Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти першій (бакалаврський)

Інформаційний мобільний застосунок
з використанням хмарних сервісів

(тема)

Виконала:
здобувачка 4 року навчання,
групи КІУКІ-21-4
Еліна ЛІ
(власне ім'я, прізвище)

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

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

Освітня програма
Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник: ст.викл. Андрій БУГРІЙ
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти першій (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

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

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“ ” _____ 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Лі Еліні Євгенівні
(прізвище, ім’я, по батькові)

1. Тема роботи Інформаційний мобільний застосунок з використанням хмарних сервісів

затверджена наказом по університету від “ 26 ” травня 2025 р. № 424ст

2. Термін подання студентом роботи до екзаменаційної комісії 17 червня 2025 р.

3. Вхідні дані до роботи

1. Технології розробки хмарних систем (GoogleMaps, AWS Storage)
2. Типи мобільних застосувань (Native, Hybrid)
3. Мови програмування (Java, HTML) та середовища розробки (Eclipse)
4. Формати передачі даних (XML, JSON)

4. Перелік питань, що потрібно опрацювати в роботі

1 Аналіз предметної області
2 Розробка мобільного застосування
3 Імплементація проєкту
4 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)

Слайд-презентація – 12 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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області	26.05.25-30.05.25	
2	Розробка, вибір моделей, методів, засобів	31.05.25-01.06.25	
3	Реалізація алгоритмів	02.06.25-04.06.25	
4	Розробка структури програмних засобів	05.06.25-07.06.25	
5	Розробка програмних модулів	08.06.25-09.06.25	
6	Оформлення матеріалів кваліфікаційної роботи	10.06.25-11.06.25	
7	Подання кваліфікаційної роботи керівникові та її попередній захист	12.06.25-13.06.25	
8	Подання кваліфікаційної роботи на рецензування	14.06.25-15.06.25	

Дата видачі завдання 26 травня 2025 р.

Здобувачка _____
(підпис)

Керівник роботи _____ ст. викл. Андрій БУГРІЙ
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 52с., 13 рис., 4 табл., 1 дод., 17 джерел.

МОБІЛЬНІ ДОДАТКИ, РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ГНУЧКІ МЕТОДОЛОГІЇ, ХМАРНІ СЕРВІСИ, ОПТИМІЗАЦІЯ ПРОЦЕСІВ

Робота присвячена дослідженню сучасних підходів до розробки мобільних додатків та оптимізації процесів їх створення. Запропоновано методологію розробки, що дозволяє підвищити ефективність створення мобільних застосунків через використання адаптивних підходів, зокрема гнучких методологій та хмарних сервісів. Досліджено ключові проблеми, з якими стикаються розробники мобільного програмного забезпечення, зокрема короткі цикли розробки, швидкі технологічні зміни, складність дизайну інтерфейсу користувача та питання безпеки. Представлено розроблений мобільний додаток, що реалізує запропоновані принципи, а також проведено його тестування. Отримані результати підтвердили ефективність запропонованого підходу у контексті швидкості розробки, продуктивності та адаптивності програмних рішень.

ABSTRACT

Bachelor’s thesis: 52 pages, 13 figures, 4 tables, 1 appendice, 17 sources.

MOBILE APPLICATIONS, SOFTWARE DEVELOPMENT, FLEXIBLE METHODOLOGIES, CLOUD SERVICES, PROCESS OPTIMIZATION

The work is devoted to the study of modern approaches to the development of mobile applications and the optimization of their creation processes. A development methodology is proposed that allows increasing the efficiency of creating mobile applications through the use of adaptive approaches, in particular, flexible methodologies and cloud services. The key problems faced by mobile software developers are investigated, in particular, short development cycles, rapid technological changes, the complexity of user interface design and security issues. A developed mobile application that implements the proposed principles is presented, and its testing was also conducted. The results obtained confirmed the effectiveness of the proposed approach in the context of development speed, productivity and adaptability of software solutions.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ОС – операційна система

ПЗ – програмне забезпечення

ADT – Android Development Tools

API – Application Programming Interfase

AVD – Android Virtual Device

AWS – Amazon Web Services

JSON – JavaScript Object Notation

IDE – Integrated Development Environment

OS – operation system

OTP – One Time Password

REST – Representational State Transfer

SDK – Software Development Kit

UI – user interface

UML – Unified Modeling Language

UX – user experience

XML – Extensible Markup Language

ВСТУП

Зараз зростає інтерес до розробки мобільних додатків. Однак розробники часто ігнорують або принаймні суттєво адаптують існуючі процеси розробки програмного забезпечення відповідно до своїх цілей, враховуючи існуючі специфічні обмеження. Такі коригування можуть ввести варіації та нові тенденції в існуючі процеси, про які в багатьох випадках наукове співтовариство не повідомляє через відсутність офіційної документації, що виправдовує подальші дослідження. У цій роботі представлено дослідження та характеристику сучасних процесів розробки мобільних додатків. Розглядається набір реальних прикладів для дослідження поточного процесу розробки мобільних додатків, які використовуються компаніями-розробниками програмного забезпечення, а також незалежними розробниками. Результатом цього дослідження є ідентифікація процесів розробки мобільного програмного забезпечення, а саме гнучких підходів, а також недоліків у поточних методологіях.

Робота присвячена дослідженню сучасних підходів до розробки мобільних додатків та оптимізації процесів їх створення. Запропоновано методологію розробки, що дозволяє підвищити ефективність створення мобільних застосунків через використання адаптивних підходів, зокрема гнучких методологій та хмарних сервісів. Досліджено ключові проблеми, з якими стикаються розробники мобільного програмного забезпечення, зокрема короткі цикли розробки, швидкі технологічні зміни, складність дизайну інтерфейсу користувача та питання безпеки. Представлено розроблений мобільний додаток, що реалізує запропоновані принципи, а також проведено його тестування. Отримані результати підтвердили ефективність запропонованого підходу у контексті швидкості розробки, продуктивності та адаптивності програмних рішень.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз ринку розробки мобільних додатків

Зараз мільйони програм доступні в різних онлайн-магазинах для користувачів смартфонів. Найуспішніші мобільні програми було завантажено понад мільярд разів [1], і щодня на ринок мобільних пристроїв з'являються нові програми, що робить його надзвичайно привабливим як для компаній, так і для незалежних розробників, щоб інвестувати свій час і гроші. Такий попит часто змушував розробників програмного забезпечення для мобільних пристроїв (ПЗ) адаптувати усталені методології розробки ПЗ або подавати нові пропозиції, які відповідають обмеженням, пов'язаним із розробкою ПЗ для мобільних пристроїв.

Особливості розробки програмного забезпечення для мобільних пристроїв різноманітні, але, безсумнівно, включають короткі та часті цикли розробки, часті технологічні зміни (платформи, операційні системи, датчики тощо), обмежену документацію, особливі вимоги та ресурси групи розробників і клієнтів.

Крім того, всі ці можливі фактори схильні до постійних інновацій [2]. Етапи розробки мобільних додатків можуть відрізнятися залежно від проекту чи встановлених технологій.

Мобільні програми є результатом кількох дій, які виконуються, наприклад, розподіл ролей у робочій групі, визначення цілей і заходів, впровадження набору передових практик командної роботи та співпраці, встановлення розкладу діяльності, управління ризиками тощо. Зрештою, інструкції повинні бути адаптовані відповідно до наявних ресурсів і вимог замовника [3] [4]. Крім того, розробка мобільних додатків вимагає постійних удосконалень та адаптацій для задоволення нових технологічних потреб і змін, що створює значні проблеми, такі як: дизайн інтерфейсів користувача

для різних розмірів екранів мобільних пристроїв, досвід користувача, пов’язаний із можливостями мобільних пристроїв, методи взаємодії з користувачем, які забезпечують мобільні платформи, архітектури тощо [5].

Ця потреба у швидкості адаптації є однією з причин обмеженої формальної та наукової документації у сфері розробки мобільних додатків. Наразі бракує наукової документації, яка б відображала процеси розробки, орієнтовані на мобільні додатки та їх діяльність [6].

Процеси розробки мобільних додатків на рівні програмного забезпечення ще не повністю встановлені в сучасній галузі. Мета цього дослідження полягає в тому, щоб краще зрозуміти поточні прийняті методології, щоб визначити, які та як процеси або методології, що мають відношення до розробки мобільних додатків, здійснюються в двох контекстах: академічному та промисловому, враховуючи останнє з тематичними дослідженнями малих, середніх і великих компаній. Це дослідження також сприяє кращому розумінню процесів розробки мобільних додатків, вивчаючи реальні проблеми, з якими стикаються, діяльність, що виконується, і враховуючи характеристики команди розробників з використанням реальних сценаріїв.

1.2 Проблеми процесів розробки мобільних застосувань

У цьому розділі ми представляємо роботи, в яких розглядалися проблеми, пов’язані з дослідженням сучасних процесів розробки мобільних додатків.

У роботі [6] досліджують проблеми, з якими стикаються під час процесів розробки мобільних додатків, шляхом опитування спільноти дослідників і розробників мобільних пристроїв. Деякі компанії зосереджуються на одній платформі, що зменшує охоплення їхніх додатків; кілька учасників висловили занепокоєння з приводу відсутності поінформованих і досвідчених ресурсів для розробки мобільних додатків,

оцінки зусиль, необхідних на початку процесу розробки, а саме гнучких підходів, а також недоліків у поточних методологіях, що застосовуються в промисловості та академічній спільноти, а саме відсутність циклу та зауваження, що команда розробників витрачає багато часу на розуміння та аналіз вимог клієнтів, збільшуючи навантаження на розробку, інтеграцію та тестування додатків [6]. З іншої точки зору, у [8] в ході опитування для розробників мобільних додатків дійшли висновку, що більшість мобільних додатків є відносно невеликими, з одним або двома розробниками, відповідальними за проектування та реалізацію всієї програми, рідко з використанням формальних процесів розробки та з дуже невеликими доробками у подальшому за невеликою кількістю показників.

Крім того, кілька досліджень сходяться на тому, що більшість розробників мобільних додатків використовують гнучкі підходи або майже спеціальні підходи, враховуючи той факт, що, оскільки мобільні телефони розвиваються дуже швидко, потрібен короткий час розробки та планування, а цикли розробки можуть бути лімітованими [6] [7].

Розробка програмного забезпечення мобільних додатків робить сильний акцент на дизайні інтерфейсу користувача (UI). У роботі [6] встановлено, що розробникам мобільних пристроїв важко якнайкраще використовувати обмежений простір екрана, а дизайн інтерфейсу користувача набуває більшого значення, ніж будь-коли. Цей факт пов’язаний із користувальницьким досвідом (UX), де розробники вказують, що менший дисплей, макети екрану та різні стилі взаємодії з користувачем мають великий вплив на розробку мобільних додатків, що зрештою призводить до низького сприйняття та використання програми [6]. Незважаючи на те, що розробники можуть контролювати більшість аспектів UX, мобільні додатки часто мають загальні елементи інтерфейсу користувача з іншими додатками і тому повинні дотримуватися зовнішніх розроблених інструкцій щодо інтерфейсу користувача, багато з яких реалізовано в наборах для розробки програмного забезпечення, які є частиною платформи [8]. Крім того,

розробники мобільних додатків повинні передбачити цільові пристрої для розробки інтерфейсу користувача. Зокрема, якщо націлено на одну платформу, розробники можуть вирішити створити єдину програму для всіх платформ із ризиком деяких функціональних невідповідностей або натомість розглянути можливість створення кількох версій, націлених на кожну апаратну платформу [7].

