик веб-фреймворків на представити найбільш оптимальних для заданого проєкту.

Публікації та апробація. Результати кваліфікаційної роботи апробовані та опубліковані у матеріалах:

- VIII Науково-практичної конференції молодих вчених і студентів «Інтелектуальні комп'ютерні системи та мережі». 5 грудня 2023 р.
- Міжнародна наукова інтернет-конференція «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення», 7 грудня 2023 р.

Практичне значення отриманих результатів. Запропонований метод надає можливість оптимально підібрати веб-фреймворк для розробки веб додатків базуючись на поставлених задачах і обмеженнях використання майбутнього продукту.

Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел та додатків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Історія і еволюція веб-фреймворків

Як було зауважено вище, за останні роки основною платформою для розповсюдження інформації та послуг стала мережа Інтернет. Таким чином, для успішного ведення бізнесу компаніям, незалежно від розміру чи форми, стало необхідністю бути представленими в Інтернеті і мати там свої веб сторінки або веб застосунки.

Відносно проста документо-орієнтована структура ранніх веб сторінок, що переважно складалися тільки з мови розмітки HTML, перестала задовольняти користувачів і з розвитком інформаційно-комунікаційних технологій вона стала видозмінюватися і наповнюватися динамічним контентом. Веб сторінка отримала змогу реагувати на дії користувача в браузері і виконувати серверний код на основі нових даних.

З плином часу і розвитком технологій стали доступними нові можливості для веб сайтів, такі як: стилізація з допомогою CSS та використання інтерактивних елементів динамічного контенту за допомогою JavaScript, які були створені в середині 90-х років. Програми для відображення інтернет сторінок (інтернет-браузери), такі як Netscape та Internet Explorer, стали більш функціональними.

На той момент, розробка веб сторінки виконувалась суто вручну і була доволі складним та помилконебезпечним процесом через великий вплив людського фактору і відсутність можливості автоматичного тестування. В подальшому веб технології розвивалися ще більш стрімко, вдосконалювалися існуючі мови розмітки і програмування і з’являлись нові технології які мали суттєвий вплив на індустрію і процес розробки веб сторінок. Однією з них була технологія AJAX (Asynchronous JavaScript and XML) що дозволяла надсилати асинхронні запити з браузера на сервер без повного перезавантаження сторінки

[1]. Це дало поштовх до створення більш інтерактивних веб-додатків, що в подальшому розвинулись і сформувались в окремий підтип односторінкових додатків (Single page applications - SPAs) [2]. Поява Single Page Applications стала результатом прагнення зменшити час завантаження сторінок та зробити взаємодію з користувачем більш плавною.

Очевидно що вдосконалення веб сторінок і розширення їх функціональності значним чином ускладнили процес розробки. Тогочасні інженери стикалися з великою кількістю проблем, таких як:

- обмежена можливість маніпулювати DOM: JavaScript був не такий потужний що ускладнювало динамічну зміну вмісту сторінок взаємодію користувача з інтерактивними елементами;
- відсутність стандартів та документації: в перші років веб-розробки було важко знайти надійні рекомендації та документацію щодо кращих практик;
- сумісність браузерів: різні браузери інтерпретували HTML та CSS по-різному, що призводило до проблем із сумісністю та відображенням;
- маніпуляція зображеннями та мультимедійним контентом: відсутність потужних інструментів для роботи з графікою та мультимедійним контентом ускладнювала розміщення зображень та мультимедійних елементів на сторінці;
- інтерактивність та анімації: динамічні ефекти та анімовані зміни контенту були складні для досягнення без JavaScript-бібліотек.

Всі ці проблеми й спричинили потребу в розвитку бібліотек та фреймворків, які полегшували б процес розробки та забезпечували більшу ефективність та надійність у розробці веб-сторінок. Якщо розглянути історичний розвиток лише

Перша бібліотека JavaScript під назвою jQuery була розроблена Д. Резігом у 2006 році на базі бібліотеки cssQuery, створеної Д. Едвардсом [7, 8]. jQuery є JavaScript-бібліотекою, призначеною для спрощення навігації по дереву HTML DOM та його маніпулювання, а також обробки подій, CSS-анімації та використання технології Ajax [3].

Проте на початку, бібліотека відповідала лише частково вимогам розробки того періоду, і в 2010 році компанія Google представила AngularJS, що вказувало на необхідність структурованого підходу до розробки front-end. AngularJS надавав можливість створювати односторінкові додатки (Single Page Applications - SPA) та дозволяв прив'язувати дані до DOM-елементів. Декілька років потому Facebook (зараз Meta) випускає React, який пропонує віртуальну DOM та компонентний підхід до розробки інтерфейсів. React надає високу ефективність і дозволяє швидше відображати складні веб-сторінки.

Але AngularJS та React при всіх своїх перевагах мали кілька суттєвих недоліків. Через широкий функціонал і потребу вирішувати багато проблем розробки вони доволі громіздкі і мають високий поріг входу нових інженерів тобто важкі для освоєння, що в свою чергу відображається на швидкості розробки складних проектів та оволодіння ними новими спеціалістами.

В 2014 році з'явився Vue.js який позиціонувався як легкозасвоюваний фреймворк, який дозволяє поступово впроваджувати сучасні підходи до розробки. Vue був створений для усунення недоліків Angular, але насправді він поєднує найкращі частини двох найпопулярніших фреймворків: AngularJS і ReactJS.

Розуміючи наявні недоліки свого фреймворку і неможливість їх виправити в наступних оновленнях, Google прийняла рішення його кардинально переробити і випускає нову версію, яка є зовсім відмінною від попереднього AngularJS. Цей фреймворк отримав назву Angular і став більш швидким, ефективним і підтримує сучасні стандарти розробки.

На теперішній час React, Angular і Vue.js залишаються найпопулярнішими фреймворкуми для front-end розробки. Однак треба відмітити що jQuery досі є найпоширенішою бібліотекою і станом на серпень 2022 року jQuery використовують 77% із 10 мільйонів найпопулярніших веб-сайтів [5]. Веб-аналіз показує, що це найпоширеніша бібліотека JavaScript із великим відривом, щонайменше в 3-4 рази частіше, ніж будь-яка інша бібліотека JavaScript [5, 6].

Подальший розвиток веб-фреймворків можна описати як поступове удосконалення і розширення функціоналу, згідно до вимог сучасних веб технологій.

Розповсюдження і ріст популярності мовних моделей на основі штучного інтелекту вже призвів до залучення його до інструментів розробки. Такі інструменти, як CoPilot від GitHub та Ghostwriter від Replit, які допомагають у кодуванні, є першими показниками розширення ролі штучного інтелекту в програмуванні, що свідчить про майбутнє, де штучний інтелект поширюватиметься не тільки на допомогу, а й на повне керування процесом програмування. Уявіть звичайний сценарій, коли програміст забуває синтаксис для реверсу списку на певній мові. Замість пошуку на онлайн-форумах і статтях CoPilot пропонує негайну допомогу, зосереджуючи програміста на досягненні мети [4].

З огляду на це можна спрогнозувати появу елементів штучного інтелекту в майбутніх ітераціях веб-фреймворків або інтерфейсів для доступу до їх API для більш динамічної розробки програмного продукту.

1.2 Огляд сучасних веб-фреймворків

Як і веб загалом, процес розробки веб застосунків та сайтів постійно змінювався і вдосконалювався, також і інструменти розробки зазнали змін. Основний набір технологій для розробки front-end включає в себе HTML, CSS і JavaScript. Основною мовою для створення веб-сторінок є мова розмітки гіпертексту HTML (Hyper Text Markup Language). Вона використовується для логічної структури веб-сторінки (документа) та формування загальної структури документа. Використання HTML має переваги у простоті освоєння та можливостях вибору редактора для написання коду.

Для створення простої HTML-сторінки вистачає, проте використання складного форматування при створенні HTML-документа та зміни вигляду

елементів сторінки може призводити до труднощів і збільшення обсягу кінцевого документа. У таких випадках застосовується CSS (Cascading Style Sheets - каскадні таблиці стилів). CSS може бути підключений як окремий файл або вбудований у код розмітки, де використовується спеціальна макромова для задання форматування сторінки. Тобто, файл CSS виступає як шаблон, для форматування тексту, таблиць та інших компонентів HTML-документа. Той самий файл CSS можна підключати до різних сторінок. Мінусами цієї технології є відсутність підтримки CSS старими браузерами, проте цей відсоток користувачів постійно зменшується з плином часу.

Для додання динаміки (випадаючі меню, анімація) або інтерактивності до веб-сторінок використовується мова написання скриптів JavaScript.

JavaScript – це мова програмування створена компанією Netscape. Вона використовується в структурі HTML-сторінок для розширення їхнього функціоналу. JavaScript розроблена на основі мови Java від корпорації Sun. Використання JavaScript не вимагає встановлення або налаштування будь-яких додаткових модулів на сервері, оскільки вона виконуються безпосередньо у браузері користувача.

HTML, CSS і JavaScript є основними компонентами веб-документа, проте у браузерах документи представлені як набір об'єктів, які утворюють об'єктну модель браузера (BOM - Browser Object Model). Веб-консорціум запропонував стандартизувати об'єктну модель документа (DOM - Document Object Model) як стандартний спосіб представлення веб-сторінок за допомогою набору об'єктів.

На противагу об'єктній моделі браузера, DOM включає лише набір об'єктів, які стосуються вмісту документа і не містить об'єктів, що відповідають за управління вікнами та рамками вікон (рис. 1.1). DOM є інтерфейсом програмування для HTML та XML докуметів. Вона опрацбовує сторінки з можливістю програмам змінювати структуру, стиль та вміст документа. DOM відображає документ у вигляді вузлів та об'єктів, що дозволяє мовам програмування взаємодіяти зі сторінкою.

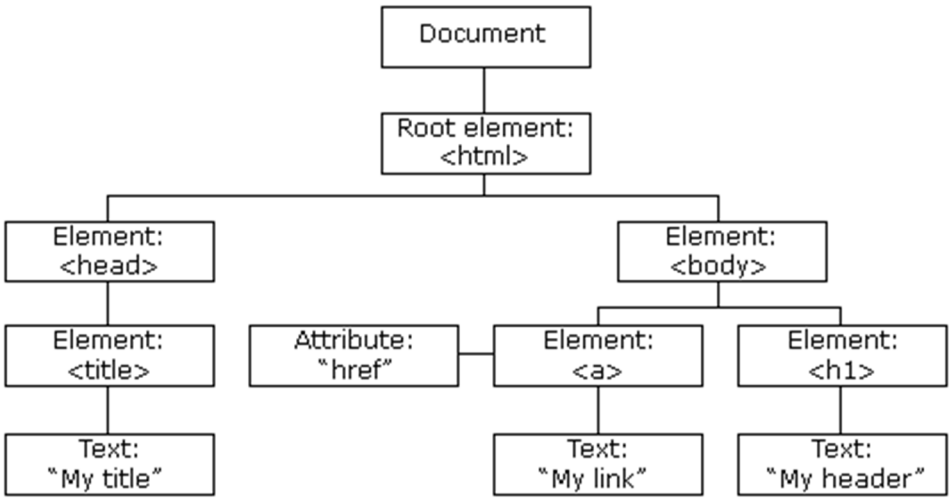


Рисунок 1.1 – Об’єктна модель документа (DOM)

Сучасні веб-фреймворки у front-end розробці часто використовують підхід з маніпуляції DOM об’єктами для репрезентування інформації користувачеві без перезавантаження сторінки. Тобто в злежності від дій користувача, той чи іншие елемент може з’являтися, зникати або змінюватися. Розглянемо найбільш поширені на даний момент:

1.2.1 React

React (React.js / ReactJS) - безкоштовна бібліотека JavaScript із відкритим кодом [11, 12] для створення інтерфейсів користувачів. React був розроблений у 2013 році компанією Meta (раніше Facebook) і спільнотою окремих розробників і компаній і швидко набув популярності завдяки своєму декларативному та компонентному підходу до розробки інтерфейсу користувача [13-15].

React можна використовувати для розробки мобільних, серверних та односторінкових додатків. Оскільки React пов’язаний лише з інтерфейсом користувача та відтворенням компонентів у DOM, програми React часто покладаються на бібліотеки для маршрутизації та інших функцій на стороні клієнта.

Основна архітектура React обертається навколо компонентів. Компоненти можна представити як модульні багаторазові будівельні блоки, які інкапсулюють певні елементи інтерфейсу користувача та їх поведінку. Цей компонентний

підхід покращує модульність коду, зручність обслуговування та багаторазове використання.

Однією з ключових інновацій React є Virtual DOM. Замість того, щоб безпосередньо маніпулювати фактичним DOM, React створює його віртуальне представлення в пам’яті. Коли стан або властивості змінюються, React обчислює різницю між поточним і віртуальним DOM. Потім він оновлює лише необхідні частини фактичного DOM. Це мінімізує прямі маніпуляції DOM, що призводить до покращення продуктивності.

React використовує JSX, розширення синтаксису для JavaScript. JSX дозволяє розробникам писати компоненти інтерфейсу користувача за допомогою XML-подібного синтаксису. Це полегшує створення елементів інтерфейсу користувача в більш читабельний і виразний спосіб. JSX транслюється в JavaScript перед відтворенням у браузері.

React забезпечує односпрямований потік даних. Дані передаються від батьківських компонентів до дочірніх компонентів через реквізити. Коли стан компонента змінюється, React ефективно оновлює інтерфейс користувача, забезпечуючи передбачуваний потік даних.

Компоненти React мають методи життєвого циклу, які розробники можуть використовувати для виконання дій на різних етапах, таких як створення, оновлення та знищення компонентів. Це дозволяє виконувати певну логіку в різних точках життєвого циклу компонента. Основними перевагами React можна назвати:

- багаторазові компоненти, де компонентна архітектура React сприяє створенню повторно використовуваних компонентів інтерфейсу користувача. Розробники можуть інкапсулювати функціональні можливості та стилі всередині компонентів, сприяючи повторному використанню коду та зручності обслуговування;

- віртуальний DOM для продуктивності, що забезпечує ефективне оновлення шляхом мінімізації прямого маніпулювання фактичним DOM. Це

призводить до швидшого рендерингу та покращення загальної продуктивності, особливо в програмах із динамічними даними, які часто змінюються;

- декларативний синтаксис, що дозволяє розробникам описувати бажаний результат, а React піклується про основні процеси. Це спрощує код, робить його більш читабельним і менш схильним до помилок;

- сильна підтримка спільноти, що перетворюється на велику кількість зовнішніх додаткових бібліотек, інструментів і ресурсів. Велика кількість вмісту, створеного спільнотою, полегшує навчання та вирішення проблем;

- React Native для мобільної розробки, що є розширенням React, дозволяє розробляти власні мобільні програми за допомогою JavaScript і React. Це дозволяє повторно використовувати код між веб-додатками та мобільними додатками, спрощуючи процес розробки.

Незважаючи не це, треба зауважити і недоліки властиві цьому фреймворку:

- складний для новачків, особливо для тих, хто не знайомий з такими концепціями, як JSX, компоненти та віртуальний DOM. Однак після засвоєння ці поняття сприяють більш ефективному розвитку;

- складність інструменту, оскільки React включає різні інструменти для таких завдань, як групування, транспіляція та управління станом. Велика кількість варіантів може вразити новачків. Однак ця різноманітність пропонує розробникам гнучкість у виборі інструментів, які найкраще відповідають їхнім потребам;

- шаблонний код, оскільки додаткам React іноді може знадобитися шаблонний код, особливо для таких завдань, як налаштування нового проекту або керування станом. Хоча такі інструменти, як Create React App, спрямовані на спрощення цього процесу, деякі проекти все ж можуть включати певний рівень шаблонності.

- складність архітектури Flux, що про проявляє себе коли ви маєте справу з великомасштабними програмами, керування станом у React може стати складним. Хоча React надає деякі функції керування станом, прийняття додаткових шаблонів, таких як Flux, або використання зовнішніх бібліотек

(наприклад, Redux) стає необхідним для більш організованого та масштабованого керування станом;

- складнощі з SEO, оскільки програми React здійснюють візуалізацію зі сторони клієнта, можуть зіткнутися з проблемами пошукової оптимізації (SEO - Search Engine Optimization). Хоча такі рішення, як відтворення на стороні сервера (SSR), існують, їх реалізація вимагає додаткових зусиль.

