

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти (освітній ступінь))

на тему КОМП'ЮТЕРНА СИСТЕМА ДЛЯ ВЗАЄМОДІЇ З ПЛАТФОРМОЮ
АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ПРИСТРОЇВ ІНТЕРНЕТУ РЕЧЕЙ

Виконав: студент 2 курсу, групи КНТ-612м
спеціальності _____
123 Комп'ютерна інженерія

Освітня програма (спеціалізація)
Спеціалізовані комп'ютерні системи
(код і назва спеціальності)

ДОРОФЕЄВ С. О.

(прізвище та ініціали)

Керівник ІЛ'ЯШЕНКО М.Б.

(прізвище та ініціали)

Рецензент КАПЛІЄНКО Т.І.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій

Кафедра «Комп'ютерні системи та мережі»

Ступінь вищої освіти (освітній ступінь) магістерський

Спеціальність 123 Комп'ютерна інженерія
(код і назва)

Освітня програма (спеціалізація) Спеціалізовані комп'ютерні системи
(назва)

ЗАТВЕРДЖУЮ
Зав. кафедри Кудерметов Р.К.
“ 14 ” Листопада 2023 року

ЗАВДАННЯ НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

ДОРОФЕЄВ Станіслав Олександрович

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Комп'ютерна система для взаємодії з платформою автоматизованого тестування пристроїв інтернету речей

керівник проекту (роботи) ІЛЬЯШЕНКО Матвій Борисович, к. т. н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “24” жовтня 2023 року № 400

2. Строк подання студентом проекту (роботи) 01 грудня 2023 року

3. Вихідні дані до проекту (роботи) скбд, база даних postgresQL, vue, мова програмування, серверне обладнання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) _____

1) Аналіз практик тестування

2) Вибір frontend-фреймворків для дослідження;

3) Практична реалізація.

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

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	ІЛЬЯШЕНКО М.Б., к. т. н., доцент		
Нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання 01.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз технічного завдання	10.10.2023 р.	
2	Вибір ПЗ для виконання роботи	15.10.2023 р.	
3	Проектування бази даних	01.10.2023 р.	
4	Побудова логічної моделі бази даних	25.11.2023 р.	
5	Проектування програмних засобів	01.12.2023 р.	
6	Оформлення отриманих результатів у ПЗ	10.11.2023 р.	
7	Оформлення графічного матеріалу	15.11.2023 р.	
8	Захист роботи	16.12.2023 р.	

Студент  **Станіслав ДОРОФЕЄВ**
(підпис) (ініціали та прізвище)

Керівник проекту (роботи)  **Матвій ІЛЬЯШЕНКО**
(підпис) (ініціали та прізвище)

РЕФЕРАТ

ПЗ: 61 с., 31 рис., 15 джерел

ВЕБ-ДОДАТОК, ІНТЕРНЕТ РЕЧЕЙ, ПЛАТФОРМА, ПЛАТФОРМА ТЕСТУВАННЯ, ТЕСТУВАННЯ ПРИСТРОЇВ

Метою дослідження є розробка та реалізація підсистеми взаємодії з автоматизованою платформою тестування пристроїв взаємодії з автоматизованою платформою тестування пристроїв Інтернету речей.

Розглянуто наявні рішення в галузі інформаційних систем у форматі конструкторів; вивчено frontend-фреймворки систем у форматі конструкторів; вивчено frontend-фреймворки, які найбільше підходять під вимоги, розроблено і протестовано WEB-додаток підсистеми взаємодії у вигляді конструктора тестів і моніторингу стану платформи.

Для розробки використано Visual Studio Code для frontend-частини додатка, Visual Studio 2019 для backend-частини, бібліотека React з додатковими інструментами React, C#.

Роботу присвячено розв'язанню проблеми ручного тестування пристроїв Інтернету речей. Пропонується використовувати розроблену платформу автоматизованого тестування вбудованого програмного забезпечення. Для зручної взаємодії з платформою розроблено клієнтський веб-додаток, який містить конструктор тестів і моніторинг стану платформи, стану платформи. Веб-додаток конструктора тестів реалізовано для створення тестів для IoT-пристроїв. Наявність моніторингу дає змогу визначати працездатність платформи та відстежувати виконання процесу тестування пристроїв. Автоматизація дала змогу значно скоротити час випробувань порівняно з ручним тестуванням.

ABSTRACT

Explanatory note to the master's work: 61 p., 31 figures, 15 sources.

INTERNET OF THINGS, DEVICE TESTING, PLATFORM, TESTING PLATFORM, WEB APPLICATION.

The aim of the study is to develop and implement a subsystem for interaction with an automated device testing platform for interaction with an automated device testing platform of the Internet of Things.

Existing solutions in the field of information systems in the format of constructors are considered; frontend frameworks of systems in the format of constructors are studied; frontend frameworks that best fit the requirements are studied; a WEB application of the interaction subsystem in the form of a test designer and platform status monitoring is developed and tested.

For the development, Visual Studio Code was used for the frontend part of the application, Visual Studio 2019 for the backend part, the React library with additional React, C# tools.

The work is devoted to solving the problem of manual testing of IoT devices. It is proposed to use the developed platform for automated testing of embedded software. For convenient interaction with the platform, a client web application has been developed that contains a test designer and monitoring of the platform status and platform status. The test builder web application is implemented to create tests for IoT devices. Monitoring allows you to determine the performance of the platform and track the progress of the device testing process. Automation made it possible to significantly reduce test time compared to manual testing.

ЗМІСТ

Вступ	7
1 Аналіз практик тестування.....	15
1.1 Додаток Flipabit.....	15
1.2 Огляд додатку Appy Pie.....	16
1.3 Огляд додатка Appgy.io	18
2 Вибір технології розробки.....	20
2.1 Вибір frontend-фреймворків для дослідження.....	20
2.2 Огляд Angular	26
2.3 Огляд React	29
2.4 Огляд Vue.....	31
2.5 Продуктивність програмного забезпечення.....	34
2.6 Висновки	43
3 Практична реалізація	44
3.1 Основні показники	44
3.2 Спеціалізовані показники. Підключення бібліотек.....	49
3.3 Інвентаризація обладнання.....	50
3.4 Результат впровадження моніторингу	55
Висновки	59
Перелік джерел посилання	60

ВСТУП

Актуальність теми дослідження. У процесі розробки IoT-систем, в тому числі IoT-пристроїв, важливим етапом є тестування. Тестування пристроїв Інтернету речей (IoT) є важливим завданням ще на етапі розробки рішення IoT. Тестування дозволяє виявити відповідність між реальною та очікуваною поведінкою пристрою чи програми, провести валідацію, з метою випустити на ринок якісний продукт.

Розробка та впровадження підсистеми, що включає в себе конструктор тестів та моніторинг стану платформи, є багатограним процесом, спрямованим на підвищення ефективності та надійності автоматизованої платформи. Ця підсистема слугує важливим інтерфейсом між користувачами та автоматизованою платформою, надаючи інструменти для створення тестів та моніторингу робочого стану платформи в режимі реального часу.

На початковому етапі проводиться ретельний збір вимог, щоб визначити конкретні потреби та очікування від автоматизованої платформи. Це закладає основу для подальшої розробки конструктора тестів і компонентів моніторингу стану платформи.

Конструктор тестів задуманий як інструмент, орієнтований на користувача, з інтуїтивно зрозумілим інтерфейсом. Його дизайн зосереджений на простоті, що дозволяє користувачам легко створювати, редагувати та керувати тестовими кейсами. Функціональність для визначення параметрів тесту, вхідних даних та очікуваних результатів дозволяє користувачам формулювати складні сценарії тестування. Для досвідчених користувачів передбачені можливості написання сценаріїв, що дозволяють створювати власні тестові сценарії відповідно до спеціалізованих вимог до тестування. Функції контролю версій реалізовані для відстеження змін у тестових кейсах, забезпечуючи прозорість і відстежуваність протягом усього процесу тестування.

Водночас, компонент моніторингу стану платформи розроблений таким чином, щоб надавати інформацію про стан автоматизованої платформи в реальному часі. Це передбачає постійне відстеження стану обладнання, програмних компонентів і загальної продуктивності системи. Інтегрована система сповіщення для оперативного інформування відповідних зацікавлених сторін про будь-які порушення, виявлені в процесі моніторингу. Створено надійну систему реєстрації та звітності для ведення всебічного обліку робочого стану платформи з плином часу, що полегшує пост-аналіз та усунення несправностей.

Вирішальне значення для успіху цієї підсистеми має її безперешкодна інтеграція. Конструктор тестів і компоненти моніторингу стану платформи працюють злагоджено, дозволяючи користувачам співвідносити результати тестування з робочим станом платформи в реальному часі. Така синергія підвищує ефективність користувачів у виявленні та вирішенні проблем під час тестування.

Найкращі практики впровадження включають міркування щодо масштабованості, гарантуючи, що підсистема може безперешкодно обробляти зростаючий обсяг тестових кейсів. Вживаються заходи безпеки для захисту конфіденційних даних, що беруть участь у процесах тестування. Юзабіліті-тестування проводиться для того, щоб гарантувати, що як конструктор тестів, так і інтерфейси моніторингу стану платформи є зручними для користувача і легко навігаційними.

Ретельне тестування, включаючи модульне, інтеграційне та наскрізне тестування, підтверджує надійність і точність підсистеми. Після успішного тестування розгортання виконується методично, починаючи з контрольованого середовища, перш ніж перейти до виробництва.

Надаються вичерпні навчальні матеріали та документація, щоб надати користувачам знання, необхідні для ефективного використання конструктора тестів та інтерпретації даних моніторингу стану платформи.

Процес завершується створенням основи для постійного вдосконалення, що дозволяє здійснювати ітеративні оновлення та вдосконалення на основі відгуків користувачів, вимог, що змінюються, та змін в екосистемі автоматизованої

платформи. Такий адаптивний підхід забезпечує постійну актуальність та ефективність розробленої підсистеми.

Після розгортання підсистема стає невід'ємною частиною екосистеми автоматизованої платформи, значно підвищуючи ефективність та результативність процесів тестування.

Для ознайомлення користувачів з функціоналом конструктора тестів та інтерпретацією даних моніторингу стану платформи проводяться навчальні сесії. Вичерпна документація слугує довідковим посібником, гарантуючи, що користувачі мають необхідні ресурси для використання всіх можливостей підсистеми.

Взаємодіючи з розробником тестів, користувачі отримують інструмент, який не тільки спрощує процес створення тестових кейсів, але й дає їм можливість адаптувати та налаштовувати сценарії тестування відповідно до вимог платформи, що змінюються. Можливості написання сценаріїв орієнтовані на просунутих користувачів, дозволяючи їм створювати складні тестові сценарії для всебічного покриття тестування.

Водночас, компонент моніторингу стану платформи є безцінним у підтримці працездатності та стабільності автоматизованої платформи. Дані в режимі реального часу в поєднанні з механізмами оповіщення забезпечують оперативне інформування зацікавлених сторін про будь-які аномалії, що дає змогу швидко вжити заходів для виправлення ситуації. Система реєстрації та звітності допомагає в пост-аналізі, пропонуючи цінні дані для постійного вдосконалення та оптимізації.

Безшовна інтеграція між дизайнером тестів і моніторингом стану платформи дозволяє отримати цілісне уявлення про середовище тестування. Користувачі можуть співвідносити результати тестування з одночасним робочим станом платформи, що сприяє ефективному виявленню та вирішенню проблем.

Масштабованість залишається ключовою перевагою підсистеми, яка легко адаптується до зростаючої складності та обсягу тестових кейсів. Заходи безпеки

продовжують захищати конфіденційні дані, вселяючи впевненість у користувачів щодо цілісності процесів тестування.

Регулярні оновлення та покращення впроваджуються на основі відгуків користувачів та нових вимог. Ітеративний характер цих оновлень гарантує, що підсистема залишається узгодженою з мінливими потребами автоматизованої платформи та динамічним ландшафтом методологій тестування.

На закінчення, розробка і впровадження конструктора тестів і підсистеми моніторингу стану платформи є постійним прагненням до розширення можливостей тестування автоматизованої платформи. Цей адаптивний і орієнтований на користувача підхід позиціонує підсистему як ключовий елемент у забезпеченні надійності, масштабованості та ефективності процесів тестування, що сприяє загальному успіху автоматизованої платформи.

Оскільки підсистема стає невід'ємною частиною щоденних операцій, її вплив на автоматизовану платформу стає все більш очевидним. Користувачі отримують вигоду від спрощеного процесу тестування, що дозволяє швидше виявляти та вирішувати проблеми. Універсальність конструктора тестів дозволяє використовувати широкий спектр сценаріїв тестування, сприяючи гнучкості в адаптації до мінливих вимог проекту.

Можливості написання сценаріїв дозволяють досвідченим користувачам досліджувати складні сценарії тестування, додаючи рівень гнучкості до фреймворку тестування. Ця гнучкість стає особливо цінною, коли автоматизована платформа розвивається, впроваджуючи нові можливості, інтерфейси та функції.

Компонент моніторингу стану платформи з його аналітикою в режимі реального часу та системою сповіщень виявляється безцінним для підтримки загального стану автоматизованої платформи. Проблеми виявляються оперативно, мінімізуючи час простою та потенційні збої. Функції реєстрації та звітності сприяють створенню надійного циклу зворотного зв'язку, надаючи цінні дані для пост-аналізу, виявлення тенденцій та стратегічного планування.

Відгуки користувачів продовжують відігравати ключову роль у розвитку підсистеми. Регулярна взаємодія з користувачами через сеанси зворотного зв'язку,

опитування чи форуми гарантує, що підсистема тісно пов'язана з потребами та очікуваннями спільноти тестувальників, які постійно змінюються. Цей ітеративний цикл зворотного зв'язку сприяє постійному вдосконаленню та адаптації підсистеми.

Прихильність до безпеки та масштабованості залишається непохитною. У міру зростання автоматизованої платформи підсистема плавно масштабується, щоб задовольнити зростаючі вимоги до тестування. Суворі заходи безпеки постійно оновлюються, щоб випереджати потенційні вразливості, вселяючи впевненість користувачам і зацікавленим сторонам.

Документація та навчальні матеріали регулярно оновлюються, щоб відображати найновіші функції та вдосконалення. Нові користувачі, які потрапляють у середовище тестування, знаходять вичерпні ресурси, що полегшують процес входження в систему, дозволяючи їм швидко навчитися користуватися конструктором тестів і орієнтуватися в інтерфейсі моніторингу стану платформи.

Успіх підсистеми вимірюється не лише її технічними можливостями, але й позитивним впливом на загальну ефективність, надійність та адаптивність автоматизованої платформи. Симбіотичний зв'язок між розробником тестів і моніторингом стану платформи сприяє створенню екосистеми тестування, яка є гнучкою, швидко реагує і здатна відповідати динамічним викликам сучасної розробки програмного забезпечення.

По суті, постійний розвиток і вдосконалення підсистеми відображає прагнення до досконалості в практиці тестування. Це є свідченням спільних зусиль розробників, тестувальників і зацікавлених сторін, які працюють разом, щоб забезпечити подальший успіх автоматизованої платформи у створенні високоякісного програмного забезпечення.

Чим раніше відбувається виявлення помилок у програмному забезпеченні будь-якої системи, тим нижчі витрати на їх виправлення. Особливо це актуально для підприємств, які займаються масовим виробництвом електронних пристроїв, приладів, «розумних» лічильників. До таких пристроїв належать лічильники, які

вимірюють споживання води, тепла чи інших енергоресурсів. Ці прилади зазвичай розміщуються в квартирах, будинках, промислових приміщеннях з метою передачі показань у систему автоматичного збору та обліку даних. Важливо, щоб ці дані були точними, передавались своєчасно та коректно фіксувалися у базі даних.

На даний момент на ринку немає відповідного інструменту, що дозволяє проводити тестування вбудованого програмного забезпечення автоматично. Емуляція мікроконтролера для виконання програм не є ефективним рішенням, тому що є непередбачуваним впливом похибок емуляції на результат роботи програми [1]. Тому тестування таких пристроїв проводиться вручну.

Завдання тестування пристроїв інтернету речей на даний момент не вирішене, є актуальним та затребуваним для виробників пристроїв.

Використання створеного ядра тестування пристроїв Інтернету речей через консоль налагодження незручно, користувач при її використанні не має можливості автоматизувати свої дії. Крім того, відсутність візуального інтерфейсу значно ускладнить процес тестування для користувачів, незнайомих з програмними кодами. Сучасна система має бути максимально «дружньою» для користувача в подібних ситуаціях.

Аналіз актуальності зумовили вибір теми дослідження: "Розробка підсистеми взаємодії з платформою автоматизованого тестування пристроїв Інтернету речей".

