МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІМ. Ю.М. ПОТЕБНІ
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему **Порівняльний аналіз PWA-додатків з нативними Web- і мобільними-додатками.**

Виконав: студент 2 курсу, групи 8.1212-іпз-2
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Інженерія програмного забезпечення
(код і назва освітньої програми)
Б.О. Чорний
(ініціали та прізвище)
Керівник доцент, к.т.н. Полякова Н.П.
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ Дісітел
П.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІМ. Ю.М. ПОТЕБНІ
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

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

ЗАТВЕРДЖУЮ

Завідувач кафедри_____ _Т.В. Критська
“ _01_”__вересня_____2023_року

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

Чорному Богдану Олеговичу
(прізвище, ім’я, по батькові)

1.Тема роботи Порівняльний аналіз PWA-додатків з нативними Web- і мобільними-додатками.

керівник роботи Полякова Н.П., доцент, к.т.н.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від 09.10.2023 р. №1577 -с

2. Строк подання студентом кваліфікаційної роботи 28.11.2023
3. Вихідні дані магістерської роботи
- комплект нормативних документів ;
- технічне завдання до роботи.

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

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
11 слайдів презентації

6. Консультанти розділів магістерської роботи

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

7. Дата видачі завдання 01.09.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	02.09-10.09.23	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	11.09-12.09.23	виконано
3	Аналіз існуючих методів рішення	13.09-14.09.23	виконано
4	Дослідження крос-платформної розробки додатків	15.09-20.09.23	виконано
5	Дослідження прогресивних веб-додатків	21.09-26.09.23	виконано
6	Узгодження подальших дій з науковим керівником	27.09-28.09.23	виконано
7	Розробка дизайну	29.09-13.10.23	виконано
8	Верстка сторінок	14.10-16.10.23	виконано
9	Впровадження PWA в додаток	17.10-19.10.23	виконано
10	Реалізація функціоналу	20.10-09.11.23	виконано
11	Тестування	10.11-17.11.23	виконано
12	Оформлення звіту	18.11-7.12.23	виконано

Студент _____ Чорний Б.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Полякова Н. П.
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

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

АНОТАЦІЯ

Сторінок: 76

Рисунків: 26

Таблиць: 4

Джерел: 17

Чорний Б. О. Порівняльний аналіз PWA-додатків з нативними Web- і мобільними-додатками. : кваліфікаційна робота магістра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник Н. П. Полякова. Запоріжжя : ЗНУ, 2023. 76 с.

Мета і завдання дослідження полягають у вивченні сучасних підходів до розробки PWA-додатків, нативних Web і мобільних додатків, а також у проведенні порівняльного аналізу цих технологій.

У процесі дослідження були розглянуті основні принципи роботи PWA-додатків, нативних Web і мобільних додатків. Було проведено порівняльний аналіз цих технологій за такими параметрами, як продуктивність, доступність, зручність використання та інші.

Проведене дослідження з’ясовує вплив зручності користування додатком як основного фактору, що формує поведінку користувачів. Формує висновки про зручність веб-додатків, на прикладі створення веб-додатку “Планер”.

Ключові слова: PWA-додатки, нативні Web-додатки, мобільні додатки, порівняльний аналіз, продуктивність, доступність, зручність використання.

SUMMARY

Pages: 76

Figures: 26

Tables: 4

Sources: 17

Chornyi B. O. Comparative analysis of PWA applications with native web and mobile applications: qualification work of the master of specialty 121 "Software Engineering" / Science head N. P. Polyakova. Zaporizhzhia : ZNU, 2023. 76 p.

The goal and objectives of the research are to study modern approaches to the development of PWA (Progressive Web App) applications, native web, and mobile applications, as well as to conduct a comparative analysis of these technologies.

During the research, the fundamental principles of PWA (Progressive Web App) applications, native web, and mobile applications were examined. A comparative analysis of these technologies was conducted based on parameters such as performance, accessibility, user-friendliness, and others.

The research investigates the impact of user convenience as a primary factor shaping user behavior. It draws conclusions about the convenience of web applications, illustrated by the example of creating the web application "Planner".

Key words: PWA applications, native web applications, mobile applications, comparative analysis, performance, accessibility, user-friendliness.

ЗМІСТ

ВСТУП

Актуальність теми

Розвиток інформаційних технологій, особливо веб-технологій, створює нові можливості для розробки додатків. Одним з актуальних напрямків є розробка Progressive Web Apps (PWA), нативних веб-додатків та мобільних додатків. Вибір технології для розробки додатків залежить від багатьох факторів, включаючи продуктивність, доступність, зручність використання та інші.

Важливим є проведення порівняльного аналізу цих технологій, що дозволить розробникам вибрати найбільш оптимальний варіант для конкретного проєкту. Такий аналіз допоможе визначити переваги та недоліки кожної технології, а також рекомендації щодо їх використання в залежності від конкретних вимог та обмежень.

Таким чином, актуальним є проведення дослідження, спрямованого на порівняльний аналіз PWA-додатків, нативних веб-додатків та мобільних додатків, що дозволить розробникам вибрати найбільш оптимальний варіант для конкретного проєкту.

Мета і завдання дослідження

Мета і завдання дослідження полягають у вивченні сучасних підходів до розробки PWA-додатків, нативних Web і мобільних додатків, а також у проведенні порівняльного аналізу цих технологій. Це дослідження допоможе визначити переваги та недоліки кожної технології, а також рекомендації щодо їх використання в залежності від конкретних вимог та обмежень.

Об’єкт дослідження

Об’єктом дослідження є PWA-додатки, нативні Web і мобільні додатки.

Предмет дослідження

Предметом дослідження є порівняльний аналіз PWA-додатків, нативних Web і мобільних додатків

Методи дослідження

Для вирішення поставленої задачі використовуються наступні методи дослідження:

1. Аналіз особливостей та існуючих рішень для розробки PWA-додатків, нативних Web і мобільних додатків.
2. Аналіз технологій та фреймворків, що використовуються для розробки PWA-додатків, нативних Web і мобільних додатків.
3. Аналіз особливостей та існуючих рішень для розробки PWA-додатків, нативних Web і мобільних додатків.
4. Аналіз методів тестування продуктивності, доступності та зручності використання PWA-додатків, нативних Web і мобільних додатків.
5. Експериментування з розробленим PWA-додатком для перевірки його продуктивності, доступності та зручності використання.
6. Аналіз результатів експерименту та внесення відповідних змін до розробленого PWA-додатка.

Наукова новизна одержаних результатів

Наукова новизна одержаних результатів дослідження полягає у тому, що для вирішення задачі розробки PWA-додатка був використаний новий та ефективний підхід, а саме: була проведена детальна аналіз та порівняльний аналіз PWA-додатків, нативних Web і мобільних додатків, що допомогло виявити та виправити існуючі проблеми в розробці PWA-додатків. На основі цього дослідження було розроблено новий PWA-додаток, який враховує ці виправлення та покращення.

Практичне значення одержаних результатів

Практичне значення одержаних результатів дослідження полягає у тому, що було розроблено PWA-додаток “Планувальник завдань”, який працює в офлайн-режимі. Цей додаток є оптимальним з точки зору продуктивності та

доступності, оскільки він працює безпосередньо в браузері та не вимагає додаткового завантаження або встановлення. З цього можна зробити висновок, що представлений у роботі підхід може бути використаний в якості ефективного та доступного рішення для розробки подібних PWA-додатків.

Глосарій

PWA (Progressive Web Apps) — це веб-додатки, які використовують сучасні веб-технології та можливості для надання користувачам досвіду, подібного до досвіду роботи з нативними додатками.

Нативні веб-додатки (англ. Confidence) — це додатки, які розроблені спеціально для певної платформи або операційної системи, використовуючи специфічні для цієї платформи мови програмування та інструменти.

Мобільні додатки — це додатки, які розроблені спеціально для роботи на мобільних пристроях, таких як смартфони або планшети.

Vue.js — це прогресивний JavaScript-фреймворк для розробки користувацьких інтерфейсів.

Офлайн-режим — це режим роботи додатка, при якому він продовжує функціонувати, навіть коли відсутнє інтернет-з’єднання. PWA-додатки часто використовують цей режим для забезпечення кращого досвіду користувача.

JavaScript — це високорівнева мова програмування, яка широко використовується для створення динамічних веб-сайтів. Вона визначена як мова сценаріїв (scripting language), і виконується безпосередньо в обладнанні користувача (наприклад, в браузері), що дозволяє реагувати на події та змінювати вміст сторінки без необхідності повного перезавантаження.

React — це бібліотека JavaScript, створена Facebook, яка дозволяє розробникам легко будувати інтерактивні веб-інтерфейси. Вона використовує компоненти та віртуальний DOM для ефективного управління станом додатків та оновлення інтерфейсу в реальному часі.

TypeScript — це мова програмування, яка представляє сучасний синтаксис JavaScript, розширений статичною типізацією. Вона розроблена компанією

Microsoft та є відкритим програмним забезпеченням. TypeScript дозволяє розробникам писати більш безпечний та підтримуваний код, додавши статичну типізацію до JavaScript.

JSON (JavaScript Object Notation) — це легкий формат обміну даними, який є текстовим та незалежним від мови програмування. Виникнув як спосіб передачі структурованих даних між веб-сервером і веб-клієнтом, але широко використовується для обміну даними взагалі.

Material-UI — це популярна бібліотека компонентів для реактивного користувацького інтерфейсу в екосистемі React. Вона базується на дизайн-мові Google Material Design і надає готові до використання React компоненти, які спрощують розробку сучасних та стилізованих веб-додатків.

LocalStorage — це частина Web Storage API, яка дозволяє вам зберігати пари ключ-значення в браузері на довший термін. Збережені дані доступні навіть після закриття вкладки браузера або перезапуску комп'ютера, і вони залишаються в межах того ж самого домену, з якого вони були записані.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРОБЛЕМ PWA ДОДАТКІВ

1.1 Огляд проблем PWA додатків

У дослідженні, опублікованому в журналі “IPTEK The Journal for Technology and Science”, було проведено систематичний огляд застосувань та проблем, пов’язаних з прогресивними веб-додатками (PWA). Дослідження мало на меті встановити знання про практику методу PWA на основі опублікованого дослідження, проблеми, з якими може зіткнутися метод PWA, та підхід методу PWA до експериментальних досліджень. Вони також шукали практики PWA, які можуть вирішити проблеми з попереднім методом [1].

Згідно з дослідженням, було виявлено 31 практику PWA, шість викликів попереднього підходу до розробки мобільних додатків, які були вирішені за допомогою PWA, та сім викликів, пов’язаних з практикою PWA. Зокрема, було виявлено, що PWA вирішує проблеми попередніх методів, хоча це не можна узагальнювати. Однак, незважаючи на свої переваги, PWA все ще має деякі виклики, переважно пов’язані з підтримкою браузера.

Також варто відзначити, що PWA були введені в 2015 році, і з 2017 року дослідження на тему PWA почали зростати. Це свідчить про те, що PWA стають все більш популярними, але вони все ще мають деякі проблеми та виклики, які потребують подальшого дослідження та вирішення.

Інше джерело наводить статистику, яка підкреслює зростаючу популярність та ефективність PWA і порівнює їх з нативними додатками та мобільними веб-сайтами. Зокрема, такі відомі бренди, як Lancôme, AliExpress, Flipkart і Jumia, вже розробили PWA і повідомляють про поліпшення в різних сферах. Lancôme, зокрема, повідомив про 17-відсоткове збільшення коефіцієнта конверсії, 15-відсоткове зменшення коефіцієнта прямої конверсії та 51-відсоткове збільшення кількості мобільних сесій після впровадження PWA.

1.2 Огляд літератури

У 2015 році дизайнер Френсіс Берріман (Frances Berriman) та інженер Google Chrome Алекс Рассел (Alex Russell) ввели термін "прогресивні вебдодатки” для опису додатків, що використовують переваги нових функцій, які підтримуються сучасними браузерами, включаючи Service-Worker та маніфести веб-додатків, які дозволяють користувачам оновлювати їх в рідній операційній системі маніфести додатків, які дозволяють користувачам оновлювати веб-додатки до прогресивних веб-додатків у своїй рідній операційній системі (ОС) [3].

У блозі "Прогресивні веб-додатки: Escaping Tabs Without Losing Our Soul" Рассел пояснює, як вони прийшли до цієї фрази коли вони обговорювали найкращі практики створення веб-додатків. Необхідні технології вже існували деякий час необхідні технології існували вже деякий час, але ніхто не давав їм назву [4].

У своєму блозі Рассел визначає PWA як веб-додаток, який має наступні характеристики:

1. адаптивність: Додаток буде масштабуватися і працювати з будь-яким розміром або формою екрану, наприклад, мобільного пристрою, планшета, телевізора тощо;

2. незалежність від підключення: додаток буде працювати навіть за відсутності;

3. за допомогою Service Worker. Це, звичайно, обмежує доступ до додатку лише вміст, який був завантажений раніше;

4. свіжість: Додаток завжди буде актуальним завдяки тому, що це по суті веб-сторінка, яка може бути отримана знову, коли користувач перезавантажить її;

5. безпечність: додаток буде обслуговуватися через захищене з'єднання HTTPS;

6. відкритість: додаток може бути знайдений пошуковими системами

7. встановлюваність: Додаток може бути доданий на домашній екран користувача;

8. можливість поділитися: Додаток має свій унікальний URL-адресу, щоб ним міг поділитися і отримати доступ будь-хто;

Це перше визначення, яке було дано прогресивним веб-додаткам [5].

1.3 Огляд досліджень

У 2007 році Стів Джобс представив веб-додатки як основний метод розробки додатків для iPhone. Нативні додатки для iOS App Store були випущені роком пізніше. На той час веб-технології були недостатньо розвинені, і ідея підтримки лише веб-додатків не розглядалася, оскільки нативні додатки працювали набагато краще.

У 2015 році Міжнародний союз електрозв'язку підрахував, що до кінця цього року майже половина населення світу - близько 3,2 мільярда людей - користуватиметься інтернетом. Для веб-розробників спроба адаптуватися до всіх цих різних сценаріїв здається щонайменше складним завданням. Саме тут на допомогу приходять прогресивні веб-додатки (PWAS), які дозволяють розробникам створювати швидші, гнучкіші та цікавіші веб-сайти, доступ до яких отримають мільярди людей по всьому світу [6].

У 2014 році кількість користувачів, які виходили в Інтернет з мобільних пристроїв у всьому світі, перевищила кількість користувачів, які виходили в Інтернет зі стаціонарних комп'ютерів.

Згідно з дослідженням, охоплення мобільного інтернету значно перевищує охоплення нативних додатків. Кількість унікальних відвідувачів на місяць становила 11,4 мільйона порівняно з 4 мільйонами для нативних додатків. Водночас, статистика взаємодії користувачів з сервісами показала, що користувачі схильні проводити більше часу в нативних мобільних додатках порівняно зі стандартними веб-додатками. В середньому 188,6 хвилин було проведено в додатках порівняно з 9,3 хвилинами, проведеними онлайн.

Отже, ідея була зрозумілою. Вони хотіли надати користувачам мобільного інтернету такий самий цікавий досвід, як і в мобільному додатку. Таким чином, були розроблені прогресивні веб-додатки.

Результати дослідження, проведеного Fransson у 2017 році, показують, що фотографування за допомогою нативного додатку є значно швидшим порівняно з аналогічним рішенням з використанням PWA.

З іншого боку, додатки, які потребують геолокаційних сервісів, насправді працюють дещо швидше з PWA порівняно з нативним додатком.

Основною перевагою PWA є те, що додаток потрібно закодувати один раз, і він буде працювати практично з будь-яким сучасним браузером на будь-якому пристрої. З нативними додатками доводиться створювати окремі додатки – по одному для кожної платформи, яку вони хочуть підтримувати.

