МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ МАРІУПОЛЬСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ФАКУЛЬТЕТ КАФЕДРА

До захисту допустити:

Завідувач кафедри _____

_____

(підпис) *(ПІБ завідувача кафедри)*

«____» _____ 20__р.

ТЕХНОЛОГІЇ ТА МЕТОДИ КРОС-ПЛАТФОРМНОГО ОБМІНУ ДАНИМИ ТА УПРАВЛІННЯМ ПРИСТРОЄМ

Кваліфікаційна робота здобувача вищої освіти першого (бакалаврського) рівня вищої освіти освітньо-професійної програми

«Комп'ютерні науки»

(назва освітньо-професійної програми)

Танчик Богдана Олеговича

(прізвище, імя, по батькові здобувача вищої освіти)

Науковий керівник:

Міщенко А.В., дтн., професор

(прізвище, ініціали, науковий ступінь, вчене звання,)

Рецензент:

_____

(прізвище, ініціали, науковий ступінь, вчене звання, місце роботи)

Кваліфікаційна робота

захищена з

оцінкою_____

Секретар ЕК_____

«____» _____ 20__ р.

Маріуполь – 2023

ЗМІСТ

ВСТУП 5

1. АНАЛІЗ ДОДАТКІВ З ВИКОРИСТАННЯМ СЕНСОРІВ, КАМЕРІ І СПАЛАХУ НА БАЗІ ОС ANDROID ТА ОБҐРУНТУВАННЯ ТЕМИ 9

11.1.1 Мобільні застосунки та їх особливості 9

1.1.2 Склад платформи Android 12

1.1.3 Віртуальна машина Dalvik (Dalvik VM) 13

1.2 Кросплатформна розробка додатків 14

1.2.1 Поняття крос-платформного додатку 14

1.2.2 Різниця між нативною та кросплатформовою розробкою 15

1.2.3 Архітектура iOS/Android та нативні API 16

1.3 Огляд існуючих підходів для роботи з камерою, сенсорами, спалахом 18

1.3.1 Датчикі пристроїв Android 18

1.3.3 Приклади реалізації мобільних додатків 24

1.4 Постановка та формулювання завдання розробки додатку 26

2. Засоби і технології створення МОБІЛЬНОГО ДОДАТКУ 27

2.1 Опис концепції розроблюваного програмного забезпечення 27

2.1.1 Використання Activity 27

2.1.2 Виклик активності за допомогою інтенту 28

2.2 Системні вимоги до розробки додатків на ОС Android. 30

2.3 Засоби розробки додатку 31

2.3.3 Система збірки Gradle 31

2.3.1 Мова програмування Java 33

2.3.2 Крос-платформні компоненти кода і середовища розробки . 37

2.3.3 Ресурси мови XML 38

3. РОЗРОБКА МОБІЛЬНОГО-ДОДАТКУ ДЛЯ РОБОТИ ЗІ СПАЛАХОМ І СЕНСОРАМИ 42
3.1 Основні компоненти додатку 42
3.2 Об'єктно-орієнтована модель додатку 44
3.3 Інтерфейс користувача 47
ВИСНОВКИ 50
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 51
Додаток А 52
Додаток Б 54

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (54 с., 15 рис., 2 додатки).

Ключові слова: АНДРОЇД, ДОДАТОК, СЕНСОРИ, КАМЕРА, СПАЛАХ, КОМПАС, JAVA, ANDROID, ANROID STUDIO, CAMERA, SENSOR MANAGER.

Об’єкт розробки – Мобільний-додаток з використанням сенсорів та спалаху камери і функціями ліхтарика, компаса, засобу контролю орієнтації пристрою.

В ході роботи був проведений детальний аналіз існуючих альтернативних рішень проблеми роботи з сенсорами пристроїв Android та інструментів розробки програм і визначено, що доцільно використовувати існуюче середовище розробки Android Studio і можливості Android SDK.

Середовище розробки і обрана мова програмування є крос-платформними і можуть використовуватись на різних платформах – Windows або Linux незалежно від дистрибутива.

Елементи графічного інтерфейсу користувача були реалізовані за допомогою мови розмітки XML і вбудованого набору віджетів Android.

В розробленому додатку створено наступний перелік функцій:

- Потужний ліхтарик з використанням світлодіодів спалаху;
- Легкий ліхтарик зі зміною кольору екрану в залежності від орієнтації екрану;
- Компас з використанням датчика магнітного поля.

Додаток, що розроблено, надає користувачам максимально зручне рішення для використання апаратних можливостей мобільного пристрою.

