БАКАЛАВРСЬКА РОБОТА

ДРБ. ІП - 69.00.00.000 ПЗ

Група ІП-21-3

Данильців Лілія

2025

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Данильців Лілія Миколаївна

(прізвище, ім'я, по батькові)

УДК 004.942
(індекс)

БАКАЛАВРСЬКА РОБОТА

«Методології крос-платорформенної розробки»
(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня _____ Данильців Л. М. _____
(підпис, ініціали та прізвище здобувача)

Науковий керівник ____Ваврик Тетяна Олександрівна, асистент_____
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

_доц._____Бандура В.В.__
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2025

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доцент. В.В. Бандура
“ ____ ” _____2025 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ БАКАЛАВРА СТУДЕНТОВІ

Данильців Лілії Миколаївні

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) "Методології крос-платорформенної розробки"

керівник проекту (роботи) асистент Ваврик Т.О.

затверджені наказом вищого навчального закладу від “ 28 ” квітня 2025 р. № 264/7

2. Строк подання студентом проекту (роботи) 10 червня 2025 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

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

1. Аналіз кросплатформної мобільної розробки ..
2. Структура оцінювання для cpdt .....
3. Реалізація та експерименти

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

1. Програма BioChem Euchre Deck на BlackBerry (ліворуч), Android (у центрі) та iOS (праворуч) (рис.1.1, ст.13)
2. Архітектура IBM Worklight (рис.1.2, ст.21)
3. Структура оцінки CPDT28 (рис.2.1, ст.28)
4. Процедура порівняльного аналізу (рис.2.2, ст.34).
5. Процедура перевірки вхідних даних (рис.3.2, ст.44)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

2. Дата видачі завдання 15 лютого 2025 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Визначення та обґрунтування теми роботи	15.02.2025	виконано
2	Огляд існуючих концепцій, рішень та сервісів в даній області	25.02.2025	виконано
3	Побудова моделі або алгоритму власного рішення	15.03.2025	виконано
4	Документування реалізації власного оригінального рішення вибраними засобами	25.04.2025	виконано
5	Оформлення пояснювальної записки бакалаврської роботи	10.06.2025	виконано

Студент _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Бакалаврська робота містить 60 сторінок, 12 рисунків, 7 таблиць, список використаних джерел із 20 найменування,

Метою роботи оцінити ефективність різних CPDT при розробці мобільних додатків для пошуку книг та надати рекомендації щодо вибору найкращого інструменту для цього типу додатків, враховуючи продуктивність, функціональність, зручність інтеграції та підтримку різних платформ.

Об'єкт дослідження: Кросплатформні інструменти для мобільної розробки (CPDT) та їх ефективність у створенні програмних рішень для мобільних додатків, таких як "Каталог для пошуку книг"..

Предмет дослідження: Процес вибору, порівняння та оцінки ефективності кросплатформних інструментів для мобільної розробки, зокрема для створення додатка, що дозволяє користувачам знаходити книги в каталогах.

Результати дослідження: У ході дослідження було оцінено кілька CPDT, і зроблено кілька важливих висновків щодо їх переваг і недоліків.

В першому розглянуто теоретичні основи крос-платформної розробки, включаючи принципи роботи гібридних і нативних підходів, архітектурні моделі та особливості інтеграції з апаратними можливостями пристроїв

В Другому розділі розглядається структура оцінювання CPDT, включаючи критерії та показники продуктивності,

В Третьому розділі реалізація експериментів, оцінка ефективності та рекомендації щодо вибору інструментів для кросплатформної розробки.

Висновок: вибір інструменту за допомогою CPDT для розробки мобільних додатків повинен базуватись на конкретних вимогах проекту, таких як продуктивність, підтримка специфічних функцій платформ і бюджет.

КЛЮЧОВІ СЛОВА: КРОС-ПЛАТФОРМНА РОЗРОБКА, FLUTTER, REACT NATIVE, XAMARIN, IONIC, .NET MAUI, ПРОДУКТИВНІСТЬ, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС, ПОРІВНЯЛЬНИЙ АНАЛІЗ, МОБІЛЬНІ ДОДАТКИ.

ANNOTATION

The bachelor's thesis contains 60 pages, 12 figures, 7 tables, a list of used sources with 20 titles,

The purpose of the work is to evaluate the effectiveness of various CPDTs in the development of mobile applications for finding books and to provide recommendations on choosing the best tool for this type of application, taking into account performance, functionality, ease of integration and support for various platforms.

Research object: Cross-platform mobile development tools (CPDTs) and their effectiveness in creating software solutions for mobile applications, such as "Book Search Catalog".

Research object: The process of selecting, comparing and evaluating the effectiveness of cross-platform mobile development tools, in particular for creating an application that allows users to find books in catalogs.

Research results: During the study, several CPDTs were evaluated, and several important conclusions were made regarding their advantages and disadvantages.

The first section discusses the theoretical foundations of cross-platform development, including the principles of hybrid and native approaches, architectural models and features of integration with device hardware capabilities.

The second section discusses the CPDT evaluation structure, including performance criteria and indicators,

The third section discusses the implementation of experiments, performance evaluation and recommendations for choosing tools for cross-platform development.

Conclusion: the choice of a tool using CPDT for mobile application development should be based on specific project requirements, such as performance, support for specific platform features and budget.

KEYWORDS: **CROSS-PLATFORM DEVELOPMENT, FLUTTER, REACT NATIVE, XAMARIN, IONIC, .NET MAUI, PRODUCTIVITY, USER INTERFACE, COMPARATIVE ANALYSIS, MOBILE APPLICATIONS.**

ЗМІСТ

ВСТУП 8

РОЗДІЛ 1. АНАЛІЗ КРОСПЛАТФОРМНОЇ МОБІЛЬНОЇ РОЗРОБКИ ... 11

1.1. Мотивація та передумови 11

1.2. Поточний стан ландшафту кросплатформної розробки 14

1.3. Кросплатформні фреймворки для мобільної розробки 17

1.4 Висновки по розділу……25

РОЗДІЛ 2. СТРУКТУРА ОЦІНЮВАННЯ ДЛЯ CPDT 26

2.1. Каркас 26

2.2. Можливості CPDT 27

2.3. Еталонні показники продуктивності 30

2.4. Обговорення досвіду розробки ..... …..33

1.4 Висновки по розділу……35

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТИ 36

3.1. Параметри експерименту 36

3.2. Еталонні показники продуктивності 38

3.3. Обговорення досвіду розробки 50

3.4. Оцінка ефективності CPDT..... 52

3.5. Висновки та рекомендації щодо вибору CPDT для кросплатформної розробки……..... 55

1.4 Висновки по розділу……61

ВИСНОВКИ…… 63

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……65

БІБЛІОГРАФІЧНА ДОВІДКА

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

JVM - віртуальній машині Java

Java ME - Java Micro Edition

PIM - Personal Information Manager

SDK - комплекти розробки програмного забезпечення

UI - користувацький інтерфейс

UX - користувацький досвід

ВСТУП

Актуальність теми зумовлена тим, що за останні роки світ мобільних операційних систем стає все більш фрагментованим. Кожна компанія або консорціум розробив власний нестандартний метод розробки додатків для своїх пристроїв, використовуючи різноманітні мови програмування та комплекти розробки програмного забезпечення (SDK). Це означає, що програма, створена для операційної системи Google Android, не працюватиме на конкуруючій платформі Apple iOS. Існує подальша фрагментація всередині платформ, оскільки деякі програми не можуть працювати на різних версіях або пристроях однієї платформи, що значно ускладнює розробку.

Фрагментація ринку зумовлена багатьма факторами. Окрім різноманітності програмної платформи, варіації апаратного забезпечення створюють труднощі з перенесенням користувацького досвіду (UX), включаючи метод взаємодії та користувацький інтерфейс (UI) з одного пристрою на інший [3]. Оскільки у 2011 році було продано понад 20 мільйонів планшетних пристроїв, було важко дозволити програмам максимізувати більші екрани цих пристроїв [64]. Багато розробників змушені вибирати підтримку лише деяких платформ і версій через обмежені фінансові ресурси або знання методів кодування для кожної платформи. Без широкої підтримки розробників платформи вважаються слабкими та мають труднощі з маркетингом продуктів. Це стосується нових операційних систем і нових випусків, які порушують сумісність із застарілим програмним забезпеченням.

Прикладом цього є перехід Research in Motion на платформу BB10, що порушує сумісність із власно розробленими програмами BlackBerry OS [57]. Зважаючи на кількість використовуваних мов, розробники повинні бути дуже універсальними, а підприємствам потрібно витрачати значні ресурси, щоб їх програмне забезпечення було доступне на кількох платформах. Навіть з урахуванням додаткових витрат, опитування Appcelerator і IDC за серпень 2012 року показує, що компанії продовжують дуже цікавитися кількома платформами

1 незважаючи на труднощі [7] . У 2011 році розробники виявили інтерес до роботи на удвічі більшій кількості платформ, ніж у аналогічному опитуванні попереднього року, і зараз у середньому отримують неймовірні чотири операційні системи [4] [6]. Ця тенденція продовжувала посилюватися в 2012 році [7]. З огляду на таку велику зацікавленість у доступності програм для багатьох платформ, існує очевидна проблема, з якою стикаються розробники, вимагаючи можливості зробити свої програми доступними для якнайширшої аудиторії, але не вистачає ресурсів для створення нативно для кожної платформи.

Щоб розширити охоплення додатків, не витрачаючи значний час, необхідний для вивчення особливостей кожної платформи, використання кросплатформних засобів розробки (CPDT) може стати відповіддю. Генеральний директор InRuntime сказав, що використання CPDT скоротило час виходу на ринок на 70% [68]. Це показує, що використання кросплатформних інструментів може бути дуже корисним у багатьох відношеннях. Однак не існує адекватних заходів, які б гарантували, що ці CPDT такі ж потужні, як власні інструменти розробки. Через це розробники не впевнені, чи зможуть вони використовувати ці інструменти для створення потужних додатків із нативною функціональністю та продуктивністю.

Метою цього дослідження Оцінити ефективність різних CPDT при розробці мобільних додатків для пошуку книг та надати рекомендації щодо вибору найкращого інструменту для цього типу додатків, враховуючи продуктивність, функціональність, зручність інтеграції та підтримку різних платформ.

Завданнями дослідження є Аналіз поточного стану ландшафту кросплатформної мобільної розробки, оцінка можливостей різних CPDT, експериментальна оцінка ефективності CPDT, порівняння продуктивності різних CPDT, розробка рекомендацій.

Результати дослідження: показали, що вибір інструменту для кросплатформної мобільної розробки має значний вплив на ефективність і продуктивність розробки додатків, таких як "Каталог для пошуку книг". У ході дослідження було оцінено кілька CPDT, і зроблено кілька важливих висновків

щодо їх переваг і недоліків.

Об'єкт дослідження: Кросплатформні інструменти для мобільної розробки (CPDT) та їх ефективність у створенні програмних рішень для мобільних додатків, таких як "Каталог для пошуку книг".

Предметом дослідження: Процес вибору, порівняння та оцінки ефективності кросплатформних інструментів для мобільної розробки, зокрема для створення додатка, що дозволяє користувачам знаходити книги в каталогах.

Методи дослідження: були використанні такі аналіз літератури, метод порівняння, моделювання, оцінки продуктивності, експериментальне тестування, ці методи допоможуть здійснити комплексну оцінку ефективності та доцільності використання різних CPDT у розробці мобільних додатків.

Бакалаврська робота містить три розділи, 60 сторінок, 12 рисунків, 7 таблиць, список використаних джерел із 30 найменуванням.

РОЗДІЛ 1. АНАЛІЗ КРОСПЛАТФОРМНОЇ МОБІЛЬНОЇ РОЗРОБКИ

1.1 Мотивація та передумови

До початку дослідження в цій дипломній роботі в рамках роботи в CMER ми розробили додаток BioChem Euchre Deck для платформи BlackBerry [19]. Ця програма надає навчальний інтерфейс для студентів, які використовують флеш-картки, створені професором Джоном Доусоном з Університету Гвельфа. Після початкової розробки ми залишилися в роздумах, як найкраще надати програму студентам на всіх платформах. Під час вибору кросплатформного рішення та розробки програми було відмічено ряд недоліків. Відсутність адекватного порівняння цих інструментів дуже ускладнила завдання пошуку інструменту, який би найкраще відповідав нашим вимогам, і підкреслила потребу в методі оцінки. У нашій неструктурованій оцінці PhoneGap розглядався як ідеальний кандидат для простих, широко розгорнутих програм. Програма, показана на рисунку 1.1, була створена та розгорнута. Більш широке охоплення завдяки використанню CPDT дозволило отримати понад 6000 завантажень на Android і 2000 на iOS і BlackBerry, що демонструє переваги кросплатформенності.

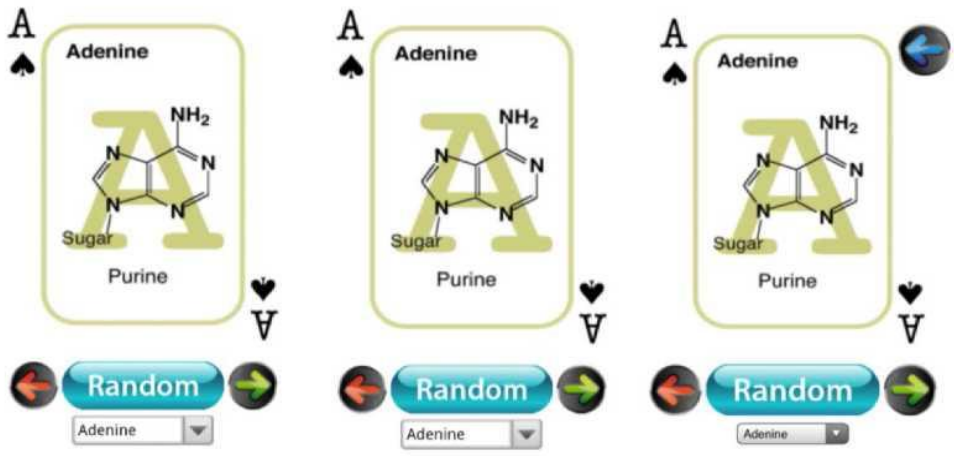

Рисунок 1.1 - Програма BioChem Euchre Deck на BlackBerry (ліворуч), Android (у центрі) та iOS (праворуч)

На жаль, через відсутність доступної оцінки ми не знали, чи відповідав інструмент потребам нашої програми до початку розробки. У службі PhoneGap

Build були очевидні деякі обмеження, наприклад неможливість використовувати фокусований перегляд на платформі BlackBerry, таким чином створюючи програми, які мають курсор миші. Очікується, що це буде вирішено в наступному випуску PhoneGap, але нам доведеться скомпілювати версію програми BlackBerry за допомогою стандартного WebWorks SDK і іншого файлу config.xml, який надає інформацію про програму. Однак саму програму не потрібно було змінювати. Якби у нас була доступна оцінка інструменту PhoneGap до розробки цієї програми, ми б знали, що через це обмеження вона не ідеально підходить для нашої мети.

Було розроблено деякі CPDT і реалізовано численні функції для роботи на багатьох пристроях, але важко визначити, яких функцій бракує. Значна частина інформації, доступної для цих інструментів, надходить від постачальників, і розробникам доступний невеликий незалежний аналіз. Крім того, не вистачає порівняння між цими фреймворками та їхніми рідними аналогами для розробки. Наскільки добре вони працюють? Ви втрачаєте функціональність? Які найкращі? Оскільки поточних тестів немає, необхідний процес тестування фреймворків один з одним та їхніми рідними аналогами, щоб надати вказівки та відповісти на головне питання пошуку кросплатформного рішення для розробки без компромісів. Необхідно розробити протоколи та інструменти порівняльного аналізу, щоб перевірити багато фреймворків, а також провести дослідження особливостей і недоліків кожного.

