ДОДАТОК А

Перелік джерел посилання за науковими напрямами керівника та науковців кафедри програмної інженерії

4. Кобзєв В., Зеленська Ю. Дослідження архітектурних рішень користувацького інтерфейсу для ефективного використання мобільних додатків. 27-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті», м. Харків. 2023. С. 125–126.

14. Груздо І., Россоха С. Современные проблемы кроссплатформенности в технологиях веб-разработки для мобильных приложений. Інформаційні та моделюючі технології : Всеукр. наук.-практ. конф., м. Черкаси, 29–31 трав. 2014 р. 2014. С. 10.

22. Афанасьєва І., Корецький О. Використання реактивного програмування у розробці мобільних застосунків. SCIENCE, RESEARCH, DEVELOPMENT. Technics and technology.#4 (НАУКА, ИССЛЕДОВАНИЯ, РАЗВИТИЕ. Техника и технология. #4), 29–30 квіт. 2018 р. С. 79–83.

ДОДАТОК Б

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

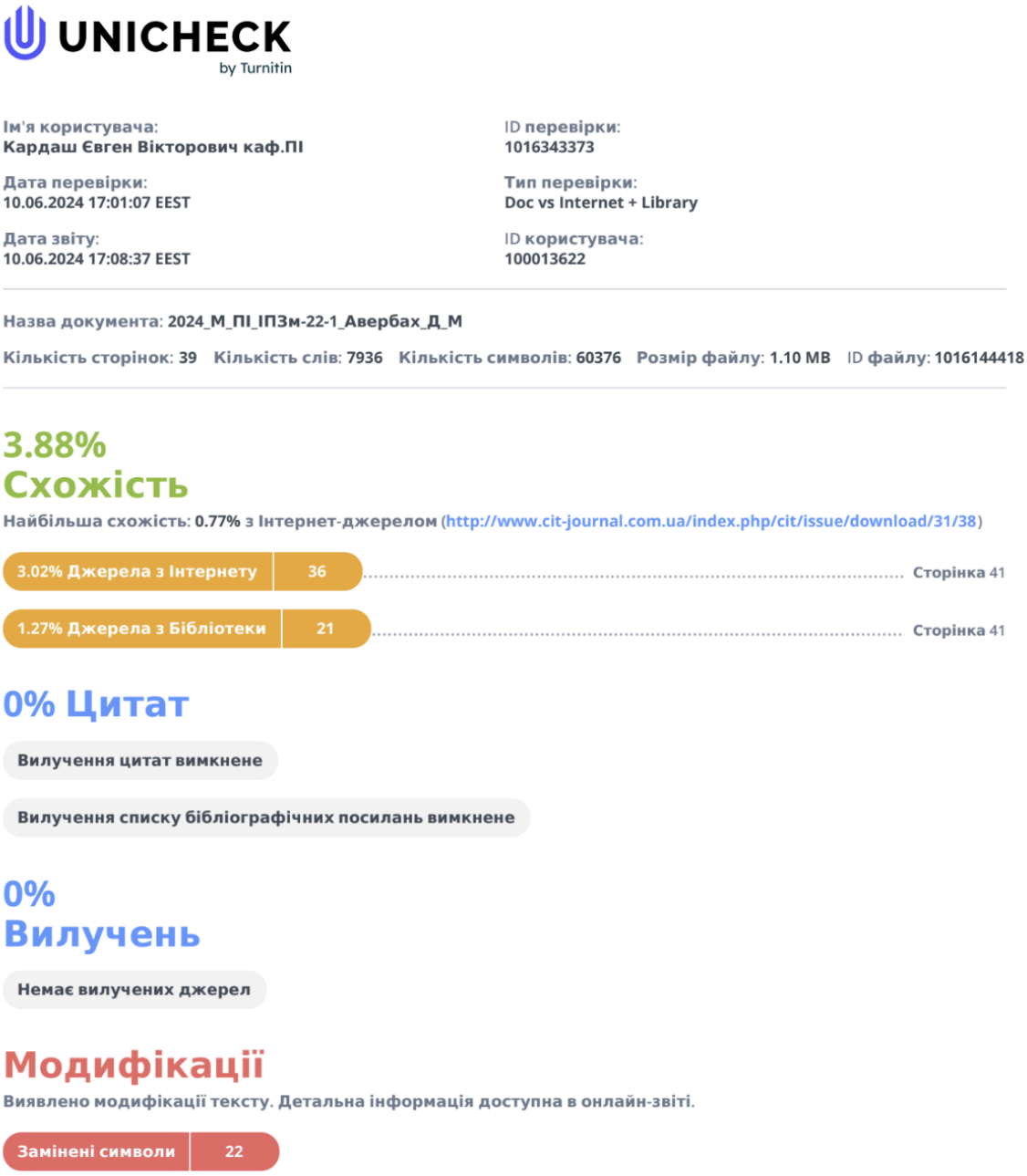


Рисунок Б.1 – Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ (виконаний самостійно)

ДОДАТОК В

Слайди презентації

Дослідження методів розробки прогресивних та гібридних веб додатків порівняно з нативними додатками

Виконав:
ст. гр. ІПЗм-22-1 Авербах Д.М.