Реалізація конструктора тестів у вигляді візуального інтерфейсу покращить досвід користувача і дозволить не тільки економити час користувача-тестувальника на створенні сценаріїв, але і в цілому спростить використання платформою. Моніторинг дозволить відстежувати в реальному часі хід тестування пристроїв, статус платформи та обмін повідомленнями платформи з Інтернет-пристроями.

Метою дослідження є розробка та реалізація підсистеми, що складається з конструктора тестів та моніторингу стану платформи, для взаємодії з автоматизованою платформою.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- аналіз існуючих рішень у галузі інформаційних систем у форматі конструкторів;
- вибір оптимальних засобів розробки з урахуванням виявлених критеріїв;
- розробка, тестування WEB-додатку по взаємодії з платформою автоматизованого тестування пристроїв IoT.

Об'єктом дослідження є клас підсистем взаємодії з платформ тестування.

Предметом дослідження є підсистема з конструктора тестів моніторинг стану в платформі тестування пристроїв Інтернету речей.

Методи дослідження включають:

- аналіз, порівняння, систематизація та узагальнення даних про існуючі та розроблені способи автоматизації тестування та створення конструкторів;
- апробація сучасних WEB-технологій під час побудови веб-додатку;
- тестування роботи інтерфейсу на ПК;
- аналіз технологій та підходів, що дозволяють масштабувати та оптимізувати WEB-додаток;

Теоретичною основою дослідження стали:

- дослідження та описи рішень щодо організації процесу тестування та створення візуальних конструкторів;
- сучасні підходи та технології у розробці веб-додатків;
- документація до різних використовуваних у сучасному WEB-програмування фреймворків та бібліотек.

Теоретична та практична значимість. Робота відноситься до галузі досліджень застосування інформаційних та WEB-технологій для створення інтерфейсу до ядра тестування пристроїв IoT.

Виявлено, обґрунтовано та описано переваги інформаційних технологій як інструменту тестування. Показано, що за допомогою цього інструменту стає можливим підвищити ефективність та зручність процесу тестування нових пристроїв.

Проведено аналіз традиційного підходу до процесу тестування, а також кількох візуальних конструкторів, створено схему роботи WEB-додатка в комплексі платформи тестування пристроїв Інтернету речей, створено WEB-додаток з підсистемою взаємодії з платформою.

1 АНАЛІЗ ПРАКТИК ТЕСТУВАННЯ

1.1 Додаток Flipabit

Основа Flipabit — візуальний редактор для Windows або MacOS, за допомогою якого можна візуально (і з невеликим додаванням простого коду) конструювати програми. Його аудиторія - дизайнери, просунуті користувачі, малий бізнес, які можуть швидко створювати та публікувати нативні програми для будь-яких платформ.

Для початку Flipabit надає кілька шаблонів, за допомогою яких можна зробити готову програму. Окремо є для iOS та Android Master-Detail додаток, додаток із закладками, WebView додаток, додаток із картами тощо. На рисунку 1.1 представлений стартовий екран архітектора.

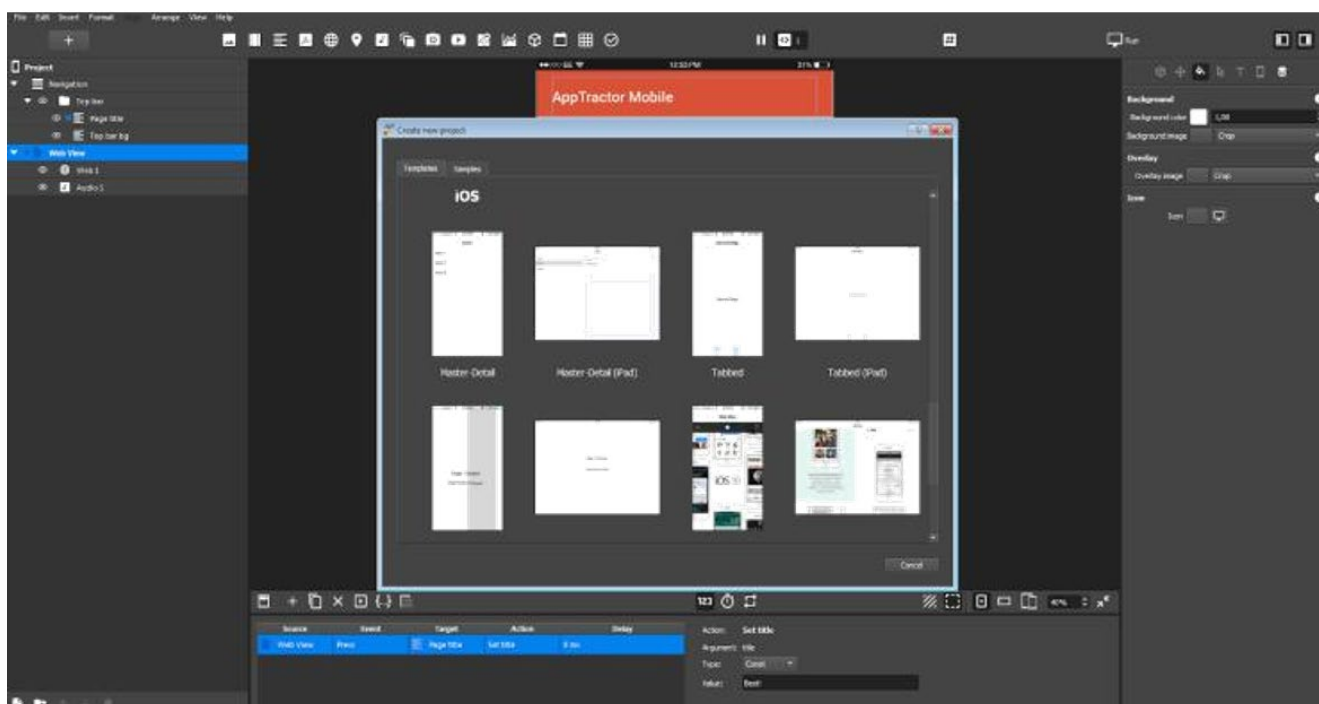


Рисунок 1.1 – Доступні шаблони програми

Після вибору шаблону потрапляємо в редактор WYSIWYG середовище, в якому існує можливість використовувати віджети і наочно конструювати свою програму. Віджети є практично для всього, що може знадобитися у роботі -

зображення, відео, карти, веб-сторінки, графіки та чарти, таблиці, календарі, база даних.

Праворуч можна побачити структуру створеної програми. Зліва – налаштування для елементів (джерело даних, розташування на екрані, кольори та відступи, шрифти тощо). У кілька кліків можна вибрати необхідні дані та вид для елемента (для видів існують свої шаблони, так що можна подивитися та змінити їх відповідно до потреб – наприклад, список може бути таблицею, картками, горизонтальним або вертикальним тощо). У конструкторі є можливість за допомогою JavaScript QML змінювати поведінку програми. Джерела подій, самі події (натискання, жести тощо), цілі для змін та дії для них можна змінювати так само.

В основі конструктора додатків лежить технологія Qt QML(У редакторі є можливість писати код на QML). QML (Qt Meta Language or Qt Modeling Language) — декларативна мова програмування, заснована на JavaScript, призначена для дизайну додатків, що роблять основний упор на інтерфейс користувача. Є частиною Qt Quick, середовища розробки інтерфейсу користувача, що поширюється разом з Qt. В основному використовується для створення програм, орієнтованих на мобільні пристрої з сенсорним керуванням.

З Flipabit складно реалізувати складну програму, але цей конструктор дозволяє швидко створити MVP (Minimal Value Product) і далі досить легко підтримувати невеликі програми.

1.2 Огляд додатку Appy Pie

Appy Pie – система створення мобільних додатків. У ній для створення програми достатньо перетягувати елементи за допомогою Drag-and-drop без завантаження додаткових плагінів. У конструкторі можна створити

мультиплатформну програму на базі HTML5, яка буде працювати з Android, iOS, і доступна для публікації в AppStore, Google Play.

Конструктор орієнтований створення аудіо- і фото-приложений, оскільки підтримує інтеграцію з такими аудіо-сервісами, як Soundcloud, Beatport і Shoutcast, і сервісами обміну фото: Instagram, Flickr, Picasa. Також при створенні існує можливість налаштувати інтеграцію із соціальними мережами Facebook, Twitter та Youtube.

Appy Pie - це інноваційний онлайн-сервіс, який дає змогу створювати власні мобільні додатки без необхідності в програмуванні. За його допомогою користувачі можуть легко та інтуїтивно розробляти додатки для різних цілей, чи то бізнес, освіта, чи то маркетинг.

Основними рисами Appy Pie є простота використання і багатофункціональність. Платформа надає зручний конструктор, де елементи додатка можуть бути легко переміщені та налаштовані, навіть для тих, хто не має досвіду в програмуванні.

Appy Pie надає різноманітні шаблони, які користувачі можуть вибирати для створення додатків. Це прискорює процес розробки і робить його доступним для широкого кола користувачів.

Сервіс також підтримує створення застосунків для різних платформ, включно з Android та iOS. Це означає, що користувачі можуть охопити більшу аудиторію, публікуючи свої додатки в Google Play і App Store.

Однією з переваг Appy Pie є можливість інтеграції з різними сторонніми сервісами та функціями. Це забезпечує додаткову гнучкість і розширені можливості для створюваних додатків.

Крім цього, Appy Pie надає інструменти для управління даними і базами даних усередині застосунків, що може бути важливим для складніших проєктів.

Залежно від обраної версії і плану, користувачі можуть також скористатися маркетинговими інструментами, що допомагають просувати і популяризувати свої додатки.

Загальна ідея Appy Pie полягає в тому, щоб надати інструмент для створення мобільних застосунків, який доступний для широкого кола користувачів, незалежно від їхнього досвіду в галузі програмування.

Під час створення програми можна використовувати існуючі шаблони, а також код програми можна змінювати вручну. На рисунку 1.2 представлений стартовий екран конструктора.

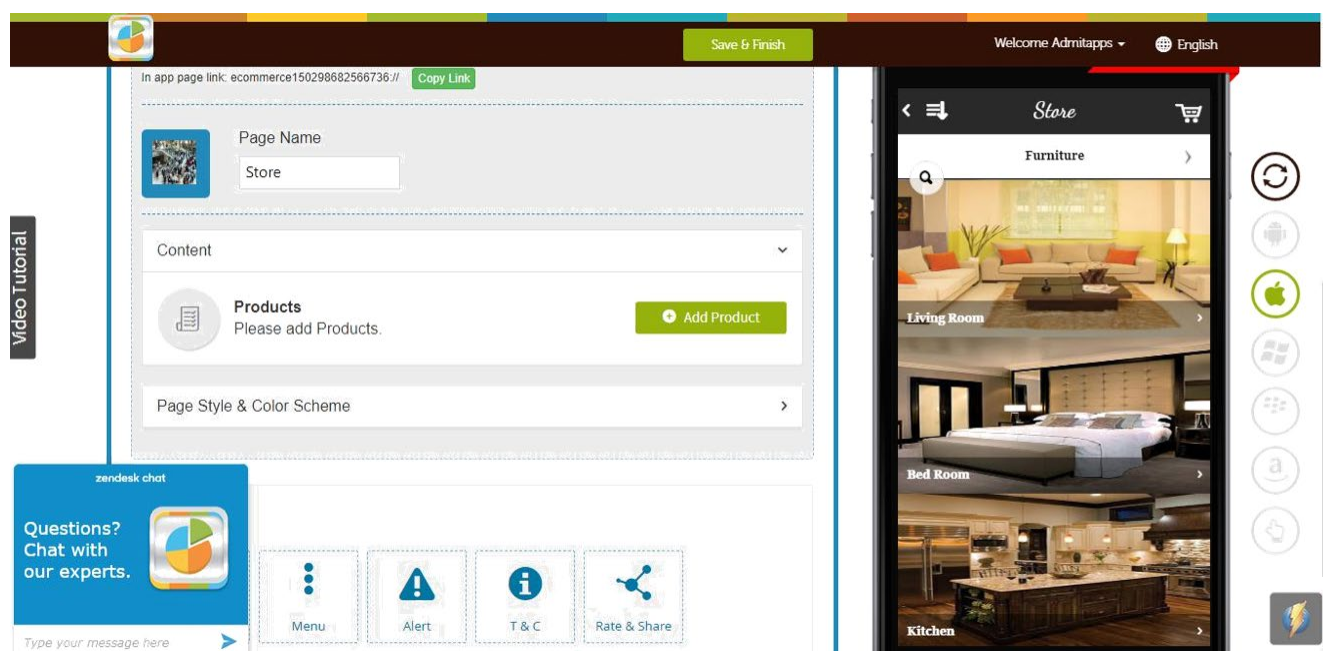


Рисунок 1.2 – Приклад створення простої програми в Appy Pie

1.3 Огляд додатка Appy Pie

Конструктор мобільних програм, здатний створювати нативні програми (Android, iOS), адаптивні веб-додатки та гібридні програми (для всіх пристроїв). Даний конструктор аналогічно додатку Appy Pie є Drag-n-drop візуальний редактор. Його перевагою є те, що протягом усього процесу розробки є можливість не писати програмний код, що особливо зручно для користувачів, які не мають навичок

програміста. Конструктор надає на вибір різні компоненти, шаблони додатків, якими можна розмістити інтерфейс користувача.

Для більш детального опрацювання програми є безліч додаткових функцій, починаючи від внутрішніх сервісів, таких як хмарна база даних, push-сповіщення, REST API-інтерфейси, кілька плагінів для мультимедіа, розсилки в соціальні мережі, і закінчуючи підтримкою останніх версій iOS та таких веб-технологій, як jQuery, Angular, Bootstrap. На рисунку 1.3 представлений стартовий екран конструктора.

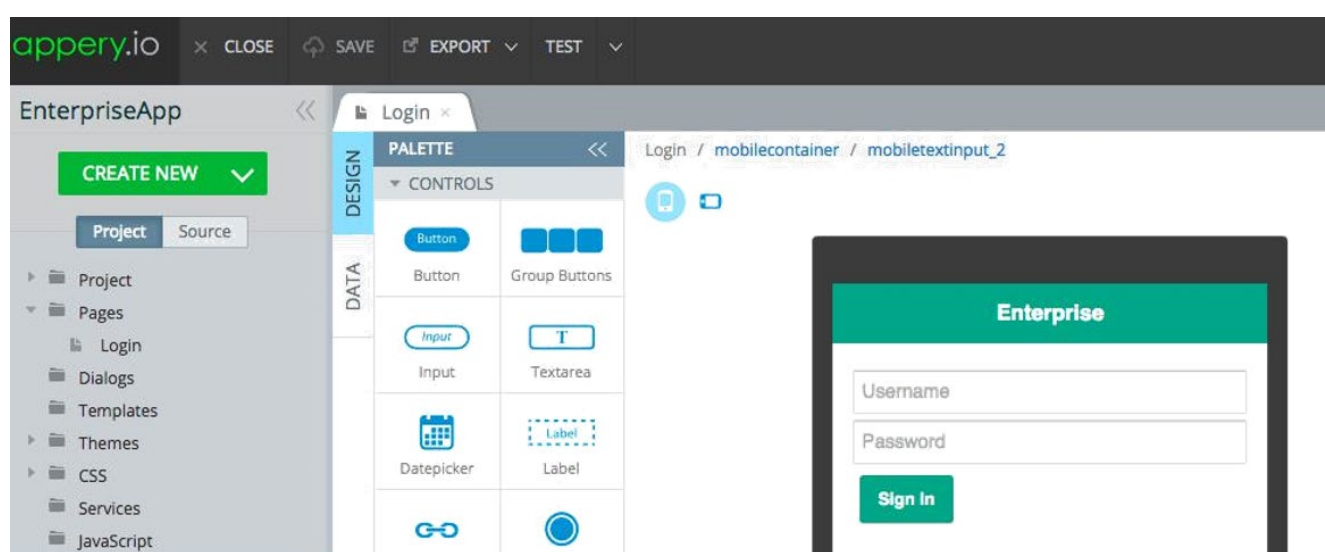


Рисунок 1.3 – Приклад створення простої програми в Appery.io

Конструктори роблять процес створення програми відносно простим. Використання кодів допускається у більшості продуктів.

Всі розглянуті програми-конструктори пропонують простий Drag-and-Drop інтерфейс, що дозволяє користувачам без навичок програмування, без проблем створювати нескладні програми. Замість кодування більшості конструкторів додатків використовується підхід WYSIWYG. Це означає, що кожна зміна, будь то за допомогою додавання зображення або завантаження контенту, демонструється в режимі реального часу у відповідному розділі, прев'ю програми, що додатково спрощує взаємодію з додатком. Подібні підходи будуть використовуватися під час створення програми.

У двох з представлених програм використовуються технології, засновані на HTML5, в одному - стек Qt з можливістю використання QML. Для конструктора тестів також використовуватиметься HTML5 у зв'язку з його мультиплатформенністю.