Основною метою цього дослідження була розробка системи оцінювання. Це включає інструменти порівняльного аналізу та обговорення отриманих кількісних результатів для кількох CPDT на ринку. Щоб визначити, чи має центральна теза надійне підґрунтя, було зроблено кілька кроків: 1. Перегляньте інструменти, доступні як для нативної, так і для міжплатформної розробки, 2. Відкрийте для себе попередні дослідження оцінки таких інструментів розробки, 3. Створіть структуру оцінювання високого рівня, використовуючи запитання щодо реального розвитку, 4. Розробити контрольні показники для досягнення кількісних результатів і застосувати їх до

кількох CPDT і рідних інструментів, 5. Проаналізуйте результати, зробіть висновки. Хоча CPDT наразі існують, досі невідомо, чи відповідають ці інструменти вимогам щодо продуктивності та функцій, щоб бути порівнянними з їхніми рідними аналогами. На цю тему було проведено дуже мало досліджень, оскільки всі ці CPDT є досить новими, і більшість аналізів продуктивності є анекдотичними, а не науковими. Вважається, що мобільні веб-платформи CPDT можуть страждати від проблем з продуктивністю, але невідомо, чи так це з огляду на останні досягнення в продуктивності JavaScript [4]. Необхідно розробити систему порівняльного аналізу, щоб перевірити ці CPDT один на одного та нативний код. Питання про розробку будуть отримані шляхом огляду поточних досліджень, опитувань розробників, обговорень з нашим галузевим партнером і виявлення можливостей нативних інструментів. Під час розробки цієї системи оцінювання необхідно було взяти до уваги певні припущення. Очікується, що інструменти матимуть можливість запускати схожі розроблені порівняльні тести, залишаючись кросплатформними. Тести розроблено з використанням технологій, які зручні для планшетів і смартфонів, але зосереджені лише на тестуванні смартфонів. Тестування для планшетів залишилося для подальшої роботи. Зрозуміло, що певні фонові процеси неможливо зупинити та вони можуть вплинути на тестування, тому слід бути обережним, щоб зменшити їхній вплив на результати. Це дослідження проводиться в рамках проекту NSERC Engage спільно з галузевим партнером Desire2Learn, лідером у розробці програмного забезпечення для електронного навчання. Desire2Learn надав реальні запитання та відгуки про те, як цей фреймворк може найкраще принести користь мобільним розробникам. Експертний досвід і знання персоналу Desire2Learn у питаннях розвитку були включені в розробку кожного етапу цього дослідження.

Важливо розуміти кожну з мобільних платформ і мов розробки, перш ніж ви зможете зрозуміти масштаб проблеми, яку вирішують CPDT. У цьому розділі надано довідкову інформацію про мобільний ландшафт і огляд різних CPDT. Цей

огляд необхідний для розуміння того, які аспекти є найважливішими та які слід враховувати в будь-якій системі оцінювання. Після цього відбудеться обговорення відповідної роботи для оцінювання CPDT

1.2 Поточний стан ландшафту кросплатформної розробки

Сучасний ландшафт включає вісім основних платформ смартфонів, перелічених у таблиці 1.1. Однак багато з них втрачають значення, оскільки iOS, Android, BlackBerry 10 і Windows Phone стають основними гравцями в осяжному майбутньому [7].

Таблиця 1.1

Середовище програмування для мобільних платформ

Операційне середовище	**Бажана мова програмування**
ОС RIM BlackBerry	Java ME, HTML, JavaScript
RIM BlackBerry 10	C/C++, HTML, JavaScript, Java
Apple iOS	Objective-C
Google Android	Java (з ароматом Harmony), C і C++
HP WebOS	HTML, JavaScript
Windows Phone	C#, .NET
Symbian	C, C++
Samsung Bada	C++, HTML, JavaScript

З огляду на кількість перерахованих мов, розробники повинні бути дуже універсальними, а підприємствам потрібно витрачати значні ресурси, щоб їх програмне забезпечення було доступне на кількох платформах. Comscore, лідер у вимірюванні частки цифрового ринку, оцінює, що в Сполучених Штатах Android має понад 52% частки ринку, а решта розподіляється між Apple, RIM, Microsoft і Symbian [23]. На рисунку 1.1 показано рівень зацікавленості розробників мобільними платформами. Ця цифра включає в себе згадки як для планшетів, так і для смартфонів на кожній платформі.

З огляду на таку велику зацікавленість у доступності для багатьох платформ, існує очевидна проблема, з якою стикаються розробники, вимагаючи можливості зробити свою програму доступною для якнайширшої аудиторії, але не вистачає ресурсів для створення нативно для кожної платформи. Щоб

розширити охоплення програм, не витрачаючи значний час, необхідний для освоєння кожної платформи; використання CPDT може бути відповіддю.

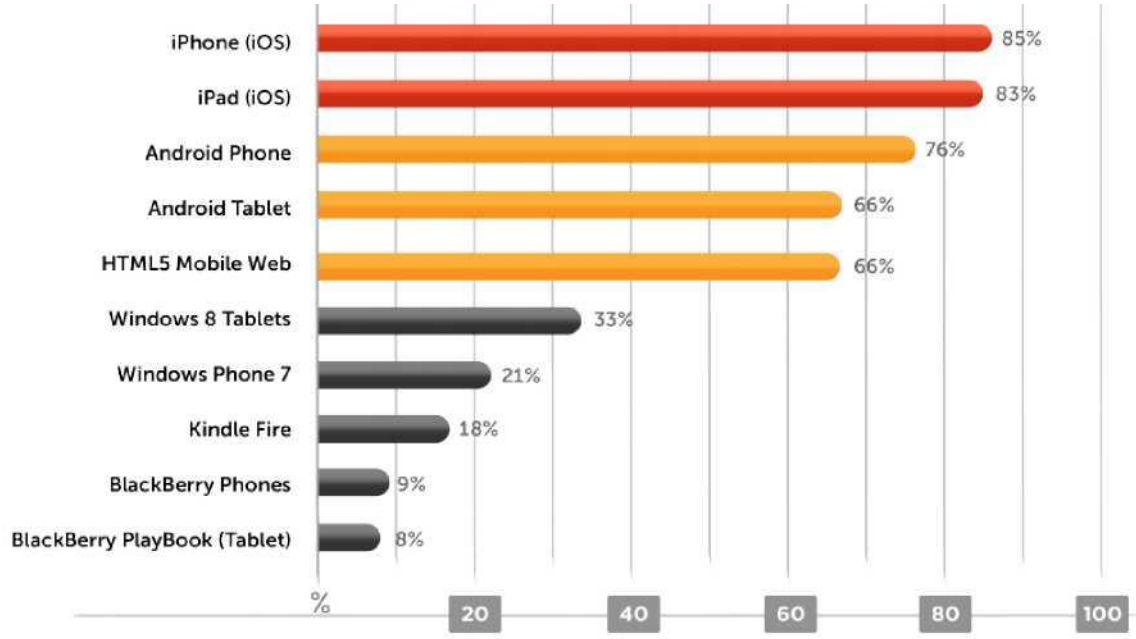

Рисунок 1.2 - Інтереси розробників до мобільних платформ [7]

Java вважалася чудовою для розробки ПК, оскільки дозволяла програмі бути написана один раз і перекладена на мову-посередник перед запуском на віртуальній машині Java (JVM) у системі користувача. Це дозволяло запускати одну програму на Windows, Mac і Linux без залежності від платформи. Java Micro Edition (Java ME) з'явилася на мобільних пристроях, але так і не стала масовою, часто вважають, що через її обмежені можливості, передбачувані проблеми з продуктивністю та фрагментацію пристрою [16] [24]. Невдовзі деякі виробники мобільних телефонів відмовилися від Java або розгалужені на спеціальні версії з додатковими функціями. З виходом на ринок Bedrock, Celsius, NeoMAD і alcheMo багатоплатформні інструменти, які розширили архітектуру Java Mobile, отримали певну популярність [54]. Ці інструменти дозволяли запускати додатки на значній кількості пристроїв, які підтримували Java, але інструменти були випереджені значними стрибками в технології для рідних платформ SDK. Ці інструменти, здавалося, втратили актуальність із приходом нової ери надзвичайно універсальних смартфонів. З 2011 року на ринок виходить новий

набір CPDT, до якого з часом швидко додаються нові функції. Це останнє покоління інструментів буде обговорено в наступному розділі.

Сьогодні на ринку є кілька інструментів для розробки кросплатформних мобільних додатків з різними рівнями функціональності та сумісності. Це нове покоління CPDT забезпечує більше контролю, функціональності та різноманітності, ніж попереднє покоління інструментів на основі Java [24] [37] [54]. Інструмент Cabana в [27] допомагає подолати прірву між старим і новим поколінням з точки зору функцій. Ми дослідили п’ять інструментів, як показано в таблиці 2.2. Ці інструменти були обрані через їхню гнучкість, підтримку функцій і популярність серед розробників. Вибрані інструменти надають зразки різних підходів до крос-платформної розробки зі стилями крос-компіляції, виконання та веб-обгортки. Від них також вимагалося підтримувати лише мінімум дві різні платформи, що дозволяло б проводити більш широкий спектр аналізу нових CPDT. Програми на основі мобільного веб-браузера включено з метою порівняння.

Таблиця 1.2

Інструменти для розробки міжплатформних додатків

CPDT	ОС BlackBerry	BlackBerry 10	iOS	Android	Windows Phone 7	Бада
Мобільний Інтернет					**V"**	
Adobe PhoneGap					**V"**	
Appcelerator Titanium		Бета				
Ромобільний Родос			3.0+	1,6+	**V"**	
Adobe Air						
MoSync					**c/**	

CPDT, описані в таблиці 1.2, значно відрізняються за можливостями, мовою та підтримкою платформи. Їх можна розділити на категорії залежно від методу компіляції та запуску програм, створених за допомогою інструментів. У той час як деякі CPDT, такі як MoSync, можуть крос-компілювати програму у рідний код, інші використовують веб-обгортку, по суті, запускаючи програму в об’єкті браузера, сумісному з HTML5. Традиційні середовища виконання та розширення, такі як ті, що використовуються в Adobe Flash, широко поширені на настільних комп’ютерах і ноутбуках, але мають обмежену підтримку на мобільних платформах і поступово припиняються. Альтернативою є

використання подібного підходу для подолання розриву, який є в Adobe Air [72]. Кожен із цих методів має явні переваги перед суто веб-додатками, додаючи можливі покращення продуктивності, інтерфейсу користувача та сповіщень, недоступні для веб-розробників. Ці додаткові можливості можуть швидко стати цікавою пропозицією для розробника, який бажає мати більш повнофункціональні програми. Деякі з цих інструментів уже використовуються такими компаніями, як NBC Universal, eBay і Сенат штату Нью-Йорк, що дозволяє їм надавати додаток якомога більшій кількості клієнтів за низькою ціною [8].

Таблиця 1.3

Функції CPDT

CPDT	**розвиток Мова**	**Компіляці Тип**	**Рідна інтерфейс**	**Доступ до**	**Файл система**	**Сповіщення користувача**	**додаток зберігати**
Мобільний Інтернет	HTML5 + JavaScript	Запускається в браузері		Обмеже-ний	Обмеже-ний		
Adobe PhoneGap	HTML5 + JavaScript	Рідний веб-додаток					
Appcelerator Титан	HTML5 + JavaScript / Python / Ruby /	Власний код і середовище виконання	v'	v'	v'	v'	v'
Ромобіль Родос	рубін	Час виконання					
Adobe Air	ActionScript/ HTML/AJAX	Час виконання					
MoSync	C / C++ /	Рідний код	v'	v'	v'	v'	v'

Інструменти, представлені в таблиці 1.2, використовують різні мови сценаріїв і надають різноманітні функції, сумісні не з усіма мобільними платформами. У таблиці 1.3 ми показуємо багато функцій, які надає кожен інструмент, включаючи обрану мову розробки.

1.2 Кросплатформні фреймворки для мобільної розробки

Обіцянка мобільної мережі як стандартної архітектури полягає в тому, що вона однаково відображатиме програми, незважаючи на те, що вони на різних платформах і пристроях. Розширені веб-програми можна писати за допомогою

HTML5 і JavaScript. Вони забезпечують високий рівень функціональності, як у веб-програмі Google GMail. Ця ідея не є новим підходом, і хоча вона вирішує багато проблем для розробників, вона додає кілька нових проблем. Мобільні веб-програми, які запускаються через веб-переглядач телефону, часто не відповідають рідному інтерфейсу користувача телефону, що може бути дуже заплутаним. Однак ці додатки можуть використовувати фреймворки JavaScript і CSS (каскадні таблиці стилів), такі як JQuery Mobile [65], Zepto [30] і Sencha Touch [60], щоб забезпечити настільки необхідні покращення інтерфейсу користувача з невеликою роботою з боку розробника. Існує багато таких фреймворків для покращення досвіду веб-розробки, і лише три з них оптимізовано для мобільних телефонів. Історично захист веб-переглядача не дозволяв зберігати локальні дані або доступ до датчиків пристрою, наприклад акселерометра.

Фреймворк webinos, запропонований у [42], надає альтернативу для забезпечення більшої функціональності, але не має широкого застосування. Крім того, самі браузери були фрагментовані різними реалізаціями JavaScript і механізмів візуалізації, що може призвести до неузгодженості функціональності та інтерфейсу на різних пристроях [59]. Найбільше занепокоєння часто викликає офлайн-доступ, коли немає доступу до програми за відсутності підключення. Чи ці проблеми перешкоджають використанню цього як середовища розробки? Відповідь досить суб’єктивна, але мінусів стає все менше й менше з розвитком технологій. HTML5 дозволив глибшу інтеграцію з телефоном, а WebKit майже повсюдно поширений на всіх мобільних пристроях, тому проблеми візуалізації минулого стають менш важливими [20]. На жаль, життя в браузері все ще блокує ці програми на ринках програм, які корисні для того, щоб допомогти кінцевим користувачам знайти роботу розробників. Хоча мобільні віджети додають деякі функції, вони все ще в основному обмежені браузером [40]. Багато розробників CPDT використовують концепцію та ідеали віджетів і мобільного Інтернету та вдосконалюють їх функціями, яких бракувало. У наступних розділах ми обговоримо інструменти, які використовують комплексний підхід до кросплатформної розробки.

Adobe PhoneGap PhoneGap [1] є реалізацією нещодавно перейменованого проекту з відкритим кодом Apache Cordova. Cordova та PhoneGap тісно пов'язані між собою і розглядатимуться як один інструмент. PhoneGap є одним із найпопулярніших підходів до кросплатформної розробки для мобільних пристроїв. Інструмент використовує переваги стандартизованих веб-технологій, які використовуються для мобільної веб-розробки, і переносить їх у власні веб-додатки. Подібно до стандарту віджетів W3C; PhoneGap використовує HTML, CSS і JavaScript для написання додатків, які потім інкапсулюються в об'єкт браузера, щоб виглядати як додатки, створені за допомогою власного коду. Завдяки такому підходу розробник може відправити програму в магазин програм, щоб увімкнути придбання та високу видимість. Окрім глибокої інтеграції з операційною системою та апаратним забезпеченням телефону, програми можуть використовувати локальне сховище та бути повністю доступними за відсутності підключення до мережі. Багато з цих переваг відсутні в стандартних мобільних веб-додатках. У таблицях 1.2 і 1.3 ми бачимо широку підтримку пристроїв і функцій із PhoneGap. За допомогою API розробники можуть використовувати різні функції JavaScript, використовуючи базові власні можливості пристрою. Найбільше бракує рідного Елементи інтерфейсу користувача, проте цей компроміс був зроблений для можливості мати його доступним на багатьох платформах.

Багато інструментів сторонніх розробників доступні для використання з PhoneGap разом із уже згаданими бібліотеками JavaScript; appMobi XDK для PhoneGap [12] надає IDE та середовище тестування, щоб значно допомогти розробникам. Він поставляється з інструментами для емуляції зовнішнього вигляду мобільної програми на різних пристроях. appMobi додатково надає аналітичну платформу та власний сервіс збірки, схожий на той, що безпосередньо надається PhoneGap [11]. Ця послуга дозволяє компілювати програми без необхідності встановлення SDK для кожної мобільної платформи. З додаванням цього XDK життєвий цикл розробки PhoneGap починає набагато більше нагадувати життєвий цикл нативних програм. Іншим інструментом, який

покращує PhoneGap, є IBM Worklight [36]. Він надає додаткові API та інтеграційні структури на стороні сервера для програм корпоративного рівня. Архітектура Worklight, показана на рисунку 1.2, активно використовує PhoneGap для передачі коду JavaScript на нативний пристрій. Він додає API Worklight для підвищення безпеки, додавання аналітики та доступу до проміжного програмного забезпечення, сервера Worklight. Цей сервер забезпечує підключення до серверних систем підприємства для мобільного додатку [37].

