Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи і засоби розробки нативних додатків для платформи Android

(тема)

Виконав:

Студент II курсу, групи СПм-19-1
Янчук К.В.
(прізвище, ініціали)

Спеціальність
123 – Комп'ютерна інженерія
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Системне програмування
(повна назва освітньої програми)

Керівник: доц. Федорченко В.М.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп'ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. Кафедри _____
(підпис)
"____" _____ 20___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Янчук Катерині Володимирівні
(прізвище, ім'я, по батькові)

1. Тема роботи Методи і засоби розробки нативних додатків для платформи Android

затверджена наказом по університету від " 30 " жовтня 2020 р. № 1486 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 14 грудня 2020 р.

3. Вхідні дані до роботи 1) Документація Android SDK; 2) Документація Xamarin; 3) Android Studio; 4) Visual Studio 2017; 5) Документація по C# 7.0/.NET.

4. Перелік питань, що потрібно опрацювати в роботі

1) Огляд існуючих типів мобільних додатків;
2) Огляд мов програмування і фреймворків для розробки під Android;
3) Визначення вимог для нативних додатків;
4) Вибір технологій для розробки;
5) Порівняння додатків, висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Демонстраційні матеріали. Презентація – 11 слайдів.

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		Підпис	Дата
	залишити пустим, якщо кон-		
	сультантом є керівник роботи		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд існуючих типів мобільних додатків	03.11.20-09.11.20	
2	Огляд технологій для розробки під Android	10.11.20-14.11.20	
3	Визначення вимог для порівняння	15.11.20-19.11.0	
4	Вибір та обґрунтування технологій	20.11.20-23.11.20	
5	Розробка, проведення експериментів	24.11.20-01.12.20	
6	Оформлення матеріалів атестаційної роботи	02.12.20-07.12.20	
7	Подання атестаційної роботи керівникові та	08.12.20-09.12.20	
	її попередній захист		
8	Подання атестаційної роботи на рецензування	10.12.20-11.12.20	

Дата видачі завдання 02 листопада 2020 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Федорченко В.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 68 с., 25 рис., 5 табл., 3 дод., 20 джерел.

НАТИВНИЙ ДОДАТОК, ОПЕРАЦІЙНА СИСТЕМА ANDROID, ANDROID SDK, XAMARIN.

Метою атестаційної роботи є підвищення ефективності використання різних платформ для розробки нативних додатків під OS Android.

Об'єктом дослідження є процес розробки мобільних додатків з однаковим функціоналом, але використовуючи різні технології.

Предметом дослідження є засоби та методи для розробки нативних додатків для ОС Android.

У ході виконання атестаційної роботи було розглянуто види мобільних додатків, їх недоліки та переваги, а також існуючі технології для розробки під Android. В ході виконання основна увага приділялася мовам програмування і фреймворкам. Були визначені вимоги та розроблено два нативних мобільних додатка з ідентичним функціоналом, але розроблені на різних технологіях. Додатки були виконані з використанням мови програмування Java і C#, використані Android SDK і Xamain відповідно.

ABSTRACT

Master’s thesis: 68 pages, 25 figures, 5 tables, 3 appendices, 20 sources.

NATIVE APPLICATION, ANDROID OPERATING SYSTEM, ANDROID SDK, XAMARIN.

The major goal of this thesis is to increase the efficiency of using different platforms for the development of native applications for Android.

The object of research is the process of developing mobile applications with the same functionality, but using different technologies.

The subject of this thesis are tools and methods for developing native applications for Android.

During the certification work, the types of mobile applications, their disadvantages and advantages, existing technologies for Android development were considered. During the execution the main attention is paid to programming languages and frameworks. The requirements were defined and two mobile applications with identical functionality, but on different technologies, were developed. Applications were made using programming language Java and C#, using Android SDK and Xamarin respectively.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Мобільний додаток (застосунок) – програмне забезпечення, призначене для роботи на мобільних пристроях

Нативний додаток – це додаток побудований повністю на засадах технологій, що відносяться до тієї чи іншої операційної системи

ОС – операційна система

Android – операційна система і платформа для мобільних пристроїв, створена на базі ядра Linux

IDE – інтегроване середовище розробки (англ., Integrated development environment)

IL – Intermediate Language

NDK – Native Development Kit

SDK – Software Development Kit

ВСТУП

Згідно зі статистикою використання різних мобільних операційних систем в світовому масштабі на 2020 рік Android є системою, що використовують більше 70% користувачів. Існує безліч фреймворків, за допомогою яких можна розробити додатки під ОС Android, але використання різних технологій може обмежити можливості конечного продукту.

Тому розвиток мобільних платформ вимагає чіткого розуміння, які платформи необхідно використовувати для вирішення задач, що виникають. Отже мета атестаційної роботи – підвищення ефективності використання різних платформ для розробки нативних додатків під OS Android – є актуальною.

Об'єктом дослідження є процес розробки мобільних додатків з однаковим функціоналом, але використовуючи різні технології.

Предметом дослідження є засоби та методи для розробки нативних додатків для ОС Android.

В ході виконання роботи було проведено аналіз типів мобільних додатків, розглянута платформа Android, проаналізовано технології для розробки під вищезазначену операційну систему. На основі вибраних платформ було розроблено і порівняно два додатка на мові програмування Java і C#, використані Android SDK і Xamain відповідно.

1 АНАЛІЗ ЗАСОБІВ ТА МЕТОДІВ ДЛЯ РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ

1.1 Види мобільних додатків

У даний час розробка мобільних додатків є одним з найпопулярніших завдань у сфері інформаційних технологій. Пов'язано це з тим, що мобільні пристрої стали незамінним інструментом для вирішення великої кількості завдань. Сьогодні мобільний додаток – це спеціально розроблене під функціональні можливості гаджетів програмне забезпечення. Загалом додатки переслідують, як мінімум, дві явні цілі: не дати власнику нудьгувати та полегшити побут – іншими словами некорисні та корисні мобільні додатки. Тому призначення ПО може бути найрізноманітнішим, а саме: обмін даними, швидкий доступ до актуальної інформації, мобільні сервіси оплати, контроль протікання бізнес процесів, аналітичні звіти тощо. Функціональність додатків є надзвичайно різноманітною: від ігор та сервісів виклику таксі, до офісних програм та фітнес-трекерів.

Тому ринок мобільних пристроїв безперервно зростає і розвивається. Приблизна кількість розробників мобільних додатків на сьогоднішній день становить 2,3 млн чол. Це означає, що кожен восьмий з усіх розробників в світі створює мобільні додатки. Ці додатки завантажуються і встановлюються самим користувачем через мобільні маркетплейси. Найбільші майданчики – AppStore, Google Play. Технічно всі програми створюються під конкретну платформу мобільного гаджета і називаються нативними. Найбільш популярні операційні системи – iOS, Android, Windows Phone. У 2018 р компанія «Apple» під час World Wide Developer Conference оголосила, що в AppStore опубліковано вже 1,6 млн додатків, які користувачі скачали 50 млрд раз, а розробники отримали дохід в 5 млрд дол. США.

На рисунку 1.1 зображена кількість додатків для кожної з

найпопулярніших платформ. 450 тисяч додатків існують на обох платформах, вони є кросплатформними. Це більше 28% додатків в Apple App store і 14% в Google Play Store. Це виглядає досить вагомою частиною, але частка нативних додатків поки що займає більшу частину.

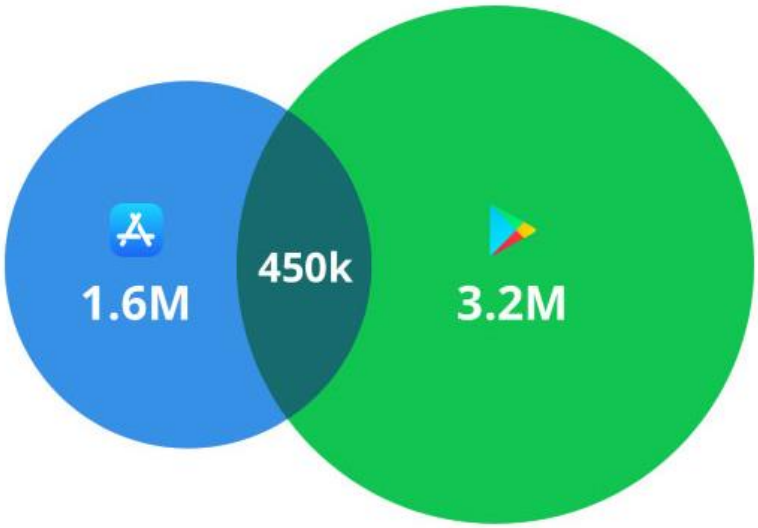


Рисунок 1.1 – Частка нативних і кросплатформних додатків на ОС iOS і Android

Але не дивлячись на таку популярність мобільних додатків, деякі люди не розуміють навіщо вони потрібні, якщо існують веб-сайти. Виділимо моменти, що можуть відлякувати власників сайтів створювати мобільні додатки [1]:

- якщо розмістити QR-коди або якорі на сайт і на додаток, то переходів на сайт завжди буде більше ніж установок мобільного додатку. Частина користувачів після переходу за посиланням на додаток в магазин все-таки його не встановлює;

- кросплатформеність – сайт доступний на всіх платформах і пристроях, навіть на тих, про існування яких мало хто знає;

- миттєві оновлення – для оновлення програми в магазинах завжди

потрібен певний час на їх перевірку. Оновлений сайт доступний користувачам практично відразу;

- вартість розробки й підтримки сайту, як правило, завжди нижча ніж відповідні цифри щодо мобільних додатків;

- сайт просувати, однозначно, дешевше аніж мобільний додаток. Ціна одного користувача безпосередньо залежить від порогу його входження, чим він вищий – тим вища ціна його залучення. Для забезпечення заходу користувача на сайт можна встановити певну вартість кліку по рекламному блоці, після якого можна відразу отримати відвідувача. У випадку з мобільним додатком по суті треба зробити те ж саме, тільки не факт що після переходу в магазин додатків, користувач його встановить. Як наслідок – ціна установки завжди значно вища ціни кліка.

Але на противагу, від мобільних додатків можливо отримати беззаперечні плюси:

- повна взаємодія з користувачами. На відміну від мобільних версій сайтів, у випадку з мобільним додатком є можливість відправляти Push-повідомлення. Наприклад, якщо на сайті з’явилася нова публікація, користувачі дізнаються про це тільки коли туди зайдуть. З мобільним додатком можна просто відправити усім активним користувачам повідомлення, обмеження по символам, як в SMS-кампаніях не має;

- локальні оповіщення працюють приблизно однаково. Якщо користувач виконав якусь дію в додатку, яке очікує від нього реакції через певний час, додаток нагадає йому про це;

- так само з’являються більш широкі можливості зворотного зв’язку – користувачі можуть залишити повідомлення, свої відгуки в магазинах додатків, персонально через додаток, соцмережах – і вони автоматично транслюватимуться на усі ресурси;

- якісніший інтерфейс. Всі елементи управління: кнопки, текстові поля, посилання повинні бути зручними для натискання пальцем на мобільному девайсі, а не для курсора миші. Тож невідомо як виглядає сайт на

тому чи іншому пристрої, тому не зрозуміло чи зручно або навіть реально ним користуватися. Мобільні операційні системи для таких ситуацій мають свою спеціально навчену логіку, щоб відрізняти пристрій із яким працюють. Але ця логіка працює по різному на різних платформах і пристроях;

- якість інтерфейсу позначається і в навігації. Кожна мобільна операційна система має свою логіку переходу між робочими екранами в додатках. Користувач кожної операційної системи звик до тієї ж поведінки в кожному додатку. Навігація ж на кожному сайті зроблена по-своєму, і зайшовши на черговий з них потрібно кожного разу шукати очима кнопки «Ок», «Назад», «Скасування» т.д., яких може зовсім й не бути;

- високий рівень персоналізації. Мобільні телефони знають про свого власника дуже багато, і використання цієї інформації для збільшення рівня сервісу є однією з ключових причин бурхливого зростання мобільних додатків. У мобільного додатку завжди є можливість запам’ятовувати всі дані користувача і змінювати інтерфейс в залежності від його потреб. Навіть на різних пристроях при включеній синхронізації користувач буде бачити додатки з заповненими персональними даними;

- це ж пункт відноситься і до заліза пристрою – залежності від типу програми, дозволів, підтягуються дані з камери, акселерометра, компаса, барометра і т.п.;

- локалізація – якщо додаток, наприклад, має відобразити список ресторанів, то він це скоріше за все зробить враховуючи відстань від користувача, знаючи його поточне місце розташування. Логіка роботи мобільного сайту завжди простіша, вона може не враховувати безліч даних, які можуть надавати мобільні пристрої;

- робота в офлайні. Інтернет є усюди, але не завжди. Навіть при наявності мобільного інтернет-з’єднання, його якість в середньому гірша ніж якість домашніх й офісних інтернет-ліній. Якщо поставлена ціль, щоб користувачі не втрачали зв’язок з продуктом відразу, як тільки обірвався інтернет-зв’язок, розробка мобільного додатку – єдине можливе рішення.