Крім того, в [8] зроблено висновок, що незважаючи на існування величезної кількості мобільних додатків сьогодні, процеси розробки мобільного ПЗ не повністю адаптовані до проектів розробки мобільних додатків, і їх складно стежити, оскільки характеристики мобільних додатків у їхньому робочому та реальному середовищах породжують новий набір тем дослідження з різними підходами.

Сучасні підходи представляють загальні опитування, які зазвичай не фокусуються на розробці конкретної програми, що призводить до відносно абстрактних інтерпретацій.

Щоб описати сценарій і визначити процеси розробки мобільного програмного забезпечення на практиці, ми використовували дослідження [9], з метою ідентифікації та характеристики процесів і методологій розробки мобільних додатків, які зараз застосовуються в академічному середовищі та промисловості. Розглянемо деякі дослідницьких питання, і можливі відповіді.

Для збору даних використовувалися методи [11], такі як інтерв’ю та анкетування, які дозволяють прямий контакт з джерелами інформації. Для збору інформації для кожного окремого дослідження було створено анкету з відкритими та закритими запитаннями, які намагалися відповісти на підпитання дослідження.

Крім того, для підтвердження інформації з кожним керівником групи розробки були проведені особисті інтерв’ю. Зібрану інформацію впорядковано відповідно до підпитань. Застосовувана стратегія базується на наданні набору можливих відповідей на кожне з досліджуваних питань, як представлено в таблиці 1.1.

Таблиця 1.1 – Крітерії порівняння засобів розробки мобільних застосувань

Предмет оцінювання	Варіанти відповідей	Коментарі
Мобільні застосування	Native, Web, Hybrid App	-
Платформа розробки	Android, IOS, Windows,…	-
Розмір компанії: кількість робітників	Фріланс (F), вевелика (S), середя (M), велика (L).	F: 1-5, S: 5-40, M: 41-150, L: > 150
Тип процесу розробки	Agile, Not Agile	-
Методологія розробки	Ім'я методології	Якщо не вказано, класифікується як AD-HOC
Які види діяльності	Список активностей	-
Досвід розробників (кількість років)	Немає досвіду (NoE), новачок (New), фахівець (I), Експерт (E)	NoE: 1-2, New:2-5, I: 5-10, E: >10
Строк розробки (місяців)	1-3, 3-6, 6-12, >12	-
Ступінь відповідальності	Low (L), Medium (M), High (H)	L: 0-2, M: 3-6, H:7-10
Проблеми даного типу додатку	Список проблем	Апріорні та виникаючи під час розробки

В таблицях 1.2 та 1.3 проаналізовані розробки від академічного середовища та від промисловості. Найбільшу кількість розробок включають програми, які розроблені лише для Android. Решта п’ять розробили мобільні додатки для Android та iOS. Нарешті, у прикладі немає представгиків, які б включали лише розробку для iOS.

Таблиця 1.2 – Платформи розробки мобільних додатків

	Університет/ фірма	Додаток	Платформа		Пристрій		
			Andr.	iOS	Phone	Tab	Note
Академ.	MIST	MISTINF	+	+	+	+	
	MIST	GetDate	+		+	+	
	USB	BIOCIT	+	+	+		
Фірми	NekDev	GizModo	+	+	+	+	
	GojiLabs	WWF	+		+		+
	Emizentech	Chomp	+	+	+		
	Techhead	Hightech	+	+	+	+	
	INOXOFT	Shelf	+		+		

Таблиця 1.3 – Тип мобільних додатків

	Університет/ фірма	Додаток	Платформа		
			Native	Web	Hybrid
Академ.	MIST	MISTIINF	+		
	MIST	GetDate			+
	USB	BIOCIT	+		
Фірми	NekDev	GizModo	+		
	GojiLabs	WWF	+		
	Emizentech	Chomp			+
	Techhead	Hightech	+		
	INOXOFT	Shelf		+	

Таблиця 1.3 показує, що 5 представників (2 з академічної спільноти, 3 з промисловості) розробили власні мобільні програми, один мобільний веб-додаток, а три розробили гібридні мобільні додатки.

Згідно зі введеною шкалою, 20% є мікропідприємством (фріланс), яке є лише у середньому із 7 працівниками, 40% є невеликими компаніями, які входять до цієї цифри, GojiLabs: 12 співробітників і Emizentech: 20

співробітників. Решта 40% – великі компанії, серед яких NekDev: понад 2000 співробітників, и Emizentech – 248.

Аналіз типів процесу розробки поділяється відповідно до двох типів: гнучкий і не гнучкий, з кожною зі своїх методологій або процесів (таблиця 1.4). Зауважте, що компанії зазвичай обирають гнучкі методології.

Таблиця 1.4 – Методології розробки мобільних додатків

	Університет/ фірма	Додаток	Тип		Методологія
			Agile	No agile	
Академ.	MIST	MISTIINF		+	WATERFALL
	MIST	GetDate		+	ADHOC
	USB	BIOCIT	+		ADHOC
Фірми	NekDev	GizModo	+		SCRUM
	GojiLabs	WWF	+		SCRUM
	Emizentech	Chomp	+		XP
	Techhead	Hightech	+		SCRUM
	INOXOFT	Shelf	+		ADHOC

У цьому підрозділі проведено аналіз методологій або процесів розробки мобільних додатків, які застосовують промисловість і наукові кола, незважаючи на тисячі мобільних додатків на ринку. Враховуючи більший інтерес до швидкої розробки додатків, більшість гравців ринку діляться своїм досвідом розробки з науковцями. Таким чином, головна мета цього дослідження полягала в тому, щоб сприяти кращому розумінню процесів розробки програмного забезпечення для мобільних пристроїв або методологій, які застосовують промисловість і наукові кола для розробки мобільних додатків. Розробники мобільних програм мають кілька процесів, спеціально визначених для мобільного домену. Однак серед ідентифікованих процесів SCRUM стає еталонним, оскільки він здебільшого використовується в промисловості. Визначені процеси базуються на гнучких і негнучких

методах і методологіях, які навряд чи інтегрують конкретні дії мобільного додатку, не маючи можливості адекватно використовувати функціональні можливості та характеристики мобільних пристроїв. Крім того, підтверджено, що прихованою вадою в процесі розробки мобільних програм є відсутність поінформованих і досвідчених ресурсів для розробки мобільних програм.

Ці результати показують необхідність поділитися та запропонувати конкретні методи розробки мобільних додатків, які об'єднують функції та обмеження мобільних пристроїв, а також методи зручності використання для покращення взаємодії з користувачем під час використання мобільного додатка. Подальші дослідження вирішуватимуть ці потреби та включатимуть ширший практичний досвід, як з точки зору одиниць аналізу, так і дослідницьких питань.

1.3 Особливості програмування для операційної системи Android

Android — це комплексна платформа з відкритим кодом, розроблена для мобільних пристроїв. Його підтримує Google і належить Open Handset Alliance. Метою альянсу є прискорення інновацій у мобільних обчисленнях і пропонування споживачам багатшого, дешевшого та кращого мобільного досвіду. Android – засіб для цього. Android — це операційна система на базі Linux, яка в основному використовується для роботи мобільних пристроїв, таких як смартфони та планшетні комп'ютери. Його зручність не обмежується мобільними пристроями. Завдяки своїм відкритим і настроюваним функціям він використовується в широкому спектрі електронних пристроїв, таких як ноутбуки, смарт-телевізори, камери, навушники, наручні годинники, ігрові консолі, автомобільні програвачі компакт-дисків і DVD, домашня автоматизація та багато іншого [10]. ОС Android не залежить від апаратного забезпечення та працює на пристроях від різних виробників, на відміну від інших власних операційних систем, таких

як iOS (продукти Apple Inc.), Blackberry OS (Blackberry), ОС S40 (Nokia), ОС Windows (Windows Phone) тощо, які ліцензовані та контролюються певними компаніями. Станом на 2024 рік Android домінує на ринку смартфонів, на нього припадає 70,93% світових продажів смартфонів.

Android — це повноцінна операційна система та повний стек програмного забезпечення для мобільних пристроїв. Android API — це багатий набір системних служб, загорнутих у файли інтуїтивно зрозумілих класів, які забезпечують легкий доступ до таких функцій, як місцезнаходження, Інтернет, телефонія, Wi-Fi, медіа, камера тощо. Усі інструменти, фреймворки та програмне забезпечення, необхідні для розробки мобільних додатків, доступні безкоштовно.

Додаток Android – це програма для мобільних пристроїв, розроблена для використання на пристроях на платформі Android від Google. Програма для Android може бути написана кількома різними мовами програмування. В роботі буде використовуватися Java. Незважаючи на те, що ця програма закодована на Java, вона глибоко покладається на величезний набір рідних бібліотек, написаних на C++. Розробники програм можуть легко скористатися величезним набором системних служб, інструментів і бібліотек для використання в програмах, якщо потрібно.

Android створено на базі Linux. Linux є основою для всіх пакетів програм в Android. Є багато причин для вибору Linux як основи стеку Android, таких як портативність, безпека, мережеві можливості, чудове керування пам’яттю та процесами, а також підтримка спільних бібліотек. Структура операційної системи Android наведена на рисунку 1.1. Архітектурно в неї можна виділити наступні рівні: застосувань, фреймворк, бібліотеки та ядро.