Незважаючи на труднощі, переваги React, включаючи можливість повторного використання, оптимізацію продуктивності та надійну спільноту, продовжують позиціонувати React як кращий вибір для інтерфейсної розробки. Оскільки React розвивається, він залишається ключовим інструментом у наборі інструментів веб-розробників, стимулюючі інновації та формуючи майбутнє розробки інтерфейсу користувача.

1.2.2 AngularJS

AngularJS це платформа JavaScript з відкритим вихідним кодом, розроблена та підтримується Google, відіграла ключову роль у формуванні сучасної веб-розробки. AngularJS, випущений у 2010 році, запровадив зміну парадигми, представивши декларативне програмування та потужну архітектуру модель-представлення-контролер (MVC) [25].

Особливістю архітектури AngularJS є гнучкий шаблон через це його часто називають фреймворком MVW (Model-View-Whatever, Модель-Представлення-Щозавгодно), де “Whatever” (Щозавгодно) охоплює різноманітні архітектурні шаблони, включаючи Model-View-Controller (MVC) і Model-View-ViewModel (MVVM). Ця гнучкість дозволяє розробникам вибирати архітектурний шаблон, який найкраще підходить для їх застосування.

Однією з видатних функцій AngularJS є двостороннє зв’язування даних. Це означає, що зміни в інтерфейсі користувача (View) автоматично відображаються в даних програми (Model), і навпаки. Цей двонаправлений потік даних зменшує шаблонний код і покращує синхронізацію між представленням і моделлю.

AngularJS використовує ін’єкцію залежностей, шаблон проектування, який сприяє створенню слабо зв'язаних компонентів. Впровадження залежностей полегшує керування залежностями між різними частинами програми, що призводить до створення більш модульного та зручного для обслуговування коду [26].

Директиви є основною функцією AngularJS, що дозволяє розробникам розширювати HTML за допомогою спеціальних елементів і атрибутів. Директиви дозволяють створювати повторно використовувані компоненти, інкапсулюючи як поведінку, так і презентацію. Це сприяє чіткішому розподілу завдань у кодовій базі.

AngularJS використовує служби як єдині елементи для інкапсуляції бізнес-логіки, обробки даних та інших спільних функцій. Послуги можна вставляти в різні компоненти, що сприяє повторному використанню коду та зручності обслуговування [27].

Перевагами AngularJS можна назвати:

- декларативне програмування, що представляє декларативний стиль програмування, де розробники описують бажаний результат, а AngularJS обробляє основні процеси. Цей підхід веде до більш читабельного та виразного коду;
- двостороннє прив’язування даних, що спрощує синхронізацію між представленням і моделлю. Це зменшує потребу в ручних маніпуляціях DOM і покращує чуйність інтерфейсу користувача;
- модульність і багаторазове використання, завдяки яким сприяють такі функції, як директиви та служби, сприяє повторному використанню коду та зручності обслуговування. Розробники можуть створювати інкапсульовані компоненти та повторно використовувати їх у різних частинах програми;
- ін'єкція залежності, що полегшує створення слабо зв'язаних компонентів. Це не лише покращує модульність коду, але й спрощує тестування та заохочує ефективні методи проектування;

- багата екосистема та підтримка спільноти, де розробники можуть використовувати широкий спектр сторонніх бібліотек, інструментів і розширень, прискорюючи розробку та вирішення проблем.

Але також треба зауважити на недоліки властиві AngularJS:

- складність використання, оскільки містить великий набір функцій, включаючи директиви, служби та ін'єкцію залежностей, що може бути складним для новачків;

- шаблонний код, оскільки додаткам AngularJS для таких завдань, як конфігурація та налаштування зазвичай необхідно такий код. Хоча такі інструменти, як Angular CLI, спрямовані на оптимізацію цього процесу, деякі проекти все ще можуть включати певний рівень шаблонності.

- накладні витрати на цикл дайджесту, що являє собою механізм в AngularJS, відповідальний за виявлення змін і оновлення представлення, може призвести до надмірних витрат на продуктивність у великих програмах. Щоб пом'якшити цю проблему, потрібна ретельна оптимізація та використання найкращих практик.

- перехід спільноти на Angular (2+), оскільки Angular еволюціонувала, причому Angular (2+) є значним відхиленням від AngularJS з точки зору архітектури та синтаксису. Ця зміна може призвести до зменшення підтримки спільноти та оновлень для AngularJS у довгостроковій перспективі;

AngularJS залишив незгладимий слід у сфері веб-розробки, вплинувши на дизайн наступних фреймворків і бібліотек. Його двостороннє зв'язування даних, модульна архітектура та впровадження залежностей стали характерними рисами сучасної розробки інтерфейсу.

1.2.3 Vue JS

Vue.js - прогресивний фреймворк JavaScript для створення інтерфейсів користувача, отримав значну популярність у спільноті веб-розробників після свого випуску в 2014 році. Vue.js, відомий своєю простотою, гнучкістю та

легкістю інтеграції, що дає розробникам змогу створювати динамічні та реактивні програми [28]. Vue.js має наступні архітектурні рішення:

- компонентна архітектура, де компоненти, які інкапсулюють як структуру, так і поведінку, можна складати та повторно використовувати у всій програмі. Цей модульний підхід сприяє організації коду, повторному використанню та зручності обслуговування;

- реактивність, оскільки Vue.js відстежує залежності між властивостями даних і автоматично оновлює DOM, коли базові дані змінюються. Цей реактивний підхід спрощує процес розробки та покращує швидкість реакції інтерфейсу користувача;

- віртуальний DOM, що використовується для оптимізації продуктивності візуалізації. Віртуальний DOM мінімізує пряме маніпулювання фактичним DOM шляхом створення представлення в пам'яті. Коли відбуваються зміни, Vue.js обчислює мінімальну кількість оновлень, необхідних для синхронізації віртуальної DOM із фактичною DOM;

- директиви, які дозволяють розробникам розширювати HTML додатковими функціями. Директиви, такі як v-if, v-for і v-bind, дозволяють декларативне відтворення та маніпулювання DOM на основі умов даних. Це спрощує розробку динамічних та інтерактивних інтерфейсів користувача;

- компоненти одного файла, що є концепцією однофайлових компонентів (SFC), де шаблон компонента, сценарій і стилі інкапсульовані в одному файлі. Цей модульний підхід покращує організацію коду та його читабельність, одночасно полегшуючи використання препроцесорів, таких як Babel і Sass [29].

Перевагами Vue.js можна вважати розширення можливостей веб-розробників за рахунок наступних факторів:

- простота інтеграції, коли розробники можуть поступово впроваджувати Vue.js у існуючі проекти, забезпечуючи поступовий перехід без важкого процесу навчання за вимоги освоєння більшості функціоналу фреймворку одразу. Крім того, Vue.js можна легко інтегрувати з іншими бібліотеками або існуючими проектами;

- гнучкість і прогресивна структура, оскільки Vue.js часто називають "прогресивним фреймворком" через його поступове впровадження та гнучкість. За потреби розробники можуть використовувати лише частини фреймворку, що робить його придатним для широкого діапазону проектів, від невеликих додатків до великомасштабних корпоративних рішень;

- Vue CLI та DevTools, де офіційний Vue CLI (command-Line-Interface - інтерфейс командного рядка) спрощує налаштування проекту та робочі процеси розробки. Vue DevTools як розширення для браузера, надає набір інструментів для налагодження та перевірки програм Vue.js, покращуючи досвід розробки;

- активна спільнота та екосистема, що сприяє доступності сторонніх бібліотек, плагінів та інструментів;

- детальна документація, що охоплює всі аспекти фреймворку, що робить її безцінним ресурсом як для початківців, так і для досвідчених розробників. Чіткі приклади та посібники полегшують процес навчання та розробки.

Розглянемо недоліки властиві цьому фреймворку:

- низька корпоративна підтримка, що порівняно з такими популярними фреймворками, як React (Facebook / Meta) і Angular (Google), Vue.js має меншу корпоративну підтримку. Хоча фреймворк підтримується широкою спільнотою та його творцем, Еваном Ю, при виборі майбутнього фреймворку для розробки свого продукту, деякі компанії можуть віддати перевагу фреймворком із сильнішою корпоративною підтримкою;

- розповсюдженість та зрілість, оскільки через відносну новизну фреймворку він може сприйматися як менш зрілий, ніж React або Angular, особливо з точки зору впровадження на підприємствах. Деякі великі компанії можуть віддати перевагу фреймворкам із більш тривалим досвідом і усталеним використанням у галузі.

Vue.js став потужним гравцем у світі веб-розробки, пропонуючи баланс між простотою та потужністю. Його компонентна архітектура, система реагування та гнучкість заслужили для нього віддану та зростаючу базу

користувачів. Він продовжує розвиватися з постійними вдосконаленнями, сильною спільнотою та прагненням забезпечити чудовий досвід розробника.

Оскільки організації оцінюють свій вибір для розробки інтерфейсу, Vue.js виділяється як універсальна та ефективна структура. Його легкість інтеграції, прогресивний характер і акцент на простоті роблять його привабливим варіантом для проектів різного масштабу та складності.

1.2.4 Ember.js

Ember.js, фреймворк JavaScript з відкритим вихідним кодом, був популярним для розробників які створюють амбітні веб-додатки, починаючи з моменту свого дебюту в 2011 році. Підхід до конфігурації та зосередженість на продуктивності дозволили Ember.js розвинутися, щоб задовольнити потреби масштабних, складних проектів. Особливостями Ember.js є:

- угода щодо конфігурації, де Ember.js непохитно дотримується парадигми конфігурації. Це означає, що розробники звільнені від прийняття численних рішень щодо структури своєї програми. Ember.js накладає набір умовностей, оптимізуючи розробку та сприяючи узгодженості між проектами;

- архітектура модель-представлення-контролер (MVC): В основі Ember.js лежить архітектура Model-View-Controller (MVC) з усіма своїми перевагами. В цьому контексті Ember.js використовує маршрутизатор для керування станом програми та переходами між різними маршрутами;

- механізм створення шаблонів керування, таких як Handlebars для динамічного відтворення переглядів. Handlebars полегшує створення шаблонів шляхом вбудовування виразів у подвійні фігурні дужки (подібно до Liquid code). Потім ці вирази замінюються відповідними значеннями під час виконання;

- Ember.js містить Ember Data, бібліотеку для керування моделями та зв’язками в програмі. Ember Data бездоганно інтегрується в екосистему Ember.js, забезпечуючи стандартизований спосіб взаємодії з серверними API;

- Ember CLI (інтерфейс командного рядка, інновація Ember яку згодом використали в VueJS) - це потужний інструмент, який автоматизує типові

завдання розробки. Це оптимізує налаштування проекту, створення макетів, тестування та розгортання.

Всі ці особливості надають Ember.js певні переваги над іншими фреймворками:

- конвенційний підхід Ember.js значно підвищує продуктивність розробника. Дотримуючись встановлених умов, розробники можуть більше зосередитися на логіці програми, а не витрачати час на налаштування інфраструктури;

- Ember.js надає структуровану та впевнену структуру для створення програм. Ця властива структура забезпечує узгодженість проектів і полегшує співпрацю розробників, особливо у великих командах;

- Ember CLI служить комплексним набором інструментів для розробки, автоматизації завдань і надання узгодженого інтерфейсу. Це спрощує звичайні робочі процеси, такі як тестування, збірка та розгортання, сприяючи більш оптимізованому процесу розробки;

- Ember.js може похвалитися багатою екосистемою з різноманітними доповненнями та розширеннями. Ця екосистема забезпечує низку функціональних можливостей, від автентифікації користувачів до візуалізації даних, що дозволяє розробникам використовувати готові рішення та прискорювати розробку;

- Ember Inspector - розширення для браузера, яке покращує можливості налагодження (рішення теж було згодом іикористане у VueJS). Він надає інформацію про структуру програми, маршрути, компоненти та служби, допомагаючи розробникам ефективно виявляти та вирішувати проблеми.

Конвенційний характер Ember.js, хоч і сприятливий для продуктивності, але може стати проблемою для новачків в освоєнні цього інструменту розробки. Як і перелічені наступні недоліки фреймворка:

- менша гнучкість: Жорсткі ідеї та конвенції Ember.js, хоч і є перевагами в багатьох сценаріях, можуть обмежити гнучкість для певних вимог проекту.

Розробники, які шукають архітектуру з високим ступенем індивідуальності, можуть вважати Ember.js обмеженнями;

- накладні витрати для малих проектів: Для невеликих проектів або програм із простими вимогами умовності та структура Ember.js можуть створити непотрібну складність і накладні витрати. У таких випадках легша структура або бібліотека можуть бути більш доречними;
- розмір спільноти: Хоча Ember.js має спеціальну спільноту, вона дещо нішева і не така велика, як деяких інших фреймворків. Це може вплинути на доступність сторонніх ресурсів, навчальних посібників і підтримки порівняно з більш поширеними фреймворками.

Ember.js зайняв свою нішу як надійна структура для створення амбітних веб-додатків. Його впевнений характер, традиційний підхід до конфігурації та потужний інструментарій роблять його привабливим вибором для проектів зі складними вимогами та великими командами розробників.

Оскільки ландшафт веб-розробки розвивається, Ember.js продовжує свій шлях, адаптуючись до нових викликів і враховуючи відгуки своєї спільноти. Розробники, які шукають структуру, яка підкреслює конвенційність, структуру та продуктивність, знайдуть Ember.js як надійного супутника в їх прагненні створювати масштабовані веб-програми, які можна підтримувати. Незалежно від того, чи Ember.js залишається непохитним вибором, чи поступається новим технологіям, його вплив на історію веб-розробки є значущим.

1.2.5 jQuery

Хоча jQuery і не є повноцінним фреймворков, цю бібліотеку варто розглянути через те що її вплив на розвиток веб розробки був дійсно революційний. Вона була розроблена на початку 2000-х років, виникла як одна з перших бібліотек, яка змінила ландшафт клієнтської веб-розробки. jQuery, шанована своєю простотою, кросбраузерною сумісністю та потужними маніпуляціями з об’єктною моделлю документа (DOM), став вибором для незліченної кількості розробників, які прагнули підвищити інтерактивність і

оперативність своїх веб-сайтів. Основною перевагою за яку цінували й досі використовують багато розробників була маніпуляції DOM. За своєю суттю jQuery — це в основному бібліотека для маніпулювання DOM. Він забезпечує стислий і виразний синтаксис для вибору елементів HTML і керування їхніми атрибутами, вмістом і стилями. Відомий знак долара ($) слугує скороченням для функції jQuery, роблячи код лаконічним і читабельним. Також ключовимим особливостями цієї бібліотеки, яки зробили її настільки розповсюдженою, є:

- обробка подій, де jQuery спрощує обробку подій, дозволяючи розробникам легко приєднувати слухачі подій до елементів HTML. Будь то клацання, наведення або будь-яка інша подія, jQuery надає уніфікований інтерфейс для обробки безлічі взаємодій користувача;
- AJAX: jQuery популяризував AJAX (асинхронний JavaScript і XML), абстрагуючись від складності XMLHttpRequest. Функція $.ajax() забезпечує безперебійний асинхронний зв’язок із сервером, спрощуючи пошук і маніпулювання даними без необхідності оновлення сторінки;
- ефекти та анімація, де jQuery містить безліч вбудованих ефектів і анімації. jQuery спрощує створення динамічних і візуально привабливих користувацьких інтерфейсів, будь то згасання елементів, переміщення їх угору чи вниз або власна анімація;
- плагіни, де jQuery пропонує широкий набір додаткових функцій, від повзунків зображень до складних візуалізацій даних. Ця можливість розширення покращує корисність jQuery та адаптивність до різноманітних вимог проекту.