На основі всього перерахованого можна зробити висновок, що потреба в розробці якісного і відповідного сучасним тенденціям продукту все ж таки зростає.

Залежно від виконуваних функцій слід визначити, якого типу буде додаток. Розрізняють нативні, веб і гібридні додатки. Розглянемо кожний окремо.

Нативний додаток – це додаток побудований повністю на засадах технологій, що відносяться до тієї чи іншої операційної системи. Це може бути Android, IOS, Windows тощо. Для Android нативні додатки, як правило, побудовані із використанням Java, в той час як для IOS, додаток може бути побудовано з використанням Objective C або Swift.

Веб-додатки використовують стандартні веб-технології, такі як HTML5, JavaScript і CSS. Цей підхід дає можливість створити крос-платформні мобільні додатки, які працюють на декількох пристроях.

Гібридні додатки поєднують в собі деякі функції нативних і веб-додатків, а саме: мультиплатформність і можливість використання програмного забезпечення телефону або смартфону. Вони можуть бути завантажені через магазини додатків, і при цьому мають можливість незалежного оновлення інформації (вимагають підключення до Інтернету, оскільки веб частина оновлюється через Інтернет). Гібридні додатки на сьогоднішній день є найбільш популярними, оскільки, розробка відбувається швидше і дешевше, ніж у випадку з нативними додатками, хоча оболонка і написана на «рідній» мові програмування, внутрішня частина програми може бути написана в тому чи іншому обсязі на HTML5. У такому випадку користувач, швидше за все, не помітить різницю між нативним додатком і гібридним [3].

Кожен додаток має свої позитивні та негативні сторони, з якими можна погодитись або ні. Розглянемо їх детальніше.

1.1.1 Нативні додатки

Нативні додатки є унікальними для кожної операційної системи, і таким чином, щоб підтримувати декілька мобільних операційних систем, необхідно створити окремі додатки для кожної ОС. Тому такі додатки, як правило, вимагають спеціалізованого розробника для кожної операційної системи (Java для Android, Objective C/Swift для IOS, C# для Windows), вартість роботи яких значно дорожче, ніж робота одного веб-розробника. Якщо потрібне оновлення, то кожен додаток має бути оновлено незалежно один від одного. Для того щоб додаток підлаштовувався під різні розміри пристроїв/екрану і орієнтації в процесі розробки створюються різні макети.

Розглянемо детальніше переваги нативних додатків:

- просте опанування. Інтерфейс та графічна складова нативних додатків наслідують ідеологію дизайну, закладену у конкретній операційній системі. Положення елементів керування, кольорова палітра, анімація – все це сприймається природним. Тому освоїти такий додаток дуже просто, а його використання не буде викликати дисонансу з вже існуючим досвідом користувача;

- висока швидкість роботи. Нативні додатки оптимізовано під певну мобільну ОС, за рахунок чого вони працюють швидко і надзвичайно стабільно. Окрім того, дані таких додатків зберігаються переважно на пристрої, що також підвищує швидкість їх роботи та зменшує залежність від Інтернету;

- широкі можливості. Такий тип додатків має доступ до всіх можливостей ОС та компонентів пристрою: GPS, камери, календаря, адресної книги та іншого. Завдяки цьому, нативні додатки мають широку функціональність та легко інтегруються між собою.

1.1.2 Веб-додатки

Додатки на HTML5 можуть працювати на різних операційних системах і типах пристроїв. Додаток масштабується залежно від розміру пристрою. Там, де необхідно оновлення, необхідно оновити і протестувати один додаток і він буде доступним для всіх пристроїв відразу.

Щодо вартості розробки, то варто наголосити на тому, що HTML5 додатки, як правило, дешевше розвивати і підтримувати, ніж нативні, оскільки одного додатку буде достатньо для підтримки декількох ОС і тому цей додаток може бути розроблено одним веб-розробником. Розробка додатків під певну платформу вимагає більше теоретичних знань, ніж розробка веб-додатків на HTML5.

Важливим щодо використання HTML5 є питання безпеки, оскільки в мобільних додатках ця проблема стоїть не так гостро, як у веб-додатках. У даному питанні нативні додатки перемагають HTML5. У HTML5 додатках великою проблемою в безпеці є можливість подивитися вихідний код. Це означає, що можна не лише зрозуміти, як воно працює, а й використовувати це у своїх цілях [2].

Грань між web-додатком і мобільним сайтом дуже тонка, але різниця є. Сайт містить більше статичну інформацію і є чимось на зразок цифрової брошури. В web-додатку користувач може управляти якоюсь частиною цієї інформації – створити власні сторінки, міняти місцями посилання, тексти та інше, що робить його дуже динамічним.

Web-додатки – це онлайн-сервіси, які раніше розроблялися на Flash, а тепер на HTML 5. Говорячи про web-додатки, слід зазначити сучасну тенденцію до аналогічних варіантів реалізації сайту для мобільних пристроїв. До таких можна віднести створення:

- адаптивного дизайну. Для реалізації адаптивного дизайну зазвичай використовуються CSS3. Залежно від розміру екрана користувач бачитиме різну картинку. Перевагами адаптивного дизайну є зручність розробки і

постійний URL сайту. Однак є й недоліки, такі як повільне завантаження через великий обсяг сайту і надлишок функцій, не потрібних для портативного доступу;

- RESS (Responsive Design + Server Side). Використання RESS-технології передбачає адаптацію сайту до конкретного пристрою. З плюсів можна відзначити мінімізацію трафіку і можливість використання таргетування, тобто відповідного контенту для різних пристроїв. З мінусів – складність в розробці і не налагоджений механізм визначення пристроїв з боку самих пристроїв;

- окрема мобільна версія сайту. Окрема мобільна версія, вона ж web-додаток, має безліч плюсів, таких як швидкість і зручність користування, а також мінусів, таких як наявність різних URL мобільної і десктопної версії, а для когось і обмеженість в контенті, який урізається для мобільної версії.

1.1.3 Гібридні додатки

Гібридні додатки – це універсальні додатки, які створені для багатьох платформ одразу і мають аналогічну функціональність незалежно від самої платформи. Можна сказати, що гібридний додаток – це по суті веб-сайт у звичному форматі мобільного додатку.

Переваги гібридних додатків:

- універсальність. Гібридні додатки створюються для декількох платформ одразу, після чого відносно просто адаптуються під кожну. Це може включати не тільки версії для iOS та Android, а також веб- та десктопні версії. Завдяки цьому, гібридні додатки доступні значно більшій аудиторії користувачів;

- менша вартість розробки. Такий тип додатків простіший та швидший у розробці, тому їх початкова собівартість нижча, ніж у нативних програм. Це робить гібридні додатки привабливим рішенням для компаній, що мають обмежений стартовий бюджет;

- швидший вихід на ринок. Простіша розробка гібридних додатків надає ще одну перевагу – менший термін розробки, що дозволяє відносно швидко випустити продукт і отримати перший прибуток. Якщо компанія не має можливості витрачати багато часу на розробку, то такий тип додатків є розумним рішенням.

До переваг гібридних додатків також можна віднести наступне:

- гібридні додатки є портативними;
- гібридні додатки можуть бути побудовані практично з тією ж швидкістю, з якою додаток HTML5 (базова технологія ідентична);
- гібридний додаток може бути розроблено практично за тією ж вартістю, що і додаток HTML5, але для більшості фреймворків потрібна ліцензія, яка додає витрати на розробку;
- гібридні додатки можуть бути доступні і поширені через відповідний магазин додатків, так само, як нативні;
- гібридні додатки мають доступ до апаратних ресурсів, як правило, за рахунок власних API.

Щодо недоліків гібридних додатків, то варто пам’ятати про таке:

- не всі апаратні ресурси доступні для гібридних додатків;
- гібридні додатки будуть відображатися для кінцевого користувача, як нативні, але працюватимуть значно повільніше. Те ж саме обмеження на HTML5. Візуалізація складних макетів CSS займе більше часу, ніж візуалізація відповідного макета.

Таблиця 1.1 – Порівняння основних видів мобільних додатків

Тип мобільного додатка	Доступ до функціоналу пристрою	Швидкість роботи	Вартість розробки	Поширення через магазин
Нативний	Повний	Дуже висока	Висока	Доступно
Гібридний	Повний	Дуже висока	Прийнятна	Доступно
Веб-додаток	Частковий	Висока	Прийнятна	Відсутнє

Для того щоб наочно розглянути переваги і недоліки існуючих типів мобільних додатків, як приклад наведена порівняльна таблиця основних критеріїв (Таблиця 1.1). Як основні ознаки варто виділити швидкодію нативних додатків та універсальність використання для різних платформ гібридних.

1.2 Операційна система Android

Сфера використання мобільних додатків розділена на частини, в першу чергу операційними системами, що використовуються в мобільних пристроях. Нижче наведена статистика використання різних мобільних операційних систем в масштабі України на 2020 рік (Рисунок 1.2).

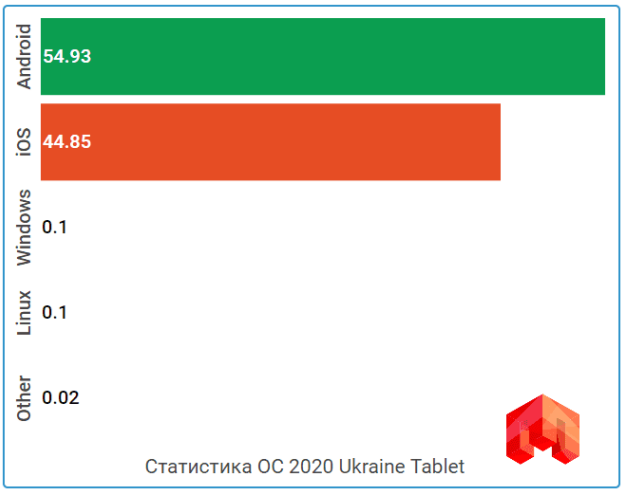


Рисунок 1.2 – Статистика використання ОС для смартфонів в Україні в 2020 році

Але якщо розглядати рейтинг операційних систем в світовому масштабі на 2020 рік то ситуація відрізняється: серед мобільних телефонів лідером залишається Android – 70,43%, iOS на другому місці – 29,06% – практично кожен третій смартфон (Рисунок 1.3).

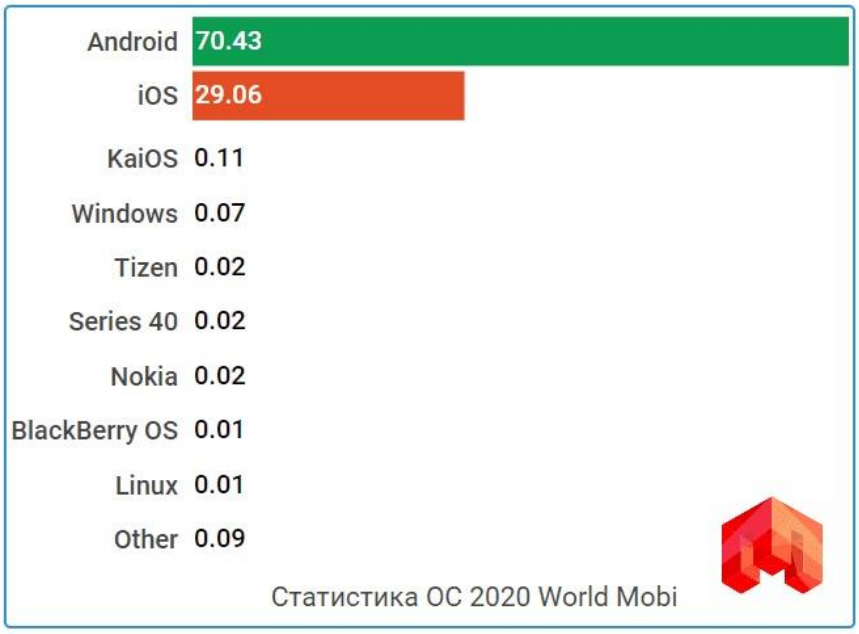


Рисунок 1.3 – Статистика використання ОС для смартфонів в світі в 2020 році

Критерієм вибору операційної системи для розробки також є комерційний успіх додатків. Кілька років тому статистика монетизації додатків говорила про те, що розробники для iOS отримують дивіденди більш ніж в 4 рази більші, ніж розробники для Android. На даний момент ситуація для Android-розробників стабілізується і рівень доходів від створення додатків для Android виходить на один рівень з конкурентами від Apple, що, в свою чергу, робить пріоритетними Android-розробників. Таким чином, очевидно, що при створенні додатка варто віддати перевагу розробці додатків для операційної системи Android [4].

Однак це лише фінансова і статистична сторони питання. Їх варто розглядати в сукупності з тим, який функціонал необхідно отримати після реалізації проекту, що також є критерієм при аналізі технологій розробки мобільних додатків.

Основними перевагами Android є підтримка двух сім-карт, можливість використовувати сторонню карту пам’яті, швидке заряджання, величезний вибір пристроїв з цією оперативною система.