2 ВИБІР ТЕХНОЛОГІЇ РОЗРОБКИ

2.1 Вибір frontend-фреймворків для дослідження

Для дослідження було обрано найбільш популярні фреймворки: Angular, Vue та React. Фреймворк сама по собі бібліотека React не є, проте бібліотека, як правило, використовується в поєднанні з такими інструментами, як React Redux, React Router. Об'єднання React, React Redux та React Router у цій роботі визначимо як фреймворк React.

Кожна з цих платформ заснована на компонентах і сприяє швидкому створенню функцій інтерфейсу користувача. Їх можна використовувати як взаємозамінні, ну майже для створення інтерфейсних програм. Однак вони не на 100% однакові.

Перш ніж використовувати фреймворк або бібліотеку з відкритим кодом, важливо перевірити ліцензію. Усі три фреймворки використовують ліцензію MIT, ця ліцензія не накладає обмеження на використання навіть у пропрієтарному програмному забезпеченні.

Насамперед було проаналізовано тематичні опитування, серед яких Stackoverflow, The State of JavaScript. Щорічно вони надають найбільш точну інформацію щодо популярності та стану розвитку різних середовищ розробки. Крім того, скористалися інструментом для порівняння кількості завантажень NPMtrends.

На рисунку 2.1 можна побачити найпопулярніші фреймворки згідно з опитуваннями Stackoverflow [2]. Незважаючи на стрімкий розвиток Vue, цей

фреймворк посідає лише 7 місце в рейтингу (як серед усіх респондентів, так і серед професіоналів). React та Angular поділили 2-ге та 3 місце. Примітно, що згідно із загальним результатом 2-е місце посідає React, але виключно професійні розробники вивели на 2-е місце саме Angular.

Web Frameworks

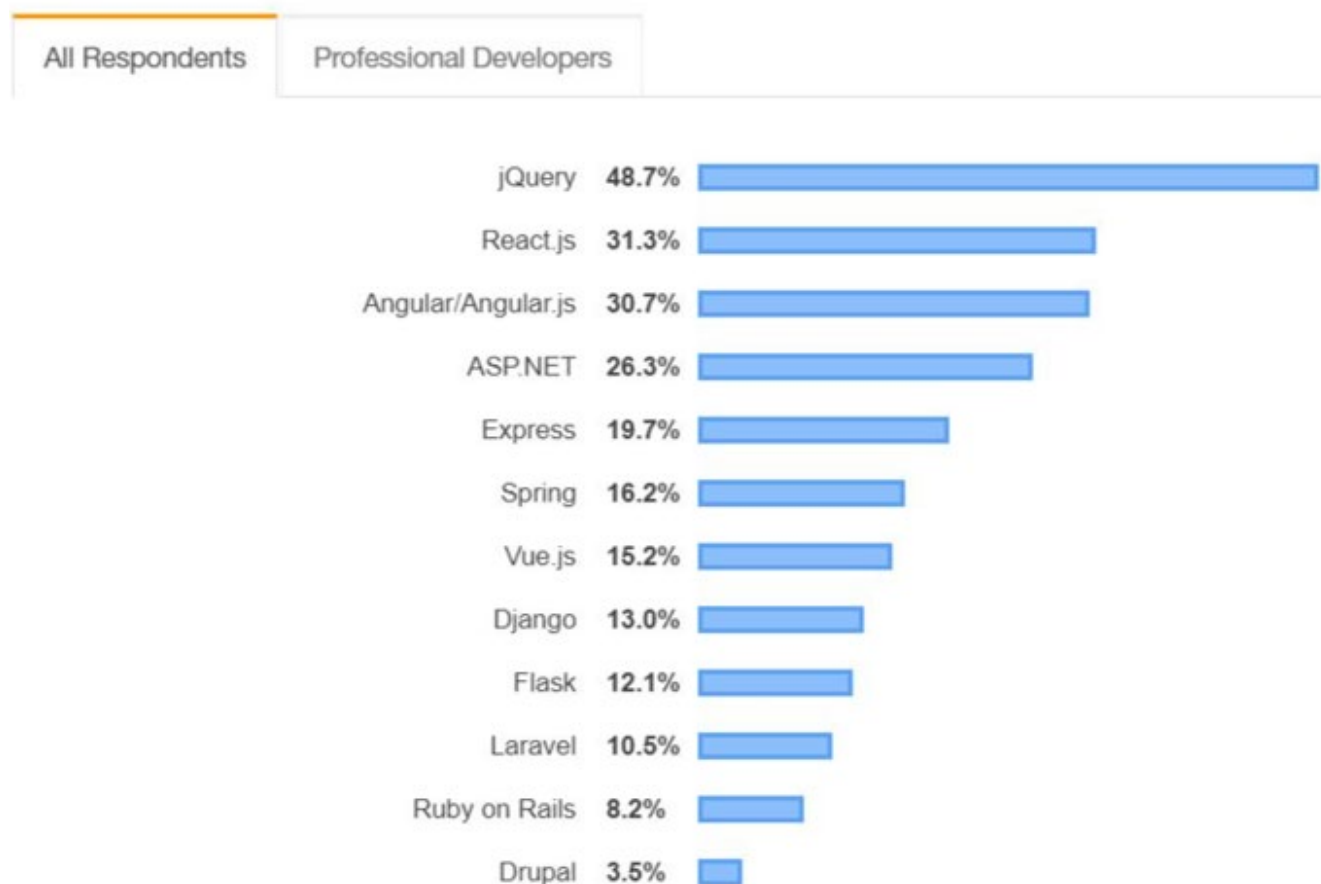


Рисунок 2.1 – Найпопулярніші веб-фреймворки

Водночас, розробникам найбільше подобається працювати саме з React та Vue, коли Angular виявився практично наприкінці списку. Серед тих, хто раніше не розробляв програми за допомогою цих фреймворків, найбільш бажаним для вивчення виявився React. За ним слідує Vue, а третє місце посідає Angular [3]. Рейтинг найбільш відданих фреймворків зображений рисунку 2.2.

Most Loved, Dreaded, and Wanted Web Frameworks

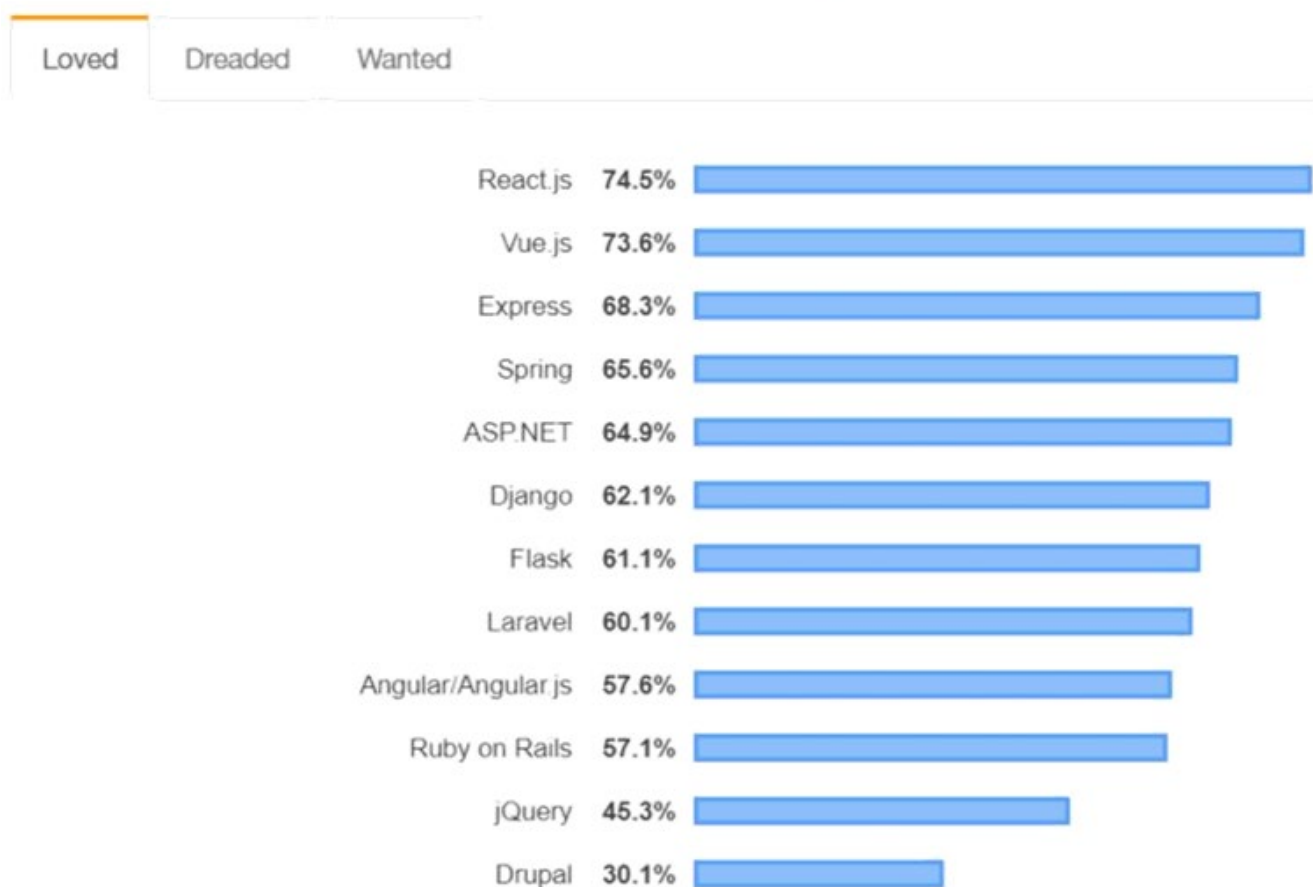


Рисунок 2.2 – Фреймворки, що найбільш віддають перевагу

Один із важливих показників розвитку технологічного стеку чи середовища розробки – кількість завантажень пакетів. NPM trends показує найточнішу статистику для кожного інструмента, включаючи не лише кількість завантажень, а й дані Github[4]. Графік кількості завантажень представлений рисунку 2.3.

Як можна побачити, React у цьому плані значно випереджає своїх конкурентів. Angular втратив значну кількість завантажень у порівнянні з 2021 роком.

Downloads in past 1 Year ▾

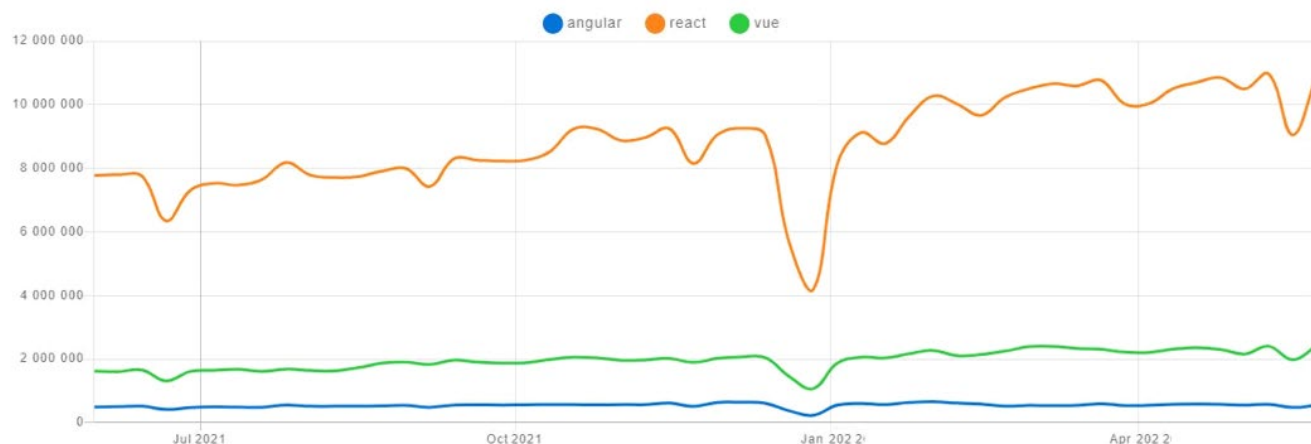


Рисунок 2.3 – Кількість скачувань Angular, React та Vue

Починаючи з середини 2022 року він так і не перевищив позначку в 1 мільйон. Vue поступово набирає обертів. Так, на початку 2022 року у нього було не більше 500 тисяч завантажень, тоді як сьогодні показник стабільно вищий за 1 мільйон. Рейтинги фреймворків на Github[4] представлені рисунку 2.4.

Stats

	stars 🌟	issues ⚠️	updated 🔄	created 🗓️	size 📦
angular	59 607	471	Jun 1, 2023	Jan 6, 2010	minzipped size 62.3 KB
react	169 209	721	Jun 2, 2023	May 24, 2013	minzipped size 2.8 KB
vue	184 008	536	Jun 1, 2023	Jul 29, 2013	minzipped size 22.9 KB

Рисунок 2.4 – Рейтинг фреймворків на Github

Ці показники відповідають результатам опитування Stackoverflow. Рейтинг Vue і React значно вищий за Angular. Це є причиною великої різниці у кількості завантажень пакетів.

Цей щорічний звіт[5] є головним показником розвитку JavaScript в цілому та всіх інструментів, які з ним пов'язані. На рисунку 2.5 відображено рейтинг фреймворків за кількістю використання.



Рисунок 2.5– Рейтинг фреймворків згідно State of JavaScript

Огляд думок про досліджувані технології з часом (з 2018 по 2022 роки) представлений на рисунку 2.6. На графіках можна відзначити такі особливості: відповідь «використовуватиму в наступних проектах» React дали майже 70% опитаних.

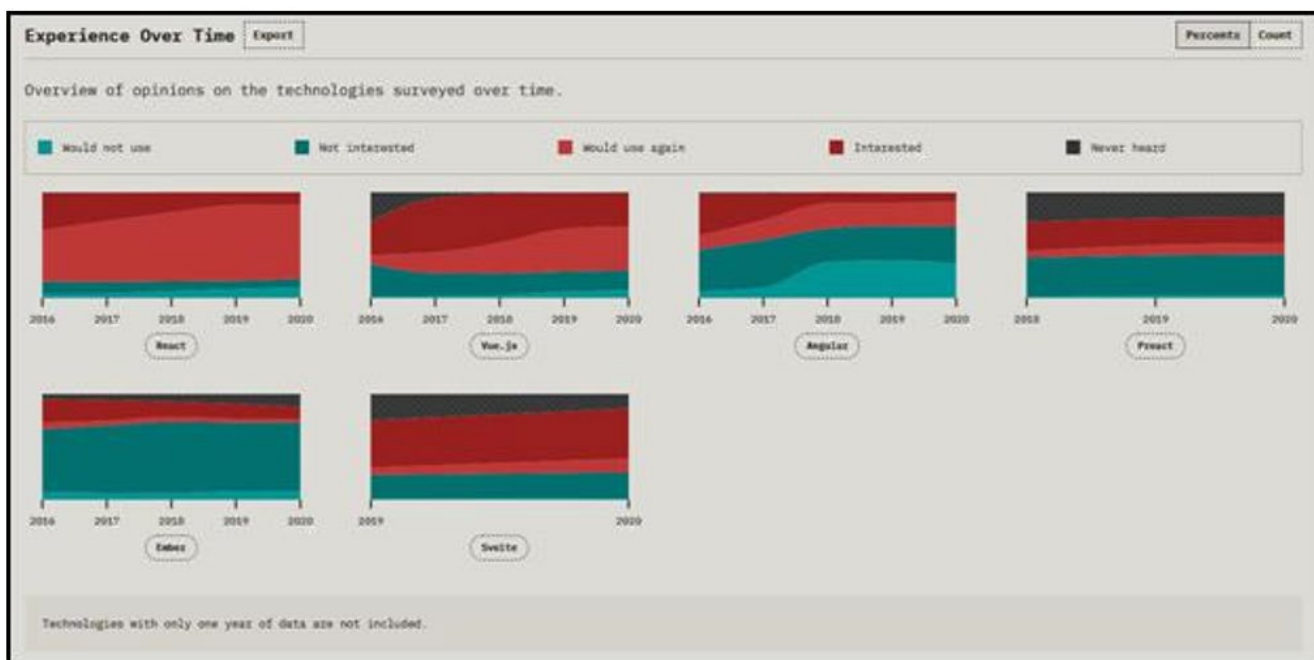


Рисунок 2.6 – Думки про досліджувані технології з часом

Відсоток тих, хто зацікавлений чи використовувати, залишається приблизно одному рівні. Vue демонструє стабільний інтерес та співвідношення між «зацікавленим у вивченні» та «не зацікавленим у вивченні». Angular інтерес знижується щороку на кілька відсотків.

Поділ за принципом «зацікавлений у вивченні», «використовуватиму в наступних проектах» проти негативних «не зацікавлений у вивченні» і «не використовуватиму у наступних проектах» відображено на рисунку 2.7. Примітно, що відсоток задоволених фреймворками React і Vue в цілому схожий при цьому відсоток значно вищий, ніж відсоток аналогічних відповідей про Angular.