Головною перевагою бібліотеки є кросбраузерність. На початку 2000-х років веб-розробники зіткнулися з проблемами непослідовних реалізацій браузерів. jQuery став рятівником, абстрагувавши ці невідповідності, забезпечивши уніфікований інтерфейс, який бездоганно працював у різних браузерах:

- простота використання, де синтаксис jQuery розроблений інтуїтивно зрозумілим і лаконічним. Його API зручний для початківців, що дозволяє розробникам виконувати складні завдання лише за допомогою кількох рядків

коду. Така простота використання значно сприяла широкому впровадженню jQuery;

- підтримка спільноти, де доступні незліченні навчальні посібники, форуми та плагіни, що полегшує розробникам пошук рішень типових проблем або вдосконалення своїх проектів додатковими функціями;

- швидкий розвиток, де для проектів зі стислими термінами виконання або простими вимогами здатність jQuery оптимізувати типові завдання прискорює розробку. Він чудово підходить у сценаріях, де основна увага приділяється швидкому впровадженню інтерактивних функцій без потреби у комплексній структурі додатків;

- вивчення jQuery часто вважають воротами у світ JavaScript і веб-розробки. Його синтаксис і концепції доступні, що робить його чудовою відправною точкою для тих, хто новачок у розробці інтерфейсу.

При всій простоті і революційності підходів, jQuery також не була позбавлена певних недоліків:

- продуктивність, що передбачає введення рівня абстракції, який може вплинути на продуктивність у певних сценаріях. Для сучасних веб-додатків зі складною взаємодією можна віддати перевагу більш легким альтернативам;

- надмірна залежність, де через його історичну популярність деякі розробники можуть надмірно покладатися на jQuery, навіть у ситуаціях, коли сучасний JavaScript і можливості браузера пропонують більш ефективні рішення. Надмірне використання jQuery в сучасній веб-розробці може призвести до неоптимальної продуктивності;

- великий розмір файлу, де включення всієї бібліотеки jQuery для простих проектів може призвести до непотрібного роздування. Останнім часом із збільшенням уваги до оптимізації продуктивності розробники часто розглядають компроміс між зручністю jQuery та бажанням мати менші веб-сайти, які швидше завантажуються;

- глобальне забруднення простору імен, де jQuery приєднується до глобального простору імен JavaScript, що потенційно може призвести до

конфліктів імен у великих проектах або під час інтеграції кількох бібліотек. Ця проблема призвела до прийняття більш модульних і обмежених підходів у сучасній розробці JavaScript;

- еволюція веб-стандартів, де з розвитком веб-стандартів і можливостей веб-переглядача багато функцій, для вирішення яких спочатку був розроблений jQuery, стали частиною рідного JavaScript API. Отже, для сучасних проектів потреба в jQuery не така виражена, як раніше.

Вплив jQuery на веб-розробку незаперечний. Він відіграв ключову роль у спрощенні створення інтерактивних і динамічних веб-сторінок в епоху неузгодженості браузерів. Хоча його популярність зменшилася з еволюцією веб-стандартів і розвитком сучасних фреймворків JavaScript, jQuery залишається свідченням трансформаційної сили добре спроектованої бібліотеки.

Розробники продовжують використовувати jQuery для конкретних сценаріїв, де його переваги, такі як кросбраузерна сумісність і простота використання, переважують потенційні недоліки. Його спадщина зберігається не лише в безлічі веб-сайтах, які все ще використовують його, але також у принципах і шаблонах, які він популяризував, сприяючи основам сучасної інтерфейсної розробки.

1.3 Постановка задачі дослідження

Існує багато фреймворкив які використовуються для розробки front-end частини застосунків. Також є back-end фреймворки які допомагають розробляти серверний код написаня на різних мовах програмування Laravel, Symfony, CodeIgniter, Zend, Yii - на PHP; Django, CherryPy, Pyramid, Flask - на Python; Ruby on Rails на Ruby, ExpressJS на NodeJS.

Вибір фреймворку, потрібного для реалізації того чи іншого проекту, в основному залежить від того яку мову програмування розробник обрав для написання серверної частини застосунку та додаткових параметрів котрі не

завжди враховуються. Тому запропонований метод оптимального вибору веб-фреймворків дасть змогу оптимально провести оцінювання існуючих веб-фреймворків.

Метод повинен бути реалізований програмно, буде мати змогу обробляти вхідні дані по особливостях проєкту, які визначає користувач і по оцінюванню веб-фреймворків: наскільки оптимальним буде застосувати той чи інший веб-фреймворк для реалізації відповідного проекту.

Оцінювання програмної розробки (проекту) повинно забезпечувати наступне:

- можливість вибору користувачем важливих характеристик проекту;
- можливість вибору користувачем одного або більше фреймворків для порівняння;
- можливість відображення інформації про результати оцінювання оптимальності вибору;
- можливість розширення в подальших ітераціях через додавання нових фреймворків чи особливостей проекту;

Програма повинна бути веб-застосунком, який може запускатися як в мережі Інтернет у браузері, так і локально. Апаратна підтримка не складає особливо високих вимог, лише персональний комп'ютер або інший пристрій, який може відображати веб-сторінки:

- процесор з тактовою частотою 1,6 ГГц (архітектура x86 або x64) і вище;
- обсяг оперативної пам'яті (ОЗУ) не менше 1 ГБ;
- обсяг внутрішнього накопичувача не менше 20 ГБ;
- наявність пристроїв введення (миша, клавіатура);
- наявність монітора.

Базовою програмною платформою повинен бути сучасний веб браузер, версії не нижче за: Google Chrome v.56; Mozilla FireFox v.50; Microsoft Edge v.15; Internet Explorer v.11; Safari v.10.

Реалізація методу оптимального вибору веб-фреймворка повинен забезпечити наступні функції:

- ввід даних про існуючий веб-фреймворк

- вибір характеристик веб-фреймворку;
- вивід існуючих в системі даних;
- відображення результатів оцінювання;
- можливість автоматизованого розрахунку;
- можливість видалення/редагування в системі внесених фреймворків та їх характеристик.

Висновки до розділу 1

1. Проведено огляд історії та еволюції веб-фреймворків.
2. Проаналізовано найбільш популярні сучасні веб-фреймворки, що дало змогу виділити їх основні відмінності, переваги і недоліки.
3. Поставлено задачу дослідження, що дало змогу розпочати розробку та реалізацію методу.

2 МЕТОД ОПТИМАЛЬНОГО ВИБОРУ ВЕБ-ФРЕЙМВОРКІВ

2.1 Методи оцінювання програмного забезпечення

Веб-фреймворк як інструмент розробки веб застосунків є елементом програмного забезпечення і до нього можуть бути застосовані методики оцінювання програмного забезпечення.

Задача оцінювання програмного забезпечення з’явилася доволі давно. Вважається, що результатом такого оцінювання повинен стати деякий інтегральний показник якості програмного забезпечення.

Під поняттям якості програмного забезпечення (ПЗ) розуміють сукупність його властивостей, що обумовлюють здатність задовольняти певні потреби користувачів і спеціалістів, що беруть участь у розробці і підтримці ПЗ.

Властивістю (характеристикою) ПЗ називають об'єктивну особливість програми або документації, яка виявляється при його розробці, використані та підтримці.

Показник якості програми представляє собою кількісну міру властивостей ПЗ, яка розглядається в рамках умов його розробки, використання та підтримки. Також можуть використовуватися якісні оцінки, відомі як ознаки [17].

Показники якості програмного забезпечення можуть бути класифіковані за кількістю характеристик властивостей, поділяючись на одиничні та комплексні (групові). Одиничний показник відноситься до конкретної властивості, тоді як комплексний охоплює декілька характеристик ПЗ одночасно.

Оцінювання може призводити до формування загальної інтегральної оцінки якості програмного забезпечення. Оцінка якості ПЗ базується на міжнародних стандартах, таких як:

- стандарт IEEE 1061-1998 для методології показників якості програмного забезпечення, де якість ПЗ визначається як ступінь наявності необхідної комбінації властивостей;
- стандарт ISO 8402:1994 управління якістю та забезпечення якості, де

якість ПЗ визначається як сукупність характеристик, що відносяться до його здатності задовольняти встановлені і передбачувані потреби.

На сьогоднішній день багато розробників користуються багаторівневою моделлю визначення якості програмного забезпечення, яка заснована на стандарті ISO 9126, де висвітлено 6 характеристик якості ПЗ, котрі визначаються наборами атрибутів з відповідними метриками (див. рисунок 2.1).

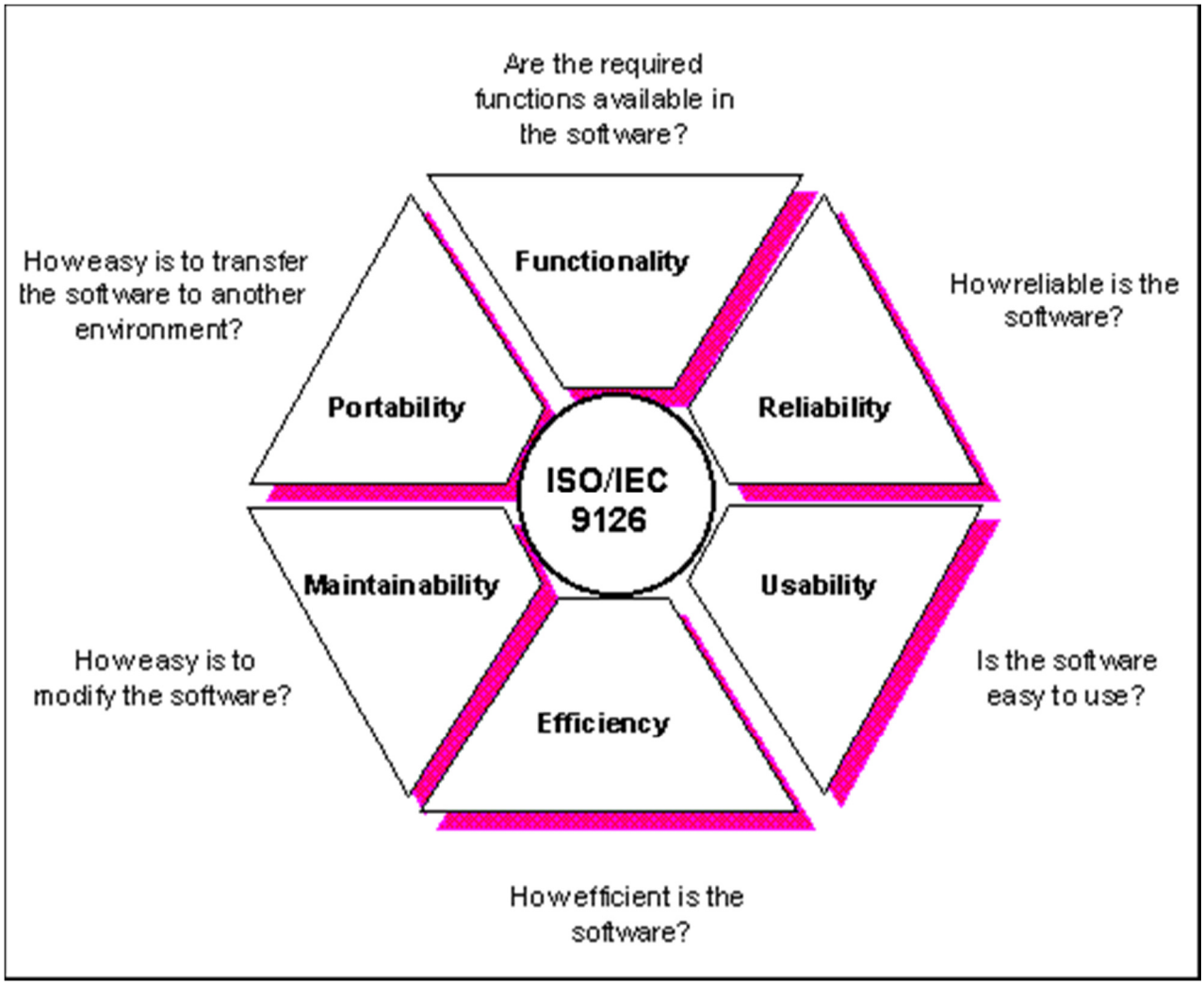


Рисунок 2.1 – Узагальнена модель оцінки якості програмного забезпечення згідно ISO 9126

Ця узагальнена модель служить основою для інших моделей оцінки якості програмного забезпечення, які ґрунтуються на критеріях якості та пов’язаних із ними показниках (метриках). Метрики якості можна розділити на такі види:

- Перший вид: теоретичні моделі, засновані на гіпотезі відносин між

змінними якості;

- Другий вид: моделі "управління даними", засновані на статистичному аналізі;
- Третій вид: моделі комбіновані, де використовується інтуїтивний підхід до необхідної моделі, і для встановлення моделі констант використовується аналіз даних.

Одна з перших моделей для оцінки якості програмного забезпечення [22] мала має три базові властивості для оцінки якості:

- використання (метрики – коректність, надійність, ефективність, цілісність і практичність);

- модифікація (метрики – тестованість, гнучкість, а також супроводжуваність);

- переносимість (метрики – мобільність, багаторазове використання, сумісність).

Іншу модель сформулював Boehm B.W та інші [18]. Вона дозволяє більш точно визначати якість основних характеристик програмного забезпечення, заданих набором показників та метрик. Ця модель представляє характеристики програмного забезпечення в більшому масштабі, ніж модель МакКолла, оскільки в її основі лежить оцінка якості програмного забезпечення на всіх етапах його життєвого циклу.

Обидві розглянуті вище моделі відносяться до ієрархічних моделей якості, в яких структурування показників здійснюється спочатку на основі високорівневих, далі на основі проміжних і, наприкінець, індивідуальних характеристик.

Ця модель дозволяє оцінювати програмне забезпечення за кількома ключовими аспектами. Перш за все, є поняття практичності. Воно вказує на те, наскільки легко та ефективно можна використовувати програму для вирішення конкретних завдань. Це якраз те, що робить програмне забезпечення дійсно корисним у певних сценаріях. Другий аспект - супроводжуваність. Це визначає, наскільки легко можна змінювати програмне забезпечення і проводити його

повторне тестування. Гнучкість та можливість адаптації грають важливу роль у довгостроковому використанні програм. Не останню роль відіграє мобільність. Тут ми не маємо на увазі рухливість фізичних пристроїв, а отже, можливість використання програмного забезпечення на різних програмних та апаратних платформах. Це забезпечує універсальність та доступність.

Але, як у кожній системі, є й свої недоліки. Модель Боема має тенденцію до автоматичності, що, в свою чергу, може впливати на точність визначення характеристик. Така особливість важлива при об'єктивній оцінці програмного забезпечення.

Пізніше вчені R.B. Grady і D.L. Caswell, аналогічно до моделей Боема і МакКолла, запропонували використовувати модель FURPS [19]. Основною відмінністю цієї моделі є наявність двох шарів показників якості, де спершу задіяні основні якісні характеристики, а далі вони представляються пов'язаними атрибути. Пізніше була вдосконалена модель FURPS і отримала нову назву FURPS +. Ця модель операційно працює з основними категоріями показників якості програмного забезпечення. Функціональність включає основні функції та додаткові показники, такі як особливості та безпека в інноваційних програмних проектах. Практичність охоплює аспекти, такі як можливість обліку людського фактора, ергономічність та наявність повної користувальницької документації.

Надійність враховує частоту відмов у роботі програмного забезпечення, можливості для відновлення інформації та прогнозованість дій користувача. Продуктивність охоплює час відгуку, пропускну здатність обробки інформації, точність рішень та інші аспекти доступності програмного забезпечення.

Експлуатаційна придатність розглядає можливості тестування, розширення, адаптованість, сумісність з іншими системами і можливість зміни конфігурації. Оновлена модель FURPS + додає обмеження проекту з інформаційних ресурсів, вимоги до мов та засобів розробки, а також функціональні вимоги та вимоги до інтерфейсу.

Концептуальна основа моделі якості FURPS/FURPS+ полягає в декомпозиції характеристик програмного забезпечення на функціональні та

нефункціональні (ускладнюються якістю, надійністю, продуктивністю, підтримкою).

При використанні такого підходу до створення моделі можливо оцінювати якість програмного забезпечення, використовуючи ці категорії як вимоги, а також вони служать основними показниками при здійсненні оцінки. Популярність цієї моделі пояснюється наявністю у неї найбільш універсального переліку характеристик для оцінки якості програмного забезпечення.