Якщо подивитися на десять iPhone різних власників, все що їх буде відрізняти один від одного – це шпалери на робочому столі і список додатків. На цьому всі можливості кастомізації закінчуються. Якщо ж ми беремо будь-який Android-смартфон, тоді перед нами відкривається величезне поле для самовираження:

- зміна лаунчера. Лаунчер – це типу оболонка смартфона (як виглядає робочий стіл, список додатків, всілякі ефекти);
- організація робочого стола. Можна заповнити робочий стіл іконками, як на iPhone, можна взагалі все іконки прибрати, можна залишити тільки один ряд в будь-якій частині екрану (на iPhone все іконки розташовані в строгому порядку – зверху вниз);
- зміна шрифтів. Якщо користувачю не подобається шрифт на Android смартфоні, можна завантажити будь-який інший шрифт з маркету і використовувати його;
- зміна іконок. На Android присутнє таке поняття як icon pack (або набір іконок). Стиль іконок можна змінити, щоб вони виглядали як на iPhone або щоб все було виконано в чорно-білих тонах. Такі набори також можна знайти в маркеті.

Ще одна дуже важлива можливість смартфона с ОС Android це можливість встановлювати будь-які додатки [5]. У випадку з iPhone можна встановити будь-який додаток виключно з одного джерела – офіційного магазину App Store. Якщо додаток якимось чином порушує правила магазину – його видаляють. Наприклад, в App Store немає жодної програми для безкоштовного перегляду фільмів і серіалів онлайн, що є дуже популярною темою на Android. Також, з iOS не можна завантажити будь-який додаток/гру з інтернету. Для Android існує безліч різних сайтів, форумів, каталогів з додатками, яких немає офіційно в Play Store (магазин додатків від Google). Особливо ця властивість актуальна в даній роботі, адже для порівняння експериментальних додатків немає необхідності відправляти їх в офіційний магазин.

Також дуже важливою особливістю будь-якої операційної системи є її відкритість. Android – це відкрита операційна система і її плюс полягає в тому, що будь-який бажаючий користувач може доопрацювати управління пристроєм під себе, вносячи зміни в програмну частину. Що стосується iOS – вона є закритою, що означає, що в неї вносять доопрацювання тільки фахівці Епл, не дозволяючи звичайним користувачам робити зміни в програмі самостійно. Якщо користувача не влаштовує певний момент в управлінні Епл, в організації роботи апарату, то користувач не може це змінити.

Тому на основі всіх вищеперерахованих показників доцільно вибрати саме платформу Android для розгляду в даній роботі.

1.3 Мови програмування для розробки під Android

Існує безліч основних мов програмування, таких як Java, C#, C++, JavaScript, HTML, CSS, Python, які можна застосовувати при розробці додатків для мобільних пристроїв. Таким чином, можливості сучасних технологій розробки дозволяють створювати мобільні додатки різної складності. Вибір тієї чи іншої платформи залежить від вимог, що пред'являються до майбутнього додатка.

Ринок розробки додатків на ОС Android надзвичайно широкий. Аналітики провідної світової компанії Canalys прорахували, що вже в 2016 році обсяг світового ринку мобільних гаджетів перевищить 1,2 мільярда пристроїв. Тому тема дослідження мов для розробки під цю ОС є досить актуальною.

Java. Основна частина програм для операційної системи Android пишеться на мові Java. Не буде великим перебільшенням назвати Java офіційною мовою Android. У всякому разі, майже вся освітня документація, всі інтернет-курси засновані на цьому. Java – одна з найважливіших і найпопулярніших комп'ютерних мов у світі. Крім того, вона утримує своє лідерство протягом багатьох років. На відміну від деяких інших мов

програмування, вплив яких з часом зменшився, Java стала лише сильнішою. Основною причиною успіху цієї мови є швидка мінливість. Починаючи з першого випуску 1.0, вона безперервно адаптується до змін в середовищі програмування та підходів до програмування. Саме тому вивчення Java має бути першочерговим завданням для будь-якого Android-розробника. Нехай це буде непросто (все-таки мові 22 роки, а легкість ніколи не була її коником), нехай теоретично можна обійтися більш сучасними мовами, проте неможливо домогтися істотних успіхів на Android, абсолютно не розуміючи Java.

Kotlin. Офіційно випущений лише рік тому, Kotlin дуже швидко завойовує серця розробників по всьому світу практично повною відсутністю недоліків. Він схожий на Java, але в багатьох відносинах трохи легше. Як і Java, він працює на віртуальній машині Java. Він повністю сумісний з Java і не викликає ніяких перешкод або збільшення розміру файлів. Використання Kotlin не викликає жодних проблем при розробці нативних додатків для Android. На Kotlin можна розробляти використовуючи середовище IntelliJ IDEA або Android Studio [8].

Основна відмінність полягає в тому, що Kotlin вимагає менше «шаблонного» коду, тобто простіша для читання система. Він також усуває такі помилки, як виняток нульового покажчика, і навіть звільняє Вас від необхідності закінчувати кожен рядок крапкою з комою. Це чудова мова програмування для початку розробки програм для Android. Kotlin є легшою відправною точкою для початківців, і той факт, що можна використовувати Android Studio, є великим плюсом.

При цьому попит на фахівців Kotlin поки низький, а значить, отримавши досвід роботи з ним, можна в майбутньому отримати конкурентну перевагу.

C/C ++. Насправді Google надає розробникам два середовища розробки: SDK, призначена для роботи з Java, і NDK, де нативними мовами є C/C ++. Мова C++ призначена для розробки високопродуктивного програмного

забезпечення та надзвичайно популярна серед програмістів. При цьому вона забезпечує концептуальний фундамент, на який спираються інші мови програмування. Так, написати цілий додаток з використанням лише цих мов буде важче, але з їх допомогою можна створити бібліотеку, яку згодом за допомогою Java можна підключити до основного тіла програми.

Незважаючи на те, що переважній більшості розробників немає ніякого діла до NDK, проте задіявши цей інструмент можна отримаєте кращі результати по продуктивності і використанню внутрішніх ресурсів. А це саме те, що в Android відрізняє хорошу ідею додатку від оптимальної реалізації [5].

C#. При всьому нескінченному скепсисі, спрямованому в бік продуктів Microsoft, варто визнати, що C# цього не заслуговує. Це чудова мова, що увібрала в себе все краще від Java, при цьому врахувавши і виправивши багато недоліків [9].

Що стосується розробки додатків під Android, то тут до ваших послуг одні з найбільш функціональних середовищ Visual і Xamarin Studio. А ще знання C# стане для вас приємним бонусом, коли доберетеся до використання Unity 3D. З таким набором можливості будуть безмежні.

C# трохи зручніше для початківців на відміну від C або C++. Він підтримується деякими дуже зручними інструментами, які відмінно підходять для розробки ігор і крос-платформних додатків.

Мови веба. Стандартний мовний набір працівника веба складається з HTML, CSS і JavaScript. Не знаючи цих технологій можна розраховувати на розробку додатків досить вузької спрямованості. Навіть якщо немає бажання безпосередньо торкатися вебу в майбутній роботі, то гібридних додатків уникнути вийде навряд чи. Працювати з HTML, CSS і JavaScript можна використовуючи середовище Adobe Cordova. Великих знань воно не вимагатиме, а результат забезпечить. Знання вищезазначених технологій також знадобиться для створення елементарних кроссплатформенних додатків за допомогою фреймворка PhoneGap.

Кожна з представлених мов програмування має свої переваги та недоліки. Але для повноцінної розробки додатків на ОС Android найкращою у використанні є Java та Kotlin.

1.4 Платформи і фреймворки для розробки під Android

Спектр мов для розробки під Android дуже широкий. Розглянемо, які платформи і фреймворки можна використовувати з кожною з них [6].

Android SDK – це універсальний засіб розробки мобільних додатків для операційної системи Android. Відмінною рисою від звичайних редакторів для написання коду є наявність широких функціональних можливостей, що дозволяють запускати тестування і налагодження вихідного кода, оцінювати роботу програми в режимі сумісності з різними версіями ОС Android і спостерігати результат в реальному часі. Підтримує велику кількість мобільних пристроїв, серед яких виділяють: мобільні телефони, планшети, розумні окуляри (в тому числі Google Glass), сучасні автомобілі з бортовими комп'ютерами на ОС Андроїд, телевізори з розширеним функціоналом, особливі види наручних годинників і багато інших мобільних гаджетів. Офіційною мовою для розробки є Java [10].

Android Native – це платформа розробки Android, яка входить до числа найпопулярніших у світі платформ для розробки мобільних додатків. NDK надає бібліотеки і інструменти розробника, необхідні для створення, тестування і відладки додатків для платформи Android.

Android Native популярний серед розробників, яким необхідно реалізувати свої застосування в нативному коді, використовуючи мови програмування C і C++. Android Game SDK (одна із складових Android Native) надає розробникам ігор неймовірне охоплення завдяки більш ніж 2,5 млрд щомісячно активних пристроїв на платформі Android.

Android Native використовують Google, Slack і Instacart. Особливості і переваги Android Native полягають в тому, що завдяки використанню C і C++

в розробці додатків Android Native має найшвидший код, який використовують для створення додатків і ігор на Android. Мовами розробки окрім C, C++ є Java.

NativeScript – ще один варіант Android-фреймворка з відкритим кодом для створення мобільних додатків з використанням Angular, Vue.js, JavaScript, TypeScript і CSS при розробці кросплатформних додатків, який використовують якщо необхідно прискорити розробку мобільного додатку.

NativeScript є ідеальним вибором для розробників, яким потрібна швидкість при розробці додатків для декількох платформ. З його допомогою можна отримати 100% прямий доступ до усіх можливостей операційних систем Android і iOS.

NativeScript застосовують такі компанії як NortonLifeLock, Xerox, Dell, Microsoft, а також тисячі розробників, які використовують його для створення кросплатформних мобільних додатків працюючих у сфері B2B.

Особливості і переваги NativeScript полягають у тому, що він надає розробникам все, що треба для створення власних мобільних застосувань з використанням JavaScript, TypeScript, Angular або Vue.js.

Corona SDK є безкоштовним фреймворком для створення додатків і ігор для мобільних пристроїв, настільних комп'ютерів і телевізійних приставок. Використовує єдину кодову базу для усіх пристроїв.

Corona SDK не лише широко використовується розробниками ігор, але також являється і одному з популярних середовищ розробки Android. Базова структура додатка створюється за допомогою мови програмування Lua. Фреймворк працює як на Windows, так і Mac OS X, підтримує тестування в реальному часі, має вбудований призначений для користувача інтерфейс, движок і рекламну платформу.

Corona SDK застосовують для створення ігор Warcraft, Fun Run 2, Angry Birds, The Lost City і Hopiko.

Особливості і переваги Corona SDK полягають у тому, що завдяки простому синтаксису, Corona SDK ідеально підходить для початківців.

React Native визнаний восьмим за популярністю кросплатформним фреймворком для розробки додатків, а також третім найбільш «жаданим» фреймворком в недавньому опитуванні серед 90 тис. розробників. Ця швидко зростаюча платформа з відкритим вихідним кодом набрала 89 тис. зірок на Github. Платформа починалася як внутрішній хакатон-проект Facebook в 2013 р. Через два роки вона була випущена як кросплатформне середовище розробки з відкритим вихідним кодом.

React Native використовує вбудовані компоненти інтерфейсу і API-інтерфейси. Фреймворк дозволяє розробляти мобільні додатки, які майже не відрізняються від нативних додатків, що використовують Java або Kotlin.

За минулі роки React Native отримав значну популярність і використовується компаніями Facebook, Instagram, Tesla, Intuit, Bloomberg, Uber, Yahoo і Walmart.

Особливості і переваги React Native полягають в тому, що віе є дуже популярним фреймворком для розробки елегантних призначених для користувача інтерфейсів на різних платформах. Також допомагає розробникам значно скоротити витрати і час на розробку додатків.

Xamarin – це платформа з відкритим вихідним кодом для створення ефективних додатків Android, iOS і Windows на базі технології .NET. У згаданому вище огляді Xamarin був 10-м за популярністю кросплатформним фреймворком для розробки додатків.

Xamarin має дружнє середовище розробки, а його складова частина Xamarin.Forms дозволяє створювати додатки із застосуванням коду для призначеного для користувача інтерфейсу, написаного на C# або XAML. Xamarin дозволяє розробникам писати всю бізнес-логіку додатка, використовуючи одну мову програмування [7].

Xamarin призначений для розробників, які хочуть обмінюватися кодом і тестувати бізнес-логіку на різних платформах, а також писати кросплатформні додатки на C#, використовуючи середовище розробки Microsoft Visual Studio.

Xamarin застосовують Olo, Rumble, doubleSlash.

Особливості і переваги Xamarin полягають в тому, що це безкоштовний Android-фреймворк з відкритим вихідним кодом для кросплатформної розробки. Ідеально підходить Android- розробникам для створення додатків з використанням технології .NET і мовою програмування C#.

Unity є популярною тим, що вважається кращою платформою для створення 2d і 3d-ігр, віртуальної реальності, ігор з доповненою реальністю і застосуванням штучного інтелекту (Artificial Intelligence). Крім того, згідно з даними дослідження Statista, Unity вважається головним фреймворком для розробки ігор не лише для облаштувань Android.