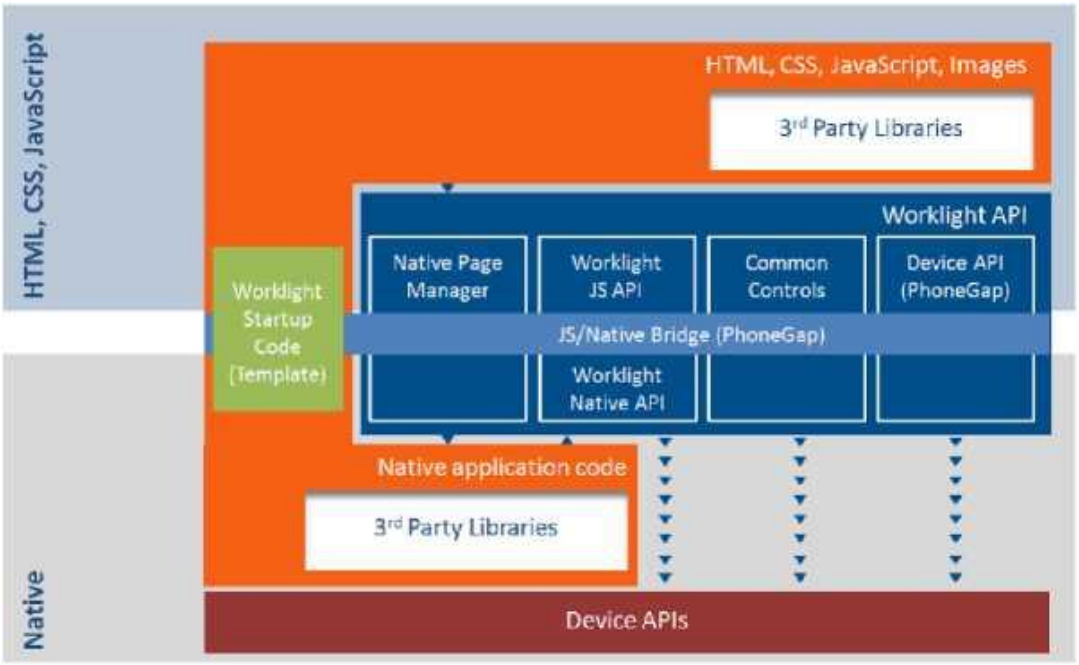

Рисунок 1.3 - Архітектура IBM Worklight [35]

Замість використання підходу на основі кадрів браузера Chrome, як у appMobi XDK, Worklight має IDE під назвою Worklight Studio, яка базується на Eclipse. Після розгортання програм уся аналітика та push-повідомлення обробляються через консоль Worklight [37]. Незалежно від того, використовуєте PhoneGap безпосередньо, appMobi XDK або набір інструментів Worklight, PhoneGap забезпечує неймовірно гнучкий підхід для розробки мобільних додатків. Для невеликих додатків , які потрібно швидко завершити до великих корпоративних додатків, існує варіант цього інструменту, який підходить для багатьох випадків використання.

Appcelerator Titanium [8] має багато спільних рис з PhoneGap і мобільним Інтернетом. Розробка для всіх трьох використовує стандартні мови веб-розробки, однак Titanium відрізняється в деяких важливих областях. Працюючи на ядрі з відкритим вихідним кодом, продукти Appcelerator дозволяють додаткам працювати на пристроях без вбудованого об'єкта браузера, як ми бачили у PhoneGap. Натомість їхній підхід використовує об'єкт виконання та метод компіляції для оптимізації та компіляції коду. Надаються параметри компіляції та запуску за допомогою середовища виконання; вбудований у веб-додаток або гібридний додаток, подібний до PhoneGap. Твердження полягає в тому, що рівні продуктивності підвищуються, щоб відповідати рівням власних програм при використанні підходу, заснованого на виконанні [10] [28]. Це твердження постачальника вимагає незалежного аналізу за допомогою стандартної архітектури порівняльного аналізу для визначення точності. Підхід інтерфейсу користувача також значно відрізняється від використання власних елементів інтерфейсу користувача, а не зосередження на суто веб-інтерфейсі. Багато людей вважають це великою перевагою, оскільки програма не викличе плутанини, оскільки матиме інший вигляд і відчуття, ніж ті, що створені за допомогою рідних інструментів. Інші, особливо ті, хто переходить на мобільні веб-додатки, можуть вважати це негативом і вважати, що це ускладнює настроюваність інтерфейсу [6].

Як і розширення IBM Worklight для ядра PhoneGap, Titanium надає безліч послуг для аналітики та розміщення на стороні сервера. Подібно до Worklight, доступні такі функції на сервері, як зберігання даних і push-повідомлення. Команда Appcelerator не надає додатку той самий рівень наскрізних гарантій безпеки, як у Worklight, і є більше орієнтованою на споживача, а не на корпоративну архітектуру [10] [27]. Titanium включає Titanium Studio, IDE на основі Eclipse і набір інструментів тестування та емуляторів. Appcelerator Titanium не має підтримки платформи: повністю підтримуються лише iOS і Android, а підтримка ОС BlackBerry була випущена в бета-версію рік тому [9]. Цю бета-версію було припинено та замінено майбутньою бета-версією ОС Blac

kBerry 10. Підтримка лише двох основних мобільних платформ забезпечує лише мінімум, необхідний для того, щоб називатися кросплатформною, і великі сегменти ринку залишаються поза увагою при розробці програм за допомогою цього інструменту.

Rhomobile. Замість використання веб-стандартів компанія Motorola Solutions на Родосі застосувала інший підхід [25]. Rhodes розроблено компанією Rhomobile, яка була придбана Motorola Solutions, підрозділом Motorola, що працює на підприємствах і урядах. Ця частина компанії не була придбана Google і не залежить від проекту Google Android [28]. Цей метод базується на мові програмування Ruby, що дозволяє поточним розробникам Ruby легко перейти. Не всі дорогоцінні камені Ruby переведені для роботи на Родосі; однак за потреби можна додати більше [20]. Інструмент надає потужні інструменти інтерфейсу користувача, включаючи доступ до рідних елементів інтерфейсу користувача для Android та iOS. На відміну від багатьох інших інструментів, Rhodes має повну підтримку Model View Controller (MVC), що позбавляє потреби мати бізнес-логіку у представленні, як JavaScript може мати в інших CPDT [23]. Це розділення може мати великий позитивний вплив у процесі розробки для різних розмірів екрана.

Роудс використовує підхід на основі середовища виконання, яке має середовище виконання, створене за допомогою C++ NDK для Android, який інтерпретує байт-код Ruby для платформи. Постачальник Rhodes вважає це більш ефективним, ніж використання Java на Android. Rhodes пропонує RhoStudio IDE із інструментами збірки, налагоджувачами та симуляторами. Розміщення додатків і підключення до корпоративних даних є сильним поштовхом для Родса з RhoConnect і сервером RhoSync. Це дозволяє RhoSync конкурувати за корпоративний ринок і Worklight. Нарешті, RhoHub об’єднує служби в інтерфейс із керуванням джерелами на основі Git і онлайн-службою збірки, подібною до PhoneGap Build. Незважаючи на те, що вихідний код відкритий за ліцензією Массачусетського технологічного інституту, велика частина функцій доступна лише через платну підписку на послуги від RhoMobile [28].

Adobe Air Adobe. Air призначено для створення багатофункціональних інтернет-програм, які можна запускати на багатьох платформах. Він використовує Adobe Flash, Adobe Flex, HTML, ActionScript і Ajax для розробки сценаріїв. Навички використання Flash легко знайти в галузі, і це сприймається як головна перевага Adobe Air. Існуючі Flash-програми часто можна перевести на запуск як нативні програми за допомогою Air, а не через розширення браузера. Програми Air, на відміну від Flash, вимагають встановлення середовища виконання, а також кожної програми. Підтримка мобільних пристроїв на Android та iOS доступна з незалежним середовищем виконання, доступним на Android, і необхідними елементами, які входять до програми в iOS. Починаючи з версії 2.6, більшість функцій є однаковими для версій платформи Android та iOS. Припускається, що подібні середовища виконання можуть бути ефективним способом боротьби з кросплатформною проблемою; виробники пристроїв сприйняли неоднозначно [18] [20].

Інструменти Air Developer є одними з найбільш надійних і підтримуваних у галузі. Оскільки інструмент походить від популярного інструменту Flash, використовуються ті самі інструменти, що й Flash Для розробки можна використовувати Builder і Dreamweaver. У поєднанні з технологіями Adobe Flex можна створювати корпоративні програми з розширеними можливостями інтерфейсу користувача, які можуть перевершити можливості інших платформ. Adobe не надає інтегрованого досвіду та наскрізних функцій, інструментів і послуг, що надаються деякими CPDT, про які йшлося раніше. Ці підходи на основі часу виконання, які є в Appcelerator Titanium, Rhomobile Rhodes і Adobe

Air, мають значні недоліки в деяких областях. Основний з них - це залежність від самого середовища виконання. Виробники мобільних ОС можуть у певний момент заблокувати їх у своїх магазинах, і включене середовище виконання може призвести до збільшення розміру файлу. Функції також можуть відставати від того, що випускається нативно, оскільки це інтегровано у середовище виконання. Оскільки під час виконання часто використовується своєчасна компіляція, на продуктивність може вплинути використання будь-

якого з цих середовищ виконання [23].

MoSync дуже схожий на Air, однак головна відмінність MoSync від інших CPDT полягає в тому, що він використовує крос-компіляцію, де код перекладається на рідний код Objective C і Java залежно від платформи. Компілятор C++, який використовується в MoSync, виводить проміжну мову, яка потім оптимізується та виводиться як двійковий файл для потрібної платформи. Це інший підхід, який може призвести до підвищення продуктивності програм.

MoSync також пропонує MoSync Wormhole PhoneGap, як CPDT, який використовує об'єкт браузера для відображення програми. Код C і C++ можна перекласти за допомогою їхньої технології червоточин для використання в цих веб-додатках. MoSync пропонує сумісність з OpenGL ES, що дозволяє використовувати 3D-графіку для розробки ігор, тому, на відміну від багатьох інших CPDT, це підходить для кросплатформних ігор [68]. Крім того, він досить розширюваний, що дозволяє додавати бібліотеки C і C++ до ваших програм. Перехресна компіляція може бути дуже корисною технікою, яка дає можливість мати справу зі скомпільованим власним кодом для підвищення оптимізації.

Вибір одного CPDT замість іншого може бути складним завданням, наявні навички та вимоги проекту часто диктують, який використовувати під час порівняння можливостей CPDT. Вибір значно звужується, коли в CPDT потрібні спеціальні функції, такі як рідний інтерфейс користувача та можливість 3D-графіки. CPDT, засновані на середовищі виконання, повинні включати середовище виконання з додатком і в багатьох випадках створювати більші розміри файлів [24]. Перехресна компіляція може бути ефективним способом виконання завдання; однак ці обговорювані підходи все ще мають обмежені API. Веб-платформи CPDT, такі як Appcelerator Titanium і PhoneGap, пропонують багато функцій, але в деяких випадках мають обмежену підтримку платформи. Відсутність власних елементів інтерфейсу користувача може бути проблематичною, але також забезпечує свободу для індивідуально розроблених макетів. Усі вивчені інструменти сумісні з використанням таких функцій пристрою, як акселерометр, камера та сповіщення. Вони забезпечують набагато

більш інтегрований досвід на відміну від мобільних веб-додатків, де інтеграція все ще невелика, зокрема зі сповіщеннями для користувачів. Ці CPDT мають різноманітні витрати та ліцензії, які швидко коригуються. Деякі, такі як PhoneGap, є безкоштовними та мають відкритий код, а інші, як Appcelerator, мають безкоштовні та платні рівні. Типовою є багаторівнева модель із параметрами підтримки на вищих рівнях, яка також використовується PhoneGap. З розвитком інструментів розвивалися і моделі ціноутворення, які є рухомою ціллю. З кожним із кросплатформних методів адекватні функції налагодження та документація все ще були відсутні або застарілі [29]. Наразі багато ресурсів обмежено документацією, наданою самими виробниками інструментів, тому повне об’єктивне порівняння важко встановити в ці ранні дні та вимагає особистого досвіду роботи з кожним CPDT.

1.4 Висновки по розділу

результаті дослідження було розроблено обґрунтовану систему оцінювання CPDT, що дозволяє об’єктивно порівнювати інструменти мобільної розробки. Це дає змогу розробникам приймати більш зважені рішення при виборі фреймворків відповідно до потреб проєкту.

Сучасний ландшафт кросплатформної розробки характеризується зростанням популярності CPDT, які дозволяють створювати додатки для кількох мобільних платформ із мінімальними витратами. Нове покоління інструментів забезпечує ширшу функціональність і кращу сумісність порівняно з попередніми підходами.

Кросплатформні фреймворки для мобільної розробки, попри різні підходи — від веб-орієнтованих до виконання рідного коду, — дозволяють суттєво спростити створення додатків для кількох платформ, але вибір інструменту залежить від конкретних потреб проєкту, доступних ресурсів **і** вимог до продуктивності та функціональності.

РОЗДІЛ 2 . СТРУКТУРА ОЦІНЮВАННЯ ДЛЯ CPDT

2.1 Каркас

Поява багатьох CPDT на ринку залишила важливе питання без відповіді; чи впливає використання CPDT для створення та розгортання вашої програми на якість програми? Крім того, чи зменшуються витрати на розробку з використанням CPDT замість того, щоб створювати програму нативно кілька разів? Здатність CPDT мати всі функції та продуктивність, а також зменшення витрат у грошовому вираженні та часу на розробку порівняно з рідною розробкою наразі залишаються відкритими питаннями. У цьому розділі представлено структуру для оцінки CPDT, щоб надати рекомендації розробникам і визначити, чи потрібно робити будь-які компроміси при використанні CPDT.

Оцінка CPDT повинна проводитися різними методами. Оскільки наразі мало доступних досліджень, які б порівнювали можливості CPDT між собою та з рідними платформами, ця структура використовує стандартні практики, адаптовані з інших структур у подібних областях, щоб вирішити цю проблему. Мета цієї структури полягає в тому, щоб забезпечити наукову та справедливу основу для визначення поточного стану ефективності CPDT. Сфера дії фреймворку обмежена додатками, які не потребують інтенсивної роботи з графікою, тому його не слід застосовувати для розробки ігор. Розробка ігор - це особливий випадок, який потребує різних інструментів, не схожих на ті, що використовуються у стандартній розробці додатків, і, отже, вимагатимуть інших методів оцінки. В іграх використовуються різні механізми та 3D-графіка, де частота кадрів є важливим фактором, а не обговорюваними в цій структурі елементами. Подібним чином, це дослідження зосереджено на смартфоні, але в майбутньому його можна поширити на планшети.

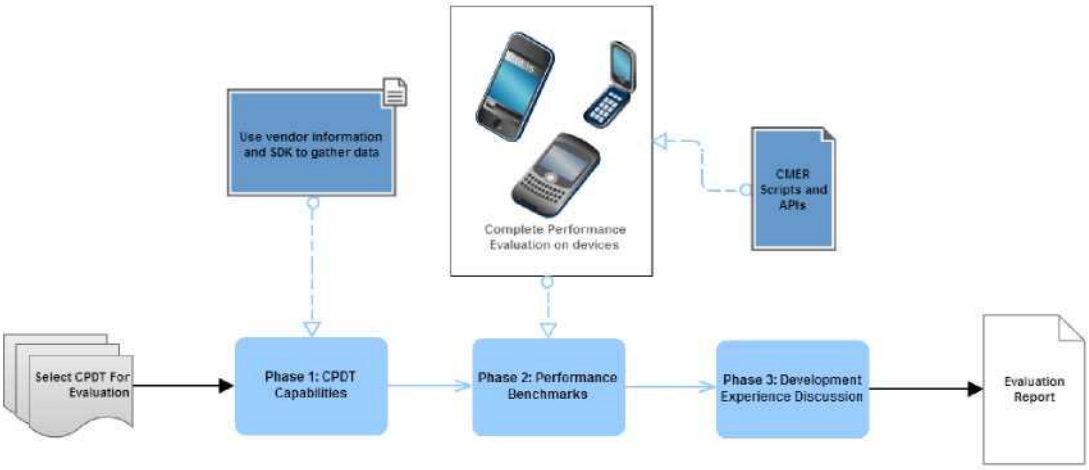

Рисунок 2.1- Структура оцінки CPDT

Структура оцінювання складається з трьох етапів, як показано на рисунку 2.1. Перший етап зосереджений на визначенні можливостей фреймворку за допомогою великого набору функцій, які пропонують деякі інструменти. Друга фаза забезпечує набір тестів, які необхідно реалізувати за допомогою CPDT і перевірити на час завершення. Після цього йде заключний етап, на якому обговорюються більш суб'єктивні питання досвіду розробки, щоб забезпечити контекст і додаткову інформацію для розуміння результату. Завершення кожного з цих етапів дозволить отримати повне уявлення про можливості досліджуваного інструменту, а результати мають бути узагальнені у звіті про оцінку.