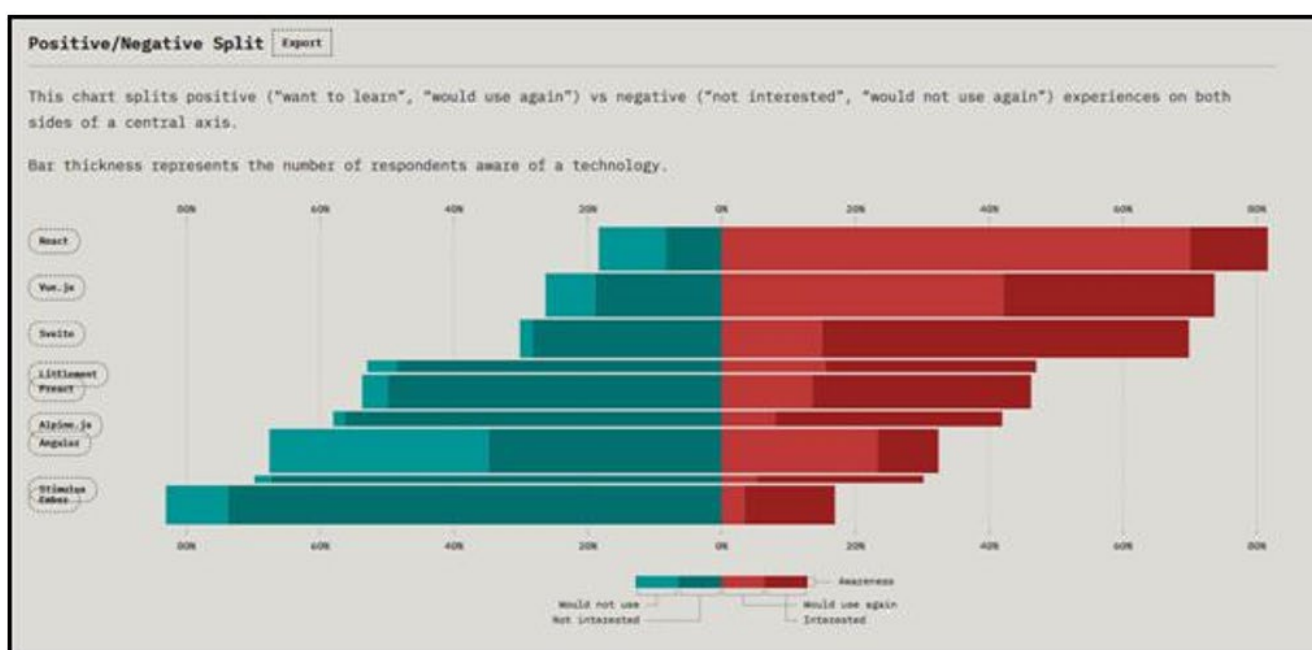


Рисунок 2.7 – Поділ по позитивному чи негативному відношенню до технологій

Примітно, що відносно невеликі компанії віддають перевагу React та Vue, тоді як великі розробники (від 100 і понад 1000 співробітників) частіше використовують Angular. Проте ці показники приблизно рівні. Статистика наведено на риснку 2.8.

Usage by Company Size Export Share

For each technology, how usage is spread among respondents who picked different company size ranges.

Number of Employees	1	1-5	5-10	10-20	20-50	50-100	100-1000	1000+
React	7.6%	7.2%	7.3%	9.3%	12.2%	11.6%	23.6%	26.2%
Vue.js	8.6%	8.5%	8%	10.9%	14.5%	11.3%	21.6%	18.4%
Angular	2.7%	4.5%	6%	8.4%	13.8%	11.9%	21.9%	22.9%

Рисунок 2.8 – Статистика використання фреймворків залежно від розміру компанії

2.2 Огляд Angular

На сьогоднішній день трійку Angular, Vue і React відносять до найкращих JS-фреймворків, але для вибору найбільш придатного для проекту необхідно врахувати безліч факторів. Для використання фреймворків розглянемо їх переваги та недоліки у контексті використання у проектах. Почнемо розгляд платформ із Angular.

Angular - це інтерфейсний JavaScript-фреймворк із відкритим вихідним кодом, який найкраще використовувати для складних великомасштабних проєктів за допомогою досвідченого розробника Angular. Angular належить до стеку MEAN, одного з найпопулярніших технологічних стеків для стартапів. Це середовище розробки веб-застосунків засноване на TypeScript. Angular здатний і переважно використовується для створення односторінкових веб-додатків або SPA.

Загальна інформація:

- розроблено Google;
- вперше випущений у 2010 році;
- фреймворк JavaScript на основі TypeScript;
- остання стабільна версія – Angular 11.1.1, випущена у січні 2023;

- швидко зарекомендувала себе як основна розробка; завантажено мільйонами розробників;
- в даний час близько 200 тисяч діючих веб-сайтів використовують Angular.

Основні характеристики:

- пропонує надійний набір компонентів, що спрощує написання,
- зміна та використання коду; скорочує час початкового завантаження веб-сторінки за рахунок поділу задач на логічні фрагменти через структуру MVC;
- модель MVC допомагає зменшити кількість запитів у фоновому режимі, дозволяючи поділ уявлень;
- Angular пропонує повністю настроюваний дизайн, надаючи розробникам та дизайнерам більше можливостей; пропонує безшовну сторонню інтеграцію для покращення функціональності продукту; «випереджаючий компілятор» забезпечує більш швидке завантаження та підвищену безпеку;
- під час розробки Angular компілює HTML і TypeScript JavaScript, що прискорює компіляцію коду навіть до того, як браузер завантажить ваш веб-додаток;
- за допомогою Angular впровадження залежностей можна використовувати як компоненти, що розділяють зовнішні елементи, що робить їх повторно використовуваними, більш простими в управлінні та тестуванні.

Плюси Angular:

- пропонує хорошу структурованість;
- проста двостороння прив'язка даних; архітектура MVC;
- початкове скорочення часу завантаження веб-сторінки; вбудована модульна система; велика спільнота та екосистема.

Мінуси Angular:

- обмежені можливості SEO;
- слабка доступність для пошукових роботів; складний вивчення;
- складні SPA повільні та неефективні.

Коли використовувати Angular:

- існують випадки, коли цей фреймворк найкраще підходить;
- Angular має сенс використовувати, якщо планується:
- розробка прогресивного веб-додатку (PWA);
- великий корпоративний додаток зі складною інфраструктурою; веб-сайти з динамічним контентом; оновлення дизайну веб-додатку;
- покращення користувацького досвіду великих веб-сайтів за рахунок редизайну.

Найбільш відомі веб-додатки та мобільні програми, створені з використанням фреймворку Angular зображені на рисунку 2.9.



YouTube TV



Google Cloud



Gmail



Forbes



UpWork



PayPal

Рисунок 2.9 – Найбільш відомі веб-програми на Angular

2.3 Огляд React

React - це інтерфейсна бібліотека JavaScript з відкритим вихідним кодом, яка використовується для розробки інтерфейсів користувача. Стек MERN є одним з найпопулярніших стеків для створення високопродуктивних бізнес-додатків, включає React, орієнтований на створення інтерактивних користувацьких інтерфейсів.

Інтерфейс користувача грає вирішальну роль у веб-розробці, коли користувач бачить додаток і взаємодіє з ним через нього. Завдяки своїй простоті та орієнтації на взаємодію з користувачем React став одним із найпопулярніших фреймворків.

Загальна інформація:

- розроблено Facebook; реліз у 2013 році;
- зовнішня бібліотека, що використовується для створення компонентів інтерфейсу користувача з відстеженням стану і багаторазового перевикористання;
- остання стабільна версія – React 17.0.2, випущена у 2023 році; із загальної кількості веб-сайтів розроблених з використанням React,
- 59,3% веб-сайтів використовують React v16, 19,1% використовують v15, 17,5% використовують React v0 та 4,1% використовують React 17;
- створює інтерфейси інтерфейсу користувача спеціально для SPA;
- приблизно 2 мільйони веб-сайтів використовують React. 2.2.2.2

Основні характеристики. Virtual DOM пропонує ефективність та оптимізовану продуктивність для складних програм, що вимагають частих змін. Віртуальна модель DOM дозволяє вносити зміни та оновлювати лише ту частину, де це потрібно. Це змушує браузер повторно відображати лише невелику частину сторінки, а чи не всю її.

React підтримує зв'язування та перемішування дерева DOM. Ці функції використовуються зниження навантаження на ресурси кінцевих користувачів.

React пропонує найкращий контроль над проектом, підтримуючи односторонню прив'язку даних. React легко тестується та відстежується, що покращує весь процес розробки.

Ключові переваги, які пропонує React та завдяки яким він виділяється:

- підвищує швидкість процесу розробки, дозволяючи реагує веб-розробникам писати окремі частини як на стороні клієнта, так і та на стороні сервера без зміни логіки програми;
- код React гнучкий завдяки своїй модульній структурі, економлячи багато часу та коштів;
- код React легше підтримувати;
- пропонує високу продуктивність складних програм;
- React пропонує розробку мобільних програм, підтримуючи мобільні нативні програми для платформ Android і iOS.

Мінуси React:

- React не реалізує MVC, що підштовхує до використання;
- додаткових бібліотек для реалізації станів та моделей;
- постійні оновлення React та використання супутніх бібліотек щодо його підтримки призводять до поганої документації;
- швидке прискорення технологій React призводить до написання неправильних інструкцій через брак часу;
- спеціалістам з React іноді складно прискорити постійні оновлення через швидкий розвиток технологій React.

Останнім часом Facebook випускає дуже багато оновлень, через що старіють деякі інструменти.

Чистий React потрібний не для кожного проекту. Але при використанні разом із Redux React стає потужним інструментом. Якщо програма досить проста, є можливість писати на JavaScript.

React найкраще підходить для проектів, які включають безліч компонентів з активним або неактивним станами, елементами навігації, розгорнутими або згорнутими розділами, динамічними введеннями, активними або вимкненими кнопками, а також правами входу та доступу користувачів. Керування такими станами, що змінюються, і подання користувачам різних уявлень залежно від стану з React не викликає проблем.

Якщо програма має шанс зрости в масштабі та обсязі, то React найкраще підійде, оскільки його декларативний характер компонентів дозволяє легко працювати з такими складними структурами.

Найбільш відомі веб-програми та мобільні програми, створені з використанням фреймворку React зображені на рисунку 2.10.

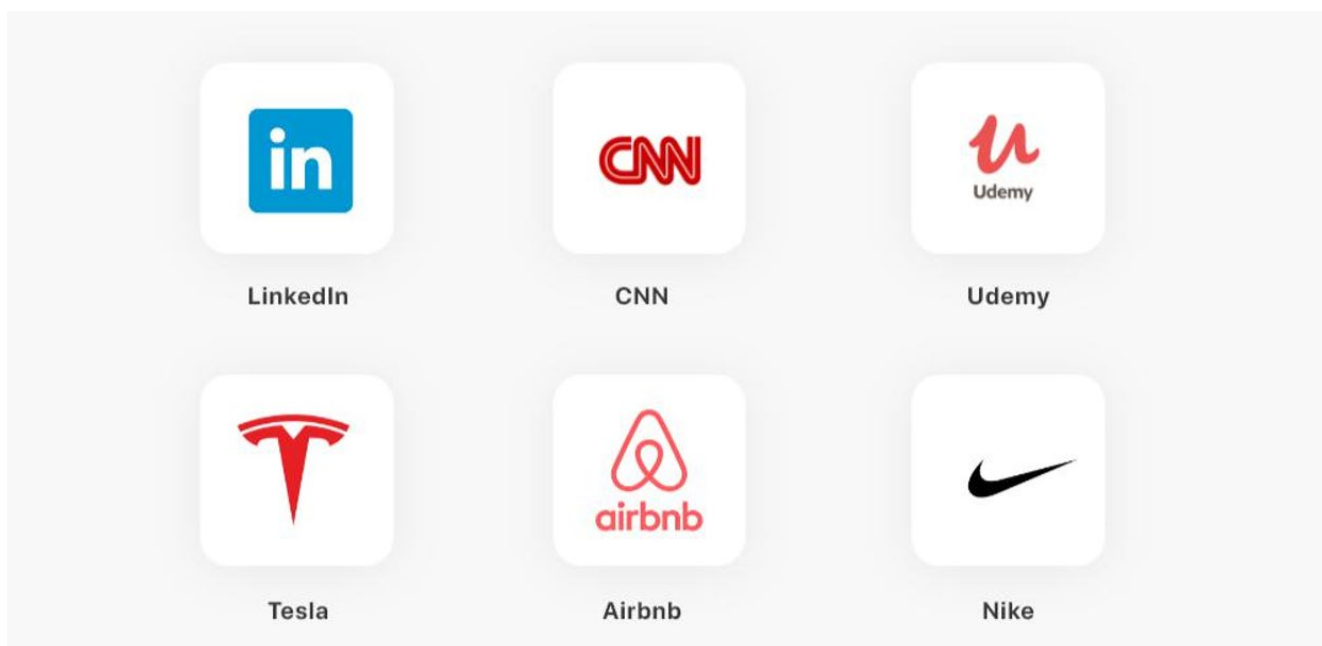


Рисунок 2.10 – Найбільш відомі веб-програми на React

2.4 Огляд Vue

Vue - це інтерфейсний JavaScript-фреймворк Model-View-View-Model відкритим вихідним кодом, який використовується для розробки інтерфейсів

користувача і односторінкових додатків (SPA). Цей JavaScript-фреймворк доступний, високопродуктивний та універсальний. Vue сприяє оптимальній та динамічній взаємодії з користувачем в Інтернеті.

Загальна інформація:

- розроблено Еваном Ю. у 2014 році; реліз у 2014 році;
- Vue створено для поступового впровадження користувачами;
- остання стабільна версія – 3.0.6, випущена у 2023 році; понад 1 мільйон веб-сайтів, створених за допомогою Vue.

Vue найкраще підходить для вирішення короткострокових проблем. Він може легко інтегруватись з існуючими блоками коду. Vue - виключно швидкий інструмент і володіє висококласними функціями, простою кривою навчання, високоефективним, швидким і складним односторінковим додатком, що робить його одним із найпопулярніших JS-фреймворків згідно зі статистикою StackOverflow[15].

Плюси Vue:

- розмір фреймворку досить малий, що спрощує завантаження та встановлення; вимагає меншої оптимізації від розробників Vue через своє
- розміру;
- через розмір фреймворку він стимулює розробку веб-додатків Vue.js та дозволяє експертам Vue відокремлювати віртуальну модель DOM від шаблону від компілятора;
- віртуальний DOM в Vue дозволяє оновлювати елементи на веб-сторінці без рендерингу всієї DOM;
- Vue забезпечує легкий доступ до всіх своїх функцій, які підтримують налаштування;
- Vue сумісний і гнучкий, оскільки Vue використовує HTML для рендерингу об'єктів і систему шаблонів для безшовної інтеграції з існуючими програмами;

- Vue можна використати повторно, якщо він правильно налаштований; кодова база не обтяжується в міру масштабування програми.

Мінуси Vue:

- Vue порівняно новий, і йому знадобиться час, щоб розширити його спільнота;
- ринок Vue не так широко популярний у порівнянні з React та Angular; потрібен час, щоб побачити високий попит на вакансії для кваліфікованих розробників Vue;
- Vue не можна використовувати для великомасштабних проєктів, оскільки йому бракує масштабованості;
- Vue має менше плагінів у порівнянні з React та Angular;
- програми Vue мають проблеми з продуктивністю зі старими версіями мобільних браузерів Safari та iOS.

Коли використовувати Vue. Фреймворк Vue має сенс використовувати, якщо:

- у додатку повно анімації та інтерактивних елементів;
- додаток вимагає безшовної інтеграції з кількома додатками; необхідно створити прототип без просунутих навичок;
- потрібно швидко запустити MVP.

Найбільш відомі веб-додатки та мобільні програми, створені з використанням фреймворку Vue зображені на рисунку 2.11.

