МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА WEB ДОДАТКУ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ VUE.JS»**

Виконала: студентка 2 курсу, групи 8.1218
121 інженерія програмного забезпечення
спеціальності
(шифр і назв1а спеціальності)
інженерія програмного забезпечення
освітньої програми
А.Д. Кисельова
(ініціали та прізвище)
Керівник доцент кафедри програмної інженерії, доцент, к.т.н. Мухін В.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Решевська К.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

« 29 » 05 2019 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Кисельовій Анастасії Дмитрівні
(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) Розробка Web додатку з використанням фреймворку Vue.js

керівник роботи (проекту) Мухін Віталій Вікторович, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 29 » 05 2019 року № 811-с

2. Строк подання студентом роботи

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3.Виконання практичного завдання

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	30.05.2019	
2.	Збір вихідних даних.	20.06.2019	
3.	Обробка методичних та теоретичних	06.07.2019	
	джерел.		
4.	Розробка першого та другого розділу.	03.10.2019	
5.	Розробка третього та четвертого розділу.	05.11.2019	
6.	Оформлення та нормоконтроль	19.12.2019	
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи.	15.01.2020	

Студент _____ А.Д. Кисельова
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.В. Мухін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.В. Кудін
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка web додатку з використанням фреймворку Vue.js»: 51 с., 15 рис., 19 джерел.

НАТИВНИЙ, ПРОГРЕСИВНИЙ ВЕБ ДОДАТОК, ФРЕЙМВОРК, JAVASCRIPT, PWA, SERVICE WORKER, VUE.JS

Об’єкт дослідження - JavaScript фреймворк.

Мета роботи – реалізувати веб-додаток, використовуючи сучасні можливості для нативного відображення додатка на мобільних пристроях.

Методи дослідження – аналіз та узагальнення.

За допомогою покращеної технології для створення прогресивних веб додатків, було знижено період розробки, підвищено якість та продуктивність програмного продукту. Таку технологію можна впровадити у будь-який проект з розробки веб додатку. Вона забезпечить швидке завантаження в незалежності від стану та якості мережі, офлайн використання за відсутності зв’язку або некоректною роботою сервера та гарантує появу свіжого контенту.

Використовуючи готові фреймворки, можна скоротити час та ресурси на розробку PWA, а саме вони представляють собою сучасний підхід до мобільності сайту. Тому, якщо бізнес може отримати нові продажі або інші види прибутку завдяки тому, що додаток завжди під рукою користувача, то однозначно треба спробувати застосувати дану технологію.

SUMMARY

Master’s Qualification Thesis «Developing a Web Application using the Framework Vue.js»: 51 pages, 15 figures, 19 references.

NATIVE, PROGRESSIONAL WEB APP, FRAMEWORK, JAVASCRIPT, PWA, SERVICE WORKER, VUE.JS

The object of the study is Javascript framework.

The aim of study is to implement the web application, using the modern capabilities to natively display the application on mobile devices.

The methods of research are analysis and generalization.

Improved technology for creating advanced web applications has reduced the development time, increased quality and performance of the software. Such technology can be implemented in any web application development project. It will provide fast downloads, regardless of network status and quality, offline use in the absence of communication or improper server operation, and guarantees the appearance of fresh content.

By using frameworks, you can reduce the time and resources available to develop PWAs, which is a modern approach to site mobility. Therefore, if a business can get new sales or other types of profits due to the fact that the application is always at the user's fingertips, then you definitely need to try to apply this technology.

ЗМІСТ

ВСТУП

У сучасному світі, який постійно прагне автоматизувати всі процеси, сфера інформаційних технологій є головним пріоритетом, оскільки вона стосується будь-якого економічного або соціального сектора.

Більшість компаній використовують IT в багатьох департаментах, включаючи відділ кадрів, фінансів, виробництва та безпеки. Ось чому розробники програмного забезпечення зацікавлені у більш ефективній та якісній роботи зі своїм кодом при одночасній економії часу.

В цій роботі представлена розробка системи для автоматизованого написання HTML і JavaScript, а також для підвищення продуктивності коду за допомогою сучасних алгоритмів оптимізації.

Таке рішення є актуальним на сьогоднішній день, оскільки за версією IBM JavaScript посів чільне місце серед кращих мов для вивчення, а також він активно використовується як для клієнтської, так і для серверної частин, допомагає проектувати привабливі інтерфейси, збагачувати веб-додатки численними функціями й змінювати веб-сторінки в реальному часі.

Метою роботи стало розробка програмного продукту з використанням фреймворку Vue.js та реалізація веб-додатка, використовуючи сучасні можливості для нативного відображення на мобільних пристроях.

За допомогою даного програмного рішення можна розробляти як повноцінні сайти, так і функціональні модулі. Значно полегшується розробка SPA-додатків (Single Pages Applications), завдяки можливості здійснювати двостороннє зв'язування для динамічної зміни даних.

Така система відображає чітку структуру програми та реалізується з використанням так званих «паттернів проектування», наприклад MVVM (Model-View-ViewModel).

В ході написання роботи поставлені наступні задачі:

- провести наукове дослідження шляхом аналізу предметної області та огляду сучасної професійної літератури за темою;

- ознайомитись із сучасними можливостями створення веб додатків;

- розробити прогресивний веб додаток для повноцінного використання на мобільних пристроях та підвищення його продуктивності.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження і обґрунтування обраної технології

JavaScript – одна з найпопулярніших мов програмування, використовуваних в інтерфейсі веб-розробки. Як правило, JavaScript використовується в поєднанні з HTML і CSS3 для створення привабливого та інтерактивного дизайну для майбутнього користувача.

З моменту створення Інтернету пройшло багато років і з'явилося багато змін, з'явилося багато технологій, JavaScript, AJAX і інших. Сайти набувають інтерактивність та стають чимось більшим, ніж просто статичні сторінки.

Тепер у розробників досить технологій, щоб змінити поведінку традиційного сайту і зробити його схожим на нативний додаток операційної системи. Для цього всі зміни на сторінці сайту виробляються з використанням скриптів, а запити до сервера використовують AJAX. Коли ви відкриваєте такий сайт, вам потрібно тільки завантажити його, і зміни, які не вимагають нових даних, виконуються швидше, оскільки вони не очікують обробки запитів на сервері. Ця концепція сайту називається односторінковим додатком (Single Page Application). Завдяки цьому, створюється все більше і більше швидких та інтерактивних сайтів.

Як стверджують дослідники Chuan, Y., Wang, H., найпопулярнішою мовою програмування для браузера сьогодні є JavaScript [1]. Через велику кількість функцій, які виконує JavaScript, і різноманітних потреб програмування, а також з метою полегшення роботи веб-програмістів були розроблені JavaScript-frameworks (JFs). Для них важливо вибрати структуру, яка краще відповідає їхнім потребам, пропонуючи високоякісний код і високу продуктивність.

Є кілька інших факторів, які можуть вплинути на ваш вибір, таких як обслуговування, валідність (validity), активна підтримка спільноти тощо. У

цьому сенсі вивчення цієї роботи має гарантувати повну оцінку якості, обґрунтованості та ефективності шести найпопулярніших JF.

Поступово було запропоновано і розроблено багато показників якості програмного забезпечення. Вибір правильних показників має вирішальне значення для точної оцінки. Автори Barkmann, H., Lincke, R., Lowe, W., [2] вивчили і оцінили різні показники якості програмного забезпечення і виявили кореляцію між ними, вказавши, що деякі з них вимірюють одні і ті ж властивості.

В ході дослідження автори розробили інструменти для аналізу продуктивності великої кількості програмних проектів (146 проектів з близько 70 000 класів і інтерфейсів і більше 11 мільйонів рядків коду). Стверджується, що перевірка показників якості програмного забезпечення повинна бути зосереджена на відповідних показниках, тобто пов'язані показники не повинні бути перевірені незалежно.

Виходячи зі статистичної бази, визначається кореляцію між кількома метриками з відомих об'єктно-орієнтованих показників та наводяться ранні результати типових значень показників й можливих порогових значень.

Проте, порівняння JavaScript-структури не є визначеною областю і є порівняно новою в галузі досліджень. Найближчим для проведення паралелі із JavaScript-framework є порівняльна архітектура програмного забезпечення. Але архітектура порівняння програмного забезпечення є теж досить молодою дисципліною, порівняння – це метод аналізу архітектури програмного забезпечення (SAAM), який був заснований в 1996 році (Фернандес-Вілламор, Касільяс, Іглезіас, 2008) [3].

Цей проект був реалізований в Decerno – невеликій компанії, яка займається IT-консалтингом у Швеції, коли було доведено, що вони створюють власні системи для своїх клієнтів. У їх фокусі завжди перебувала розробка веб-сайтів, і власних системи, вони цілком оцінюють інструменти та структури, які вони збираються використовувати, і, що саме вимагає

конкретний проект. Сьогодні їх увага зосереджена на створенні додаткових додатків, що використовують нові та нові системи JavaScript.

Основною проблемою, з якою стикаються веб-розробники, як правило, є вибір відповідного мови або framework для виконання роботи. В епоху веб-розробки JavaScript є найпопулярнішою мовою програмування на стороні клієнта і отримав визнання у всьому співтоваристві веб-розробників. З огляду на це, JavaScript стає все більш цікавим, особливо з появою нових JavaScript Frameworks.

Найчастіше вирішальним фактором у виборі правильного JF для роботи стає обізнаність розробника щодо того чи іншого фреймворку, що за словами Lavanya, Ramachandran, & Mustafa [4], не є раціональною підставою для вибору структури, оскільки вона має тенденцію до суб'єктивності. Проте один із факторів, про який часто забувають, це продуктивність JavaScript-Framework, що є важливим чинником, особливо для підприємств, що розробляють складні програми.

Для веб-розробників важливо вибрати структуру, яка найкраще відповідає їхнім потребам, і одночасно таку, що забезпечує код високої якості та продуктивності.

Тому веб-розробники неохоче адаптують новий Framework до системи, до якої вони вже звикли, оскільки зміна структури означає, що їм потрібно буде виділити час на навчання та розуміння нового функціоналу. Згідно з дослідженням проведеним авторами Gizas, Christodoulou & Papatheodorou [5], розробники нерідко не враховують поліпшень, які надає кожний фреймворк.

Основним результатом їх роботи є те, що вони описують в загальних рисах плюси і мінуси системи управління JavaScript в різних галузях, а також продуктивність найбільш популярних JSF-файлів в різних браузерах і вказують, які проблемні моменти коду можуть бути поліпшені в майбутніх випусках.

1.2 Аналіз успішних технічних рішень

За останні 10 років з'явилося нове сімейство структур JavaScript, що відповідають шаблону архітектури ModelView-Controller (MVC) (або її варіантами).

1.2.1 Vue.js

Vue.js часто називають новим доповненням до ринку користувачами. Незважаючи на те, що структура існує приблизно з 2013 р., вона лише завоювала популярність за останні пару років [6].

Vue став найпопулярнішим Front-end проектом GitHub у 2018 році, який слідував framework 2018 року із 117 тисячами зірочок та понад 29 тисяч форками. Створений Еваном Ю, який працював над численними проектами Angular.js свого часу в Google, Vue.js – це легкий аналог Angular.js, а також виявляється привабливою альтернативою цьому.