Розробка також відбувається швидше і коштуватиме менше грошей у порівнянні зі створенням нативного додатку. Існує набагато більше людей, які здатні займатися веб-розробкою, ніж тих, хто вміє кодувати нативні додатки для Android або iOS [7].

З іншого боку наскільки добре працюють PWA, залежить від браузера, який використовує користувач. Наприклад, Safari для iOS не підтримує Push API. Це означає, що ці користувачі не можуть отримувати push-сповіщення через PWA. Це одна з причин, чому хтось захоче зробити нативний додаток замість PWA.

Нативні додатки також вважаються більш безпечними з точки зору захисту даних. А недавнє дослідження показує деякі з причин, що стоять за цими твердженнями. Вони з'ясували, що пуш-повідомлення можна легко використовувати для обману людей, видаючи себе за когось іншого відправник пуш-повідомлення може налаштувати зовнішній вигляд повідомлення. Також помітили, що Service Worker може бути використаний для зловживань через те, що він може продовжувати працювати у фоновому режимі.

У таблиці 1.1 наведено порівняння між нативним додатком, PWA та стандартним веб-сайтом за різними важливими параметрами.

Таблиця 1 — *Порівняння нативного додатку, PWA та веб-сайту*

Функції	**Нативний додаток**	**PWA**	**Веб-сайт**
Установка	Необхідно зайти в App store або Play Store, натиснути завантажити	Просто натиснути кнопку, щоб додати їх на головний екран свого телефону	Установка не потрібна
Оновлення	Потрібно зайти в App store або Play Store, після чого завантажити користувачем	Оновлення відбуваються миттєво	Оновлення відбуваються миттєво
Розмір	Займають час для завантаження на пристрої користувача	Невеликі та швидкі	Невеликі та швидкі
Офлайн доступ	Доступний	Потрібно один раз скористатися додатком онлайн, після чого повинна бути можливість доступу до кешованого контент в автономному режимі	Не потрібно
Користувацький досвід (UX)	Добре, коли додаток добре розроблений	Заплутаний через подвійне меню (меню програми і меню браузера)	Те ж саме, що і в прогресивному веб-додатку
Push-повідомлення	Так	Так (тільки для Android)	Так (можливо тільки з сторонніх сервісів)

Деякі функції ще не доступні у веб-додатках. У таблиці 2 наведено функції, які наразі підтримуються в Safari для iOS і Chrome для Android.

Результати показують, що iOS Safari не підтримує Web Bluetooth, Push API та API фонової синхронізації. Ці функції вже працюють у Chrome для Android.

Таблиця 2 — *Функції веб-додатків*

Функції	**iOS Safari**	Chrome для Android
Сервісний працівник (Service Worker)	Так	Так
Push API	Ні	Так
API фонової синхронізації	Ні	Так
File API	Так	Так
MediaStream API	Так	Так
Web Share API	Так	Так

Архітектура оболонки програми підтримується Service Worker, а потім доставляється вміст. Зазвичай вони кешуються з джерела працівниками сервісу за допомогою API-запитів.

Сайти, які люди відвідують частіше, зможуть утримувати останній контент, який людина відвідала, чекаючи, поки мережа динамічно завантажить останнє оновлення. У моделі архітектурної оболонки основна увага приділяється утриманню оболонки інтерфейсу програми та вмісту всередині неї окремо, і вони кешуються окремо. В ідеалі, контент кешується таким чином, щоб він завантажувалася якомога швидше, коли користувач відвідує і повертається пізніше. Теоретично, роздільне завантаження оболонки та вмісту покращує сприйняття користувачем продуктивності і зручність використання програми.

1.4 Технології та цільові показники

Комплексного дослідження прогресивних веб-додатків ще не проводилося. Тому більшість інформації базується на інтернет-джерелах. Так чи

інакше, визначення прогресивного веб-додатку не є зрозумілим для багатьох людей. Визначення варіюються від суто технічних визначень до визначень, які описують нетехнічні аспекти, такі як користувацький досвід.

Прогресивні веб-додатки, досить добре конкурують з нативними додатками. Єдиним основним браузером, який ще не зовсім підтримує всі основні функції PWA, є Safari на iOS. Push-сповіщення поки що не підтримуються Safari, що може призвести до необхідності створення нативного додатку для iOS, хоча інші платформи можуть працювати з PWA як повна заміна нативних додатків.

Наразі відсутні деякі з найважливіших функцій Chrome, такі як доступ до файлової системи та додавання значків бейджів до іконок додатків. У iOS Safari відсутня можливість перегляду контактів користувача. Safari також не підтримує push API, але є цікаве обхідне рішення: Використання push-повідомлень Facebook Messenger. Для цього на пристрої має бути встановлений додаток Facebook Messenger [12].

1.5 Постановка задачі дослідження

Стрімка діджиталізація економічних процесів має значний вплив на функціонування компаній, держав та суспільств. У цих умовах компанії зацікавлені у пошуку нових каналів взаємодії з клієнтами у веб-просторі, щоб отримати конкурентну перевагу та домінувати на ринку.

У цьому контексті особливого значення набуває створення веб-сайтів і мобільних додатків та дослідження новітніх технологій веб-розробки.

Веб-сайти не можуть забезпечити користувачам такий самий користувацький досвід (UX), як нативні додатки (наприклад, офлайн, робота у фоновому режимі, push-сповіщення, тощо).

Незважаючи на значну зацікавленість науковців і дослідників сутністю, особливостями та перевагами розробки PWA, відсутність комплексного підходу до визначених питань стримує широке розповсюдження технології.

Метою є дослідження створення прогресивних веб-додатків, порівняння PWA з нативними додатками й веб-сайтами, а також визначення доцільності використання PWA як уніфікованої технології кросплатформної розробки.

Інтернет зараз буквально скрізь - на годинниках, смартфонах, планшетах, комп'ютерах і автомобілях. Завдяки сучасним браузерам програмне забезпечення, що використовується для відображення веб-контенту, може бути абсолютно однаковим на всіх цих пристроях.

Хоча браузер залишається основним засобом взаємодії користувача, кожен пристрій тепер має власний магазин додатків. Android, Chrome OS, Mac і Windows дозволяють користувачам встановлювати програмне забезпечення, специфічне для їхніх пристроїв, з їхніх власних магазинів.

Суть інстальованого програмного забезпечення полягає в тому, що воно інтегрується з пристроєм так, як очікує користувач.

Предметом дослідження є технологія Progressive Web Application (PWA), яка є різновидом прикладного програмного забезпечення.

Для компаній, що працюють у сфері електронної комерції та інформації, необхідно визначити переваги використання PWA. Яка простота використання таких додатків, який рівень UI/UX, чи є вони менш вибагливими до швидкості та ресурсів, чи працюють без підключення до інтернету, чи допомагають вони компаніям розширити свою мобільну присутність у веб-просторі тощо.

Чи є перевагою нижча вартість розробки та підтримки PWA порівняно з аналогічними нативними аналогами?

Тому слід провести дослідження впливу юзабіліті додатків як ключового фактору, що формує поведінку користувачів. Необхідно визначити, чи слід обирати для використання в бізнесі розширені веб-додатки або нативні додатки. Відповідь на це питання залежить від функцій та можливостей, які бізнес очікує від додатку. Тому, проводячи дослідження, можна зробити деякі висновки щодо юзабіліті веб-додатків на прикладі створення веб-додатку "Планувальник".

1.6 Висновки до розділу 1

У ході дослідження прогресивних веб-додатків (PWA) стало зрозуміло, що ця технологія відіграє важливу роль у сучасній розробці веб-додатків. Огляд літератури показав, що термін "прогресивні веб-додатки" вперше був введений у 2015 році та описує додатки, що використовують переваги нових веб-технологій. Це визначення включає такі характеристики, як адаптивність, незалежність від підключення, застосування Service Worker та інші.

Дослідження підкреслило значущість PWA в порівнянні з нативними додатками, особливо в контексті розробки для мобільних пристроїв. Результати показали, що PWA добре конкурують з нативними додатками, забезпечуючи при цьому універсальність та зручність у використанні.

Однак важливо враховувати, що є деякі технічні обмеження, зокрема, щодо підтримки Safari на iOS. Проблеми з push-сповіщеннями та іншими функціями можуть змушувати розробників обирати нативний підхід для iOS-платформи.

Також, незважаючи на технічні обмеження, дослідження вказує на широкі переваги у вартості та швидкості розробки, які надає використання PWA. Можливість створення додатка один раз, а не для кожної платформи окремо, визначає значний потенціал економії ресурсів та часу.

В цілому, прогресивні веб-додатки визначають новий етап у розвитку веб-розробки, де універсальність та продуктивність стають ключовими факторами вибору для розробників та підприємств. Однак для досягнення повного потенціалу PWA, важливо надалі працювати над покращеннями та усуненням обмежень, щоб забезпечити стабільність та надійність цієї технології.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ ТА АНАЛІЗ РОЗРОБКИ ДОДАТКІВ

2.1 Базовий рівень: Нативна розробка та веб додатки

Природним вибором для розробки додатків є нативний комплект розробки програмного забезпечення (SDK). Зазвичай він надається вендором мобільної платформи. SDK пропонує досвід розробки, адаптований до платформи: Зазвичай є кілька мов програмування. У випадку з iOS, можуть бути використані Objective-C та Swift. На Android нещодавнє додавання мови Kotlin, що входить до набору Javasuperset, було позитивно сприйнято спільнотою розробників, яка в іншому випадку покладається на Java. При розробці для платформи Windows, так званого додатку для універсальної платформи Windows можна використовувати декілька мов, включаючи C++, C#, Visual Basic та JavaScript. Доступ до функцій пристрою можна отримати через власні інтерфейси прикладного програмування (API) платформи [13].

Графічний інтерфейс користувача (GUI) складається з нативних елементів інтерфейсу. Нативний зовнішній вигляд та короткий час реакції є прямим наслідком. Додатки з'являються і з ними можна взаємодіяти так само, як і з додатками для конкретної платформи відповідно до самої мобільної операційної системи. продуктивність є високою, принаймні, для ретельно розроблених додатків.

Функції, впроваджені на мобільних платформах, швидко поширюються на всі сучасні платформи, якщо вони вважаються корисними, але залишаються несумісними між платформами. Коріння несумісності лежить глибоко: не тільки графічні інтерфейси виглядають по-різному (якщо дотримуватися рекомендацій щодо дизайну), але й весь процес розробки відрізняється. Загалом API не розробляються однаково, мови програмування нав'язують певний стиль, а екосистеми платформ, включаючи інструменти та парадигми, відрізняються.

Оскільки розробляти те саме програмне забезпечення, що й нативні додатки, на різних платформах дуже дорого, це призвело до поширення крос-платформних фреймворків для розробки. Створення таких фреймворків ускладнюється тими ж проблемами, про що ми поговоримо в наступному розділі. Сам термін "веб-додаток" є досить розпливчастим. Не існує узгодженого визначення. Загалом, багато додатків, що доставляються через Інтернет і працюють за моделлю клієнт-сервер, вважаються веб-додатками. У літературі, присвяченій мобільним обчисленням, під веб-додатками зазвичай маються на увазі мобільні веб-додатки [14].

Перші веб-додатки використовували лише вбудований браузер платформи. Їм бракувало таких функцій, як публікація в додатках для смартфонів або розповсюдження через магазини додатків. Вони також не мали або мали обмежений доступ до функціональних можливостей конкретних пристроїв і покладалися на загальний веб-вигляд і відчуття. Веб-додатки все ще розробляються з використанням тих самих базових веб-технологій, але власне програмування збагатилося широким спектром фреймворків. Крім того, покращилися функціональність і продуктивність. Браузер-залежні веб-додатки сумісні на всіх платформах, на яких працюють досить потужні браузери. Однак, оскільки різні браузери по-різному реалізують та відповідають специфікаціям CCS HTML та JavaScript, веб-додатки не працюють належним чином у всіх браузерах. Крім того, більшість сучасних браузерів мають можливість відключати JavaScript. Тому ефективне вимкнення логічного рівня, а часто і рівня користувацького інтерфейсу сучасних веб-додатків призведе до того, що додаток відобразить порожню сторінку. До того, як розробка крос-платформних додатків набула широкого розповсюдження, розробникам доводилося робити вибір між нативними додатками та веб-додатками. У "крайніх" випадках рекомендації залишалися незмінними, але для "середніх" вони змінювалися. Високопродуктивні додатки, наближені до апаратного забезпечення, вимагають нативної розробки. Надзвичайно прості додатки на основі форм шви-

дко розробляються як веб-додатки. Краща продуктивність досягається за допомогою HTML5 і сучасного JavaScript, а також можна використовувати ширший спектр можливостей пристрою. Фреймворки (особливо для графічного інтерфейсу та JavaScript) надають багато додаткових можливостей, таких як веб-додатки, що імітують нативний вигляд і відчуття. Як результат, веб-додатки стають дедалі популярнішими. З одного боку, нативна розробка, з іншого боку, веб-додатки продовжують залишатися еталоном для всіх конкурентів. Це стосується не тільки крос-платформних фреймворків розробки, але й інших технологій розробки, навіть якщо вони не є специфічно крос-платформними. Тому, обговорюючи інтегровану розробку, слід порівнювати можливості, широту і якість нативних і універсальних веб-додатків [15].

2.2 Крос-платформна розробка додатків

Кросплатформені підходи до розробки додатків не є чимось новим, проте технологічні можливості з часом змінилися.

Існують базові характеристики, спільні для всіх фреймворків для розробки крос-платформних додатків. Крім того, фреймворки суттєво відрізняються за своїм базовим підходом, парадигмою та сферою застосування.

Очевидно, що вибір - використовувати кросплатформенну розробку чи ні. Якщо ви вирішите не використовувати кросплатформенну розробку додатків, то єдиним варіантом залишиться нативна розробка. При ближчому розгляді вибір трохи більш розмитий: хоча PWA, можливо, досі не вважався кросплатформенним підходом до розробки, концепція полягає в тому, щоб спробувати підтримати кілька платформ з однієї кодової бази (яка, по суті, орієнтована на веб-орієнтованість).

Кросплатформенну розробку можна розділити на дві основні парадигми: середовища виконання та генеративні підходи. В останньому випадку кінцевою метою є створення нативного додатку для кожної підтримуваної платформи. Далі можна розрізняти методи генерації цього нативного додатку з єдиної

кодової бази. Підхід, керований моделлю (позначений на схемі як MDSD: Model-Driven Software Development), спирається на незалежну від платформи модель для створення додатку. Зазвичай для опису програми використовується текстова або графічна мова, специфічна для конкретної галузі (DSL). Транслятори перетворюють (зазвичай нативний) код, написаний для однієї платформи, у нативний код для іншої платформи.

Підхід без генерації додатків спирається на середовище виконання. Замість того, щоб виконувати код безпосередньо на платформі, середовище виконання з'єднує додаток і платформу. Існує кілька технічних реалізацій цієї парадигми. Веб-додатки, які вже називають базовим підходом, просто запускаються у веб-браузері, що надається платформою. Гібридні додатки зазвичай покладаються на веб-технології, але надають нативну упаковку. Традиційно це було зручно, наприклад, з точки зору доступу до специфічної для пристрою функціональності, але зовнішній вигляд цих додатків обмежувався тим, щоб бути максимально наближеним до нативних додатків [16].

Нарешті, додатки, засновані на автономних середовищах виконання, схожі на гібриди, але використовують власні елементи графічного інтерфейсу.

2.3 Прогресивні веб-додатки

PWA є новою концепцією, хоча вона значною мірою спирається на існуючі технологічні артефакти та концепції.