Науковий керівник:
доц. кафедри ПІ Русакова Н.Є.

Рисунок В.1 – Слайд 1 (виконаний самостійно)

Актуальність дослідження

- Дослідження методів розробки мобільних додатків є дуже важливою у світлі стрімкого розвитку інформаційних технологій та поширеності мобільних пристроїв серед населення.
- Інновації в мобільній розробці можуть призвести до значного зниження витрат на розробку та підтримку додатків, що є важливим для стартапів та малих бізнесів. Використання ефективних методів розробки дозволяє скоротити час виходу продукту на ринок та зменшити витрати на його обслуговування.
- Вибір виграшної основи серед усіх підходів до розробки додатків є складним завданням. Залежно від потреб проекту, кількості розробників та їх навичок, бюджету та часових рамок, потрібно зробити вибір на користь будь-якої з трьох стратегій.

2

Рисунок В.2 – Слайд 2 (виконаний самостійно)

Прогалини у наявних дослідженнях

- Більшість досліджень базуються на теоретичних моделях або лабораторних умовах, які не завжди відображають реальні сценарії використання додатків.
- Недостатньо досліджено, наскільки ефективно прогресивні та гібридні додатки можуть забезпечити однаковий рівень функціональності та UX на різних платформах.
- Мало досліджень, що порівнюють використання оперативної пам'яті між різними типами додатків.
- Недостатньо порівняльних досліджень часу завантаження додатків.

Рисунок В.3 – Слайд 3 (виконаний самостійно)

Постановка задачі

- Визначити основні методи розробки, що використовуються в прогресивних, гібридних та нативних додатках.
- Проаналізувати інструменти розробки, що використовуються для кожного методу.
- Розробити додатки трьох типів, що розглядаються на прикладі додатку для прогнозу погоди з однаковим функціоналом.
- Провести експериментів і тестів для отримання даних про продуктивність і функціональність кожного методу.
- Порівняти результати, щоб визначити сильні та слабкі сторони кожного методу розробки.
- Визначте випадки використання, в яких певний метод, ймовірно, буде найбільш підходящим.
- Зробити висновки на основі проведеного дослідження.
- Надати рекомендації щодо вибору методології розробки мобільних додатків для конкретних умов та різних типів проектів.

Рисунок В.4 – Слайд 4 (виконаний самостійно)

Визначення нативних додатків

Нативні додатки розробляються спеціально для конкретної платформи (iOS або Android) і використовують мову програмування та інструменти, специфічні для цієї платформи. Вони мають прямий доступ до всіх функцій пристрою і забезпечують найкращу продуктивність та користувацький досвід.

Методи розробки:

- **Objective-C/Swift (для iOS):** Використовуються для розробки додатків під iOS. Розробка ведеться в Xcode.
- **Java/Kotlin (для Android):** Використовуються для розробки додатків під Android. Розробка ведеться в Android Studio.

Рисунок В.5 – Слайд 5 (виконаний самостійно)

Визначення гібридних додатків

Гібридні додатки поєднують елементи нативних та веб-додатків. Вони зазвичай використовують веб-технології для створення інтерфейсу, але запускаються як нативні додатки на мобільних пристроях за допомогою контейнерів.

Методи розробки:

- **HTML, CSS, JavaScript:** Використовуються для створення інтерфейсу та логіки додатка.
- **Frameworks (React Native, Cordova, Ionic):** Використовуються для створення гібридних додатків, які дозволяють запускати веб-додаток всередині нативного контейнера.
- **Plugins:** Містки між веб-технологіями та нативними функціями пристрою (наприклад, доступ до камери, GPS).

Рисунок В.6 – Слайд 6 (виконаний самостійно)

Визначення прогресивних веб-додатків

Прогресивні веб-додатки - це веб-додатки, які використовують сучасні веб-технології для забезпечення досвіду, схожого на нативні додатки. Вони працюють у браузері, але можуть бути встановлені на пристрій і працювати офлайн.