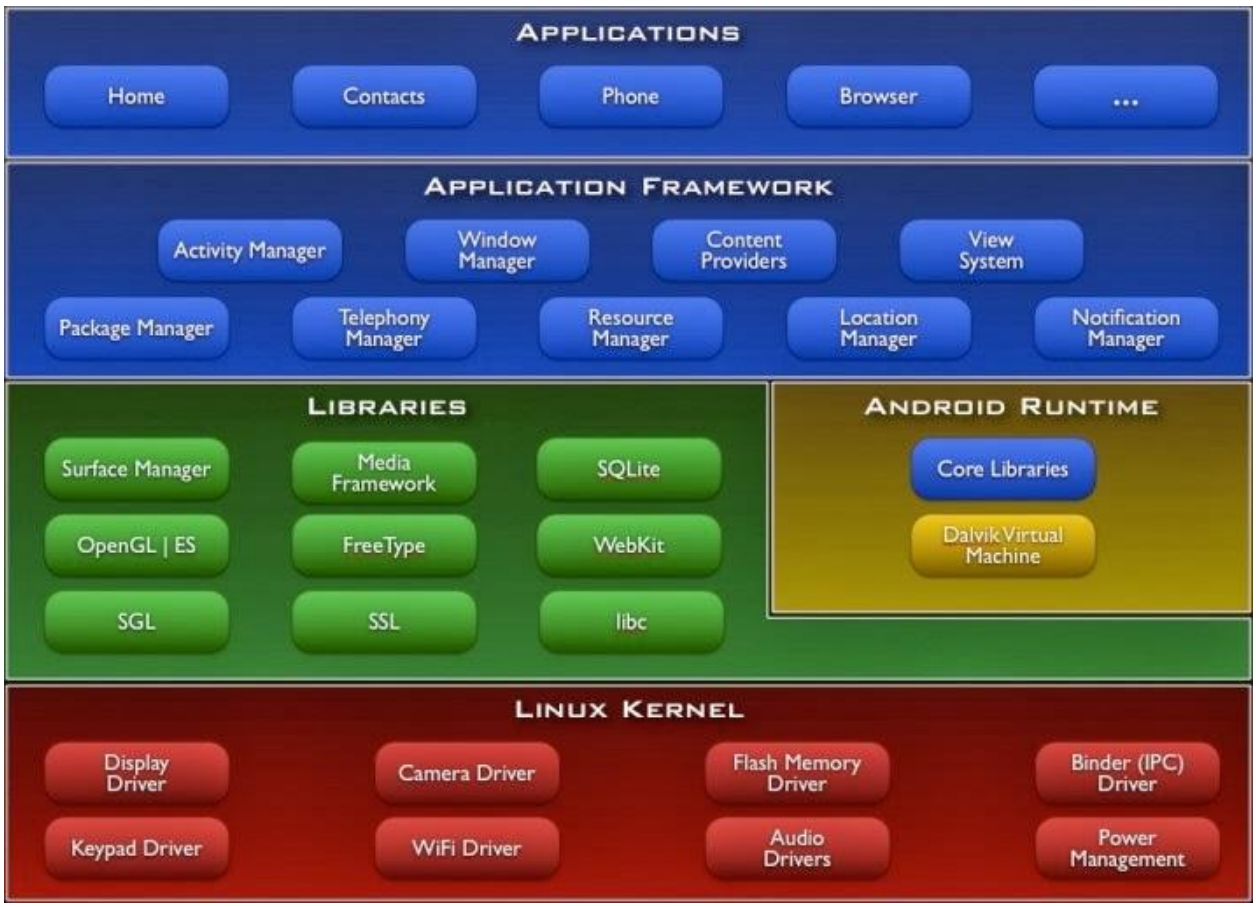

Рисунок 1.1 – Архітектура операційної системи Android

Основні будівельні блоки – це компоненти, які розробник використовує для створення програми Android. Ці компоненти допомагають розбити роботу на невеликі концептуальні одиниці, щоб розробник програми міг працювати над ними незалежно та об’єднати їх у повний пакет.

Є п’ять компонентів програми, які є важливими для створення програми Android. Ці компоненти програми дуже важливі для детального розуміння розробниками програм, оскільки всі основні дії (перемикання між екранами/програмами, маніпуляції з базою даних, ініціювання подій, отримання сповіщень тощо), які виконує програма, обробляються ними.

Діяльність (Activity) – це компонент програми, який надає екран, з яким користувачі можуть взаємодіяти, щоб виконувати певні завдання, наприклад набирати номер телефону, фотографувати, надсилати електронний лист, переглядати карту та багато іншого. Одна програма може мати кілька дій, які користувач гортає вперед і назад на пристрої [10]. Запуск активності є

важливою частиною процесу розробки додатків для Android. Клас Activity надається платформою Android, яка надає широкий спектр можливостей, як-от відображення інтерфейсу користувача, створення нового процесу Linux і виділення пам'яті для об'єктів інтерфейсу користувача. Як правило, програма Android має одну основну дію, яку користувач бачить, коли програму запускає, і користувач може перейти до інших дій за потреби (рисунок 1.2).

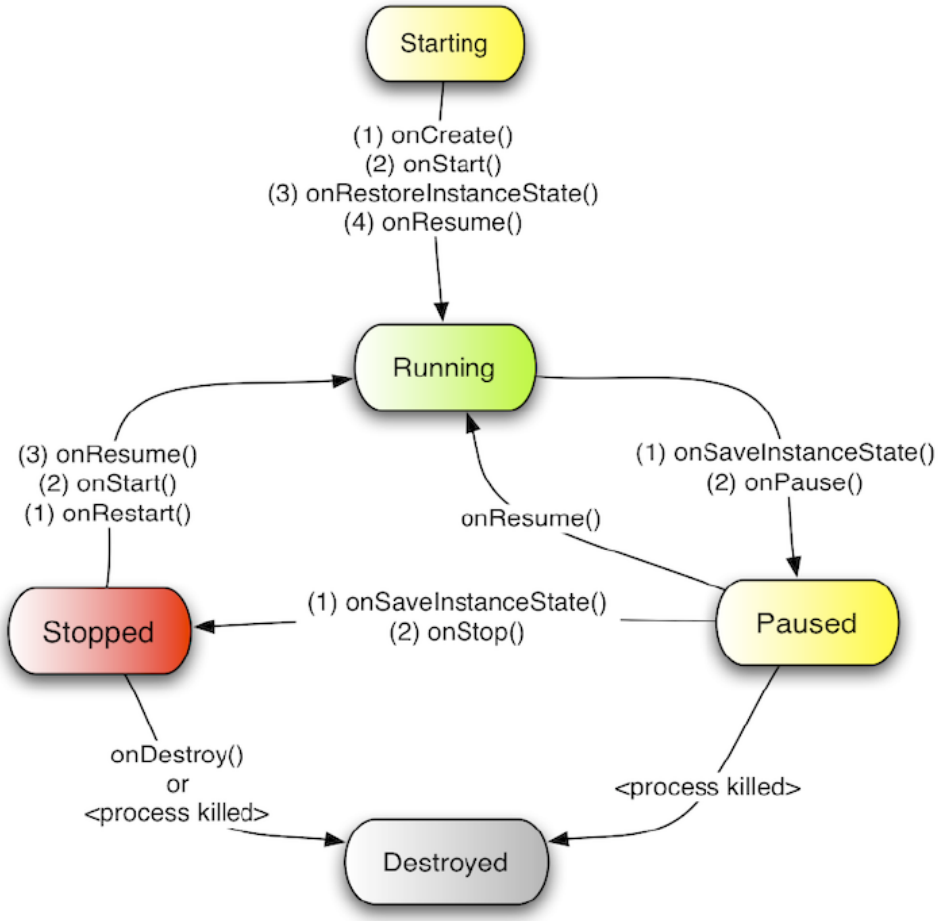

Рисунок 1.2 – Цикл діяльності в Android

Одна дія може запускати/зупиняти інші дії для виконання різних дій у програмі. Коли користувач запускає нову дію, попередня дія припиняється, а система android зберігає процес активності в стеку. Попередню дію можна відновити в будь-який час, натиснувши кнопку «Назад» щоразу, коли користувач завершить поточну дію. Android має дуже чітко визначений

життєвий цикл активності. ОС Android керує процесом діяльності, змінюючи її стан.

Broadcast Receiver — це публічний механізм підписки на основі намірів в Android. Цей компонент програми дозволяє користувачам реєструвати системні події та отримувати сповіщення про активацію зареєстрованої події, наприклад SMS-повідомлення, час роботи акумулятора тощо (рисунок 1.3.).

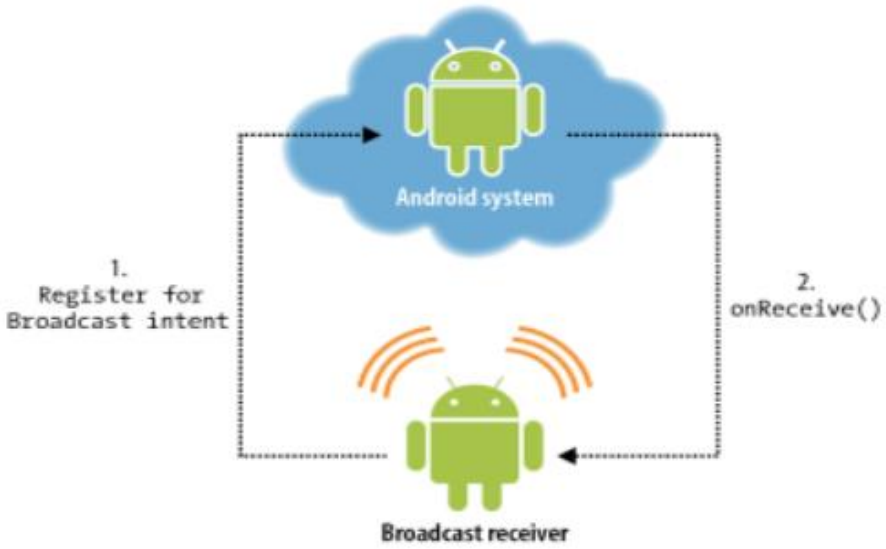

Рисунок 1.3 – Android Broadcast Receiver

Приймач — це просто стек коду в програмі, який активується, коли запускаються події, на які ви підписалися. Система транслює події весь час, і трансльовані події можуть викликати будь-яку кількість приймачів. Трансляції можна надсилати з однієї частини програми в іншу або в зовсім іншу програму. Самі трансляційні приймачі не мають графічного представлення, а також не працюють у пам’яті.

Наміри (Intents) представляють дії або події, які ініціюють активність для запуску, службу для запуску/зупинки або трансляцію в програмі. Наміри – це асинхронні повідомлення, які надсилаються серед основних будівельних блоків. Дія надсилає один або кілька намірів іншій програмі для виконання заданого завдання, наприклад, відкриття веб-сторінки, відтворення медіафайлу тощо. Програми, здатні виконувати такі завдання,

можуть конкурувати за виконання завдання (рисунок 1.4). Якщо є програми-конкуренти, Android просить користувача вибрати між програмами, і користувач може встановити будь-яку програму за замовчуванням.

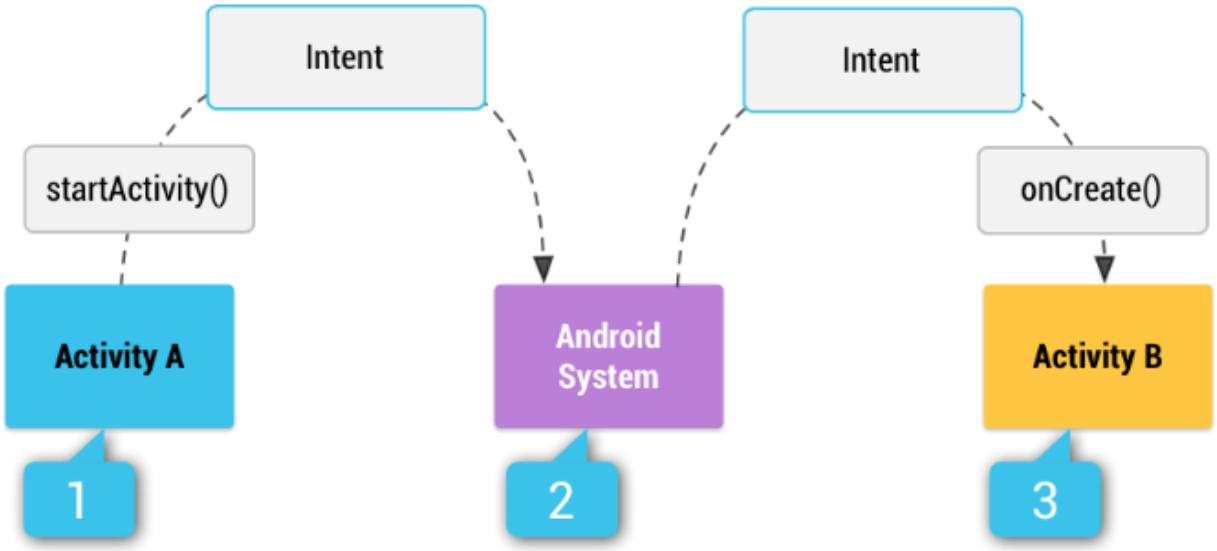

Рисунок 1.4 – Намір Android переходити від однієї діяльності до іншої

Служби (Services) – це компоненти програми, які можуть виконувати тривалі операції у фоновому режимі. Компоненти служби працюють непомітно, оновлюючи джерела даних і видимі дії та запускаючи сповіщення. Це компонент програми, який може запускати службу та продовжувати працювати у фоновому режимі, навіть коли користувач перемикається між різними мобільними програмами. ОС Android надає та обробляє попередньо визначені системні служби, які мають бути оголошені в кожній програмі Android (рисунок 1.5).