PWA відрізняються від звичайних веб-сайтів, нативних додатків і кросплатформних мобільних додатків. Основна відмінність між звичайним веб-сайтом і PWA полягає в додатковій функціональності та користувацькому досвіді (UX), який пропонує останній. Зі звичайним веб-сайтом практично неможливо працювати в режимі офлайн, оскільки користувачеві доводиться відкривати браузер, вводити URL-адресу і чекати, поки завантажиться весь контент, тоді як з PWA ці дії потрібні лише при першому відвідуванні. Після налаштування домашнього екрану всі статичні файли, необхідні для веб-сайту,

такі як HTML, CSS, JavaScript, зображення і шрифти, зберігаються на телефоні користувача і готові до використання в автономному режимі. Усі динамічні дані кешуються для використання в автономному режимі (або при поганій якості з'єднання) і можуть бути перезавантажені за потреби, наприклад, коли з'являться нові дані і мобільний телефон матиме хороше з'єднання з мережею.

Якщо звичайний веб-сайт обернутий у браузер з видимими артефактами браузера (такими як адресний рядок або меню) (наприклад, Chrome Android), PWA працює аналогічно в екземплярі браузера, але без цих артефактів.

Тому PWA виглядають як звичайні додатки. Якщо PWA правильно стилізовані згідно з рекомендаціями щодо дизайну відповідної мобільної платформи, візуально відрізнити PWA від звичайних нативних або крос-платформних додатків значно складніше.

Якщо порівнювати PWA зі звичайними нативними або крос-платформними мобільними додатками, то однією з ключових особливостей і переваг PWA є найменший слід, який вони залишають на мобільному телефоні користувача. У будь-якому випадку, PWA на два порядки менші за аналогічні нативні додатки [17].

З технічної точки зору, прогресивний веб-додаток - це звичайний веб-сайт, який містить документ метаданих на основі JSON (маніфест), оболонку програми та фонові скрипти (Service Worker), написані на JavaScript. Оболонка програми - це динамічний, залежний від з'єднання та інтероперабельний додаток. Оболонка програми - це мінімальний статичний графічний інтерфейс і логіка, необхідні для представлення програми без динамічного вмісту, залежного від з'єднання. Цей графічний інтерфейс і логіка можуть включати елементи інтерфейсу користувача, а також частини програми, такі як логіка маршрутизації та навігації. Оболонка програми доступна в автономному режимі і завантажується з кешу програми так, що вона відображається миттєво (вимірюється як час до появи першого кольору), що робить програму швидшою, ніж звичайний веб-сайт без PWA. Потім сценарій Service Worker керує всіма

мережевими з'єднаннями, логікою кешу і фоновими завданнями, щоб забезпечити продуктивність в автономному режимі, яку може забезпечити PWA, а проксі-сервер Service Worker може визначити, чи потрібно завантажувати новий контент з веб-служби, чи потрібно завантажувати новий контент з веб-служби або чи є кешований контент все ще актуальним для користувача, уникаючи непотрібних мережевих викликів і навіть відображаючи кешований контент в автономному режимі. Крім того, веб-сайт повинен обслуговуватися за протоколом HTTPS, а його дизайн повинен бути адаптивним.

З точки зору розробника, є багато факторів, які слід враховувати при розробці прогресивних веб-додатків. Наприклад, Google пропагує використання (експериментальної) моделі PRPL (скорочення від Push - Render - Pre-cache - Lazy-load). Ця модель базується на ідеї, що інтернет на мобільних пристроях працює повільно, і що час, необхідний для того, щоб додаток став інтерактивним (вимірюється як час взаємодії), має вирішальне значення для оптимізації. Це досягається за допомогою таких технологій, як централізоване кешування динамічного контенту, новий стандарт HTTP/2 для інтелектуального запиту та отримання статичних файлів з веб-сервера, App Shell та інтелектуальне завантаження контенту. Google також випустив інструмент PWA-тестування для розробників під назвою Lighthouse. Цей інструмент допомагає перевірити, чи є веб-сайт сумісним з PWA, і полегшує розробникам розробку PWA-сайтів.

2.4 Подальші підходи

Межі між крос-платформними додатками, PWA і веб-додатками розмиті. Це можна пояснити відсутністю чіткого визначення та розподілом деяких технічних аспектів. Крім того, деякі підходи не ставлять багатоплатформеність як основну мету, але все ж сприяють багатоплатформенності. Концепція офлайн-сайтів на мобільних телефонах не є новою. Певний час кешування додатків (AppCache) було рішенням для досягнення саме цієї мети.

Однак, з появою Service Workers, AppCache застаріває і замінюється на Service Workers.

Інший альтернативний підхід - Hosted Web Apps (HWA), запропонований компанією Microsoft. Ця концепція, як і Progressive Web Apps, займає проміжне положення між веб- і нативними додатками. Основною перевагою використання HWA перед PWA є їхня тісна інтеграція з платформою Windows. Це дозволяє використовувати такі функції, як виклик власних API Windows з кодової бази HWA JavaScript. HWA також працюють на різних платформах Windows, включаючи ПК, пристрої Windows Phone і такі пристрої, як Xbox. HWA - це дійсно цікавий тип гібридного рішення, оскільки він є веб-орієнтованим і поєднує в собі API платформи та виклики платформи. Цей підхід можна порівняти з Cordova для розробки гібридних мобільних додатків, хоча він обмежений платформами Windows.

Уніфіковані вимоги до розробки можна розглядати з двох точок зору. По-перше, можна ретельно розглянути вимоги до проєктування додатків у багатоплатформенний спосіб, наприклад, щодо графічного інтерфейсу та продуктивності – концептуальний погляд. По-друге, можна розглянути технічні вимоги, наприклад, щодо передумов і сполучних елементів – технологічний погляд [18].

Наступні вимоги є важливими для безперешкодного створення додатків для декількох платформ одночасно:

– користувальницький інтерфейс (UI), який може бути побудований з типових елементів, і який забезпечує різні макети;

– можливість забезпечити потік управління і різні форми взаємодії в додатку;

– зовнішній вигляд, який узгоджується з рекомендаціями щодо дизайну та юзабіліті для відповідних платформ;

– можливість визначення типів даних та забезпечення базових операцій створення, отримання, оновлення та видалення (CRUD), як на пристрої, так і в клієнт-серверній моделі;

– відображення полів форм на модель даних, включаючи перевірку введених даних та можливість збереження даних;

– засоби реагування на вхідні дані, події та зміни стану, а також можливість доступу принаймні до найпоширеніших апаратних засобів.

Крім того, фреймворк повинен мати швидку криву навчання, підтримувати основні платформи та надавати своєчасні результати.

Як показує дослідження PWA, суворою вимогою є достатня довгострокова реалістичність.

Ця оцінка приблизно відповідає передумовам для бізнес-додатків, як це було запропоновано. Базуючись на дослідженні можна виділити декілька технологічних передумов для створення додатків, які охоплюють декілька платформ:

— адекватна підтримка розробки за допомогою середовища розробки та інструментів, а також хороша тестованість додатків;

— достатній ступінь масштабованості;

— продуктивність, близька до нативної;

— хороший рівень супроводжуваності та розширюваності додатків;

— інтерфейси для бізнес-функціональності, такі як підтримка бекенд-систем, і принаймні базовий рівень безпеки, інтегрований у фреймворк.

2.5 Висновки до розділу 2

У світі сучасної розробки додатків ключовою метою стає багатоплатформенність, аби забезпечити широкий охоплення аудиторії та оптимальний користувацький досвід. Крос-платформність та прогресивні веб-додатки (PWA) представляють собою важливі та перспективні підходи до досягнення цих цілей.

Вирізняючись серед різних підходів, крос-платформна розробка дозволяє створювати додатки, які працюють на різних платформах без значних зусиль для розробників. Тут важливо враховувати технічні та концептуальні відмінності між середовищами виконання та генеративними підходами.

Прогресивні веб-додатки (PWA) виявляються ефективним рішенням для поєднання можливостей веб-сайтів та нативних додатків. Забезпечуючи роботу в автономному режимі та поліпшений користувацький досвід, PWA відкривають нові перспективи для розробників та бізнесу.

Межі між крос-платформними додатками, PWA та веб-додатками стають все більше розмитими, виходячи за рамки жорстких визначень. Розширення концепцій, таких як офлайн-сайти, Service Workers та Hosted Web Apps, свідчить про пошук найбільш оптимальних та зручних рішень.

Уніфіковані вимоги до розробки наголошують на важливості адаптивного дизайну, швидкої кривої навчання та технічної ефективності для безперешкодної роботи на декількох платформах.

Загалом, довгостроковий успіх вимагає ретельного підходу до вибору технологій, зосередженості на потребах бізнесу та постійного вдосконалення для відповіді на зростаючі вимоги та очікування користувачів.

РОЗДІЛ 3 ПОРІВНЯЛЬНИЙ АНАЛІЗ , PWA-ДОДАТКУ ТА WEB-САЙТУ

Для того, щоб проаналізувати цю сферу, було проведено порівняння між PWA та нативним додатком веб-сайту "Планер". Для порівняння було обрано сервіс з різними реалізаціями на різних платформах - https://todoist.com. Було визначено дев'ять характеристик, які слід враховувати при порівнянні PWA та нативних додатків.

3.1 Встановлення та пам’ять

Найпомітніша відмінність між PWA та нативними додатками – об'єм пам'яті пристрою, який вони займають.

Це досягається шляхом інтеграції на веб-сайт у мобільному або десктопному браузері без необхідності завантаження. У випадку з веб-сайтами може бути незручно щоразу відкривати браузер. Замість цього є можливість "додати на домашній екран". Таким чином, іконка буде виглядати, як іконка нативного додатку.

У деяких випадках, таких як Twitter Lite, PWA можна встановити з магазинів додатків. Однак це стосується лише Google та Microsoft, тоді як Apple не дозволяє публікувати PWA у своєму магазині додатків.

Це дозволяє PWA скористатися перевагами їхньої малої ваги порівняно з нативними програмами. Більшість PWA мають розмір менше 1 МБ, що є рідкістю для нативних додатків.

3.2 Продуктивність і швидкість

Якщо порівнювати PWA та веб-сайти, то перемога за PWA очевидна. Однак нативні додатки працюють краще:

Порівняння було проведено за допомогою сервісу browserstack та google page speed.

3.2.1 Функціональне тестування

Для перевірки працездатності сервісу були проведені функціональні тести згідно з наступним чек-листом.

1. Встановлення.
2. Видалення.
3. Оновлення версій.
    3.1. Вхід в Play маркет.
    3.2. Пошук додатку.
    3.3. Натискання кнопки оновити.
4. Запуск програми (відображення Splash Screen).
5. Працездатність основного функціоналу додатка.
    5.1. Авторизація (за номером телефону/через соц. мережі/e-mail).
    5.2. Реєстрація (за номером телефону/через соц. мережі/e-mail).
    5.3. Валідація обов'язкових полів.
    5.4. Навігація між розділами додатка.
    5.5. Редагування даних у профілі користувача.
    5.6. Тестування фільтрів.
6. Стресове тестування (не має з'єднання з інтернетом).
7. Скрол/свайп елементів.
8. Тестування PUSH повідомлень.
9. Згортання/розгортання сервісу.
10. Різні типи підключень (мобільний інтернет/Wi-Fi).
11. Робота додатка у фоні.
12. Політики конфіденційності та інші посилання на документи.

Результат функціонального тестування наведено у табл. 3.1.

Також результат тестування наведено у вигляді діаграми на рисунок 3.1.

Таблиця 3 — *Результат функціонального тестування*

Параметри	**Нативний додаток**	**PWA додаток**	**Веб-сайт**
Встановлення	21 с	7 с	Не вимагає встановлення
Видалення	4 с	5 с	Не вимагає видалення
Оновлення	1хв 2с	Автоматичне	Автоматичне
Запуск програми (відображення Splash Screen)	0.8 с	0.7с	2 с
Авторизація (за номером телефону/через соц. мережі/e-mail)	32 с	33 с	40 с
Реєстрація (за номером телефону/через соц. мережі/e-mail)	45 с	48 с	1хв 4с
Додавання елементу	6 с	6 с	9 с
Валідація обов'язкових полів	Ідентично		
Навігація між розділами додатка	Ідентично		
Редагування даних у профілі користувача	Ідентично		
Тестування фільтрів	Ідентично		
Стресове тестування (не має з'єднання з інтернетом)	Безперебійна робота	Безперебійна робота	Робота неможлива
Скрол/свайп елементів	Ідентично		
Тестування PUSH повідомлень	Надсилання PUSH повідомлень без обмежень	Android, Windows – без обмежень IOS – обмежена	Обмежено
Згортання/розгортання сервісу	1хв 1с	0.9 с	1хв 12с
Різні типи підключень (мобільний інтернет/Wi-Fi)	Безперебійна робота	Безперебійна робота	Робота Wi-Fi швидше на 43% ніж мобільний інтернет

Робота додатка у фоні	Займає 80МБ.	Займає 10МБ.	Займає 40МБ.

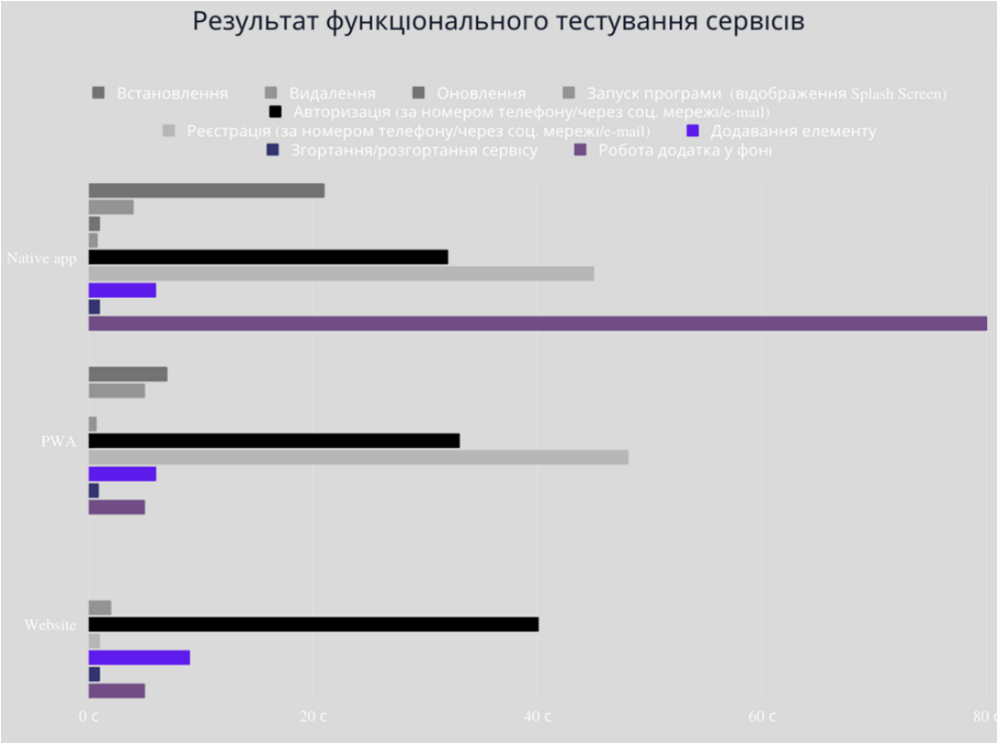

Рисунок. *1 — Діаграма компонентів системи*

3.2.2 Налаштування середовища для впровадження системи

Тестування сумісності використовується для забезпечення сумісності програми з іншими версіями операційних систем, різними оболонками та сторонніми сервісами, а також апаратним забезпеченням пристроїв.

Було перевірено сервіс на тестування переривань, а саме:

1. вхідний дзвінок;
2. смс;
3. Push;
4. будильник;
5. режим "Не турбувати”.

Висновок: сервіси працюють ідентично.

Також перевірено сервіси під час підключення зовнішніх пристроїв:

1. сім-карти;

2. навушників;
3. смарт годинників.

Висновок: сервіси працюють ідентично.

3.2.3 Стресове тестування

Стрес-тестування має на меті визначити ефективність роботи програми під підвищеним навантаженням. У цьому контексті стрес-тестування фокусується лише на локальних і PWA-додатках.

Було перевірено поведінку сервісу при таких умовах:

1. високе завантаження центрального процесора.
2. брак пам'яті.
3. завантаження батареї.
4. низька пропускна здатність мережі.
5. велика кількість взаємодій користувача з додатком (штучно).