Однією з головних причин, чому розробники люблять Vue, це те, що це прогресивний framework. Це означає, що він елегантно адаптується до потреб розробника, починаючи з 3 рядків для керування всім шаром перегляду. VueJS можна швидко інтегрувати в додаток через тег "script", де він поступово починає досліджувати простір.

Але Vue працює особливим чином, оскільки він підбирає найкращі варіанти та можливості, які надають такі framework-и, як Angularjs, Reactjs і Knockout. Це робить його найкращою версією всіх framework-ів, складених у відмінний пакет, який постачається.

Плюси Vue.js:

– розмір. Це одна з найкращих родзинок Vue. Готова конструкція Vue на диво легка – всього 18 кбіт після архівування. Це забезпечує йому дуже потрібну швидкість і точність порівняно з іншими основними framework-ами. У той же час, навіть екосистема навколо Vue невелика і швидка, оскільки

дозволяє користувачам розділяти компілятор шаблону на віртуальний DOM, а також час виконання;

– інтеграційний потенціал. Vue пропонує одну з найкращих можливостей інтеграції, оскільки його можна використовувати для створення односторінкових програм і складних веб-додатків. Невеликі інтерактивні частини основи можна легко інтегрувати з іншими структурами або бібліотеками, такими як Django, Laravel та WordPress;

– масштабованість та універсальність. Ви можете використовувати Vue і як бібліотеку, і як повноцінний framework. Vue може бути легко використаний для розробки великих і багаторазових шаблонів з мінімальними зусиллями, завдяки своїй простій структурі;

– адаптивність. Більшість розробників, які адаптують Vue, переходять на нього з інших основних фреймворків. Період перемикання часто буває швидким через подібність Vue з React та Angular;

– читання. Vue досить легко читати та розуміти для розробників, оскільки функції дуже доступні. Крім того, обробка блоків HTML може бути оптимізована з використанням різних компонентів.

Мінуси Vue.js

– брак ресурсів. Через невелику частку ринку на даний момент, Vue ще належить пройти довгий шлях. Відносний вік також додав клопоту навколо загальних плагінів, що ускладнює роботу із зовнішніми інструментами;

– швидка еволюція. Хоча швидка фаза розвитку корисна для framework-у, яка хороша з точки зору можливостей, крива навчання продовжує змінюватися. Як результат, інтернет-ресурси можуть стати застарілими, а документація іноді може бути єдиним ресурсом для розробників;

– застереження реактивності. Документація чітко зазначає деякі застереження реактивності, такі як безпосередньо встановлення значення елементу з масиву. Це можна зробити за допомогою items[key]=value або

додавання нової властивості даних. Отже, з цим потрібно ефективно впоратися в процесі розвитку.

1.2.2 React.js

React – ще одна популярна бібліотека JavaScript, яка підтримується Facebook. Розроблений та відкритий за допомогою Facebook у 2013 році, він швидко став відомим для розробки величезних веб-додатків, що передбачають динамічну обробку даних [7].

У січні 2019 року Facebook розширив підтримку бібліотеки, перемістивши додаток create-react (інструмент CLI для створення програми React) з інкубації до офіційного репозиторію Facebook.

Один з головних випадків, коли React знаходить широке використання, це коли розробникам потрібно розбивати складні коди та використовувати їх повторно в середовищі без помилок, яка має можливість обробляти дані в реальному часі. Елементи, пов'язані з хуками життєвого циклу, такими як декоратори, допомагають покращити роботу користувачів.

React використовує віртуальний DOM, який допомагає йому інтегруватися з будь-яким додатком. Він також використовує JSX для структурування компонентів і допомагає розробити більш зручні для SEO веб-сторінки порівняно з іншими бібліотеками або структурами JS.

Плюси React.js:

– простота навчання. React демонструє простоту з точки зору свого синтаксису, що передбачає багато навичок написання HTML. На відміну від Angular, який вимагає крутої кривої навчання, щоб володіти TypeScript, залежність React від HTML робить його однією з найпростіших бібліотек JavaScript;

– бібліотека, а не Framework. React – це бібліотека JavaScript замість фреймворку. Це означає, що він забезпечує декларативний метод визначення компонентів інтерфейсу. Його також можна легко інтегрувати з іншими

бібліотеками. Наприклад, бібліотека React-Redux забезпечує необхідну інтеграцію між бібліотеками React і Redux;

– дані та презентація. React забезпечує повне розділення шарів даних та презентації. Хоча він має поняття "стан", його найкраще використовувати для зберігання, яке триває дуже короткий проміжок часу;

– прив'язка до DOM. Розробникам не доводиться переживати прив'язку елементів DOM до функціональності. React вирішує це, розділяючи прив'язку на кілька областей коду лише з обов'язками;

– компоненти для багаторазового використання. Facebook розробив React таким чином, що він забезпечує можливість повторного використання компонентів коду будь-якого рівня в будь-який час. Це економить багато часу під час розробки. Ця здатність головним чином покращує ефективність дизайну та полегшує розробникам можливість оновлення, оскільки будь-яка зміна одного компонента не впливає на інший;

– односпрямований потік даних. React використовує спадну прив'язку даних. Це допомагає переконатися, що будь-які зміни, внесені до структури дитини, не торкнуться їх батьків. Це робить код більш стабільним, і все, що розробник повинен зробити, щоб змінити об'єкт, змінити його стан та застосувати оновлення.

Мінуси React.js:

– темп високого розвитку: середовище React дуже динамічне і постійно змінюється. Це покладає на розробників додаткову відповідальність за те, щоб продовжувати прикладати зусилля та вивчати нові навички. Це може бути головним болем для розробників, які тільки починають вивчати фреймворк;

– JSX як бар'єр: React активно використовує JSX, який є розширенням синтаксису, що дозволяє змішувати HTML з JavaScript. Хоча JSX може захистити код від ін'єкцій (серед інших переваг), він передбачає високий рівень складності та круту криву навчання;

– проблеми з SEO: Користувачі повідомили про проблеми щодо SEO, оскільки пошукові системи погано індексують динамічні веб-сторінки із відображенням на стороні клієнта. Хоча думки переважно спекулятивні, експерти рекомендують розробникам дізнатись, як сканери Google та інших систем відчувають їх.

1.2.3 Angular.js

Angular вважається одним з найпотужніших фреймворків з відкритим кодом. Це рішення "все в одному", яке має на меті забезпечити розробників усіма можливими варіантами. Від маршрутизації до обробки HTTP-запитів та прийняття TypeScript, все вже налаштовано [8].

Хоча крива навчання Angular була досить крутою в попередніх версіях, вона згладилася в версії Angular 5 і 6. Це стало можливим завдяки надійному CLI, який позбавив необхідності знати деталі низького рівня для простих додатків.

Плюси Angular.js:

– документація: Angular поставляється з детальною документацією. Це полегшує новим розробникам вивчати фреймворк, а також зрозуміти код інших проектів порівняно з більш ранніми версіями Angular;

– простіший PWA: прогресивні веб-додатки зросли на популярність, і команда Angular швидко інтегрувала можливості у фреймворку;

– оптимізатор збірки: Оптимізатор збірки видаляє всі непотрібні коди виконання. Це робить додаток легшим та швидшим, оскільки розмір JavaScript у програмі зменшується;

– гачки маршрутизатора: Цикли маршрутизаторів тепер можна відслідковувати від початку до завершення;

– прив’язка даних та MVVM: дозволяється двосторонній зв'язок даних. Це зводить до мінімуму ризик можливих помилок та дає змогу користуватися додатком. З іншого боку, функція Model View View Model дозволяє

розробникам працювати окремо в одному і тому ж розділі програми з тим самим набором даних.

Мінуси Angular.js:

– складний синтаксис: Подібно до першої версії Angular, Angular останньої версії також включає складний набір помилок синтаксису;

– міграція: міграція програми із старих версій Angular може бути проблемою, особливо для масштабних програм;

– хоча фреймворк є досить популярним з точки зору використання, він також пов’язаний з більш крутою кривою навчання порівняно з іншими популярними сьогодні альтернативами.

1.3 Обраний фреймворк

Всі 3 фреймворку сьогодні досить популярні і підтримуються командою розробників. Але з усіх фреймворків, присутніх у списку вище, Vue.js є одним із найлегших для вивчення через його простоту. Цей факт зробив великий внесок у його популярність. Vue.js дозволяє розробникам створювати користувальницькі інтерфейси. Ця відкрита версія JavaScript має функцію адаптованої архітектури, яка дозволяє інтегрувати проект з різними бібліотеками JavaScript [6]. Розробникам потрібно покласти мінімальні зусилля для оптимізації веб-розробок.

Бібліотека досить проста і функціональна. Вимоги до стеку відсутні, тому Vue.JS можна використовувати на будь-якому проекті. Фреймворк зовсім небагато важить.

За допомогою даного програмного рішення можна розробляти як повноцінні сайти, так і функціональні модулі. Значно полегшується розробка SPA-додатків (Single Pages Applications), завдяки можливості здійснювати двостороннє зв'язування для динамічної зміни даних. Така система відображає чітку структуру програми та реалізується з використанням так

званих «паттернів проектування», наприклад MVVM (Model-View-ViewModel).

Односторінковий веб-додаток (SPA) – це концепція веб-сайту, що робить його дуже інтерактивним й дуже схожим на звичайний додаток, зберігаючи при цьому всі переваги веб-сайту. Ідея SPA не нова, вона почала з'являтися як тільки з'явилися JavaScript та AJAX і веб-сайти почали набувати динаміки будучі вже не лише статичними сторінками. Мабуть кожному розробнику сайтів рано чи пізно приходили в голову ідеї про сайт-додаток, який завантажується лише один раз, а все інше відбувається за рахунок скриптів та асинхронних запитів. Проте об'єм роботи, який треба було зробити не був вартий цього, адже для створення веб-сайту за концепцією SPA треба прикласти значних зусиль. Полегшити розробку такого виду сайтів допоможе використання безліч технологій, фреймворків й інструментів, що були для цього розроблені в останній час. Використовуючи ці технології можна будувати привабливі багатофункціональні веб-додатки витрачаючи на це час порівняний з часом, необхідним на розробку традиційного нативного додатку.

2 ПРОГРЕСИВНІ ВЕБ-ДОДАТКИ

Раз на кілька років Інтернет переживає ключовий момент. Момент, коли кілька окремих технологій стискаються разом. Це можуть бути існуючі технології, які існують роками, або новіші, які тільки зараз отримали підтримку браузера. Але зовнішньому спостерігачеві здається, що існує один сяючий момент, коли мережа раптово робить стрибок уперед.

Можна помітити, що це сталося з Ajax, який одного разу вибухнув ніби з нізвідки (незважаючи на більшу частину основних технологій, наприклад, XMLHttpRequest, яка існує довгі роки) і змінила наше уявлення про Інтернет як низку пов'язаних, переважно статичних, сторінок.

Сам Ajax був частиною революції Web 2.0, іншого потужного імені, яке, здавалося б, вирвалося з нізвідки в 2004 році і вибухнуло за ніч.

Через кілька років стратегія mobile-first (нова модель індексації, при якій Google при ранжируванні сайтів, враховує саме мобільну версію сайту) потрапила у центр уваги та сигналізувала, що прийшов час змінити спосіб підходу до веб-розробки.

Такі моменти часто з'являються не тоді, коли народилася технологія, а коли вона названа [11].