Постачальник контенту є елементом додатку, який призначений для керування базами даних програми та забезпечення спільного доступу до даних. Різні додатки можуть обмінюватися спільними даними різними способами залежно від їхнього типу. Одне джерело даних може одночасно використовуватись кількома додатками. Постачальники контенту вважаються найефективнішим засобом для обміну даними між додатками. У самому Android передбачено вбудованих постачальників контенту, зокрема

для аудіо, відео, зображень та контактної інформації користувача.

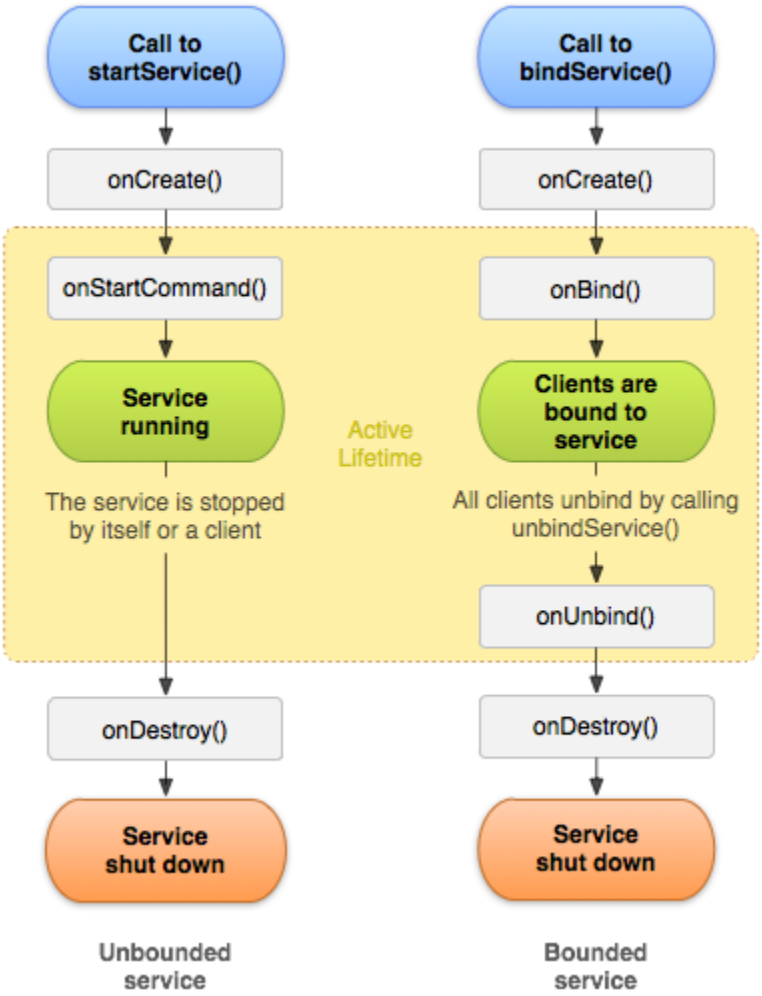

Рисунок 1.5 – Сервіси операційної системи Android

Постачальники вмісту є кращим способом обміну даними через межі програми. Сам Android включає власних постачальників контенту, які керують такими даними, як аудіо, відео, зображення та особиста контактна інформація.

Постачальник вмісту (Content Provider) – це компонент програми, який використовується для керування базами даних програми та спільного використання. Кілька програм можуть обмінюватися одними даними багатьма різними способами залежно від типу даних. Кілька програм можуть підключатися до одного джерела даних одночасно. Постачальники вмісту є кращим способом обміну даними через межі програми. Сам Android включає

власних постачальників контенту, які керують такими даними, як аудіо, відео, зображення та особиста контактна інформація (рисунок 1.6).

Постачальники контенту – це найзручніший спосіб обміну даними між різними додатками. Операційна система Android має вбудованих постачальників контенту, які відповідають за обробку таких типів даних, як аудіофайли, відео, зображення та особиста інформація з контактів.

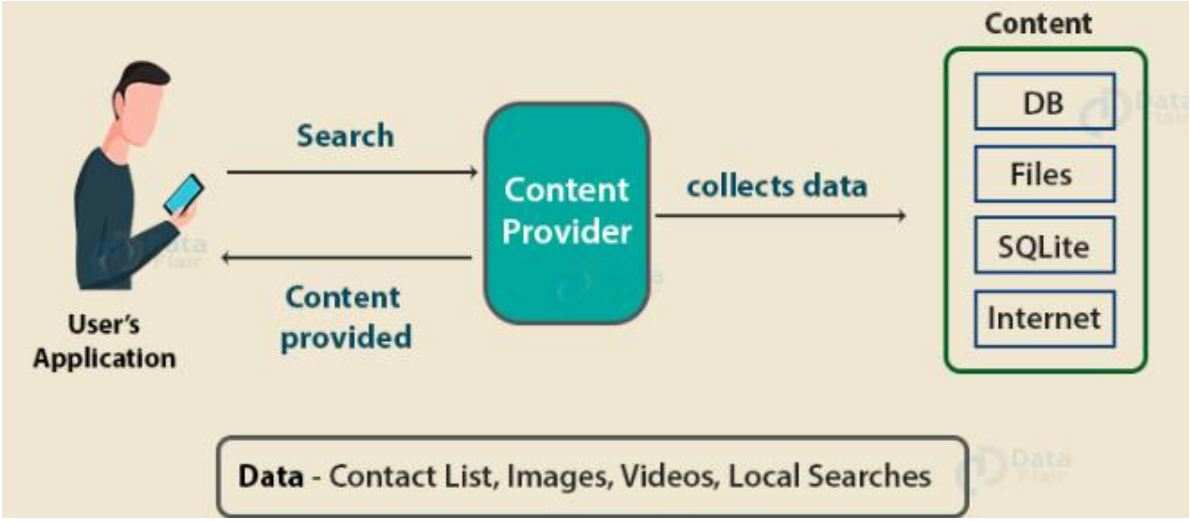

Рисунок 1.6 – Постачальник вмісту Android

1.4 Вибір мов програмування

Мобільний додаток може бути написаний кількома різними мовами та платформами. Даний проект було розроблено з використанням двох мов програмування та формату для зберігання та обміну структурованими даними через мережеве з'єднання, відомого як JSON.

Java — це універсальна, структурована, загальна, заснована на класах мова комп'ютерного програмування. Програми для Android написані мовою програмування Java [11]. Додаток для Android значною мірою базується на основах Java. Java містить кілька потужних функцій і бібліотек багатьох потужних мов програмування, таких як C, C++. Причини вибору Java як рідної мови програмування для програми Android:

- її легко зрозуміти та навчитися;

- вона незалежний від платформи та безпечна;
- вона об'єктно-орієнтована;
- код Java компілюється та виконується віртуальною машиною.

XML – це мова розмітки. Вона містить деякі з дуже простих, масштабованих і гнучких текстових форматів, які можна читати як людиною, так і машиною [12]. Вона визначає набір правил для кодування документа та зручність використання в Інтернеті. XML – це широко використовуваний формат даних в Інтернеті. XML легко використовувати для сборки та маніпулюваням програмами. Ресурси Android попередньо обробляють XML у стиснений двійковий формат і зберігають його на пристрої. Більшість макетів інтерфейсу користувача, елементів екрана оголошено у файлах XML.

JSON (JavaScript Object Notation) — це легкий формат обміну текстовими даними. JSON використовує синтаксис JavaScript для опису об’єктів даних, але JSON усе ще не залежить від мови та платформи [13]. Парсери JSON і бібліотеки JSON існують для багатьох різних мов програмування. Розробнику програми легко читати та писати, а пристроям Android – аналізувати та генерувати дані. JSON походить від мови сценаріїв JavaScript для представлення простої структури даних і асоціативних масивів, які зазвичай називаються об’єктами JSON.

1.5 Вибір середовища розробки

Створити середовище для розробки мобільного додатку для пристроїв Android досить легко. Для початку процесу розробки потрібно лише встановити Eclipse, Android SDK і емулятор Android, хоча пізніше під час процесу можна встановити більше програмного забезпечення та інструментів розробника. Eclipse вважається найкращим доступним інструментом розробки Java, Eclipse IDE для розробників Java забезпечує чудове редагування Java із перевіркою, компіляцією та перехресними посиланнями. Android SDK – це набір для розробки програмного забезпечення, який

дозволяє розробнику створювати програми для платформ Android. Android SDK містить інструменти розробки програм, зразки проектів із вихідними кодами та необхідні бібліотеки для створення програми Android. Емулятор Android - це віртуальний мобільний пристрій, що працює на комп'ютері. Програмне забезпечення емулює пристрій Android, що працює під керуванням ОС Android, для налагодження програм без потреби в різноманітних пристроях і версіях ОС.

Для різних операційних систем доступні різні версії програмного забезпечення, залежно від операційної системи потрібно встановити правильну версію програмного забезпечення [14]. SDK містить усі програми, необхідні для початку процесу розробки програми. За потреби пізніше під час процесу можна встановити більше програмного забезпечення та інструментів розробника.

Eclipse — це колекція інструментів програмування з відкритим кодом, спочатку створена IBM для Java. Зараз більшість розробників у спільноті Java віддають перевагу Eclipse як інтегроване середовище розробки (IDE). Eclipse живе на [15]. Eclipse — це багатомовне середовище розробки програмного забезпечення, яке має інтегровані робочі області інструментів і розширювану систему плагінів. Пакет ADT містить версію Eclipse IDE із вбудованим інструментом розробника Android (ADT) для оптимізації розробки програм Android.

Android SDK надає всі бібліотеки API та інструменти розробника, необхідні для створення, тестування та налагодження програм для Android [16]. Пакет ADT має IDE, уже завантажений із SDK. За замовчуванням інстальовано лише останню версію Android, і, оскільки розробка триває, потрібно інсталювати інші версії Android, щоб підтримувати широкий спектр мобільних пристроїв Android. Не всі пристрої Android використовують останню версію Android, тому розробнику програми важливо встановити діапазон API програми, оскільки деякі класи та бібліотеки знецінюються, починаючи з певного рівня API.