З Unity можна легко працювати, маючи знання C#. UnityScript – мова Unity, яку легко вивчити і яка нагадує за синтаксисом JavaScript. Плюси цієї платформи – величезна кількість інструментів, висока гнучкість і швидке створення додатків.

Для створення медіаконтенту і ігор Unity активно використовують Magic Design Studios, Pixar, Magnopus, ustwo Games і багато інших.

Особливості і переваги Unity полягають в тому, що інструменти Unity – це комплексне рішення для мобільних ігор, яке дозволяє створювати і поставляти 2d або 3d-ігри по всьому світу. Потужні рішення Unity по оперативному управлінню і монетизації забезпечують високу прозорість, бездоганну продуктивність і зростання доходів.

PhoneGap дозволяє розробляти Android-додатки силами веб-розробки. Програма буде відображатися через WebView, але як би в обгортці мобільного додатка. Для розробників PhoneGap – це щось на зразок моста для доступу до нативних функцій смартфона або планшета на кшталт акселерометра або камери.

Основними причинами використовувати PhoneGap для розробки мобільних додатків на платформі Android є скорочення термінів і вартості розробки програми для декількох платформ відразу та можливість використовувати вже наявного сайту клієнта і його фрагменти.

Особливості і переваги PhoneGap полягають в тому, що на сьогоднішній день PhoneGap являється одним з лідируючих інструментів кросплатформної розробки, за допомогою якого програмісти створюють рішення на CSS3, HTML5 і JavaScript. Ще один плюс фреймворка – безліч наданих готових плагінів [2].

Проаналізуємо платформи і фрейморки, які розглядалися вище, з точку зору можливостей платформи і типу вихідного додатка для визначення підходящої технології:

- Android Native використовується з метою поліпшення продуктивності додатка на Андроїд за допомогою C і C++. Його можна використовувати для оптимізації складних ділянок коду або при імпортуванні існуючих бібліотек. Проте ціллю роботи є розробка додатка, а не його оптимізація;

- NativeScript – як один з варіантів при розробці кросплатформних додатків. Хоча виробники і запевняють, що використання NativeScript прискорить розробку мобільного додатку, дасть можливість писати одразу для декількох платформ і з його допомогою можна отримати 100% прямий доступ до усіх можливостей операційної системи Android, кроспларформна розробка з використанням цього фреймворку все ж має декілька мінусів. Багато плагінів необхідно завантажувати окремо для компонентів. Не всі плагіни доступні або перевірені (ретельно протестовані). Розмір програми набагато більше, ніж, наприклад, ReactNative і Ionic 2. Якщо у ваших користувачів повільне підключення до Інтернету, це може бути проблемою для вас. Аналізуючи всі вище зазначені особливості, можна зробити висновок, що це не найкращий інструмент для розробки суто нативного додатка;

- Solar2D (до 2020 року Corona SDK) теж є фреймворком для створення кросплатформних додатків і спеціалізується на іграх для мобільних пристроїв, настільних комп'ютерів і телевізійних приставок. Використовує єдину кодову базу для усіх пристроїв. Але це все ж

кросплатформий інструмент. До того ж компіляція програми відбувається віддалено на сервері, а не локально і в безкоштовній версії відбувається з деякою затримкою;

- якщо мова зайшла про ігри, то варто згадати Unity, адже ця платформа є популярною тому, що вважається кращою для створення 2d і 3d-ігр. Але версія движка має ряд невирішених проблем з продуктивністю, споживанням пам'яті і працездатністю на мобільних пристроях;

- React Native також є кросплатформним фреймворком. Він використовує вбудовані компоненти інтерфейсу. Фреймворк дозволяє розробляти мобільні додатки, які майже не відрізняються від нативних, які використовують Java або Kotlin. Але все ж додатки на React Native теж мають недоліки – їх складно адаптувати під всі андроїди, тому що занадто великий пул девайсів, дуже багато різних оболонок. За фактом, це скоріше проблема розробки під андроїд, ніж самого React Native, проте для нативного додатка варто пошукати щось інше;

- що стосується PhoneGap то він, по суті, дозволяє розробляти Android-додатки силами веб-розробки. Ваша програма буде відображатися через WebView, але як би в обгортці мобільного додатка. Для розробників PhoneGap – це щось на зразок моста для доступу до нативних функцій смартфона, але на жаль ці додатки працюють повільніше.

У підсумкову таблицю (Додаток В) не включено Solar2D і Unity, тому що ці платформи спуціалізуються на іграх.

Як підсумок варто зазначити, що гібридні інструменти мобільної розробки розвиваються досить швидко, але їм як і раніше не вистачає продуктивності і власних можливостей. Саме це пропонує Xamarin – платформа з відкритим вихідним кодом для створення ефективних додатків Android на базі технології .NET. У той же час порівняння Xamarin і нативних додатків на Android стає більш складним: обидва варіанти доводять, що вони цінні з точки зору якості і продуктивності. До того ж останнім часом багато розробників схильні погоджуватися з тим, що Xamarin може вважатися

нативним інструментом розробки. Справді, існує думка, що «все, що можна зробити в додатку iOS з використанням Objective-C або Swift, і все, що можна зробити в додатку Android за допомогою Java, можна зробити і на мові C# за допомогою Xamarin». Проте існує багато підводних каменів у суперництві нативної платформи і платформи Xamarin, навіть не дивлячись на те, що її продуктивність близька до рідної. Тому в рамках роботи буде доцільно розглянути саме Xamarin і Android SDK.

2 ВИЗНАЧЕННЯ ВИМОГ І ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ НАТИВНИХ ДОДАТКІВ ДЛЯ ОС ANDROID

2.1 Визначення вимог для розробки нативного додатка

Вже згадувалося, що нативні додатки працюють з ОС пристрою так, що мають можливість працювати швидше та гнучкіше, ніж альтернативні типи програм. Наприклад, додаток Facebook колись був написаний в HTML5, щоб використовувати той самий код для iOS, Android та мобільної мережі. Проте програма була повільною для користувачів iOS, що привело розробників Facebook до створення окремого коду для iOS.

Таким чином можна зробити висновок, що не кожний підхід буде однаково гарно вирішувати будь-яку задачу. Що стосується нативних додатків вони повинні підтримувати всі нативні технології і апаратні можливості конкретної платформи, в даному дослідженні ОС Android [4].

Залежно від використаних технологій, на вигляд однакові програми можуть споживати різну кількість ресурсів смартфона. Іншими словами у них буде різна продуктивність. Додатки можуть споживати різну кількість пам'яті, процесорного часу, по різному впливати на витрату батареї, завантажувати більше або менше даних по мережі. В кінцевому підсумку все це впливає на look and feel і продуктивність вашої андроїд програми. Тому в першу чергу при проектуванні самого додатка треба думати про продуктивність.

Дуже зручно, що деякі середовища для розробки навіть дозволяють відстежити всі головні метрики продуктивності вашого додатку. Наприклад, якщо існую проект в Eclipse, можна відкрити перспективу DDMS. Там можна профілювати програми та отримати уявлення про те, скільки часу займає кожен метод/що займає найбільше часу.

Також дуже зручний вбудований в Android Studio профілювальник дозволяє відстежити всі головні метрики продуктивності, такі як: пам'ять, процесор, використання мережі і споживання енергії (Рисунок 2.1).

Рисунок 2.1 – Профілювальник додатку в Android Studio

Метою цього дослідження є огляд методів і засобів для розробки нативних додатків, а задача – розробити додаток з однаковим функціоналом, але виконанний на різних плаформах. Але при такій постійно зростаючій кількості конкурентів, перед розробниками мобільних додатків постає питання – який підхід і технології варто використовувати в розробці, щоб нові програми не тільки мали можливість доступу до нативних функцій, володіли швидкодією і високою продуктивністю, а й були конкурентоспроможні. Наприклад така можливість нативних додатків як позначення геолокації дозволяє компаніям підлаштовувати свої програми таким чином, щоб споживачі могли отримувати повідомлення, коли вони знаходяться біля фізичних магазинів. Дані дій користувача можуть бути легко зібрані та проаналізовані, таким чином полегшуючи оцінку ефективності всієї програми або її окремих функцій.

Для того щоб вибрати платформу, треба чітко знати задачу, яку має вирішувати додаток і яким саме функціоналом він має володіти. Оцінюючи технології розробки мобільних додатків, слід зазначити, що даний напрямок вимагає вибору критеріїв оцінки.

Показники оцінки нативних додатків можна поділити на технічні та якісні. На швидкість та якість розробки впливають такі якісні показники:

- популярність мови;
- наявність фахівців;
- наявність повної документації та технічна підтримка;
- зручність розробки і налагодження (зручний синтаксис (синтаксичний цукор), наявність коллекцій (контейнерів), лямбда-вирази тощо).

Що до технічні, то можна виділити такі показники як:

- вага додатка;
- можливість доступу до нативних функцій (безпроблемна взаємодія програми з мобільною ОС);
- швидкість роботи і відгуку інтерфейсу;
- швидкодія;
- продуктивність (споживання пам'яті, процесорного часу, вплив на витрату батареї, завантаження даних по мережі);
- надійність;
- безпечність;
- легкість підтримки/оновлення.

На основі порівняльної таблиці 1.2, а також якісних параметрів, в рамках роботи буде доцільно розглянути саме Xamarin і Android SDK. Для порівняння таких технічних критеріїв як вага додатка, швидкодія та продуктивність необхідно розробити два додатка, які б задіяли нативні функції, наприклад доступ до телефонної книги.

2.2 Визначення технологій і мов

Вище було зазначено, що Xamarin і Android SDK технології, що найбільше підходять для розробки нативних додатків. Беручи до уваги рейтинг мов програмування на 2020 рік з офіційного сайту спільноти програмістів DOU, очевидно, що Java і C#, посідаючи друге і третє місце, все ще є одними з найпопулярніших мов (Рисунок 2.2).

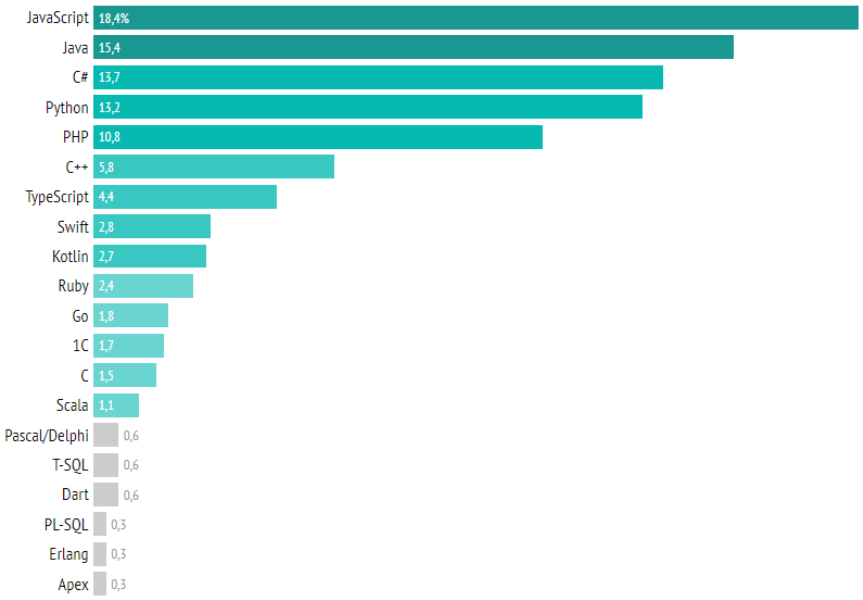


Рисунок 2.2 – Рейтинг мов програмування на 2020 рік

В цілому, програми під Android можна створити практично на будь-якій популярній мові. Однак щоб стати професійним Android-розробником є сенс використовуються всі можливості операційної системи і мати доступ до найновіших функцій Android. Найкраще для цієї задачі підійдуть Java або Kotlin. До того ж якщо розглядати останні новини щодо розвитку Андроїда як оперативної системи, то варто зазначити, що в травні 2019 року компанія Google оголосила Kotlin кращою мовою для Android-розробки, тому суперечки щодо вибору мови розгорілися з новою силою [8]. З одного боку Java все ще можна вважати офіційною мовою андроїда, але вже існує ряд

факторів, які здатні схилити чашу терезів на користь Kotlin, такі як: менша кількість коду; досить передбачувана перспектива розвитку, яка орієнтована саме на Android-розробку; позитивні особливості, яких нема в Java.

Дійсно, Java вимагає написання набагато більшої кількості коду, ніж Kotlin (Рисунок 2.3), тому існує більш високий ризик помилок.

Рисунок 2.3 – Порівняння кількості коду на Java і Kotlin

Нижче наведена узагальнююча порівняльна таблиця 2.1.

Не дивлячись на всі переваги Kotlina, його документація передбачає знання Java [8]. Так, з технічних аспектів різниця суттєва, але щоб займатися Android-розробкою, треба вчити обидві мови. Більшість популярних бібліотек підтримують зворотну сумісність з Kotlin, тобто код Java можна використовувати в Kotlin і навпаки, а ось щоб вирішити проблему або банально зрозуміти всі нюанси документації, варто знати Java.