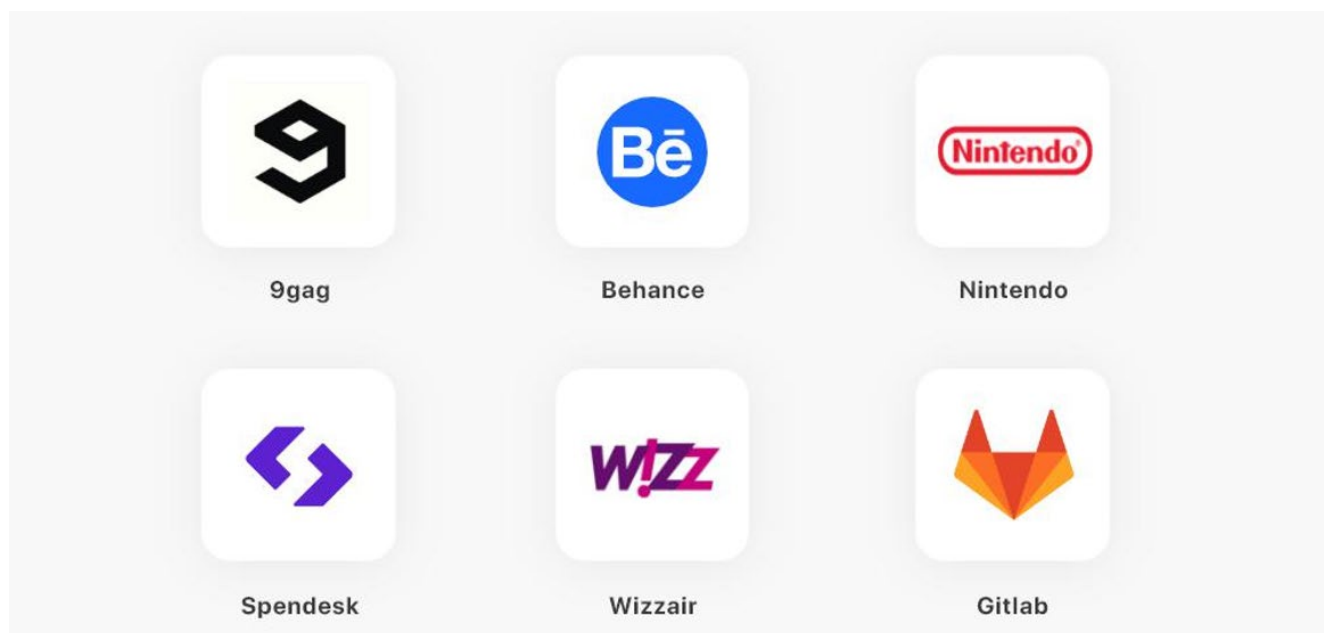


Рисунок 2.11 – Найбільш відомі веб-програми на Vue

2.5 Продуктивність програмного забезпечення

Вибираючи фреймворк чи бібліотеку, потрібно також думати про продуктивність. У багатьох випадках про продуктивність не потрібно турбуватися, особливо під час створення невеликого проекту. Однак чим більший масштаб і вищий складність проекту, продуктивність може (і буде) викликати занепокоєння. Важливо відзначити, що якість розробки та дотримання кращих практик, коли йдеться про веб-продуктивність, важливіше, ніж вибір фреймворку. Але деякі області роботи програми виміряти можливо, скористаємося метриками продуктивності і розглянемо їх і постараємося з'ясувати, у чому переваги чи недоліки фреймворків, і як кожен з них може вплинути на процес розробки.

Вибір фреймворка для реалізації проекту - це стратегічний процес, що вимагає уважного аналізу та обґрунтованого рішення. Під час підходу до цього вибору, необхідно врахувати низку ключових аспектів.

Перш за все, необхідно чітко визначити цілі проєкту. Кожен проєкт має свою унікальну специфіку, виходячи з якої можна визначити вимоги до функціональності, продуктивності, безпеки та масштабованості.

Дослідження ринку фреймворків і бібліотек для обраної мови програмування слід розглядати як етап, на якому відбувається зіставлення доступних інструментів із вимогами проєкту. Важливо звернути увагу на популярність і активність підтримки, оскільки вони свідчать про життєздатність інструменту.

Оцінка функціональності фреймворків стає важливим етапом. Проєкт може потребувати певних інструментів, а функціональні можливості фреймворка мають відповідати вимогам. Проведення тестових прототипів із кількома фреймворками дає змогу оцінити зручність використання та продуктивність.

Наступним кроком стає розгляд спільноти та підтримки навколо обраного фреймворку. Велика та активна спільнота забезпечує стабільність, актуальність документації та оперативне вирішення можливих проблем.

Важливим фактором є також адаптація до навичок команди розробників. Якщо команда вже знайома з певним фреймворком, це може підвищити продуктивність і якість розробки.

Також необхідно враховувати ресурси, які можна виділити на проєкт, включно з часом і бюджетом. Різні фреймворки можуть вимагати різного часу на освоєння і застосування.

Прогнозування майбутнього розвитку фреймворка, розгляд кейсів використання в аналогічних проєктах і ухвалення обґрунтованого рішення на основі аналізу всіх перерахованих вище чинників стають заключними етапами вибору фреймворка для реалізації проєкту.

Цей комплексний підхід до вибору фреймворка допоможе забезпечити відповідність технологічних рішень вимогам проєкту і збільшить ймовірність успішної розробки.

Вибір бібліотеки для реалізації проєкту - це складне рішення, яке включає в себе низку кроків. Спочатку необхідно зрозуміти цілі проєкту, ретельно

визначити, які завдання належить вирішити. Це первинне розуміння допоможе звужити коло пошуку і сконцентруватися на тому, що дійсно необхідно.

Далі слід аналізувати вимоги проєкту. Важливо виявити, які функціональні та нефункціональні характеристики бібліотеки мають відповідати вимогам. Технологічний стек проєкту також є ключовим моментом, оскільки обрана бібліотека має легко інтегруватися в наявну технологічну інфраструктуру.

Дослідження ринку бібліотек, доступних для вашої мови програмування, дає змогу порівняти їхню популярність та оцінити відгуки розробників. Це може слугувати гарним показником того, наскільки бібліотека користується довірою в спільноті.

Наступним кроком є оцінка функціональності бібліотек. Необхідно визначити, які з них якнайкраще підходять для вирішення конкретних завдань проєкту.

Важливим аспектом є також огляд ком'юніті та підтримки. Жвава і чуйна спільнота може істотно вплинути на процес розробки, надаючи швидкі відповіді на запитання та актуальну інформацію.

Сумісність та інтеграція з іншими компонентами проєкту стають не менш важливими факторами. Бібліотека повинна без проблем взаємодіяти з іншими частинами програми.

Аналіз продуктивності та безпеки бібліотек - це ще один етап вибору. Ефективність і надійність відіграють ключову роль, особливо якщо проєкт вимагає високої продуктивності та безпеки даних.

Також важливо врахувати ліцензування кожної бібліотеки, а також її довгострокові плани підтримки та розвитку. Це допоможе уникнути проблем у майбутньому і забезпечить стабільність у довгостроковій перспективі.

Остаточним етапом є тестування бібліотек у контексті створення прототипів. Це дає змогу оцінити їхню зручність використання та відповідність вашим очікуванням на практиці.

Зрештою, ухвалення обґрунтованого рішення вимагає врахування всіх вищезазначених аспектів, щоб обрана бібліотека якнайкраще відповідала потребам і цілям вашого проєкту.

Duration in milliseconds $\pm$ 95% confidence interval (Slowdown = Duration / Fastest)

Name Duration for...	vue-v3.0.6	angular-v12.0.1	react-v17.0.1
Implementation notes			
create rows creating 1,000 rows	106.3 $\pm$ 2.6 (1.00)	135.7 $\pm$ 1.5 (1.28)	138.7 $\pm$ 1.3 (1.30)
replace all rows updating all 1,000 rows (5 warmup runs).	93.2 $\pm$ 1.2 (1.00)	109.0 $\pm$ 1.6 (1.17)	112.7 $\pm$ 1.5 (1.21)
partial update updating every 10th row for 1,000 rows (3 warmup runs). 16x CPU slowdown.	153.4 $\pm$ 8.5 (1.06)	145.1 $\pm$ 1.8 (1.00)	193.2 $\pm$ 2.2 (1.33)
select row highlighting a selected row. (no warmup runs). 16x CPU slowdown.	147.4 $\pm$ 3.0 (2.05)	71.9 $\pm$ 1.8 (1.00)	102.6 $\pm$ 3.5 (1.43)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4x CPU slowdown.	45.0 $\pm$ 1.0 (1.00)	349.0 $\pm$ 1.9 (7.76)	342.6 $\pm$ 1.7 (7.62)
remove row removing one row. (5 warmup runs).	21.3 $\pm$ 0.4 (1.14)	18.7 $\pm$ 0.3 (1.00)	21.8 $\pm$ 0.9 (1.17)
create many rows creating 10,000 rows	922.6 $\pm$ 4.1 (1.00)	1,093.9 $\pm$ 15.5 (1.19)	1,470.9 $\pm$ 36.7 (1.59)
append rows to large table appending 1,000 to a table of 10,000 rows. 2x CPU slowdown.	221.3 $\pm$ 4.0 (1.00)	251.0 $\pm$ 4.3 (1.13)	263.2 $\pm$ 2.0 (1.19)
clear rows clearing a table with 1,000 rows. 8x CPU slowdown.	106.4 $\pm$ 0.8 (1.00)	215.7 $\pm$ 5.2 (2.03)	125.0 $\pm$ 1.5 (1.17)
geometric mean of all factors in the table	1.11	1.47	1.57
compare: Green means significantly faster, red significantly slower	compare	compare	compare

Рисунок 2.12 – Час, необхідний обробку операцій із таблицями

Результати тесту JS Framework Benchmark[17] показують, що всі фреймворки досить добре працюють у більшості тестів, таких як створення або додавання рядків до таблиці (рисунок 2.12), час завантаження програми з використанням інструменту Lighthouse (рисунок 2.13), використання пам'яті додатком (рисунок 2.14).

Startup metrics (lighthouse with mobile simulation)

Name	vue-v3.0.6	angular-v12.0.1	react-v17.0.1
consistently interactive a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	2,105.0 ± 10.1 (1.00)	2,443.8 ± 1.3 (1.16)	2,579.1 ± 1.4 (1.23)
script bootup time the total ms required to parse/compile/evaluate all the page's scripts	5.9 ± 0.5 (1.00)	179.8 ± 11.4 (30.63)	5.9 ± 0.5 (1.01)
total kilobyte weight network transfer cost (post-compression) of all the resources loaded into the page.	197.9 ± 0.0 (1.00)	293.8 ± 0.0 (1.48)	274.2 ± 0.0 (1.39)
geometric mean of all factors in the table	1.00	3.75	1.20

Рисунок 2.13 – Час завантаження програми

Memory allocation in MBs $\pm$ 95% confidence interval

Name	vue-v3.0.6	angular-v12.0.1	react-v17.0.1
ready memory Memory usage after page load.	1.2 (1.00)	1.8 (1.48)	1.3 (1.06)
run memory Memory usage after adding 1000 rows.	3.5 (1.00)	4.2 (1.19)	4.4 (1.25)
update each 10th row for 1k rows (5 cycles) Memory usage after clicking update every 10th row 5 times	3.7 (1.00)	4.5 (1.23)	5.2 (1.42)
replace 1k rows (5 cycles) Memory usage after clicking create 1000 rows 5 times	4.1 (1.00)	5.2 (1.28)	5.2 (1.28)
creating/clearing 1k rows (5 cycles) Memory usage after creating and clearing 1000 rows 5 times	2.6 (1.00)	3.5 (1.38)	3.0 (1.15)
geometric mean of all factors in the table	1.00	1.31	1.23

Рисунок 2.14 – Використання пам'яті програмою

Як можна помітити, Vue значно повільніше, ніж Angular та React при виборі та підсвічуванні рядків. З іншого боку, Angular і React не дуже ефективні при зміні порядку рядків. Це єдині істотні відмінності в тестах рендерингу, які в більшості випадків не дають результатів, що відрізняються. Оскільки вибір рядків є більш поширеною функцією, ніж їх заміна, я сказав би, що цей тест ставить Vue на третє місце після Angular і React, які займають лідируючу позицію.

Що стосується пам'яті та часу завантаження, React та Vue показують дуже хороші результати, але Angular працює трохи повільніше. Angular може знадобитися 150 мс для завантаження базового скрипта і потребує більше пам'яті для запуску.

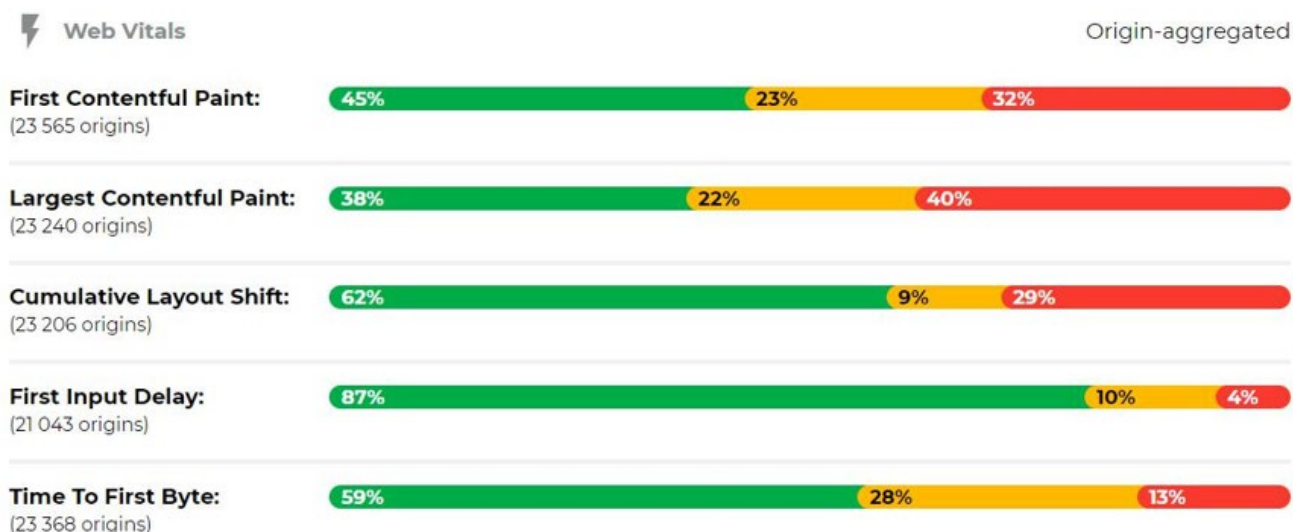


Рисунок 2.15 – Статистика програм на Angular

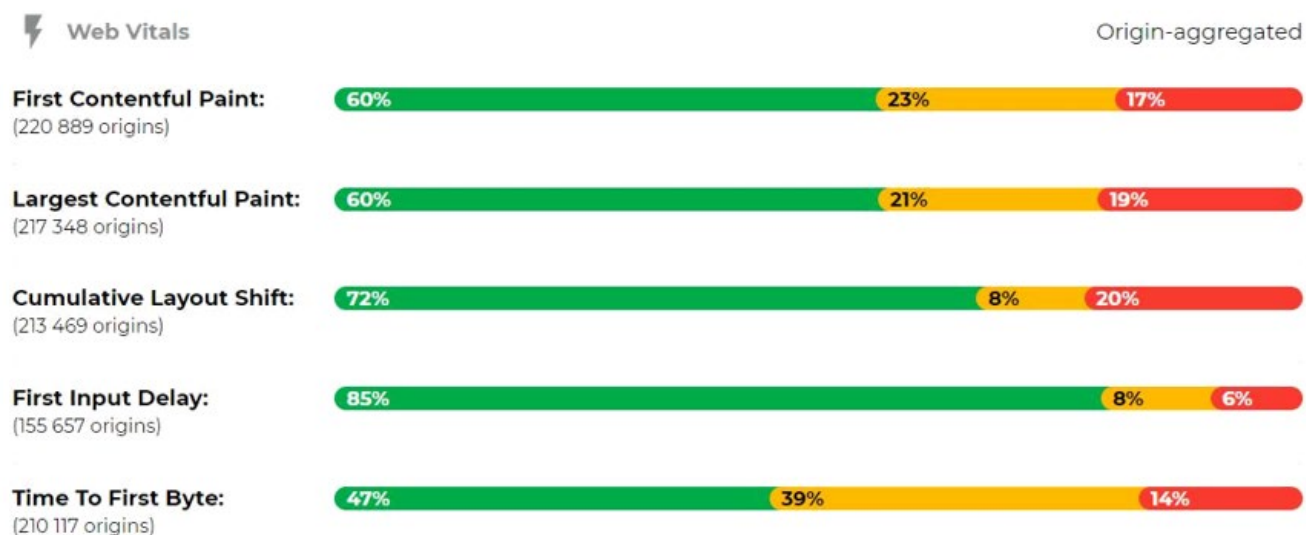


Рисунок 2.16 – Статистика програм React

Perf Track від Google Chrome Labs показує продуктивні дані з тисяч веб-сайтів. На цю статистику впливають багато інших чинників, а не лише вибір

фреймворку, але давайте подивимося на дані. Статистика Perf Track за трьома фреймворками наведена на рисунках 2.15, 2.16, 2.17 [16].