Емулятор Android — це віртуальний пристрій Android, що працює на комп’ютері. Емулятор Android імітує всі функції апаратного та програмного забезпечення типового мобільного пристрою, за винятком того, що він не може здійснювати фактичні телефонні дзвінки. Емулятор дозволяє розробнику програми тестувати програму Android на різних рівнях API без використання фізичного пристрою [17]. Віртуальний пристрій Android (AVD) – це конфігурація пристрою, яка запускається в емуляторі Android. Він працює з емулятором, щоб створити віртуальне середовище для конкретного пристрою, у якому можна встановлювати та запускати програми Android. Менеджер AVD надає графічний інтерфейс користувача, у якому розробник може моделювати різні конфігурації пристроїв Android, які потрібні емулятору Android.

1.6 Постановка мети та завдань роботи

Метою роботи є розробка мобільного застосунку шляхом використання адаптивних підходів та гнучких методологій та хмарних сервісів, що дозволяє зменшити час розробки, підвищити продуктивність і покращити якість кінцевого продукту.

Задачі роботи:

- аналіз сучасного стану розробки мобільних додатків та існуючих методологій;
- визначення основних проблем у процесах створення мобільних додатків;
- розробка ефективного підходу до проектування та впровадження мобільного застосунку;
- реалізація мобільного додатка на основі обраних засобів;
- оцінка ефективності розроблених засобів через експериментальне тестування.

2 РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУВАННЯ

2.1 Пропоноване рішення та розробка

Пропонується розробка нової програми, спрямованої на розподіл інформації між користувачами однієї групи. Додаток дозволить користувачам розміщувати інформацію про предметну область, створюючи платформу для співробітників (постачальників), щоб пропонувати спільну інформацію іншим користувачам або сусіднім групам.

Цей підхід є унікальним, оскільки існуючі платформи в основному зосереджені на спільному обміні інформацією, а не на корпоративному середовищі. Окрему інформацію можна пропанувати як безкоштовно, так і за символічну плату, що відрізняє цю платформу від інших. Ця функція дозволяє співробітникам потенційно заробляти гроші на спільної інформації, причому вартість значно нижча, ніж вартість індивідуального замовлення. Програма має зручний інтерфейс, що дозволяє користувачам легко шукати доступні варіанти залежно від їхнього місцезнаходження або інтересів. Постачальники можуть пропонувати інформацію безкоштовно або за ціною, яка відповідає їхнім уподобанням. Щоб забезпечити безпеку, захист даних і надійність, програма включає процес перевірки, вимагаючи від постачальників надати дійсну адресу електронної пошти для реєстрації.

Після використання співробітники можуть оцінити послугу та надати відгук майбутнім користувачам. Цей інноваційний підхід до вирішення проблеми розподілу інформації, що корисна як постачальникам, так і споживачам. Це зменшує витрати, задовольняє потреби тих, хто шукає доступне джерело інформації, і сприяє розвитку комунікацій.

Методологія, прийнята для розробки додатка дотримувалася підходу Agile, який передбачає ітераційні двотижневі спринти. Кожен спринт включав сеанси догляду та планування для розподілу завдань, а потім

цілеспрямовану розробку та тестування конкретних функцій. Після розробки двотижневий етап тестування підтвердив надійність, а для оцінки було проведено спринт-перевірки прогресу та збору відгуків для постійного вдосконалення.

2.2 Архітектура системи

Архітектура програмного забезпечення складається з набору принципів, які визначають, як проектуються та розробляються програмні системи. Ця структура визначає структуру та організацію програмної системи. Як показано на рисунку 2.1, пропонований додаток включає два основні типи користувачів: постачальників і споживачів.

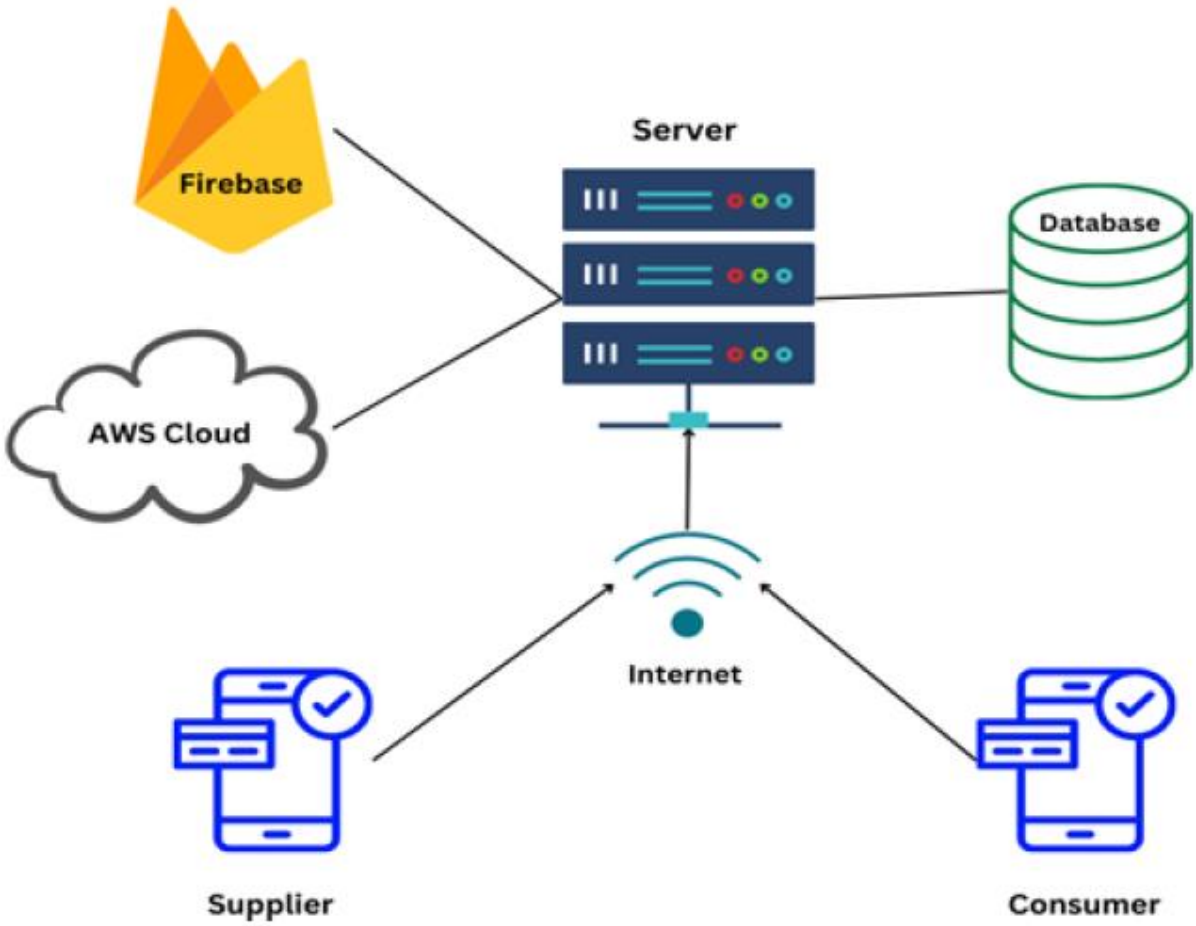

Рисунок 2.1 – Архітектура системи

В обох випадках мобільний додаток підключається до Інтернету для виконання своїх операцій. Коли з’явиться запит, мобільна програма надсилає запит на сервер для отримання даних у реальному часі. Сервер обробляє цей запит, використовуючи дані, що зберігаються в базі даних. Хмарні сервіси AWS (хмарні сервіси AWS) використовуються для зберігання зображень через S3, тоді як Firebase забезпечує функціональність чату в реальному часі.

Додаток надає широкий спектр онлайн-послуг. пропонує послуги за допомогою різних видів діяльності (екранів), містить багато завдань, кожна з яких служить різним цілям програми.

Події (пошук, редагування, відображення) оголошуються в діях, кожна подія, коли вона запускається, викликає ряд дій, а результати відображаються в діях. Процес отримання вмісту з сервера та відображення його на мобільному пристрої можна завершити трьома різними діями, які відрізняються одна від одної. Додаток пропонує користувачам багато послуг.

За домовленістю додаток охоплює одну з дуже важливих послуг. Ця програма отримує загальну інформацію (назва, адреса, контакти, послуги, зображення/відео, карта) про підприємства з бази даних компанії та відображає їх на мобільних пристроях.

Вимоги цього проекту задовольняються трьома різними видами діяльності, а саме: HomePageActivity: це домашня сторінка, яка містить меню, пункти меню, вкладку, поле пошуку, лічильник і кнопку пошуку.

ListDisplayActivity: це дія, яка відображає результати пошуку. CompanyDetailActivity: це сторінка, на якій відображається повний профіль однієї конкретної компанії. Діаграма UML (рисунок 2.2) містить детальний план роботи програми.

Основним фокусом цього проекту є одна з вкладок (компаній) у HomePageActivity, хоча всі інші вкладки, меню, пункти меню та дії можна спроектувати та розробити таким же чином.

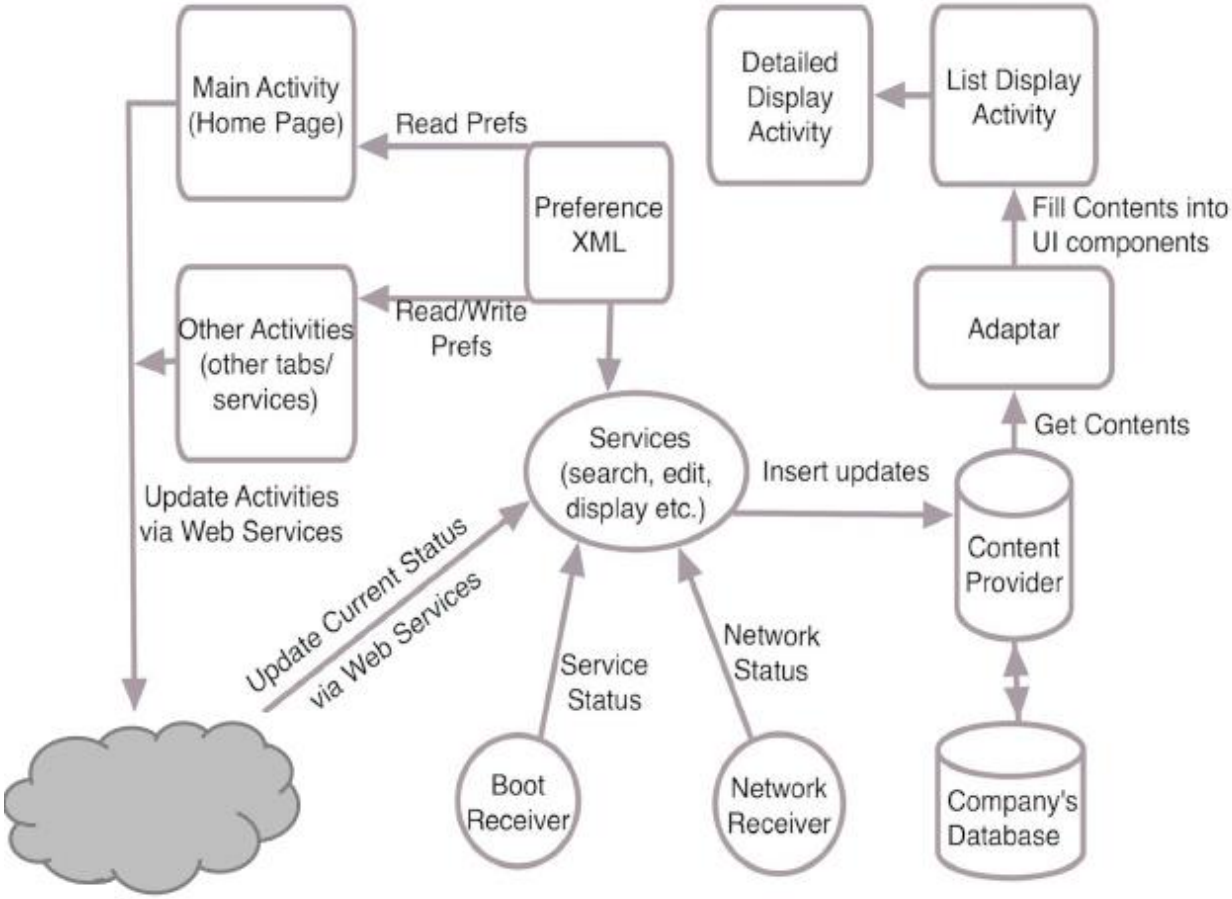

Рисунок 2.2 – UML діаграма системи

2.3 Процес розробки системи

Системна парадигма — це базовий підхід або модель, яка визначає те, як система проектується, розробляється та працює. Він охоплює фундаментальні принципи, концепції та стратегії, які формують спосіб задуму, організації та управління системою.