2.2 Можливості CPDT

Фаза I забезпечує оцінку можливостей і функцій CPDT під час тестування на основі великого набору функцій, які потенційно можуть дозволити платформи розробки. Оцінка базується на контрольному списку та вимагатиме від оцінювача ознайомитися з документацією та SDK CPDT, щоб перевірити, чи підтримуються функції, і створити діаграму сумісності. Стовпці в цій діаграмі згруповані в різні основні категорії розвитку, такі як доступ до датчиків і безпека. У майбутні ітерації нативних платформ і CPDT можуть бути включені нові функції, яких наразі немає в списку. Структура розроблена таким чином, що основні категорії залишатимуться узгодженими, а піделементи, можливо,

потребуватиме оновлення.

Деякі функції можна вважати підтримуваними, непідтримуваними або частково підтримуваними. У рамках оцінювання слід включити огляд функцій, які це складають, щоб узагальнити результат.

Усі оцінки CPDT викличуть деякі загальні основні запитання, наприклад, які мобільні платформи підтримуються та під якою ліцензією CPDT. У цьому першому розділі слід окреслити ці основи, які необхідно знати, щоб зрозуміти CPDT, і включати функції, наявні в поточних нативних SDK, які просто неможливо класифікувати інакше. Цей розділ також включає початкові та поточні витрати на використання CPDT, які в деяких випадках можуть бути значними.

Дізнатися про середовище розробки важливо, коли вибираєте, який CPDT найкраще відповідає вашим потребам. Ця категорія включатиме надійність IDE та тип, мови розробки, доступні для використання з CPDT та середовищем налагодження. Це полегшує оцінку часу, необхідного для вивчення нових інструментів. Також важливо знайти тип компіляції, який використовує CPDT, існує багато різновидів, які можуть давати різну продуктивність. Із середовища розробки розробникам потрібні інструменти для створення додатків, які мають таку ж якість, що й власні засоби розробки. Вони повинні дозволяти швидке налагодження та можуть надавати прості інтегровані методи компіляції програм. Основні характеристики цього середовища мають бути перераховані та доступні, а будь-яка інтеграція зі спеціалізованими API задокументована.

Щоб створити привабливу програму, потрібен привабливий інтерфейс користувача створений. У цьому розділі ми обговоримо можливість мати інтерфейс користувача, який 27 відповідає вказівкам щодо інтерфейсу, наданим кожним постачальником платформи. Оскільки ці вказівки часто змінюються, структура буде зосереджена на підтвердженні доступної гнучкості CPDT для відповідності таким вказівкам.

Пристрої мають багато внутрішніх місць зберігання інформації ОС, які містять файли пристрою та дані програм або інформацію користувача. Щоб деякі

програми працювали якнайкраще, доступ до сховищ інформації має бути доступним у CPDT. Крім того, підключення до функцій пристроїв нижчого рівня та характеристик апаратного забезпечення може підвищити продуктивність у певних мережевих і графічно інтенсивних програмах.

Мобільні пристрої мають безліч датчиків, які можуть збирати дані для передачі в програми для необмеженої кількості цілей. Швидко додаються нові датчики, які швидко адаптуються до рідних SDK, однак CPDT можуть не мати доступу до кожного з них. У цьому розділі має бути описано, які датчики доступні.

Деякі спеціальні датчики можна використовувати в поєднанні з мобільною ОС, щоб увімкнути геолокацію пристрою. Незважаючи на те, що GPS може здаватися єдиною функцією, необхідною для адекватного позиціонування, існує багато інших можливостей для пошуку місцезнаходження. Деякі інструменти можуть не дозволяти використання кожного з цих методів або не мати можливості вибрати, який метод використовувати, це має бути описано тут.

Важлива інформація часто повинна надсилатися на пристрій із зовнішніх служб, щоб повідомити програму або користувача про певну інформацію або виконати дію. Наявність методу надійного виконання цих дій необхідна для багатьох програм. Постачальники платформ можуть запропонувати власний метод надсилання сповіщень пристрою, і слід вказати інтеграцію з ними або можливість використання стороннього рішення.

Щоб уможливити монетизацію програми, CPDT має підтримувати різноманітні функції, щоб надати розробникам свободу використання обраної ними схеми ціноутворення. Багато безкоштовних програм можуть захотіти використовувати рекламну платформу, яку можуть надати постачальники платформи або третя сторона, де платні розробники програм можуть бути найбільше зацікавлені в тому, щоб їхні користувачі могли легко здійснювати покупки в програмі. Інші моделі монетизації можуть стати поширеними та можуть бути включені далі. Слід включити можливість отримати програму в руки користувачів із високою видимістю через різні торгові точки, такі як

магазини програм.

Більшість постачальників послуг турбуються про безпеку, щоб переконатися, що дані користувача не обробляються неправильно, а вихідний код програми не стає доступним для тих, хто не повинен мати його. Методи безпечного зберігання інформації та її отримання через мережеві з'єднання повинні бути включені в CPDT. Крім того, CPDT має забезпечити, щоб під час розгортання користувачі не мали можливості читати вихідний код програми.

2.3 Еталонні показники продуктивності

Показники продуктивності є важливою частиною тестування здатності цих CPDT виконувати завдання так само, як і вбудовані програми. Для цього етапу порівняльні тести будуть розроблені як набір і розгорнуті для кожного CPDT оцінювачем і порівняні з аналогічно розробленими рідними тестами. Важливим фактором, який досліджується, є відмінність результатів на тому самому пристрої від однієї реалізації до іншої. Коли новий CPDT стане доступним, структура вимагатиме розробки тестового набору за допомогою нового інструменту, дотримуючись вимог, викладених у структурі. Ефективність базової технології та оптимізації компіляції кожного CPDT перевірятимуть шляхом порівняльного аналізу програми, розробленої з використанням цього до ol. Результати цього порівняльного тесту покажуть, чи здатний CPDT створювати високопродуктивні програми. Наскільки це можливо, тести виконуватимуться за добре оціненими алгоритмами, які раніше були включені в інші пакети тестів для мобільних пристроїв або ПК, згадані в 1.4. Деякі тести можуть бути нереалізовані на кожному CPDT або на кожній платформі через обмеження в мобільній операційній системі або в самому інструменті.

Спочатку ми оцінимо процесорно інтенсивні програми та розглянемо SunSpider JavaScript Benchmark версії 0.9.1 для шифрування AES і алгоритмів перевірки введення [21]. Це два часто використовувані елементи на мобільних пристроях, які постійно доповнюватимуть мобільні програми. Використовуючи

код JavaScript SunSpider як орієнтир для алгоритму, як AES, так і тести перевірки введення дадуть результат за мілісекунди. Тест хмари тегів також можна використовувати з SunSpider для обробки великої кількості тексту та визначення частоти вживання слів. Це може бути корисним під час додавання функції індексованого пошуку до програми. Ще один тестовий пакет, який можна використовувати для порівняння, це програма Linpack для Android на основі Linpack Java [25]. Це Java-програма з відкритим кодом, яка надає багато тестів, вивчених протягом останніх 40 років і використаних для порівняння в [17]. Тест Linpack вимірює, наскільки швидко система може розв'язувати лінійні рівняння з елімінацією Гауса. Це забезпечить оцінку в мільйонах операцій із плаваючою комою за секунду (MFLOPS). Було показано, що цей тест покращився в пізніших версіях Java на тому ж апаратному забезпеченні завдяки новій ефективності віртуальної машини та надасть корисне розуміння того, як працюють інші інструменти [25]. Метою використання будь-якого з цих добре вивчених тестів є стрес-тест ЦП, щоб побачити, чи код із кросплатформних інструментів такий же ефективний, як рідний код нижчого рівня.

Мобільні програми повинні використовувати багато джерел даних і часто поєднувати їх у доступну для користувачів форму. Щоб імітувати тестування для цього, буде здійснюватися доступ до локальних і віддалених баз даних за допомогою API CMER, видобувати дані та використовувати алгоритми сортування для сортування великих масивів даних. Потенційні тести можна знайти в [17] і [26] з використанням алгоритму сортування купи. Деякі тести також мають використовувати внутрішні функції сортування, які надає CPDT API. API CMER надаються для отримання набору даних із нашого сервера CMER за допомогою локального з'єднання. Це важливо для будь-якого віддаленого тестування, щоб усунути нерівності на основі зв'язку через стільникову мережу. Реалізація цих API може відрізнятися залежно від тестів, які реалізуються в конкретному оцінюванні. Це може бути дуже обмеженим або широким залежно від реалізації фреймворку.

Мобільні програми мають можливість отримувати інформацію з кількох

датчиків і зберігати інформацію або використовувати її в програмі. Тести будуть проводитися опитуванням мікрофона та записом аудіо роликів довільної тривалості. Час, витрачений на ініціалізацію запису, буде задокументовано. Подібний тест можна провести для ініціалізації та часу знімка до знімка для камери. Оскільки тести з використанням власних SDK і CPDT проводяться на одному пристрої, характеристики пристрою не вплинуть на результат.

Одним із найважливіших аспектів взаємодії з програмою є швидкість реагування. Ці тести швидко додадуть і видалять компоненти інтерфейсу користувача та візуалізують нові екрани та елементи програми. Це буде зроблено для оцінки здатності CPDT відтворювати компоненти інтерфейсу та швидко переходити від одного екрана до іншого. Якщо CPDT страждає від візуалізації інтерфейсу користувача, це забезпечить поганий досвід для користувачів програм, створених з його використанням.

Коли ви будете готові перейти до фази тестування, слід виконати набір дій. Перед тестуванням на будь-якому пристрої слід виконати певні дії, щоб переконатися, що тести є максимально точними. Спочатку оновіть системне програмне забезпечення пристрою до останньої версії, включаючи будь-які доступні виправлення помилок і патчі. Потім слід виконати заводське очищення, щоб видалити будь-які користувацькі програми та налаштування, які можуть забрати цикли процесора від тесту. Після встановлення тестової програми виконайте повне скидання пристрою, щоб очистити всі програми в пам’яті. Крім того, закрийте всі запущені фонові завдання, які можна закрити. Запустіть контрольний тест із повними 3 зовнішніми ітераціями з усередненими результатами, які відображаються на екрані та надсилаються на сервер журналювання. Візуальний опис процедури тестування можна знайти на рисунку 2.2. Під час тестування з кількома CPDT кожну програму слід запускати окремо, а результати скомпілювати для фази II звіту про оцінку. Після завершення тестування можна скласти звіт про оцінку, використовуючи досвід розробки тестів, щоб відповісти на запитання.

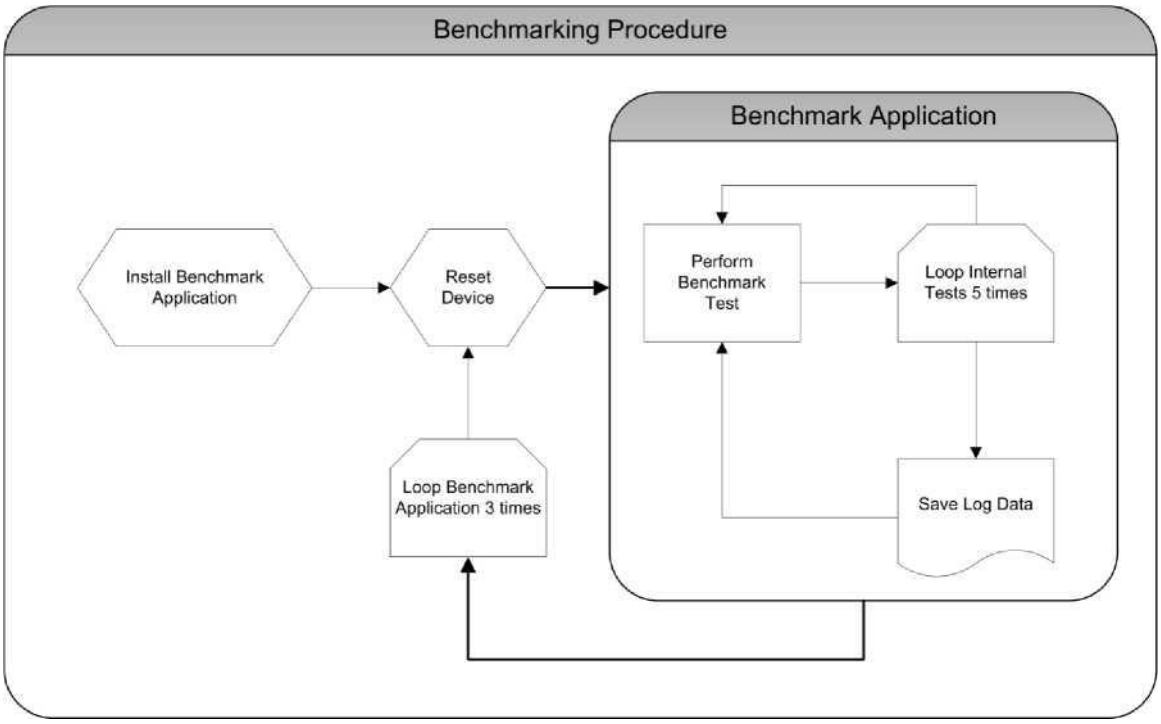

Рисунок 2.2 - Процедура порівняльного аналізу

2.4 Обговорення досвіду розробки

На етапі оцінювач дослідить критерії, які ви не можете виміряти будь-яким простим способом. Через характер цих CPDT не все можна включити в міру чи контрольний список, натомість їм потрібно використовувати досвід, отриманий під час проходження етапів I та II, щоб детальніше обговорити частини інструменту. У цьому розділі буде обговорено деякі функції, щоб читач міг зробити значущі висновки про рівень наданої функціональності. Обговорення проводитиметься таким чином, щоб обмежити можливу упередженість оцінювача, і його слід використовувати для надання контексту для розуміння результатів.

Деякі характеристики CPDT може бути важко описати у формі діаграми, і вони потребують тривалого пояснення. Тут слід обговорити характеристики інструментів, які обговорюють, наскільки добре підтримується CPDT для оновлень, нових функцій і виправлень помилок. Крім того, можна включити подальше обговорення витрат, пов’язаних із використанням інструменту, і будь-яких поточних витрат. Це включатиме пов’язані витрати та функції, доступні для

будь-яких пропонованих хмарних або аналітичних служб. Обговорення має також обертатися навколо IDE розробки та підтримуваних функцій. Дайте відповідь, чи включені функції та методи налагодження забезпечують очікуваний рівень функціональності.

Метою цього розділу є використання досвіду розробки, отриманого під час виконання інших етапів цієї оціночної системи, для обговорення таких питань, як відносний розмір програми та потужність набору інструментів інтерфейсу користувача. Гнучкість побудови інтерфейсу може бути важливою, і її слід зазначити. Оцінювач повинен спробувати обговорити, чи дозволяє інструмент одноразово писати, запускати будь-де розробку або рівень налаштування, необхідний для різних платформ. Додатково обговоріть загальну надійність досвіду розробки та наведіть конкретні приклади того, де використання інструменту є корисним, і будь-які недоліки. Нарешті, розкажіть про криву навчання для використання цього CPDT у цьому розділі звіту.

Вибір інструментів розробки для створення мобільних додатків ніколи не був таким великим. Хоча власне створення для кожної платформи може забезпечити найшвидший доступ до нових функцій, CPDT мають своє місце на ринку. Через впровадження кожного етапу цієї структури, CPDT, доступні сьогодні, або нові інструменти, випущені в майбутньому, можна оцінити на основі критеріїв, розглянутих у цьому розділі. Це забезпечить збалансований погляд на функції, продуктивність і досвід розробки для будь-якого CPDT. Озброївшись цією інформацією, розробники отримують детальне та неупереджене уявлення про можливості інструментів і зможуть найкращим чином вирішити, який із них відповідає їхнім критеріям для їхніх програм. Ця структура вимагає взяти головні теми та застосувати їх більш детально, щоб створити адекватні тести не лише для сучасних CPDT, але й для майбутніх інструментів. Можливості, показники продуктивності та обговорення досвіду в цьому розділі забезпечують предмет подальшого тестування. Щоб перевірити структуру на придатність, її застосували до різних CPDT. Це не лише продемонструє силу та корисність фреймворку, але й дозволить оцінити поточні

інструменти верхнього рівня, щоб надати негайні вказівки щодо придатності кожного з них для різних програм. У наступному розділі ми запровадимо цю структуру для створення конкретних тестів, які мають відношення до CPDT на ринку сьогодні. Експерименти, які були проведені, будуть описані та обговорені.