Методи розробки:

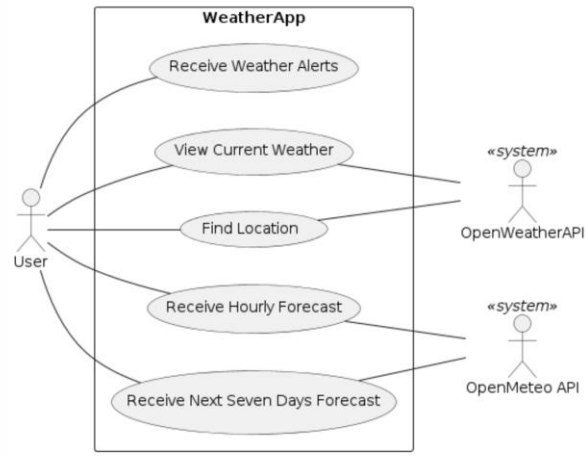
- **HTML, CSS, JavaScript:** Основні веб-технології для створення інтерфейсу та функціоналу.
- **Frameworks (React, Angular, Vue):** Використовують для спрощення розробки.
- **Service Workers:** Скрипти, які працюють у фоновому режимі і дозволяють додатку працювати офлайн та отримувати push-сповіщення.
- **Manifest Files:** Файл, який визначає як PWA виглядатиме і функціонуватиме після встановлення (наприклад, іконка, стартова сторінка).

software engineering

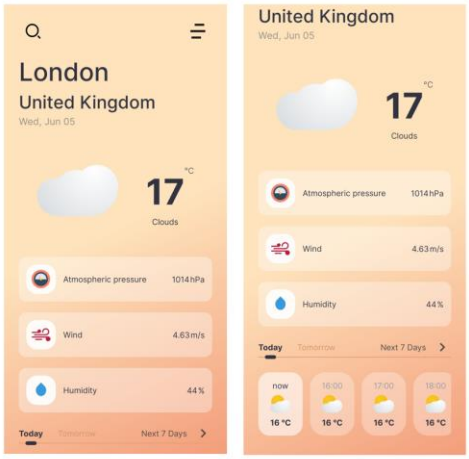
7

Рисунок В.7 – Слайд 7 (виконаний самостійно)

Практична реалізація



Діаграма прецедентів



Додаток для прогнозу погоди (головна сторінка)

software engineering

8

Рисунок В.8 – Слайд 8 (виконаний самостійно)