Ще однією моделлю є модель Ghezzi C, яка використовує різноманітні підходи до визначення якості програмного забезпечення та процесу його експлуатації в рамках програмного проекту [20]. Згідно з цією моделлю, якість програмного забезпечення визначається через наступні показники:

- надійність;
- стійкість;
- цілісність;
- продуктивність;
- супроводжуваність;
- практичність;
- можливість багаторазового використання;
- можливість взаємодії;
- уфективність;
- швидкість реакції на дії користувача;
- верифікованість;
- мобільність;
- зрозумілість.

Модель, яку запропонував G.R. Dromey [21], ґрунтується на критеріях оцінки характеристик якості та їх підхарактеристиках. Основна мета цієї моделі полягає в оцінці якості програмного проекту як інформаційної системи, враховуючи умови, що можуть призвести до відмінностей у оцінках якості між різними проектами.

Цю модель оптимально використовувати для виявлення можливих помилок у програмному забезпеченні та ідентифікації його складових, в яких потенційно можуть виникнути помилки. З використанням відносин між характеристиками та підхарактеристиками якості ця модель надає можливість отримувати оцінки якості окремих характеристик та властивостей як окремого програмного забезпечення та його складових, так і всього інноваційного програмного проекту.

Концептуально іншою моделлю є модель метрик SATC (Software Assurance Technology Center / NASA). Основна відмінність цієї моделі від всіх розглянутих раніше полягає в тому, що оцінки якості визначаються для кожної окремої складової програмного проекту. В процесі, спочатку дається оцінка якості розробки документації та вимог специфікацій, програмного забезпечення та його складових, тестуванню його складових, а також виконанню операцій. Вподальшому, інтегральний показник якості формується на основі отриманих оцінок. Завдяки описаному підходу дана модель забезпечує не тільки оцінку ризиків інформаційного проекту, але й ефективність виконуваних операцій, і як наслідок, якість розробленого програмного проекту. Для цього в моделі SATC формується набір цілей, які пов'язані з програмним проектом і атрибутами операцій, що ним виконуються, відповідно до структури моделі якості ISO 9126 [22].

Проаналізувавши описані вище моделі, можна сформулювати узагальнений алгоритм функціонування таких моделей. Цей підхід дасть змогу оцінити якість фреймворку як програмного забезпечення для подальшого процесу розробки веб додатку та використати ці дані в формування системи оцінювання оптимальності вибору веб-фреймворку для реалізації проекту.

2.2 Визначення критеріїв оцінки веб-фреймворків

Під час проектування майбутнього веб-продукту важливо аналізувати вхідні дані проекту та визначати основні показники якості веб-фреймворку. Задача полягає у формуванні запиту для вибору оптимального веб-фреймворку та визначенні критеріїв якості для обраної моделі.

Методи визначення показників якості програмного забезпечення можна класифікувати за способами та джерелами отримання інформації. Способи включають вимірювальний, реєстраційний, органолептичний та розрахунковий підходи. Джерела інформації поділяються на традиційні, експертні та соціологічні.

У контексті розгляду трирівневої архітектури, наявної більшість розробників використовує багаторівневу модель визначення якості програмного забезпечення згідно стандарту ISO 9126. Ця модель базується на 6 основних характеристиках якості, кожна з яких має набір атрибутів і відповідні метрики.

Основна мета - визначення показників якості та їх подальше використання в контексті проектування веб-фреймворку для ефективного та якісного розробництва майбутніх веб-продуктів.

Вимірювальний метод базується на властивостях та характеристиках об’єкта з використанням інструментальних засобів. В випадку оцінювання веб-фреймворку, за допомогою цього методу визначається розмір файлів - число рядків вихідного коду програми і кількість рядків - коментарів, кількість операторів і операндів, кількість виконаних операторів, кількість гілок в програмі, число точок входу/виходу, час виконання гілки програми, час реакції, та ін.

Реєстраційний метод базується на інформації з випробувань або функціонування фреймворку, коли реєструються і підраховуються певні події, такі як: час роботи програми, кількість помилок і відмов, час передачі управління інших модулів, тощо.

Органолептичний метод ґрунтується на обробці інформації, яку одержали в результаті аналізу сприйняття органів почуттів людини (зору, слуху), і застосовується для визначення таких показників як ефективність, зручність застосування, зрозумілість тощо.

Розрахунковий метод, в свою чергу, ґрунтується на використанні теоретичних і емпіричних залежностей (до появи статистичних даних), статистичних даних, що накопичуються при ітераціях тестів, використанні та підтримці програмного забезпечення. За його допомогою визначаються тривалість і точність обчислень, час реакції, тощо.

Експертний метод визначення показників якості використовується у випадках, коли задача не може бути вирішена жодним іншим з описаних способів, або ті способи є значно більш трудомісткими або не ефективними. Рекомендується застосовувати експертний метод при визначенні показників щодо характеристик документації, таких як: структурованість, повнота, легкості освоєння, наявність прикладів. Для визначення значень показників якості експертним методом, залучається група експертів-фахівців, компетентних у вирішенні подібних завдань, на базі їх досвіду попередніх робіт та інтуїції.

Соціологічні методи базуються на даних отриманих зі спеціальних анкет-опитувань. В контексті даної роботи вони можуть розглядатися як отримання відгуків розробників зі спеціально розроблених форм.

Найвищим пріоритетом при виборі показників якості в більшості випадків є призначення та функціональна придатність відповідного програмного продукту до вирішення поставлених задач.

Процес вибору та встановлення метрик для опису характеристик якості програмних засобів можна концептуально розглядати як два послідовні етапи.

На першому етапі здійснюється вибір та обґрунтування набору вихідних даних, які відображають загальні особливості та етапи життєвого циклу проекту програмного продукту та його користувачів. Кожен з цих

аспектів впливає на конкретні характеристики якості комплексу програм.

Другий етап передбачає вибір та встановлення конкретних метрик та шкал вимірювання для характеристик якості проекту. Це необхідно для подальшої оцінки цих характеристик та порівняння їх з вимогами специфікацій під час випробувань чи сертифікації на різних етапах життєвого циклу програмного продукту.

Як було зауважено раніше, є велика кількість веб-фреймворків для створення різних частин веб застосунків (front-end, back-end). Для моделей оцінювання фреймворків різних типів характеристики якості можуть суттєво відрізнятися або взагалі не бути присутніми і це треба враховувати на етапі проектування моделі.

Існує багато характеристик за якими можна порівнювати саме веб-фреймворки клієнтської частини веб застосунку й актуальність їх вибору. Треба зауважити, що в кожному окремому проекті вони матимуть різні значення пріоритетності, в контексті майбутнього веб продукту але розглянемо найважливіші.

Функціональністю як характеристика наявності та якості вбудованих функцій й можливостей, таких як маршрутизація, автентифікація, валідація форм і т.д є важливою характеристикою будь якого фреймоврка, так як це одна з передумов його створення. Якість вбудованих функцій здебільшого вирішує на скільки багато коду для повторного використання напише розробник.

Якість офіційної документації та наявність прикладів для розробки важко переоцінити, адже це єдине джерело інформації щодо можливостей фреймворку. Також важливо як активно підтримується фреймворк розробниками, чи виходять оновлення і доповнення, виправляються помилки.

Продуктивність готових рішень фреймворку для оптимізації веб продукта напряму впливає на його швидкодію і як наслідок на враження від продукта кінцевого користувача. Зараз на сторінках веб застосунків

присутня велика кількість графічного контенту який суттєво сповільнює швидкість завантаження та рендерінгу, тому методи оптимізації є вкрай важливими.

Також важливою характеристикою є наявність заходів для запобігання атакам, можливість здійснення автентифікації та авторизації, захист від SQL-ін'єкцій та інших загроз від кіберзлочинців, які і часто атакують вразливі сайти та застосунки.

Співвідношення витрат і користі від застосування фреймворку лоя розробки це оцінка того, на скільки вартує використання фреймворку в порівнянні з його користю для проекту. Тобто, на етапі проектування треба оцінити потенційну користь від застосування того чи іншого веб-фреймворку і недоліки, якими є час на опанування фреймворком, збільшення розміру проекту, ціна фреймворку або додатків, тощо.

Розширюваність та здатність до масштабування проекту, можливість легкого розширення функціональності, додавання власних компонентів та модулів також є важливим аспектом для оцінювання. Яким би функціональним не був фреймворк, але написання кастомних функцій і модулів буде присутнє в середніх і великих проектах і простота їх інтеграції значно полегшить розробку.

Обов’язковою характеристикою є набір технологій та стандартів, які підтримує фреймворк (наприклад, REST API, WebSocket, робота з різними типами баз даних тощо). Це є основною характеристикою вибору для багатьох проектів де в технічному завданні прописані технології, які має використовувати майбутній веб продукт або вони зумовлені інфраструктурою хостингу.

Розмір і швидкодія фреймворку мають прямий вплив на час обробки, компіляції та запакування коду та майбутнього завантаження сторінок веб додатку. Швидкодія частково пересікається з продуктивністю але загалом розглядає характеристики відносно роботи коду фреймворку.

Варто також зазначити наявність та активність спільноти розробників. Різні тематичні форуми, блоги та чати де розробники можуть поділитися власним досвідом вирішення певних задач, виправлення помилок і створенням нових рішень є суттєвою допомогою для розробників при опанування нового фреймворку або вирішення нетривіальних задач [30].

На основі розглянутих характеристик якості можна сформувати модель якості та з урахуванням особливостей веб-фреймворків можна визначити нижні рівні системи показників якості.

1. Ефективність;
    1.1. Швидкодія:
        - час відгуку;
        - оптимізація завантаження сторінок;
    1.2. Масштабованість:
        - здатність до масштабування трафіку;
        - здатність до масштабувати вертикально (на одному сервері) та горизонтально (додавання серверів).
    1.3. Безпека:
        - наявність вбудованих засобів захисту від хакерських атак;
        - наявність оновлення та підтримки безпеки;
    1.4. Ресурси та продуктивність
        - ефективне використання ресурсів сервера.
        - швидкість виконання коду;
    1.5. Функціональність:
        - наявність вбудованих модулей та бібліотек;
        - підтримка специфічних функцій;
2. Зручність використання;
    2.1. Зрозумілість;
        - зрозумілість загальної структури фреймворку;
        - зрозумілість коду вбудованих функцій;

2.2. Зручність освоєння

- наявність та якість документації та загальної підтримки;
- наявність та якість документації та тематичної підтримки;
- наявність та кількість прикладів готових рішень;
- наявність та активність спільноти розробників;
- підтримка різних мов;

3. Надійність;

3.1. Завершеність;

- невідповідності і неточності в різних Web-браузерах;
- наявність орфографічних та пунктуаційних помилок;

3.2. Стійкість до відмов;

- наявність помилок при виконанні;
- наробіток на відмову;

3.3. Здатність до відновлення;

- відновлення працездатності після збою;

3.4. Узгодженість (відповідність стандартам) ;

- відповідність коду вимогам W3C;

4. Супроводжуваність (зручність супроводу);

4.1. Аналізованість;

- ведення лог-файлів;
- зручність читання лог-файлів;
- інформативність повідомлень, що виводяться при помилці;

4.2. Стабільність;

- програмна обробка помилок;
- збереження працездатності після помилки;

4.3. Тестованість;

- тестованість модулів фреймворку;
- тестованість програмного коду додатку;

Запропоновані показники характеризують фреймворк з точки зору розробника, як інструмент, та з точки зору користувача готового продукту, як характеристика якості веб додатку, розробленого за допомогою даного фреймворку. Таким чином, ці показники дають змогу отримати всебічну характеристику фреймворку.

2.3 Розробка методу оптимального вибору веб-фреймворку

Показники якості об'єднані в ієрархічну де кожен вищий рівень містить в якості складових показники нижчих рівнів з можливістю додавання додаткових.

Для отримання оцінки по показниках якості використовують фактори якості (1-й рівень): ефективність, зручність використання, надійність, супроводжуваність.

Кожному з факторів якості відповідає свій набір критеріїв якості (комплексні показники 2-го рівня). Критерії якості визначаються однією або кількома метриками (3-й рівень), які в свою чергу складаються з оціночних елементів (одиничні показники – 4-й рівень), що визначають властивість, яка задана в метриці. Кількість оціночних елементів, що включає в себе метрика не має обмежень і може коливатися в різні сторони. Розглянемо декілька способів побудови оцінки вибору фреймворка.

Метод сум, тобто сумування всіх фактичних значень показників:

$$Ip_i = \sum_{j=1}^{n} \frac{x_{ij}^{ф}}{x_{ij}^{б}}$$

де $x_{ij}^{ф}$ та $x_{ij}^{б}$ - відповідно фактичне і базисне значення j-го показника для і-го об’єкта.

Метод відстаней, базується на близькості об'єктів до еталону, котрим може служити і умовний об'єкт з максимальними значеннями. Розрахунок

комплексної оцінки проводиться по формулі евклідової відстані [23].

Метод геометричних середніх, який використовується для оцінки показників, значення y_{ij} яких знаходиться в межах від 0 до 1:

$$Ip_i = \left[\prod_{j=1}^{n} y_{ij} \right]^{\frac{1}{n}}$$

Але приведені вище методи не задовольняють наші умови оцінки через те, що в них вважається що всі показники є рівноправними і більше того, порівняння відбувається з певним еталонним об’єктом.

При розробки оцінки оптимальності веб-фреймворку, здебільшого достатньо важко визначитися з базовими показниками. Більш того, для різних веб-фреймворків (веб додатків) різні показники якості можуть набувати різної ваги. Наприклад, для створення інтернет магазину більш важливими є швидкодія та надійність, тим часом для невеликого веб додатку для передачі даних користувача (форм) більш важливою є зручність і швидкість завантаження сторінок.

Враховуючи велике розмаїття рівнів впливу кожної характеристики на кінцевий продукт, важливо встановити механізм порівняння цих характеристик між собою. З цією метою пропонується впровадити концепцію "критерії оптимальності". Для кожного критерію визначаються відносно обернені значення показників впливу на проект F_{Ei} (з ідеєю, що найвище значення буде призначено найменш впливовому критерію), які можна розрахувати за допомогою формули:

$$F_{E_i} = 100 - \frac{F_i}{F_{\max}} \times 100 \qquad 1)$$

де F_i – критерій впливу характеристики і-того фреймворку в вибірці;

$F_{\max}$ – максимальне значення показника для фреймворку [24].

Таким чином можна побудувати індивідуальну оцінку доцільності вибору кожного з веб-фреймворків з урахуванням особливостей проекту. Отже, наведемо алгоритм методики оцінювання веб-фреймворка:

1. Визначити загальні критерії впливу показника щодо кожного фреймворку у вибірці;

2. Визначити максимальне значення показника для кожного критерію $F_{\max}$;

3. Розрахувати значення критеріїв впливу характеристик для кожного з фреймворків;

4. На базі отриманих розрахунків зробити висновок щодо того наскільки веб-фреймворк з певними характеристиками підходить для вирішення поставленої задачі.

Побудована методика оцінювання веб-фреймворків є доволі простою, що дає змогу використовувати її на практиці розробниками веб продуктів. Основною задачею для успішного застосування є правильна оцінка критеріїв та наявність інформації щодо характеристик фреймворків які розглядаються. Вона може стати важливим етапом при проектування веб додатків і сайтів, що в свою чергу позитивно вплине як на якість кінцевого продукту так і на ресурси що компанія затратила на розробку.

Висновки до розділу 2

1. Проаналізовані методи оцінювання програмного забезпечення, що дало змогу підібрати необхідний для оцінювання якості веб-фреймворків метод.

2. Визначені основні критерії оцінки веб-фреймворків та методи визначення показників якості програмного забезпечення. На основі розглянутих характеристик сформована модель оцінки якості.

3. Розглянуто методи комплексної оцінки якості. Розроблена методика оцінювання вибору веб-фреймворку та обрана формула розрахунку критеріїв

впливу, що дало змогу побудувати індивідуальну оцінку доцільності вибору кожного з веб-фреймворків з урахуванням особливостей проекту.