В останні роки використання мобільних телефонів зросло до того, що зараз люди проводять вдвічі більше часу на мобільних пристроях, ніж за комп'ютерами чи ноутбуками, а в багатьох країнах мобільні телефони – єдиний пристрій, яким вони користуються.

Прогресивні веб-додатки – це нова порода веб-додатків, яка поєднує в собі переваги нативного додатка і можливості роботи в Інтернеті.

Прогресивні веб-програми виглядають як прості веб-сайти, але, як користувач взаємодіє з ними, вони прогресивно набувають нових повноважень. Вони перетворюються з веб-сайту в щось набагато більше, як традиційний, рідний додаток на мобільному пристрої [11].

У книзі Таль Атер «Building Progressive Web Apps» [11] наводиться хороший приклад того, як прогресивні веб-додатки можуть бути корисними у повсякденному житті: «Уявіть, що прокидаєтесь вранці, берете телефон і відвідуєте веб-сайт місцевої поїзної компанії. Ви швидко перевіряєте розклад поїзда, за допомогою якого можна дістатися до роботи, закриваєте веб-переглядач і кладете телефон назад у кишеню. Наприкінці цього дня ви знову відвідуєте сайт і перевіряєте, коли відправляється наступний поїзд (навіть не помічаючи, що ліфт, яким ви їдете, не має мобільного прийому, тому що сайт компанії поїздів зараз працює навіть у відсутності в мережі). Наступного дня, коли ви знову відвідуєте веб-сайт, ваш веб-переглядач запитає, чи хочете ви додати ярлик до нього на домашньому екрані, і ви раді погодитися. Пізніше того дня, коли ви запускаєте сайт із піктограми на головному екрані, ви отримуєте повідомлення, що через деякі будівельні роботи можливі затримки, і вас запитають, чи бажаєте ви отримувати сповіщення про майбутні зміни на маршруті. Наступного ранку, коли ви прокидаєтесь, ви отримуєте на телефон повідомлення про те, що ваш потяг має затримку на 15 хвилин. Ви натискаєте кнопку затримки на сигналі будильнику і отримуєте ще кілька дорогоцінних моментів сну».

Замість того, щоб намагатися відправити вас у магазин додатків, сподіваючись, що ви встановите їх додаток, компанія поїздів заробила постійне місце на вашому телефоні – крок за кроком. Прогресивні веб-додатки формують довіру до своїх користувачів та набувають нових повноважень у міру необхідності.

2.1 Виникнення прогресивних веб-додатків

Концепція прогресивного веб-додатка (PWA) вперше була введена в 2015 році [9]. PWA – це веб-додаток, який можна використовувати на будь-якому пристрої та в будь-яких умовах мережі. Прогресивні веб-додатки

(PWA) користуються значними успіхами у сучасних веб-браузерах, веб-API та Front-end фреймворках, щоб забезпечити чудовий досвід роботи для мобільних та десктопних користувачів. Великі компанії, такі як Twitter та Pinterest, демонструють, як PWA можуть збільшити участь та використання додатку користувачами та, зрештою, дохід [10].

Прогресивні веб-додатки – це веб-сайти, які використовують сучасні веб-технології для створення мобільних веб-додатків, ближчих до тих, які надають нативні (рідні) програми.

За допомогою посилання на головному екрані користувачі можуть насолоджуватися функціями, які раніше були складними в Інтернеті: доступ до вмісту в режимі офлайн, отримання сповіщень від уподобаних брендів, додавання вибраних на домашній екран. PWA допомагають покращити взаємодію з користувачами та забезпечити безперебійну конверсію – незалежно від електронної комерції, продуктивності праці, публікації, ігор чи засобів масової інформації.

Незважаючи на те, що мобільний телефон – один з головних драйверів технології, настільний комп'ютер – це зростаючий ринок. PWA дають однакові переваги як для користувачів мобільних пристроїв, так і для користувачів планшетів та настільних ПК. І тому незалежно від платформи вони відіграють ключову роль [9].

Коротка історія доставки програмного забезпечення до користувача. Історично склалося, що програмне забезпечення постачалося як додаток. Потрібно було купити його на дискетах, компакт-дисках або накопичувачах пам’яті, встановити його та тільки після цього можна було використовувати. Цей громіздкий процес означав, що програмне забезпечення рідко оновлювалось. Вартість створення фізичних продуктів та розсилки їх була високою, і коли з'являлась помилка у додатку, виправити її було нелегко.

Але завдяки Інтернету, ця процедура змінилась. Замість замовлення фізичних носіїв з’явилась можливість просто завантажити та встановити потрібне програмне забезпечення. З розвитком браузерів та технологій навіть

цей процес спрощувався: замість завантаження та встановлення, можна ввести URL-адресу або натиснули на посилання. Нарешті, було отримано відповідне програмне забезпечення на замовлення.

Поява смартфонів знову підштовхнула до прогресу.

Формат програмного забезпечення в браузері не був ідеальним для мобільних пристроїв – вводити URL-адреси було складно та незручно.

Інтерфейси, які будувалися за допомогою веб-технологій (HTML, CSS та JavaScript), часто оптимізувалися для великих екранів та взаємодії миші. Але головна проблема полягала в тому, що вони вимагали, щоб користувач працював в Інтернеті – а мобільне підключення, як правило, було менш надійним, ніж дротове з'єднання на робочому столі чи ноутбуці.

Рідні програми на iOS та Android надавали те, що не могли мобільні веб-сайти: можливість роботи в автономному режимі та функціонал, оптимізований під можливості смартфона. Отже, програмне забезпечення почали пропонувати встановлювати через спеціальний магазин з додатками.

Але стало зрозуміло, що і це не ідеальне рішення. Поширювати програмне забезпечення кінцевим користувачам через магазин зручно для розробників. Проте це також означає створення декількох версій нативних програм для різних операційних систем та функціоналу, що не кожен може собі дозволити. Розробники покладаються на магазини, щоб публікувати та розповсюджувати додаток, і, можливо, доведеться ділитися прибутками з ними. Постачальник магазину може накласти певні обмеження для нових версій продукту і деякі користувачі не будуть раді неодноразово оновлювати великі додатки. Тому розробникам складніше публікувати нові версії.

Прогресивні веб-додатки використовуються для доставки веб-контенту протягом певного часу. Замість того, щоб очікувати, що користувачі матимуть певний веб-переглядач чи певне обладнання, ми прагнемо поставити щось, що працює для всіх. Більш сучасні системи можуть надавати новіші та складніші функції, але основні функції повинні працювати на будь-якому пристрої.

PWA надаються за посиланням в електронній пошті, чаті, результатах пошуку, QR-коді або з оголошення. PWA забезпечує відповідний інтерфейс для пристрою та функціонал, який одразу доступний і користувач може спробувати його перед покупкою додатка. Поки користувач взаємодіє з PWA, можна перевірити можливість його пристрою та завантажити більше даних у фоновому режимі. Також можна запропонувати розширені функціональні можливості в подальшому – офлайн-використання, сповіщення про наявність нових даних та різні інші зручності [9].

2.2 Як працюють PWA

PWA використовують нові, сучасні веб-API, найбільш важливі Service Workers та маніфести веб-додатків. Хоча ці дві API є відносно новими, вони вже мають повну підтримку в Chrome (та браузерах на базі Chrome, таких як Samsung Internet) та Firefox, та підтримку в бета-версії для Safari та Edge. Ці API-програми дозволяють передавати повнофункціональну програму користувачеві, яка досі не можлива в Інтернеті [10].

Service Workers – найважливіша частина, яка допомагає PWA досягти надійності, швидкості та привабливості для користувачів та разробників. Вони дозволяють PWA працювати в автономному режимі – що традиційно неможливо в Інтернеті – надійно завантажуватись незалежно від стану мережевого підключення користувача, а також включати такі функції, як push-сповіщення та синхронізація фонових даних.

Service Workers також надають тонкий контроль над кешуванням через JavaScript API, дозволяючи створити власну офлайн-програму, пристосовану для певної програми. Наприклад, можно запрограмувати Service Workers завжди спочатку відображати коричтувачеві кешований вміст, зберігаючи швидкість додатка. Але потім оновити кеш у фоновому режимі, щоб наступного разу, коли користувач відкрив PWA, вміст був оновлений.

Маніфести веб-додатків (Web App Manifests) дозволяють прогресивному веб-додатку забезпечити зовнішній вигляд, який люди очікують. Вони дозволяють вказати піктограму, ім’я програми та колір екрана, а також дозволяють користувачам встановлювати цей PWA на свій пристрій та відображати його поряд із нативними (рідними) програмами.

2.3. Важливість прогресивних веб-додатків

Світ додатків є дуже конкурентоспроможним і все більш переповненим. Лише за останні кілька років кількість додатків у магазину додатків збільшилася вдвічі, до більш ніж 2 мільйоні. Вплив цього двоякий: втома програм для користувачів та величезний тиск на розробників додатків та архітекторів підприємств, щоб їх додатки виділялися з натовпу та забезпечували найкращий інтерфейс для користувачів. Додаток, який завантажується більше декількох секунд, або потребує кілька зайвих кліків, щоб знайти та завантажити з магазину додатків, може коштувати втратою користувача.

З позитивного боку, додаток, який легко та швидко надає чудовий вміст, незалежно від того, яким пристроєм користувач користується, може допомогти компанії. Ось чому прогресивні веб-додатки (PWA) швидко стають стандартним способом передачі фунціоналу для різних користувачів в Інтернеті. Насправді, Gartner прогнозує [10], що до 2020 року 50% мобільних додатків загального призначення, орієнтованих на споживачів, будуть PWA.