В проект додатку закладено можливість подальшої модифікації для покращення функціоналу та розширення з додаванням нових функцій.

ВСТУП

Мобільні пристрої в теперішній час є важливою складовою життя багатьох людей. Ці пристрої використовуються для комунікації між людьми, прослуховування або перегляду музики та відео, записник, компас, ліхтарик та багато іншого.

Найбільш розповсюдженою оперативною системою є Android, який підтримує велику кількість різноманітних пристроїв від різних виробників. ОС Android розповсюджена серед розробників, бо має зрозумілі та безкоштовні засоби розробки додатків під цю платформу. Гаджети на платформі досить доступні, тому зручні для широкого кола користувачів.

Актуальність теми обумовлена широким розповсюдженням мобільних пристроїв, які мають різні типи сенсорів, камеру, спалах. Основною перевагою мобільних додатків є те, що користувач може отримати доступ до них, перебуваючи в будь-якому місці та в будь-якій ситуації. Наприклад, одним з безлічі типів додатків для мобільних пристроїв є додатки для подорожей і туризму. Тому створення мобільного-додатку, який має функції ліхтарика і компаса є дуже актуальним для туристичної галузі.

Об’єкт дослідження – крос-платформна технологія створення мобільного-додатку, який використовує сенсори та спалах камери і має функції ліхтарика, компаса, засобу контролю орієнтації пристрою.

Предмет дослідження – крос-платформний мобільний додаток, з можливістю використання сенсорів та спалаху камери, який має функції ліхтарика та компаса з датчиком магнітного поля.

Мета дослідження - аналіз існуючих технологій та методів крос-платформного обміну даними та управлінням пристроєм (на прикладі мобільного пристрою), вивчення програмних рішень для роботи з сенсорами мобільних пристроїв, крос-платформних інструментів розробки програм, створення мобільного-додатку, який забезпечує

можливість управління деякими сенсорами та приладами мобільного пристрою.

Завдання дослідження:

- Перегляд і засвоєння існуючих рішень для управління апаратними можливостями мобільних пристроїв;
- Перегляд і засвоєння існуючих рішень для роботи з сенсорами мобільних пристроїв;
- Вибір і налагодження крос-платформних інструментів розробки програм для ОС Android;
- Розробка Android-додатку, який використовує сенсори та спалах камери і має функції ліхтарика, компаса, засобу контролю орієнтації пристрою.

Методологія досліджень включала

- методи об'єктно-орієнтовного моделювання,
- побудову тестових програм на платформах Linux або Windows,
- відлагодження тестових програм або додатку в цілому,
- вивчення існуючих аналогів,
- вивчення існуючих технологій розробки для ОС Android.

Практичне значення отриманих результатів полягає в створенні працюючого додатку, який має усі необхідні функції. Додаток готується до розміщення в Android Store.

Обрані середовище розробки і мова програмування є крос-платформними і можуть використовуватись на різних платформах – Windows або Linux незалежно від дистрибутива.

Елементи графічного інтерфейсу користувача були реалізовані за допомогою мови розмітки XML і вбудованого набору віджетів Android.

В розробленому додатку створено наступний перелік функцій:

- Потужний ліхтарик з використанням світлодіодів спалаху;
- Легкий ліхтарик зі зміною кольору екрану в залежності від орієнтації екрану;

- Компас з використанням датчика магнітного поля.

Додаток, що розроблено, надає користувачам максимально зручне рішення для використання апаратних можливостей мобільного пристрою.

В проект додатку закладено можливість подальшої модифікації для покращення функціоналу та розширення з додаванням нових функцій.

При пошуку додатків для завантаження на свій пристрій користувачі обирає найбільш зручні та потрібні для користувача програми. Програми для роботи з технічними можливостями гаджетів – камерою, спалахом, сенсорами – досить розповсюджені і широко використовуються.

Одним з прикладів програм, що розраховані на технічні можливості смартфонів, є програми-ліхтарики, або комплексні програми з використанням деяких типів сенсорів.

Є досить багато безкоштовних програм для ліхтариків в Google Play і їх корисно завжди мати під рукою. Невідомо, коли зникне електрика або користувач вийде в темне місце, але практично завжди під рукою є мобільний телефон. Раніше вам доводилося ходити в темряві шукати свічки чи домашній ліхтар, але завдяки мобільним телефонам у нас завжди є ліхтарик, яким ми можемо користуватися, доки батарея смартфона тримає нас увімкненими. На що ми повинні дивитися? Це буде залежати від того, що вам потрібно… Вам може знадобитися лише світло і нічого більше. Або вам потрібні кольори або ліхтарик не через спалах телефону, але через екран самого мобільного. Деякі висвітлюють сам екран кольорами, повідомленнями. Вам може знадобитися найпростіше або вам потрібні кольори та ефекти.