Результат стресового тестування наведено у таблиці 3.2.

Таблиця 4 — *Результат стресового тестування*

Параметри	**Нативний додаток**	**PWA додаток**	**Веб-сайт**
Високе завантаження центрального процесора	Показник подає на 74%	Показник подає на 72%	Безперебійна робота
Брак пам’яті	Показник подає на 80%	Показник подає на 74%	Безперебійна робота
Низький заряд батареї	Показник подає на 87%	Показник подає на 86%	Показник подає на 90%
Запуск програми (відображення Splash Screen)	Безперебійна робота	Безперебійна робота	Показник подає на 90%
Авторизація (за номером телефону/через соц. мережі/e-mail)	Безперебійна робота	Безперебійна робота	Показник подає на 90%

Результат стресового тестування також наведено у вигляді діаграми по кожному з параметрів на рисунку 2 — 6.

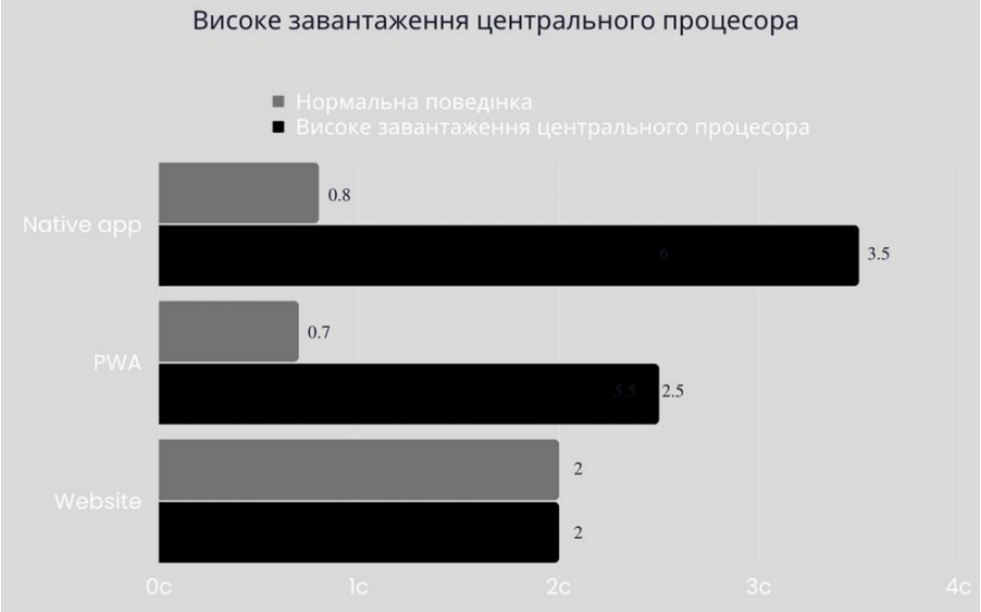

Рисунок *2* — *Результат тестування на високе завантаження центрального процесора*

Рисунок *3* — *Результат тестування на брак пам'яті*

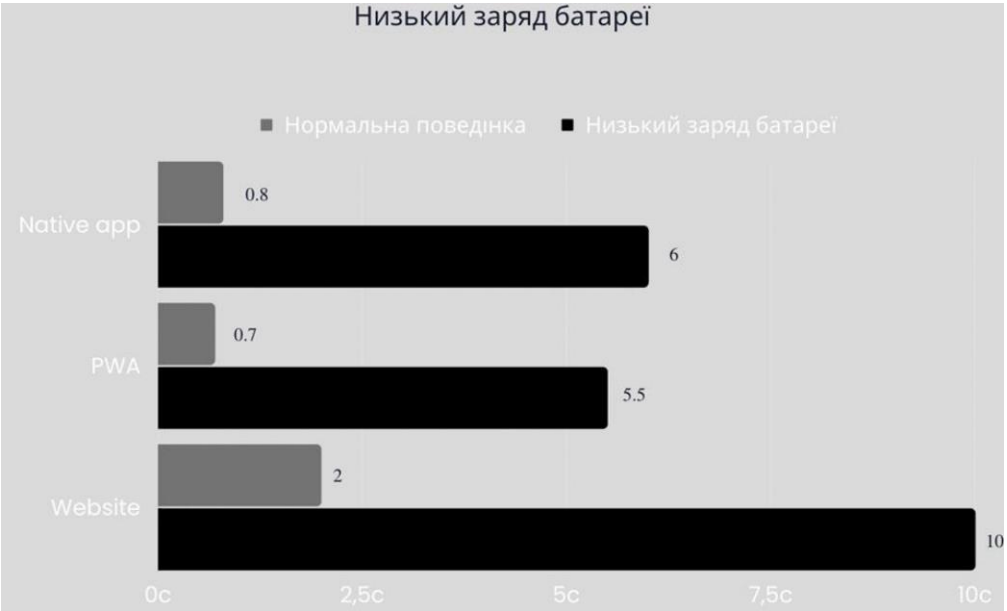

Рисунок *4* — *Результат тестування на низький заряд батареї*

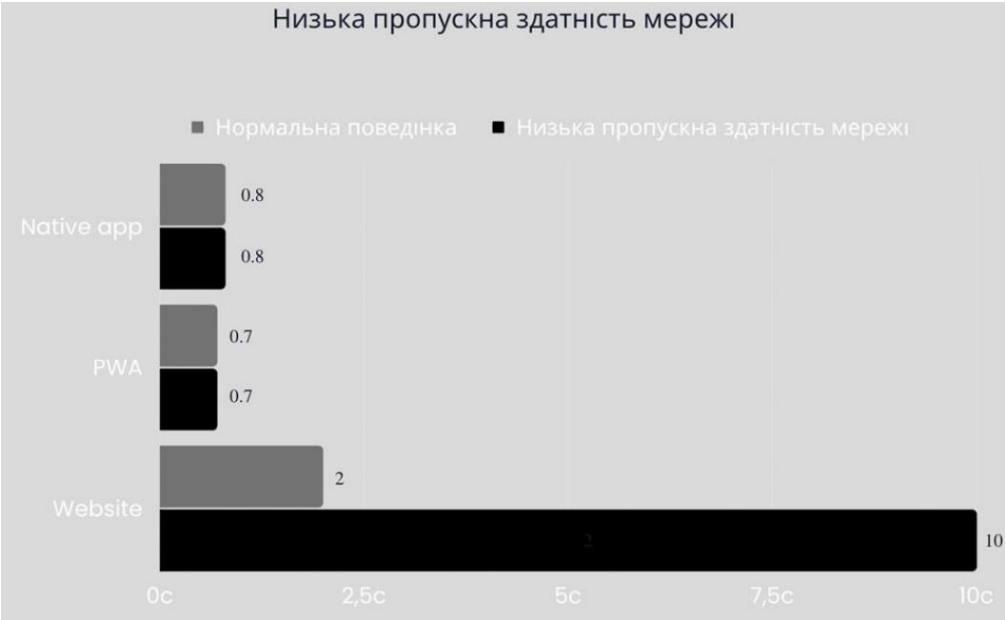

Рисунок *5* — *Результат тестування на низьку пропускну здатність мережі*

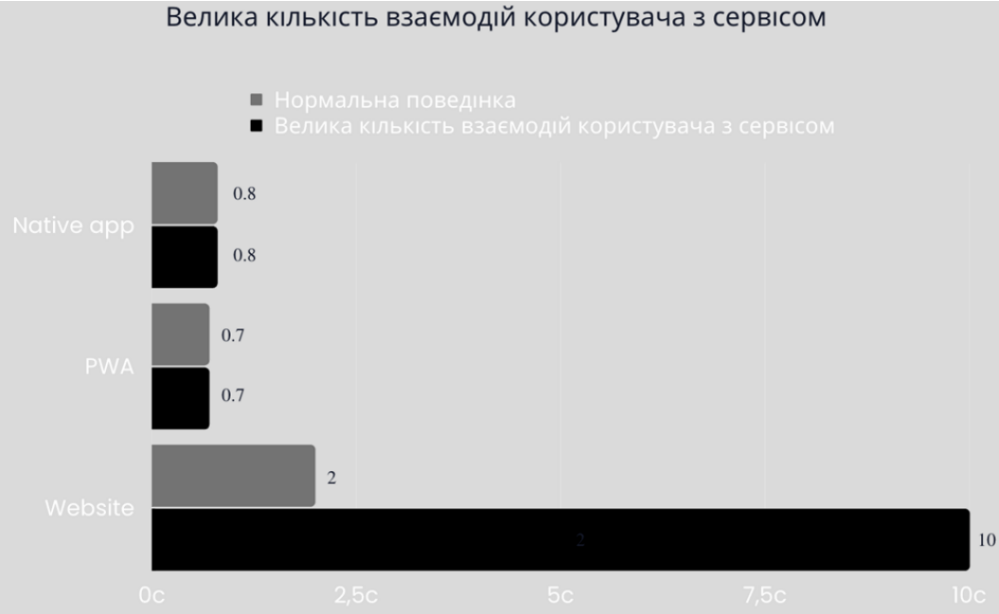

Рисунок *6* — *Результат тестування на велику кількість взаємодій користувача з сервісом*

Результат автоматизованого тестування продуктивності сервісів можна побачити на рисунку 7-8.

Продуктивність веб-сайту складає 61%, що являється не критично.

Кожен PWA базується на Service Worker. Ці скрипти працюють у фоновому режимі і не залежать від веб-сторінки. PWA включають розширене кешування, GraphQL та інші методи для прискорення роботи веб-сайту. Це забезпечує швидкість, порівнянну зі швидкістю нативних додатків.

Оскільки PWA - це все ще веб-сайт, він залежить від усіх елементів мобільного веб-сайту, таких як швидкість сторінки, час відгуку сервера та рендеринг CSS на критичному шляху.

Однак PWA працюють через браузер. Як наслідок, вони мають більшу затримку та енергоспоживання, ніж нативні додатки. Нативні програми можна інтегрувати в операційну систему, щоб вони могли отримати доступ до апаратного забезпечення пристрою для виконання більшої кількості обчислень і забезпечення кращого користувацького досвіду.

Тому з точки зору швидкості нативні додатки виграють, хоча і з невеликим відривом. Нативні додатки потужніші, а нативний код на Kotlin/Swift швидший.

У даному випадку така різниця не є критичною.

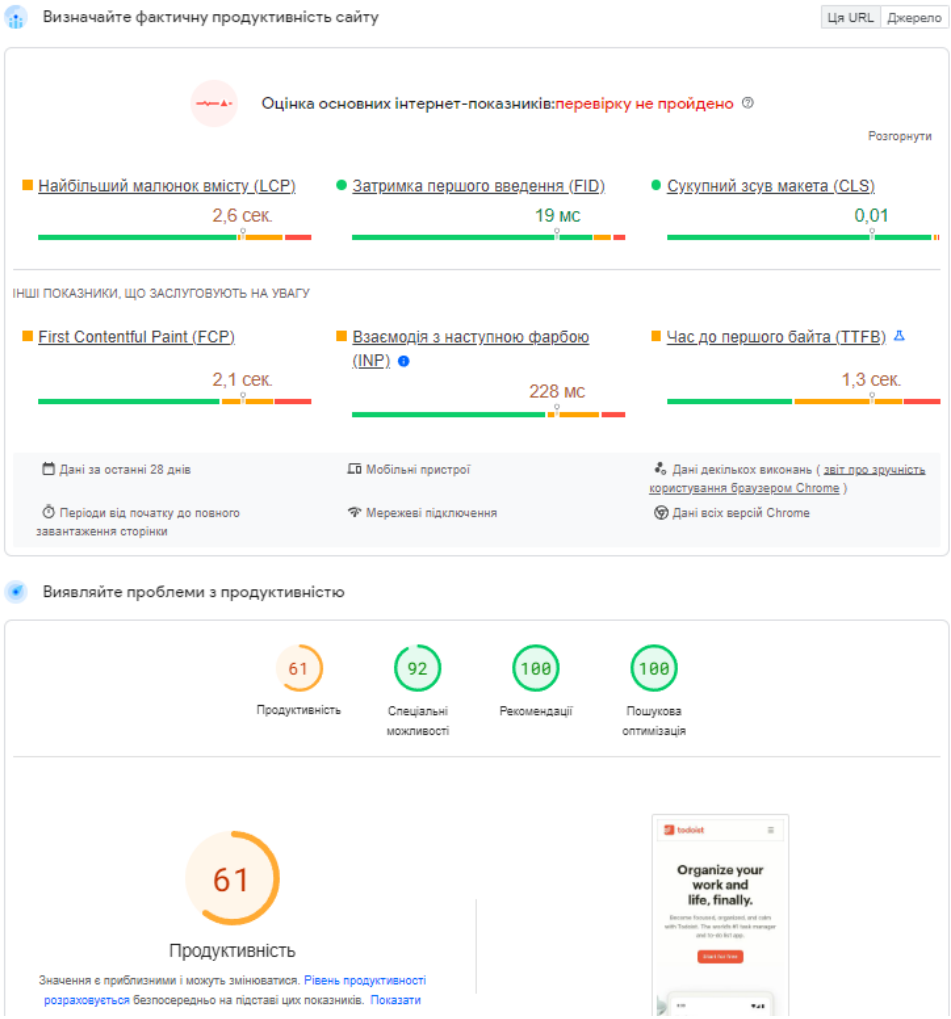

Рисунок *7 — Результат проведення тестування продуктивності веб-сайту*

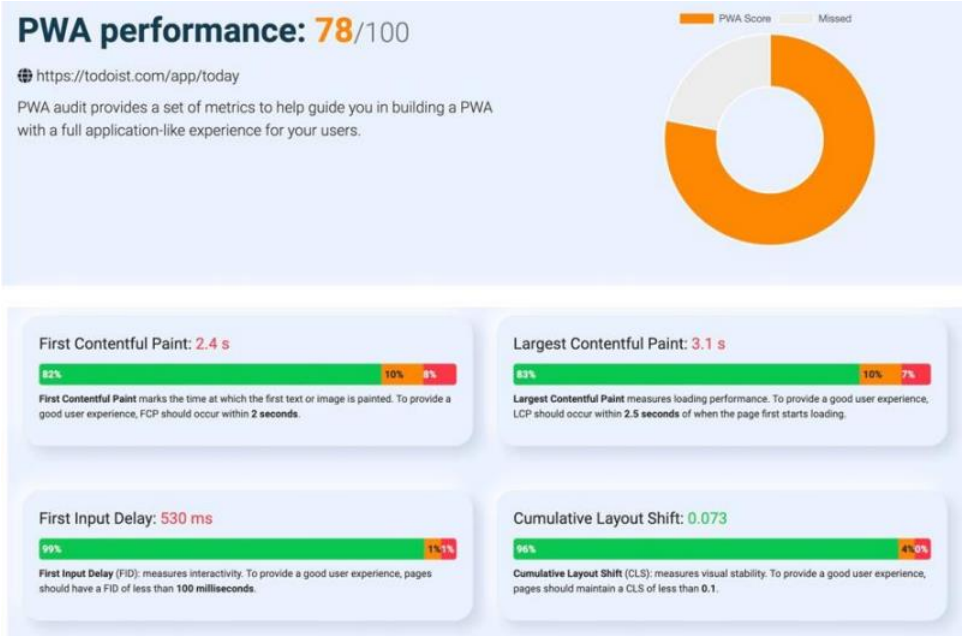

Рисунок *8 — Результат проведення автоматизованого тестування продуктивності PWA додатку*

3.3 Особливості UI/UX

Чи є нативні додатки ідеальним варіантом для мобільного UX? На відміну від веб-сайтів, нативні додатки створюються спеціально для смартфонів, що дозволяє користувачам отримувати максимум від своїх пристроїв. Однак створення нативного додатку вимагає хорошого знання нативних мобільних мов і фреймворків, таких як Swift і .NET, а також проєктування інтерфейсів щонайменше для двох платформ.