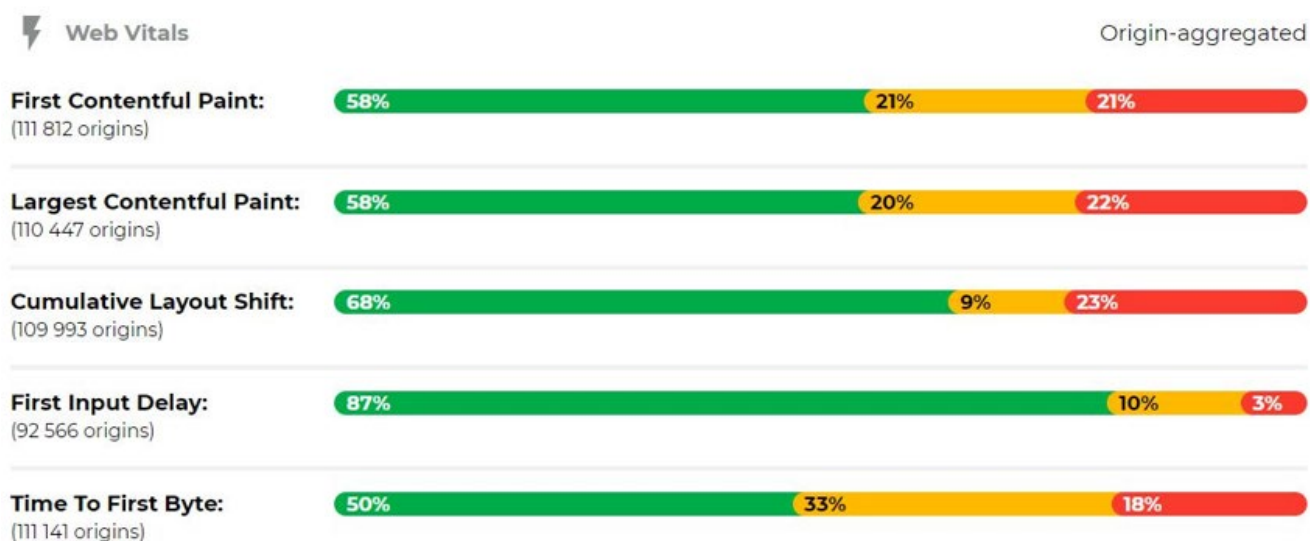


Рисунок 2.17 – Статистика програм на Vue

Час відображення першої частини контенту: веб-сайти Vue та React мають більш високий рейтинг за цим показником у порівнянні з Angular, який може зайняти більше часу для завантаження та представлення контенту користувачеві.

Час відтворення найбільшої частини контенту: Angular також є найповільнішим серед трьох фреймворків. При рендерингу повному сторінки лише 27% веб-сайтів на Angular мають оцінки у прийнятному діапазоні.

Зміщення контенту: додатки на Angular з хорошими та середніми показниками становлять 70% від числа всіх додатків на Angular, що є найгіршим результатом порівняно з React (80%) та Vue (77%).

Затримка при першому введенні користувача: для всіх трьох платформ понад 80% веб-сайтів знаходяться в допустимому діапазоні для затримки, яка показує, скільки часу потрібно, щоб користувач міг взаємодіяти зі сторінкою.

Вага програми: найбільш легковажними програмами на сьогоднішній день є програми, розроблені за допомогою Vue, при цьому 68% програм Vue завантажують менше 1 МБ JavaScript. Програми на Angular та React, як правило, мають більший розмір коду.

Здається, що за цими даними можна помітити тенденцію, але висновки робити зарано. Для Vue, наприклад, пояснення швидкості може у тому, що Vue використовується розробки більш легких додатків, тоді як Angular використовується для більших проєктів.

Зрештою статистика може допомогти вам прийняти правильне рішення, але вона не повинна бути ключовим фактором порівняння, оскільки дану статистику не можна використовувати для доказу, що один фреймворк швидше або краще за інший.

Для визначення складності вивчення кожного з фреймворків звернемо увагу на складність початку роботи з технологією, концепцій, які вони вводять.

Найпростіший випадок React - найменш складний із трьох фреймворків. Це тому, що вам потрібно тільки імпортувати бібліотеку, після чого ви можете почати писати свою програму React за допомогою кількох рядків коду. Але за винятком прикладу Hello World більшість програм React засновані на компонентах і не просто відображають кілька елементів на сторінці.

Одна річ, яку деякі розробники знаходять у React дивною чи складною, – це те, що вивчення JSX – це фактично необхідність. Звичайно, можна використовувати чистий JavaScript, але оскільки більшість розробників React використовують JSX, його вивчення значною мірою неминуче.

Вивчення JSX – це основна складність у кривій навчання. З іншого боку, React – це бібліотека, яку легко освоїти розробникам, які знають JavaScript та розуміють концепції веб-розробки.

Vue трохи складніше налаштувати, ніж React. Його можна використовувати як бібліотеку для визначення компонентів, які можна використовувати в HTML, але, як і в React, більшість проєктів створюється іншим чином. Більшість проєктів Vue матиме кореневий компонент з ім'ям App.vue та кілька дочірніх компонентів для відображення різних елементів.

Коли справа доходить до синтаксису, єдине, що вам потрібно вивчити - це синтаксис шаблонів Vue, який дуже легко зрозуміти, якщо ви знаєте HTML. Базові директиви, такі як v-if та v-for для умовного рендерингу та рендерингу списку,

легко зрозуміти навіть новачкам. Крім того, однофайлові компоненти Vue зберігають весь інтерфейсний код в одному місці, що полегшує організацію нових проектів. На мій погляд, Vue найлегше вивчити через його простоту та інтуїтивно зрозумілий синтаксис.

У Angular - найскладніша структура проекту з трьох фреймворків, і оскільки це повноцінний інтерфейсний фреймворк, він спирається на більшу кількість концепцій. Крім компонентів, Angular підтримує модулі та служби. Передбачається, що ви напишете та спроектуєте свою кодову базу особливим чином, який зробить ваш проект зручнішим для супроводу під час його масштабування.

Що стосується синтаксису, оскільки Angular найкраще працює з TypeScript, необхідно знати TypeScript під час створення проекту Angular. Як і у випадку з Vue, потрібно ознайомитися з синтаксисом, подібним до HTML, для створення нових функцій UI. На мій погляд, Angular найважче вивчити середньому розробнику, тому що він складніший і заснований на TypeScript.

2.6 Висновки

Було проведено порівняння архітектурних відмінностей, знайшли сильні та слабкі сторони кожного фреймворку та порівняли сфери застосування. Angular, React та Vue знаходяться у стадії дуже активної розробки. Вони регулярно випускають нові версії та підтримують існуючі. Оскільки поточний рівень підтримки в кожному випадку високий, можна без побоювань використовувати будь-який з цих кадрів.

Важливо відзначити, що Angular росте не так швидко, як раніше, тоді як Vue, хоч і з'явився відносно недавно, сильно зростає. Важко передбачити, які фреймворки залишаться актуальними у довгостроковій перспективі, але за кожним із них стоїть велика спільнота, і вони постійно розвиваються.

Результатом дослідження можна позначити наступне: виходячи з вимог до складності використання та масштабованості, а також з огляду на відсутність досвіду в будь-якому з цих фреймворків, було обрано варіант з реалізацією на React.

3 РОЗРОБКА ПІДСИСТЕМИ ВЗАЄМОДІЇ

3.1 Опис платформи

Платформу тестування можна логічно поділити на 3 частини: frontend, backend та ядро тестування. Як СУБД використовуються PostgreSQL та Redis для тимчасового зберігання даних ядра. Схема взаємодії компонентів відображена рисунку 3.1.

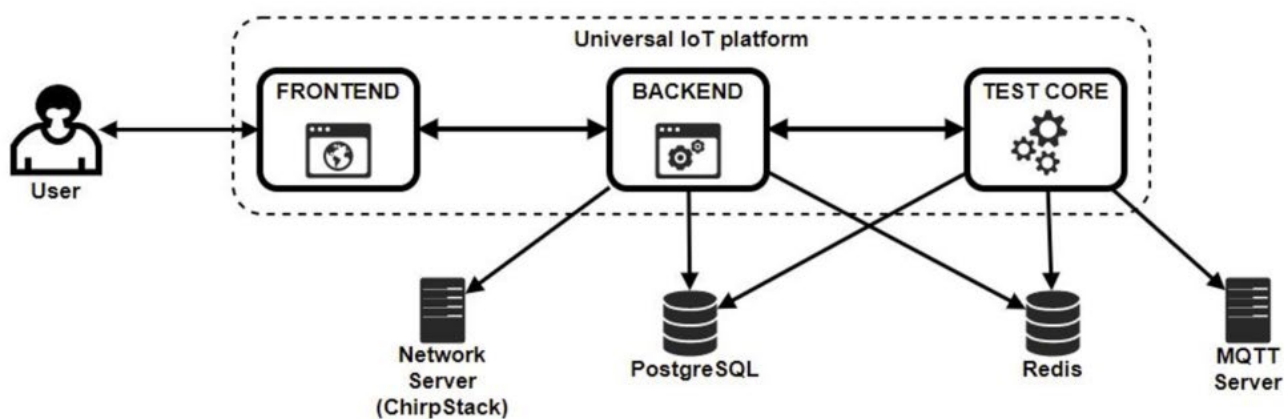


Рисунок 3.1 – Компоненти платформи тестування

PostgreSQL - це потужна, відкрита система керування реляційними базами даних (СУБД), розроблена з акцентом на розширюваність і відповідність стандартам. Від початку свого створення 1986 року, PostgreSQL активно розвивалася і перетворилася на одну з найбільш передових і надійних баз даних у світі.

Однією з важливих особливостей PostgreSQL є її відкритий вихідний код, що дає змогу користувачам вільно використовувати, змінювати та поширювати програмне забезпечення відповідно до ліцензії PostgreSQL. Це сприяє активній спільноті розробників і забезпечує постійне вдосконалення СУБД.

PostgreSQL надає повний спектр функціональності реляційних баз даних, включно з підтримкою SQL, транзакціями для забезпечення цілісності даних, а також широкими можливостями оптимізації запитів. Система підтримує об'єктно-реляційну модель, даючи змогу створювати користувацькі типи даних, функції та мовні розширення, що робить її надзвичайно гнучкою для різноманітних додатків.

Одним із важливих напрямків розвитку PostgreSQL є підтримка JSON-даних, що дає змогу працювати з неструктурованими та напівструктурованими даними, що є актуальним у контексті сучасних вимог до зберігання й обробки інформації. PostgreSQL також надає засоби для повнотекстового пошуку, що розширює її застосовність у додатках, де важливий точний і ефективний текстовий пошук.

СУБД забезпечує можливості для створення відмовостійких систем через налаштування PostgreSQL, що почала свій шлях у 1986 році як проект POSTGRES в університеті Каліфорнії в Берклі, з плином часу зазнала значних змін і стала затребуваним інструментом у світі баз даних. Одним із ключових моментів, який виділяє PostgreSQL, є його прагнення до повноти стандартів SQL. Це робить його потужним засобом для тих, хто звик працювати з реляційними базами даних.

З відкритим вихідним кодом PostgreSQL надає розробникам необмежені можливості для налаштування та оптимізації під свої унікальні потреби. Можливість створення користувацьких функцій, типів даних і навіть мов розширень дає змогу адаптувати базу даних до конкретних бізнес-сценаріїв.

JSONB (бінарний JSON) і підтримка NoSQL-типів даних відкривають двері для роботи з гнучкішими і напівструктурованими даними. В епоху, де неструктурована інформація стає все більш важливою, PostgreSQL надає надійні інструменти для роботи з різноманітними форматами даних.

Крім того, PostgreSQL активно розробляється з урахуванням вимог сучасних додатків і хмарних технологій. Багатокористувацька підтримка і можливості масштабування роблять його відповідним вибором для розподілених і відмовостійких систем.

Завдяки повнотекстовому пошуку, PostgreSQL може ефективно обробляти запити, пов'язані з текстовими даними. Це може бути ключовим аспектом, особливо в додатках, де важливий точний і швидкий пошук за великими обсягами текстової інформації.

Redis (Remote Dictionary Server) - це високопродуктивна система керування даними, яка слугує для зберігання та обробки структурованих даних. Вона належить до категорії NoSQL-баз даних і часто використовується в сценаріях, де потрібен швидкий доступ до даних і підтримка високих навантажень.

Redis розроблено для роботи з ключ-значення, що означає, що кожне значення в базі даних пов'язане з унікальним ключем. Ця проста структура даних дає змогу ефективно виконувати операції читання і запису, роблячи Redis ідеальним вибором для завдань, де швидкодія відіграє ключову роль.

Однією з основних переваг Redis є його здатність зберігати дані в пам'яті, що забезпечує високу швидкість доступу. Це особливо корисно у випадках, коли дані необхідно отримувати дуже швидко, наприклад, у кешуванні даних або зберіганні лічильників для аналітики.

Крім того, Redis підтримує різні типи даних, зокрема рядки, списки, множини, хеші та відсортовані множини. Це розширює можливості використання Redis і дає змогу ефективно моделювати різні структури даних у застосунках.

Однією з чудових особливостей Redis є підтримка публікації/підписки. Цей механізм дає змогу клієнтам підписуватися на певні події та отримувати повідомлення про них. Це корисно для побудови асинхронних систем, опрацювання подій та оновлень у реальному часі.

Redis також забезпечує високу доступність через механізми реплікації та шардингу. Реплікація дає змогу створювати резервні копії даних, а шардинг ділить дані між кількома серверами для балансування навантаження.

Слід зазначити, що Redis хоча і має високу продуктивність і ефективність у певних сценаріях, вона може бути неоптимальною для складних запитів або аналітики, де реляційні бази даних можуть бути більш придатними. Однак, завдяки своїй швидкості та простоті використання, Redis залишається одним із найпопулярніших інструментів у сфері зберігання даних у пам'яті.

Redis також часто застосовується як кеш даних для поліпшення продуктивності додатків. Завдяки зберіганню даних у пам'яті та можливості завдання часу життя ключів, Redis може слугувати ефективним механізмом кешування, знижуючи навантаження на основне сховище даних.

Ще одним важливим аспектом Redis є його простота у використанні та налаштуванні. Клієнтські бібліотеки для багатьох мов програмування полегшують взаємодію з Redis, а його зручний інтерфейс командного рядка дає змогу легко взаємодіяти з базою даних для тестування і налагодження.

Redis також підтримує транзакції, що забезпечує атомарність виконання кількох команд. Це важливо в сценаріях, де цілісність даних є критичним аспектом.

Варто зазначити, що порівняно з деякими іншими СУБД, Redis не надає довгострокове зберігання даних на диску в тому вигляді, в якому вони знаходяться в пам'яті. Тому в разі перезапуску сервера дані можуть бути втрачені. Цей аспект важливий під час проектування систем, де потрібне збереження даних на довгий термін.

На закінчення, Redis є потужним і гнучким інструментом для роботи з даними в пам'яті. Його простота використання, висока продуктивність і різноманітність функцій роблять його популярним вибором для сценаріїв, де потрібна швидкодія та ефективність у роботі зі структурованими даними.

Клієнтська програма містить два модулі, а саме: налаштування пристроїв і підключень і управління випробуваннями. Налаштування пристроїв та підключень містить інформацію про пристрої, їх описи та технічну інформацію та підключення до суміжних систем, таких як ядро тестування та шлюз. Розширена схема відображена рисунку 3.2.