В деяких варіантах таких додатків можна вибрати джерело світла - спалах або ж екран. Крім того, яскравість світла теж регулюється, на відміну від засобів, які вбудовані в прошивку. Існують також програми, що надають можливість імітувати різні джерела світла. В якості опцій присутні також ряд налаштувань (на кшталт включення світла під час запуску) і установка віджета для швидкого доступу, можливість відображення світлового сигналу

SOS азбукою Морзе. «Морзянкою» вийде передати і будь-який довільний текст: досить просто ввести його в спеціальне поле.

1. АНАЛІЗ ДОДАТКІВ З ВИКОРИСТАННЯМ СЕНСОРІВ, КАМЕРІ І СПАЛАХУ НА БАЗІ ОС ANDROID ТА ОБҐРУНТУВАННЯ ТЕМИ

11.1.1 Мобільні застосунки та їх особливості

Мобільний застосунок (Mobile app) - програмне забезпечення, призначене для роботи на смартфонах, планшетах та інших мобільних пристроях.

Велику долю ринку смартфонів та інших мобільних пристроїв займає компанія Apple з її власною операційною системою iOS. Apple не дозволяє роботу ОС на мобільних телефонах інших фірм. iOS є похідною від OS X, отже, є за своєю природою Unix-подібною операційною системою.

Але на світовому ринку домінує ОС Android, на неї припадає 71% використання смартфонів на початку 2021 року. Частка Android на ринку за останні два роки трохи знизилася, оскільки iPhone 11 та 12 від Apple були добре прийняті аудиторією. [4, с. 346]

Android - операційна система для мобільних пристроїв, заснована на ядрі Linux. Вона є однією з найпопулярніших ОС для смартфонів та планшетів. Також використовується в таких пристроях, як смарт-годинник, електронні книги, музичні плеєри та нетбуки. Її власником, і фактичним розробником, є компанія Google. Історія операційної системи Android починається в 2005 році. Тоді Google купила компанію Android inc, фактичного творця даної ОС та у 2008 році офіційно представлена перша версія операційної системи - Android 1.0. Ця та кожна наступна версія отримувала своє кодове ім'я. Перша версія вийшла з назвою «Apple Pie» (Яблучний пиріг). [2, с. 23]

До переваг операційної системи Android відносяться:

- легка синхронізація пристроїв на Android один з одним або з іншими пристроями (для передачі даних немає необхідності в спеціальних програмах для передачі файлів, як на iOS або Windows Phone);

- відкритість (можливість встановлювати сторонні програми без використання магазину застосунків, що неможливо на інших платформах);
- наявність різних магазинів застосунків (крім Google Play);
- великий вибір смартфонів на Android (вона не є закритою, тому будь-який виробник може встановлювати її на свої смартфони). Тому на даній ОС можна знайти як бюджетні моделі, так і преміум-класу.

До недоліків відносяться: швидкий витрата заряду батареї (найпоширеніша причина критики операційної системи Android), відкритість (завдяки відкритості на Android існує велика кількість вірусів), розтрата трафіку (при підключенні до мережі система самостійно витрачає інтернет трафік на свої потреби). [3, с. 56]

Також безумовно слід враховувати, що на сьогодні під управлінням ОС Android працюють більше 80% всіх смартфонів у світі.

Далі наведемо коротку характеристику деяких популярних мов програмування для розробки мобільних застосунків для Android та iOS:

- Java - строго типізована об'єктно-орієнтована мова програмування, розроблена компанією Sun Microsystems (в подальшому придбаної компанією Oracle). Платформа: Android, основна IDE: Android Studio;
- Swift - мова, розроблена компанією Apple і призначена для розробки застосунків під iOS і OS X. Swift запозичив досить багато з C++ і Objective-C. Платформа: iOS, основна IDE: Xcode; [5, с.45]
- JavaScript - прототипно-орієнтована сценарна мову програмування. Найбільш широке застосування знайшла в браузерах як мова сценаріїв для додання інтерактивності веб-сторінок, а також в кроссплатформних фреймворків (React Native, Ionic, Sencha і т.п.). Платформа: iOS, Android і практично будь-яка інша;
- C# - об'єктно-орієнтована мова програмування розробки застосунків для платформи Microsoft.NET Framework. В області розробки

мобільних застосунків і використовується у фреймворку Xamarin. Платформа: iOS, Android, Windows 10, основна IDE: Visual Studio;

- Objective-C - об'єктно-орієнтована мова програмування корпорації Apple, побудована на основі мови C і Smalltalk. Зараз її замінює новий і більш простий Swift. Проте, деякий час розробники на Objective-C будуть дуже затребувані на ринку. Платформа: iOS, macOS, watchOS і tvOS.