З іншого боку, код PWA адаптується до різних платформ, пропонуючи однакову функціональність і шаблони користувацького інтерфейсу на Android, iOS і десктопах.

Однак немає явного переможця з точки зору користувацького досвіду, оскільки за прогресивними веб-додатками слідують нативні додатки з плавним та інтуїтивно зрозумілим інтерфейсом та інтерфейсом користувача.

PWA можуть аналізувати операційну систему на рівні додатків і використовувати різні бібліотеки інтерфейсу, що дозволяє визначати, чи користу-

вач використовує додаток на мобільному телефоні чи комп'ютері. З цієї причини інтерфейс може динамічно змінюватися, і PWA можуть адаптуватися до конкретної операційної системи.

PWA та веб-сайт ідентичні на мобільній версії, а нативний додаток має деякі відмінності:

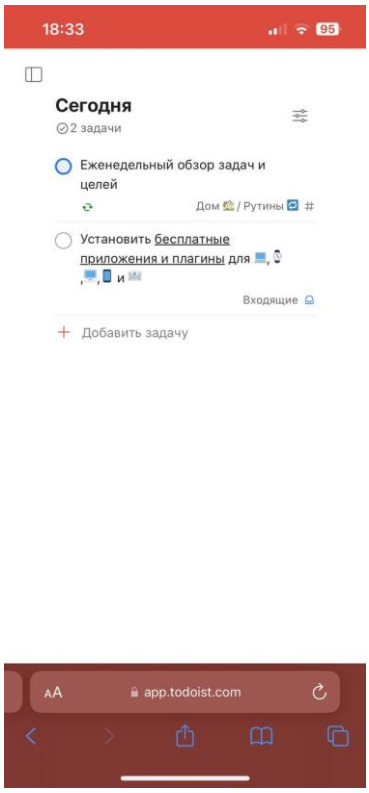

Рисунок 9 — *Інтерфейс веб-сайту*

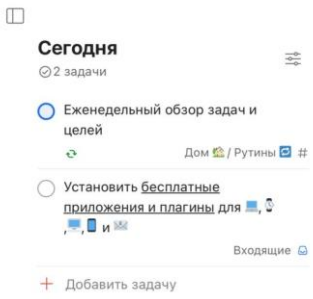

Рисунок *10 — Інтерфейс PWA додатку*

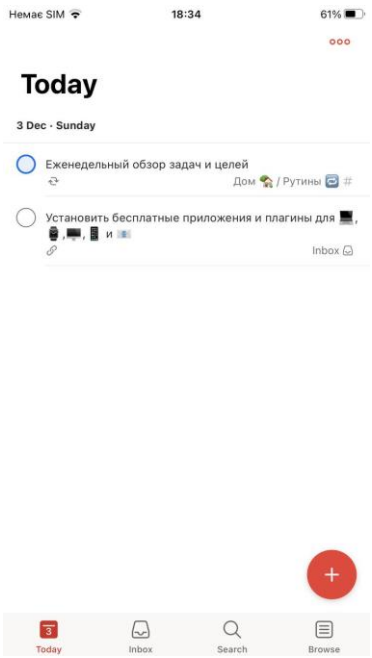

Рисунок *11 — Інтерфейс нативного додатку*

3.4 Функція push – повідомлень

Пуш-сповіщення повторно залучають відвідувачів. Цю функцію можна створити з нуля або інтегрувати в додаток за допомогою сторонніх сервісів,

таких як PushBots або OneSignal. Нативні додатки також можуть використовувати цю функцію, оскільки вона безперебійно працює в усіх операційних системах.

PWA можуть надсилати push-сповіщення через працівників сервісу. Однак ця функція залежить від платформи: Вона доступна в Chrome, Opera, Safari і Mozilla, але не так просто на iOS. Пуш-сповіщення на PWA добре працюють на Android, але для користувачів iOS ця функція обмежена.

З розвитком технологій ці обмеження будуть подолані, і PWA та локальні додатки опиняться в більш рівних умовах. Наразі, останні є лідером у цьому відношенні.

3.5 Офлайн-режим

Як PWA — рішення, так і нативні додатки забезпечують офлайн-доступ і працюють при повільному інтернет-з'єднанні.

Service Worker PWA забезпечує автономне управління запитами, попереднє завантаження, кешування певних ресурсів і синхронізацію даних з віддаленими серверами. Додатки можна додавати на домашній екран, а потім негайно запускати для використання в автономному режимі або в умовах низької пропускної здатності.

Сторінки, що були відкриті раніше, повернуть результати. Інформація оновлюється після відновлення з'єднання. Таким чином, і PWA, і нативні додатки дозволяють працювати в автономному режимі (незалежно від стабільності інтернет-з'єднання).

3.6 Можливості та обмеження

Нативні програми призначені для роботи в певному програмному середовищі. Наприклад, нативні програми для iOS підтримують специфічні функції Apple, такі як Face ID, а програми для Android і Windows отримують апаратні можливості.

Прогресивні веб-додатки мають справу з різними обмеженнями при використанні HTML, JS і CSS. Ці обмеження залежать від платформи або системи, на якій працює додаток. Наприклад, iOS не підтримує PWA push-сповіщення. На противагу цьому, Android має кращу підтримку PWA і розвиває свою екосистему швидше, ніж iOS.

Тому PWA не мають доступу до низькорівневих апаратних функцій нативних додатків.

3.7 Доступність для пошукових систем

Пошукові системи не ранжують локальні додатки. Тому SEO-стратегії не можуть бути використані для підвищення впізнаваності бренду серед потенційних клієнтів. Що стосується PWA, Google індексує їх як веб-сайти. Крім того, швидкість і мобільність PWA вважаються важливими факторами SEO-рейтингу.

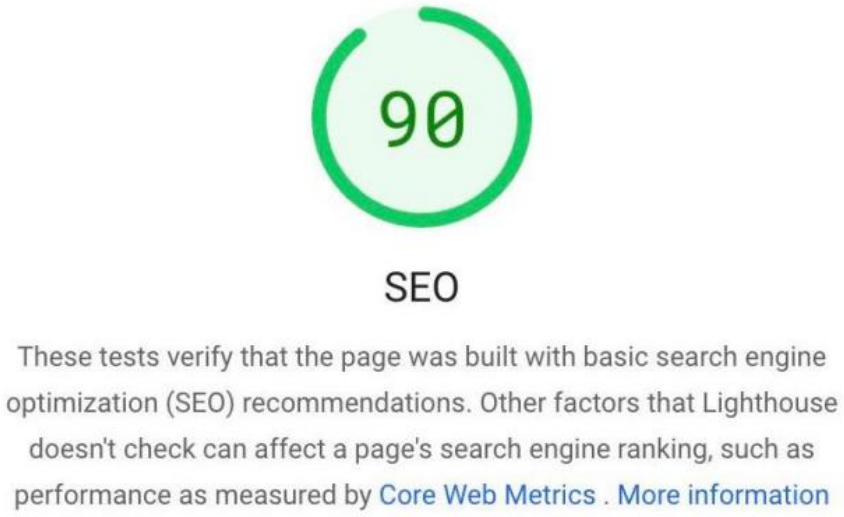

Рисунок *12 – Інтерфейс нативного додатку*

SEO — елементи на веб-сайтах і PWA-додатках дають 90% результату, коли у нативних додатках немає SEO-елементів (рисунок 12).

3.8 Безпека

Нативні додатки вважаються більш безпечними з наступних причин:

1. вони можуть використовувати багатофакторну аутентифікацію;
2. забезпечують більш безпечну комунікацію за допомогою закріплення сертифікатів;

Коли нативний додаток звертається до апаратного забезпечення, операційна система надає йому дозвіл на доступ до його модулів. Як наслідок, нативні програми є більш надійними, ніж URL-адреси.

Водночас PWA не позбавлені важливих елементів безпеки, оскільки вони працюють за протоколом HTTPS і мають такі функції, як маніфести веб-додатків і працівники служби. Це гарантує, що передача даних між клієнтом і сервером є безпечною. Таким чином, вміст і взаємодія захищені так само, як і на захищеному веб-сайті.

Безпека PWA покладається на браузер, який виступає в ролі віртуального захисника. PWA працюють у пісочниці браузера, і їхня функціональність вже обмежена цим дуже обмеженим та ізольованим середовищем. Таким чином, PWA можуть отримати доступ лише до ресурсів браузера і тільки до тих ресурсів, які браузер дозволяє. Це ізолює додаток від апаратного забезпечення смартфона та конфіденційних даних користувача.

3.9 Витратити на розробку

Розробка та підтримка PWA коштує дешевше, ніж нативних додатків. Це не так, як витрачати гроші окремо на нативні додатки для Android та iOS, які мають власні інструменти розробки та мови коду. Крім того, PWA сумісні з різними браузерами, тому додатки можна використовувати як на десктопних, так і на мобільних пристроях.

Нативні додатки вимагають вивчення мов і створення окремих версій для кожної платформи (принаймні одна версія для iOS і одна для Android).

Такимчином,можна зробити висновок, що переваги PWA полягаютьу наступному:

1. PWA працює в будь-якому браузері на десктопних і мобільних пристроях;
2. додаток швидко завантажується та використовується на будь-якому пристрої та з будь-яким підключенням до Інтернету (і навіть є можливість працювати в офлайн-режимі);
3. PWA реалізували найкращі практики UI/UX нативних додатків, щоб залучити користувача за допомогою інтуїтивно зрозумілого та привабливого інтерфейсу;
4. вони включають в себе просту функцію "Додати на домашній екран", що дозволяє зберегти легкий PWA на своєму пристрої як ярлик для швидкого доступу;

5. за зразком нативних додатків, вони підтримують push-повідомлення, які мають вирішальне значення для утримання клієнтів;
6. оскільки прогресивний веб-додаток – це веб-сайт, його сторінки можуть бути виявлені пошуковими системами.

З іншого боку, у PWA є і недоліки:

1. існує обмежений набір апаратних функцій, доступних для PWA для iOS. Багато функціональних можливостей все ще заблоковані в підтримці iOS PWA;
2. однією з найбільш критичних відсутніх функцій є можливість надсилання push-повідомлень користувачам iOS;
3. PWA споживають більше заряду акумулятора. Оскільки додаток розробляється за допомогою фреймворків, PWA вимагає більше процесорної потужності і виснажує акумулятор.

Переваги нативних додатків полягають у наступному:

1. вони завантажуються в пам'ять мобільного пристрою, що дає їм повний доступ до апаратного забезпечення, API, функціональних можливостей і даних користувача. Вони також сумісні з іншими продуктами Google та Apple;
2. вони розроблені виключно для мобільного використання для інтеграції з ОС та мають відшліфований UI/UX;
3. з огляду на цей набір висновків, PWA краще підходять для простих додатків які не мають великої кількості низькорівневих функцій.

3.10 Висновки до розділу 3

З аналізу можна зробити висновок, що обираючи між розробкою прогресивних веб-додатків (PWA) та нативних додатків для мобільних пристроїв, слід враховувати ряд ключових факторів.

PWA пропонують широкий охоплення аудиторії, оскільки вони працюють в різних браузерах і на різних платформах. Вони забезпечують високий

рівень доступності для пошукових систем, що сприяє поліпшенню SEO-рейтингу. Зручний інтерфейс та можливість використання на різних пристроях роблять PWA зручними для широкого кола користувачів. Офлайн-режим та низькі витрати на розробку є ще однією перевагою.

З іншого боку, нативні додатки володіють більшою надійністю та швидкодією, а також здатністю використовувати специфічні функції кожної операційної системи. Забезпечуючи більше можливостей для використання апаратного забезпечення та інтеграції з іншими продуктами, нативні додатки підходять для складніших завдань та функцій.

Залежно від конкретних потреб та обмежень проєкту, обирання між PWA та нативним додатком є питанням збалансованого вибору між доступністю, швидкістю та можливостями.

РОЗДІЛ 4 ТЕХНОЛОГІЧНІ ПЕРСПЕКТИВИ ТА ОБМЕЖЕННЯ

На Google I/O 2017 було оголошено про важливу ініціативу, орієнтовану на розробників, і Едді Османі продемонстрував технічну базову лінію під назвою HNPWA для тестування фреймворків JavaScript на відповідність критеріям PWA. Базова лінія, яка називається HNPWA, має на меті допомогти розробникам вибрати фреймворк JavaScript для створення PWA. Тест включає ряд фреймворків, зокрема React, Vue.js та Angular. Усі тестові кодові бази мають відкритий вихідний код, і тести з використанням нових фреймворків та підходів можуть бути додані на сайт HNPWA, щоб інші могли з ними ознайомитися.

У PWA доступ до API пристроїв і платформ обмежений тими API, які підтримуються браузером користувача. Це одне з головних обмежень підходу PWA порівняно з нативною або крос-платформною розробкою, де всі або більшість відкритих API пристроїв доступні розробникам через нативні SDK або подібні абстракції. Дещо обнадіює той факт, що команда Google Chrome додала 215 нових API лише за останній рік, і прогрес у подоланні цього розриву, безумовно, покращується. Однак неминуче додавання нових функцій до мобільних платформ у майбутньому означає, що доступ до нової функціональності через PWA все ще обмежений, доки специфікація HTML/JavaScript не буде формалізована W3C і (принаймні) впроваджена виробниками браузерів. Фрагментація платформ, версій операційних систем, функціональних можливостей браузерів і пристроїв не сприяє розвитку Інтернету в єдиному напрямку. Навіть у сфері мов програмування Інтернет може зазнати подальшої фрагментації. Однією з помітних технологій, яка повільно набирає обертів у галузі, є WebAssembly (wasm) - формат та інструмент для написання веб-додатків з використанням інших мов, окрім JavaScript. Це може створити можливості для перенесення такого програмного забезпечення, як ігри, які традиційно запускалися лише як нативні десктопні програми.

4.1 Статус Quo прийняття PWA

На конференції для розробників Google, Google I/O 2017, було представлено низку прогресивних ініціатив у сфері веб-додатків, як на сцені, так і у вигляді згадок. Великі підприємства та ключові гравці мобільної веб-індустрії почали з великим успіхом перетворювати свої існуючі веб-додатки на PWA.

Серед них - провідні компанії у своїх галузях, такі як вищезгадані Twitter та OLA. Компанії також опублікували статистику щодо розміру своїх додатків, повідомляючи про збільшення використання додатків після прийняття PWA.

У той час як нативний додаток Twitter для Android має розмір понад 23 МБ, а додаток для iOS - понад 100 МБ, PWA становить 0,6 МБ (600 КБ) і все ще пропонує більшу частину очікуваного функціоналу. Компанія також демонструє, що її PWA, Twitter Lite, робить значний внесок у розвиток ринків, що розвиваються, щодня відкриваючи понад мільйон домашніх екранів. Аналогічно, OLA, найбільший в Індії додаток для виклику таксі, виявив, що його PWA важить 0,20 МБ (200 КБ), що значно легше, ніж його рідні додатки для Android та iOS, які важать 60 МБ і 100 МБ відповідно. oLA сегментувала своїх клієнтів на рівні залежно від їхнього місцезнаходження. За даними OLA, в регіонах рівня 3 мобільний трафік збільшився на 68% з моменту запуску PWA, а коефіцієнт конверсії на 30% вищий, ніж у рідних додатках. У регіонах рівня 2 коефіцієнти конверсії з PWA були такими ж, як і з нативних додатків.

Це свідчить про бізнес-потенціал і важливість PWA для таких компаній, як OLA, що працюють у різних галузях і мають справу з ситуаціями, пов'язаними з низьким рівнем зв'язку.

На конференції також було представлено сім доповідей, присвячених потенціалу мобільного інтернету наступного покоління для покращення та захоплення PWA в контексті мобільного користувацького досвіду (UX), підт-

римки технічного фреймворку, тестування продуктивності та міграції. Очевидно, що Google просуває PWA як ініціативу, спрямовану на покращення користувацького досвіду мобільного інтернету.