2.4 Кращі приклади PWA

Великі гравці в технологічній галузі вже випробували силу створення прогресивних веб-додатків. Серед них є такі технічні гіганти [12]:

– Twitter. У 2017 році компанія запустила PWA Twitter Lite. Для написання клієнтської частини PWA та Node.js для розробки на сервері. Розробники використовували React, Redux, Webpack, Babel. Результати впровадження PWA були вражаючими – Twitter скоротив середній час завантаження на 30%. Коефіцієнт відмов знизився на 20%, а показник сторінок за сеанс виріс на 65%;

– Forbes. Світовий медіа-гігант вирішив створити свій власний PWA у березні 2017 року. Основною метою розбудови PWA було посилення залучення користувачів та надання читачам персоналізованого та швидкого веб-досвіду. І вони це здобули – Forbes PWA завантажується за 0,8 секунди, і, отже, компанія збільшила сеанс на кожен показник на 43%, а залученість користувачів на 100%;

– Pinterest. Це яскравий прогресивний приклад веб-додатків. Компанія проаналізувала поведінку користувачів мобільних веб-сайтів і дійшла висновку, що лише 1% користувачів входять у систему або реєструються через свій мобільний веб-додаток. Вони хотіли збільшити цю кількість. І рішення знайшлося – вони зробили PWA протягом 3 місяців. Цим кроком вони збільшили кількість користувачів на 60%, час на 40%, а прибуток від реклами, створеної користувачем, на 44%. Як можна побачити на рисунку 2.1, PWA виглядає схоже в різних браузерах;

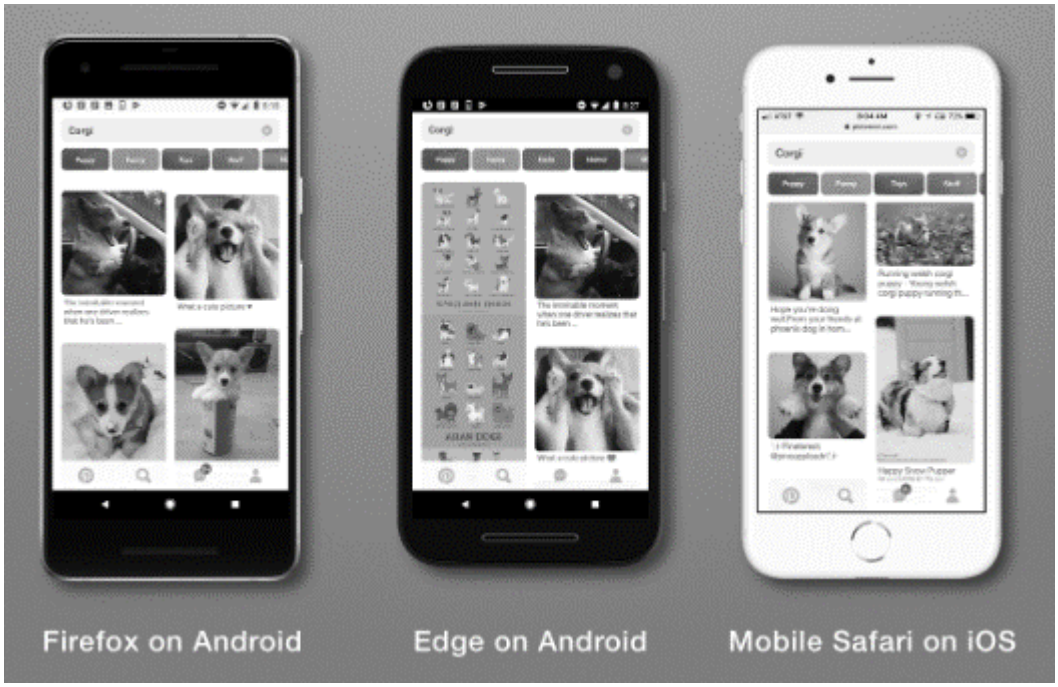

Рисунок 2.1 – Вигляд PWA у різних браузерах

– AliExpress. Це один з найпопулярніших веб-сайтів електронної комерції. Перш ніж створити PWA, AliExpress використовував свій мобільний веб-сайт, щоб запросити веб-користувачів встановити нативну програму. Але ця стратегія спрацювала не так добре, як вважала компанія Alibaba. Компанія вирішила інвестувати в PWA. Незабаром він збільшив конверсію користувачів на 104% і подвоїв сторінку за метрикою сеансу;

– Flipkart – ще один великий сайт електронної комерції. У 2015 році компанія припинила підтримувати мобільний сайт Flipkart і зосередилась на рідному додатку. Тоді компанія вирішила змінити стратегію і розгорнула Flipkart Lite. Як результат, це збільшило конверсію на 70%, повторну участь на 40% та втричі перевищує час, який користувачі витрачають на веб-додаток;

– Tinder. PWA цієї послуги знайомств була створена за допомогою React та Redux. PWA важить на 90% менше, ніж рідний (нативний) додаток. Як результат, PWA завантажується за 4,69 секунди, тоді як нативний додаток завантажується за 11,91 секунди;

– Trivago – цей додаток для бронювання готелів допомагає користувачам обирати найкраще проживання в готелі та сервіс відповідно до їх інтересів та бюджету. PWA Trivago з багатомовною підтримкою доступний для користувачів понад 55 країн. Додаток містить вбудовані функції, включаючи швидкість, офлайн-режим та натискання сповіщень. За допомогою прогресивних веб-додатків Trivago вдалося збільшити залучення користувачів, а частота кліків збільшилась;

– 9Gag – дозволяє користувачам завантажувати, переглядати та обмінюватися вмістом, таким як зображення, відеокліпи та меми, так само, як у рідному мобільному додатку. В основному, функції, які можуть виконувати користувачі, включають перевірку каналів, фільтрування, коментування тощо;

– OLX – створення PWA покращило швидкість та рейтинг кліків на 146%. OLX хотіли, щоб переваги мобільного додатку були у веб-сайту. Вони запровадили такі функції, як push-повідомлення для повторного залучення користувачів та кнопка "Додати до робочого столу".

3 РОЗРОБКА ВЕБ ДОДАТКУ

Для того, щоб скористатися усіма можливостями прогресивних веб додатків, створимо додаток під назвою «KryptoCurrencyTreker». Його функція полягає у відображенні цінових оновлень трьох криптовалют (Bitcoin, Ethereum і Litecoin) в режимі реального часу. Оновлення цін будуть отримані з API Cryptocompare.

KryptoCurrencyTreker також зможе надати потрібну інформацію протягом п'яти днів в минулому і отримувати дані за ці дні.

3.1 Засоби розробки

Для розробки додатка потрібно:

– знати Vue.js;

– встановити Vue CLI, Node та NPM;

– налаштувати Pusher Application.

Для зборки проекту використовувала WebPack. Це один з найбільш потужних та гнучких інструментів для збірки frontend:

– практичний для роботи з односторінковими додатками (SPA);

– сприймає як require () – так і import-синтаксиси модуля;

– дозволяє здійснювати поділ коду;

– Hot Reload для більш швидкої розробки за допомогою React, Vue.js і подібних фреймворків (рис. 3.1).

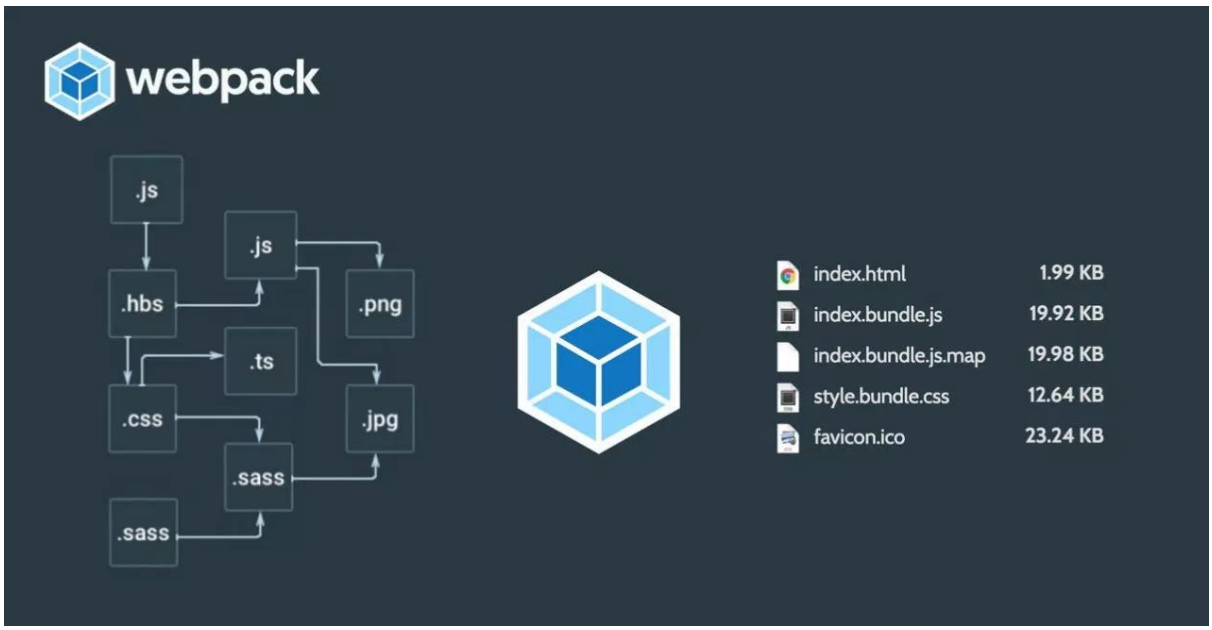

Рисунок 3.1 – Схема збору WebPack

3.2 Архітектура програмного забезпечення

Компонент – це певний об’єкт, який має ресурси та поведінку, він може залежати від інших компонентів або містити їх у собі. Розділення на компоненти – дуже популярний підхід, який використовується у багатьох фреймворках графічного інтерфейсу.

Оскільки наш алгоритм розрахований переважно на SPA, які містить велику кількість скриптів та HTML, то розробка програмного забезпечення відбуватиметься з урахуванням потреби в продуманій структуризації, бо в іншому випадку підтримка та масштабування стануть неможливими. В традиційних сайтах структурної одиницею виступає сторінка. В SPA ж доводиться реалізувати структуризацію іншими способами, зокрема розділенням на компоненти.

3.3 Інструкція користувача

Для початку необхідно встановити node.js, це можна зробити використовуючи офіційний веб-додаток модуля. Запустіть програму встановлення node.js та слідуйте інструкції в програмі установки (рис. 3.2).

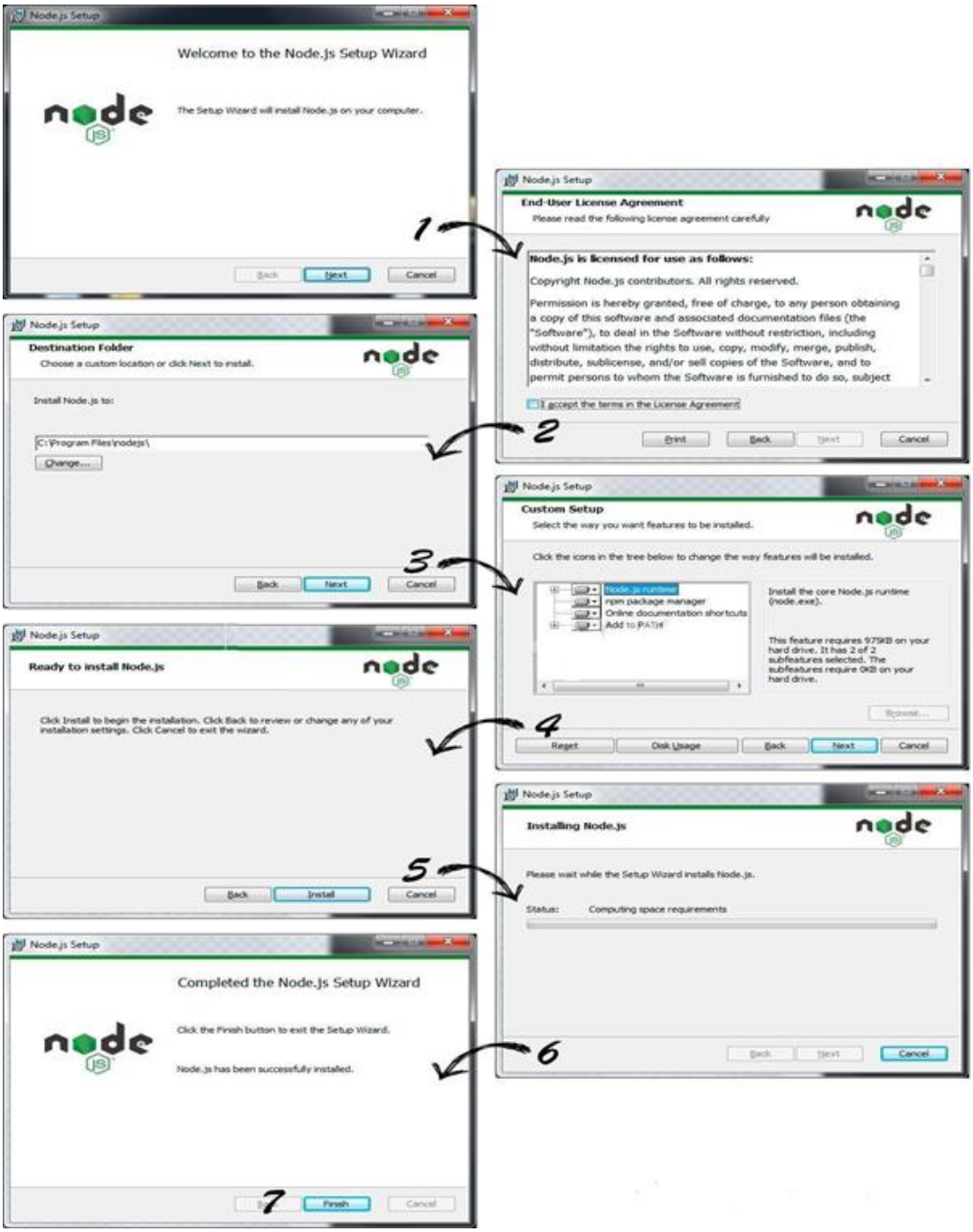

Рисунок 3.2 – Установка Node.js

Після цього необхідно перезавантажити комп’ютер та перевірити коректність встановлення: відкрити командну строку системи та у командному рядку ввести node -v. Якщо результат буде як на рисунку 3.3, то можна переходити до наступного кроку.