2.5 Висновки по розділу

Отже, запропонована структура оцінювання CPDT дозволяє всебічно оцінити інструменти кросплатформної розробки за функціональністю, продуктивністю та зручністю використання. Це допомагає розробникам приймати обґрунтовані рішення щодо доцільності використання CPDT замість нативної розробки.

Фаза I структури оцінювання CPDT дозволяє систематично дослідити функціональні можливості інструментів, охоплюючи широкий спектр технічних та практичних аспектів розробки. Такий підхід допомагає визначити сильні та слабкі сторони кожного CPDT ще до початку безпосередньої реалізації проекту.

Еталонні показники продуктивності дозволяють об'єктивно оцінити ефективність CPDT у порівнянні з нативними рішеннями, демонструючи здатність інструментів забезпечити прийнятний рівень продуктивності. Поєднання стандартних тестів, аналізу часу реакції, роботи з датчиками та базами даних забезпечує всебічну перевірку практичної придатності кожного CPDT.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТИ

Використовуючи структуру, розроблену в Розділі 2, ми обговоримо, як цю структуру було реалізовано, щоб оцінити сьогоднішні CPDT. Оскільки структура розроблена для окреслення лише основних принципів, першим кроком є детальне розбиття кожної фази, щоб створити чесний набір тестів, які мають відношення до інструментів розробки, обраних для оцінюванняПісля цього надається детальний план і специфікації для кожного етапу оцінювання. Реалізація, представлена в цьому розділі, повинна задовольняти критеріям, викладеним, в той час як вона впроваджується та зосереджується на сучасному середовищі CPDT. Обсяг оцінювання не обов'язково охоплює всі аспекти, але повинен забезпечити повну основу порівняння для CPDT, включених до дослідження.

3.1 Параметри експерименту

Щоб забезпечити структуру експерименту, у наступних підрозділах буде описано інструменти, включені в дослідження, пристрої, що використовуються для тестування, і будь-які припущення, враховані в результатах.

Це дослідження було проведено з використанням багатьох інструментів розробки. Для порівняння будуть включені як кросплатформні, так і рідні комплекти розробки. Включені інструменти та підтримувані платформи можна знайти в таблиці 3.1. Записи, позначені тире, несумісні.

Усі тести кожної платформи проводяться на одному пристрої з використанням як рідних, так і міжплатформних інструментів. Пристрої, які будуть використовуватися, наведені в таблиці 3.2.

Хоча Adobe Air сумісний із Blackberry 10, його виключено з нашого тестування через численні проблеми з бета-версією програмного забезпечення.

Таблиця 3.1

Матриця сумісності для інструментів, включених до дослідження

	Android	**iOS**	**BlackBerry 10**	**BlackBerry 7**
WebWorks	-	-	сумісний	сумісний
BB10 Native SDK	-	-	сумісний	-
Android Java SDK	сумісний	-	-	-
iOS Native	-	сумісний	-	-
PhoneGap	сумісний	сумісний	-	сумісний
Appcelerator Титан	сумісний	сумісний	-	-
Adobe Air	сумісний	сумісний	сумісний	-
MoSync	сумісний	сумісний	-	-

Це можна включити в подальше тестування, коли платформа стабілізується.Програмне забезпечення BlackBerry 7 Beta від Appcelerator Titanium було недоступне для тестування, оскільки його розробку припинено. Тести продуктивності на етапі II будуть розроблені для кожного сумісного інструменту та комбінації платформ. Цей вибір інструментів дозволяє проводити різноманітні порівняння з кількома інструментами, що працюють на кожній платформі. Деякі CPDT пропонують різноманітні методи розробки.

Таблиця 3.2

Пристрої, використані для тестування

Платформа	**Модель пристрою**	**Версія програмного забезпечення**
Android	Samsung Galaxy S II	4.0.3
	(i9100)	
iOS	Apple iPhone 4s	5.1.1
BlackBerry 7	BlackBerry Bold 9900	7.0.0.579
Blackberry 10	BlackBerry 10 Dev Alpha	10.0.4.197

Розробка Titanium зосереджена на додатках, створених і запущених за допомогою їхньої технології виконання, тоді як програми MoSync зосереджені лише на крос-компіляції. Програми Adobe Air використовують кросплатформний підхід до розробки ActionScript.

Щоб перевірити CPDT, необхідно взяти до уваги певні припущення. Було визнано доцільним перевірити на одному пристрої для кожного CPDT. Порівняння, виконане в цій структурі, стосується не продуктивності пристрою, а різниці в продуктивності, яка є рідною для CPDT. На цей фактор мало впливатиме пристрій, на якому виконуються тести, якщо всі тестування виконуються на одному пристрої. Існує невелика ймовірність виникнення певної помилки пристрою, яка може вплинути на результати одного CPDT, а не інших, однак ми припустили, що це дуже малоймовірно, і завершення всіх тестів на одному пристрої дасть необхідні результати. Тестування було проведено лише на пристроях, після виконання процедур. Цих процедур достатньо, щоб закрити необхідні фонові програми та очистити достатню кількість пам'яті для належного завершення тесту з мінімальним заважанням інших процесів. Очікується, що тести можуть відрізнятися залежно від версії CPDT, що використовується. Версія, яка використовується для тестування, має бути показана під час Фази I оцінки та залишатися незмінною протягом кожної фази. Крім того, оскільки різні розробники можуть реалізувати алгоритми дещо різними способами, вони повинні бути написані таким чином, щоб відповідати нормам і найкращим практикам, викладеним для конкретного CPDT. Кожна реалізація тесту має однаковий результат і кроки незалежно від використовуваного інструменту чи мови.

3.2 Еталонні показники продуктивності

Оцінка продуктивності є ключовим аспектом оцінки CPDT. На даний момент невідомо, наскільки ефективні CPDT на основі Інтернету порівняно з нативними, оскільки доступні лише неофіційні докази. Крім того, подібні

інструменти, що використовують однакові методи компіляції, можуть мати покращення продуктивності від постачальника, що часто рекламується як причина для вибору одного інструменту замість іншого. Не все можна включити в тести продуктивності, але тести розроблені для імітації типових завдань і дій. Кожен тест розроблений таким чином, щоб забезпечити можливість реалізації широкого спектру інструментів тест із всеохоплюючою каркасною програмою для виконання адміністративних завдань із запуску тесту. Алгоритми тестування та специфікації будуть викладені в наступних підрозділах. Реалізація кожного з цих тестів відбуватиметься за допомогою стандартної мови розробки для кожного з CPDT. У всіх випадках уникайте плагінів або сторонніх доповнень до інструментів і надано певний код.

Ця програма має надавати основні функції, необхідні для виконання тестів оцінки ефективності. Скелет відноситься до простого графічного інтерфейсу користувача, який дозволяє вибрати набір тестів і кнопку для початку тестування. Після завершення всіх тестів відображається підсумковий екран із середніми результатами для ітерацій тесту. Усі тести мають бути завершені один раз, а потім цикл починається знову, доки не буде досягнуто задану кількість ітерацій. Детальні результати містяться у файлі журналу. Ці тести мають бути модульними і їх можна додавати та видаляти за бажанням. Цей скелет і наступні тести можна реалізувати на кожному CPDT із кількістю ітерацій, визначеною структурою. Щоб пришвидшити розробку, потрібно надати функцію налагодження, якщо для кожного вибраного тесту виконується лише одна ітерація.

Інтерфейс - це простий список тестів із кнопкою запуску, як показано на рисунку 3.1. Деякі зміни можуть знадобитися залежно від можливостей інструменту, наприклад переміщення кнопок ліворуч у версії Titanium. Під час завершення тесту відображається індикатор прогресу. Цей індикатор прогресу має відображати загальний прогрес тестування та оновлюватися лише між тестами, щоб переконатися, що він не впливає на результати. Оскільки деякі тести включають використання компонентів інтерфейсу користувача,

користувач бачить їх перед тим, як повернутися до екрана прогресу під час цих тестів.

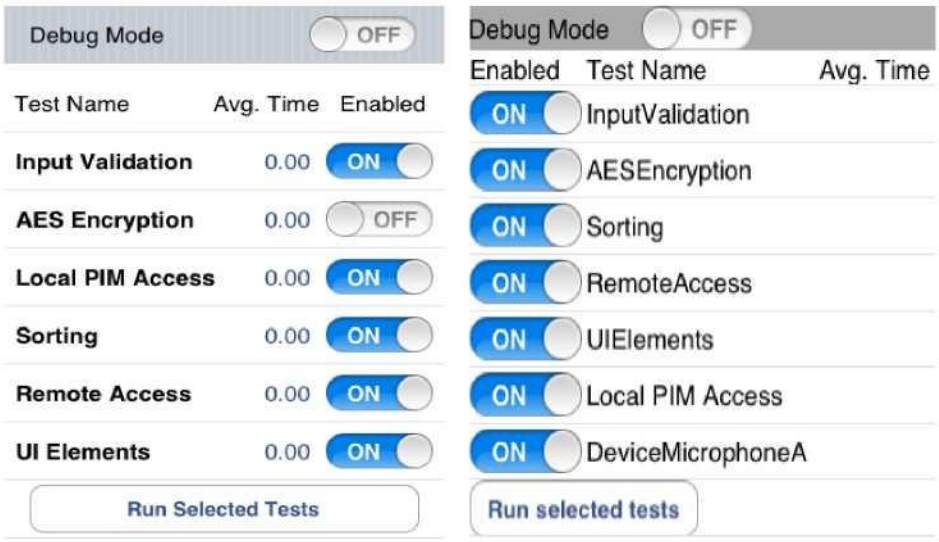

Рисунок 3.1 - Нативна реалізація в iOS (ліворуч), реалізація Titanium в iOS (праворуч)

Кожен модуль має бути самодостатнім із функцією запуску під назвою «test», яка викликається скелетною програмою. Наприкінці тестового запуску набір значень передається до функції під назвою addResult у скелеті, який обробляє дані. Він приймає три параметри: рядок для назви тесту, long для початкового часу та long для кінцевого часу. Ця функція збереже інформацію для надсилання на сервер після завершення всіх тестів. Після завершення всіх ітерацій усіх вибраних тестів викликається інша функція для впорядкування збережених даних, обчислення середніх значень, додавання впорядкованих і повних даних до рядка та надсилання цього рядка на сервер журналу. Рядок надсилається через HTTP POST із параметром під назвою «дані», який дорівнює згенерованому рядку.

Два файли зберігаються під час кожного оцінювання. Ці файли мають містити префікс дати та часу, щоб не видалити попередні результати. Ці файли мають бути збережені як файл із значеннями, розділеними комами (CSV). Перший файл журналу містить необроблені результати кожного тесту. Якщо тест виконується з 5 ітерацій, 'testname' матиме 5 ідентичних записів з різними

значеннями тривалості. 41 Ім’я файлу має бути CPDTTYPE-DEVICE-DATE-TIME.csv із заміною CPDTTYPE, DEVICE, DATE і TIME під час створення серверним сценарієм. Якщо розробник еталонного тесту захоче зберегти файли CSV локально, це не вплине на результати тестування та може бути зроблено. Через обмеження пристроїв це не завжди можливо, і може знадобитися вхід на сервер. Структура даних [testname, starttime, endtime, duration]. Ці назви стовпців мають відображатися в першому рядку, а потім записи для кожного тесту. Усі тривалості вказані в мілісекундах з одним знаком після коми.

Другий файл має назву CPDTTYPE-DEVICE-DATETIME.csv. Він має містити лише заголовок, який показує структуру та результат середньої тривалості для кожного з тестів. Структура [testname,average]. Таким чином, незалежно від кількості ітерацій тесту в скелеті, кінцевий результат буде середнім значенням цих тестів, збережених у цьому файлі.

Дані надсилаються на сервер у такому форматі:

DEVICE,CPDTTYPE,OSVERSION,DATE,TIME;

testname,AVERAGE; testname,AVERAGE; ....;

назва тесту, час початку, час закінчення, значення;

назва тесту, час початку, час закінчення, значення; .;

Сервер проаналізує цю інформацію та створить файли журналу таким чином. У папці «Необроблені журнали» назва файлу «CPDTTYPE-DEVICE-DATE-TIME.csv» DEVICE,CPDTTYPE,OSVERSION,DATE,TIME (заголовок інформації про тест) назва тесту, час початку, час закінчення, значення (заголовок категорії)

назва тесту, час початку, час закінчення, значення (дані)

назва тесту, час початку, час закінчення, значення (дані)

У папці «Averages/CPDTYPE» до імені файлу «CPDTTYPE-DEVICE-OSVERSION.csv» додайте наступне:

DEVICE,CPDTTYPE,OSVERSION (тестовий інформаційний заголовок)

testname,datetime,AVERAGE (заголовок категорії)

testname,datetime,AVERAGE (дані)

testname,datetime,AVERAGE (дані)

Реалізацію скелета програми можна знайти в Додатку B для реалізації WebWorks. Подібна реалізація використовується на всіх інших платформах. Для WebWorks уся логіка міститься у файлах JavaScript, і для кожного плагіна уникають використання.

Тести виконуються послідовно, а потім повторюються, доки не буде досягнуто загальну кількість ітерацій. Для оцінки CPDT кожен тест було виконано 5 разів із усередненими результатами, які відображалися користувачеві та зберігалися у файлі журналу середніх значень. Деякі тести можуть мати розділи, які внутрішньо повторюються, таким чином створюючи велику кількість загальних ітерацій, коли програму порівняльного тесту запускають іззовні 3 рази для остаточних результатів. Усі тести повертають одне значення, тому внутрішні ітерації або усереднюються, або береться значення загальної тривалості перед поверненням до скелета.

Щоб завершити тестування, буде проведено тестування процесора, інтенсивності даних, доступу до пристрою та взаємодії з користувачем. У цьому розділі будуть детально описані алгоритми тестування.

Мета цього тесту - зашифрувати та розшифрувати уривок тексту. Тривалість між стартом і фінішом буде надано тестовий скелетон. Алгоритм бути 43 використовується алгоритм Rijndael, який є тією самою стандартною технікою AES, що використовується в SunSpider JavaScript benchmark версії 0.9. Щоб забезпечити швидку реалізацію цього алгоритму, код у тесті SunSpider Crypto-AES [21] можна використовувати як керівництво для перенесення на інші платформи. Алгоритм також був реалізований у Java, C та C++ з великою документацією, представленою на веб-сайті Національного інституту науки та технологій [28]. Текст, ключі шифрування та алгоритм можна знайти в Додатку D. Цей тест повертає результат загального часу, необхідного для одного циклу шифрування та дешифрування.

Метою цього тесту є спроба змоделювати перевірку правильності введених користувачем елементів, як-от адреси електронної пошти та поштові індекси США. Алгоритм, задокументований у тесті string-validate-input, який використовується як рекомендація з SunSpider JavaScript benchmark версії 0.9 [11]. Щоб завершити цей тест, буде згенеровано та перевірено 4000 електронних адрес, а потім 4000 поштових індексів. Тест шукатиме довжину, недопустимі символи та правильну структуру. Заповнюючи це багато разів, ви зможете зрозуміти, чи є у CPDT труднощі з порівнянням рядків. Цей тест повертає загальну тривалість виконання 4000 перевірок електронної пошти та 4000 перевірок ZIP. Процес цього тесту наведено на рисунку 3.2.

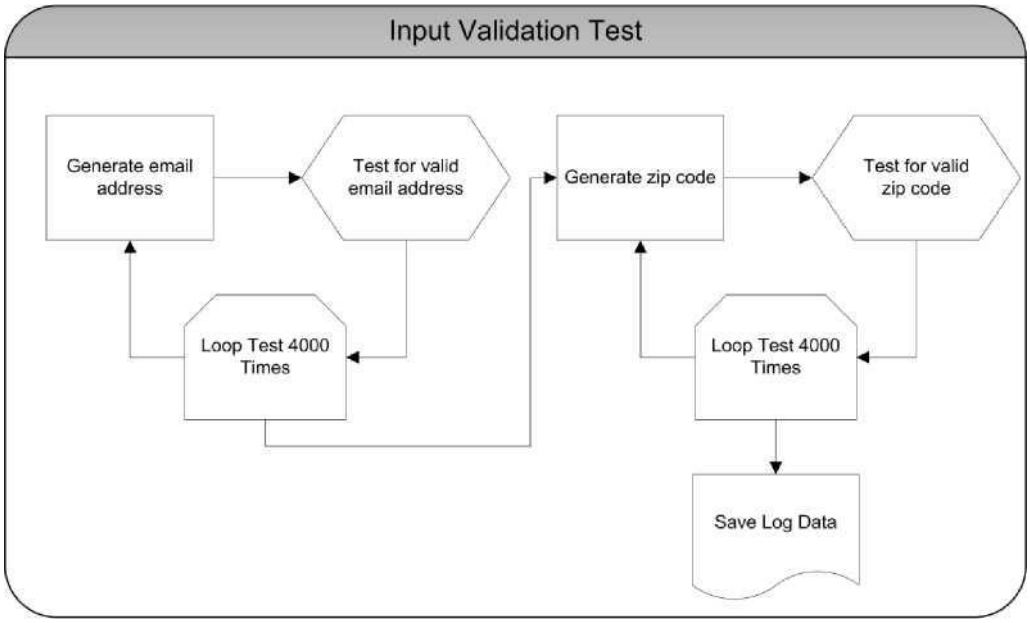

Рисунок 3.2 - Процедура перевірки вхідних даних

Тести інтенсивного використання даних зосереджені на маніпулюванні великими обсягами даних у структурованих форматах і доступі до баз даних пристрою. У наступних розділах будуть описані ці тести.