Мобільна розробка - нативна, кросплатформенна чи веб-розробка - стрімко розвивається, постійно з'являються нові технічні фреймворки, підходи та методи. Як наслідок, існує фрагментарність спільноти розробників, підходів до розробки та думок про те, як розробляти додатки.

По-перше, популяризація Kotlin як першокласної мови програмування для платформи Android заохотить більшу кількість розробників долучитися до розробки під Android. Але ще цікавішою є нова розробка розробника Kotlin, компанії Jetbrains, під назвою Kotlin/Native. Цей бекенд на основі LLVM допомагає розробникам запускати Kotlin на платформах, що не є (J)VM, таких як iOS. Це означає, що незабаром Kotlin може підтримувати Android за замовчуванням, що зробить його придатним для крос-платформної розробки. Кросплатформна підтримка не обмежується Android та iOS, але повідомляється, що також підтримуються такі платформи, як MacOS, Ubuntu та Raspberry Pi. Kotlin можна конвертувати в JavaScript разом з іншими рішеннями, такими як CoffeeScript та TypeScript, і розробники зможуть використовувати Kotlin для розробки програмного забезпечення на різних платформах.

По-друге, зараз, ми спостерігаємо поширення технічних фреймворків та підходів для розробки мобільних додатків. Хоча крос-платформні фреймворки для додатків, такі як PhoneGap, Titanium Appcelerator, DragonRad і Rhodes, часто фігурували в попередніх роботах, нещодавно з'явилася низка нових, більш технічно просунутих фреймворків.

Хоча останні дослідження висвітлюють такі фреймворки, як React Native, Ionic та Fuse, інші рішення, такі як NativeScript, Quasar та Apache Weex, ще не розглядалися в академічному контексті, наприклад, на тому ж рівні, що й PhoneGap. По мірі того, як PWA ставатимуть дедалі популярнішими, з'являтимуться й інші напрямки для таких фреймворків. Вони відійдуть від нативної

розробки додатків і перейдуть до веб-орієнтації без нативних абстракцій, як, наприклад, Cordova.

4.2 Висновки до розділу 4

Отже PWA надає обмежений доступ до API пристроїв та платформ, що визначається підтримкою браузера користувача. Однак з введенням нових API в браузерах, таких як Google Chrome, спостерігається певний прогрес. Фрагментація платформ та обмежений доступ до функціональності залишають відкритими питання розвитку PWA.

У контексті веб-розробки, важливим кроком є рух до використання WebAssembly (wasm), що дозволяє писати веб-додатки за допомогою інших мов програмування, окрім JavaScript.

У висновку слід відзначити широкий спектр технологій, що швидко розвиваються в галузі мобільної розробки, та постійні зміни в підходах до створення веб-додатків. Фрагментація і різноманітність цих технологій відкривають нові перспективи та виклики для розробників у створенні сучасних мобільних додатків.

РОЗДІЛ 5 РОЗРОБКА ДОДАТКА «ПЛАНУВАЛЬНИК»

5.1 Середовище розробки

Для розробки PWA-додатку я обрав середовище розробки (IDE) WebStorm. WebStorm - це продукт від компанії JetBrains, який спеціалізується в розробці на JavaScript та пов’язаних з ним технологіях.

Вибір WebStorm обумовлений рядом його переваг. По-перше, WebStorm надає інтелектуальне редагування коду, що включає автоматичні підказки та аналіз коду. Це допомагає виявляти помилки ще до виконання коду, що є важливим для ефективної розробки.

По-друге, WebStorm має потужні інструменти для пошуку та навігації, що дозволяють швидко знайти потрібні частини коду. Це полегшує роботу з великими проєктами та сприяє продуктивності.

Враховуючи ці переваги, WebStorm є ідеальним вибором для розробки PWA-додатків. Його функції та інструменти можуть значно полегшити та прискорити процес розробки.

5.2 Стек технологій

Для розробки свого додатку я буду використовувати декілька інструментів.

React - це бібліотека для створення користувацького інтерфейсу. React дозволяє розбивати інтерфейс на незалежні компоненти, що спрощує розробку та підтримку коду. Крім того, React має велику спільноту розробників та багато навчальних матеріалів, що робить його відмінним вибором для будь-якого проєкту.

Material-UI - це бібліотека компонентів користувацького інтерфейсу, яка допомагає швидко створювати гарні та зручні інтерфейси. Вона базується на

принципах дизайну Material Design від Google, це гарантує, що додаток буде виглядати сучасно та професійно.

А також TypeScript, він використовується для підвищення рівня надійності та структурованості коду. Статична типізація TypeScript допомагає виявляти та усувати помилки на етапі розробки, що полегшує процес підтримки та розширення кодової бази.

5.3 Можливості додатку

Додаток надає користувачам зручний інструмент для управління їхніми завданнями та обліку особистих цілей. На головній сторінці відображаються всі створені користувачем завдання, з можливістю сортування за категоріями та швидкого пошуку. Якщо списку завдань немає, сторінка залишається чистою та готовою для нових записів.

Кожне завдання можна змінити, відзначити як виконане чи невиконане, а також закріпити чи відкріпити для відзначення важливості. Для детального огляду і управління кожним завданням доступне відкриття окремого меню з додатковими опціями, такими як редагування, створення дублікату чи повне видалення.

Створення нового завдання - це простий та інтуїтивно зрозумілий процес, де користувач може вказати назву, опис, емодзі, дату та час нагадування, категорію та кольорове позначення. Також, існуючі категорії можна змінювати чи додавати нові, для зручного групування завдань.

Профіль користувача дозволяє внесення змін у власне ім'я та зображення профілю, а також можливість видалення облікового запису. Мобільні можливості та локальне збереження даних роблять додаток доступним та ефективним для використання навіть у відсутність Інтернет-з'єднання.

Зручна функція імпорту та експорту у форматі JSON дозволяє зберігати та відновлювати дані, забезпечуючи зручний механізм резервного копіювання.

Також, відображення звітів у вигляді діаграм додає аналітичний аспект до відстеження завдань і планування.

Додаток може працювати повністю в offline режимі. Це стає можливо завдяки тому, що всі данні зберігаються в Local Storage. Структура Local Storage рисунок 12.

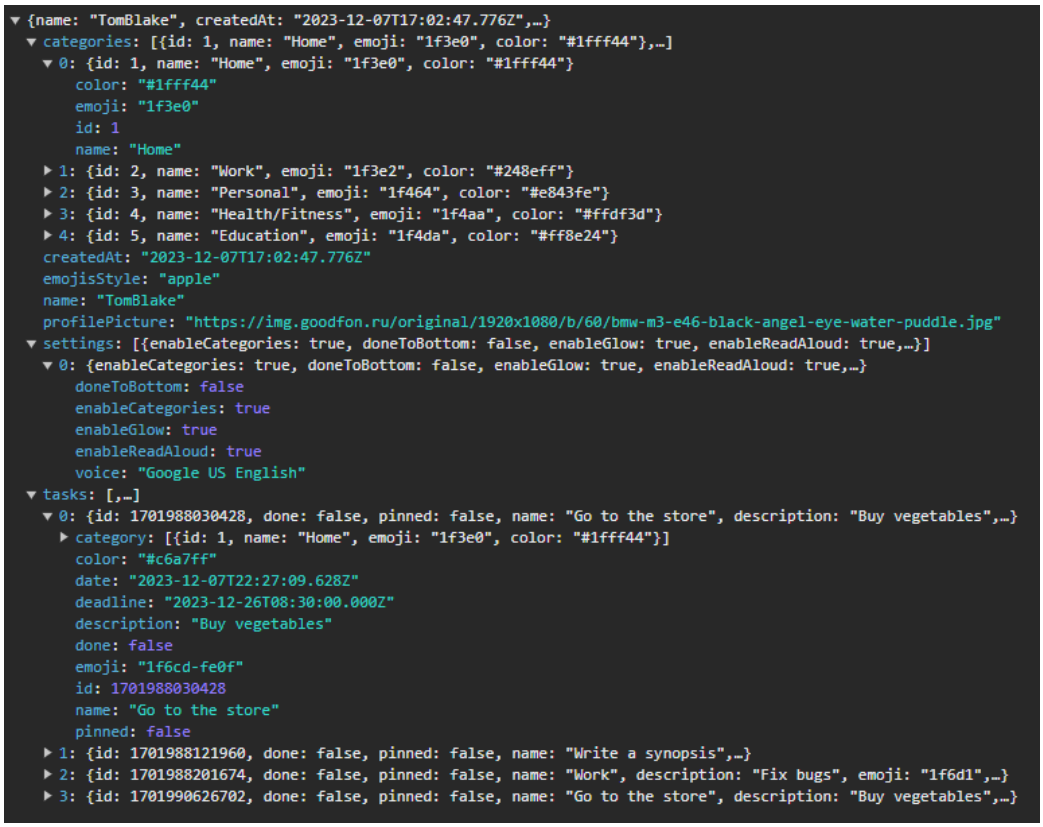

Рисунок *12 — структура даних в local storage*

5.4 UI додатку

Початкова сторінка додатку (рисунок 13). На ній знаходяться задачі, які створив користувач. Якщо ніяких задач нема, то початкова сторінка буде виглядати як на (рисунок 14).

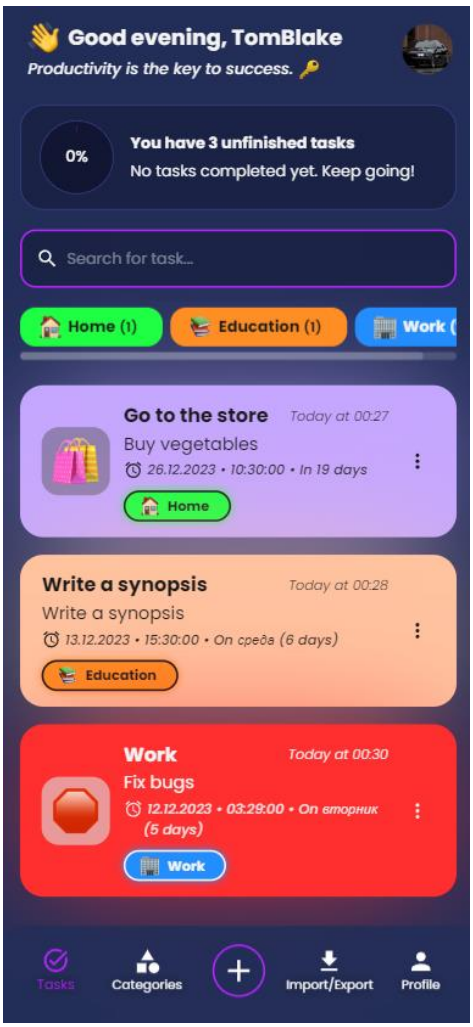

Рисунок *13 — Головна сторінка додатку*

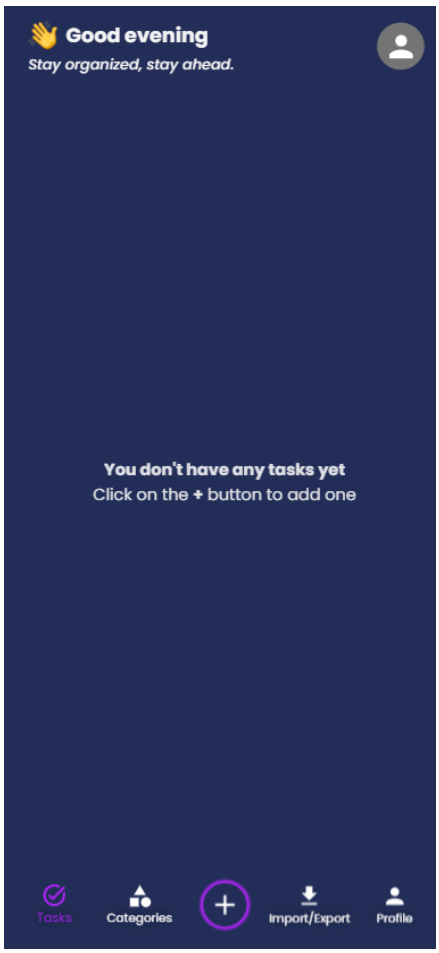

Рисунок *14 — Головна сторінка додатку, якщо задачі відсутні*

Задачі можна сортувати за категоріями. Також є можливість пошуку задач по назві або опису. Якщо натиснути на три крапки з правого боку задачі, то відкриється меню див на (рисунок 15).

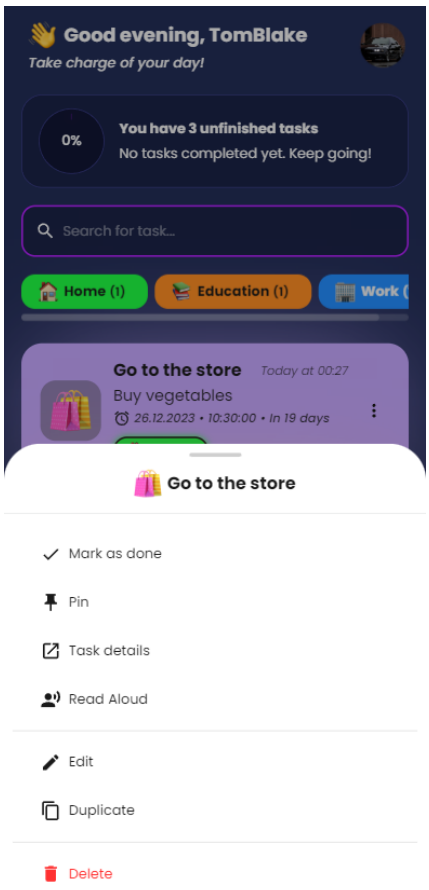

Рисунок *15 — Меню задачі*

В меню є такі функції:

1. Mark as done або Mark as not done – позначити як виконане або не виконане, відповідно.
2. Pin або Unpin – закріпити або відкріпити. Закріпляє завдання поверх решти пунктів.
3. Task details – відкриває сторінку з детальною інформацією про завдання див. (рисунок 16).
4. Read Aloud – читати в голос. Голосовий асистент читає основну інформацію про завдання.
5. Edit – відкриває діалогове вікно, в якому є можливість редагувати завдання див. (рисунок 17).
6. Duplicate – робить дублікат задачі.
7. Delete – безповоротно видаляє завдання з списку.

Рисунок *16* — *Сторінка з детальною інформацією про задачу*

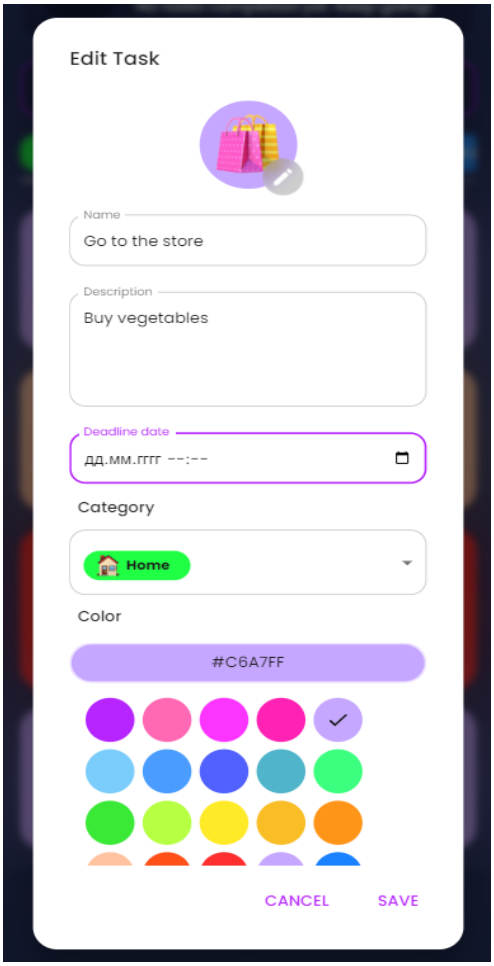

Рисунок *17 – Діалогове вікно з можливістю зміни налаштувань завдання*

Якщо нажати на знак плюс внизу екрана по середині то ми перейдемо на наступну сторінку.