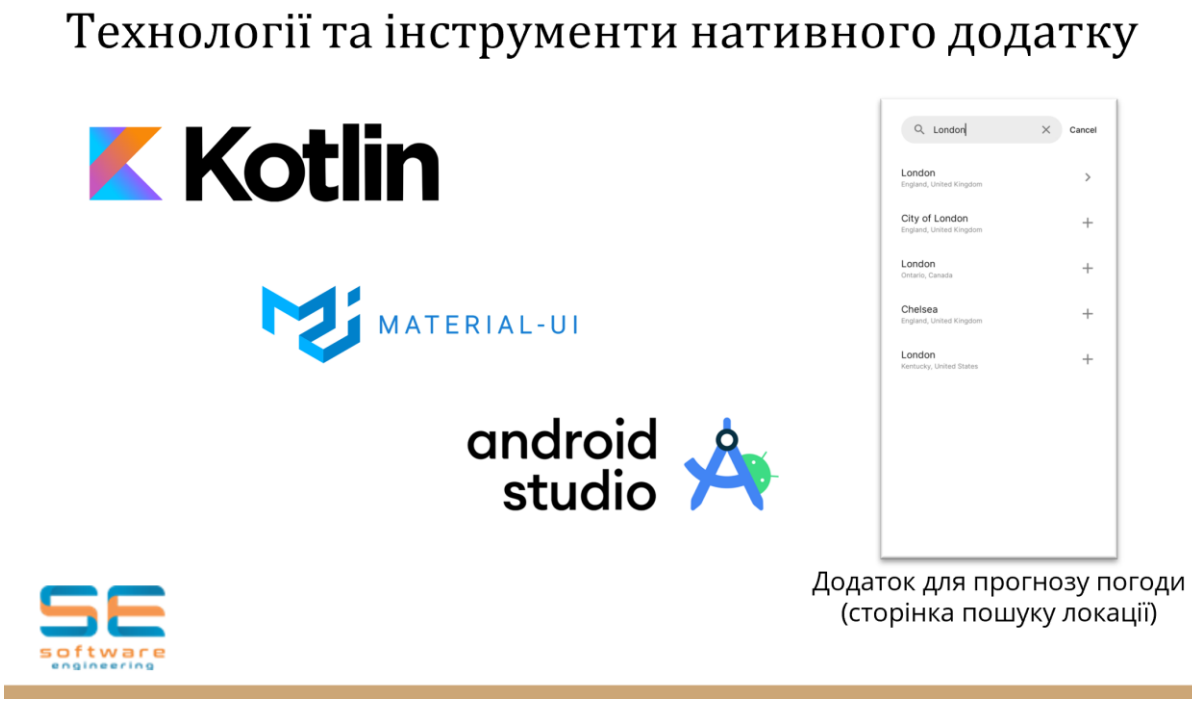


Рисунок В.9 – Слайд 9 (виконаний самостійно)

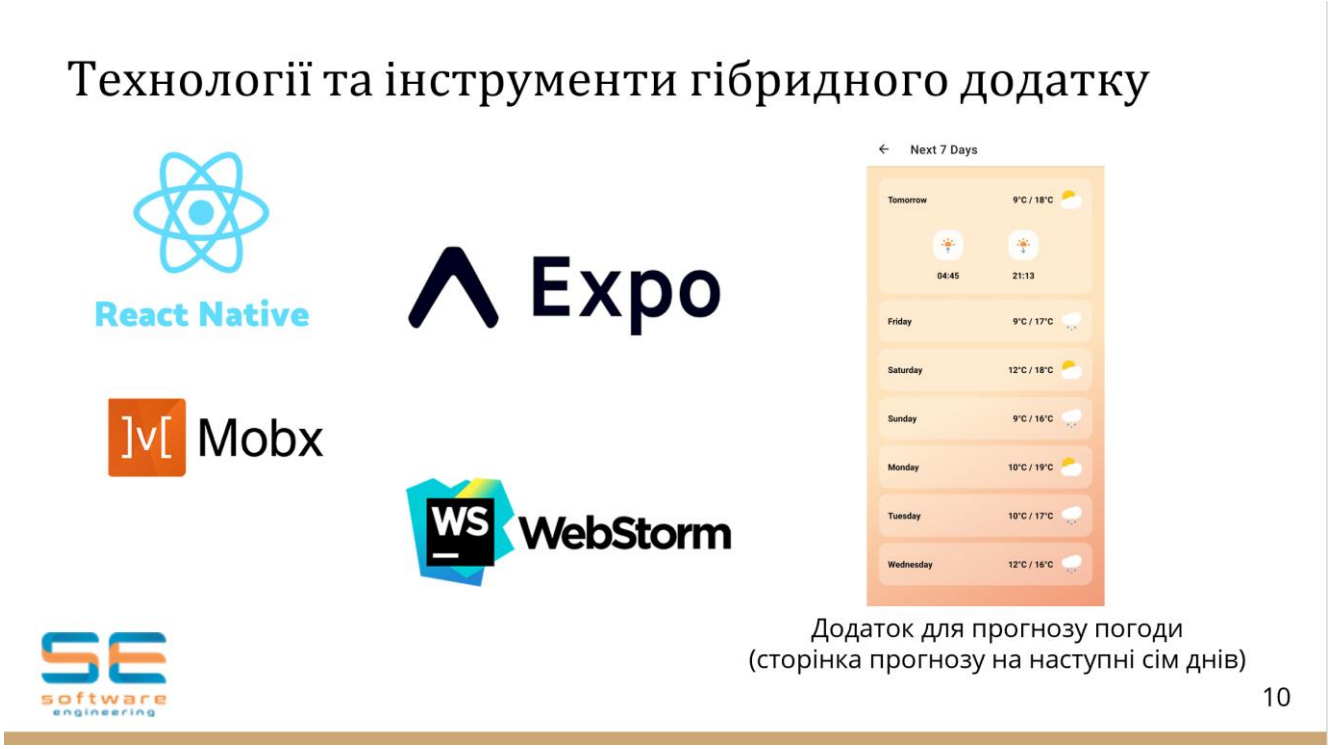


Рисунок В.10 – Слайд 10 (виконаний самостійно)

Технології та інструменти прогресивного додатку

Додаток для прогнозу погоди
(сторінка налаштувань)

11

Рисунок В.11 – Слайд 11 (виконаний самостійно)

Критерії порівняння додатків

- порівняння продуктивності;
- порівняння користувацького досвіду;
- порівняння доступності;
- порівняння використання в автономному режимі;
- порівняння складності підтримки;
- порівняння безпечності.