Одне із питань, що виникають на початку розробки мобільного застосунку: створювати мобільний (адаптивний) веб-сайт, нативний або гібридний застосунок. Від прийнятого рішення буде залежати подальший процес розробки. Для розуміння ключових відмінностей, наведемо характеристику нативного та гібридного способів розробки.

Нативні (рідні) застосунки написані на мові програмування, специфічній для платформи, для якої вони розробляються. Зазвичай це Objective-C або Swift для iOS та Java або Kotlin для Android. Нативні застосунки зазвичай мають кращу продуктивність при рендерінгу і анімації, ніж гібридні програми. Але якщо вам необхідна реалізація як для Android, так і для iOS – необхідно буде створити два застосунки.

Гібридний застосунок - це мобільний застосунок, який містить веб-представлення (по суті, ізольований екземпляр браузера) для запуску веб-застосунку із використанням вбудованої оболонки, яка може взаємодіяти із платформою пристрою та веб-представленням. Це означає, що веб-застосунки можуть працювати на мобільному пристрої, маючи доступ до, наприклад, камери або функцій GPS. [10, с.79]

Гібридні застосунки можливі завдяки створеним інструментам, які полегшують зв'язок між веб-представленням і платформою. Ці інструменти не є частиною офіційних платформ iOS або Android, та є сторонніми інструментами, такими як Apache Cordova. Коли гібридний застосунок буде створено, він буде скомпільований, перетворивши ваш веб-застосунок в нативний застосунок. [1, с.456]

Далі зупинимось на створенні саме нативних застосунків під ОС Android.

1.1.2 Склад платформи Android

Платформа Android складається з багатьох компонентів. У неї входять базові застосунки (наприклад, Контакти), набір програмних інтерфейсів (API) для управління зовнішнім виглядом і поведінкою застосунків, а також багатьох допоміжних файлів і бібліотек (рис. 1.1).

При побудові застосунків доступні ті ж API, які використовуються базовими застосунками. За допомогою цих API ви керуєте зовнішнім виглядом і поведінкою своїх застосунків. Під інфраструктурою застосунків розташовується рівень бібліотек C і C++. Для роботи з ними використовуються API. Виконавче середовище Android включає набір базових бібліотек, що реалізують більшу частину мови програмування Java. Кожен Android-застосунок виконується в окремому процесі. У самій основі системи лежить ядро Linux. В Android воно забезпечує роботу драйверів, а також таких базових сервісів, як безпека і управління пам'яттю. [2, с.39]

Пристрої на базі Android не запускають файли.class і.jar. Замість цього для підвищення швидкості та ефективності використання акумуляторів Android пристрої використовують власні оптимізовані формати компільованого коду. Це означає, що ви не зможете скористатися звичайним середовищем розробки на мові Java - вам також знадобляться спеціальні інструменти для перетворення відкомпільованого коду в формат Android, установки його на Android-пристроях і налагодження програми, коли вона запрацює. [4, с. 124]

Всі ці інструменти входять до складу Android SDK. Пакет Android Software Development Kit (SDK) містить бібліотеки та інструменти, необхідні для розробки Android-застосунків.

Зокрема до складу SDK входять:

SDK Platform – окрема платформа для кожної версії Android.

SDK Tools – інструменти відлагодження та тестування, а також інші корисні службові програми. Також включає набір платформно-незалежних інструментів.

Рис.1.1 - Компоненти платформи Andoid

IntelliJ IDEA - одна з найпопулярніших інтегрованих середовищ розробки (IDE) для програмування на Java. Android Studio - версія IDEA, яка включає версію Android SDK і додаткові інструменти графічних інтерфейсів, що спрощують розробку застосунків. Крім редактора і доступу до інструментів і бібліотекам з Android SDK, Android Studio надає шаблони, що спрощують створення нових застосунків і класів, а також засоби для виконання таких операцій, як упаковка застосунків і їх запуск. [3, с. 378]

1.1.3 Віртуальна машина Dalvik (Dalvik VM)

Програми під платформу Android фактично являють собою програми для віртуальної машини Dalvik.