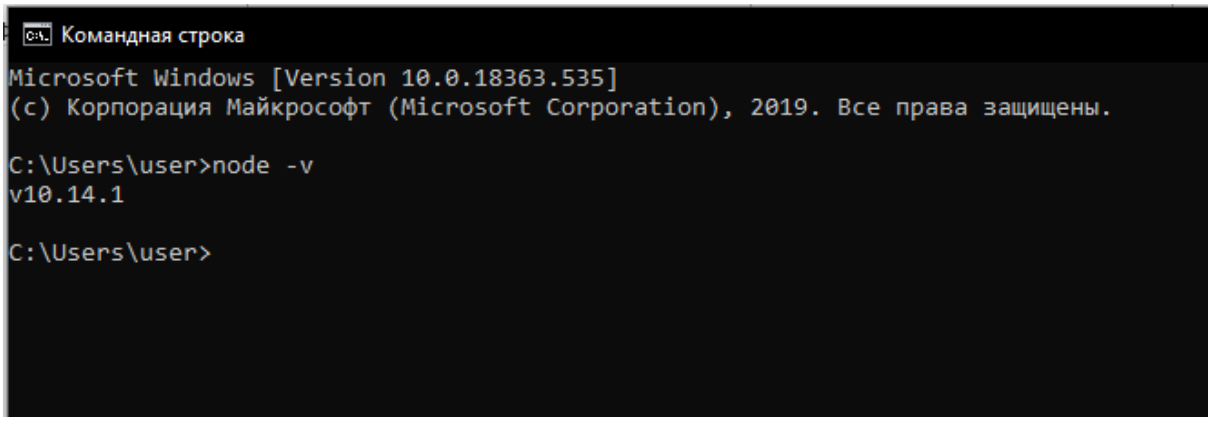

Рисунок 3.3 – Командний рядок

Потім необхідно встановити пакетний менеджер npm та Vue CLI за допомогою команди:

> npm install -g @vue/cli

Після установки з'явиться доступ до vue в командному рядку.

Vue CLI – це полегшений інструмент для створення проектів Vue.js. Щоб почати побудувати прогресивний веб додаток, потрібно використовувати інструмент Vue CLI, щоб витягнути шаблон Vue PWA, з яким ми будемо працювати.

Щоб створити додаток, необхідно виконати наступну команду на своєму терміналі:

> vue init pwa krypto-currency-treker

Шаблон дає нам приголомшливі можливості PWA за замовчуванням. Однією з таких функцій є service worker. Service worker дозволяє нашому додатку працювати в автономному режимі.

3.4 Структура та реалізація додатку

Подібно іншим сучасним JavaScript-бібліотекам і фреймворкам, таким як React, Vue дозволяє створювати компоненти при створенні додатків. Компоненти допомагають зберігати модульність додатку і гарантувати, що додатки можуть бути розділені на модулі багаторазового використання.

Для свого додатку, було створено три компонента багаторазового використання. Компонент Intro який буде містити вступну розмітку і стилі для програми. Current компонент, який буде відображати ціни монет в реальному часі. Previous компонент, який буде відображати ціни монет від «х днів тому».

Компонент Intro не має особливих функцій, оскільки він просто містить вбудовану розмітку і стилі, які зроблять додаток презентабельним. HTML йде між тегами template (рис. 3.4), а стилі йдуть в тезі styles.

```
<template>

  <div>

    <header class="hero">
      <div class="bar logo">
        <h3> KryptoCurrencyTreker</h3>
        <span class="monitor"><span class="monitorText">receive updates</span></span>
      </div>
      <h1>Realtime PWA that displays updates on cryptocurrencies</h1>
      <h2>Bitcoin, Ethereum, Litecoin</h2>
    </header>

  </div>

</template>
```

Рисунок 3.4 – Template для компоненту Intro

У компоненті Current.vue ми напишемо деякий HTML, який відображає ціни в реальному часі в міру їх поновлення. Нижче тегів template у нас буде тег script. Скрипт на рисунку 3.5 вказує props, які повиненні очікувати

компонент Current. Він буде отримувати його, currentCurrency, з батьківського компонента App.vue.

```
<template>
  <div>
    <div>
      <h2>Current prices of coins in $</h2>
    </div>
    <div class="container">
      <div class="row justify-content-md-center">
        <div id="btc" class="shadow-box col col-lg-4 col-md-12"><label>1 BTC</label>
          <p>${{currentCurrency.BTC}}</p></div>
        <div id="eth" class="shadow-box col col-lg-4 col-md-12"><label>1 ETH</label>
          <p>${{currentCurrency.ETH}}</p></div>
        <div id="ltc" class="shadow-box col col-lg-4 col-md-12"><label>1 LTC</label>
          <p>${{currentCurrency.LTC}}</p></div>
      </div>
    </div>
  </div>
</template>

<script>
  export default {
    name: 'app',
    props: {
      currentCurrency: { type: Object }
    },
    data() {
      return {
        pusher: '',
        prices: '',
      }
    }
  }

</script>
```

Рисунок 3.5 – Компонент Current.vue

Компонент Previous повинен показувати ціни на монети в минулому, максимум п'ять днів. Також повинно бути показано дати кожного з днів. У файлі Previous.vue виходить такий код, як на рисунку 3.6.

```
<template>

  <div>
    <h2>Previous prices of coins</h2>
    <div id="first" v-for="day in previousCurrency">
      <h2>Date:  {{day.DATE}}</h2>
      <p><label>1 BTC: </label>{{day.BTC}}</p>
      <p><label>1 ETH: </label>{{day.ETH}}</p>
      <p><label>1 LTC: </label>{{day.LTC}}</p>
    </div>
  </div>

</template>

<script>

  export default {
    name: 'app',

    props: {
      previousCurrency: {
        type: Array
      }
    },

    data() {
      return {}
    }
  }

</script>
```

Рисунок 3.6 – Компонент Previous.vue

У розділі script буде отримуватися об'єкт previousCurrency з батьківського компонента App.vue. Це майже все, що у нас є з трьома компонентами, вони досить прості. Велика частина складності і логіки додатків App.vue в кореневому компоненті App.vue.

Кореневий компонент включений за замовчуванням в кожній новій установці Vue в файлі src / App.vue, тому не потрібно його створювати. На відміну від інших компонентів, які було створено раніше, кореневий компонент містить логіку і складніше, ніж вони.

Тег template кореневого компонента залишається простим. В нього підключаються більш ранні компоненти, Intro.vue, Current.vue і Previous.vue,

щоб користуватись ними, як тегами та передавати відповідні props. На рисунку 3.7 можна побачити тег template та підключення трьох компонентів.

```
<template>
  <div>
    <intro></intro>
    <div id="body">
      <h2></h2>
      <div id="current" class="container">
        <current v-bind:currentCurrency="currentCurrency"></current>
      </div>
      <div id="previous">
        <previous v-bind:previousCurrency="previousCurrency"></previous>
      </div>
    </div>
  </div>
</template>

<script>

  import Intro from './components/Intro.vue';
  import Current from './components/Current.vue';
  import Previous from './components/Previous.vue';
  import Pusher from 'pusher-js'

  export default {

    name: 'app',

    components: {
      current: Current,
      previous: Previous,
      intro: Intro
    },
```

Рисунок 3.7 – Template для App.vue

Оскільки ми отримуємо дані з віддаленого API, нам потрібен HTTP-клієнт, щоб витягнути дані для нас. Для розробки цього веб додатку було використовувано орієнтований на promise HTTP-клієнт axios для відправки HTTP-запитів.

Метод getCurrencyHistory (рис.3.8) є допоміжним методом для збору обмінного курсу монети протягом певного періоду часу і для збереження відповіді в localStorage та this.previousCurrency. Ця інформація буде використовуватися пізніше в коді.

```
getCurrencyHistory: function (days) {
  let self = this;
  for (let i = 1; i >= days; i++) {
    let date = this.$moment().subtract(i, 'days').unix();
    const apiRequest = "https://min-api.cryptocompare.com/data/pricehistorical?fsym=";
    this.axios.all([
      this.axios.get(`${apiRequest}BTC&tsyms=USD&ts=${date}`),
      this.axios.get(`${apiRequest}ETH&tsyms=USD&ts=${date}`),
      this.axios.get(`${apiRequest}LTC&tsyms=USD&ts=${date}`)
    ]).then(this.axios.spread((BTC, ETH, LTC) => {
      let temp = [...self.previousCurrency,
        {
          DATE: this.$moment.unix(date).format("MMMM Do YYYY"),
          BTC: BTC.data.BTC.USD,
          ETH: ETH.data.ETH.USD,
          LTC: LTC.data.LTC.USD
        }];
      self.previousCurrency = temp;
      localStorage.setItem(this.$moment.unix(date).format("MMMM Do YYYY"), JSON.stringify(temp));
    }))
  }
  console.log(self.previousCurrency);
},
```

Рисунок 3.8 Метод getCurrencyHistory

Далі, всередині методу created() кореневого компонента було визначено код для вилучення поточної валюти з localStorage якщо клієнт знаходиться в автономному режимі. Якщо клієнт знаходиться в мережі, він отримує дані з API. Тепер все має працювати, крім функціональності в реальному часі.

```
created() {

  if (!navigator.onLine) {
    this.currentCurrency.BTC = localStorage.getItem('BTC');
    this.currentCurrency.ETH = localStorage.getItem('ETH');
    this.currentCurrency.LTC = localStorage.getItem('LTC');

    for (var i = 1; i <= 30; i++) {
      var date = this.$moment().subtract(30, 'days').unix();
      offlineData = [...this.previousCurrency,
        JSON.parse(localStorage.getItem(this.$moment.unix(date).format("MMMM Do YYYY")))]
    }
  }
  this.pusher = new Pusher('92328e0f11b44cfe765f', {
    cluster: 'eu',
    encrypted: true
  });

  this.prices = this.pusher.subscribe('price-updates');
  this.getCurrencyHistory(30);
  this.axios.get('https://min-api.cryptocompare.com/data/pricemulti?fsyms=BTC,ETH,LTC&tsyms=USD').then((response) => {

    this.currentCurrency.BTC = response.data.BTC.USD,
      localStorage.setItem('BTC', response.data.BTC.USD),
      this.currentCurrency.ETH = response.data.ETH.USD,
      localStorage.setItem('ETH', response.data.ETH.USD),
      this.currentCurrency.LTC = response.data.LTC.USD,
      localStorage.setItem('LTC', response.data.LTC.USD)
  })

},
```

Рисунок 3.9 – Метод created()

Основна концепція PWA полягає в тому, що вона працює однаково швидко з хорошим, поганим або відсутнім підключенням до Інтернету. І ось для цього використовується Service Worker. Це абсолютно окремий мережевий скрипт, який працює на задньому плані і вирішує, чи потрібно завантажувати вміст з кешу або сервера. SW – головна причина, чому прогресивні веб-програми працюють в режимі офлайн і синхронізують дані (рис. 3.10). Сервер HTTPS є основною вимогою для встановлення SW.