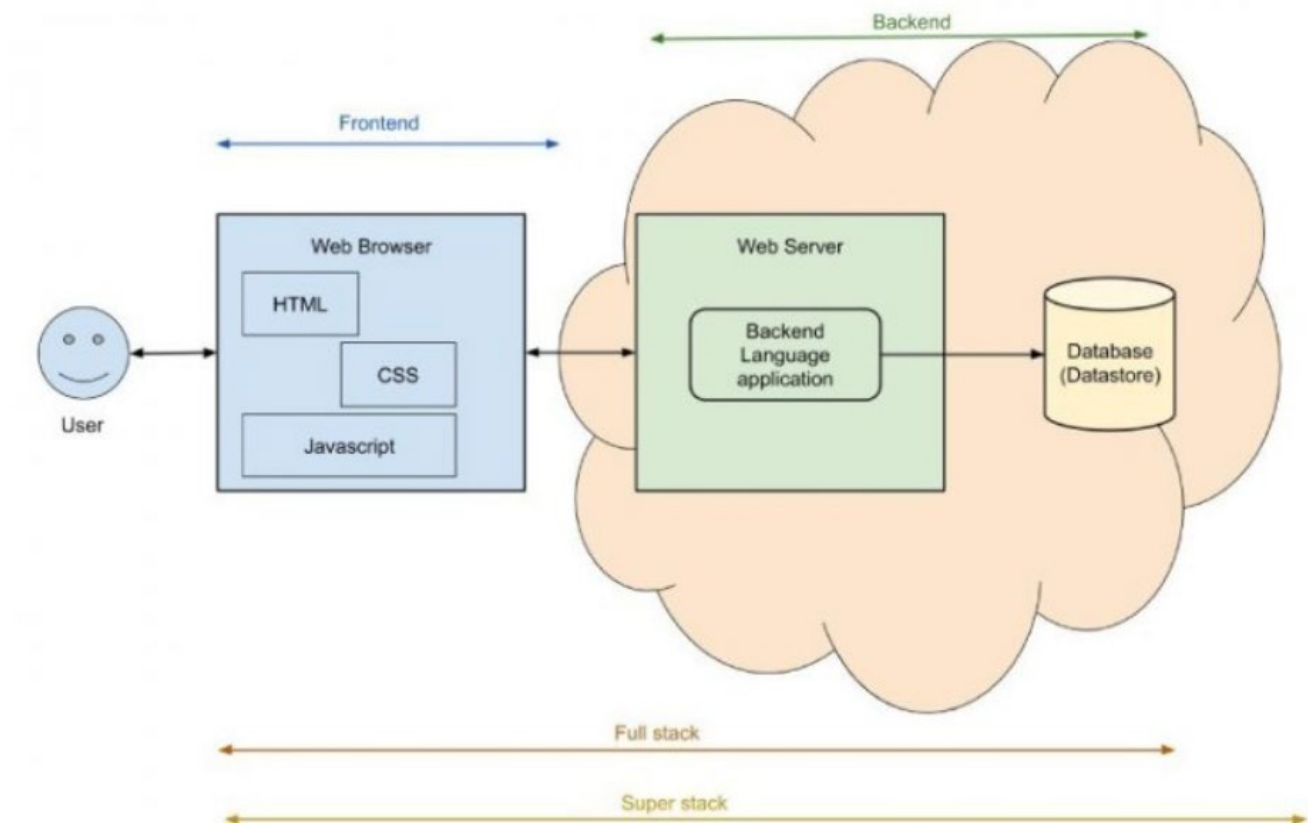


Рисунок 3.2 – Розширений ракурс платформи тестування

Модуль управління випробуваннями (клієнтський додаток) містить управління групами випробувань (створення, зміна груп), перегляд та скачування результатів випробувань та редактор коду сценаріїв. Робота зі створення сценаріїв у платформі реалізована лише у вигляді редактора коду. Редактор відображено на рисунку 3.3.

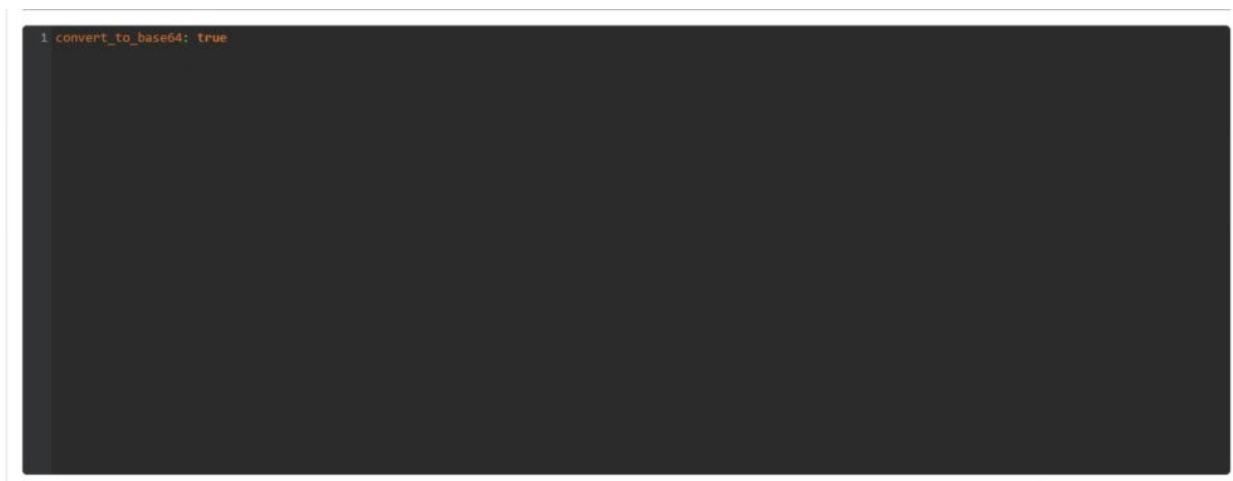


Рисунок 3.3 – Редактор сценаріїв

Створення сценаріїв із використанням редактора не має особливих труднощів для досвідченого користувача, проте він незручний для постійного використання. Було висунуто припущення, що реалізація конструктора тестів у вигляді візуального інтерфейсу покращить досвід користувача і дозволить не тільки економити час користувача-тестувальника на створенні сценаріїв, але і в цілому спростить використання платформи.

У платформі автоматизованого тестування пристроїв Інтернету речей також немає явного способу відстеження процесу тестування пристроїв у реальному часі. Це означає, що починаючи від відправки сценаріїв (або групи сценаріїв) в ядро тестування до моменту запису остаточного результату в таблицю TestResults, користувач не володіє інформацією про хід тестування. Крім відстеження ходу тестування користувач не має інформації про статус платформи, чи вона доступна, і які повідомлення вона відправляє на пристрої і отримує від них. Для вирішення цієї проблеми було запропоновано реалізувати моніторинг стану платформи.

Таким чином, конструктор тестів та моніторинг стану платформи будуть об'єднані у підсистему. Підсистема буде частиною клієнтської програми.

3.2 Розробка підсистеми

Етап розробки починається після аналізу та вибору відповідних технологій та програмних засобів. На етапі розробки спочатку визначаються окремі сутності, з якими вестиметься робота. Деякі сутності для загальної частини платформи тестування були визначені раніше, тепер необхідно уточнити, які сутності будуть ключовими в конструкторі тестів і моніторингу стану.

Головна сторінка клієнтської програми з кнопками запуску конструктора тестів та моніторингу стану відображена нарисунку 3.4.

❖ Платформа для тестування пристроїв Інтернету речей

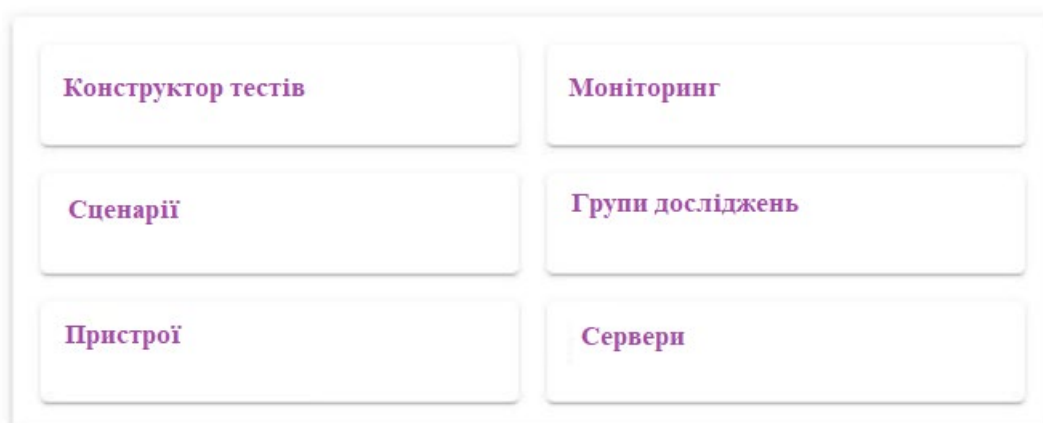


Рисунок 3.4 – Головна сторінка клієнтської програми

3.3 Конструктор тестів

Проведено декомпозицію сутності «Експеримент». В основі кожного експерименту лежить сценарій виконання, на цьому етапі можна визначити нову сутність «Сценарій», сценарій містить у собі свій текстовий опис, текст команд, прапор активності та дату створення.

Текст команд зберігати окремо не завжди необхідно та не дуже зручно погляду створення нових сценаріїв. Для спрощення розділимо сутність «Сценарій» на дрібнішу складову частину – сутність «Тест». Кожен сценарій міститиме низку тестів. До тесту можна вказати додаткові налаштування, такі як назва, прапор необхідності перетворення команд у кодування Base64, два поля налаштувань спроб перезапуску тесту.

Для гнучкішої можливості налаштувань тестів сутність «Тест» можна розкласти на кінцеву і далі неподільну складову частину: сутність «Команда». Команда містить базовий набір: назву команди, текст команди та очікувану відповідь від пристрою.

Сутності Тест, Команда, Сценарій можна зібрати в одному додатку - конструкторі тестів. Для зручності роботи з сутностями "Тест" та "Команда" додамо допоміжну сутність, тип команди - словник типів команд, в початковому стані заповнений такими значеннями: "Test", "BeforeTest", "AfterTest".

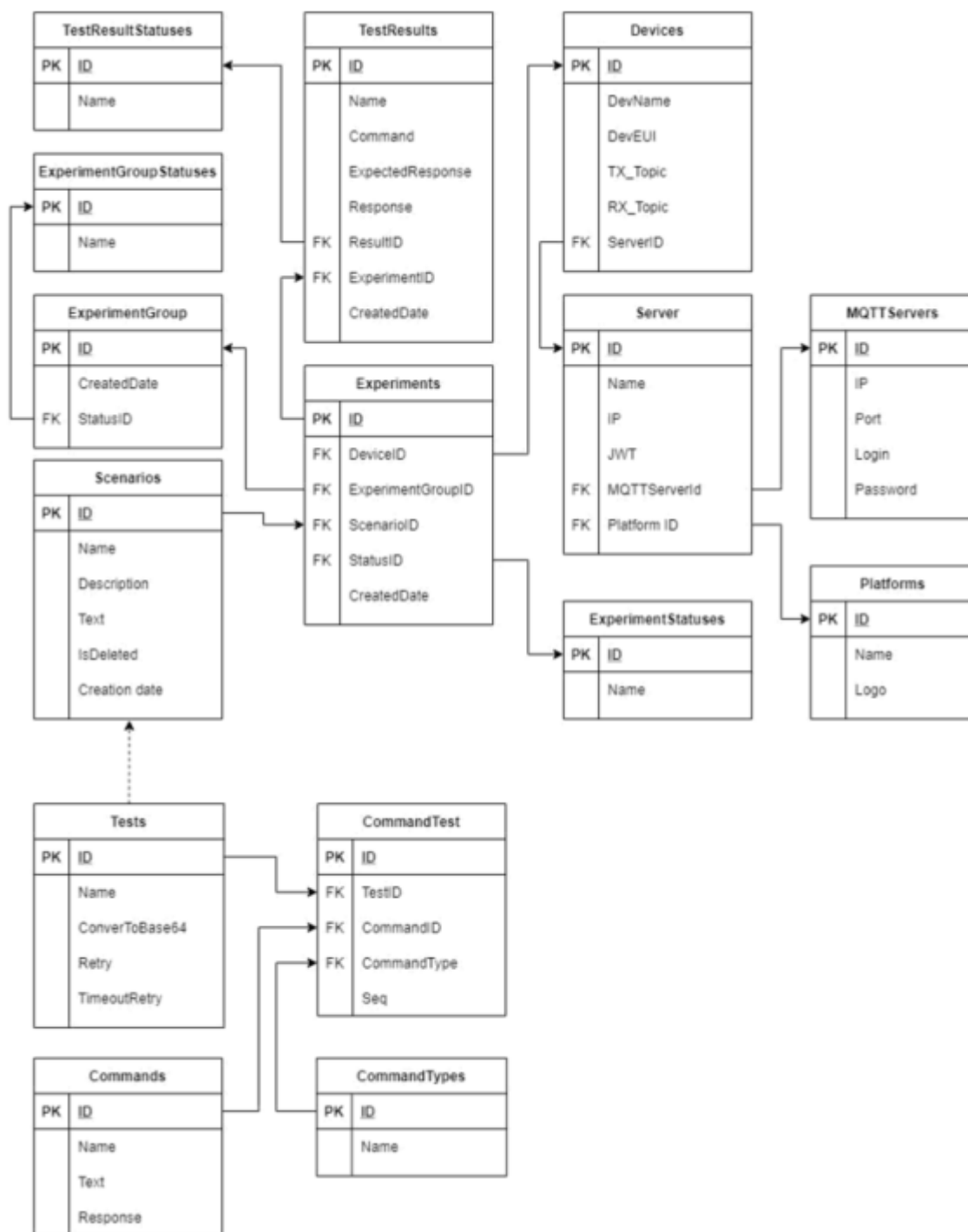


Рисунок 3.5– Загальна схема БД

Конструкторі тестів передбачається створення сутностей від базової («Команда») до фінальної – «Експеримент» по порядку. Структура загальної бази даних відображена на рисунку 3.5.

На цьому етапі розглянуто загальну структуру БД. У наступній частині буде описано процес розробки програми з використанням описаних сутностей.

Розробка всієї клієнтської програми ведеться в Visual Studio Code, оскільки цей інструмент безкоштовний, підтримує велику кількість плагінів і дозволяє легко підключати бібліотеки. У Visual Studio Code під час розробки використовувалися модулі node.js[6]. У клієнтській програмі використовується зв'язка з React[7][8] + MaterialUI[9].

Конструктор тестів можна розбити на 2 логічні частини: робота з тестами та робота з командами. Робота з тестами складається з 2 сторінок: сторінка огляду наявних тестів, сторінка створення тесту.

Сторінку огляду наявних тестів представлено на рисунку 3.6. На ній реалізовано функціонал відображення списком створених тестів, а також перехід на сторінку створення тесту при натисканні на кнопку «Створити тест» і при натисканні на «Оформити сценарій» текст команд обраних тестів буде оформлено в один загальний текст, збережений у таблиці «Сценарії».

❖ Платформа для тестування пристроїв Інтернету речей

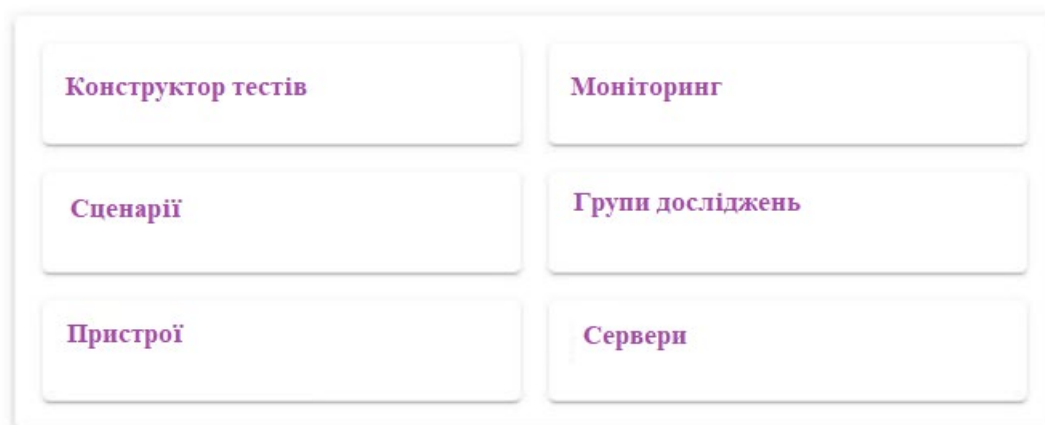


Рисунок 3.6 – Сторінка тестів конструктора тестів

Створення тесту відбувається у два етапи. На першому етапі (рисунок 3.7) користувачеві необхідно вказати назву створюваного тесту, вибрати з списку команди, що випадає, які будуть виконуватися до і/або після тесту, і додаткові параметри, такі як конвертація і налаштування перенаправлення команди пристрою.