12

Рисунок В.12 – Слайд 12 (виконаний самостійно)

Параметри продуктивності

$$TTI = T_{interactive} - T_{start\_load}$$

де TTI – час від початку завантаження до моменту, коли додаток стає повністю інтерактивним,

$T_{interactive}$ – час до досягнення інтерактивного стану,

T_{start_load} – час початку завантаження.

Час до першої взаємодії (Time to Interactive)

$$M_{usage} = M_{total} - M_{free}$$

Споживання пам'яті (Memory Usage)

де M_{usage} – споживана пам'яті,

M_{total} – загальна доступна пам'ять,

M_{free} – вільна пам'ять.

$$CPU_{usage} = \frac{CPU_{active}}{CPU_{total}} \times 100\%$$

де CPU_{usage} – відсоток використання процесора,

CPU_{active} – час активної роботи процесора,

CPU_{total} – загальний час роботи процесора.

Середній відсоток використання процесору (CPU Usage)

13

Рисунок В.13 – Слайд 13 (виконаний самостійно)

Інструменти для порівняння продуктивності

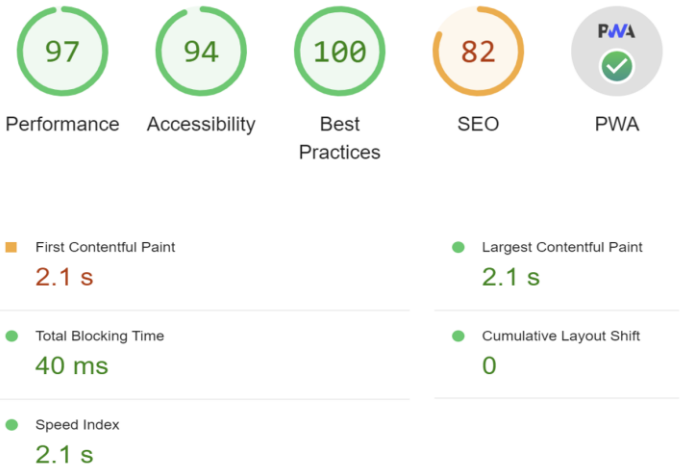
Lighthouse — **для прогресивного**

apptim — **для нативного та гібридного**

14

Рисунок В.14 – Слайд 14 (виконаний самостійно)

Результати тестування прогресивного додатку за допомогою Lighthouse



15

Рисунок В.15 – Слайд 15 (виконаний самостійно)

Результат тестування продуктивності додатків

Параметр / Тип додатку	Розмір додатку	Середнє використання пам'яті	Час до першої взаємодії	Середній відсоток використання процесору
Нативний	6.5 MB	60 MB	2.5 s	36%
Гібридний	69.2 MB	115 MB	3.5 s	40%
Прогресивний	738 KB	7 MB	2.1 s	25%

16

Рисунок В.16 – Слайд 16 (виконаний самостійно)

Загальне порівняння додатків

Тип додатку / Параметр	Нативний	Гібридний	Прогресивний
Вартість розробки	Висока	Середня	Низька
Час розробки	Довгий	Середній	Швидкий
Продуктивність	Найвища	Середня	Залежить від браузера
Користувацький досвід	Найвищий	Середній	Низький
Доступності	Найвища	Середня	Середня
Доступ до апаратних ресурсів	Повний	Повний	Обмежений
Автономна робота	Повна	Часткова	Часткова
Можливість підтримки	Через магазини додатків	Через магазини додатків	Автоматичне оновлення
Безпека	Висока	Висока	Середня

17

Рисунок В.17 – Слайд 17 (виконаний самостійно)