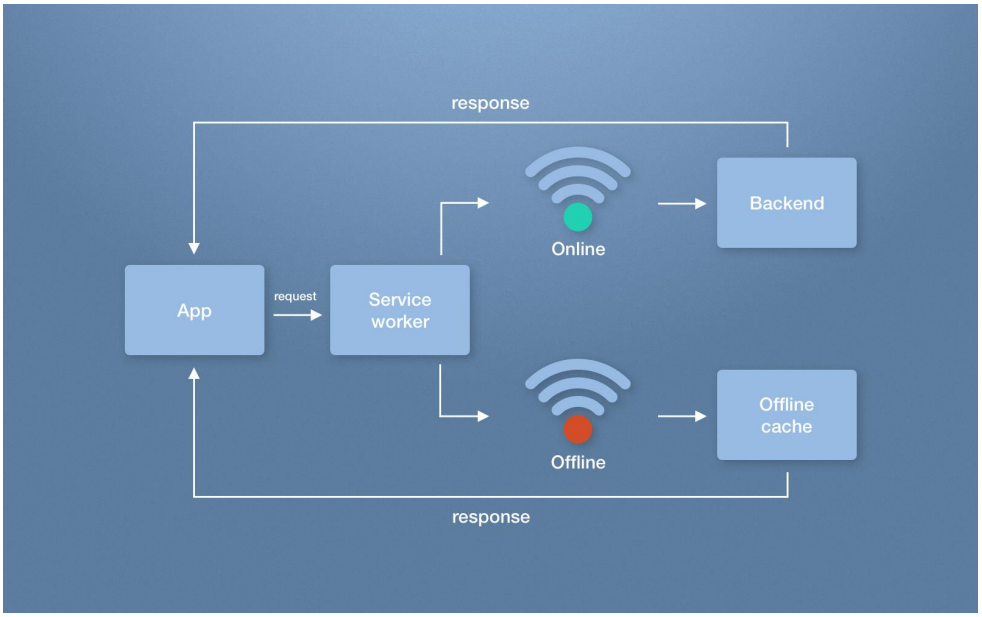

Рисунок 3.10 Приклад роботи Service Worker-а

Все кешування, виконується за допомогою Service Worker, який по суті є нічим іншим, як файлом JavaScript, який знаходиться в додатку, але виконується в окремому процесі [13], а це значить, що його не буде завершено при закритті програми в браузері (або навіть самого браузера). Після реєстрації на головній сторінці програми, Service Worker кеширує статичну інформацію в оболонці програми та починає обробляти запити, надіслані додатком, кешируя і обслуговуючи відповіді на основі логіки, запрограмованої в ній.

Зі сховищ у Service Worker'a є доступ до Cache Storage для web ресурсів, і IndexDB для даних [14]. Але, найголовніше – це повна свобода для реалізації бізнес логіки.

Крім кешування, Service Worker може відправляти повідомлення, push-повідомлення, синхронізувати локальний кеш з віддаленим сховищем даних у фоновому режимі.

Незважаючи на те, що Service Worker дозволяє кешувати всі ресурси сайту майже миттєво після завантаження, перше враження має величезне значення. Якщо перше завантаження займає більше 3 секунд, останнє дослідження DoubleClick [13] показує, що понад 53% всіх користувачів підуть.

Одна з найбільш істотних переваг веб-сайту – майже непомітний вхід – тобто, ніякої установки і майже миттєве завантаження. Користувач завжди на відстані одного кліка. Таку можливість для сайту дає технологія AMP (Accelerated Mobile Pages) – це сторінки з прискореним завантаженням, AMP забезпечує відмінну продуктивність при завантаженні сторінок для користувачів, які переглядають контент в мобільній мережі, що надзвичайно важливо для обмежених або нестійких мереж і швидко відтворює вміст перед користувачами.

Після того як браузер реєструє Service Worker, можна здійснити спробу встановлення. Це трапляється, якщо Service Worker вважається новим у веб-переглядачі, або тому, що на даний момент сайт не має зареєстрованого Service Worker-а, або тому, що між новим Service Worker-ом і раніше встановленим наявним є різниця в байтах.

Інсталяція Service Worker-а ініціює подію установки у встановленому Service Worker-у. Ми можемо включити слухача подій встановлення в Service Worker-і для виконання певного завдання, коли він встановлюється. Наприклад, під час встановлення Service Worker-и можуть попередньо кешувати частини веб-програми, щоб вона завантажувалася миттєво наступного разу, коли користувач відкриє її. Отже, після цього першого

навантаження ви отримаєте користь від миттєвих повторних навантажень, і ваш час на інтерактивність стане ще кращим у цих випадках [15].

Після успішного встановлення Service Worker-а він переходить до етапу активації. Якщо є якісь відкриті сторінки, якими керує попередній Service Worker, новий Service Worker переходить у стан очікування. Новий Service Worker активується лише тоді, коли більше немає завантажених сторінок, які все ще використовують старий Service Worker. Це гарантує, що у будь-який момент часу працює лише одна версія Service Worker-а.

Простого оновлення сторінки недостатньо для передачі контролю новому Service Worker-у, оскільки нова сторінка буде запитана перед завантаженням поточної сторінки, і не буде часу, коли старий Worker не використовує .

Отже, Service Worker може отримувати push-повідомлення від сервера, коли програма не активна. Це дозволяє програмі показувати спонукальні сповіщення користувачеві, навіть якщо воно не відкрите в браузері.

Чи буде отримано сповіщення, коли сам браузер не працює, залежить від того, як браузер інтегрований в ОС [15]. Наприклад, на настільних ОС Chrome і Firefox отримують сповіщення лише тоді, коли браузер працює. Однак Android призначений для пробудження будь-якого веб-переглядача, коли надходить push-повідомлення, і він завжди отримуватиме push-повідомлення незалежно від стану браузера.

На рисунку 3.9 ініціалізується Pusher з обліковими даними програми з панелі інструментів Pusher. У наведеному вище коді ми підписуємося на отримання оновлень на каналі price-updates. Потім ми прив'язуємося до події coin-updates на каналі. Коли подія запускається, ми отримуємо дані і оновлюємо currentCurrency.

Також, одним з найголовніших файлів, який створився при ініціалізації проекта є manifest.json (рис. 3.11).

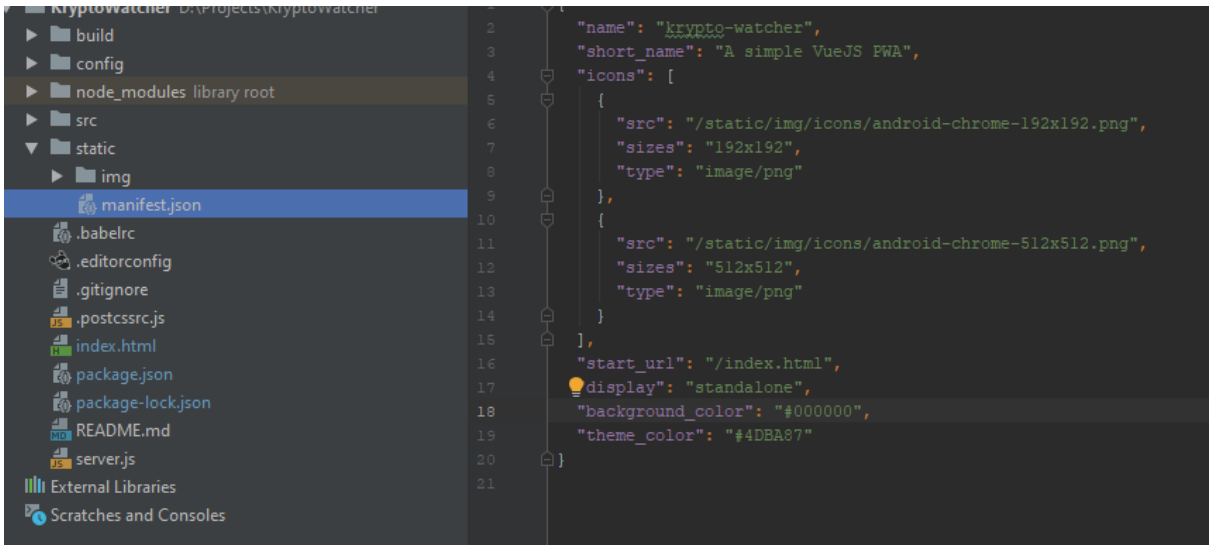

Рисунок 3.11 – manifest.json у структурі проекта

Маніфест веб-додатків надає інформацію про програму (таку як ім'я, авторство, іконку і опис) у форматі JSON-файлу. Мета маніфесту – встановити веб-додаток на домашній екран пристрою, надаючи користувачеві більш швидкий доступ і більше можливостей [16].

Маніфест веб-додатків є частиною колекції веб-технологій, поряд з іншими потужними можливостями, такими як доступ до додатка в режимі офлайн і попередженням користувача за допомогою push-повідомлень про зміну вмісту додатка, разом званими прогресивним веб-додатком. Цей додаток є веб-додатком, який може бути встановлений на домашній екран пристрою, без необхідності для користувача шукати його в магазині додатків.

Додатку потрібна справжня назва або набір назв (які зазвичай зовсім не збігаються з вмістом елемента <title> документа). Для цього використовуються ключі name і short_name у файлу manifest.json.

Ключ short_name служить назвою додатка при відображенні в умовах обмеженого простору (наприклад, під значком на домашньому екрані телефону). Ключ name може бути трохи довшим, відображаючи назву програми повністю [17]. Також він служить додатковою інформацією для користувача, який шукає ваш додаток на телефоні.

Якщо ви опустите назву, то браузер може використовувати <meta name = "application-name"> або елемент <title>.

Але треба бути уважним: деякі браузери можуть вимагати вказати назву – інакше, цей додаток може втратити статус «прогресивний веб-додаток».