Цей контрольний тест спрямований на читання локально збережених даних користувача та запис до адресної книги PIM (Personal Information Manager). Він складатиметься з операцій з базою контактів пристрою. Щоб перевірити ефективність запису даних в адресну книгу та її отримання, слід виконати низку кроків. Буде використано файл CSV, що містить 100 записів

імені, прізвища, номера телефону та електронної пошти, який буде згенеровано випадковим чином за допомогою інструментів, знайдених у [29] і включених у Додаток Е. Процедура:

1. Прочитайте локальний файл CSV із збереженою контактною
2. інформацією в пам’яті,
2. Введіть усі 100 контактів до порожньої бази даних PIM,
3. Отримайте 100 контактів із бази даних PIM, включаючи ім’я, прізвище, номер телефону та електронну адресу,
4. Видалити всі контакти з бази даних PIM. Цей тест повертає загальну тривалість виконання наведеного вище алгоритму.

Метою цього тесту є запит до віддаленої веб-служби на сервері та отримання інформації, яку можна використовувати в програмі. Ця інформація буде надіслана на пристрій і буде виміряно час відповіді. Усе тестування виконується через локальну мережу, щоб зменшити будь-яку затримку мережі. Цей тест дозволяє створити CMER API у формі сценарію, який буде розміщено на сервері з Ubuntu Server 11.04 на машині, підключеній до мережі 100 Мбіт/с із підключенням пристроїв Wi-Fi 802.11g. Машина складається з процесора Intel Core 2 Duo E6550 2,33 ГГц з 2 ГБ оперативної пам’яті. Для повернення тексту через HTTP використовуватиметься рядок PHP. Текст є жорстко закодованим рядком, а PHP-код міститься в Додатку F. Код спрощено до echo-відповіді, але в інших реалізаціях оцінки може знадобитися розширений API. Порівняльний тест складатиметься зі встановлення підключення до цього сценарію, отримання рядка та повторення процесу 100 разів. Загальна тривалість для всіх 100 внутрішніх ітерацій є поверненим значенням. Цей процес можна побачити на рисунку 3.3.

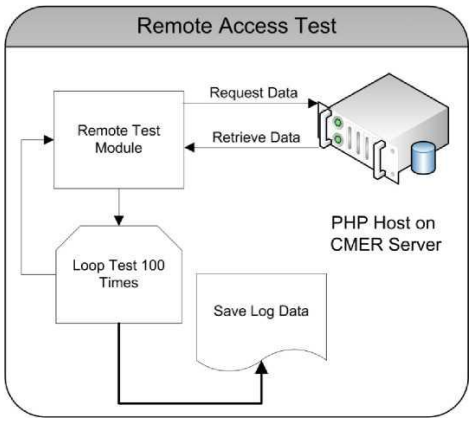

Рисунок 3.3 - Процедура перевірки віддаленого доступу

Цей тест складається з генерації 2500 випадкових цілих чисел від 0 до 2500, дозволяючи дублікати та сортуючи їх. Просте бульбашкове сортування відсортує ці числа від найменшого до найбільшого. Це виконується за допомогою циклів для порівняння сусідніх вузлів. Незважаючи на те, що це не найефективніший метод сортування, велика кількість порівняльних операцій робить його хорошим кандидатом для визначення ефективності різних CPDT. Потім цей процес буде повторено 5 разів внутрішньо. Тривалість тесту повинна включати генерацію чисел і сортування. Цей тест розроблено відповідно до тестів сортування в [17] і [16]. Загальна тривалість для всіх 5 внутрішніх ітерацій є поверненим значенням. Витяги з алгоритму сортування можна знайти в Додатку G. Вони показують той самий алгоритм, реалізований на двох CPDT. Процедура проведення тесту описана на рисунку 3.4.

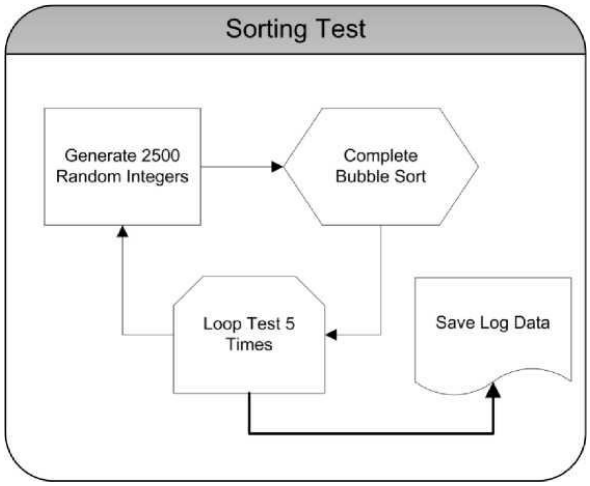

Рисунок 3.4 - Процедура перевірки сортування

Тести доступу до пристрою зосереджені на використанні функцій пристрою через API CPDT. У наступному тесті для тестування API використовуватиметься датчик мікрофона.

Цей тест буде зосереджено на зборі інформації за допомогою мікрофона та будь-якій затримці між часом від запиту до результату з використанням різних CPDT. Протокол ініціалізує мікрофон, запише 0,5 секунди звуку, який не потрібно зберігати, і запише час, витрачений на цей процес. Це буде повторено та повернено середнє значення. Процедура така:

1. Рекордний час,
2. Ініціалізація мікрофона,
3. Записати 0,5 секунди аудіо,
4. Записати час закінчення для результату,
5. Повторіть кроки 1-4 загалом 25 разів,
6. Знайдіть середнє значення 25 внутрішніх ітерацій і поверніть його як кінцевий результат.

Зареєстрований результат тесту є середнім із 25 тривалостей ініціалізації. Час початку та закінчення не є обов'язковим і можна залишити 0.

Тести взаємодії з користувачем спрямовані на забезпечення маніпуляцій передніми функціями та інтерфейсом користувача, щоб забезпечити високу продуктивність програм. Два тести в цьому розділі надають тести маніпуляції та переходу. Цей тест спробує швидко керувати модифікацією елементів екрана користувача під час послідовного тестування та записати час завершення. Елементи, які потрібно додати, включають зображення, текстові поля, мітки, кнопки та перемикачі. Простий інтерфейс буде створено, а потім змінено в певних кроках.

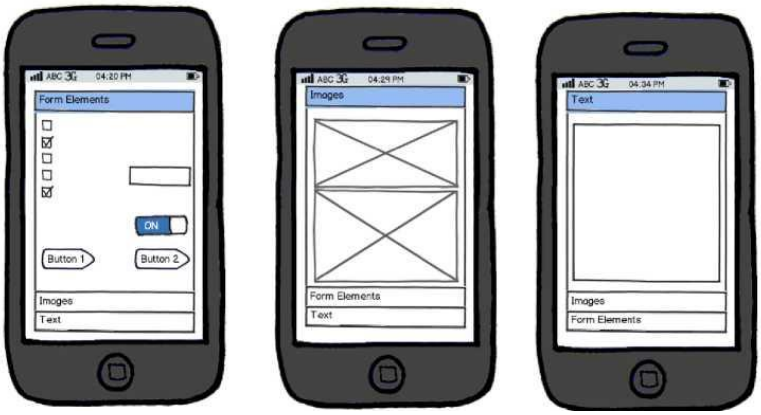

Рисунок 3.5 - Екрани для перевірки елементів інтерфейсу користувача

Буде використано макет інтерфейсу в стилі акордеона з 3 розділами.

Перший складатиметься з елементів форми, включаючи 5 прапорців, текстове поле, перемикач і 2 кнопки. Другий матиме два зображення: одне з оригінальною роздільною здатністю зображення, а інше було зменшено до розміру 20% x 20% від розміру екрана. Ці зображення включено до цього документа в Додаток G. Третій екран складатиметься з тексту, знайденого в Додатку D, і це той самий уривок із «Ромео і Джульєтти», який використовується для тестування шифрування AES. Після рендерингу початкового екрана в стилі акордеона ми гортаємо розділи та додаємо та видаляємо елементи наступним чином:

1. Рекордний час,
2. Почніть із розділу елементів форми,
3. Перейти до розділу зображень,
4. Перейти до текстового розділу,
5. Видалити весь текст і додати його назад, цикл 20 разів,
6. Перейдіть до розділу Елементи форми,
7. Довільно встановіть і зніміть прапорці 500 разів,
8. Видаліть кнопку 2 і знову додайте її, довільно вибравши верхню ліву позицію пікселя, повторіть 50 разів,
9. Перейдіть до розділу зображень. Видаліть обидва зображення та додайте їх назад, перемикаючи нормальний розмір і масштабований до 20% екрана, повторіть 20 разів.

10. Запис часу виконання.

Виконання цих кроків вважатиметься одним виконанням тесту. Тривалість буде від моменту початкового візуалізації екрана до остаточної зміни зображення. Реалізацію цього алгоритму для WebWorks можна знайти в Додатку C.

У цьому тесті буде прогортано серію екранів програми та виміряно час, потрібний для завершення візуалізації та переходу до наступного екрана. Екрани виглядатимуть так, як показано на рисунку 3.6. Переходи будуть такими: Екран A складається з двох кнопок і 9 піктограм, які можна побачити в меню більшості програм. Піктограми вкладено в Додаток G, як і зображення для екрана B. Екран B містить одне зображення, розтягнуте на весь екран. Останній екран, Екран C, має назву та текст із тесту шифрування AES, які можна знайти в Додатку D. Якщо можливо, CPDT має кешувати екран, щоб він не спричиняв додаткового навантаження. Якщо для цього немає параметра, використовується значення за замовчуванням. Об'єкт контролера керуватиме переходами екрану відповідно до діаграми на рисунку 3.7. Цей предмет також буде вимірювати час.

A→B→A→C→A→C→B→A→B→A→B→C→C→A→B→B→A→A→A→A→B
→A→B→B→B→A→B→C→C→A→C→B→C→A→A→B→C→A→B→C→A

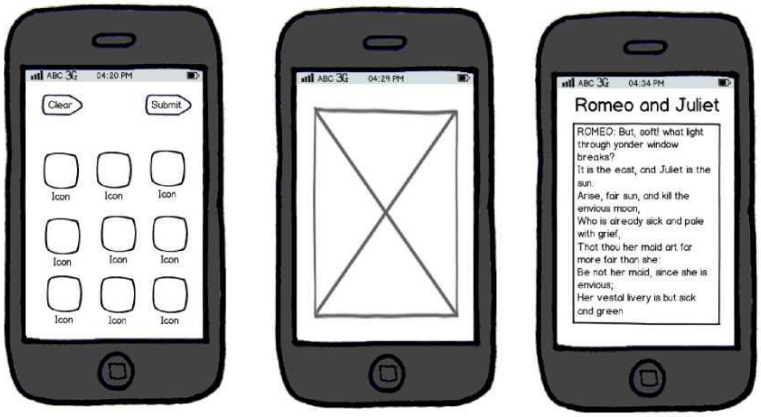

Рисунок 3.6 - Екран A, екран B та екран C для перехідного тесту

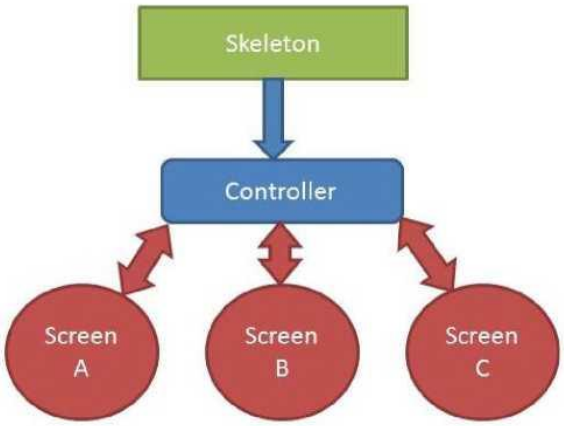

Рисунок 3.7 - Структура керування для тесту на перехід

Для повного завершення тесту необхідно завершити всі 40 переходів і записати тривалість.

Через низку факторів для цілей цієї оцінки не кожен тест був реалізований з використанням кожного CPDT. Тести, включені до цієї оцінки, наведені в таблиці 3.1. В оцінку включено достатню кількість тестів, щоб показати міцність структури, щоб забезпечити основу для порівняння CPDT і зробити висновки щодо структури та міцності певних інструментів. Більші дослідження можуть бути завершені та обговорюються як частина майбутньої роботи в розділі.

Таблиця 3.1.

Тести для оцінки CPDT

	Android	**iOS**	**BlackBerry 10**	**BlackBerry 7**
WebWorks	-	-	Введення перевірка, Сортування	AES, перевірка введених даних, дистанційне керування, сортування,
BB10 Native SDK	-	-	Введення перевірка, Сортування	-
Android Java SDK	Перевірка введення, PIM, дистанційне керування, сортування, мікрофон, перехід	-	-	-
		Перевірка введених		

Продовження таблиці 3.1

iOS Native	-	PIM, віддалений,	-	-
		Сортування,		
	AES, перевірка	AES, вхід		AES, вхід
PhoneGap	PIM, Remote, Sorting, UI Elements,	Перевірка, PIM, Remote, Сортування, UI	-	Перевірка, дистанційне керування,
	Перехід	Елементи, Перехід		елементи,
		AES, вхід		
Appcelerator	AES, перевірка	Перевірка, PIM,		
Титан	Сортування, перехід	Сортування,		
		Перехід		
Adobe Air	Перевірка введення, дистанційне керування, сортування, мікрофон, елементи інтерфейсу	Перевірка введених даних, дистанційне керування, сортування,	-	-
MoSync	Сортування, елементи інтерфейсу користувача, перехід	Сортування, елементи інтерфейсу користувача, перехід	-	-

3.3 Обговорення досвіду розробки

Кожен CPDT має свої унікальні характеристики, які може бути важко описати коротко. Раніше вважалося, що використання CPDT замість нативної розробки уповільнює впровадження нових функцій у програми. Це може стати проблемою для розробників при виборі інструменту. Оцінювач повинен обговорити в цьому розділі час затримки між основними вдосконаленнями SDK для будь-якої з мобільних платформ і коли ці нові функції були включені в API CPDT. Крім того, моделі монетизації поточних інструментів сильно відрізняються. Деякі інструменти є безкоштовними для використання, але платять за підтримку, деякі не пропонують жодних форм підтримки, а інші стягують плату за розробку, підтримку та постійне розгортання. Етап I вимагав збору інформації про вартість інструментів і поточну вартість використання CPDT. У цьому розділі оцінювач має розповісти про одноразові витрати або витрати на підписку та поточні варіанти підтримки. Питання, на які слід відповісти: чи є безкоштовні варіанти, чи надається служба push або аналітика та

за яку вартість, вартість послуги створення та інші витрати, такі як облікові записи розробників у постачальників платформ. Нарешті, використовуючи зібрану інформацію про IDE та мови, які цей інструмент використовує для розробки, оцінювач обговорить гнучкість і популярність цих мов та інструментів розробки. Слід зосередитися на тому, чи є вони повнофункціональними та гнучкими, а також на те, чи є навички розробки з їх використанням серед поточних розробників. Нові інструменти можуть бути надзвичайно корисними, проте можуть виникнути компроміси, враховуючи кількість часу, який потрібен для їх першого вивчення, що буде обговорюватися під час розробки.