Беручи до уваги системні парадигми, є два основні ключові гравці, постачальники та споживачі, як показано на рисунку 2.3. Постачальник відповідає за розміщення доступної інформації, а споживач має свободу запитувати доступну інформацію. Якщо обидві сторони погоджуються, вони можуть спілкуватися через канал, наданий у програмі, і домовитися про зустріч у визначеному місці зустрічі.

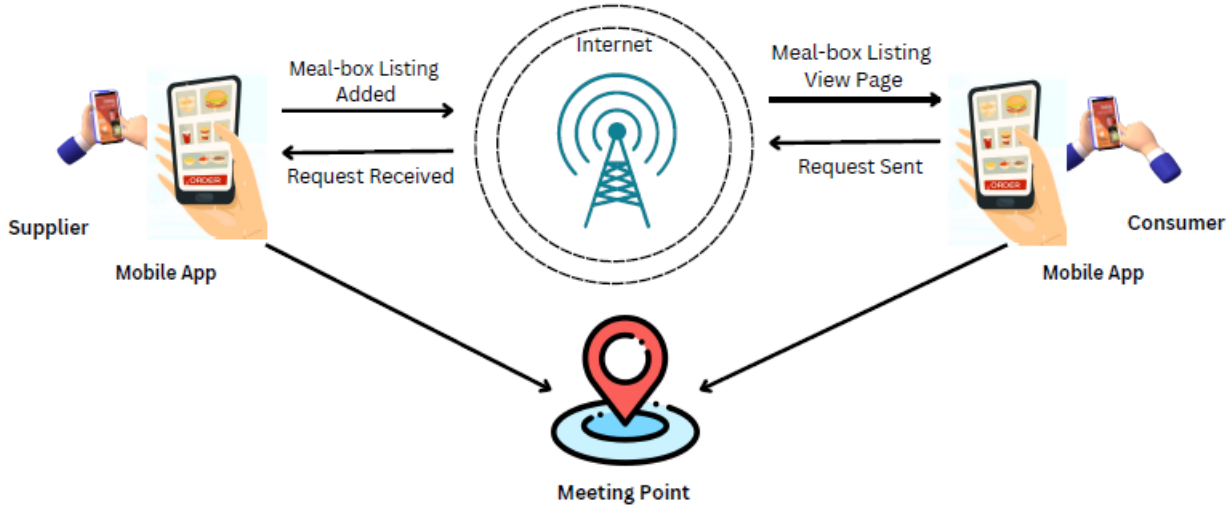

Рисунок 2.3 – Парадигма обміну даними при зміні інформації

Процес проектування починається після налаштування середовища. Процес розробки розпочався з піктограми панелі запуску, а потім вкладок, меню, пункти меню тощо.

Усі програми Android мають піктограму запуску. Піктограма запуску – це візуальне представлення програми на головному екрані пристроїв Android. Щоб запустити програму та відкрити домашню сторінку програми, клацніть піктограму запуску (рисунок 2.4).

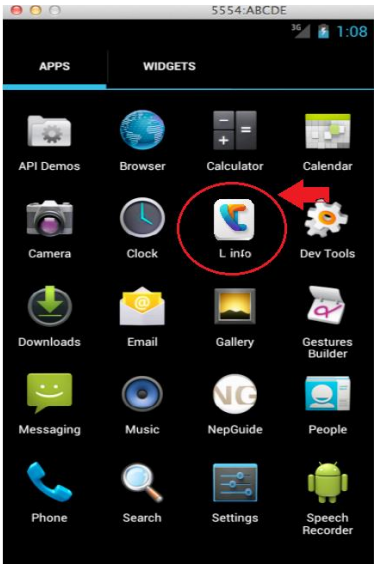

Рисунок 2.4 – Піктограми програм на пристрої Android в емуляторі із піктограмою запуску мобільного додатку "L info"

Піктограму панелі запуску можна створити за допомогою програмного забезпечення для редагування зображень або створити за допомогою стандартних інструментів і параметрів, наданих Android SDK. Піктограма запуску повинна мати розширення .png і префікс ic_launcher.

"L info" має три вкладки, розроблені та створені на домашній сторінці програми. Вкладки в Android створюються за допомогою ActionBar API. API ActionBar було вперше додано в Android 3.0 (рівень API 11), але вони також доступні в бібліотеці підтримки для сумісності з Android 2.1 (рівень API 7) і вище. Наведений нижче код (лістінг 2.1) відповідає створенню вкладки на "L info".

Лістінг 2.1 – Створення вкладок у додатку

```
public void onCreate(Bundle savedInstanceState) {
     StrictMode.ThreadPolicy policy = new
     StrictMode.ThreadPolicy.Builder().permitAll().build();
     StrictMode.setThreadPolicy(policy);
     super.onCreate(savedInstanceState);
     final ActionBar myActionBar = getActionBar();
     myActionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
     Tab MyTab = MyActionBar.newTab();
     MyTab.setText("Companies");
     MyTab.setTabListener(new TabListener<FragmentYP>(this,
     "New Information", MyTab.class));
     MyActionBar.addTab(MyTab);
     Tab WP = myActionBar.newTab();
     WP.setText("People");
     WP.setTabListener(new TabListener<FragmentWP>(this,
     "People", WP.class));
     myActionBar.addTab(WP);
     Tab Map = actionBar.newTab();
     Map.setText("Information Source");
     Map.setTabListener(new TabListener<FragmentImpNumbers>(this,
     " Information Source ",Map.class));
     myActionBar.addTab(Map);
     if (savedInstanceState != null) {
     int savedIndex = savedInstanceState.getInt("SAVED_INDEX");
     getActionBar().setSelectedNavigationItem(savedIndex);
}
```

Вкладка «Компанії» в "L info" пропонує користувачам два типи методів пошуку. Користувачі можуть шукати інформацію про компанію за

назвою або категорією. Коли користувач заповнює всі необхідні поля, програма отримує відповідну інформацію для запиту з бази даних компанії та відображає результат пошуку на екрані ListDisplayActivity. Відповідні пошукові дані з бази даних компанії аналізуються JSONParser у системі Android. У наведеному нижче коді пояснюється, як дані отримуються із сервера за допомогою JSON. Об’єкт JSON використовується для отримання всіх категорій із бази даних компанії до спадного меню (спинера), код якого наведено у лістінгу 2.2.

Лістінг 2.2 – Створення вкладок у додатку

```
JSONObject json = jParser.getJSONFromUrl(url);

final String name=json_data.getString("Name");
final String services=json_data.getString("Services");
final String compid=json_data.getString("CompId");
final String info=json_data.getString("OtherInfo");
final String contact=json_data.getString("Contact");
final String address=json_data.getString("Address");
allItems.put(compid,name+"\n"+address+"\n"+contact+"\n"+services
+"\n"+info);
```

Результати пошуку спрямовуються на екран ListDispalyActivity для відображення результатів пошуку (лістінгу 2.3). Аналогічним чином відбувається заповнення вкладки, що відповідає за відображення об'єктів відображення (компаній, ресторанів, музеїв тощо). Усі вони є екземплярами класу Intent.

Лістінг 2.3 – Відображення результатів пошуку

```
Intent searchInfo = new Intent();
searchInfo.putExtra("items", allItems);
searchInfo.setClass(getActivity(), ListDisplayActivity.class);
Intent company = new Intent();
company.putExtra("items", compid);
company.setClass(ListDisplayActivity.this,DetailActivity.class);
```

2.4 Процес розробки програмного застосунку

Діаграма варіантів використання надає візуальне представлення функціональної взаємодії між різними учасниками (користувачами або системами) і різними варіантами використання (функціональними можливостями або завданнями) системи. На рисунку 2.5 показано дії, які виконують постачальник і споживач.

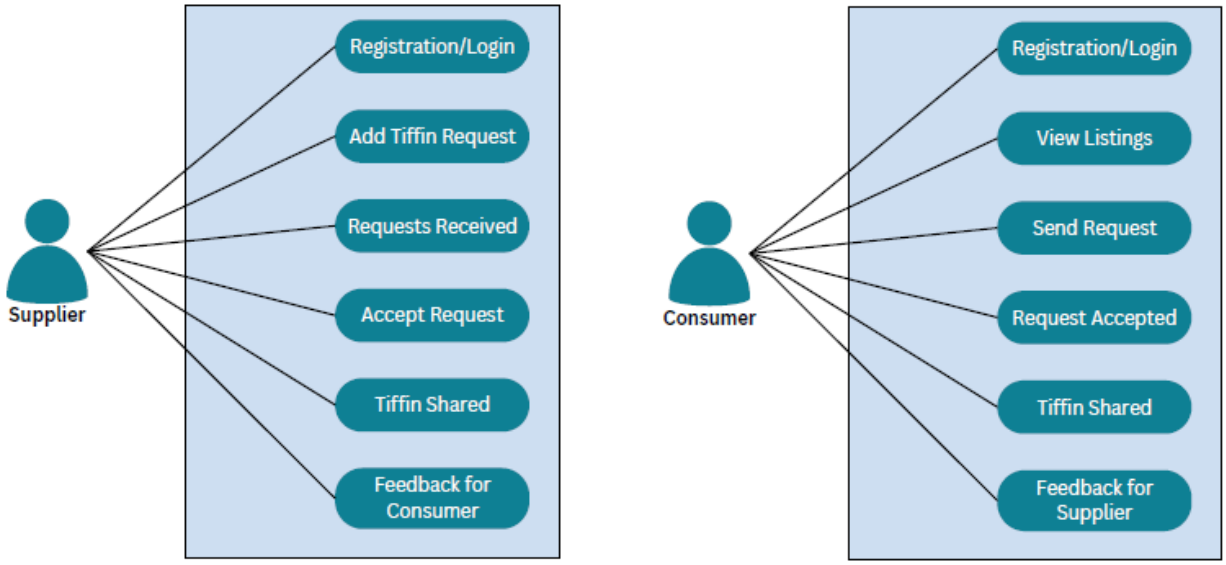

Рисунок 2.5 – Модель прецедентів використання

Блок-схема програми описує покрокову послідовність дій і взаємодій, які виконує користувач у програмі. На рисунку 2.5 показана блок-схема алгоритму програми.