Замість звичайної іконки браузера, у веб-додатка повинна бути іконка, яка буде з ним асоціюватися. Для цього в маніфесті є ключ icons. Він приймає список іконок, їх розмірів і форматів. Це робить процес вибору іконки дуже ефективним, оскільки у іконок з'являється адаптивне рішення, яке дозволяє уникнути непотрібних навантажень і допомагає іконкам завжди виглядати відмінно на широкому діапазоні пристроїв і розширення екрану.

Додатки при запуску повинні мати можливість контролювати своє відображення на екрані. Якщо це гра, то їй, ймовірно, потрібно бути в повноекранному режимі і в горизонтальній орієнтації. Для цього формат маніфесту надає два ключа.

Існує декілька режимів відображення: fullscreen (займає весь екран), standalone (відкриває додаток з рядком стану), minimal-ui (коли додаток відображається в повноекранному режимі, як на iOS, але деякі дії можуть викликати панель навігації і появу кнопок назад і вперед) та browser (відкриває додаток зі стандартним набором кнопок і панеллю інструментів).

Плюс такої вказівки орієнтації в тому, що вона виступає в якості «орієнтації за замовчуванням» для всієї програми. Тому, при переході від однієї сторінки до іншої, додаток завжди залишається в правильному положенні. Ви можете змінити орієнтацію за замовчуванням за допомогою API орієнтації екрану.

Іноді під час запуску програми потрібно, щоб користувач завжди потрапляв на певну сторінку. Ключ start_url дає можливість це вказати.

Нативні додатки мають чіткі межі: як користувач, ви впевнені, що коли ви відкриваєте нативний додаток, він несподівано не відчинить інший, непомітно для вас. Найчастіше, вам гранично ясно, що ви переключилися з

одного нативного додатка на інший [17]. Зазвичай ці візуальні підказки надає операційна система (наприклад, виклик диспетчера задач і вибір іншої програми або натискання Cmd+Tab або Alt+Tab на комп'ютері).

З інтернетом все інакше: це величезна гіпертекстова система, в якій веб-додаток може охоплювати кілька доменів: ви можете з легкістю перейти з gmail.com на docs.google.com і користувач навіть цього не помітить. На практиці, ідея існування кордонів додатків є абсолютно чужою для вебу. Адже, насправді, веб-додаток – це просто серія HTML-документів.

В інтернеті ми знаємо, що покидаємо область однієї програми і переходимо до іншої, тільки завдяки веб-дизайнерам, які були досить добрі, щоб розробити їм унікальний та помітний дизайн. У випадках, коли це не так, безліч користувачів виявляються обмануті сайтами, що маскуються під інші.

Формат маніфесту вирішує цю проблему дозволяючи вказувати «область адреси» для вашого застосування. Ця область встановлює межі для застосування. Це може бути або домен, або директорія на цьому домені.

3.5 Результат роботи веб-додатку

Після зборки проекту, можна тестувати його на мобільному пристрої. Для того, щоб мати доступ до локального проекту, я скористалась утилітою ngrok.

Ngrok – це платформа, яка за допомогою встановленої утиліти, дозволяє, організувати віддалений доступ на веб-сервер або якийсь інший сервіс, запущений ПК. Доступ організовується через створений при запуску ngrok безпечний тунель [18].

Після отримання посилання, можна побачити головну сторінку додатка та усі компоненти (рис. 3.12).

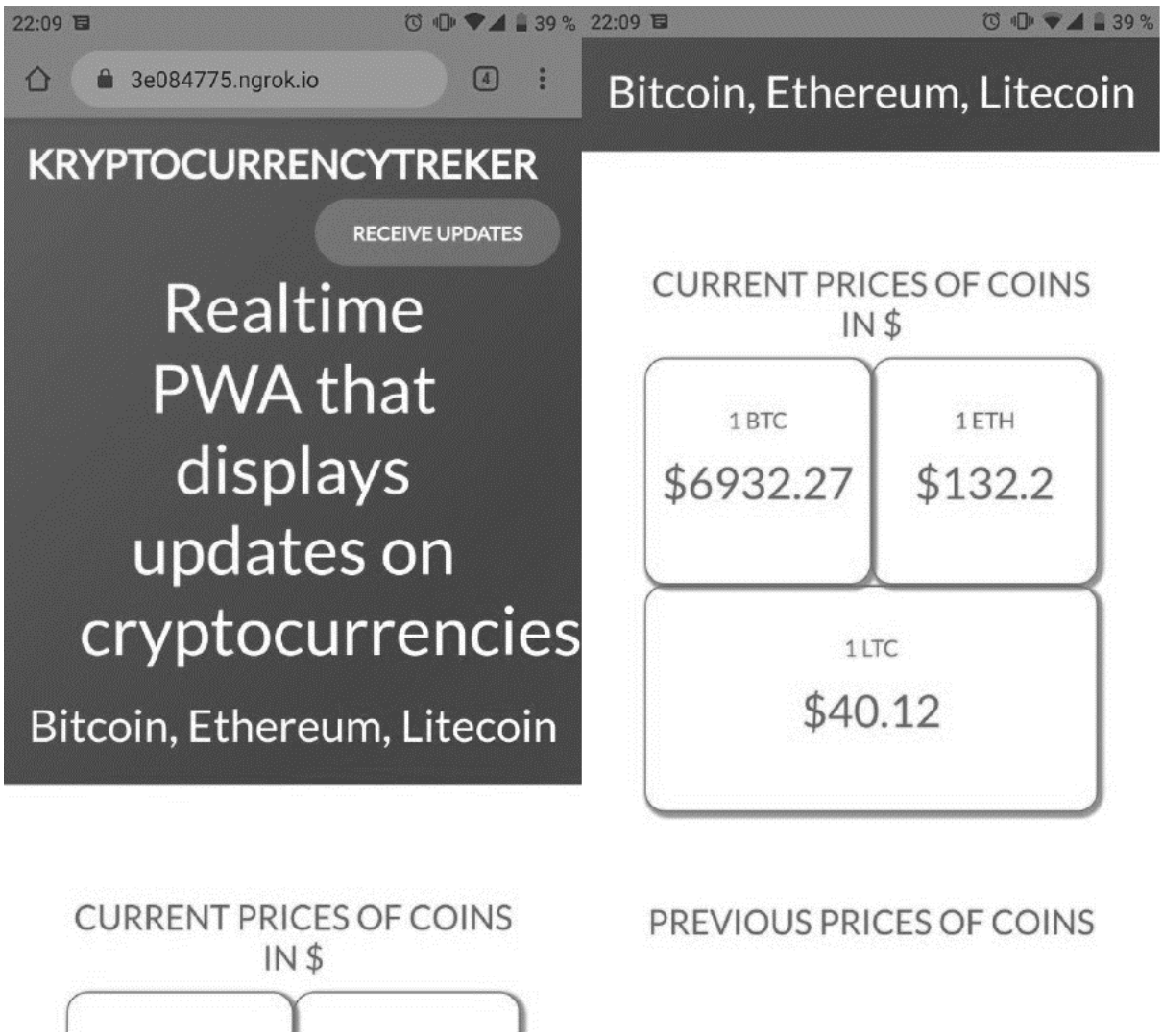

Рисунок 3.12 – Головна сторінка додатка

Знаходячись на цій сторінці та користуючись меню браузера Chrome, можна додати на головний екран посилання на додаток.

Після вибору пункта «Додати на головний екран», з’являється діалог (рис. 3.13) і можна побачити, що значення short_name та іконка з файлу manifest.json використовуються для додавання.

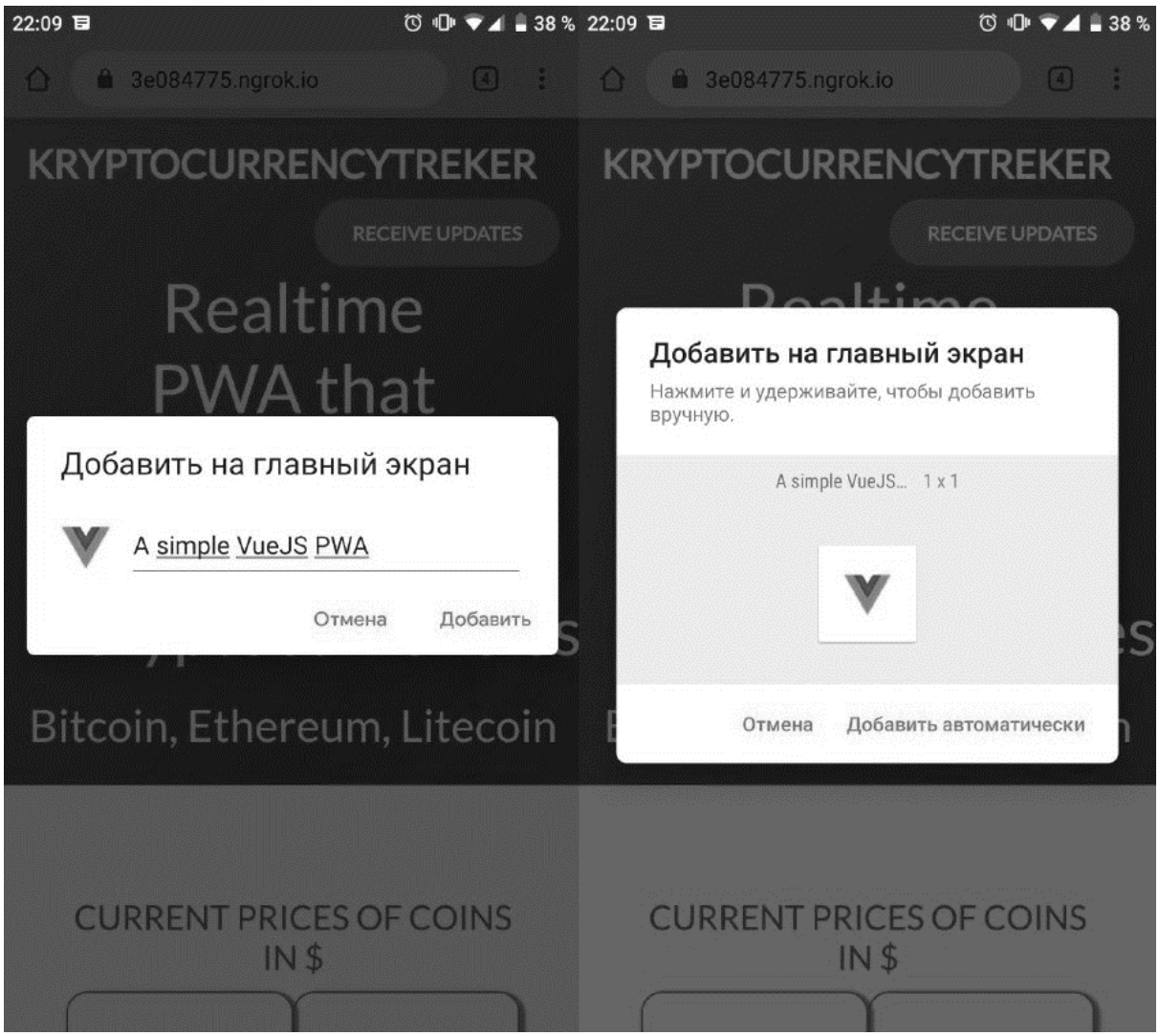

Рисунок 3.13 – Діалог додавання посилання

Як тільки ярлик на робочому столі було створено, користувач отримав доступ до всіх переваг прогресивного веб додатку. Данні біло закешовано, отже було забезпечено офлайн користування додатком. Завдяки Pusher та доступу до мережі, користувач отримує свіжі дані та може спостерігати за зміною цін на криптовалюту.