3 РОЗРОБКА СИСТЕМИ ОПТИМАЛЬНОГО ВИБОРУ ВЕБ-ФРЕЙМВОРКІВ

3.1 Архітектура веб-фреймворків

Фреймворк можна описати як структуру, яка забезпечує основу для процесу розробки програми або веб додатку і надає набір інструментів і елементів, які допомагають у швидкому процесі розробки. Він діє як шаблон, який можна використовувати та навіть змінювати відповідно до вимог проекту. Фреймворк дозволяє структурувати код і дотримуватися певного стилю написання коду, що полегшить його читабельність й іншим розробникам буде легко зрозуміти ваш код, оскільки вони також знайомі з фреймворком. Загалом фреймворк надає такі переваги в веб розробці, як:

- читабельність і форматування коду (code style),
- чистота та простота написання коду,
- зменшення загальної кількості коду в проекті завдяки вбудованим функціям та методам,
- зменшення часу розробки і вартості проекту.

Більш того, базовий функціонал яким володіє фреймворк, загалом можна змінювати та розширювати для потреб поточного проекту, що робить використання фреймворків ще більш доцільним.

За допомогою фреймворків створюється як програмне забезпечення так і веб сайти та веб додатки в мережі Інтернет. Є два основних типи веб-фреймворків які розрізняють за принципом розділення відповідальності між рівнем представлення (front-end) та рівнем доступу до даних (back-end). Більшість з найпопулярніших front-end фреймворків мають структуру шарів (layer structure) або ярусів (Three-tier structure), головною метою якої є розділення логіки для нівелювання взаємного впливу в процесі роботи. Розлянемо детальніше основні типи архитектури front-end фреймворків

Модель–вид–контроллер (MVC або Модель–представлення–контролер, англ. Model-view-controller) - це архітектурний шаблон, що використовується при проектуванні та розробці програмного забезпечення (рис. 3.1).

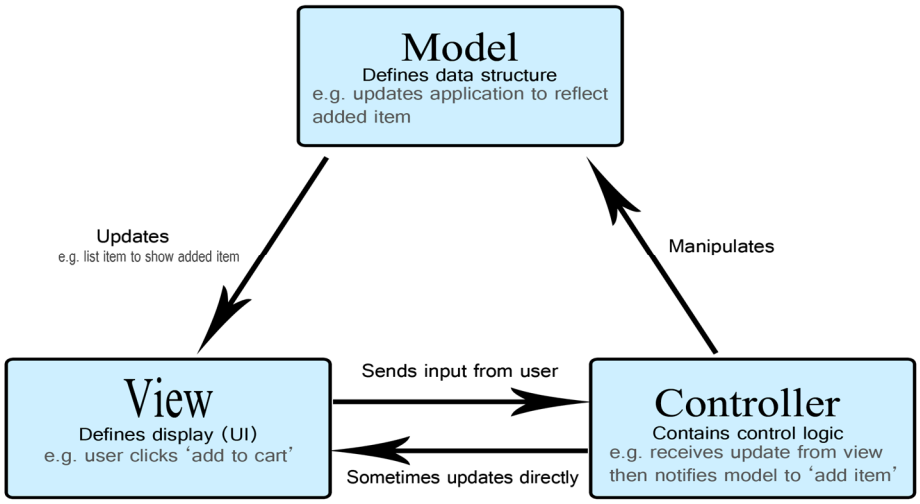


Рисунок 3.1 – Схема архітктури MVC

Модель передбачає поділ системи на три взаємопов'язані частини: модель даних, представлення (інтерфейс користувача) та контролер. Вона застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду), таким чином, щоб зміни в інтерфейсі користувача мінімально впливали на взаємодію з даними, і зміни в моделі даних могли здійснюватися без внесення змін в інтерфейс користувача. Модель представляє дані програми та бізнес-логіку, відповідає за керування даними, обробку введених даних користувача та відповідне оновлення представлення, інкапсулює стан і поведінку програми [10].

Вид (представлення) відповідає за представлення даних користувачеві та отримання введених даних. Він відображає інформацію та передає дії користувача контролеру. Також він гарантує, що інтерфейс користувача відображає стан програми.

Контролер керує даними від користувача, оновлює модель на основі цих даних та забезпечує відповідне оновлення представлення. Він діє як посередник

між моделлю та представленням. Основними функціями є обробка дій користувача, оновлення моделі і керування представленням.

Більшість фреймворків MVC дотримуються “push-based” архітектури, яку також називають "на основі дій". Ці фреймворки використовують дії, які виконують необхідну обробку, а потім "виштовхують" дані на рівень представлення (view) для відтворення результатів. Альтернативою цьому є “pull-based" архітектура, яку іноді також називають "компонентною". Ці фреймворки починаються з рівня представлення, який потім може "витягувати" результати з кількох контролерів за потреби. У цій архітектурі кілька контролерів можуть бути задіяні в одному представленні.

Це один з найперших шаблонів розроблених ще в 70 х роках 20 го сторіччя і як наслідок одна з найпоширеніших. Хоча MVC спочатку був розроблений для настільних комп’ютерів, він був широко прийнятий як дизайн для веб додатків на основних мовах програмування. Його використовують такі фреймворки як Spring MVC (Java), Ruby on Rails (Ruby), Django (Python) та інші.

Модель-Вид-ВидМодель (MVVM) представляє собою архітектурний шаблон у програмному забезпеченні, спрямований на ефективне відокремлення розробки графічного інтерфейсу користувача (виду або представлення) від бізнес-логіки чи серверної логіки (моделі). Це забезпечує незалежність від конкретної платформи моделі, будь то за допомогою мови розмітки чи коду (див. рис. 3.2).

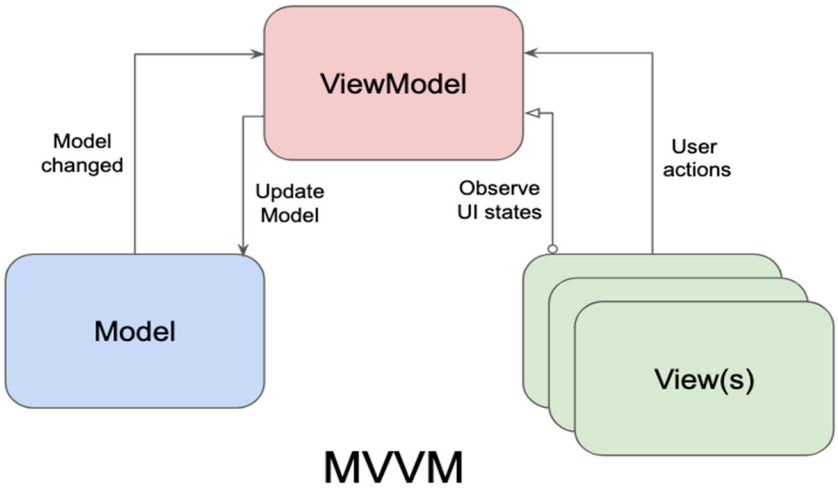


Рисунок 3.2 – Схема архітектури MVVM

Модуль ViewModel в MVVM є перетворювачем значень, тобто він відповідає за відображення (перетворення) об’єктів даних із моделі таким чином, щоб ними можна було легко керувати та представляти їх. Він є сполучною ланкою між вид і модель, реалізує та надає загальнодоступні властивості та команди, які вид використовує за допомогою зв’язування даних. Якщо відбуваються будь-які зміни стану, ViewModel сповіщає про події сповіщень ViewThrough.

Трирівнева архітектура (Three-tier organization) представляє собою ефективно організовану структуру прикладної програми, яка розподіляє програми на три логічні та фізичні рівні: рівень презентації або інтерфейс користувача; прикладний рівень, де відбувається обробка даних; і рівень даних, де зберігаються та управляються даними, пов'язаними з програмою (див. рис. 3.3).

Основна перевага трирівневої архітектури полягає в тому, що кожен рівень функціонує на власній інфраструктурі. Такий підхід дозволяє розробляти кожен рівень незалежно один від одного, використовуючи відокремлені команди розробників. Крім того, кожен рівень може бути оновлений або масштабований за потреби, не справляючи впливу на інші рівні системи.

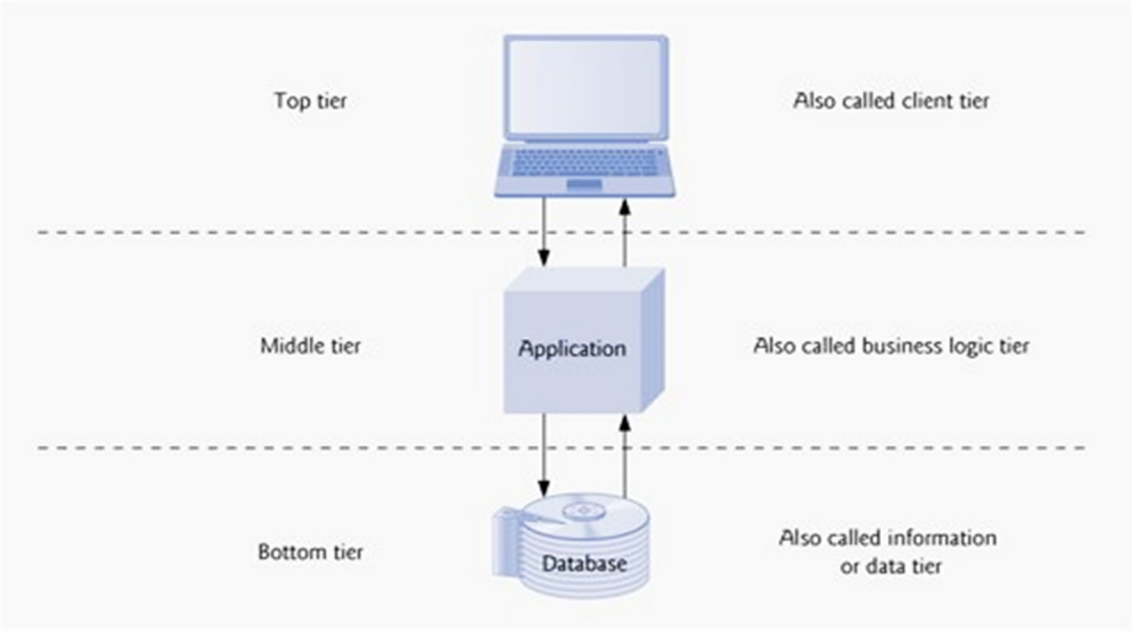


Рисунок 3.3 – Схема Three-tier архітктури

Протягом тривалого періоду трирівнева архітектура визнавалася основоположною для програм клієнт-сервер. Нині більшість програм, що використовують цей підхід, розглядаються як потенційні кандидати для модернізації, застосовуючи хмарні технології, такі як контейнери та мікросервіси, або для переходу в хмарне середовище. Розглянемо детальніше кожен з рівнів.

Рівень презентації – це інтерфейс користувача та рівень взаємодії програми, де кінцевий користувач спілкується з програмою. Його головне призначення – відображення інформації та збір інформації від користувача. Цей верхній рівень може опрацьовувати дані у веб-браузері, функціонувати як настільна програма або, наприклад, представляти графічний інтерфейс користувача (GUI). Веб-презентації зазвичай розробляються з використанням HTML, CSS і JavaScript. Настільні програми можуть використовувати різні мови програмування в залежності від платформи.

Рівень програми, відомий також як логічний чи середній рівень, виступає як центральний компонент програми. На даному рівні інформація, яка була зібрана на рівні презентації, піддається обробці - іноді в порівнянні з іншою інформацією на рівні даних - за допомогою бізнес-логіки та конкретного набору бізнес-правил. Також, рівень програми може додавати, видаляти або змінювати дані на рівні даних.

Розробка рівня програми часто виконується мовами програмування, такими як Python, Java, Perl, PHP, NodeJS або Ruby, і його взаємодія з рівнем даних здійснюється за допомогою викликів API.

Рівень даних, іноді відомий як рівень бази даних, рівень доступу до даних або бек-енд, є місцем, де зберігається та керується інформацією, обробленою на рівні програми, що зазвичай представлений системою управління базою даних.

У трирівневій архітектурі весь обмін інформацією пройде через рівень програми. Рівні презентації та даних не можуть взаємодіяти безпосередньо один з одним.

3.2 Проектування системи оптимального вибору веб-фреймворку

На основі аналізу вимог до системи було сформульовано наступні режими роботи веб застосунку:

- режим введення даних користувачем;
- режим обробки інформації (розрахунку);
- режим візуального зображення результату;
- режим вводу нових параметрів застосунку (для адміністраторів);

Відштовхуючись від вимог до застосунку і можливих режимів його роботи, в якості її основних функціональних блоків визначені наступні: блок управління даними, блок обробки інформації, блок виводу результатів, блок вводу нових параметрів.

Блок управління даними забезпечує можливість вводу даних користувачем, а саме вибору фреймворків та основних характеристик проекту, серед запропонованих.

Блок обробки інформації проводить розрахунки, згідно введених користувачем даних, і формує дані для відображення результатів розрахунків.

Блок виводу результатів здійснює графічне відображення отриманих розрахункових даних для спрощення сприйняття користувачем. Також можливий варіант цифрового представлення результатів але їх читабельність буде значно нижчою.

Блок вводу нових параметрів дозволяє додавати нові веб-фреймворки з їх характеристиками та основні функціональні характеристики проекту (в разі якщо специфіка веб додатку не покрита наявними). Це відбувається через спеціальну форму додавання даних, яка дозволяє розширювати існуючий потенціял застосунку.

Функціональна схема розробленого застосунку відображена на рис. 3.4. Загальний алгоритм функціонування застосунку представлений на рис. 3.5.

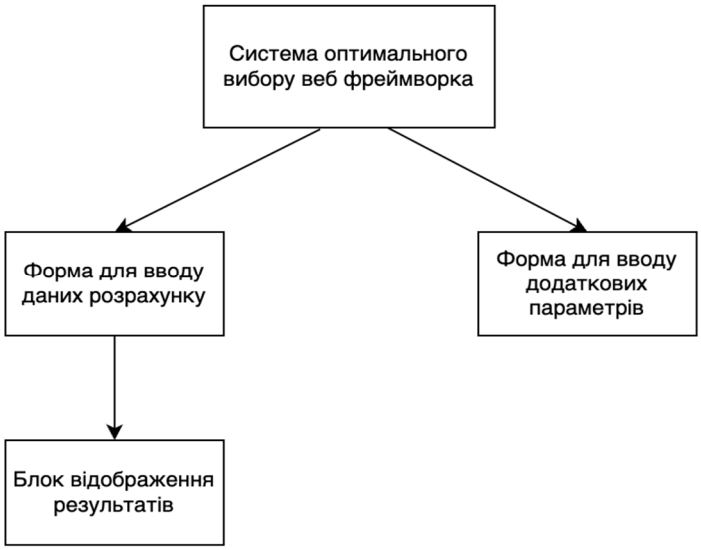


Рисунок 3.4 – Функціональна схема застосунку

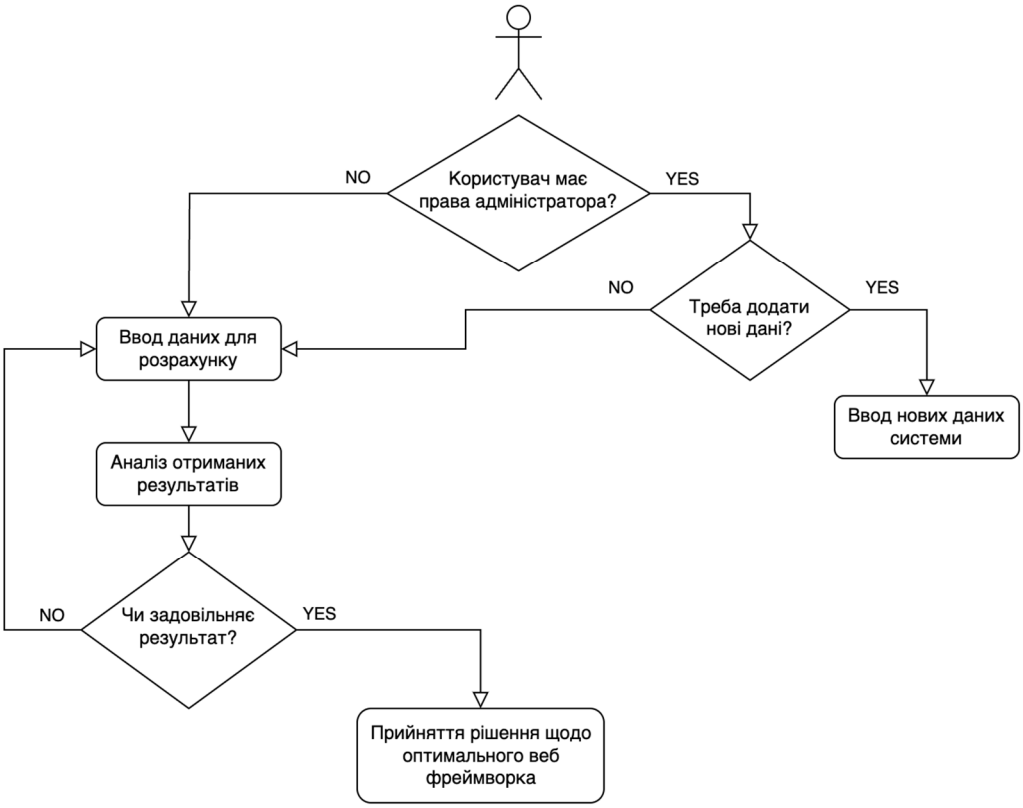


Рисунок 3.5 – Загальний алгоритм функціонування застосунку

3.3 Реалізація проекту оптимального вибору веб-фреймворку

На основі визначених вимог до системи оцінювання та її функціональної схеми, для реалізації програмної розробки додатку було вирішено наступне:

- блок управління даними функціонально реалізований на мові JavaScript з використанням фреймворку Vue JS (vue 3), код запускається в браузері клієнта, що гарантує швидкодію та простоту реалізації.