Діаграма потоку даних (DFD), як показано на рисунку 2.6, надає візуальне зображення руху даних у системі чи процесі. У ньому представлено потік даних, операції, які маніпулюють даними та формують їх, сховища даних, а також зовнішні об’єкти, що взаємодіють із системою.

Користувацький інтерфейс "L info" розроблено таким чином, щоб він був зручним для користувача. Різні дії містять різні компоненти інтерфейсу користувача, щоб служити запланованій меті. Головний екран (HomePageActivity) складається з трьох фрагментів (вкладок), панелі дій

(меню) і кількох пунктів меню. Візуальну структуру програми можна отримати в кількох режимах макета, а саме лінійному макеті, відносному макеті та списку.

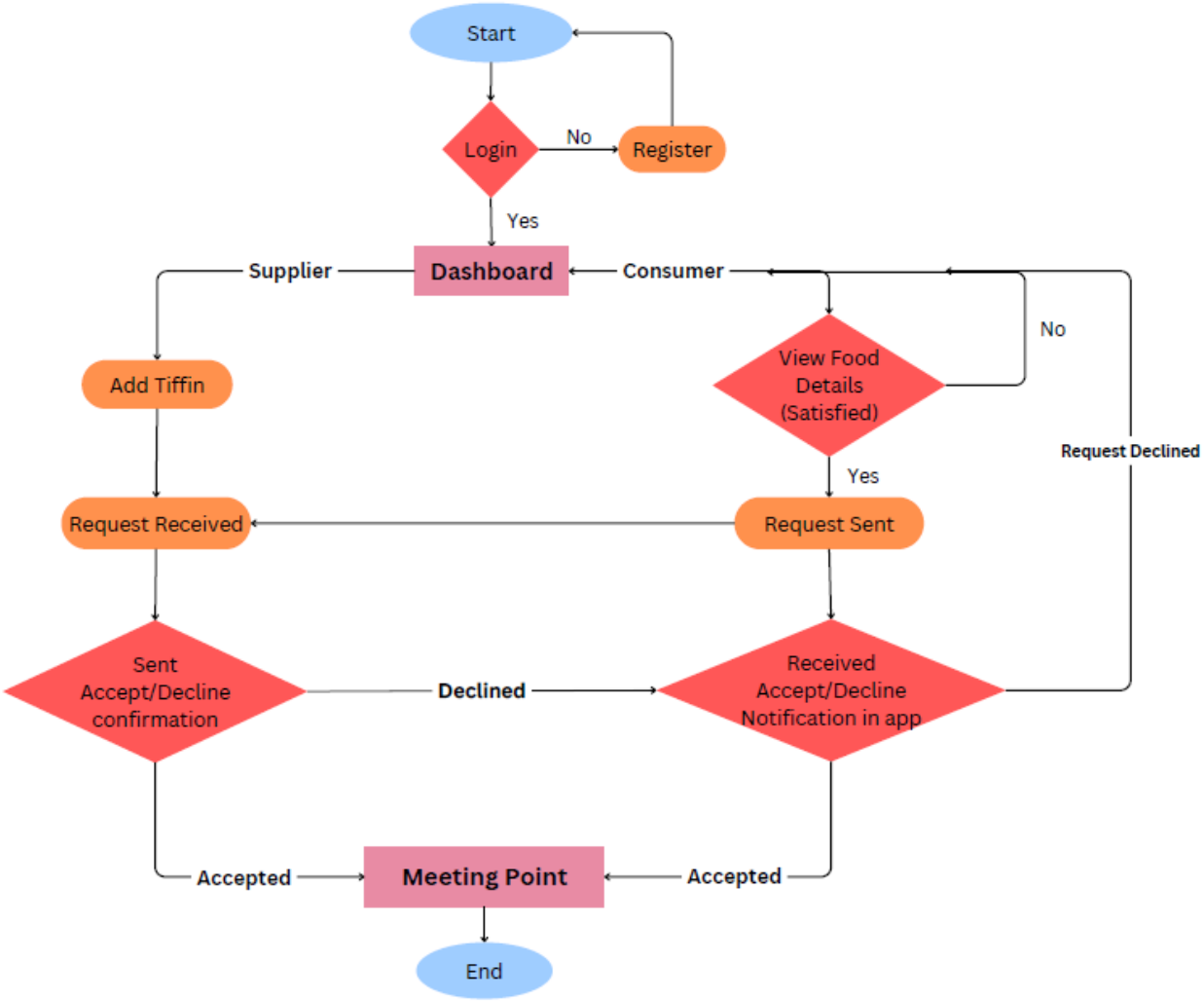

Рисунок 2.6 – Блок-схема алгоритму програми

Взаємодія користувача з програмою може бути доступна за допомогою різноманітних методів керування та введення (рисунок 2.7). Спосіб керування включає кнопки та лічильник, тоді як метод введення включає переважно текстові поля. Користувач також може переміщатися між різними видами діяльності за допомогою пунктів меню, які є загальними для різних фрагментів цієї програми. У ситуаціях, коли потрібно відобразити повідомлення про помилку чи сповіщення, "L info" відображає ці повідомлення через компонент AlertDialog. Фрагмент «компанії»

використовується для пошуку ою'єктів у певному місці (районі чи місті) за назвою чи категорією.

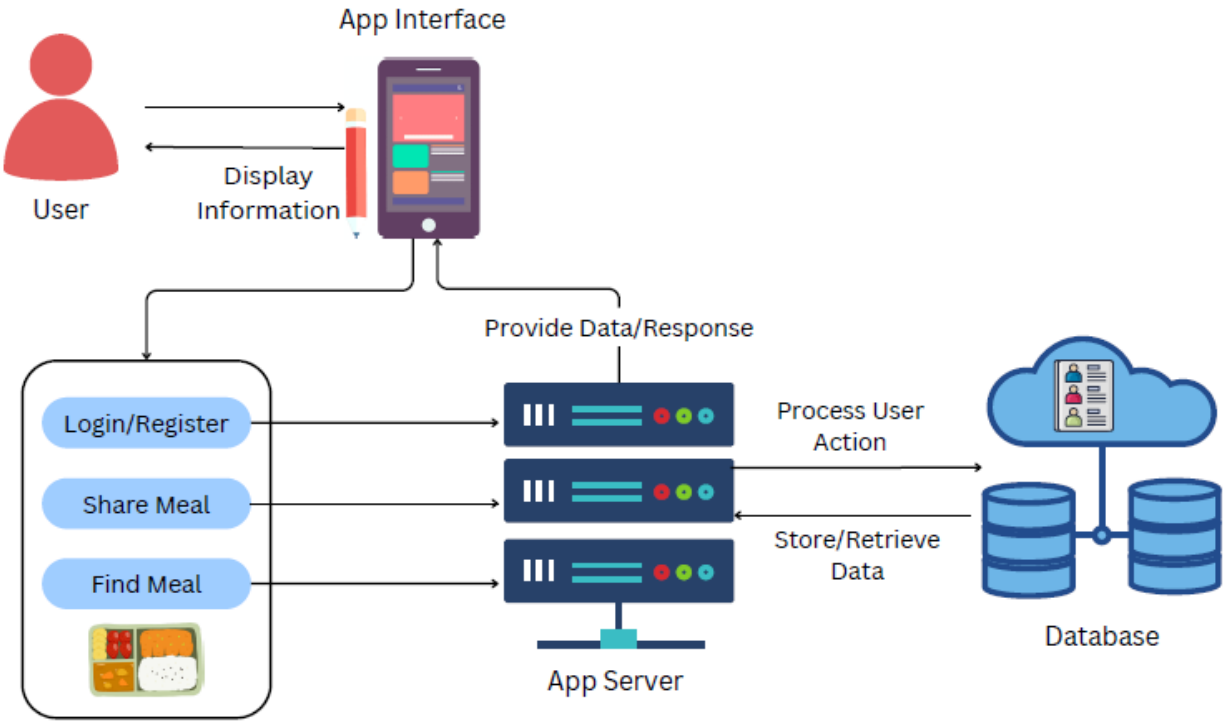

Рисунок 2.7 – Діаграма потоку даних

Результат пошукового запиту відображається у вигляді списку в ListDisplay Activity, крім того, якщо клацнути один зі списків у ListDisplayActivity, повний профіль відповідної компанії відображається в DetailActivity.

Системамає використовує базу даних MySQL, де об'єкти систематично реєструються на основі їх категорії та місця розташування. На сервері є три файли PHP, які містять усі SQL-запити та змінні PHP, необхідні для отримання даних із сервера бази даних. Ці змінні PHP аналізуються за допомогою JSON на об’єкти JSON, файли PHP використовуються в цій програмі для обробки даних.

Після написання всього вихідного коду та виправлення всіх основних помилок (виправлення деяких помилок у цьому проекті потребувало професійної допомоги), "L info" був готовий до тестування. У процесі

розробки кожен етап написання коду та вихід коду тестувався на віртуальному пристрої Android. Після успішного тестування програми на віртуальному пристрої програма була протестована на мобільному пристрої Android.

Метою роботи було досягти того, щоб вкладка об'єктів працювала належним чином. Коли програма запускається, вона спочатку запускає діяльність головного екрана, яка містить усі вкладки та пункти меню для доступу до різних дій. На вкладці компаній користувач має можливість шукати компанії за назвою або категорією в певному місті чи районі. У категорійному пошуку користувач може вибрати категорію зі списку категорій, наданого в спінері.

Результат тестування програми відповідає всім вимогам цього проекту. Програма має у функціонали методи фільтрації категоризованої інформації з сервера та процесом логічного відображення результатів пошуку на пристроях Android.

3 ІМПЛЕМЕНТАЦІЯ ПРОЄКТУ

3.1 Особливості розробки хмарної системи підтримки додатку

Пропоновану програму було розроблено з використанням фреймворку Flutter (Framework Flutter) для розробки мобільних додатків, фреймворку Spring Boot (Framework Spring Boot) для створення серверного API та MySQL як бази даних. Крім того, хмарні служби AWS використовуються для зберігання та отримання зображень і даних. Кодування реалізовано за допомогою Dart для розробки мобільних додатків і Java для створення бекенд-сервісу на основі прикладних фреймворків. Нижче наведено детальний огляд основних компонентів:

Flutter Framework для розробки мобільних додатків: Flutter, набір інструментів інтерфейсу користувача з відкритим кодом, розроблений Google, дозволяє створювати скомпільовані додатки для мобільних пристроїв, Інтернету та настільного комп’ютера з єдиної кодової бази. Його впровадження дозволяє розробляти адаптивний і високопродуктивний мобільний додаток, використовуючи єдину кодову базу Dart, заощаджуючи значний час і ресурси порівняно зі створенням окремих програм для iOS і Android.