Що стосується Xamarin, то його можна використовувати за допомогою C# або F#. Xamarin.Android – бібліотека класів для C#, що надає розробнику доступ до Android SDK.

Таблиця 2.1 – порівняльна таблиця мов програмування Java і Kotlin

	Kotlin	Java
Платформи/підтримка	Open source Конвертер Java в Kotlin Підтримка ООП і ФП	Open source (Лише реалізація OpenJDK) ООП
Час компіляції	Повільніше ніж у Java	Швидко
Довжина коду	Коротка	Надмірно багатослівна
Швидкість розробки	Висока	Досить висока
Підтримка ком’юніті	Зростаюча спільнота, слабкий канал	Гігантська спільнота на GitHub, Reddit та StackOverflow
Безпека	Більш безпечно (завдяки нульовій безпеці)	Безпечно

Тому оцінюючи мови, можна зробити висновок, що для порівняння найкращим варіантом буде обрати мови програмування Java і C#, і використати технології Android SDK і Xamain відповідно.

2.3 Xamarin

Xamarin – це платформа з відкритим кодом для створення сучасних та ефективних додатків для iOS, Android та Windows за допомогою .NET. Ця платформа розширює платформу .NET спеціальними бібліотеками для iOS, macOS, Android та інших. Xamarin – це абстрактний рівень, який управляє зв'язком спільного коду з базовим кодом платформи. Xamarin працює в керованому середовищі, яке забезпечує такі зручності, як виділення пам'яті та збір сміття [7]. Ця платформа дозволяє розробникам писати всю свою бізнес-логіку однією мовою і разом з тим досягати нативних характеристик та вигляду на кожній платформі.

Фреймворк складається з декількох основних частин:

- Xamarin.IOS – бібліотека класів для C#, що надають розробнику доступ до iOS SDK;
- Xamarin.Android – бібліотека класів для C#, що надають розробнику доступ до Android SDK;
- компілятори для iOS і Android;
- IDE Xamarin Studio;
- плагін для Visual Studio.

З точки зору виконання додатків між iOS і Android є одна ключова відмінність – спосіб їх попередньої компіляції. Як відомо, для виконання додатків на Android використовується віртуальна Java-машина Dalvik. Нативні додатки, які пишуться на Java, компілюються в проміжний байт-код, який інтерпретується Dalvik`ом в команди процесора в момент виконання програми (тобто аналогічно тому, як працює CLR в .NET). Це так звана Just-in-time компіляція (компіляція на льоту). В iOS використовується інша модель компіляції – Ahead-of-Time (компіляція перед виконанням). Xamarin враховує цю різницю, надаючи окремі компілятори для кожної з цих платформ, які дозволяють на виході отримувати справжні, нативні додатки, які можуть використовувати всі апаратні і програмні ресурси платформи.

Для Android при компіляції програми відбувається переклад коду з C# в проміжний байт-код, зрозумілий віртуальній машині Mono і сама ця віртуальна машина також додається в упакований додаток. І Mono, і Dalvik написані на C і працюють поверх ядра Linux (варто зазначити, що Android заснована на Linux). При запуску програми на Android обидві віртуальні машини починають працювати пліч-о-пліч і обмінюються даними через спеціальний механізм wrapper`ів (Рисунок 2.4).

Xamarin.Native (в даному дослідженні Xamarin.Android) – це чудовий спосіб досягти повторного використання коду та максимально використати навички .NET під час створення мобільного додатка. Використовуючи Xamarin, вся бізнес-логіка та серверний код є повністю спільними та мають

повний доступ до базової платформи. Коли розробник починає писати свій додаток на Xamarin перед ним постає питання: Xamarin.Native або Xamarin.Forms. Зі знаннями .Net можна писати як з Xamarin.Native, так і Forms, але з XF інтерфейс створюється із простим у використанні XAML. З іншого боку, якщо в наявності команда з певним досвідом роботи з Android та iOS, тоді Xamarin.Native може окупитися в плані часу і якості.

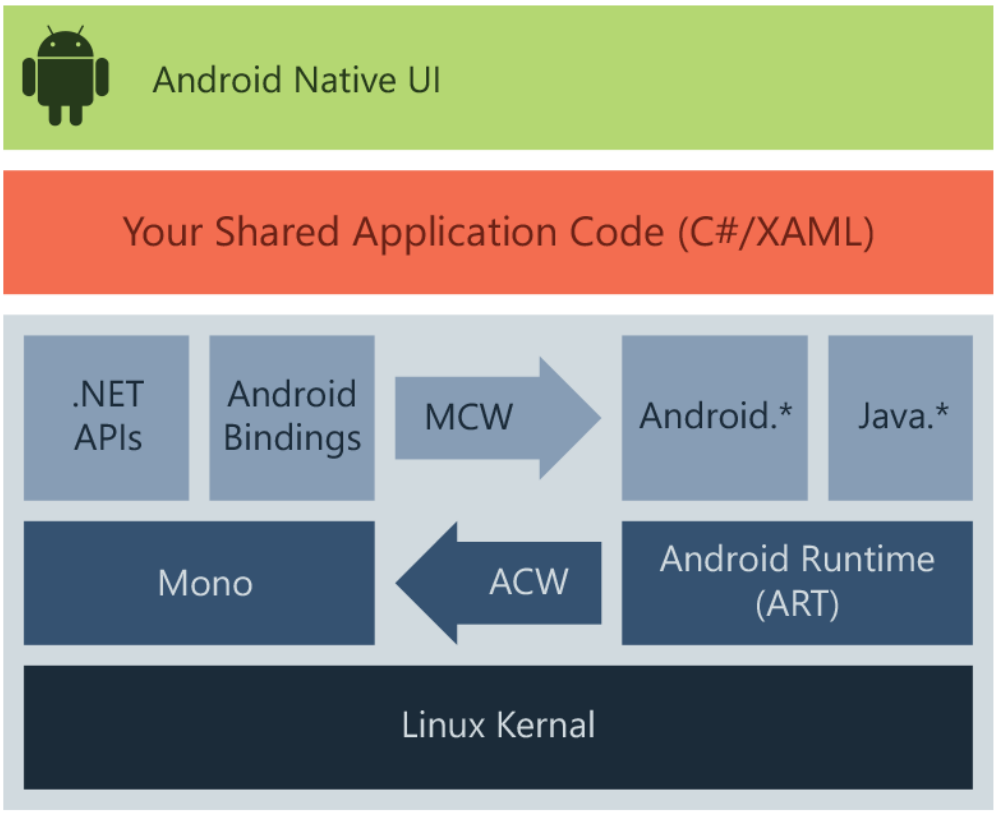


Рисунок 2.4 – Схема архітектури Xamarin.Android

Більшість власників бізнесу вибирають платформу Xamarin, так як це скорочує час виходу на ринок (Time-To-Market) і інженерних витрат за рахунок спільного використання коду і використання єдиного стека технологій.

2.4 Android SDK

Android SDK (Software Development Kit) – набір всіх компонентів від гугл, пакет засобів для розробки програм на Android – містить все необхідне для створення, компіляції та запакування Android-програм. Ці програми, в основному розробляються із застосуванням мови програмування Java. Все, що ми робимо на Android за допомогою Java, залежить від Android SDK – якщо ми створюємо додаток під певну версію, наприклад, для Android Nougat, то у нас повинні бути встановлені відповідні іструменти SDK. Це треба враховувати при розробці [10].

Android SDK містить в собі Android Debug Bridge, який є інструментом, що дозволяє підключитися до віртуального або реального пристрою з метою керування ним чи налагодження програми [6].

Компанія Google пропонує два інтегрованих середовищ для розробки нових програм:

- Android Studio, яке розроблене компанією Google. Це середовище базується на IntelliJ IDE;
- The Android Developer Tools (ADT) базується на Eclipse IDE. ADT – це набір компонентів (плагінів), які розширяють можливості Eclipse IDE для розробки під операційну систему Android.

Обидва середовища містять весь необхідний функціонал для створення, компіляції, налагодження та розгортання Android-програм. Вони також дозволяють розробнику створювати і запускати віртуальні пристрої Android для тестування.

В SDK є:

- плагіни для IDE;
- інструменти для сборки;
- інструменти для відлагодження;
- бібліотеки;
- емулятори.

Якщо відкрити SDK Manager в Android Studio, можна буде ясніше побачити, з чого складається Android SDK. На першій вкладці SDK Platform перераховані SDK кожної версії Android. Як показано на рисунку 2.5, Android 9.0 SDK (також відомий як Pie) містить:

- Android SDK Platform 28 (це API фреймворка);
- вихідний код для Android 28 (це реалізація API, яка не є обов'язковою);
- ще купа речей, наприклад, різні системні образи для емулятора Android.

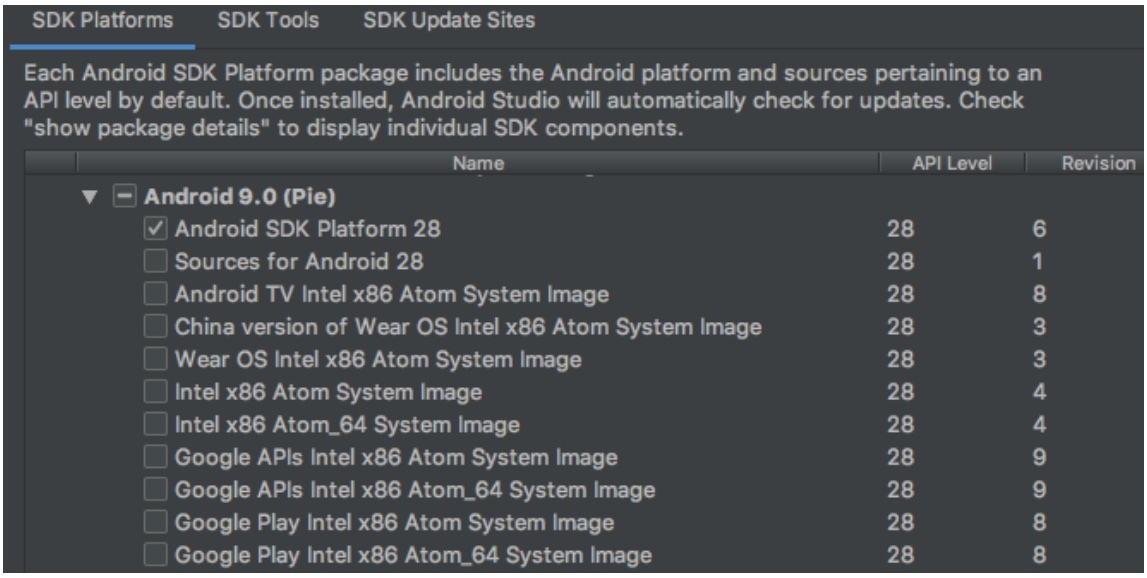


Рисунок 2.5 – Огляд SDK в Android Studio SDK Manager

Розглядаючи архітектуру Android додатків, можна побачити, що код розділений на два рівні: рівень даних містить DataManager і набір класів-помічників, рівень уявлення складається з класів Android SDK, таких як Activity, Fragment, ViewGroup тощо (Рисунок 2.6). Класи-помічники (третя колонка в діаграмі) мають дуже обмежені області відповідальності, і їх реалізують в послідовній манері.

Наприклад, більшість проектів мають класи для доступу до REST API, читання даних з бд або взаємодії з SDK від сторонніх виробників.

У різних додатків буде різний набір класів-помічників, але найбільш часто використовуваними будуть наступні:

- PreferencesHelper: працює з даними в SharedPreferences;
- DatabaseHelper: працює з SQLite;
- Сервіси Retrofit, що виконують звернення до REST API.

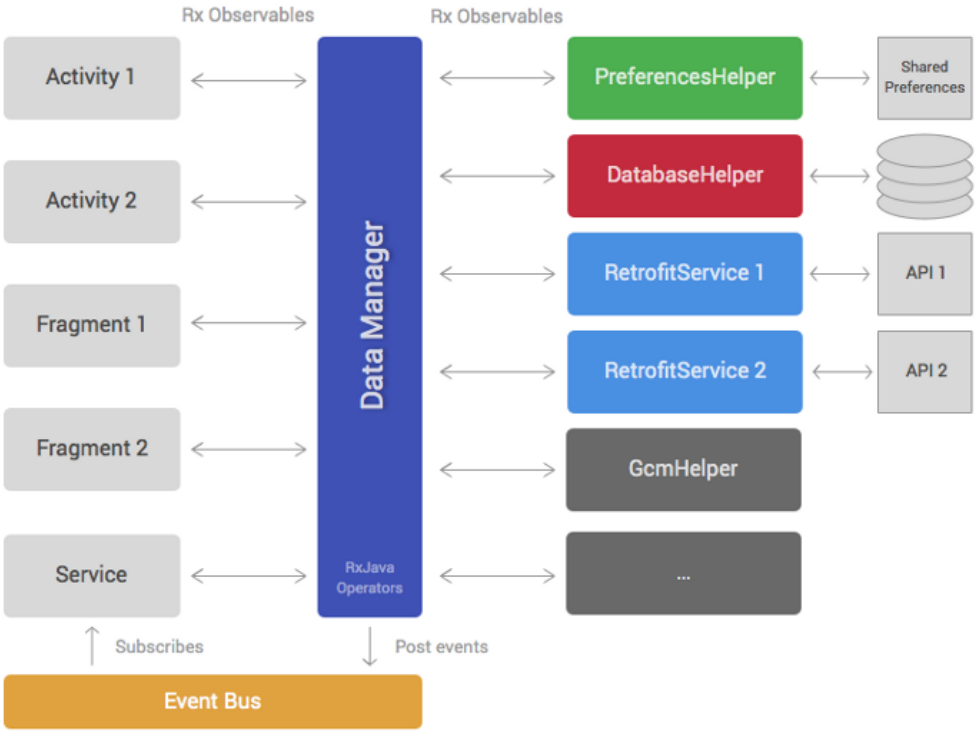


Рисунок 2.6 – Архітектура Android додатка

DataManager є центральною частиною архітектури. Він широко використовує оператори RxJava для того, щоб комбінувати, фільтрувати і трансформувати дані, отримані від помічників. Завдання DataManager полягає в тому, щоб звільнити activity і фрагменти від роботи по обробці даних – він буде виконувати всі потрібні трансформації всередині себе і віддавати назовні дані, готові до відображення.