- блок обробки інформації реалізований у вигляді HTML форми з варінтами вибору доступних фреймворків та важлививх характеристик проекту і їх рівнем впливу на проект.

- блок виводу результатів реалізований на тій же сторінці форми, робить використання застосунку простим і наглядним. Графічна складова виводу результатів побудована за допомогою плагіна vue-chart-3 [31]. Він динамічно перебудовує графік на основі даних які отримані з блоку блок обробки інформації.

- блок вводу нових параметрів може бути реалізований у вигляді форми на окремій сторінці або спливаючому вікні. Після вводу, дані проходять валідацію на структурну і логічну відповідність і додаються до існуючих. Існуючі дані основних функціональних характеристик проекту та веб-фреймворків вирішено зберігати у вигляді JSON об’єкта в коді застосунку. Це спростить реалізацію, підвищить швидкодію і збереже ресурси, так як запити до бази даних займають додатковий час і вимагають побудови серверної частини застосунку.

Для розробки застосунку використовувалась середовище розробки (IDE) Visual Studio Code (версія 1.82) та браузер Google Chrome (версія 120.0.6099.71). Застосунок представляє собою веб сторінку на якій відбуваються всі дії (згідно концепції односторінкових додатків). Введення даних, розрахунок та оцінювання отриманих результатів відбуваються безпосередньо перед користувачем і дають змогу швидко і наглядно оцінювати отримані результати. Початковий вигляд сторінки зображений на рисунку 3.6.

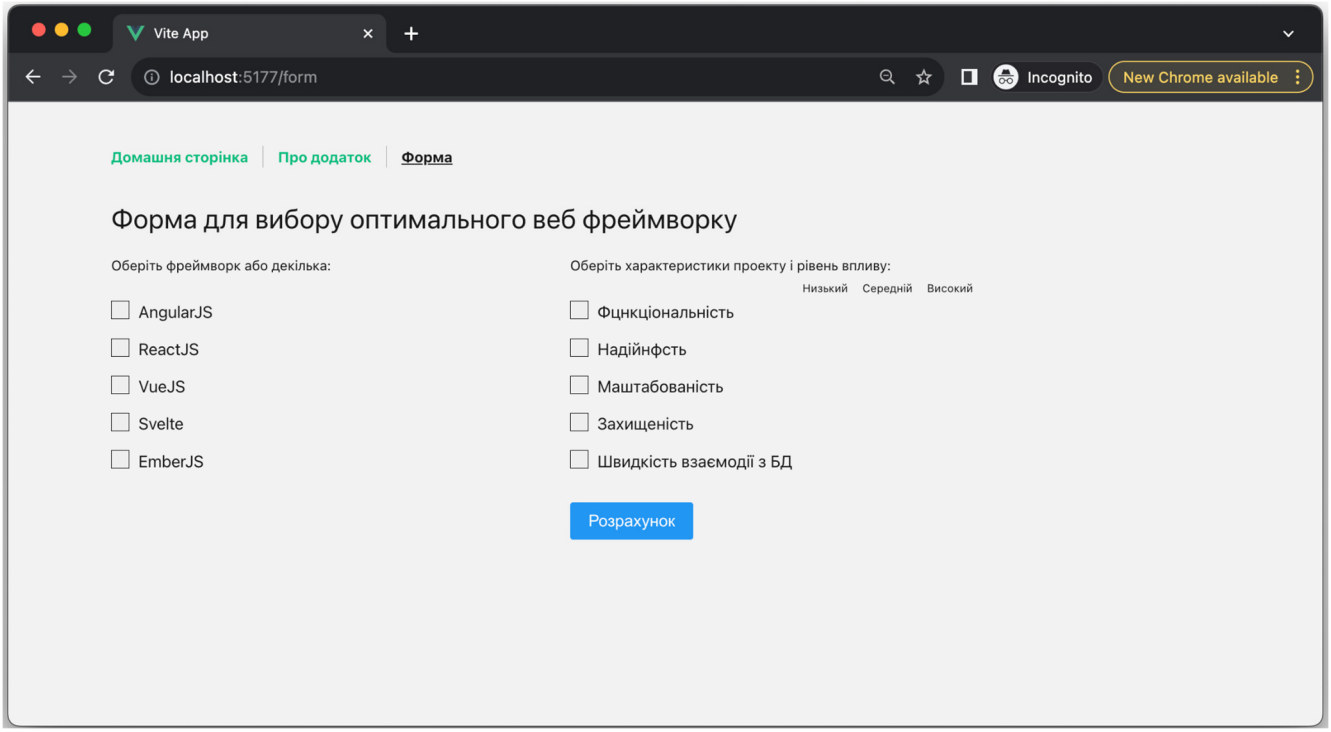


Рисунок 3.6 – Початковий вигляд сторінки

Розглянемо пограмний код форми. Реалізація цієї частини застосунку забзпечується спеціальним шаблоном <template>, який в свою чергу використовує побудовані компоненти FrameworkItem та ProjectItem, які можна використовувати багато разів де потрібно в коді застосунку. Це наглядно дмонструє “компонентний підхід” сучачних фреймворків, яким і є Vue JS. В даному випадку дані з шаблона передаються в компоненти які відтворюють динамічну структуру елементів сторінки:

```
<template>
  <div class="form-page-wrapper">
    <h1>Форма для вибору оптимального веб-фреймворку</h1>
    <form>
      <div class="form-data-row">
        <div class="form-framework-data">
          <p class="regular-text">Оберіть фреймворк або
декілька:</p>
          <p class="regular-text"> </p>
          <FrameworkItem v-for="framework in frameworks"
            :key = "framework.id"
            :frameworkName = "framework.title"
          />
        </div>
        <div class="form-project-data">
```

```
          <p class="regular-text">Оберіть характеристики
проекту і рівень впливу:</p>
          <p class="regular-text text-right"><span
class="small-text">Низький</span><span class="small-
text">Середній</span><span class="small-
text">Високий</span></p>
          <ProjectItem v-for="projectFeature in
projectFeatures"
            :key = "projectFeature.id"
            :projectFeatureName = "projectFeature.title"
            :projectFeatureTransaltion =
"projectFeature.translation"
          />
          <div class="button-data-row">
            <button @submit.prevent=""
type="submit">Розрахунок</button>
          </div>
        </div>
      </div>
    </form>
    <div class="form-result-row">
      <h2>Результати розрахунків оптимального фреймворку:</h2>
      <ChartView />
    </div>
  </div>
</template>
```

Вигляд сторінки з введеними параметрами для розрахунків зображений на рисунку 3.7.

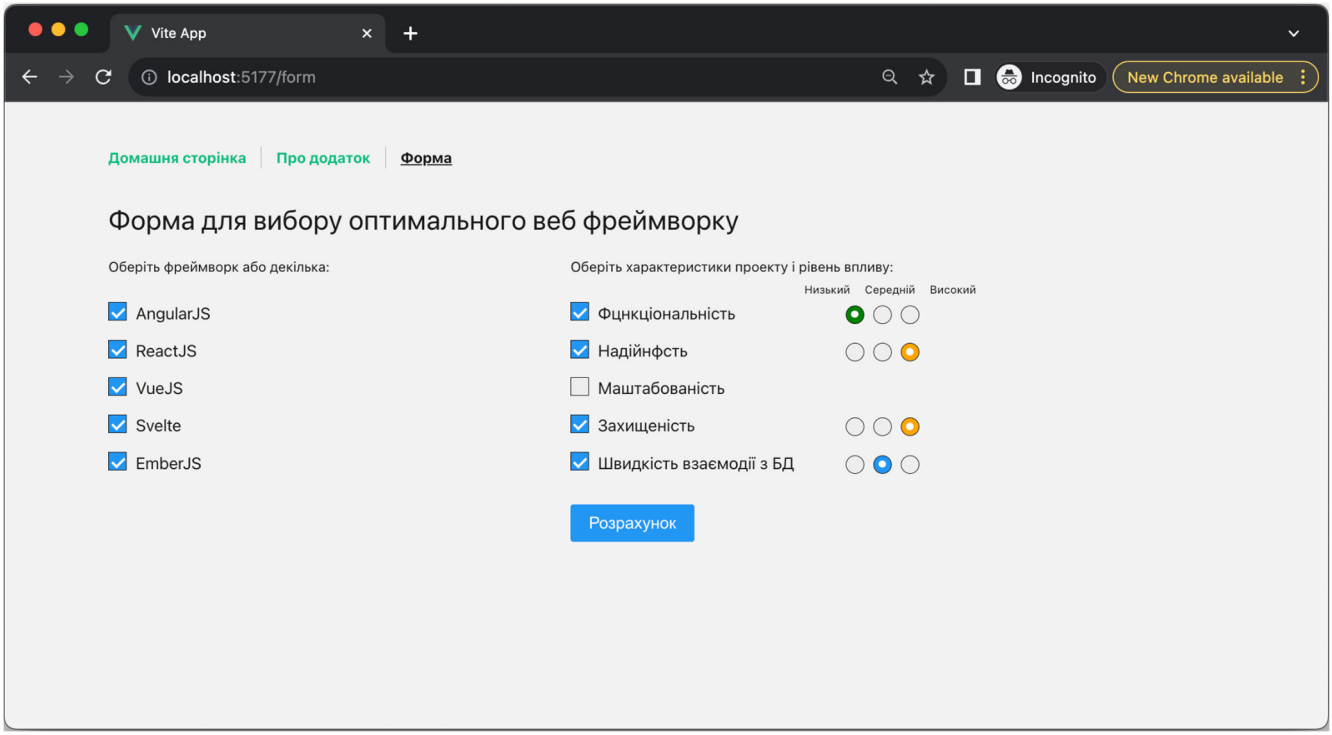


Рисунок 3.7 – Вигляд сторінки з введеними параметрами

Після натискання кнопки “Розрахунок” відбувається обробка форми. Дані, внесені в поля форми, обробляються відповідними функціями і вираховується значення оптимальності вибору того чи іншого з обраних фреймворків до проекту. Після вирахування результатів, значення для кожного з фреймворків передаються в блок графічного відображення і на їх основі будується діаграма.

Результати роботи веб додатку представлені на рисунку 3.8.

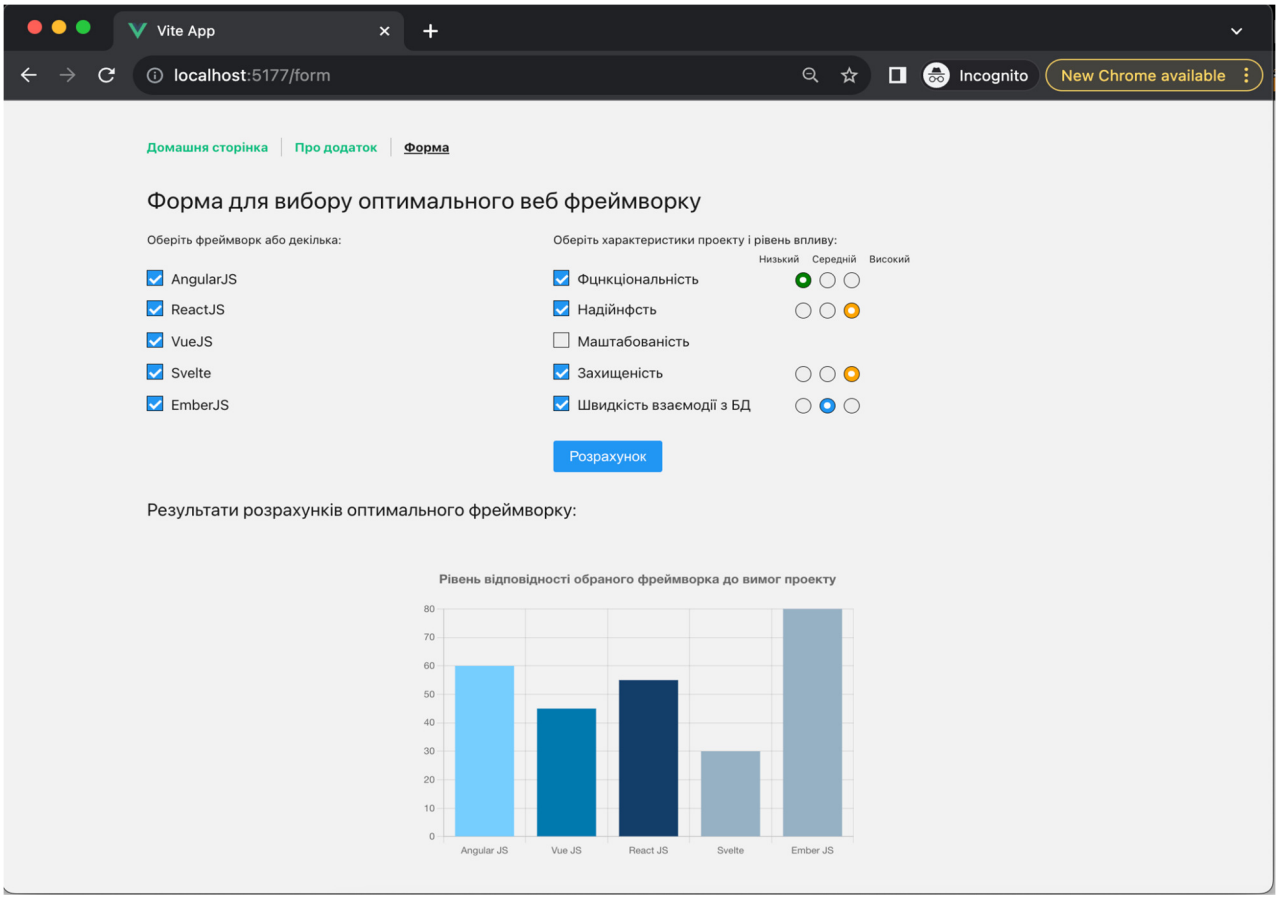


Рисунок 3.8 –Результати підбору отримального веб-фреймворку для проекту з заданими характеристиками