Велика бібліотека віджетів Flutter полегшує створення настроюваних і візуально привабливих інтерфейсів користувача, що робить її ідеальним вибором для цього проекту.

Spring Boot Framework для створення Backend API: Spring Boot, популярна платформа на основі Java, спрощує розробку надійних і масштабованих серверних систем. Завдяки таким функціям, як впровадження залежностей, вбудовані сервери та спрощені конфігурації, це спрощує налаштування REST API [35]. Потужна екосистема Spring Boot у поєднанні з

її сумісністю з різними базами даних і хмарними службами забезпечує створення безпечних високопродуктивних серверних служб.

MySQL для керування базами даних: MySQL – це широко використовувана система керування реляційними базами даних, яка забезпечує ефективне зберігання даних, пошук і можливості запитів.

Його сумісність із Spring Boot забезпечує бездоганну інтеграцію з серверною частиною. Масштабованість MySQL і широка підтримка спільноти ще більше підвищують її надійність, роблячи її практичним вибором для управління вимогами програми до бази даних.

Хмарні служби AWS для зберігання та отримання: AWS пропонує набір хмарних інструментів для безпечного зберігання, отримання та обробки зображень і даних. Програма використовує AWS S3 для масштабованого зберігання зображень і RDS для керування базами даних, забезпечуючи високу доступність і надійну роботу. Вбудовані функції безпеки AWS і заходи відповідності забезпечують додатковий рівень довіри для обробки конфіденційних даних.

Кожна з цих технологій була ретельно підібрана, щоб відповідати вимогам додатка щодо продуктивності, надійності та взаємодії з користувачем, створюючи міцну основу для розробки інтерфейсу, і серверу.

3.2 Основні програмні елементи підтримки мобільного додатку з використанням хмарних сервісів

Заставка: після запуску програми на дві секунди відображається заставка, а потім відображається сторінка входу. Щоб увімкнути це в коді фреймворка Flutter, необхідно налаштувати файл «flutter_native_splash.yaml».

Сторінка входу: на екрані входу користувачі можуть додати свою електронну адресу та пароль. Якщо користувач ще не є учасником програми, він повинен спочатку зареєструватися. На екрані реєстрації з’явиться пункт «Приєднатися».

Сторінка реєстрації: щоб отримати доступ до мобільного додатку, користувачі повинні спочатку зареєструватися за допомогою полів, зображених на малюнках нижче, і за допомогою своїх або корпоративних електронних пошт для забезпечення заходів безпеки. Крім того, користувач повинен підтвердити умови використання. Щоб реалізувати карти в додатку, ми застосували залежність “google_maps_flutter”.

Перевірка OTP як захід безпеки: коли користувач заповнює всі дані на сторінці реєстрації, одноразовий пароль буде згенеровано та надіслано на зареєстровану адресу електронної пошти. Надавши отриманий OTP, користувач може успішно увійти. Щоб реалізувати перевірку OTP у Spring Boot, було прийнято залежність “springbootstartermail”.

Екран інформаційної панелі: на екрані інформаційної панелі перераховані всі теми, додані користувачами (постачальниками). Він також має фільтр, бічний ящик і нижню панель навігації, що дозволяє користувачам отримувати доступ до інших сторінок. Користувачі можуть додавати теми до списку за допомогою функції кнопки «Додати».

Сторінка «Додати»: на сторінці «Додати» користувачі можуть вводити такі деталі, як назва, опис, кількість, тип об'єкту, час доступності, наявність характеристик, відповідні збори та місцезнаходження. Крім того, якщо станеться помилка, відобразиться спливаюче вікно.

Сторінка з інформацією про об'єкти: на сторінці з інформацією про об'єкти споживачі можуть бачити всю інформацію, яку постачальник надав під час додавання, і можуть запитувати певні об'єкти. Крім того, на сторінці деталей користувачі можуть переглянути додаткову інформацію у спеціальному розділі.

Сторінки «Мої оголошення» та «Мої запити». На сторінках «Мої оголошення» та «Мої запити» постачальники можуть переглядати, приймати або відхиляти отримані запити. Подібним чином споживачі можуть переглядати свої подані запити та перевіряти їхній статус, щоб переконатися, що вони були прийняті чи ні. Мої оголошення: на сторінці «Мій список»

постачальники можуть керувати об'єктами, якими вони поділилися. Мій запит: на сторінці «Мій запит» споживачі можуть відстежувати статус своїх запитів.

Вікно зв’язку: коли користувач натискає піктограму повідомлення на сторінці «Мій запит» і ім’я користувача на сторінці «Мій список», відкривається вікно зв’язку, що дозволяє як постачальнику, так і споживачу спілкуватися один з одним. Ця функція зв’язку була реалізована за допомогою «Firebase Cloud Firestore».

Бічна шухляда Бічна шухляда висувається з правого боку екрана та містить параметри навігації, як-от списки меню, запити, профілі, параметри виходу та додатковий вміст.

Оновлення сторінок профілю та загальнодоступного профілю: сторінка «Мій профіль» дозволяє користувачам керувати та змінювати інформацію та налаштування свого облікового запису в програмі. Ця сторінка дає змогу користувачам підтримувати точність і актуальність своїх профілів. Користувачі можуть оновлювати зображення профілю, ім’я, контактні дані та місцезнаходження. Після натискання «Переглянути публічний профіль» користувачі можуть переглянути, як їхній профіль буде відображатися для громадськості. У «Загальнодоступному профілі» відображатимуться ім’я користувача, зображення профілю, відгуки, рейтинги, деталі приєднання (дата й час) і статус перевірки. Крім того, у рамках майбутніх удосконалень він також відображатиме список продуктів, доданих цим користувачем.

AWS S3: Amazon Simple Storage Service (Amazon S3) – це хмарна служба зберігання даних, яку надає Amazon Web Services (AWS). Віна пропонує масштабоване та надійне сховище для різних типів даних, таких як документи, зображення, відео, резервні копії тощо.

ВИСНОВКИ

Запропонована програма була розроблена для вирішення проблеми сумісного використання інформації. Хоча існуючі рішення ефективно спрощують обмін даними для домогосподарств і підприємств на основі таких параметрів, як тип об'єкту, місцезнаходження, вони не зосереджені на корпоративних налаштуваннях різних груп населення. Такі програми, як OLIO та Too Good, націлені на ширшу демографію, залишаючи прогалину у задоволенні конкретних потреб корпоративних працівників, які, можливо, можуть поділитися інформацією.

Запропонована програма заповнює цю прогалину, надаючи персоналізовану платформу для різних груп населення, щоб ділитися інформацією, пропонуючи економічно ефективну альтернативу дорогим або менш відповідним варіантам. Він надає пріоритет критичним факторам, таким як розташування, безпека, фірмовий стиль і економія коштів. Високий рівень прийняття 88,9% серед респондентів підкреслює важливість цього рішення. Такі функції, як культурні міркування, підтримка локалізованої мови та планування спеціально задовольняють різноманітні потреби користувачів, підвищуючи його привабливість і зручність у використанні порівняно з більш узагальненими програмами спільного використання. Однак впровадження створює певні труднощі, такі як проблеми з безпекою і юридична відповідальність. Можуть виникнути питання щодо відповідальності, якщо хтось викладає недостовірну або завідомо невірну інформацію, і користувачі можуть вагатися щодо безпеки та якості.

Майбутня робота може вирішити ці проблеми та зосередитися на подальшому вдосконаленні програми. Це включає в себе інтеграцію локалізації та культурних налаштувань, планування та керовані штучним інтелектом алгоритми для рекомендацій відповідних листів інформації на основі вподобань користувача. Крім того, включення систем зворотного

зв’язку та оцінювання може підвищити довіру та підзвітність, одночасно сприяючи постійному вдосконаленню. Ці розробки не лише зменшать потенційні ризики, але й забезпечать масштабованість, ефективність і довгостроковий вплив програми.

Метою роботи була розробка мобільного додатку Android, який міг би взаємодіяти з внутрішнім хмарним сервером. Ці цілі були успішно досягнуті після завершення цього проекту. Проект тестували як на емуляторі Android, так і на мобільному пристрої Android. Програма працювала відповідно завданню, а компоненти інтерфейсу користувача відповідали технічному завданню.

Ця робота використовувала основи роботи з базою даних Android, але у подальшому можна додати більше послуг і функцій. Один із ключових онлайн-сервісів – це база даних. Пропозиції були розроблені та успішно реалізовані за допомогою цієї програми. Хоча ця програма розроблена для конкретних хмарних сервісів, рекомендації, надані в цій роботі, можуть бути корисними для широкого кола розробників, які тільки починають розробляти програми для Android. Пояснювальна записка містить загальну інформацію та рекомендації для розробників додатків щодо операційної системи Android, інструментів розробника та процесу розробки додатків. Оскільки Android є ОС з відкритим кодом, на форумах спільноти з відкритим кодом Android можна безкоштовно знайти безліч ресурсів, інструментів розробника, інструкцій і документації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Deloitte, “There’s no place like phone: Consumer usage patterns in the era of peak smartphone,” pp. 1–61, 2016.

2. R. Colomo-Palacios, J. Calvo-Manzano, A. De Amescua, and T. San Feliu, Agile Estimation Techniques and Innovative Approaches to SW Process Improvement. 2014.

3. V. Rahimian and R. Ramsin, “Designing an Agile Methodology for Mobile SW Development : A Hybrid Method Engineering Approach,” pp. 351–356, 2007.

4. M. Stoica, M. Mircea, and B. Ghilic-Micu, “Software development: Agile vs. traditional,” Inform. Econ., vol. 17, no. 4, pp. 64–76, 2013.

5. W. McIver, “Software Engineering Processes for Mobile Applications Development,” NSERC Mob. First, Frederict., vol. 1, no. 506, pp. 1–74, 2015.

6. H. K. Flora, X. Wang, and S. V.Chande, “An Investigation into Mobile Application Development Processes: Challenges and Best Practices,” Int. J. Mod. Educ. Comput. Sci., vol. 6, no. 6, pp. 1–9, 2014.

7. J. Dehlinger and J. Dixon, “Mobile application SW engineering: Challenges and research directions,” Work. Mob. Softw. Eng., vol. 2, p. 2, 2011.

8. Mamchych O., Volk M. A unified model and method for forecasting energy consumption in distributed computing systems based on stationary and mobile devices. Radioelectronic and Computer Systems, [S.l.], v. 2024, n. 2, p. 120-135. DOI: https://doi.org/10.32620/ reks.2024.2.10.

9. Filimonchuk T., Volk M., Ruban I., Tkachov V. Development of information technology of tasks distribution for grid-systems using the GRASS simulation environment. Eastern-European Journal of Enterprise Technologies. Information and controlling system, 2016. Vol. 3/9 (81). pp. 45–53.

10. A. Dumbravan. Clean Android Architecture: Take a layered approach to writing clean, testable, and decoupled Android applications. Packt Publishing.

2022. P. 368. ISBN : 978-1803234588

11. http://www.oracle.com/technetwork/java/javase/overview/index.html
12. http://www.w3.org/XML/
13. http://www.json.org/
14. http://developer.android.com/sdk/index.html
15. http://eclipse.org
16. http://developer.android.com/sdk/index.html
17. http://developer.android.com/tools/devices/emulator.html