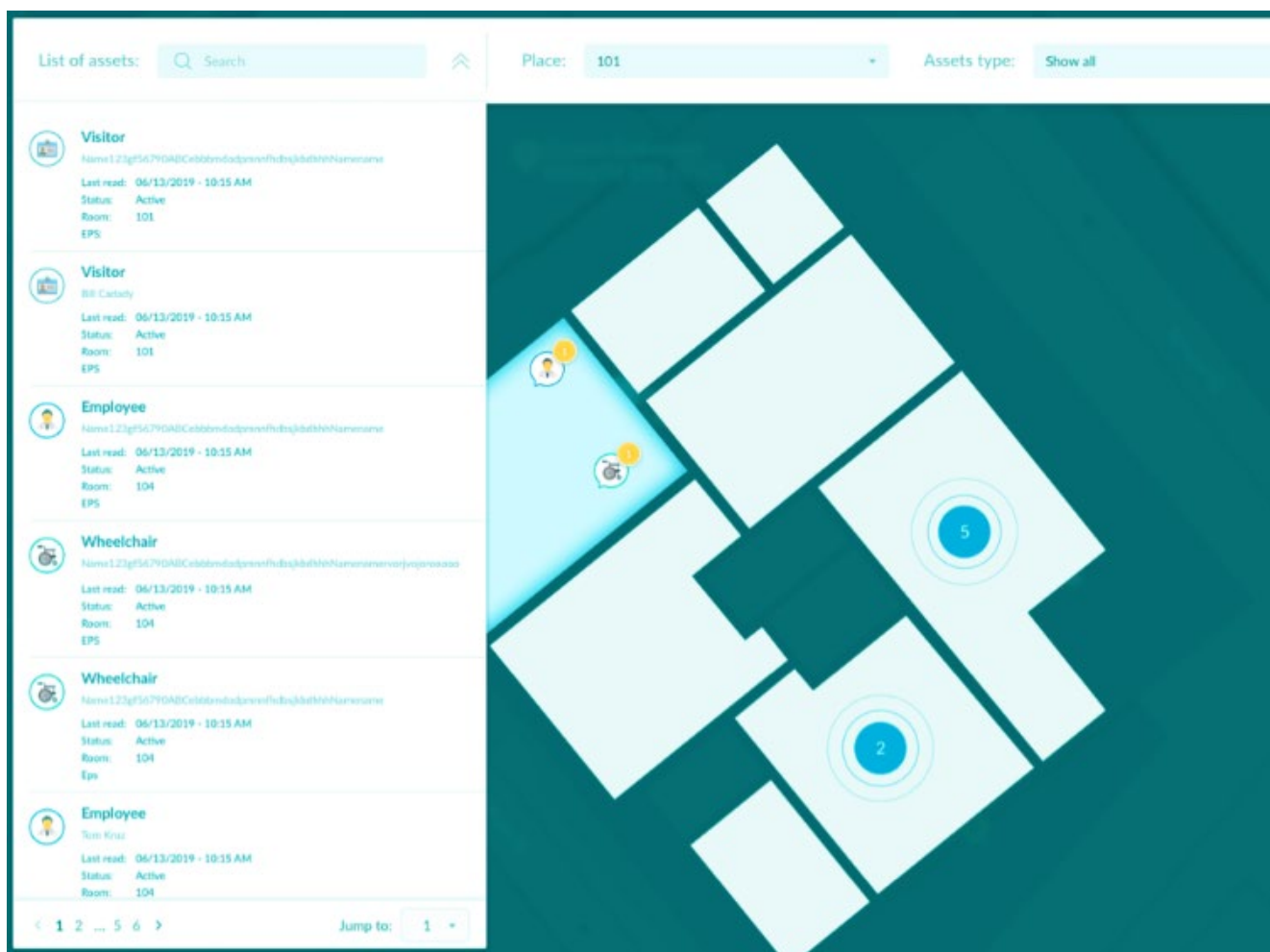


Рисунок 3.7 – Перший крок створення тесту

На другому етапі користувач вибирає зі списку раніше створені команди та додає їх у тест. У цьому створення тесту завершується.

Робота з командами аналогічно до тестів складається з 2 сторінок: сторінка огляду наявних команд і сторінка створення. Сторінка огляду (рисунок 3.7) складається зі списку команд, кнопки "Створити команду", при натисканні на яку відбувається перехід на сторінку створення.

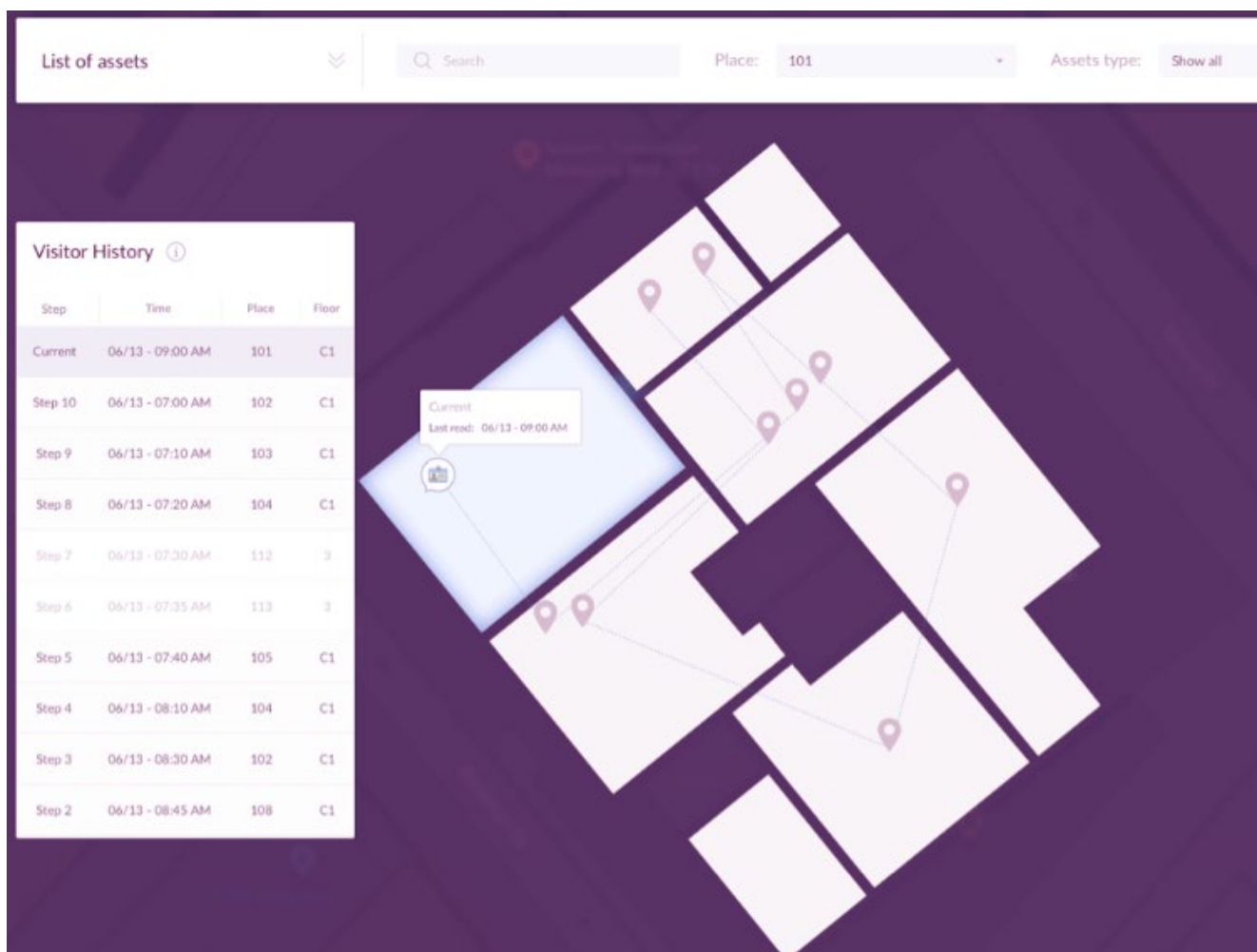


Рисунок 3.8 - Сторінка команд у конструкторі тестів

Сторінка створення в цілому аналогічна сторінці створення тесту, але містить менше полів і заповнюється в один етап. На цьому етапі (рисунок 3.8) користувач вводить назву команди, вибирає тип команди та аповнює текст (виконуваний код) та очікувану відповідь на команду від пристрою. На цьому створення команди завершується.

Робота з командами подібна до взаємодії з тестами і включає дві основні сторінки: сторінку перегляду наявних команд і сторінку створення нових. Сторінка перегляду включає у себе перелік існуючих команд та кнопку "Створити команду". При натисканні на цю кнопку відбувається перехід на сторінку створення нової команди.

3.4 Моніторинг стану платформи

Додаток моніторингу повністю ґрунтується на даних по платформі з Redis[10]. У Redis дані надходять безпосередньо з ядра тестування та описують стан платформи.

Redis використовується як інструмент для зберігання тимчасової інформації про роботу: стан, черга надісланих та отриманих повідомлень під час тестування для кожного пристрою.

За допомогою Redis автоматизована платформа тестування пристроїв інтернету речей зможе отримувати інформацію про стан ядра тестування в даний час, і відображати її користувачеві через front-end. До цієї СКБД також записуватимуть повідомлення, які були отримані та надіслані під час тестування.

Таким чином, запитуючи дані з Redis можемо отримувати дані за станом платформи. На сторінці моніторингу відображаються такі елементи: індикатор стану платформи. Список пристроїв, що перебувають на тестуванні, посилання на результати тестувань, що пройшли.

Індикатор стану платформи працює наступним чином. Redis запитує останній записаний таймстамп. Таймстамп Redis оновлюється при запиті нових сценаріїв або зміні статусу пристрою (початок тесту, кінець і кілька інших). Оновлення відбувається щонайменше раз на хвилину. Допустимим граничним значенням активності платформи було визначено значення 5 хвилин. Таким чином, якщо останній таймстамп був не пізніше ніж 5 хвилин тому, то вважається, що платформа працездатна.

Список пристроїв дані потрапляють під час виклику методу `get_test_core_list_devices` та збагачуються додатковими полями з таблиць PostgreSQL [11].

Для можливості відстеження статусу тестування було вирішено продати робота в Telegram. Telegram був обраний як популярна платформа-месенджер, з якою можна організувати зручну взаємодію.

Визначимо основні сценарії взаємодії користувача з ботом Telegram.

Публікація роботом інформації в канал тестування про завершення тестування чергового пристрою з результатом. Явний запит користувача у робота інформації щодо стану платформи. Перш ніж розпочинати розробку, бота необхідно зареєструвати та отримати його унікальний id, що є одночасно і токеном. Для цього Telegram існує спеціальний бот - @BotFather [12, 13].

Офіційне API представлено на веб-сторінках Telegram[14]. Пишемо йому /start та отримуємо список усіх його команд.

Перша і головна - /Newbot - відправляємо йому і бот просить придумати ім'я нашому новому боту. Єдине обмеження на ім'я - воно має закінчуватися на "bot". У разі успіху BotFather повертає токен бота та посилання для швидкого додавання бота до контактів. Тут же встановимо боту аватар(/setuserpic), надамо опис(/setabouttext) і вітальне повідомлення(/setdescription). Перевіримо отриманий токен за допомогою api.telegram.org/bot[YourApiToken]/getMe. На цьому етапі створено робот IoTestingBot.

Для надсилання свого повідомлення використовуємо API [https://api.telegram.org/bot\[YourApiToken\]/sendMessage?chat_id=\[ChatId\]&text=\[Text\]](https://api.telegram.org/bot[YourApiToken]/sendMessage?chat_id=[ChatId]&text=[Text]). Метод відправки вимагає токен, який був згенерований при створенні бота, id чату, куди буде відправлено повідомлення та текст повідомлення. Для робота був створений спеціальний канал IoTestingChannel, куди бот зможемо відправляти інформаційні повідомлення. Для надсилання повідомлення необхідно вказати id чату, id чату можемо отримати відправивши будь-яке повідомлення боту всередині каналу. У такому випадку викликавши [https://api.telegram.org/bot\[YourApiToken\]/getUpdates](https://api.telegram.org/bot[YourApiToken]/getUpdates), зможемо побачити прихований id чату.

У разі надсилання користувачеві замість прихованого id використовується ім'я користувача, вказане через @.

Публікація ботом інформації до каналу тестування про завершення тестування чергового пристрою з результатом починається з базового методу get

Id	Name
0	SUCCESS
1	WAITING
2	RETRY
10	FAIL_TIMEOUT
11	FAIL_WRONG_ANSWER
12	FAIL_MQTT_DISCONNECT
-1	INIT

Рисунок 3.11 – TestResultStatuses

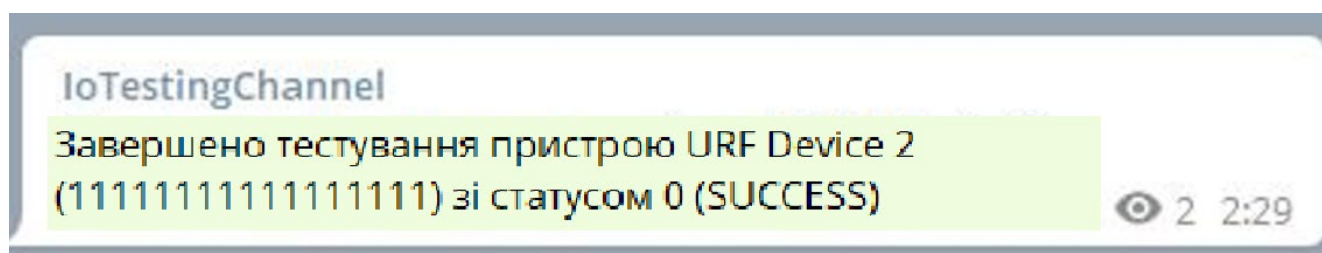


Рисунок 3.12 – Повідомлення, надіслане в канал

При явному запиті користувачем інформації за станом за командою `/status` користувачеві виводиться загальне зведення за кількістю пристроїв, що тестуються, інформація по кожному пристрою у вигляді списку (аналогічно повідомленням в канал).

За додатковою командою `/log` користувачу надсилається лог повідомлень з Redis за замовчуванням за останню годину, або з додатковими параметрами `/log дата_початку час_початку дата_кінця час_кінця`.

Розроблений функціонал дозволяє відстежувати стан платформи через веб-версію клієнтської програми або спілкування з ботом в Telegram.

ВИСНОВКИ

В результаті виконання випускної кваліфікаційної роботи було проведено дослідження конструкторів, порівняльний аналіз веб-технологій Angular, React та Vue, розроблена підсистема взаємодії платформа автоматизованого тестування пристроїв Інтернету речей.

Дослідженні конструкторів було розглянуто особливості сучасних конструкторів, відзначено їх ключові недоліки, що роблять їх використання для вирішення завдання тестування недоцільним.

Аналізі веб-технологій було здійснено повноцінне порівняння трьох популярних сучасних технологій. У порівнянні враховувалася популярність, задоволеність та інші метрики. Також було оцінено продуктивність. Як технологія, що використовується в клієнтській програмі, був обраний найбільш відповідний за вимогами React.

Підсистема взаємодії з платформою включає клієнтський додаток на React, серверний додаток, що працює по API для обміну даними між клієнтським додатком і БД, а також БД PostgreSQL і Redis, що містять інформацію, що використовується в веб-додатку.

Підсистема як частина загального рішення дозволила додатково автоматизувати та спростити роботу тестувальників, а також зробити процес виявлення неполадок ефективнішим. Автоматизація дозволила значно скоротити час випробувань, порівняно з ручним тестуванням.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Козлов А. Тестування прошивок мікроконтролерів AVR// Платформа Medium. 2020. URL: https://medium.com/@anton_oxide/testirovanie-proshivok-mikrokontrollerov-avr-f079045419c, (дата звернення: 01.10.2023).
2. Найпопулярніші фреймворки згідно з опитуваннями Stackoverflow // Платформа Stackoverflow. URL:<https://insights.stackoverflow.com/survey/2023#technology>, вільний (дата звернення: 01.10.2023).
3. Найбільш віддані фреймворки [Електронний ресурс] //Платформа Stackoverflow. URL:доступу:<https://insights.stackoverflow.com/survey/2019#most-loved-dreaded-and-wanted> (дата звернення: 01.10.2023).
4. Кількість скачувань Angular, React та Vue //СервісNPMTrends. URL:<https://www.npmtrends.com/angular-vs-react-vs-vue>, вільний (дата звернення: 01.10.2023).
5. Щорічний звіт про розвиток JavaScript в цілому та його інструментів// Платформа StateOfJS. URL: <https://2022.stateofjs.com/en-US/technologies/front-end-frameworks>, вільний (дата звернення: 01.10.2023).
6. Node.js // URL:<https://nodejs.org/en/>, вільний (дата звернення: 01.10.2023).
7. Документація з фреймворку React. Офіційний сайт React. URL:<https://ua.reactjs.org/docs> (дата звернення: 10.10.2023).
8. Документація MaterialUI [Електронний ресурс] // URL: <https://material-ui.com/getting-started/usage/> (дата звернення: 02.10.2023).
9. Документація Redis / / Сайт СКБД Redis. URL: <https://redis.io/documentation> (дата звернення: 01.10.2023).
10. PostgreSQL URL: <https://ua.wikipedia.org/wiki/PostgreSQL> (дата звернення: 21.10.2023).
11. Боти в Telegram за допомогою BotFather // Документація Telegram.

URL: <https://core.telegram.org/bots#6-botfather/>, (дата звернення: 01.10.2023).

12. API ботів у Telegram // Документація Telegram.
URL: <https://core.telegram.org/bots/api/> (дата звернення: 01.10.2023).

13. Статистика Stackoverflow 2022 // Stackoverflow.
URL: <https://insights.stackoverflow.com/survey/2019#most-loved-dreaded-and-wanted>
(дата звернення: 01.10.2023).

14. Оцінка продуктивності Perfomance Track. URL: <https://perf-track.web.app/> (дата звернення: 11.10.2023).

15. Бенчмарки продуктивності. URL: <https://krausest.github.io/js-framework-benchmark/current.html> (дата звернення: 12.10.2023).