Загалом код обчислює введені в формі значення і повертає результат обчислень, на основі якого будується графічне представлення:

```
function getReversedMetrics(metricsArr){
  const reversedMetrics = [];
  metricsObj.forEach((elem) => {
    const reversedElem = parseFloat(100 - (elem/elMax)*100);
    reversedMetrics.push(reversedElem);
```

```
  });
  return reversedMetrics;
}
function generatePopulation(size) {
  const population = [];
  for (let i = 0; i < size; i++) {
    population.push(Math.random() * 100);
  }
  return population;
}

function calculateFitness(individual) {
  // Adjust this function based on your problem
  return individual;
}

function selection(population) {
  return population.sort((a, b) => calculateFitness(b) -
calculateFitness(a));
}

function crossover(parent1, parent2) {
  const crossoverPoint = Math.floor(Math.random() *
parent1.length);
  const child = [...parent1.slice(0, crossoverPoint),
...parent2.slice(crossoverPoint)];
  return child;
}

function mutation(individual, mutationRate) {
  return individual.map((gene) => (Math.random() <
mutationRate ? Math.random() * 100 : gene));
}

// Evolutionary Algorithm
function evolutionaryAlgorithm(populationSize, generations,
mutationRate) {
  let population = generatePopulation(populationSize);

  for (let generation = 0; generation < generations;
generation++) {
    population = selection(population);

    const newPopulation = [];

    for (let i = 0; i < populationSize; i += 2) {
      const parent1 = population[i];
      const parent2 = population[i + 1];
      const child = crossover(parent1, parent2);
      const mutatedChild = mutation(child, mutationRate);
      newPopulation.push(mutatedChild);
    }

    population = [...population, ...newPopulation];
```

```
  }
  const bestIndividual = selection(population)[0];
  return bestIndividual;
}
const  getReversedMetrics =
getReversedMetrics(metricsDataObj);
const bestElement = evolutionaryAlgorithm(getReversedMetrics,
0.01);
```

Розглянемо пограмний код частини шаблона яка відповідає за виведення графічних даних. Як видно з коду шаблона, за це відповідає компонент <ChartView /> який побудований за допомогою додаткового плагіна vue-chart-3. Цей плагін в своючергу є “обгорткою” для використання JavaScript бібліотеки ChartJS, створеної для полегшення візуалізації даних за допомогою графіків і діаграм:

```
<template>
    <div id="app" style="width: 600px">
        <BarChart v-bind="barChartProps" />
    </div>
</template>
```

Як видно код шаблону компонента складається лише з кількох рядків, але основна обробка і створення зображення відбувається за допомогою JavaScript коду. В ньому відбувається підключення додаткових модулів, реєстрація графіка і передача параметрів для створення відповідного відображення:

```
<script lang="ts">
    import { Chart, registerables } from 'chart.js';
    import { BarChart, useBarChart } from 'vue-chart-3';
    import { ref, computed, defineComponent } from 'vue';

    Chart.register(...registerables);

    export default defineComponent({
        name: 'App',
        components: {
            BarChart,
        },
        setup() {
            const data = ref(getCalculatedData(data));
            const options = ref({
                responsive: true,
                plugins: {
                    legend: {
                        display: false
```

```
                    },
                    title: {
                        display: true,
                        font: {
                            size: 16,
                        },
                        padding: 30,
                        text: 'Рівень відповідності обраного
фреймворка до вимог проекту',
                    },
                },
            });

            const chartData = computed(() => ({
                labels: ['Angular JS', 'Vue JS', 'React JS',
'Svelte', 'Ember JS'],
                datasets: [
                {
                    data: data.value,
                    backgroundColor: ['#77CEFF', '#0079AF',
'#123E6B', '#97B0C4', '#97B0C4'],
                },
            ],
        }));

        const { barChartProps, barChartRef } = useBarChart({
            chartData, options,
        });

        return { barChartProps, barChartRef };
        },
    });
</script>
```

Висновки до розділу 3

1. Наведені основні види архітектур веб-фреймворків, зокрема: MVC, MVVM, MVW, Three-tire, та їх особливості в контесті можливого застосування для розробки системи оцінювання.

2. Сформульовані основні режими роботи веб застосунку для оцінки веб-фреймворку, описана структура програми. Спроектована системи оцінювання вибору веб-фреймворку та описана функціональна схема роботи застосунку.

3. Визначені основні критерії для розробки застосунку системи

оцінювання, front-end веб-фреймворк, серодовище розробки, тощо. Розроблено код застосунку для прорахування оптимального фреймворку для заданого проєкту і описані основні особливості коду.

ВИСНОВКИ

В процесі виконання даного дослідження були отримані такі результати:

1. проведено огляд історії та еволюції веб-фреймворків;
2. проаналізовано найбільш популярні сучасні веб-фреймворки, що дало змогу виділити їх основні відмінності, переваги і недоліки;
3. поставлено задачу дослідження;
4. проаналізовані методи оцінювання програмного забезпечення, що дало змогу підібрати необхідний для оцінювання якості веб-фреймворків метод;
5. визначені основні критерії оцінки веб-фреймворків та методи визначення показників якості програмного забезпечення. На основі розглянутих характеристик сформована модель оцінки якості;
6. розглянуто методи комплексної оцінки якості. Розроблена методика оцінювання вибору веб-фреймворку та обрана формула розрахунку критеріїв впливу, що дало змогу побудувати індивідуальну оцінку доцільності вибору кожного з веб-фреймворків з урахуванням особливостей проекту.
7. наведені основні види архітектур веб-фреймворків, зокрема: MVC, MVVM, MVW, Three-tire, та їх особливості в контесті можливого застосування для розробки системи оцінювання;
8. сформульовані основні режими роботи веб застосунку для оцінки веб-фреймворку, описана структура програми. Спроектована системи оцінювання вибору веб-фреймворку та описана функціональна схема роботи застосунку;
9. визначені основні критерії для розробки застосунку системи оцінювання, front-end веб-фреймворк, серодовище розробки, тощо. Розроблено код застосунку для прорахування оптимального фреймворку для заданого проєкту і описані основні особливості коду.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Ajax technology", 2023, https://developer.mozilla.org/en-US/docs/Glossary/AJAX

2. "What Are Single Page Applications and Why Do People Like Them So Much?", 2022, https://www.bloomreach.com/en/blog/2018/what-is-a-single-page-application

3. "jQuery: The write less, do more, JavaScript library". The jQuery Project. Retrieved April 29, 2010.

4. "Переосмислення аналізу даних: від інформаційних панелей до AI Copilot", 2023, https://www.unite.ai/uk/Переосмислення-аналізу-даних-від-інформаційних-панелей-до-AI-Copilot/

5. "Usage of JavaScript libraries for websites". W3Techs. Archived from the original on November 15, 2019. Retrieved November 15, 2019. “jQuery (74.1%) is 3.7 times more popular than Bootstrap (19.9%).”

6. "Libscore". Archived from the original on February 19, 2017. Retrieved February 11, 2017. “Top scripts are 1. jQuery (692,981 sites); 2. jQuery UI (193,680 sites); 3. Facebook SDK (175,369 sites); 4. Twitter Bootstrap JS (158,288 sites); 5. Modernizr (155,503 sites).”

7. York, Richard (2009). Beginning JavaScript and CSS Development with jQuery. Wiley. p. 28. ISBN 978-0-470-22779-4.

8. Resig, John (October 31, 2007). "History of jQuery". Retrieved April 15, 2019.

9. "The jQuery Team". jquery.com. JS Foundation. Retrieved May 22, 2019. “Team: Timmy Willison (jQuery Core Lead), Richard Gibson (Sizzle Lead, jQuery Core).”

10. "Model-View-Controller", 2020, https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

11. "React - A JavaScript library for building user interfaces". reactjs.org. Archived from the original on April 8, 2018. Retrieved 7 April 2018.

12. "Chapter 1. What Is React? - What React Is and Why It Matters [Book]". www.oreilly.com. Archived from the original on May 6, 2023. Retrieved 2023-05-06.

13. Krill, Paul (May 15, 2014). "React: Making faster, smoother UIs for data-driven Web apps". InfoWorld. Retrieved 2021-02-23.

14. Hemel, Zef (June 3, 2013). "Facebook's React JavaScript User Interfaces Library Receives Mixed Reviews". infoq.com. Archived from the original on May 26, 2022. Retrieved 2022-01-11.

15. Dawson, Chris (July 25, 2014). "JavaScript's History and How it Led To ReactJS". The New Stack. Archived from the original on Aug 6, 2020. Retrieved 2020-07-19.

16. Stephen H. Kan, "Metrics and Models in Software Quality Engineering", 2014, 217p.

17. Chang C., Wu C., Lin H. "Integrating Fuzzy Theory and Hierarchy Concepts to Evaluate Software Quality. Software Quality Control", 2008, vol. 16, iss. 2, pp. 263–267.

18. Boehm B.W., Brown J.R., Kaspar H., Lipow M., MacLeod G.J., Merritt M.J. "Characteristics of Software Quality". TRW Series of Software Technology, Amsterdam, North Holland, 1978, 166 p.

19. Grady R.B., Caswell D.L. "Software Metrics: Establishing a Company-Wide Program". Prentice-Hall, 1987, 275 p.

20. Ghezzi C., Jazayeri M., Mandrioli D. "Fundamental of Software Engineering". Prentice-Hall, NJ, USA, 1991, 543 p.

21. Dromey G.R. "A Model for Software Product Quality. Transactions of Software Engineering", 1995, vol. 21, iss. 2.

22. McCall J.A., Richards P.K., Walters G.F. "Factors in Software Quality: Concept and Definitions of Software Quality. Final Technical Report". Vol. 1. National Technical Information Service, Springfield, 1977

23. "Метод відстаней", https://studfile.net/preview/5056566/page:10/

24. "Алгоритм підбору оптимального веб фреймворку за критеріями", 2023, http://www.konferenciaonline.org.ua/ua/article/id-1511/

25. "AngularJS". docs.angularjs.org. Retrieved 2021-05-14, https://angular.io/docs

26. Techson, Mark (February 2, 2021). "Finding a Path Forward with AngularJs". Medium. Retrieved April 9, 2021, https://blog.angular.io/finding-a-path-forward-with-angularjs-7e186fdd4429

27. Pieszak, Mark (January 7, 2020). "Angular Universal & Server-side rendering Deep-Dive". Medium. Retrieved April 9, 2021, https://medium.com/@MarkPieszak/angular-universal-server-side-rendering-deep-dive-dc442a6be7b7

28. "Introduction". Vuejs.org. Retrieved January 3, 2020, https://vuejs.org/guide/introduction.html

29. "What is Vue.js". W3Schools. Retrieved January 21, 2020, https://www.w3schools.com/whatis/whatis_vue.asp

30. “Js-framework-benchmark”, https://github.com/krausest/js-framework-benchmark

31. “Vue vs. React in 2023 - Comparison of Two Most Popular JS Frameworks”, https://www.monterail.com/blog/vue-vs-react, 2023

32. “Factors to consider when choosing a frontend framework for web application”, 2021, https://www.ssa.group/blog/choosing-a-frontend-framework/

33. “An Introduction to performance.now()”, 2022, https://blog.bitsrc.io/a-very-quick-introduction-to-performance-now-c95493e77d06

34. “Detailed Web Framework Comparison With Pros And Cons”, 2021, https://www.monocubed.com/blog/web-development-framework-comparison/

35. Nelson, Brett,” Getting to Know Vue.js: Learn to Build Single Page Applications in Vue from Scratch”,

36. Evan You, “https://blog.evanyou.me/2014/02/11/first-week-of-launching-an-oss-project/index.html”, 2014 - 2023.

37. “Template Syntax”, https://vuejs.org/guide/essentials/template-syntax.html, 2020

38. "What You Need To Know About Node.js", 2019, https://readwrite.com/what-you-need-to-know-about-nodejs/

39. Wrapper around a Chart.JS plugin for Vue, https://vue-chart-3.netlify.app/

40. Simple yet flexible JavaScript charting library for the modern web, https://www.chartjs.org/

41. Мовчан О.В. Метод оптимально вибору веб фреймворка. VIII Науково-практична конференція молодих вчених і студентів «Інтелектуальні комп’ютерні системи та мережі». 5 грудня 2023 р. Тернопіль. Україна. С.38 https://ki.wunu.edu.ua/conference/archive/2023_2.pdf

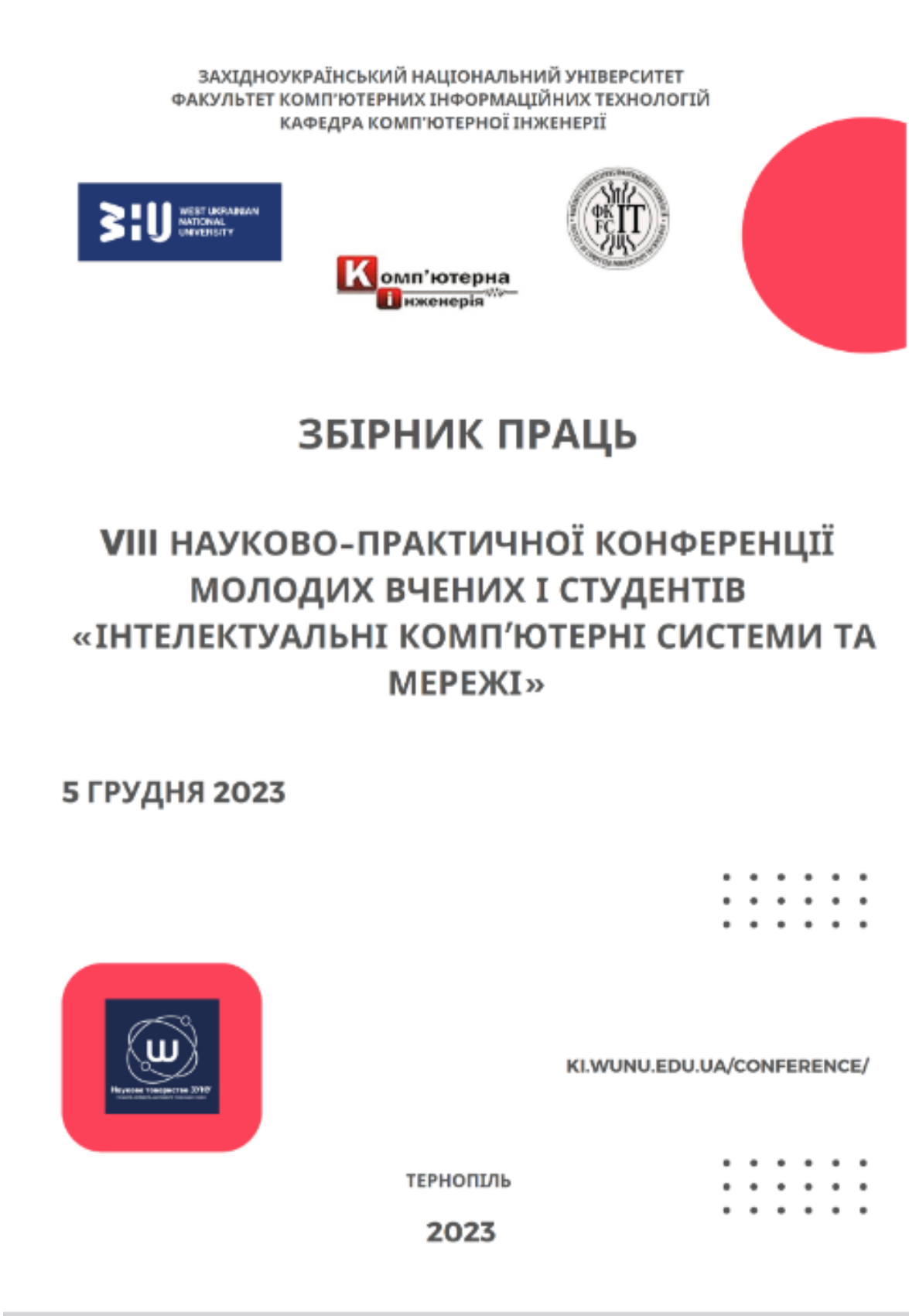
42. Мовчан О.В. Алгоритм підбору оптимального веб фреймворка за критеріями. Міжнародна наукова інтернет-конференція «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення», 7 грудня 2023 р., http://www.konferenciaonline.org.ua/ua/article/id-1511/

43. Загальні рекомендації з підготовки, оформлення, захисту та оцінювання випускних кваліфікаційних робіт здобувачів вищої освіти першого «бакалаврського» і другого «магістерського» рівнів / За ред. доц. М.І. Шинкарика. Тернопіль: ТНЕУ, 2018. 67 с.

44. Комар М.П., Саченко А.О., Васильків Н.М. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за другим (магістерським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2021. 32 с.