2.5 Аналіз і постановка задачі

Android SDK та Xamarin в першу чергу класифікуються як «Набір для розробки ПЗ» та «Платформа для кросплатформної розробки» відповідно. Порівняємо обрані технології (Таблиця 2.2).

Таблиця 2.2 – порівняльна таблиця Xamarin і Android SDK

	Xamarin	Android SDK
Стек технологій	Net framework + нативні бібліотеки	Нативні бібліотеки
Використання коду	Спільне вкористання з іншою платформою до 96% з Xamarin.Forms	Єдиний код
UI/UX (User Interface/User Expierence)	Можливе повне налаштування UI для платформи	Платформо-залежний UI
Продуктивність	Близька до рідної	Відмінна
Можливості апаратних засобів	Високі. Xamarin використовує платформо-залежні API і підтримує зв'язок з нативними бібліотеками	Високі. Нативні інструменти мають повну підтримку для можливостей системи
IDE	Visual Studio	Android Studio

У разі створення додатків зі складним інтерфейсом з використанням Xamarin, кількість загального коду кардинально зменшується. Таким чином, кроссплатформенна розробка втрачає свою головну перевагу. Але це не робить додатки на Xamarin менш якісними, просто трохи збільшує витрати. Проте ні один з інших кросплатформених інструментів не може мати такого ж рівня продуктивності, які пропонує платформа Xamarin.

Метою дослідження є розробка двох нативних додатка, які мають наступний функціонал:

- можливість відкрити телефонну книгу користувача;
- можливість обрати контакт;
- відображення інформації про вибраний контакт.

Додатки мають бути розроблені з використанням таких технологій:

а) мови програмування: в примере показано

1) Java;

2) C#.

б) інструменти/платформи:

1) Android SDK;

2) Xamarin.

в) середовища розробки:

1) Android Studio;

2) Visual Studio.

Додатки повинні підтримувати версію Android 6.0.1 і вище.

Метою дослідження є порівняння таких показників як вага додатка, швидкодія та продуктивність.

3 РОЗРОБКА І АНАЛІЗ ДОДАТКІВ ДЛЯ ЕКСПЕРЕМЕНТУ

3.1 Розробка програми на Xamarin

Xamarin.Forms представляє платформу, яка націлена на створення кроссплатформенних додатків під Android. Для розробки кроссплатформенних додатків на Xamarin потрібне середовище розробки. Для Windows таким середовищем є Visual Studio. При установці Visual Studio 2019 програмою для установника обов'язково треба вибрати пункт «Розробка мобільних додатків на .NET». Для створення кроссплатформенних додатків для Xamarin Forms в Visual Studio 2019 призначений шаблон проекту, який називається Mobile App (Xamarin.Forms). При створенні проекту в опції Platform ми можемо вказати під які ОС буде створюватися проект.

XContactSample – головний проект бібліотеки, яка містить всю основну логіку додактку, XContactSample.Android – проект для Android (Рисунок 3.1).

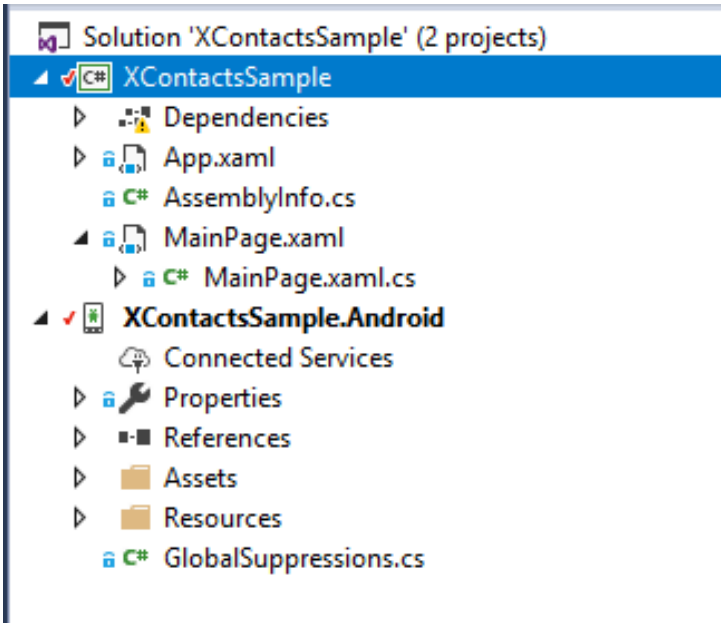


Рисунок 3.1 – Структура проекту

Виконання програми починається з файлів App.xaml і App.xaml.cs.

В якості головної сторінки встановлюється об'єкт класу MainPage, тобто єдина певна в проекті сторінка. Визначення цієї сторінки розбите на два файли. Файл MainPage.xaml представляє візуальний інтерфейс сторінки у вигляді коду XAML, який аналогічний HTML (Приклад 3.1).

```
      <?xml version="1.0" encoding="utf-8"?>
      <ContentPage               xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="XFEContactsSample.MainPage">
          <StackLayout>
              <Frame BackgroundColor="#2196F3" Padding="24" CornerRadius="0">
                  <Label   Text="Contacts   Sample"   HorizontalTextAlignment=
"Center" TextColor="White" FontSize="36" />
              </Frame>

              <Button Text="Pick Contact" Clicked="Button_Clicked" />
              <Label x:Name="resultContact" Margin="20"/>
          </StackLayout>
      </ContentPage>
```

Приклад 3.1 – Код файлу MainPage.xaml

Таким чином, інтерфейс сторінки досить простий – кнопка для вибору контакта і мітка з текстом.

Головний проект компілюється в бібліотеку dll, а XContactSample.Android містить посилання на нього. XContactSample.Android містить файл MainActivity.cs, з якого починається виконання проекту на Android. В файлі MainPage.xaml.cs проводиться запуск програми.

Щоб отримати доступ до функціоналу «Контакти», потрібна наступна установка для певної платформи: дозвіл ReadContacts, який повинен бути налаштований у проекті Android. Це можна додати в файлі AssemblyInfo.cs.

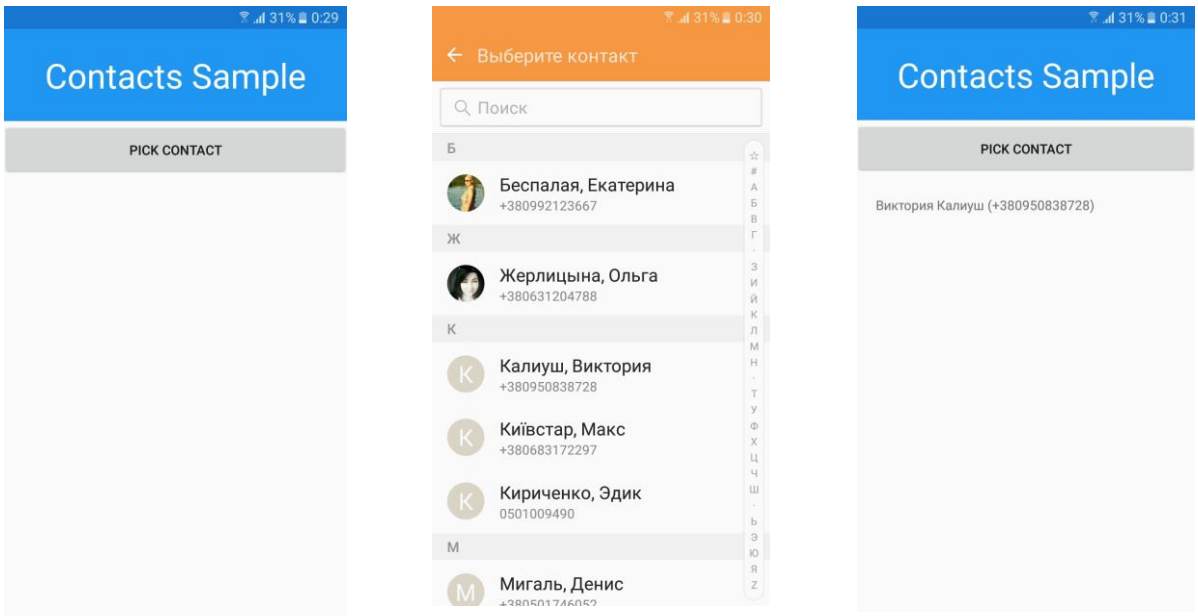
Викликавши Contacts.PickContactAsync() (Приклад 3.2), з’явиться діалогове вікно контактів, яке дозволить користувачеві отримувати інформацію про користувача.

```
async void Button_Clicked(System.Object sender, System.EventArgs e)
{
        var result = await Contacts.PickContactAsync();

        if (result != null)
            resultContact.Text = $"{result.Name} ({result.Numbers[0]
.PhoneNumber})";
}
```

Приклад 3.2 – Метод, що викликає вікно контактів і виводить дані на екран

На головній сторінці додаток містить кнопку для вибору контакта. Після натискання відкривається вікно з контактами і після натискання на будь-який з них ім’я і телефон виводяться на головний екран (Рисунок 3.2).



а) б) в)

Рисунок 3.2 – Скріни додатка: а) початкова головна сторінка; б) сторінка контактів; в) головна сторінка після вибору контакта

3.2 Розробка програми з використанням Android SDK

Android архітектура побудована таким чином, що будь-який додаток може використовувати вже реалізовані можливості іншої програми за умови,

що остання відкриє доступ на використання своєї функціональності. Таким чином, архітектура реалізує принцип багаторазового використання компонентів ОС і додатків. Коли система запускає компонент, то вона запускає процес програми, якій він належить та створює необхідні для нього екземпляри класів. Тому в ОС Android, на відміну від більшості інших систем, немає єдиної точки входу. Оскільки кожна програма запускається в окремому процесі та є обмеження на доступ до файлів, то програма не може безпосередньо активувати компонент іншої програми. Таким чином, для його активації необхідно відправити системі повідомлення про намір запустити певний компонент і система запустить його.

Необхідно звернути увагу, що при створенні додатка в полі Language в якості мови програмування за замовчуванням стоїть Kotlin. Його необхідно змінити на Java.

Проект Android може складатися з різних модулів. За замовчуванням, коли ми створюємо проект, створюється один модуль – app. Модуль має три папки (Рисунок 3.3):

- manifests (зберігає файл, який описує конфігурацію програми);
- java (зберігає файли коду на мові Java, які структуровані по окремих пакетах);
- res (містить використовувані в додатку ресурси).

Окремий елемент Gradle Scripts містить ряд скриптів, які використовуються при побудові програми. У всій цій структурі варто виділити файл MainActivity.java, який містить логіку програми та власне з нього починається виконання програми. І також виділимо файл activity_main.xml, який визначає графічний інтерфейс – по суті те, що побачить користувач на своєму смартфоні після завантаження програми.