Dalvik Virtual Machine є необхідною частиною мобільної платформи Андроїд. Dalvik VM поширюється як вільне програмне забезпечення під GPL-сумісною ліцензією Apache 2.0. Багато в чому саме цей фактор відіграв свою важливу роль у вирішенні Google відмовитися від JME (Java Micro Edition), на яку потрібно було отримувати ліцензію від Sun. Тому корпорація,

головною метою якої була розробка відкритої операційної системи, створила власну віртуальну машину.

На відміну від багатьох віртуальних машин, які є стек-орієнтованими (Java Virtual Machine також стек-орієнтована), Dalvik є регістр-орієнтованою. Вона добре підходить для роботи на процесорах RISC-архітектури, до яких належать і процесори ARM, які широко застосовуються в мобільних пристроях.

Dalvik замислювалася спеціально під платформу Андроїд. Враховувався той фактор, що платформа представляє всі свої процеси як ізольовані, що виконуються кожен у своєму адресному просторі. Віртуальна машина була оптимізована для невеликого споживання пам'яті та роботи на мобільному апаратному забезпеченні. Dalvik використовує JIT (just-in-time) компіляцію. В результаті таких особливостей, вийшла швидка і продуктивна віртуальна машина, що, звичайно ж, не може не позначатися на роботі додатків в цілому.

Dalvik Virtual Machine використовує власний байт-код. Під час розробки програми під Android перекладаються компілятором у спеціальний машинно-незалежний низькорівневий код. При виконанні на платформі саме Dalvik Virtual Machine інтерпретує та виконує таку програму.[1, с. 298]

Крім цього, Dalvik Virtual Machine здатна переводити байт-коди Java у коди свого власного формату і також виконувати їх у своєму віртуальному середовищі. Програмний код пишеться мовою Java, а після компіляції всі.class-файли конвертуються у формат.dex (придатний для інтерпретації Dalvik) за допомогою спеціальної утиліти dx, яка входить до складу Android SDK.

1.2 Кросплатформна розробка додатків

1.2.1 Поняття крос-платформного додатку

Розробляючи мобільний додаток, варто враховувати кілька факторів: функціональність, адаптивність, вартість, оптимізація. Крос-платформні

додатки передбачають створення коду відразу під кілька операційних систем. Для бізнесу – це можливість заощадити кошти, швидше запуститися та охопити більший сегмент користувачів.

Кросплатформні програми працюють відразу на кількох операційних системах. Завдання програмістів полягає у написанні коду, який добре розгортається на всіх операційних системах.

Універсальний підхід до розробки дозволяє виконати дві важливі умови: економія часу та коштів. Додаток для Android розробляється стільки ж, скільки і для iPhone. Але якщо бізнес замовляє додаток під різні системи, часу на розробку потрібно вдвічі більше (як і грошей).[2, с.455]

Але є й недоліки: кросплатформні програми не такі гнучкі, як нативні, тому що складно реалізувати всі функції, щоб вони добре працювали на різних пристроях. Магазини додатків мають свої вимоги, їх потрібно враховувати під час розробки. Це створює додатковий дискомфорт та труднощі.

Незважаючи на це, кросплатформні додатки дуже популярні та ефективні. Залежно від сфери бізнесу можна створити унікальні інструменти, з якими користувач взаємодіятиме. Швидкий запуск, широке охоплення аудиторії, порівняно невисока вартість розробки дозволяють швидко реалізувати ідеї, запустити потужні інструменти та діджіталізувати бізнес.

1.2.2 Різниця між нативною та кросплатформовою розробкою

Нативні програми розробляються під конкретну операційну систему. Використовується стек технологій, який підходить для вирішення конкретних завдань.

З переваг: висока продуктивність, максимальне використання всіх можливостей платформи, відмінний інтерфейс користувача. У магазинах такі програми краще просуваються. З недоліків: потрібно багато часу, відповідно, вартість створення нативного додатка істотно зростає. Додаток може не підтримуватись на інших операційних системах, тому доведеться розробляти

і для них окремі програми, що також коштуватиме недешево і займе багато часу.[10]

Так у чому ж ключова різниця між нативною та кросплатформовою розробкою додатків? Потрібно заздалегідь розуміти, яку функцію виконуватиме програма для бізнесу, хто входить до цільової аудиторії, навіщо взагалі потрібна розробка. Якщо немає необхідності бути присутнім на двох платформах відразу, є час і бюджет, варто віддати перевагу нативним додаткам. Якщо ж додаток буде простим, функціональним та цілеспрямованим, з чіткими завданнями та цілями, кросплатформне рішення буде кращим.