Спочатку обговоріть родича розмір програми за допомогою CPDT; чи цей інструмент містить великий час виконання з 54 додаток? Побудова інтерфейсу користувача може сильно відрізнятися між платформами. Інструменти інтерфейсу користувача, надані для використання CPDT, повинні забезпечувати рівень гнучкості, коли всі пристрої з дисплеями з дуже низькою та дуже високою роздільною здатністю повинні відображати програму у придатній для використання формі. Наявність інтерфейсу, схожого на власний користувальницький інтерфейс платформи, і відповідність інструкціям щодо дизайну не тільки допоможе користувачеві отримати більш інтуїтивно зрозумілий досвід, але й може бути обов’язковою вимогою для надсилання до магазинів програм. Інструменти інтерфейсу користувача мають забезпечувати певну форму простого налаштування мови для легкого розгортання іноземними мовами та мати здатність виглядати переконливо та добре розроблено. Деякі пристрої не мають сенсорного інтерфейсу, також обговоріть, чи допускає CPDT курсор або сенсорну панель, а не сенсорний екран.

Спочатку обговоріть розмір програми за допомогою CPDT; чи цей інструмент містить великий час виконання з 54 додаток? Побудова інтерфейсу користувача може сильно відрізнятися між платформами. Інструменти інтерфейсу користувача, надані для використання CPDT, повинні забезпечувати рівень гнучкості, коли всі пристрої з дисплеями з дуже низькою та дуже високою роздільною здатністю повинні відображати програму у придатній для

використання формі. Наявність інтерфейсу, схожого на власний користувальницький інтерфейс платформи, і відповідність інструкціям щодо дизайну не тільки допоможе користувачеві отримати більш інтуїтивно зрозумілий досвід, але й може бути обов'язковою вимогою для надсилання до магазинів програм. Інструменти інтерфейсу користувача мають забезпечувати певну форму простого налаштування мови для легкого розгортання іноземними мовами та мати здатність виглядати переконливо та добре розроблено. Деякі пристрої не мають сенсорного інтерфейсу, також обговоріть, чи допускає CPDT курсор або сенсорну панель, а не сенсорний екран.

3.4 Оцінка ефективності CPDT

Впроваджуючи тести з використанням різних CPDT, кожна фаза оцінювання надала значну інформацію щодо сильних і слабких сторін інструментів. Ці результати представлені в цій главі. Результати покажуть інструмент і платформу, використані для тесту, і результат.

Функції не розповідають повну історію, коли справа доходить до оцінки цих інструментів, тому, хоча ми тепер знаємо, на що здатний кожен, важливо перевірити, наскільки добре працюють деякі з цих функцій. Кожен із тестів продуктивності було реалізовано на деяких або всіх платформах. Не всі тести були завершені на кожній платформі через безпеку, функції чи інші обмеження. Це перешкода, з якою стикаються кросплатформні інструменти, яка може бути проблематичною, якщо не вибрати інструмент, який точно відповідає меті. Кожен із наступних розділів надасть результати та коментарі до кожного з тестів. Ці тести складаються з тесту повторюваного 5 разів і усередненого для одного запуску. Кожне тестування потім повторюється 3 рази для остаточного середнього значення, яке вказується в результатах. Це означає, що як мінімум тести виконуються 15 разів для кожної платформи, щоб знайти базовий середній результат. Усі результати тестування відображаються в матрицях результатів для інструментів і платформ. Оскільки не кожен інструмент підтримує всі

платформи, деякі місця будуть навмисно залишені порожніми. Важливо зазначити, що порівняння слід проводити лише для різних CPDT, які використовують ту саму платформу, тому порівняння PhoneGap на Android і Titanium на iOS не є справедливим порівнянням. Це пов'язано з різним апаратним забезпеченням, що робить порівняння недійсним.

Щоб перевірити це за допомогою точної версії того самого алгоритму, цей тест було завершено лише з використанням WebWorks, Titanium і PhoneGap для перевірки їхніх індивідуальних методів оптимізації. Тестування проводилося на BlackBerry 7, iOS і Android. Матриця результатів у таблиці 3.2 покаже середню кількість мілісекунд, витрачених на виконання тесту.

Таблиця 3.2

Матриця результатів тестування шифрування AES (мілісекунди)

	Android	iOS	BlackBerry 7
WebWorks	-	-	108.5
PhoneGap	82.9	275,9	145.1
Appcelerator Титан	39.3	248.3	-

З результатів ми бачимо, що Appcelerator значно швидший на Android і дещо на iOS. У всіх тестах PhoneGap відстає від інших навіть з ідентичним кодом. WebWorks, будучи від самого постачальника платформи, може мати покращення продуктивності, але це припущення. Appcelerator Titanium стверджує, що має механізм компіляції, який підвищує продуктивність JavaScript, і з цих результатів це може бути правдою. Можна помітити, що передбачувані методи оптимізації можуть витримати критику в цьому випадку, оскільки Titanium лідирує з значним відривом.

Цей тест було розроблено з використанням кожного CPDT, включеного в дослідження, за винятком MoSync через відсутність компараторів регулярних виразів. Тест було виконано на BlackBerry 7, iOS, BlackBerry 10 і Android з використанням міжплатформних і власних інструментів розробки . Таблиця 5.3

показує результати контрольного тесту перевірки вхідних даних із до 4 порівнянь для кожної платформи.

Результати чітко показують, що існує кореляція з продуктивністю та використанням певних інструментів. Результати Blackberry 10 здаються досить близькими, але BlackBerry 7, iOS і Android мають надзвичайно великий діапазон. Adobe Air у цьому тесті показала себе досить погано. Зазвичай очікується, що нативні тести працюють найкраще, однак можна побачити, що Android Java SDK, який використовується для більшості розробок на платформі, справді гірший, ніж усі CPDT.

Таблиця 3.3

Матриця результатів тесту валідації вхідних даних (мілісекунди)

	Android	iOS	BlackBerry 10	BlackBerry 7
WebWorks	-	-	104.2	172,5
BB10 Native SDK	-	-	141.8	-
Android Java SDK	1463,5	-	-	-
iOS Native SDK	-	311.9	-	-
PhoneGap	128.2	257,9	-	339,8
Appcelerator Титан	55.3	213.6	-	-
Adobe Air	549.3	1001	-	-

Разом із результатами в таблиці 3.3 ми можемо переглянути окремі тести, які складають середні значення в цій діаграмі. На рисунку 3.8 показано графіки 4 інструментів на платформі Android для кожного з 15 тестових запусків. Кожен дає досить послідовні результати з певною мінливістю.

Середні значення та стандартне відхилення наведені в таблиці 3.4, що підтверджує відносно низьку варіативність між тестами. Низька мінливість підтверджує узгодженість тесту та те, що ми використовуємо достатню кількість усереднених ітерацій.

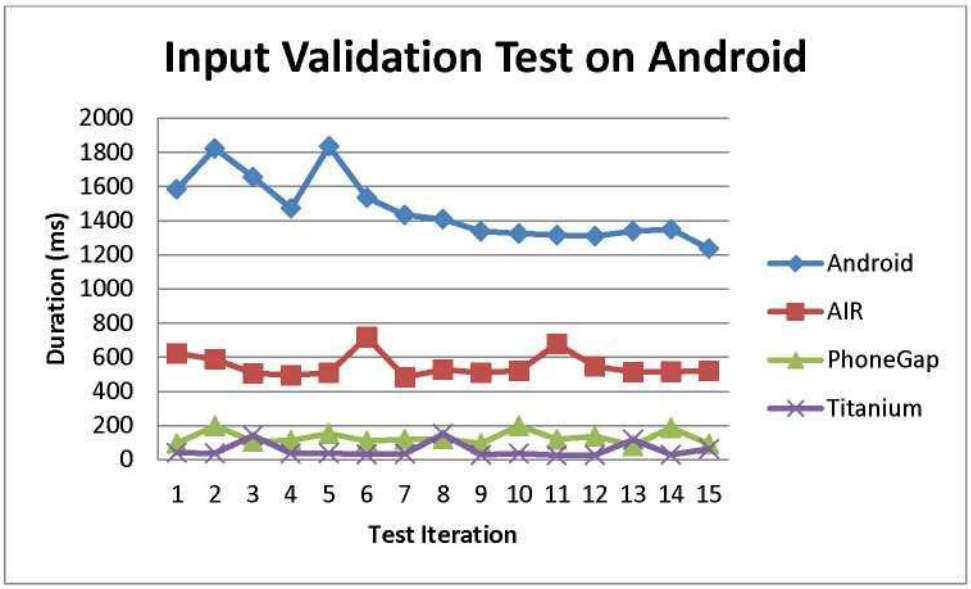

Рисунок 3.8 - Результати перевірки вхідних даних на платформі Android

Таблиця 3.4

Статистика змінності тесту для перевірки введення на Android

	Середній	Медіана	Стандартне відхилення
Android Java SDK	1463,5	1408	186,7
PhoneGap	128.2	119	38.7
Appcelerator Титан	55.3	36	43.2
Adobe Air	549.3	519	70.2

Важливо знати значущість значень і перевірити, чи є різниця в середніх значеннях статистично значущою. Для цього буде використано Т-тест, у якому порівнюються необроблені дані для 15 прогонів тесту, щоб побачити, чи можливо, що різниця в середньому значенні є просто випадковою. Щоб перевірити це, ми повинні вибрати нашу нульову гіпотезу. Для цього ми вибрали: Ноі U TOOLI = U T 0012 , тобто два середні рівні. Наша альтернативна гіпотеза буде викладена як Н А: U TOOLIO Ф U T 0012 . Високий рівень впевненості у відповіді потрібен, щоб довести, що різниця в середньому є значною, тому використовується рівень 99%, отже, з аа 0,01.

3.5 Висновки та рекомендації щодо вибору CPDT для кросплатформної розробки

Під час процесу оцінювання було накопичено великий досвід розробки з використанням цих CPDT. У кожному інструменті було виявлено багато помилок, невідповідностей і багатообіцяючих функцій, які роблять його корисним для використання розробниками. Оскільки ми оцінюємо 4 такі інструменти, обговорення буде розбито на відповідні підрозділи та обговорюватиме теми, згадані в рамках.

PhoneGap є відносно корисним інструментом розробки, але без IDE з налагодженням; стало важко тестувати програми. PhoneGap надає багато основних функцій, які більш-менш працювали належним чином для кожної з платформ. Емулятор Ripple працює, щоб перевірити деякі функції, але оскільки це просто хромована рамка; інтерфейс програми виглядає зовсім не так, як на пристрої. Емулятор також сумісний лише з версією 1.0, яка наразі стоїть за останньою версією 2.0. PhoneGap Build була головною відмінністю цього інструменту. Однак його використання означає, що існують певні припущення щодо додатків, а також певні обмеження.

PhoneGap Build наразі не підтримує плагіни, а це означає, що будь-які відсутні функції у PhoneGap не можуть бути реалізовані розробниками. Підтримка плагінів включена в дорожню карту PhoneGap Build, тому розробники можуть побачити її доступною в майбутньому. Природа PhoneGap з відкритим вихідним кодом дозволяє іншим використовувати його основні функції та створювати відсутні компоненти, щоб зробити його більш надійним інструментом. Це починає відбуватися з IBM Worklight і є перспективним для цього CPDT. PhoneGap швидко адаптується до останніх змін у мобільних платформах завдяки підтримці спільноти та швидкому циклу розробки. Було видно, що всі типи введення були доступні, а побудова інтерфейсу була обмежена веб-технологіями, які не були найкращим вибором для мобільних

додатків. Користування інструментами є безкоштовним із оплатою за підтримку, що дає перевагу в цьому відношені.

Appcelerator Titanium має багато спільного з PhoneGap, але з самого початку ми помітили, що ви не можете мати один набір коду для Android та iOS. Сегменти коду та елементи інтерфейсу користувача повинні розроблятися незалежно, лише центральна частина коду є кросплатформною. Це цікавий підхід, і, здається, він добре працює навіть із додатковою роботою. Здається, що компроміс із використанням компонентів рідного інтерфейсу вартий додаткових зусиль, необхідних для узгодженого інтерфейсу користувача з усіма функціями рідних інструментів. Titanium SDK виявив чимало помилок і проблем у Windows і Mac. Здається, наразі він не сумісний з останнім випуском Xcode, але під час роботи він працював добре. Деякі тести були неможливими, як-от версія тесту PIM для Android.

API не надають прямого доступу до контактів без взаємодії з користувачем, обмежуючи те, що можна з ними робити. Крім того, ми виявили, що вони не дозволяли стандартні HTTP-запити JavaScript і натомість мали власну реалізацію, з якою наш сервер мав деякі проблеми. Titanium є хорошим інструментом розробки з багатьма інструментами для налагодження розробників та іншими доступними інструментами, але багато помилок і обмежень у цьому ранньому випуску продукту показують, що йому бракує зрілості. Розробники Titanium заявили, що оновлять платформу протягом 30 днів після випуску нових функцій, однак це неможливо підтвердити. Крім того, витрати на використання Titanium можуть бути значними, якщо вам потрібні будь-які функції корпоративного рівня. Вартість не є загальнодоступною та залежить від кожної програми та угоди між розробником і відділом продажів.

Adobe Air є досить простим інструментом для вивчення для багатьох розробників Flash у галузі. Це простий і природний розвиток платформи Flash, який дозволяє використовувати міжплатформні програми. Здається, деякі компоненти Flex працювали дуже повільно, але альтернативні бібліотеки Flash були доступні для вирішення цієї проблеми. Подібним чином, перегляд дизайну

перетягування не працює дуже добре чи узгоджено на різних платформах, тому програмне створення компонентів може бути найкращим варіантом у багатьох випадках. Використання ActionScript для розробки інтерфейсу користувача забезпечує дуже швидке реагування візуальних компонентів і може бути скомпільовано у файл SWC, що дозволяє швидко відтворювати їх. Тести інтерфейсу користувача показали, що це було ефективним, і Air продемонстрував переважну перевагу.

Інтерфейс користувача дозволяє використовувати кілька методів введення та досить гнучко. Air не оновлюється так часто, як інші платформи, і, здається, у бібліотеках ActionScript не було багатьох необхідних нам функцій. Розробка для Adobe Air безкоштовна; однак деякі вдосконалені інструменти мають свою ціну. Ці інструменти добре розроблені та допомагають у процесі розробки. Загальний досвід використання Air був безболісним. Більшість API були доступні, лише деякі тести були неможливими. ActionScript є потужною мовою, яка надає дуже описові попередження, щоб запобігти проблемам. Хоча компонент ActionScript був дещо обмеженим у порівнянні з використанням Air із Flex, він забезпечив задовільний досвід для кросплатформної розробки. Хоча Air достатньо, він значною мірою покладається на використання специфічного для платформи коду, коли не використовується ActionScript. Це дещо обмежує нашу мету щодо створення кросплатформних тестів, і не все було можливо.

Хоча **MoSync** і здавався чудовим у порівнянні функцій, він не виправдовувався очікування. Інструмент мав багато помилок і надавав дуже мало інформації 74 опишіть, що саме пішло не так, тому, хоча це може бути проблема розробника, виявити це важче, ніж на інших платформах. Проблеми з відсутністю регулярних виразів, використанням прапорців і асинхронних методів перешкодили включенню MoSync до багатьох порівняльних тестів. Незважаючи на те, що надається середовище розробки, інсталяція на емуляторі здається набагато повільнішою, ніж безпосереднє використання пристрою з повільною компіляцією. Надане IDE не дуже добре підходить для кодування на C і C++, і хоча його інтерфейс досить надійний, завершення коду виконується

досить неточно.

Інструкції з налаштування часто були незрозумілими, і здавалося, що компіляція iOS буде невдалою, якщо ви спробуєте скомпілювати той самий код знову, він може спрацювати. Загалом середовищу розробки MoSync бракує зрілості, і здається, що методи крос-компіляції, які воно надає, не запропонували жодного підвищення продуктивності. Компоненти інтерфейсу були надійними та доступні безкоштовні рівні, але головною проблемою є стабільність середовища. MoSync пропонує багато форм розробки додатків, де ми зосередилися виключно на крос-компіляції з використанням коду C. Інші методи можуть бути кращими.

Рідні інструменти . Під час створення тестів також було зібрано певний досвід використання нативних інструментів розробки. Загалом рідні інструменти забезпечили кращий досвід розробки з деякими CPDT, близькими до них, як Adobe Air. Рідні інструменти на iOS і BlackBerry також показали кращі результати під час тестування, але їх легше розробляти. З Android у нас виникли проблеми, оскільки вважалося, що програма не відповідає під час виконання певних завдань, які потребують тривалого часу обробки та обмеженого оновлення інтерфейсу користувача. Функція автоматичного створення, налагодження та інтеграція Android SDK заощадили чимало часу порівняно з використанням CPDT. Це полегшило пошук основних помилок і швидко їх усунення. З iOS рідне середовище розробки використовує Objective C, досить складну мову з незвичним синтаксисом. Продуктивність і швидкість тестування та налагодження були чудовими з нативними інструментами, однак виникали труднощі з використанням регулярних виразів. Однак інтерфейси, які можна створити, досить дружні; використання розкадровки для їх створення має круту криву навчання.