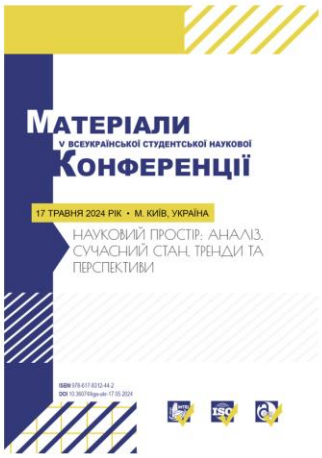
Аналіз отриманих результатів

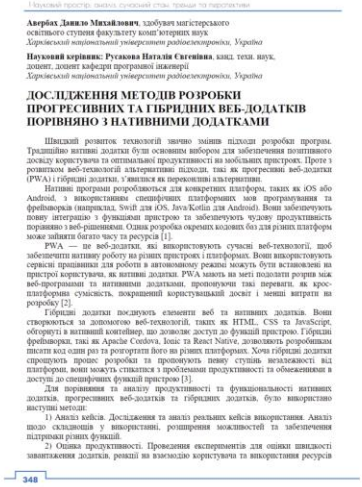
- Нативна розробка, забезпечує високий рівень продуктивності та оптимізації завдяки тому, що вона орієнтована на конкретну платформу. Використовуючи специфічні мови програмування та API, розробники можуть максимально розширити функціональність пристрою. Однак такий підхід вимагає багато зусиль і витрат, а також обмежує швидкість випуску оновлень.
- Прогресивні та гібридні додатки, з іншого боку, намагаються знайти компроміс між ефективністю та універсальністю. Вони можуть використовувати спільну кодову базу на різних платформах, що зменшує зусилля та витрати. Однак це може вплинути на продуктивність і взаємодію з функціоналом пристрою.
- Вибір підходу до розробки повинен бути обґрунтований відповідно до конкретних потреб та обставин проекту. Нативні підходи підходять для проектів, які вимагають високої ефективності та повної функціональності, тоді як прогресивні або гібридні рішення можуть бути кращими для більш універсальних та економічно ефективних проектів.

18

Рисунок В.18 – Слайд 18 (виконаний самостійно)

Апробація результатів





Авербах Данило Михайлович, здобувач магістерського освітнього ступеня факультету комп'ютерних наук
Харківський національний університет радіоелектроніки, Україна

Науковий керівник: Русакова Наталія Євгенівна, канд. техн. наук, доцент, доцент кафедри програмної інженерії
Харківський національний університет радіоелектроніки, Україна

ДОСЛІДЖЕННЯ МЕТОДІВ РОЗРОБКИ ПРОГРЕСИВНИХ ТА ГІБРИДНИХ ВЕБ-ДОДАТКІВ ПОРІВНЯНО З НАТИВНИМИ ДОДАТКАМИ

Швидкий розвиток технологій значно змінив підходи розробки програм. Традиційно нативні додатки були основним вибором для забезпечення позитивного досвіду користувача та оптимальної продуктивності на мобільних пристроях. Проте з розвитком веб-технологій альтернативні підходи, такі як прогресивні веб-додатки (PWA) і гібридні додатки, з'явилися як переконливі альтернативи.

Нативні програми розробляються для конкретних платформ, таких як iOS або Android, з використанням специфічних платформних мов програмування та фреймворків (наприклад, Swift для iOS, Java/Kotlin для Android). Вони забезпечують повну інтеграцію з функціями пристрою та забезпечують чудову продуктивність порівняно з веб-рішеннями. Однак розробка окремих кодових баз для різних платформ може зайняти багато часу та ресурсів [1].

PWA — це веб-додатки, які використовують сучасні веб-технології, щоб забезпечити нативну роботу на різних пристроях і платформах. Вони використовують сервісні працівники для роботи в автономному режимі можуть бути встановлені на пристрої користувача, як нативні додатки. PWA мають на меті подолати розрив між веб-програмами та нативними додатками, пропонуючи такі переваги, як крос-платформна сумісність, покращений користувацький досвід і менші витрати на розробку [2].

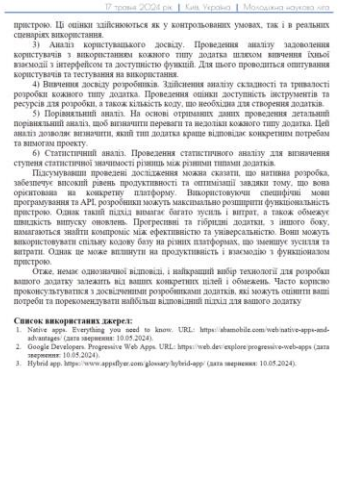
Гібридні додатки поєднують елементи веб та нативних додатків. Вони створюються за допомогою веб-технологій, таких як HTML, CSS та JavaScript, обгорнуті в нативний контейнер, що дозволяє доступ до функцій пристрою. Гібридні фреймворки, такі як Apache Cordova, Ionic та React Native, дозволяють розробникам писати код один раз та розгортати його на різних платформах. Хоча гібридні додатки спрощують процес розробки та пропонують певну ступінь незалежності від платформи, вони можуть стикатися з проблемами продуктивності та обмеженнями в доступі до специфічних функцій пристрою [3].

Для порівняння та аналізу продуктивності та функціональності нативних додатків, прогресивних веб-додатків та гібридних додатків, було використано наступні методи:

1) Аналіз кейсів. Дослідження та аналіз реальних кейсів використання. Аналіз щодо складнощів у використанні, розширення можливостей та забезпечення підтримки різних функцій.