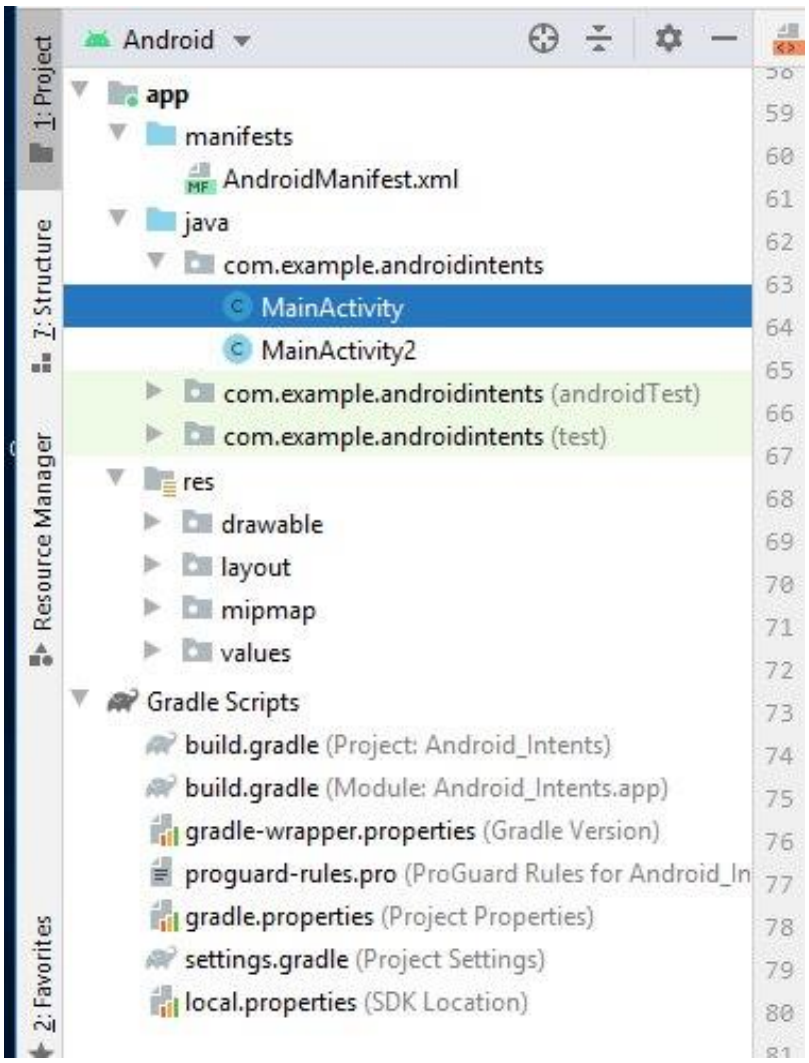


Рисунок 3.3 – Структура проекту

Для запуску і тестування програми ми можемо використовувати емулятори або реальні пристрої. Але в ідеалі краще тестувати на реальних пристроях. До того ж емулятори вимагають великих апаратних ресурсів, і не кожен комп'ютер може потягнути вимоги емуляторів.

Android Studio дозволяє працювати з візуальним інтерфейсом як в режимі коду, так і в графічному режимі. Так, за замовчуванням файл відкритий в графічному режимі, і ми наочно можемо побачити, як у нас приблизно буде виглядати вікно керування. І навіть встановити з панелі інструментів елементи управління, наприклад, кнопки або текстові поля (Рисунок 3.4).

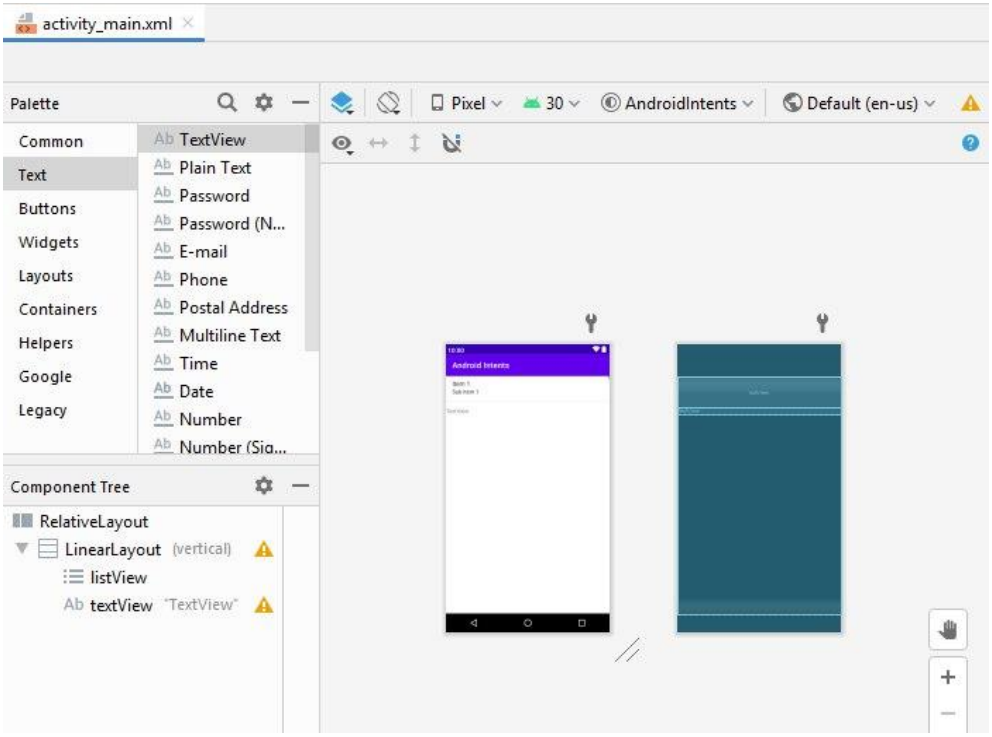


Рисунок 3.4 – Файл activity_main.xml, вкладка Design

Головний метод програми openContact() (Приклад 3.3) знаходится в файлі MainActivity.java і викликається при натисканні на Item.

```
    private void openContact() {
        Intent out = new Intent(Intent.ACTION_PICK, ContactsContract.
Contacts.CONTENT_URI);
        startActivityForResult(out, PICK_CONTACT);
    }
```

Приклад 3.3 – Метод openContact()

Для відправки використовуємо startActivityForResult. У startActivityForResult в якості параметрів ми передаємо Intent і requestCode. requestCode – необхідний для ідентифікації. Головна сторінка додатка зображена на Рисунку 3.5.

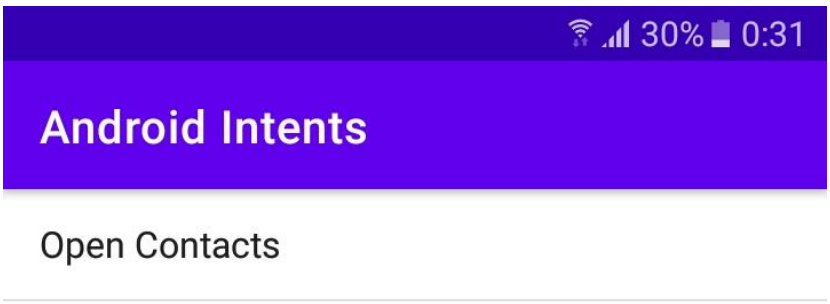


Рисунок 3.5 – Головна початкова сторінка додатку

Після натискання «Open Contacts» з’явиться список контактів, як і в додатку на C#. Вид додатку після вибору контакта виглядає як на рисунку 3.6:

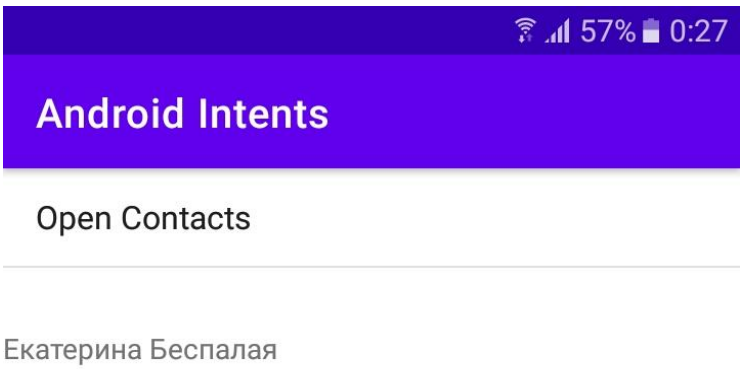


Рисунок 3.6 – Сторінка додатка після вибору контакта

3.3 Аналіз додатка на Java

Додаток на Java був розрблений в Android Studio, яка містить в собі зручний вбудований профілювальник, який дозволяє відстежити всі головні метрики продуктивності, такі як: пам'ять, процесор, використання мережі і споживання енергії (Рисунок 3.7).

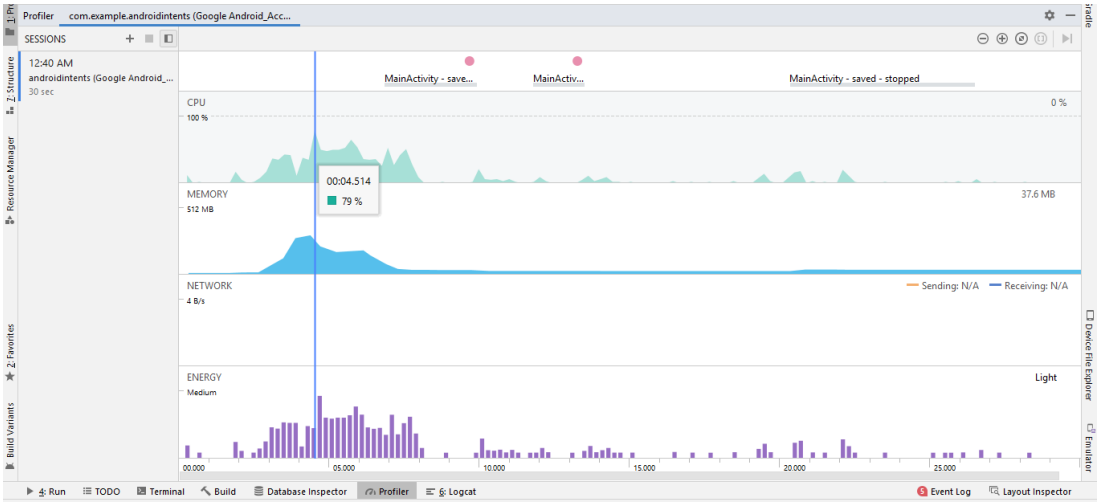


Рисунок 3.7 – Профілювання додатка на Java

Додаток займає 22,01 МБ з пам’яті пристрою. Варто зауважити, що цьому додатку не потрібен дозвіл на використання даних (Рисунок 3.8).

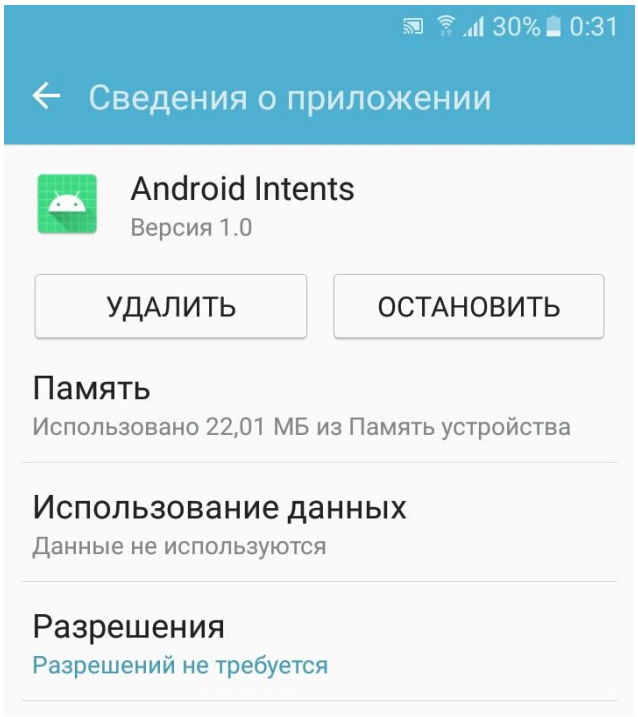


Рисунок 3.8 – Відомості про програму

3.4 Аналіз додатка на C#

Для додатків на Xamarin існує Xamarin Profiler – графічний інтерфейс для профілювання додатків Android та iOS на Windows. Але щоб скористуватися ним, потрібно бути передплатником Visual Studio Enterprise.

Також Android Studio включає в себе засіб Android Profiler. Його можна використовувати для вимірювання продуктивності додатків Xamarin Android, створеного за допомогою Visual Studio, - без необхідності в ліцензіях Visual Studio Enterprise. Однак на відміну від Xamarin Profiler засіб Android Profiler не інтегровано в Visual Studio і може використовуватися лише для реєстрації APK, який було створено та імпортовано в Android Profiler. Скористаємося ним. Для цього потрібно відкрити Android Studio, File->Profile or Debug APK і вибрати необхідний APK (Рисунок 3.9).

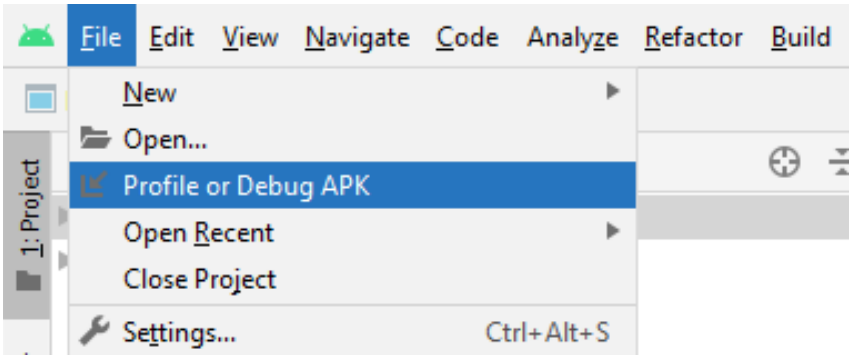


Рисунок 3.9 – Android Studio, можливість профілювати APK файл

Після завантаження APK файлу, Android Studio відображає інформацію про розмір додатка і всіх його частей (Рисунок 3.10).