Інструменти розробки BlackBerry 10 все ще перебувають на ранніх стадіях, про що свідчить відсутність таких важливих речей, як API контактів. Під час компіляції він часто згадував невирішені включення, коли вони були додані відповідно до специфікацій і скомпільований файл працював правильно. IDE є

корисною, але не так добре завершує код і виявляє помилки, ніж IDE Android. Помилки не підкреслюються до часу компіляції та не показують, чи були вони виправлені, до нової компіляції, що може ввести в оману. Завершення коду було повільним, але швидшим, ніж MoSync. Розробка за допомогою WebWorks була дуже схожа на PhoneGap. Для багатьох тестів ідентичний код було скомпільовано за допомогою обох інструментів, але WebWorks показав кращі результати.

Ті самі проблеми очевидні через відсутність адекватної IDE та емулятора. Інтерфейси API обмежені, але дозволяють додаткові налаштування за допомогою Java і розширень спільноти. Загальна продуктивність програми була досить хорошою для цього рідного інструменту. Для кожного з рідних інструментів вони отримують оновлення, щойно їх випускають, оскільки вони надходять від самого постачальника. Це може допомогти надати користувачам найновіші функції, а також забезпечити стабільність, коли додаються нові функції та вдосконалення. Витрати на розробку дуже великі, більшість пропонує безкоштовний рівень, єдиним винятком є публікація програми для iOS.

Результати оцінювання цих 4 CPDT і відповідних нативних інструментів дали ряд переможців і програшів. У деяких випадках кросплатформні інструменти працювали набагато краще в контрольованих експериментах, ніж нативні, але в інших набагато гірше. PhoneGap у середньому показав найгірші результати, а iOS Native була найкращою на своїй платформі. Під час тестування Adobe Air і Appcelerator Titanium були найбагатшими та найефективнішими інструментами, однак обидва все ще мають деякі недоліки, про які йшлося раніше. Інструменти Adobe працювали добре, але вимагали більше специфічного коду для виконання певної дії, ніж інші. Titanium також вимагав модифікації, оскільки користувальницький інтерфейс потрібно розробляти незалежно як для Android, так і для iOS. Однак було виявлено, що це забезпечує кращу продуктивність порівняно з PhoneGap, який надав найбільш портативний код у групі. Явний переможець не очевидний; однак тести показують, що залежно від необхідної розробки найкраще використовувати різні інструменти. Завдання з

інтенсивним інтерфейсом користувача віддадуть перевагу Titanium або Air, де прості програми будуть найкращими на PhoneGap для охоплення найширшої аудиторії. MoSync розглядався як розчарування у функціях, продуктивності та досвіді розробки, хоча це може змінитися, коли інструмент розвивається.

Для будь-якого розробника, який розглядає можливість використання CPDT, буде життєво важливо ознайомитися з такими оцінками, як ця, щоб знайти відповідні функції для своєї програми перед початком розробки. Результати показують, що CPDT можуть конкурувати зі своїми вітчизняними колегами в багатьох аспектах, і їх слід ретельно розглядати для розробки. Набори функцій стають більш надійними, а час економиться завдяки створенню програм за допомогою інструментів, а не кількох власних версій. Результати показують, що рідні інструменти не обов'язково означають, що програма матиме найкращу продуктивність, але натомість ви повинні підібрати інструмент до своєї програми. Сам фреймворк досить добре надавав повне уявлення про можливості та продуктивність CPDT. Структуру функцій високого рівня можна було розбити на різні компоненти та застосувати для оцінки цих CPDT, що забезпечує найкращий доступний огляд. Тести продуктивності показали перше тестування між яблуками, порівняння часу завершення алгоритму з використанням коду, скомпільованого за допомогою кількох CPDT. Ця інформація надає розробникам необхідну інформацію для вибору правильного CPDT для своєї програми. Результати цієї оцінки також показали, що поточний набір CPDT часто має слабкі сторони порівняно з їхніми рідними аналогами. Щоб усунути цю проблему, необхідно розробити методи оптимізації, які дозволять кросплатформним розробникам бути на більш рівних умовах для тих, хто використовує рідні інструменти.

У цьому розділі ми побачили реалізацію середовища оцінки, застосованого до PhoneGap, Appcelerator Titanium, Adobe Air, MoSync і колекції рідних інструментів. Тести було реалізовано для різноманітних областей дослідження в рамках, щоб показати життєздатність кожної частини. Кожен інструмент показав себе по-різному під час порівняння функцій. Кількість підтримуваних функцій

дуже різнилася і показала, що важливо пройти перевірку функцій перед вибором інструменту для розробки.

Результати цього оцінювання показали, що фреймворки, навіть із майже ідентичним кодом, дуже відрізняються під час тестування продуктивності. Ці відмінності іноді дуже значні та можуть стримувати розробників від використання таких інструментів, як MoSync або PhoneGap, які в порівнянні з ними погано працюють. Рідні інструменти показали себе досить добре, особливо на платформі iOS, де вони забезпечили найкращу продуктивність у більшості тестів. Аспекти різних інструментів показали, де одні мають труднощі, а інші ні. Порівняно з більшістю рідних інструментів середовищі розробки виявилося недостатнім. Ми побачили, що CPDTs не отримали такої високої оцінки, як деякі рідні інструменти. У наступному розділі ми зробимо висновки та обговоримо методи розширення цього дослідження.

3.6 Висновок по розділу

Вибір інструменту CPDT безпосередньо впливає на ефективність розробки, продуктивність додатків та їх здатність адаптуватися до вимог різних платформ.

Продуктивність та специфікації платформи: Рідні інструменти забезпечують найкращу продуктивність, особливо для додатків, що потребують інтеграції з рідними можливостями пристроїв та високих вимог до продуктивності.

Кросплатформні інструменти: Adobe Air та Appcelerator Titanium продемонстрували добрі результати за функціональністю та ефективністю. Хоча вони потребують додаткових зусиль для налаштування інтерфейсів під різні платформи, вони забезпечують хорошу продуктивність і є оптимальними для більш складних додатків з кросплатформною підтримкою. Проте, вони мають обмеження, які варто враховувати, такі як необхідність модифікації коду для різних платформ.

PhoneGap: Підходить для швидкої розробки простих додатків з кросплатформною підтримкою. Однак його продуктивність значно поступається іншим інструментам, що робить його менш ефективним для складних або ресурсоємних додатків.

MoSync: Платформа MoSync не виправдала сподівань через низьку стабільність і повільну компіляцію. Хоча вона має потенціал для майбутнього розвитку, наразі вона не є оптимальним вибором для серйозної розробки мобільних додатків.

Рекомендація: Для розробки мобільних додатків необхідно ретельно підходити до вибору інструменту, враховуючи вимоги до продуктивності та специфікацій платформи. Важливо також проводити тести на обраних інструментах, щоб забезпечити найкращі результати для кожного конкретного проекту.

Це дослідження підкреслює важливість перевірки можливостей кожного інструменту перед тим, як вибрати його для розробки, адже навіть невеликі відмінності у функціональності та продуктивності можуть значно вплинути на успіх проекту.

ВИСНОВКИ

Розробка програм для сучасного різноманітного ринку мобільних платформ є трудомістким і складним завданням. Ми побачили, що розробники мають відкрите питання щодо того, чи можна використовувати CPDT для економії часу розробки, зберігаючи при цьому певний рівень продуктивності та функціональності. Розробивши структуру для оцінки CPDT у цій дисертації, ми розробили метод відповіді на це питання та застосували його до багатьох інструментів. Це показало, що структура надає деталі, необхідні для відповіді на це запитання через порівняльний аналіз та оцінку.

Наше тестування показало, що явні відмінності у вибраних інструментах можуть мати великий вплив як негативно, так і позитивно на розробку мобільного додатку. Було показано, що деякі CPDT мають проблеми з продуктивністю, тоді як інші надають занадто мало функціональних можливостей. Найбільш привабливою частиною цього фреймворку є те, що його можна розширити до нових функціональних можливостей, які принесуть майбутні випуски CPDT, при цьому основні концепції залишаються. Ми побачили, що для розробників важливо вибрати правильний інструмент для їхньої конкретної програми, і ця структура оцінки надає достатньо деталей щодо перевіреного CPDT, щоб дозволити розробникам вибрати, який інструмент підходить для них. Внески цієї дисертації:

- Високорівнева розширювана структура для оцінки будь-якого CPDT,
- Реалізація фреймворку з конкретними критеріями тестування, які можна використовувати для CPDT на ринку,
- Спеціальні процедури тестування для тестів мобільних пристроїв,
- Широка оцінка кількох популярних кросплатформних інструментів із використанням стандартних тестів і нових спеціально розроблених тестів,
- Створення багаторазових тестів і скриптів, які можна перевести в API.

Завдяки нашим власним розробкам і оцінкам ми побачили, що

використання CPDT все ще має багато недоліків. Проблеми з продуктивністю та дещо крута крива навчання залишаються, поки документація вдосконалюється. Для кожної платформи необхідно враховувати інтерфейс користувача, і оцінка показала, що наразі неможливо написати один раз і запустити будь-де.

Оскільки виробники платформи мобільних ОС роблять свої браузери більш сумісними зі стандартами, такі веб -інструменти, як PhoneGap, покращаться завдяки інтеграції HTML5. Кожен із цих інструментів все ще знаходиться в зародковому стані, і з часом буде додано більше API і знято обмеження . Процес розробки мобільних додатків дуже плавний, і, схоже, це буде актуально протягом наступних кількох років. Використання цієї системи оцінювання та постійне оновлення результатів допоможе розробникам розібратися зі складним завданням вибору CPDT. Через велику кількість доступних платформ розробники мають бути дуже різноманітними, а підприємствам потрібно витрачати значні ресурси, щоб мати доступ до своїх програм на кількох пристроях, тому інтерес до використання цих інструментів лише зростатиме.

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА

1.Freeman A. Pro .NET MAUI: Cross-Platform App Development for iOS, Android, macOS, and Windows. - Apress, 2023. - 400 с. - Режим доступу: https://www.apress.com/gp/book/9781484295809

2.Hermes D. Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals. - Apress, 2015. - 432 с. - Режим доступу: https://www.apress.com/gp/book/9781484202159

3.Snell J., Powers B. Flutter in Action. - Manning Publications, 2020. - 368 с. - Режим доступу: https://www.manning.com/books/flutter-in-action

4.Masiello E., Friedmann J. Mastering React Native: A Guide to Building Professional Mobile Apps. - Packt Publishing, 2017. - 496 с. - Режим доступу: https://www.packtpub.com/product/mastering-react-native/9781785885785

5.Horton J. Ionic Framework By Example: Build Cross-Platform Mobile Apps with Ionic. - Packt Publishing, 2016. - 234 с. - Режим доступу: https://www.packtpub.com/product/ionic-framework-by-example/9781785282720

6.Hansson D. H., Heinemeier D. Agile Web Development with Rails 7: Build Cross-Platform Applications. - Pragmatic Bookshelf, 2023. - 480 с. - Режим доступу: https://pragprog.com/titles/rails7/agile-web-development-with-rails-7/

7.Griffith A. Apache Cordova in Action: Developing Cross-Platform Mobile Apps. - Manning Publications, 2015. - 264 с. - Режим доступу: https://www.manning.com/books/apache-cordova-in-action

8.Biessek A. Cross-Platform Development with Qt 6 and Modern C++. - Packt Publishing, 2021.- 426 с.- Режим доступу: https://www.packtpub.com/product/cross-platform-development-with-qt-6-and-modern-c/9781800204584

9.Troelsen A., Japikse P. Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming. - 10th ed. - Apress, 2022. - 1264 с. - Режим доступу: https://www.apress.com/gp/book/9781484278680

10. Johnson R. Professional Java Development with the Spring Framework. –

Wiley, 2005.-672c.- Режим доступу: https://www.wiley.com/en-us/Professional+Java+Development+with+the+Spring+Framework-p-9780764574832

11. Holmes J. Getting Started with React Native: Build Cross-Platform Mobile Apps. - Packt Publishing, 2015. - 172 с. - Режим доступу: https://www.packtpub.com/product/getting-started-with-react-native/9781785885181

12. **Pekka Abrahamsson та ін. (2017)** - *Agile Software Development Methods: Review and Analysis* . Систематичний огляд agile-методологій, застосованих і в крос-платформі arxiv.org.

13. **Juliano Zanuzzio Blanco & Daniel Lucrédio (2021)** - *A holistic approach for cross-platform software development* . Пропонують високий рівень абстракції для кросс-платформної розробки arxiv.org.

14. **Tony Clark та ін. (2015)** - *Applied Metamodelling* . Про підходи до мовного моделювання, важливі для DSL та генерації коду під різні платформи arxiv.org+ 1uacademic.info+ 1.

15. **Vaquero-Melchor D. et al. (січень 2025)** SARA: A Microservice-Based Architecture for Cross-Platform Collaborative Augmented Reality - пропонує *мікросервісну архітектуру для моделей кроссплатформенного AR* arxiv.org.

16. **Gao Y. та ін. (Сентябрь 2024)** *A Rule-Based Approach for UI Migration from Android to iOS — метод автоматизованої міграції UI між Android* та iOS (GUI-MIGRATOR) arxiv.org.

17. **Hau K. та ін. (лютий 2025)** LLMs in Mobile Apps *: Практичні, Challenges, та Opportunities — дослідження інтеграції LLM у мобільні приложения Android, виклики та паттерні* reddit.com+ 6arxiv.org+ 6moldstud.com+ 6.

18. Ciman, M., & Gaggi, O. (2017). Evaluating Power consumption of cross-platform frameworks для mobile development. Енергоспоживання як критерій вибору методології.

19. "The Pragmatic Programmer" (Hunt & Thomas, 2019) - хоч не про кросс-платформу безпосередньо, має добру методологічну базу (DRY, чистий код)

20. **Julian Smart, et al.** - Cross-Platform GUI Programming with wxWidgets . Огляд GUI-фреймворку та патернів для розробки на C++ ela.kpi.ua+ 6amazon.com+ 6repository.kpi.kharkov.ua+ 6.

21. **Mojtaba Shahin та ін. (2017)** - *Continuous Integration, Delivery and Deployment: A Systematic Review* . Неперервна інтеграція є важливою для підтримки проектів на різних платформах. arxiv.org.

22. Zammetti F. Practical Flutter: Improve Your Mobile Development with Google's Latest Open-Source SDK. - Apress, 2019. - 384 с. - Режим доступу: https://www.apress.com/gp/book/9781484249710

23. Bhargava A. Grokking the JavaScript Interview: A Comprehensive Guide to Cross-Platform Development. - Leanpub, 2023. - 250 с. - Режим доступу: https://leanpub.com/grokking-the-javascript-interview

24. Deeleman D. Learning Angular: A Hands-On Guide to Building Cross-Platform Applications. - 4th ed. - Packt Publishing, 2020. -444 с. -Режим доступу: https://www.packtpub.com/product/learning-angular-4th-edition/9781838648800

25. Vu M. Cross-Platform Desktop Applications: Using Electron and NW.js. - Manning Publications, 2017. - 312 с. - Режим доступу: https://www.manning.com/books/cross-platform-desktop-applications

26. Microsoft Docs. .NET MAUI Documentation: Cross-Platform App Development. - 2023. - Режим доступу: https://docs.microsoft.com/en-us/dotnet/maui/

27. Google Developers. Flutter: Build Apps for Any Screen. - 2023. - Режим доступу: https://flutter.dev/docs

28. React Native Documentation. React Native: A Framework for Building Native Apps Using React. - 2023. - Режим доступу: https://reactnative.dev/docs/getting-started

29. Ionic Framework. Ionic Docs: Cross-Platform Mobile App Development. - 2023. - Режим доступу: https://ionicframework.com/docs

30. Qt Documentation. Qt for Cross-Platform Development. — 2023. — Режим доступу: https://doc.qt.io/qt-6/cross-platform-development.html

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи бакалавра: «Методології крос-платорформенної розробки»

Обсяг пояснювальної записки: 60 аркушів

Дата закінчення бакалаврської роботи 10 червня 2024р.

Підпис студента _____