2) Оцінка продуктивності. Проведення експериментів для оцінки швидкості завантаження додатків, реакції на взаємодію користувача та використання ресурсів

348



пристрою. Ці оцінки здійснюються як у контрольованих умовах, так і в реальних сценаріях використання.

3) Аналіз користувацького досвіду. Проведення аналізу задоволення користувачів з використанням кожного типу додатка шляхом вивчення їхньої взаємодії з інтерфейсом та доступністю функцій. Для цього проводиться опитування користувачів та тестування на використання.

4) Вивчення досвіду розробників. Здійснення аналізу складності та тривалості розробки кожного типу додатка. Проведення оцінки доступність інструментів та ресурсів для розробки, а також кількість коду, що необхідна для створення додатків.

5) Порівняльний аналіз. На основі отриманих даних проведення детальний порівняльний аналіз, щоб визначити переваги та недоліки кожного типу додатка. Цей аналіз дозволяє визначити, який тип додатка краще відповідає конкретним потребам та вимогам проекту.

6) Статистичний аналіз. Проведення статистичного аналізу для визначення ступеня статистичної значимості різниць між різними типами додатків.

Підсумовуючи проведені дослідження можна сказати, що нативна розробка, забезпечує високий рівень продуктивності та оптимізації завдяки тому, що вона орієнтована на конкретну платформу. Використовуючи специфічні мови програмування та API, розробники можуть максимально розширити функціональність пристрою. Однак такий підхід вимагає багато зусиль і витрат, а також обмежує швидкість випуску оновлень. Прогресивні та гібридні додатки, з іншого боку, намагаються знайти компроміс між ефективністю та універсальністю. Вони можуть використовувати спільну кодову базу на різних платформах, що зменшує зусилля та витрати. Однак це може вплинути на продуктивність і взаємодію з функціоналом пристрою.

Отже, немає однозначної відповіді, і найкращий вибір технології для розробки вашого додатку залежить від ваших конкретних цілей і обмежень. Часто корисно проконсультуватися з досвідченими розробниками додатків, які можуть оцінити ваші потреби та порекомендувати найбільш відповідний підхід для вашого додатку

Список використаних джерел:

1. Native apps. Everything you need to know. URL: https://ubamobile.com/web/native-apps-and-advantages/ (дата звернення: 10.05.2024).
2. Google Developers. Progressive Web Apps. URL: https://web.dev/explore/progressive-web-apps (дата звернення: 10.05.2024).
3. Hybrid app. https://www.appsflyer.com/glossary/hybrid-app/ (дата звернення: 10.05.2024).

software engineering

19

Рисунок В.19 – Слайд 19 (виконаний самостійно)

Перспектива подальших досліджень

Подальші дослідження можуть бути спрямовані на оптимізацію досліджених підходів для підвищення їх ефективності та зменшення часу розробки і впровадження. Можливим напрямком є розробка нових гібридних архітектур, які поєднують найкращі властивості прогресивних веб-додатків та нативних додатків. Впровадження таких рішень у різні галузі може значно покращити якість та швидкість створення програмного забезпечення, забезпечуючи високу продуктивність і зручність для кінцевих користувачів.

software engineering

20

Рисунок В.20 – Слайд 20 (виконаний самостійно)

ДОДАТОК Г

Апробація результатів роботи

Авербах Данило Михайлович, здобувач магістерського
освітнього ступеня факультету комп'ютерних наук
Харківський національний університет радіоелектроніки, Україна

Науковий керівник: Русакова Наталія Євгенівна, канд. техн. наук,
доцент, доцент кафедри програмної інженерії
Харківський національний університет радіоелектроніки, Україна

ДОСЛІДЖЕННЯ МЕТОДІВ РОЗРОБКИ ПРОГРЕСИВНИХ ТА ГІБРИДНИХ ВЕБ-ДОДАТКІВ ПОРІВНЯНО З НАТИВНИМИ ДОДАТКАМИ

Швидкий розвиток технологій значно змінив підходи розробки програм. Традиційно нативні додатки були основним вибором для забезпечення позитивного досвіду користувача та оптимальної продуктивності на мобільних пристроях. Проте з розвитком веб-технологій альтернативні підходи, такі як прогресивні веб-додатки (PWA) і гібридні додатки, з'явилися як переконливі альтернативи.