Наступна сторінка «Додати нове завдання» див. (рисунок 18). На цій сторінкі можна вказати:

1. Емодзі.
2. Назву.
3. Опис.
4. Дату та час нагадування.
5. Категорію.
6. Колір.

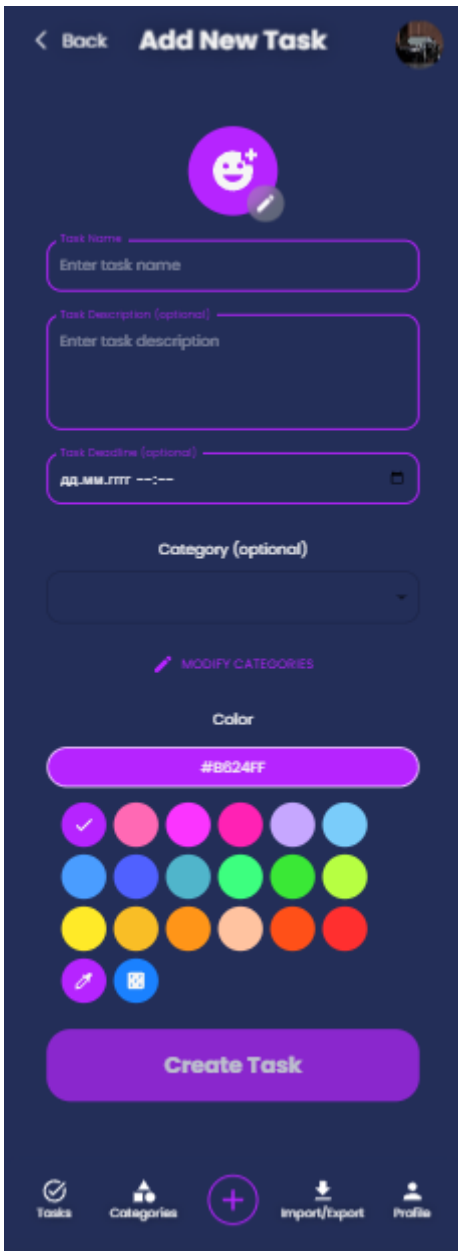

Рисунок *18 — сторінка «Додати нове завдання»*

Наступна сторінка «Категорії» (див. рисунок 19) на неї можна попасти або з навігаційного меню, або з сторінки «Додати нове завдання», якщо натиснути на кнопку «Modify Categories».

На цій сторінці представлені стандартні категорії. Також ми маємо змогу змінювати, видаляти існуючі категорії або створювати нові. В подальшому можна присвоювати будь-яку категорію для задач.

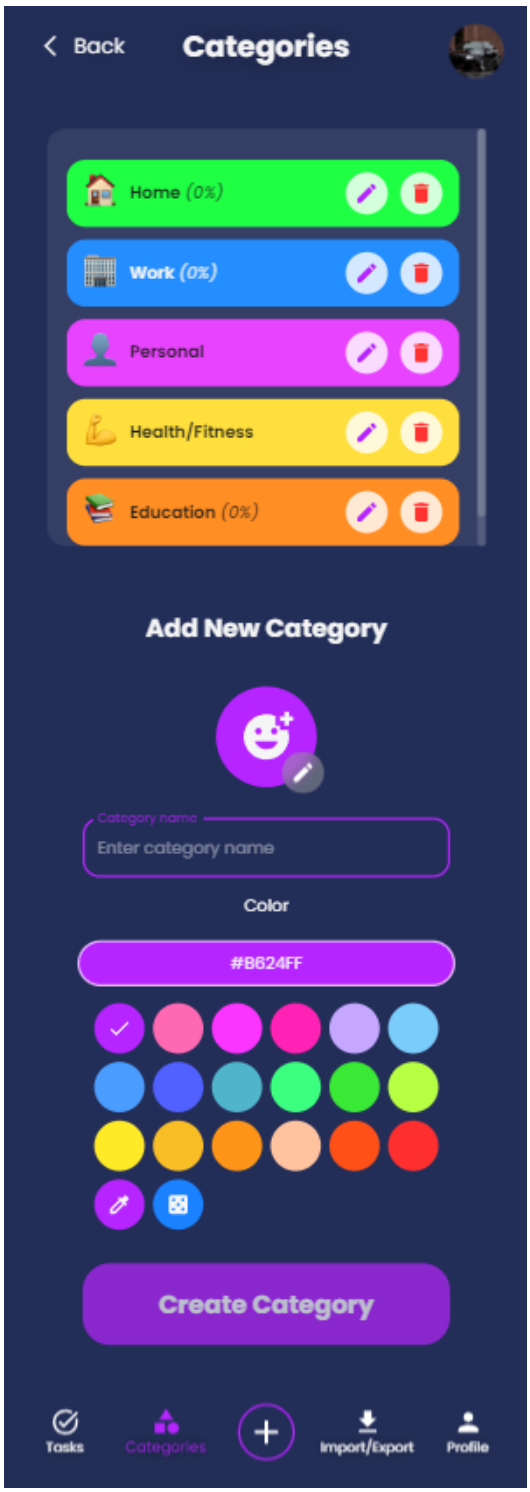

Рисунок *19* — *сторінка «Категорії»*

Наступна сторінка «Профіль» див. рисунок 20. На цій сторінці ми можемо змінювати ім’я користувача, а також картинку профіля (Аватар). Кнопка «Delete Account» видалить всі данні та обліковий запис.

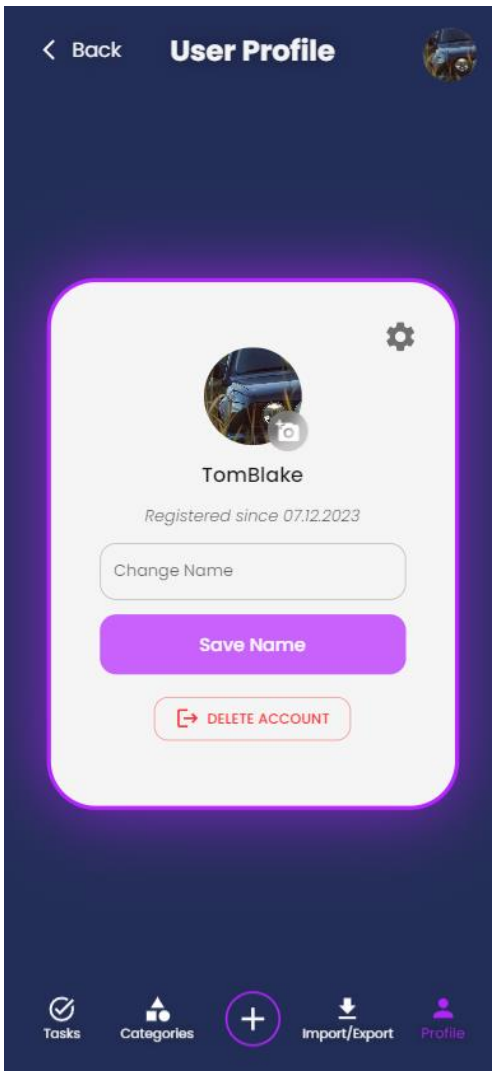

Рисунок *20 — сторінка «Категорії»*

Сторінка «Import/Export» див. рисунок 21. На цій сторінці ми можемо експортувати або імпортувати список задач, або деякі окремо. Дані про завдання експортуються в файл JSON. Лістинг 1 демонструє структуру JSON файла.

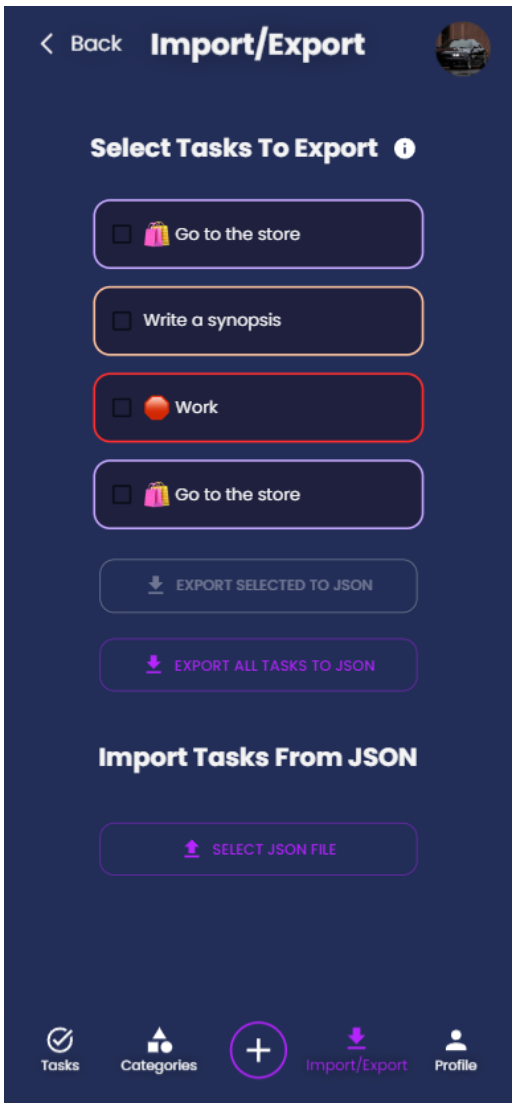

Рисунок *21* — *сторінка «Import/Export»*

Лістинг *1 Структура JSON файла*

```
[
  {
    "id": 1701988030428,
    "done": false,
    "pinned": false,
    "name": "Go to the store",
    "description": "Buy vegetables",
    "emoji": "1f6cd-fe0f",
    "color": "#c6a7ff",
    "date": "2023-12-07T22:27:09.628Z",
    "deadline": "2023-12-26T08:30:00.000Z",
    "category": [
      {
        "id": 1,
        "name": "Home",
        "emoji": "1f3e0",
```

```
      "color": "#1fff44"
    }
  ]
},
{
  "id": 1701988121960,
  "done": false,
  "pinned": false,
  "name": "Write a synopsis",
  "description": "Write a synopsis",
  "color": "#ffc3a0",
  "date": "2023-12-07T22:28:41.060Z",
  "deadline": "2023-12-13T13:30:00.000Z",
  "category": [
    {
      "id": 5,
      "name": "Education",
      "emoji": "1f4da",
      "color": "#ff8e24"
    }
  ]
},
{
  "id": 1701988201674,
  "done": false,
  "pinned": false,
  "name": "Work",
  "description": "Fix bugs",
  "emoji": "1f6d1",
  "color": "#FF2F2F",
  "date": "2023-12-07T22:30:01.032Z",
  "deadline": "2023-12-12T01:29:00.000Z",
  "category": [
    {
      "id": 2,
      "name": "Work",
      "emoji": "1f3e2",
      "color": "#248eff"
    }
  ]
},
{
  "id": 1701990626702,
  "done": false,
  "pinned": false,
  "name": "Go to the store",
  "description": "Buy vegetables",
  "emoji": "1f6cd-fe0f",
  "color": "#c6a7ff",
```

```
    "date": "2023-12-07T23:10:25.796Z",
    "deadline": "2023-12-26T08:30:00.000Z",
    "category": [
      {
        "id": 1,
        "name": "Home",
        "emoji": "1f3e0",
        "color": "#1fff44"
      }
    ]
  }
]
```


Діалогове вікно «Налаштування» рисунок 22. В налаштуваннях можна змінити набір набір емодзі. Для вибору доступні такі набори:

1. Apple.
2. Facebook, Messenger.
3. Twitter, Discord.
4. Google.
5. Native.

Також є можливість включити/виключити категорії, ефекти світіння, читання голосом. Можливо налаштувати читання голосом, а саме вибрати мову та голос помічника.

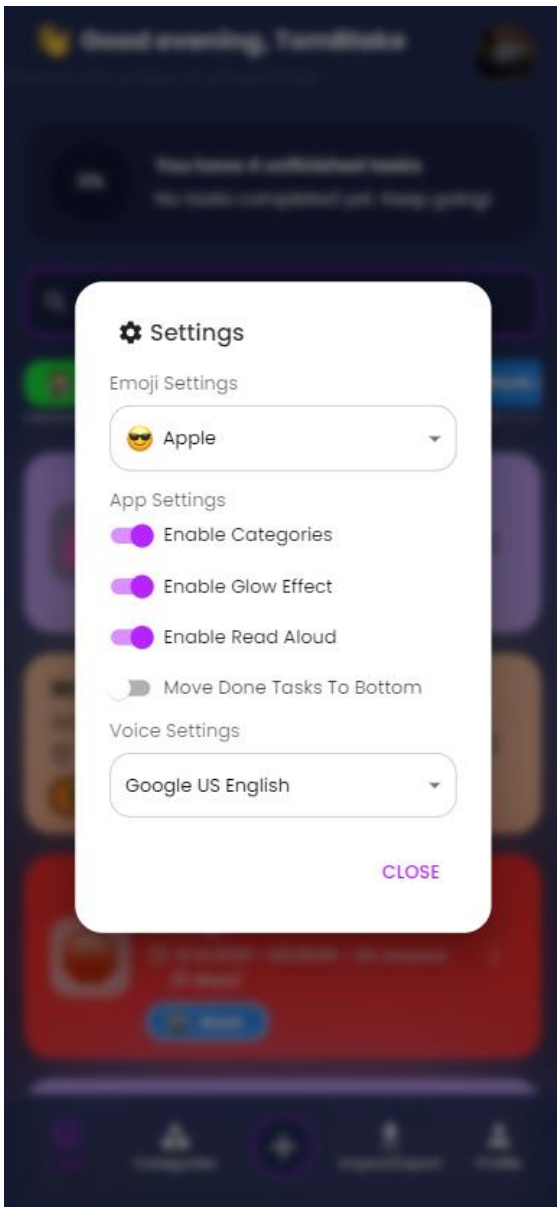

Рисунок *22 — Структура JSON файла*

5.5 Реалізація PWA — підтримки

Для ефективної підтримки прогресивних веб-додатків (PWA) у розроблюваному проєкті використовується інструмент **vite-plugin-pwa**. Цей плагін інтегрує необхідні PWA-функціональності та надає простий та швидкий спосіб забезпечення offline-режиму, додавання на головний екран та інших переваг PWA.

Для інтеграції потрібно завантажити та встановити vite-plugin-pwa за допомогою менеджера пакетів командою: `npm install vite-plugin-pwa --save-dev`. Потім налаштувати файл vite.config.js (лістинг 2).

Лістинг *2 файл vite.config.js*

```
export default defineConfig({
  server: {
    host: true
  },
  plugins: [
    react(),
    VitePWA({
      registerType: "autoUpdate",

      workbox: {
        globPatterns: ["**/*"],
      },
      includeAssets: ["**/*"],
      manifest: {
        theme_color: "#232e58",
        background_color: "#232e58",
        display: "standalone",
        scope: "/",
        start_url: "/",
        short_name: "Todo App",
        description: "Todo App",
        name: "Todo App",
        icons: [
          {
            src: "logo192.png",
            sizes: "192x192",
            type: "image/png",
          },
          {
            src: "logo256.png",
            sizes: "256x256",
            type: "image/png",
          },
          {
            src: "logo384.png",
            sizes: "384x384",
            type: "image/png",
          },
          {
            src: "logo512.png",
            sizes: "512x512",
```

```
      type: "image/png",
    },
  ],
  shortcuts: [
    {
      name: "Add Task",
      description: "Add Task",
      url: "/add",
      icons: [
        {
          src: "add.png",
          sizes: "192x192",
          type: "image/png",
        },
      ],
    },
    {
      name: "Categories",
      description: "Task Categories",
      url: "/categories",
      icons: [
        {
          src: "categories.png",
          sizes: "192x192",
          type: "image/png",
        },
      ],
    },
    {
      name: "Import Export",
      description: "Import or Export Task",
      url: "/import-export",
      icons: [
        {
          src: "import-export.png",
          sizes: "192x192",
          type: "image/png",
        },
      ],
    },
    {
      name: "Profile",
      description: "User Profile",
      url: "/user",
      icons: [
        {
          src: "profile.png",
          sizes: "192x192",
          type: "image/png",
```

```
              },
            ],
          },
        ],
      },
    }),
  ],
});
```

Плагін дозволяє зручно налаштовувати різні аспекти PWA, такі як іконки, кольори, offline-стратегії та інше. Також він автоматично генерує маніфест та Service Worker, що спрощує розгортання та управління PWA-функціональністю.