1.2.3 Архітектура iOS/Android та нативні API

Головний принцип, що лежить в основі кросплатформових рішень - поділ коду на 2 частини:

- Кросплатформну, що живе у віртуальному оточенні і має обмежений доступ до можливостей цільової платформи через спеціальний міст;
- нативну, яка забезпечує ініціалізацію програми, управління життєвим циклом ключових об'єктів і повний доступ до системних API (рис. 1.2). [4, с.187]

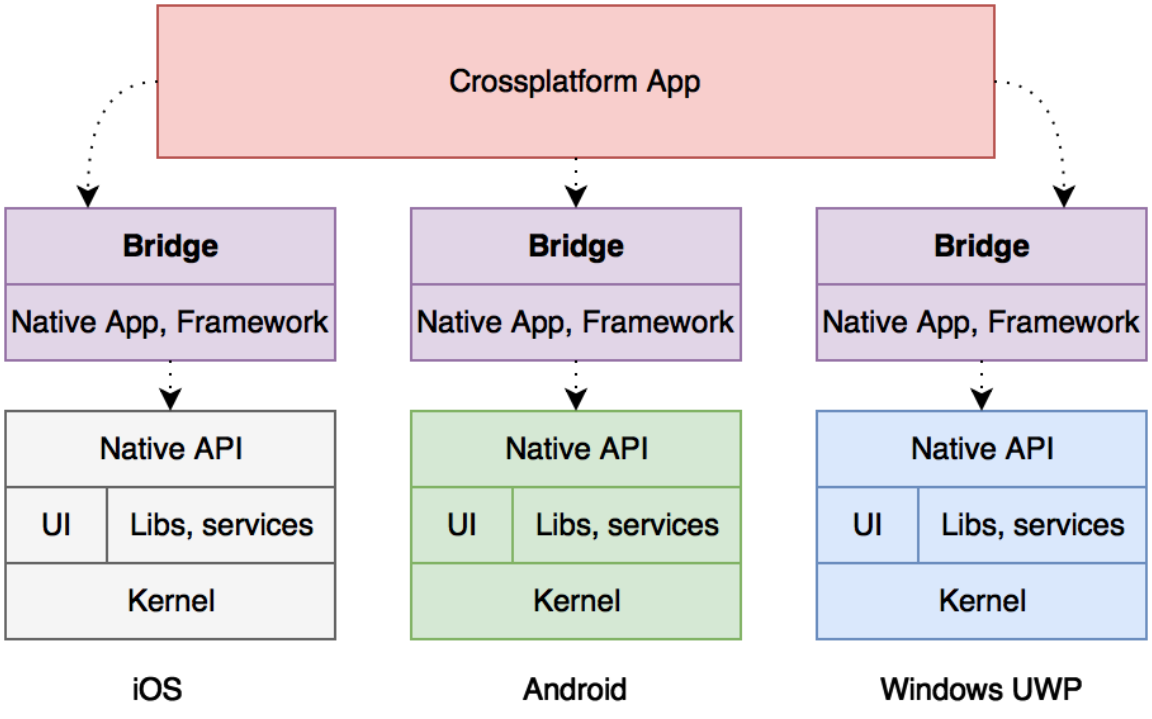


Рис. 1.2 Архітектура крос-платформного додатку

Для зв’язку нативних та крос-платформних частин додатку необхідно використовувати спеціальний міст (bridge), який і визначає можливості та обмеження крос-платформних фреймворків.

Використання bridge завжди негативно позначається на продуктивності за рахунок перетворення даних між частинами, а також конвертації викликів API та бібліотек.

Отже, всі кросплатформні додатки повинні мати нативну частину, інакше операційна система просто не зможе їх запустити.