Нативні програми розробляються для конкретних платформ, таких як iOS або Android, з використанням специфічних платформних мов програмування та фреймворків (наприклад, Swift для iOS, Java/Kotlin для Android). Вони забезпечують повну інтеграцію з функціями пристрою та забезпечують чудову продуктивність порівняно з веб-рішеннями. Однак розробка окремих кодових баз для різних платформ може зайняти багато часу та ресурсів [1].

PWA — це веб-додатки, які використовують сучасні веб-технології, щоб забезпечити нативну роботу на різних пристроях і платформах. Вони використовують сервісні працівники для роботи в автономному режимі можуть бути встановлені на пристрої користувача, як нативні додатки. PWA мають на меті подолати розрив між веб-програмами та нативними додатками, пропонуючи такі переваги, як крос-платформна сумісність, покращений користувацький досвід і менші витрати на розробку [2].

Гібридні додатки поєднують елементи веб та нативних додатків. Вони створюються за допомогою веб-технологій, таких як HTML, CSS та JavaScript, обгорнуті в нативний контейнер, що дозволяє доступ до функцій пристрою. Гібридні фреймворки, такі як Apache Cordova, Ionic та React Native, дозволяють розробникам писати код один раз та розгортати його на різних платформах. Хоча гібридні додатки спрощують процес розробки та пропонують певну ступінь незалежності від платформи, вони можуть стикатися з проблемами продуктивності та обмеженнями в доступі до специфічних функцій пристрою [3].

Для порівняння та аналізу продуктивності та функціональності нативних додатків, прогресивних веб-додатків та гібридних додатків, було використано наступні методи:

1) Аналіз кейсів. Дослідження та аналіз реальних кейсів використання. Аналіз щодо складнощів у використанні, розширення можливостей та забезпечення підтримки різних функцій.

2) Оцінка продуктивності. Проведення експериментів для оцінки швидкості завантаження додатків, реакції на взаємодію користувача та використання ресурсів

пристрою. Ці оцінки здійснюються як у контрольованих умовах, так і в реальних сценаріях використання.

3) Аналіз користувацького досвіду. Проведення аналізу задоволення користувачів з використанням кожного типу додатка шляхом вивчення їхньої взаємодії з інтерфейсом та доступністю функцій. Для цього проводиться опитування користувачів та тестування на використання.

4) Вивчення досвіду розробників. Здійснення аналізу складності та тривалості розробки кожного типу додатка. Проведення оцінки доступність інструментів та ресурсів для розробки, а також кількість коду, що необхідна для створення додатків.

5) Порівняльний аналіз. На основі отриманих даних проведення детальний порівняльний аналіз, щоб визначити переваги та недоліки кожного типу додатка. Цей аналіз дозволяє визначити, який тип додатка краще відповідає конкретним потребам та вимогам проекту.

6) Статистичний аналіз. Проведення статистичного аналізу для визначення ступеня статистичної значимості різниць між різними типами додатків.

Підсумувавши проведені дослідження можна сказати, що нативна розробка, забезпечує високий рівень продуктивності та оптимізації завдяки тому, що вона орієнтована на конкретну платформу. Використовуючи специфічні мови програмування та API, розробники можуть максимально розширити функціональність пристрою. Однак такий підхід вимагає багато зусиль і витрат, а також обмежує швидкість випуску оновлень. Прогресивні та гібридні додатки, з іншого боку, намагаються знайти компроміс між ефективністю та універсальністю. Вони можуть використовувати спільну кодову базу на різних платформах, що зменшує зусилля та витрати. Однак це може вплинути на продуктивність і взаємодію з функціоналом пристрою.

Отже, немає однозначної відповіді, і найкращий вибір технології для розробки вашого додатку залежить від ваших конкретних цілей і обмежень. Часто корисно проконсультуватися з досвідченими розробниками додатків, які можуть оцінити ваші потреби та порекомендувати найбільш відповідний підхід для вашого додатку

Список використаних джерел:

1. Native apps. Everything you need to know. URL: https://abamobile.com/web/native-apps-and-advantages/ (дата звернення: 10.05.2024).
2. Google Developers. Progressive Web Apps. URL: https://web.dev/explore/progressive-web-apps (дата звернення: 10.05.2024).
3. Hybrid app. https://www.appsflyer.com/glossary/hybrid-app/ (дата звернення: 10.05.2024).

ДОДАТОК Д

Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008: 2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-22-1 (група)

Авербах Д.М.
(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні познаки	
	7.15 Додатки	

Експерт _____ (підпис) Вадим НЕЧВОЛОД (прізвище, ініціали)

14.06.2024

Рисунок Д.1 – Експертний висновок результатів перевірки кваліфікаційної роботи