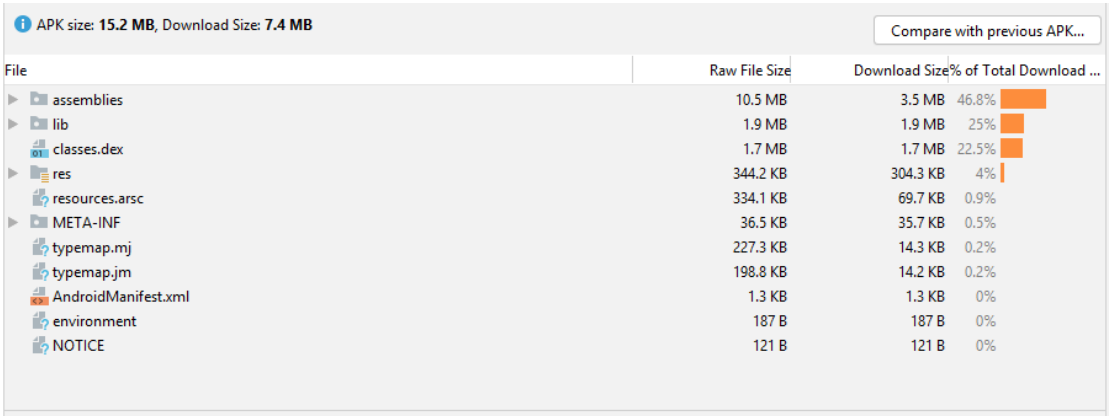


Рисунок 3.10 – Відображення інформації про розмір додатка

Для того щоб отримати всі необхідні для аналізу показники, необхідно натиснути Profile ‘назва_apk’ у верхньому лівому кутку студії. Android Studio проананалізує додаток, щоб відстежити всі головні метрики продуктивності (Рисунок 3.11).

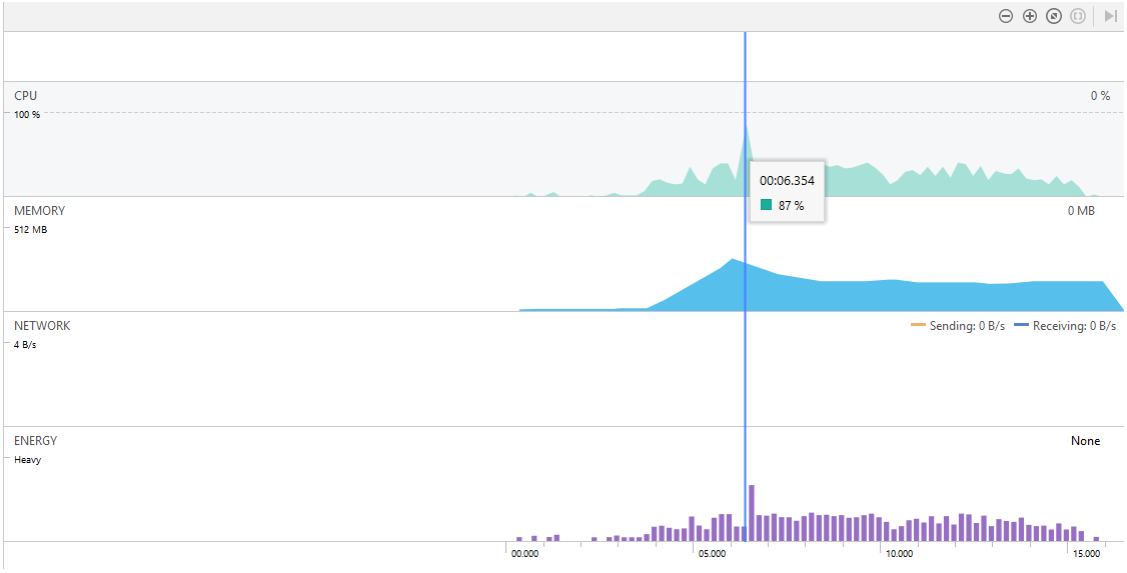


Рисунок 3.11 – Профілювання додатка на C#

Варто зазначити, що додатку на Xamarin необхідний дозвіл, щоб використовувати контакти (Рисунок 3.12), в той час як додаток на Java просто використовує вже реалізовані можливості.

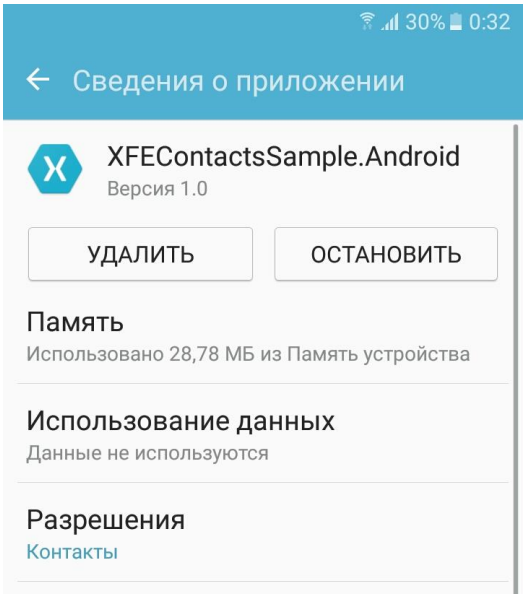


Рисунок 3.12 – Відомості про програму

3.5 Аналіз додатків

Додатки встановлювались і тестувались на смартфоні з версією Android 6.0.1 (Рисунок 3.13).

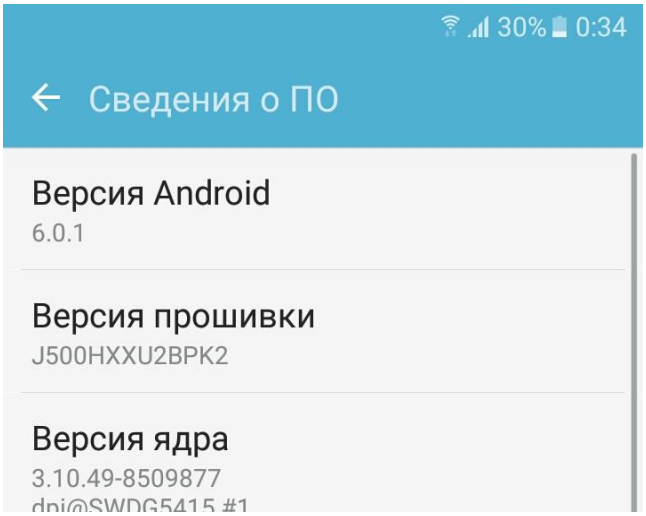


Рисунок 3.13 – Відомості про програмне забезпечення

Занесемо в таблицю 3.1 всі перераховані у пунктах 3.3 і 3.4 показники, а саме вага додатка (Рисунки 3.8 і 3.12), а також пам'ять, процесор і споживання енергії (Рисунки 3.7 і 3.11). Використання мережі залишилося стале в обох випадках. Energy Profiler відображає приблизне споживання енергії вашим додатком.

Таблиця 3.1 – Порівняльна таблиця результатів профілювання додатків

	Додаток на C#	Додаток на Java
Вага (МБ)	28,78	22,01
CPU (%)	87	79
Memory (МБ)	304,7	273,2
Energy	00:04,513	00:01,434

Аналізуючі всі показники, можна зробити висновок, що використання нативних технологій, а саме мова програмування Java і Android SDK, є найефективнішими у даному дослідженні.

ВИСНОВКИ

У дослідженні розглянуті види та проведений аналіз типів мобільних додатків і самої операційної системи Android. Крім того, розглянуто і проаналізовано технології для розробки під Android. Також у роботі розглянуті основні компоненти архітектури мобільних додатків для Android, які є блоками для побудови програми. За допомогою середовищ розробки Visual Studio і Android Studio було розроблено два додатки, які можна встановити на мобільний пристрій та переглядати з нього інформацію про контакти користувача. Додатки були розроблені на мовах програмування Java і C#, використані Android SDK і Xamain відповідно.

Java і C# є популярними мовами програмування, тому важко сказати яка з них краща за такими якісними показникам, як наявність повної документації, технічна підтримка і наявність фахівців. Обидві ці мови є сі-подібними, тому мають схожий зрозумілий синтаксис, зручний у розробці і налагодженні.

Розглядаючи такі технічні характеристики як безпроблемна взаємодія програми з мобільною ОС, варто зазначити, що платформи, які використовувалися у розробці, мають повний доступ до основних нативних функцій. Як особливість варто зазначити, що під час використання додатку на C#, программа запитувала доступ на використання контактів, в той час як додаток на Java дозволу не потребує.

Для того, щоб дати однозначну відповідь на питання яка з мов має вищу швидкість роботи і відгуку інтерфейсу, виконуються багато тестів в «реальних умовах», де тестується швидкість великих обчислень або активних запитів до бази даних. В даному дослідженні експерементально два додатки порівнювалися за такими показниками як вага додатків та їх продуктивність. При аналізі нативний додаток, розроблений на Java, показав кращі результати. Проте не можна зробити однозначний вибір на користь однієї

технології, адже при створенні того чи іншого мобільного додатку необхідно чітко визначати мету, що і буде керувати кінцевим вибором технологій.

Як висновок можна зазначити, що для найбільш ефективних результатів, при умові наявності необхідних навичок, бюджету і особливо часу, доцільно використовувати повністю нативну платформу. Якщо ж необхідний додаток для декількох ОС, а його функціонал не вимагає на кожній операційній системі дуже високу продуктивність, то логічніше було б використовувати кросплатформенний фреймворк. Він би забезпечив максимальне покриття за которотший час, ніж нативні технології для кожної операційної системи окремо.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Розробка веб-додатків, мобільних додатків та порталів: [Електронний ресурс]. – Режим доступу: http://ittel.com.ua/informacijni-texnologiyi/rozrobka-mobilnih-dodatkiv/

2. Techbeacon: [Електронний ресурс]. – Режим доступу: https://techbeacon.com/html5-mobileapp-or-native-it-depends

3. HTML5 vs Native Android App: [Електронний ресурс]. – Режим доступу: https://www.androidauthority.com/html-5-vs-native-android-app-607214/

4. Рік Роджерс, Джон Ломбардо Android. Разработка приложений – М.: ЕКОМ Паблішерс, 2010 – 400 с.

5. Голощапов А. Л. Google Android: программирование для мобильных устройств – СПб.: БХВ-Петербург, 2011 – 448 с.

6. Янчук К. В. Огляд платформ для створення мобільних додатків [Текст] : материалы VIII міжнародної наук.-техн. конф., 26-27 листопада 2020 р. : тези доп / редкол. : К. В. Янчук, В.М. Федорченко – Черкаси – Харків – Баку – Бельсько-Баяла : 2020. – 64 с.

7. What is Xamarin? 09.16.2019. [Електронний ресурс]. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin

8. Kotlin, [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Kotlin

9. Полное руководство по языку программирования C# 8.0 и платформе .NET Core 3. URL: https://metanit.com/sharp/tutorial/

10. Get the Android SDK [Electronic Resource] // Android Developers/ – URL: http://developer.android.com/sdk/index.html. SDK.

11. Ковальчук В.Б, Види мобільних додатків та їх переваги і недоліки [Електронний ресурс]. – Електронні дані. Режим доступу:

http://eprints.zu.edu.ua/25802/1/ Ковальчук%20В.%20Б.pdf

12. Інформаційне агенство ЛІГ АБізнесІнформ [Електронний ресурс] : Топ-10 популярніх мобільних додатків серед українців. Червневий рейтинг від 18.07.2019, 12:32 – Електронні дані. Режим доступу: https://uanews.liga.net/society/news/top-10-populyarnih-mobilnih-dodatkiv-sered-ukraintsiv

13. Gerber A. Learn Android Studio: Build Android Apps Quickly and Effictively / A. Gerber? C. Craig. – New York: Apress, 2015. – 468 c.

14. Chowdhury K. Mastering Visual Studio 2017: Build windows apps using WPF and UWP, accelerate cloud development with Azure, explore NuGet, and more / Kunal Chowdhury.,2017.

15. Шилд Гербер. Java 8 полное руководство.: Вильямс, 2016.720с.

16. Гріффітс Дон, Гріффітс Девін Head First. Програмування для Android. СПб.: Питер, 2016.704с.

17. Винокуров А.С., Баженов Р.И. Разработка мобильного приложения информационного сайта для студентов // Современные научные иследования и иновации. 2015. №7-2(51).С.54-62

18. Amirgaliyev E.N., Kalizhanova A.U., Kozbakova A.KH. Development of applications to mobile devices in Android platform // Труды Международного симпозиума «Надежность и качество». 2015.№1.С.240-242.

19. Thanh H.D. Nguyen Automated detection of performance regressions using statistical process control techniques / T.H. Nguyen, B. Adams, Z.M. Jiang, A. E. Hassan, M. Nasser, P. Flora // Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering – Boston, 2012. – C. 299-310.

20. Стягар А. І. Продуктивність веб-додатків / Матеріали V науковотехнічної конференції ТНТУ ім.І.Пулюя, 2018.–82 с.