Наприклад, платформа Xamarin надає міжплатформне рішення для розробки програм для мобільних пристроїв, планшетів і настільних ПК з використанням стеку технологій Microsoft (рис. 1.3). [8, с. 267]

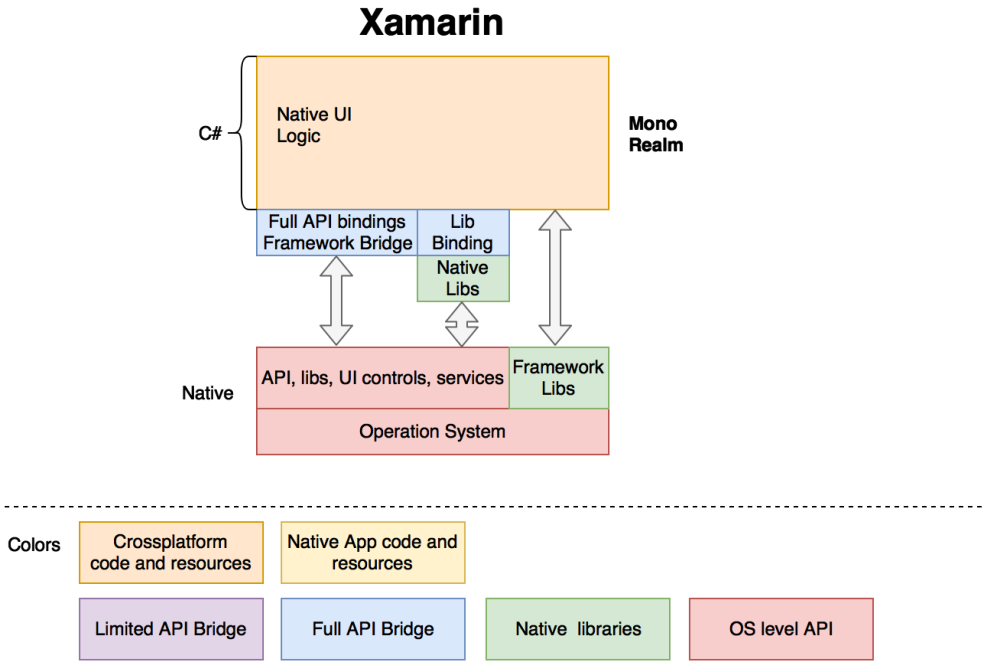


Рис. 1.3 – Архітектура додатку на платформі Xamarin

1.3 Огляд існуючих підходів для роботи з камерою, сенсорами, спалахом

1.3.1 Датчикі пристроїв Android

Більшість пристроїв Android мають вбудовані датчики, які вимірюють рух, орієнтацію та різні умови довкілля. Платформа Android підтримує три широкі категорії датчиків:

- Датчики руху;
- Датчики довкілля;
- Датчики положення.

Деякі датчики є апаратними, а деякі – програмними.

На кожному андроїд-телефоні є датчики сенсори. Одні реєструють параметри довкілля – температуру, тиск, освітленість. Інші визначають положення пристрою у просторі, або щодо магнітних полюсів.

Сенсори поділяються на апаратні – акселерометр, гіроскоп, датчик магнітного поля; та віртуальні, які комбінують або фільтрують дані з апаратних датчиків.

Перелік та призначення основних сенсорів наведено в таблиці 1.1.

Для отримання повного списку датчиків, наявних на пристрої, можна використовувати метод getSensorList класу SensorManager. [3, с 459]

Яким би не був датчик, Android дозволяє нам отримувати необроблені дані з цих датчиків та використовувати їх у додатках. Для цього Android має кілька класів.

Android надає класи SensorManager та Sensor для використання датчиків. Щоб використовувати датчики, перше, що потрібно зробити, це створити екземпляр об'єкта класу SensorManager. Це може бути досягнуто наступним чином:

```
SensorManager sMgr;
sMgr = this.getSystemService(SENSOR_SERVICE);
```

Наступне, що потрібно зробити, – створити екземпляр об'єкта класу Sensor, викликавши метод getDefaultSensor() класу SensorManager. Його синтаксис наведено нижче:[3, с510]

```
Sensor light;
light = sMgr.getDefaultSensor(Sensor.TYPE_LIGHT);
```

Таблиця 1.1 – Типи датчиків на платформі Андроїд

Датчик	**Тип**	**Опис**	**Загальне використання**
TYPE_ACCE-LEROMETER	Апаратне забезпе-чення	Вимірює силу прискорення в м/с^2, яке надається пристрої всіх трьох фізичних осях (x, y і z), зокрема силу тяжіння.	Виявлення руху (тряска, нахил тощо).
TYPE_AMBI-ENT_TEMPE-RATURE	Апаратне забезпе-чення	Вимірює температуру у градусах Цельсія (°C).	Моніторинг температури повітря.
TYPE_GRAVI TY	Програм-не чи апаратне забезпе-чення	Вимірює силу тяжкості м/с^2, яка діє пристрій на всіх трьох фізичних осях (x, y, z).	Виявлення руху (тряска, нахил тощо).
TYPE_GYRO-SCOPE	Апаратне забезпе-чення	Вимірює швидкість обертання пристрою rad/s навколо кожної з трьох фізичних осей (x, y, z).	Виявлення обертання пристрою
TYPE_LIGHT	Апаратне забезпе-чення	Вимірює рівень навколишнього світла (освітлення) lx.	Керування яскравістю екрану.
TYPE_LINE-AR_ACCELE-RATION	Програм-не чи апаратне забезпе-чення	Вимірює силу прискорення в м/с^2, яке надається пристрої всіх трьох фізичних осях (x, y, z), за винятком сили тяжіння.	Визначення прискорення вздовж однієї осі.