Додаток А

Копії публікацій



ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

ЗБІРНИК ПРАЦЬ

VIII НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ
МОЛОДИХ ВЧЕНИХ І СТУДЕНТІВ
«ІНТЕЛЕКТУАЛЬНІ КОМП'ЮТЕРНІ СИСТЕМИ ТА МЕРЕЖІ»

5 ГРУДНЯ 2023

KI.WUNU.EDU.UA/CONFERENCE/

ТЕРНОПІЛЬ

2023

Мовчан О.В.
студент 5 курсу ФКІТ ЗУНУ
Науковий керівник к.т.н. Биковий П.Є., кафедра ІОСУ ЗУНУ

МЕТОД ОПТИМАЛЬНО ВИБОРУ ВЕБ ФРЕЙМВОРКА

Вступ. Майже кожен інженер-розробник веб застосунків стикається з проблемою підбору оптимального фреймворка для реалізації проекту. Часто це доволі складна задача яка потребує ретельного аналізу багатьох даних. Вхідні дані із технічного завдання клієнта є лише їх малою частиною, адже архітектуру майбутнього застосунку, швидкість та простоту реалізації, його підтримку та потенційні можливості подальшого розширення та масштабування обумовлює саме вибір веб фреймворка. Отже розробка методики оптимального вибору веб фреймворка є актуальною задачею.

Постановка задачі. Об'єкт дослідження – інструменти сучасної розробки веб додатків. Предмет дослідження – найбільш поширені на момент публікації веб фреймворки. Мета дослідження – аналіз підходів і розробка методу оптимального вибору веб фреймворка для розробки застосунку.

Основний матеріал. Сучасна розробка веб застосунків вже не базується на написання всього коду "з нуля" і майже завжди пов'язана з використанням того чи іншого веб фреймворка, адже цей підхід має багато переваг і надає можливість сконцентркватися саме на процесі розробки. Веб-розробники можуть заощадити час і зусилля, використовуючи фреймворки для виконання численних повторюваних завдань, таких як налаштування браузера за замовчуванням, файлові структури та шаблони макета. Крім того, вони допомагають стандартизувати стиль і макет веб-додатків, полегшуючи підтримку узгодженості між різними веб-сторінками та функціями. З фреймворками складні навігаційні меню, зокрема, стають стандартизованими, оскільки їх можна використовувати послідовно на всьому веб-сайті, покращуючи взаємодію з користувачем [1-3].

Є багато характеристик за якими можна порівнювати веб фреймворки й актуальність їх вибору для проектую але розглянемо найважливіші:

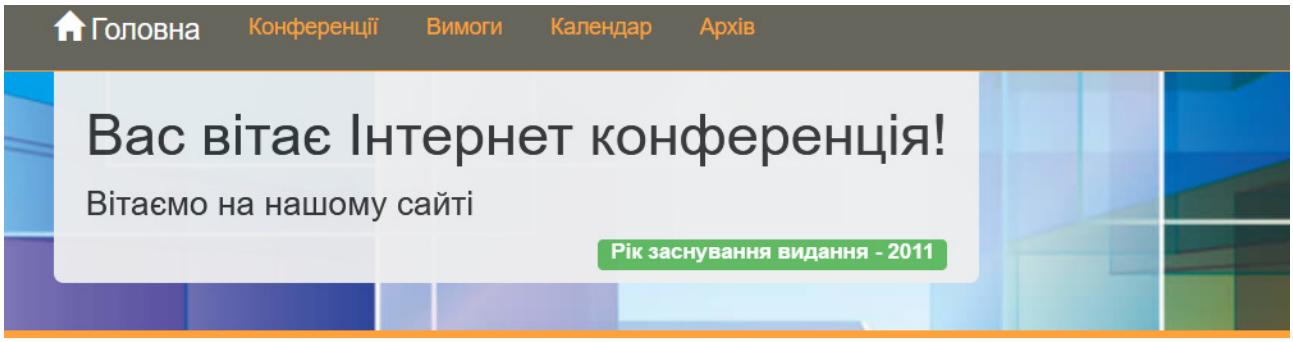
- Функціональність: наявність маршрутизації, автентифікації, валідації форм і т.д.
- Документація: якість офіційної документації та наявність прикладів для розробки. Чи активно підтримується фреймворк розробниками, чи виходять оновлення і доповнення.
- Продуктивність: швидкодія та оптимізація, можливості кешування.
- Безпека: наявність заходів для запобігання атакам, можливість здійснення автентифікації та авторизації, захист від SQL-ін'єкцій та інших загроз.
- Співвідношення витрат і користі: оцінка того, наскільки вартує використання фреймворка в порівнянні з його користю для проекту.
- Розширюваність та здатність до масштабування проекту: можливість легкого розширення функціональності, додавання власних компонентів та модулів.
- Підтримка технологій: які технології та стандарти підтримує фреймворк (наприклад, REST API, WebSocket, робота з різними типами баз даних тощо).
- Розмір і швидкодія: розмір фреймворка та вплив на час завантаження сторінок.

Детальний аналіз і порівняння зазначених характеристик дає змогу сформувати оціночну характеристику оптимальності обраного фреймворка для реалізації того чи іншого проекту.

Висновки. Практичне значення застосування метода оптимального підбору веб фреймворка полягає в тому, що користуючись ним, розробник зможе чітко визначити на скільки той чи інший фреймворк підходить для створення потрібного веб додатку.

Список літератури

1. Benefits Of Using Frameworks For Web Applications Development. 2023. URL: https://www.sencha.com/blog/benefits-of-using-frameworks-for-web-applications-development/
2. State of JavaScript. The annual developer survey of the JavaScript ecosystem. https://stateofjs.com/
3. John Dean. Web Programming With HTML5, CSS, And Javascript, Jones and Bartlett Publishers, Inc; Pap/Psc edition (9 Jan. 2018), 678 p.



АЛГОРИТМ ПІДБОРУ ОПТИМАЛЬНОГО ВЕБ ФРЕЙМВОРКА ЗА КРИТЕРІЯМИ

07.12.2023 18:50

[1. Інформаційні системи і технології]

Автор: **Мовчан Олексій Васильович, здобувач другого (магістерського) рівня вищої освіти, Західноукраїнський Національний Університет, м. Тернопіль; Биковий Павло Євгенович, кандидат технічних наук, доцент кафедри інформаційно-обчислювальних систем і управління, Західноукраїнський Національний Університет, м. Тернопіль**

ORCID: 0009-0004-8807-1408 Oleksii Movchan

Створення сучасних сайтів та веб додатків майже неможливо уявити без залучення таких потужних інструментів розробки як бібліотка або веб фреймворк. Так, прості статичні веб-сайти, цільові сторінки (landing pages) або веб-сайти малого бізнесу можуть бути створені вручну за допомогою тільки базових технологій як HTML, CSS і JavaScript, не покладаючись на бібліотеки, але додавання інтерактивних елементів на ці сторінки стане непростою та часозатратною задачею.

Використання фреймворків є поширеним явищем у сучасній веб-розробці через їхні переваги з точки зору ефективності, продуктивності та дотримання кращих практик програмування. За останнє десятиріччя вони сильно еволюціонували і тепер дають змогу використовувати готові структури, бібліотеки вбудованих функцій та різноманітних інтерфейсів, чим значно полегшують і допомагають уникнути “дитячих” помилок процесу розробки веб продуктів.

Однак, маючи таке різноманіття веб фреймворків, для інженера-розробника часто важко зробити вибір на користь одного з них. Це дійсно важливий і відповідальний етап проектування майбутнього продукту і неправильний вибір фреймворка може обернутись в довгий і складний процес розробки та підтримки проекту, що з часом може призвети навіть до потреби переробити або перенести продукт на інший фреймворк. Необхідність розуміння за якими критеріями обрати той чи інший веб фреймворк для майбутнього продукту підкреслює актуальність теми вибору оптимального веб фреймворка для реалізації проекту.

Алгоритм підбору оптимального веб фреймворка базується на аналізу його основних характеристик та вхідних даних майбутнього проекту; отже можна сформулювати наступні кроки в визначенні наскільки підходить даний фреймворк для розробки веб додатку або сайту:

- Визначення вимог до проекту: на основі інформації від зацікавлених сторін (замовника) сформувати цілі проекту, вимоги до масштабованості, функції та очікувану продуктивність [1].

- Визначення основних технологій: виходячи з потреб проекту, визначити основні технології (мову програмування, тип бази даних), які повинен підтримувати фреймворк [2].

- Оцінка спільноти та екосистеми: ознайомитись з проектами на GitHub, рейтингом фреймворку на GitHub, динамікою завантажень на npm, опитуваннями розробників на Stack Overflow і форумами спільноти розробників.

- Огляд офіційної документації та легкісті її засвоєння: переглянути офіційну документацію для кожного фреймворку, онлайн-підручники та навчальні платформи. Оцінити наскільки вона структурована та доступна для розуміння.

- Оцінка гнучкості і розширюваності: переглянути документацію фреймворку та обговорення спільноти на GitHub або форумах.

- Оцінка підтримки фреймворка розробниками: переглянути примітки до випуску, історію версій і обговорення спільноти на GitHub або офіційних форумах.

- Оглляд можливостей інтеграцій та сумісності: перевірити документацію щодо можливих інтеграцій, переглянути обговорення спільноти та приклади реальних проектів[2].

- Оцінка підтримки мобільної розробки: перевірити документацію щодо структури та наявних інструментів для мобільної розробки та тематичні дослідження мобільних проектів.

- Ознайомлення з тематичними дослідженнями та реальними прикладами проектів: відвідати офіційні веб-сайти фреймворків та форуми спільноти розробників щоб отримати дані практичних досліджень та приклади реальних проектів.

Виходячи з того що рівні впливу кожної з характеристик на кінцевий продукт можуть суттєво відрізнятися, необхідно сформувати механізм порівняння цих характеристик між собою. Для цього пропонується ввести поняття “критерія оптимальності”. Кожному з критеріїв присвоїти відосно обернені значення показників впливу на проект (тобто найбільше значення матиме найменш впливовий критерій), які можна розрахувати за формулою:

$$F_{E_i} = 100 - \frac{F_i}{F_{\max}} \times 100 \qquad (1)$$

Де F_i– критерій впливу характеристики і-того фреймворка в вибірці; F_{max} – максимальне значення показника для фреймворка.

Таким чином описана вище методика оптимального вибору веб фреймворка дає змогу оцінити наскільки оптимально той чи інший веб фреймворк, з певної кількості обраних, підходить для реалізації поставленої задачі.

Література:

1. Ghezzi C., Jazayeri M., Mandrioli D. Fundamental of Software Engineering. Prentice-Hall, NJ, USA, 1991, 543 p.

2. Sharma A., Kumar R., Grover P.S. Estimation of Quality for Software Components: An empirical approach. ACM SIGSOFT Software Engineering Notes, 2008, vol. 33, iss. 6.

3. Detailed Web Framework Comparison With Pros And Cons, 2023, https://www.monocubed.com/blog/web-development-framework-comparison/

Додаток Б

Код модулів методу

Код реалізації методу

```
function getReversedMetrics(metricsArr){
  const reversedMetrics = [];
  metricsObj.forEach((elem) => {
    const reversedElem = parseFloat(100 - (elem/elMax)*100);
    reversedMetrics.push(reversedElem);
  });
  return reversedMetrics;
}
function generatePopulation(size) {
  const population = [];
  for (let i = 0; i < size; i++) {
    population.push(Math.random() * 100);
  }
  return population;
}

function calculateFitness(individual) {
  // Adjust this function based on your problem
  return individual;
}

function selection(population) {
  return population.sort((a, b) => calculateFitness(b) -
calculateFitness(a));
}

function crossover(parent1, parent2) {
  const crossoverPoint = Math.floor(Math.random() *
parent1.length);
  const child = [...parent1.slice(0, crossoverPoint),
...parent2.slice(crossoverPoint)];
  return child;
}

function mutation(individual, mutationRate) {
  return individual.map((gene) => (Math.random() <
mutationRate ? Math.random() * 100 : gene));
}

// Evolutionary Algorithm
function evolutionaryAlgorithm(populationSize, generations,
mutationRate) {
  let population = generatePopulation(populationSize);

  for (let generation = 0; generation < generations;
```

```
generation++) {
    population = selection(population);

    const newPopulation = [];

    for (let i = 0; i < populationSize; i += 2) {
      const parent1 = population[i];
      const parent2 = population[i + 1];
      const child = crossover(parent1, parent2);
      const mutatedChild = mutation(child, mutationRate);
      newPopulation.push(mutatedChild);
    }

    population = [...population, ...newPopulation];
  }
  const bestIndividual = selection(population)[0];
  return bestIndividual;
}
const  getReversedMetrics =
getReversedMetrics(metricsDataObj);
const bestElement = evolutionaryAlgorithm(getReversedMetrics,
0.01);
```

Відтворення динамічної структури елементів сторінки

```
<template>
  <div class="form-page-wrapper">
    <h1>Форма для вибору оптимального веб фреймворку</h1>
    <form>
      <div class="form-data-row">
        <div class="form-framework-data">
          <p class="regular-text">Оберіть фреймворк або
декілька:</p>
          <p class="regular-text"> </p>
          <FrameworkItem v-for="framework in frameworks"
            :key = "framework.id"
            :frameworkName = "framework.title"
          />
        </div>
        <div class="form-project-data">
          <p class="regular-text">Оберіть характеристики
проекту і рівень впливу:</p>
          <p class="regular-text text-right"><span
class="small-text">Низький</span><span class="small-
text">Середній</span><span class="small-
text">Високий</span></p>
          <ProjectItem v-for="projectFeature in
projectFeatures"
```

```
            :key = "projectFeature.id"
            :projectFeatureName = "projectFeature.title"
            :projectFeatureTransaltion =
"projectFeature.translation"
          />
          <div class="button-data-row">
            <button @submit.prevent=""
type="submit">Розрахунок</button>
          </div>
        </div>
      </div>
    </form>
    <div class="form-result-row">
      <h2>Результати розрахунків оптимального фреймворку:</h2>
      <ChartView />
    </div>
  </div>
</template>
```

Структура компонентів шаблону сторінки форми

Компонент “фреймворк”

```
<template>
    <p>
        <label class="container"
:for="lowerName">{{frameworkName}}
            <input type="checkbox" :id="lowerName" >
            <span class="checkmark"></span>
        </label>
    </p>
</template>
```

Компонент “проект”

```
<template>
    <div class="project-value-wrapper">
        <label class="container title-container"
:for="lowerName" >{{projectFeatureTransaltion}}
            <input type="checkbox" :id="lowerName" >
            <span class="checkmark"></span>
        </label>
        <span class="radio-bttn-wrapper hidden">
            <label class="container">
                <input type="radio" class="project-value prio-
low" :name="radioName">
                <span class="radiobtn"></span>
```

```
            </label>
            <label class="container">
                <input type="radio" class="project-value prio-
low" :name="radioName">
                <span class="radiobtn"></span>
            </label>
            <label class="container">
                <input type="radio" class="project-value prio-
low" :name="radioName">
                <span class="radiobtn"></span>
            </label>
        </span>
    </div>
</template>
```

Компонент елементу “графічне відображення”

```
<template>
    <div id="app" style="width: 600px">
        <BarChart v-bind="barChartProps" />
    </div>
</template>

<script lang="ts">
    import { Chart, registerables } from 'chart.js';
    import { BarChart, useBarChart } from 'vue-chart-3';
    import { ref, computed, defineComponent } from 'vue';


    Chart.register(...registerables);

    export default defineComponent({
        name: 'App',
        components: {
            BarChart,
        },
        setup() {
            const data = ref([60, 45, 55, 30, 80]);

            const options = ref({
                responsive: true,
                plugins: {
                    legend: {
                        display: false
                    },
                    title: {
                        display: true,
                        font: {
                            size: 16,
                        },
                        padding: 30,
```

```
                        text: 'Рівень відповідності обраного
фреймворка до вимог проекту',
                    },
                },
            });

            const chartData = computed(() => ({
                labels: ['Angular JS', 'Vue JS', 'React JS',
'Svelte', 'Ember JS'],
                datasets: [
                {
                    data: data.value,
                    backgroundColor: ['#77CEFF', '#0079AF',
'#123E6B', '#97B0C4', '#97B0C4'],
                },
            ],
        }));

        const { barChartProps, barChartRef } = useBarChart({
            chartData, options,
        });

        return { barChartProps, barChartRef };
        },
    });
</script>
```