Завдяки vite-plugin-pwa процес реалізації PWA-підтримки значно спрощується, дозволяючи швидко та ефективно впроваджувати прогресивні можливості у веб-додаток.

Якщо додаток відповідає вимогам, то з’являється можливість завантаження web-додатку на пристрій.

Також досить важливо правильно налаштувати router в файлі router.tsx (лістинг 3), для правильної навігації в застосунку.

Лістинг 3 *файл router.tsx*

```
export const AppRouter = ({ user, setUser }: UserProps):
JSX.Element => {
  const userProps = { user, setUser };

  return (
    <Routes>
      <Route path="/" element={<Home {...userProps} />} />
      <Route path="/task/:id" element={<TaskDetails
{...userProps} />} />
      <Route path="/add" element={<AddTask {...userProps}
/>} />
      <Route path="/user" element={<UserSettings
{...userProps} />} />
      <Route path="/import-export" element={<ImportExport
{...userProps} />} />
      <Route path="/categories" element={<Categories
{...userProps} />} />
      <Route path="*" element={<NotFound />} />
    </Routes>
```

```
    );
};
```

5.6 Вигляд додатку на різних платформах

На різних девайсах, а також в різних додатках, по типу PWA чи WEB, зовнішній вигляд додатка трохи змінюється:

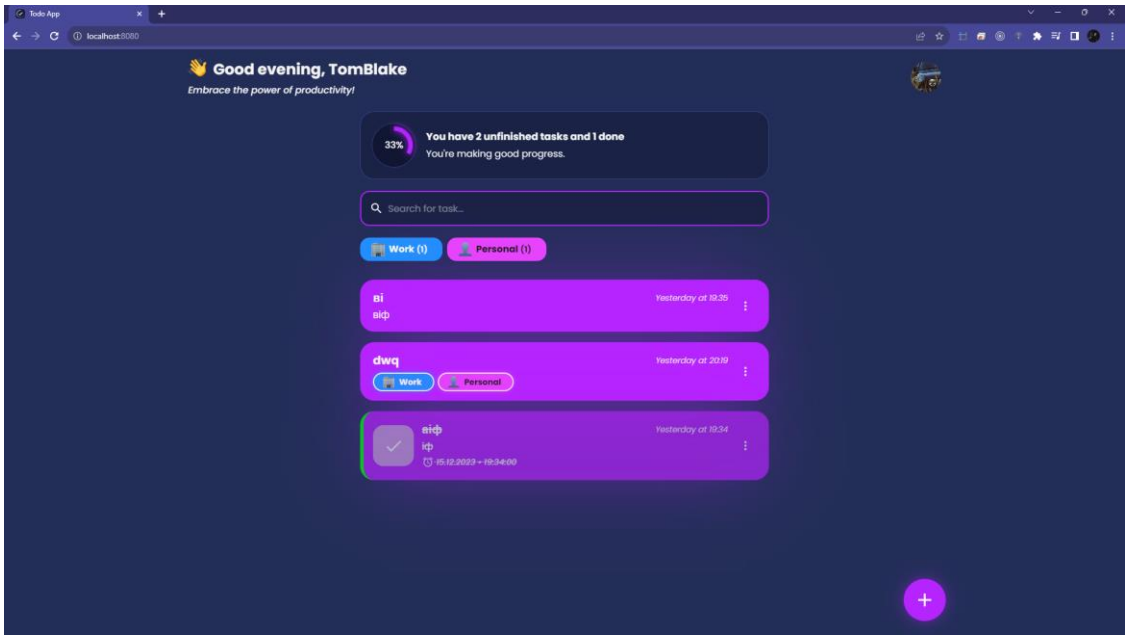

Рисунок *23 — WEB-додаток десктоп версія*

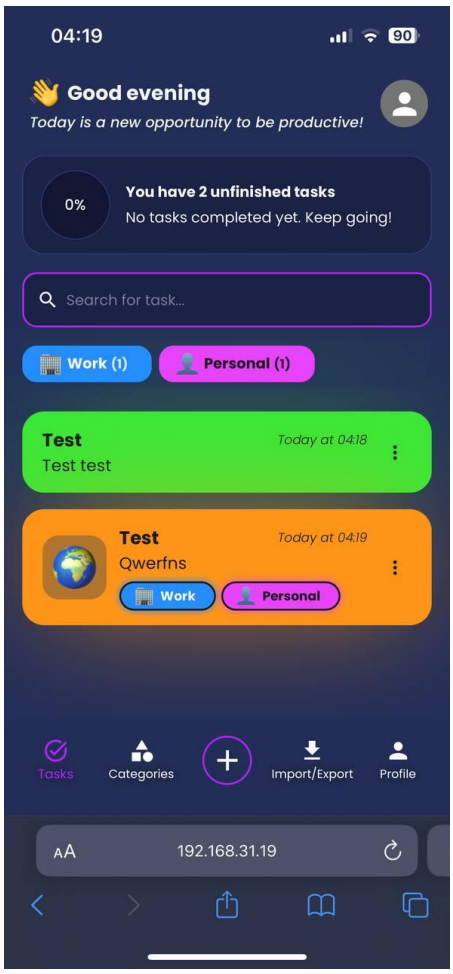

Рисунок *24 — WEB-додаток, мобільна версія*

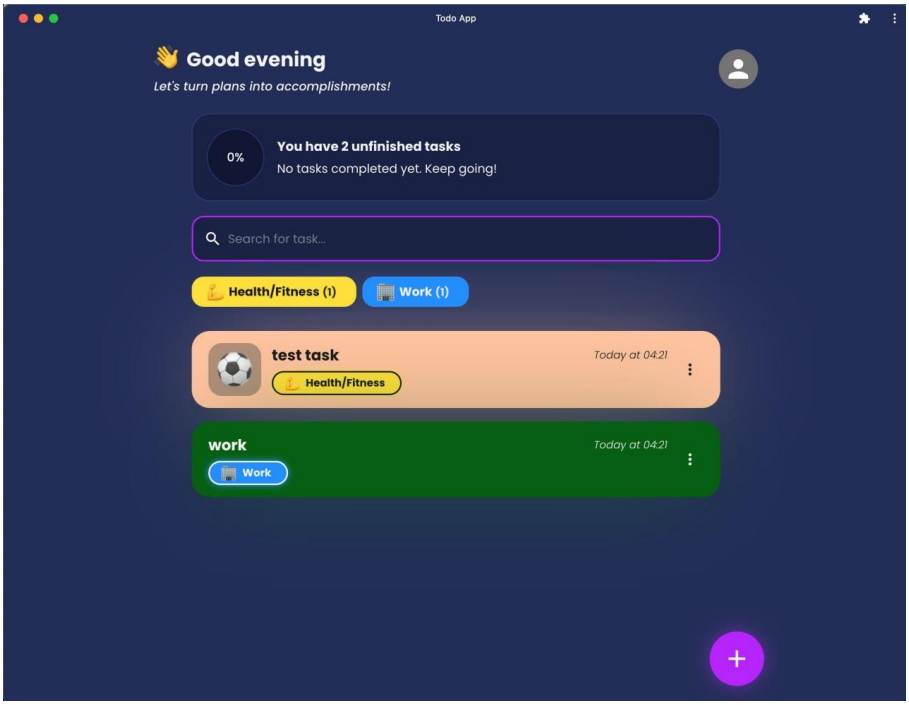

Рисунок *25 — PWA-додаток, десктоп версія*

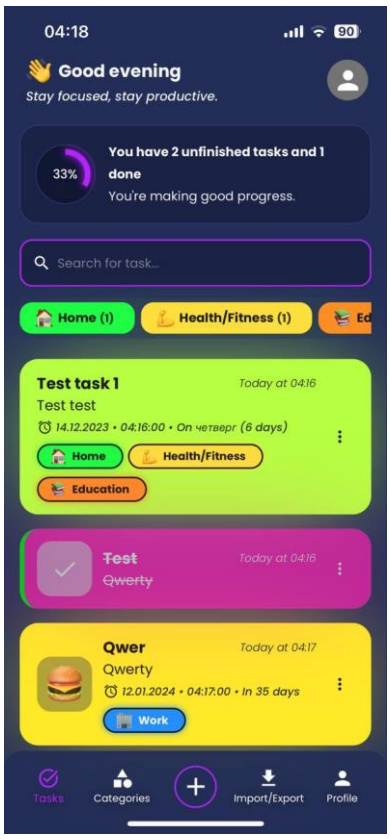

Рисунок *26 — PWA-додаток, десктоп версія*

На аналогічних девайсах, аналогічні типи додатків, виглядають практично однаково. Єдина відмінність, це те, що в PWA зникає вікно браузера.

Якщо порівнювати десктоп версію з мобільною, одразу можна помітити, що в десктоп версії зникає навігаційна панель знизу. В десктоп версії навігаційна панель з’являється після натискання на фото профіля в правому верхньому кутку.

5.7 Висновки до розділу 5

Як висновок завдяки інтеграції vite-plugin-pwa у розроблюваний проєкт, забезпечується ефективна реалізація PWA-підтримки, що дозволяє вдосконалити користувацький досвід, забезпечити можливість роботи в offline-режимі та підвищити доступність додатка для користувачів.

ВИСНОВКИ

Зі зростанням популярності прогресивних веб-додатків настав час дослідити відмінності між PWA та іншими підходами до мобільного користувацького досвіду.

PWA намагаються захопити функціональність, яка за замовчуванням увімкнена у нативних додатках, створюючи нові стандарти та впроваджуючи їх у веб-переглядачі. PWA можуть замінити нативні програми, якщо основні функції програми можна розробити, використовуючи наявну функціональність PWA. Прогресивні веб-додатки можуть бути ідеальним вибором у певних сценаріях, оскільки вони не пропонують просунуту обробку графіки, а загальна складність розробки є нижчою.

Підходи PWA і гібридних додатків дуже схожі, але існують важливі архітектурні відмінності. Гібридні додатки можуть використовувати апаратні/програмні можливості пристрою за допомогою плагінів, що дає перевагу з точки зору кількості функцій, які можна включити, та обсягу доступних обчислювальних ресурсів.

Як і у випадку з нативними додатками, PWA можна використовувати замість гібридних підходів, якщо додаток не покладається на просунутий рендеринг і знаходиться в межах поточних можливостей PWA. Крім того, важливою перевагою використання PWA-підходу є те, що цей веб-додаток, який знижує вартість і може бути використаний для SEO.

У цій роботі проаналізовано літературу на цю тему і показано, що масштабних досліджень прогресивних веб-додатків ще не проводилося.

Так чи інакше, визначення прогресивного веб-додатку не є зрозумілим для багатьох людей. Визначення варіюються від суто технічних до тих, що описують нетехнічні аспекти, такі як користувацький досвід. Один з основних висновків полягає в тому, що PWA можна розглядати як набір хороших практик, яких слід дотримуватися при створенні веб-додатків. Це має

найбільш очевидний вплив на мобільних користувачів з точки зору продуктивності та доступної функціональності. Це тому, що PWA є хорошим стандартом, якого слід дотримуватися при створенні веб-додатків. Він не вимагає багато роботи, а користувачі можуть побачити помітні покращення.

За результатами досліджень, можна зробити висновок, що при використанні таких додатків, як планер найбільш зручно використовувати додаток PWA, оскільки значущими характеристиками є:

1. всі функції повинні бути інтуїтивно зрозумілими для кожного користувача. Важливо, щоб люди могли миттєво знайти потрібну кнопку, легко орієнтуватися у функціях і бачити всі доступні опції з першого погляду.
2. швидкість завантаження додатків, користувачам потрібно якомога менше кліків при завантаженні сервісу.
3. дуже важливо, щоб такий сервіс працював в офлайн-режимі. Важлива швидка синхронізація з діями користувача та оновленнями.

У світлі вищесказаного, здатність PWA бути об'єднуючою технологією для розробки мобільних додатків була детально проаналізована. Наразі не можна сказати, що PWA задовольняє більшість вимог до уніфікованої кросплатформної розробки. Крім того, в найближчому майбутньому, ймовірно, з'являться інші цінні технології.

Як мінімум, PWA можуть сприяти багатшому досвіду розробки, а отже, і створенню кращих додатків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Журнал “IPTEK The Journal for Technology and Science” URL: https://iptek.its.ac.id/index.php/jts/article/view/13904 (дата звернення: 04.10.2023).

2. Такур П. Evaluation and Implementation of Progressive Web Application: бакалаврська робота. Гельсінський університет прикладних наук, 2018.

3. Progressive web apps (PWAs). developer.mozilla.org. URL: https:// developer.mozilla.org/en-US/docs/Web/Progressive_web_apps (дата звернення: 04.10.2023).

4. Франкстон Б. Прогресивні веб-додатки. IEEE Consumer Electronics Magazine. 2018. С. 106.

5. Фортунато Д., Бернардіно Х. Прогресивні веб-додатки: Альтернатива нативним мобільним додаткам. 2018 р.

6. Бржоушек П. Оцінка та використання технології Google Progressive Web Apps : бакалаврська робота. Брно. 2017.

7. Франссон Р. Comparing Progressive Web Applications with Native Android Applications : магістерська робота. Vaxjo. 2017.

8. What is PWA? Progressive Web Apps in eCommerce Explained. URL: https://vuestorefront.io/pwa (дата звернення: 09.10.2023).

9. Samsung. PWA Summit. URL: https://developer.samsung.com/internet/ events/en-us/2021/10/06/pwa-summit.

10. Progressive Web Apps: Is PWA the Future of Web Design? // brainhub. URL: https://brainhub.eu/library/is-pwa-the-future.

11. Why Progressive Web Apps Are The Future of Mobile Web. URL: https://medium.com/ymedialabs-innovation/why-progressive-web-apps-are-the-future-of-mobile-web-5195f381d856.

12. The current state of progressive web apps. URL: https://www2.stardust-testing.com/en/the-current-state-of-progressive-web-apps.

13. Biørn-Hansen A., Grønli T.-M., Majchrzak T.A. Progressive Web Apps: The possible web-native unifier for mobile development // Міжнародна конференція з веб-інформаційних систем та технологій. 2017.

14. Штайнер Т. Аналіз особливостей прогресивних веб-додатків, коли засобом веб-доступу не є веб-браузер. 2021.

15. Франкстон Б. Прогресивні веб-додатки // IEEE Consumer Electronics Magazine. 2022.

16. Франссон Р. Дріагін О. Порівняння прогресивних веб-додатків з нативними додатками Android.

17. Теплий Я.Б.К. Гібридний мобільний додаток для проєкту. Чеський технічний університет. Прага. 2016.

18. Фортунато Д. Бернардіно Х. Progressive web apps: An alternative to the native mobile Apps // Conference on Information Systems and Technologies (CISTI):13th Iberian Conference on IEEE. 2010.

Декларація
академічної доброчесності
здобувача ступеня вищої освіти ЗНУ

Я, Чорний Богдан Олегович, студент 2 курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти ipz18bd-10@stu.zsea.edu.ua,

— підтверджую, що написана мною кваліфікаційна робота на тему «**Порівняльний аналіз PWA-додатків з нативними Web- і мобільними-додатками**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

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

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

Дата _____ Підпис _____ студент: ___Чорний Б.О._____

Дата _____ Підпис _____ науковий керівник: ___Полякова Н. П._