Після декількох днів користування, у додатку буде відображена історія зміни цін. Це найпростіший і найлегший спосіб бути в курсі поточних цін цих трьох криптовалют.

Для розширення функціоналу та можливостей, можна додати список валют, щоб кожний користувач обирав для себе потрібні показники. А також показувати графіки зміни цін криптовалюти.

Прогресивний веб додаток після запуску через іконку на головному екрані, виглядає, як нативний (рідний) додаток (рис. 3.14) для мобільного телефону.

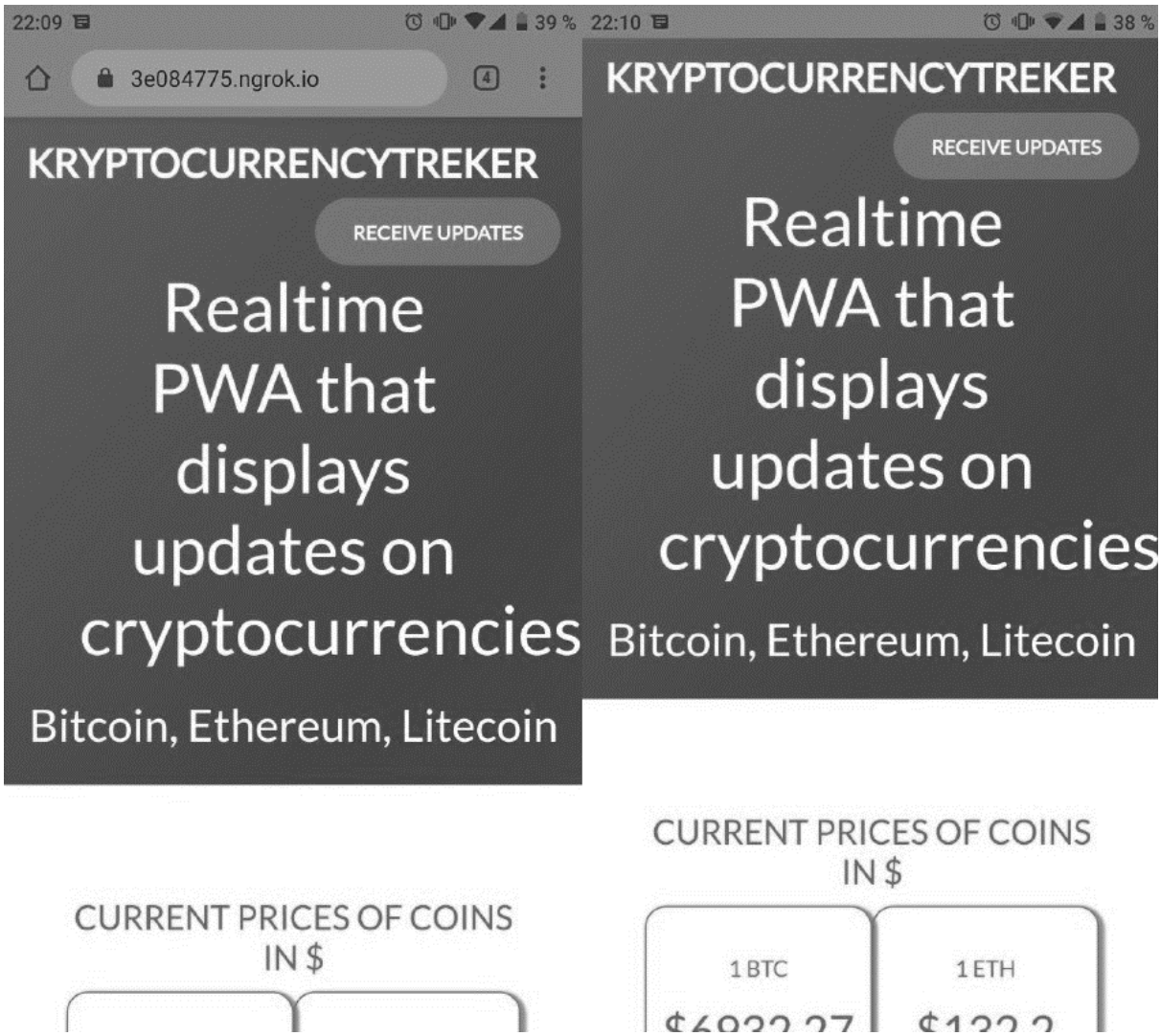

Рисунок 3.14 – Веб додаток у браузері (зліва) та PWA (справа)

Отже, прогресивний веб додаток пропонує безліч переваг в порівнянні зі звичайними додатками. За допомогою PWA можна прискорити процес розробки веб додатку та нативного додатку. Не потрібно реалізовувати два проекти та наймати дві команди спеціалістів. Тому що розробляючи PWA, в результаті отримується кросплатформлений додаток.

Відсутність необхідності встановлення та оновлення, можливість входу на головному екрані пристрою, швидке завантаження в незалежності від

якості мережі, з технологією Service Worker, офлайн використання, завжди свіжий контент, адаптивність і прогресивність, швидкий період розробки та продуманий інтерфейс.

Всі ці переваги прогресивного веб додатку допомагають прискорити процес розробки продукту, призначеного для використання на різних платформах та браузерах.

Впровадження такого додатку надає багато переваг над розробкою мобільного додатка. Це дає можливість користувачу запам'ятати сайт, різниця між закладкою в браузері і іконкою на робочому столі телефону дуже помітна та очевидна. Клієнт може сформувати замовлення, навіть коли знаходиться офлайн, а коли він буде онлайн, це замовлення буде відправлений в магазин. В такій ситуації зі звичайним сайтом власник не отримав би нічого.

Окрім цього, з’являється можливість надсилати Push-повідомлення. Звичайно, людину можна сповіщати і просто по email, але ще один канал активної взаємодії з користувачами зайвим точно не буде.

Такі веб додатки індексуються пошуковими системами, не вимагають великої кількості дій для установки та їх можна подивитися без установки, та дізнатися про функціонал додатку.

ВИСНОВКИ

У даній кваліфікаційній роботі розроблено прогресивний веб додаток для повноцінного використання на мобільних пристроях та підвищення його продуктивності з використанням технології JavaScript.

Першим етапом розробки цього проекту був аналіз предметної області. Було проведено дослідження теоретичного матеріалу у сфері сучасних мов програмування і однієї з важливих для сьогодення задач – кросплатформлена розробка додатків.

Також детально проаналізовано технологію JavaScript, яка за версією IBM посіла чільне місце серед кращих мов для вивчення, а також фреймворків на її основі.

На основі проведених досліджень стало можливим зробити обґрунтований вибір найбільш ефективного для поставленої мети фреймворку – Vue.js та шаблон для створення PWA, який надає інструмент Vue CLI.

В ході роботи була досягнена мета – було реалізовано веб-додаток, використовуючи сучасні можливості для нативного відображення на мобільних пристроях.

За допомогою даного програмного рішення можна розробляти як повноцінні сайти, так і функціональні модулі.

Наукова новизна отриманих результатів полягає в розробці покращеного веб додатку з використанням технології JavaScript та можливостей PWA на основі сучасних JS-фреймворків.

Таким чином усі поставленні завдання даної роботи були виконанні і була досягнута поставлена мета – розробка програмного продукту з використанням нових можливостей фреймворку Vue.js. Також була покращена продуктивність такого додатку за допомогою технологій Service Worker.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Chuan, Y., Wang, H. Characterizing Insecure JavaScript Practices on the Web. In Proceeding *Proceedings of the 18th international conference on World Wide Web*. Madrid, 2009, PP. 964– 965 .

2. Barkmann, H., Lincke, R., Lowe, W. Quantitative evaluation of software quality metrics in open-source projects // *Proceedings of The 2009 IEEE International Workshop on Quantitative Evaluation of large-scale Systems and Technologies (QuEST09)*. Bradford, 2009.

3. Fernández-Villamor, Casillas I. Proceedings of the 2008. *Euro American Conference on Telematics and Information Systems*. Article No. 14.

4. Lavanya Rajendran et. al. *International Journal on Computer Science and Engineering*. Vol. 02, No. 06, 2010.

5. Andreas B. Gizas, Sotiris P. Christodoulou and Theodore S. Comparative Evaluation of JavaScript Frameworks. *Papatheodorou.* Presentation April 16–20, Lyon, 2012.

6. Офіційний сайт Vue.js. URL: https://vuejs.org/v2/guide/ (дата звернення: 14.10.2019).

7. Офіційний сайт React.js. URL: https://reactjs.org/docs/getting-started.html (дата звернення: 14.10.2019).

8. Офіційний сайт Angular.js. URL: https://angular.io/docs (дата звернення:14.10.2019).

9. Google. PWAs the future of the mobile Web / Google, Microsoft, Awwwards, 2019 URL: https://www.awwwards.com/PWA-ebook/ (дата звернення: 17.11.2019).

10. The Architect’s Guide to PWAs // Ionic White Papers. – 2018. URL: https://cdn2.hubspot.net/hubfs/3776657/PWA_WP_v6.pdf (дата звернення: 17.11.2019).

11. Tal Ater. Building Progressive Web Apps. Tal Ater, 2018. 288 p.

12. Liliia H. How to Develop PWA: All You Need To Know Before Creating a Progressive Web App [Електронний ресурс] / Liliia H. // Cleveroad. – 2018. URL: https://www.cleveroad.com/blog/how-to-build-a-progressive-web-app-best-tools-and-examples (дата звернення: 21.11.2019).

13. Сабадышина Т. Progressive Web Apps делают будущее ближе, а Интернет лучше [Електронний ресурс] / Татьяна Сабадышина. – 2017. – URL: https://blog.uamaster.com/progressive-web-apps/ (дата звернення: 21.11.2019).

14. PWA — это просто [Електронний ресурс]. – 2018. URL: https://habr.com/ru/post/418923/. (дата звернення: 23.09.2019).

15. Introduction to Service Worker [Електронний ресурс] // Google Developers Site. URL: https://developers.google.com/web/ilt/pwa/introduction-to-service-worker?hl=ru. (дата звернення: 23.09.2019).

16. Манифест веб-приложения [Електронний ресурс] // MDN URL: https://developer.mozilla.org/ru/docs/Web/%D0%9C%D0%B0%D0%BD%D0%B8%D1%84%D0%B5%D1%81%D1%82.

17. Макеев В. Манифест [Електронний ресурс] / Вадим Макеев. – 2017. URL: https://medium.com/web-standards/%D0%BC%D0%B0%D0%BD%D0%B8%D1%84%D0%B5%D1%81%D1%82-%D0%B0-%D1%87%D1%82%D0%BE-%D0%B7%D0%B0%D1%87%D0%B5%D0%BC-865e609f6f47 (дата звернення: 23.09.2019).

18. NGROK [Електронний ресурс] // SysadminNotes URL: https://sysadmin.pm/ngrok/ (дата звернення: 23.09.2019).

19. List of Top JavaScript Frameworks 2018 – Classic Informatics Blog. URL: https://www.classicinformatics.com/top-javascript-frameworks-2018. (дата звернення: 23.09.2019).