TYPE_MAG-NETIC_FIELD	Апаратне забезпе-чення	Вимірює силу навколишнього геомагнітного поля всім трьох фізичних осей (x, y, z) в μT.	Створення компасу.
TYPE_ORI-ENTATION	Програм-не забезпе-чення	Вимірює градус обертання пристрою довкола всіх трьох фізичних осей (x, y, z).	Визначення позиції пристрою.

Продовження таблиці 1.1

TYPE_PRES-SURE	Апаратне забезпе-чення	Вимірює тиск навколишнього повітря на гПа або мбар.	Моніторинг зміни тиску повітря.
TYPE_PRO-XIMITY	Апаратне забезпе-чення	Вимірює близькість об'єкта в см щодо екрана пристрою. Цей датчик зазвичай використову-ється для визначення телефону біля вуха людини.	Позиція телефону під час розмови.
TYPE_RELA-TIVE_HUMI-DITY	Апаратне забезпе-чення	Вимірює відносну вологість у відсотках (%).	Моніторинг точки роси, абсолютної та відносної вологості повітря.
TYPE_ROTA TION_VECTO R	Програм-не чи апаратне забезпе-чення	Вимірює орієнтацію пристрою, надаючи три елементи вектора повороту пристрою.	Виявлення руху та обертання.
TYPE_TEMPE RATURE	Апаратне забезпе-чення	Вимірює температуру пристрою у градусах Цельсія (°C).	Моніторинг температури.

Як тільки цей датчик оголошено, вам потрібно зареєструвати його слухач і перевизначити два методи, а саме AccuracyChanged і onSensorChanged. Синтаксис цих дій виглядає наступним чином:

```
sMgr.registerListener(this,
light,SensorManager.SENSOR_DELAY_NORMAL);
public void onAccuracyChanged(Sensor sensor, int
accuracy) { … }

public void onSensorChanged(SensorEvent event) { … }
```

Можливо отримати список датчиків, що підтримуються вашим пристроєм, викликавши метод getSensorList, який поверне список датчиків, що містить їх ім'я та номер версії, та багато іншого. Потім можна перебрати список, щоб отримати інформацію. Його синтаксис наведено нижче:

```
sMgr =
(SensorManager)this.getSystemService(SENSOR_SERVICE);
List<Sensor> list =
sMgr.getSensorList(Sensor.TYPE_ALL);
for(Sensor sensor: list) { … }
```

Крім цих методів, існують класи SensorManager для керування каркасом датчиків. [3, с567]

1.3.2 Робота з камерою і спалахом

Починаючи з API 21, для роботи з камерою google рекомендує використовувати для роботи з камерою клас camera2, в якому можна включати світлодіод без монопольного захоплення камери.

При старті додатку відразу розпочати роботу з камерою неможливо. Через якісь десятки або сотні мілісекунд камера буде ініціалізована. А як тільки камера готова, спрацьовує коллбек-метод, якщо він прописаний.

Після виконання команди включення камери камера відкрита і може щось робити: виводити зображення з камери на Canvas, пересилати його далі для збереження тощо. Таким чином, вся робота з камерою, по суті, зводиться до прописування різних коллбеків. Повну схему роботи камери наведено на рис. 1.4. [2, с.596]

По-перше, камері потрібен час на включення, по-друге, у андроїда і так повно важливих справ, і ваше побажання ставиться в немалу чергу. Але зате нам не треба чекати відкриття камери в циклі в головному потоці, вішаючи все на додаток (все ж таки пам'ятають, що можна робити в UI-потоці, а що ні).

Таким чином, при роботі з камерою Google бере майже все на себе. Налагодження додатку потрібує прописати в тексті програми коллбеки, які будуть викликатись у разі потреби. Запит на включення камери: mCameraManager.openCamera(). Спрощений варіант схеми роботи камери наведено на рис. 1.5. [2, с.612]

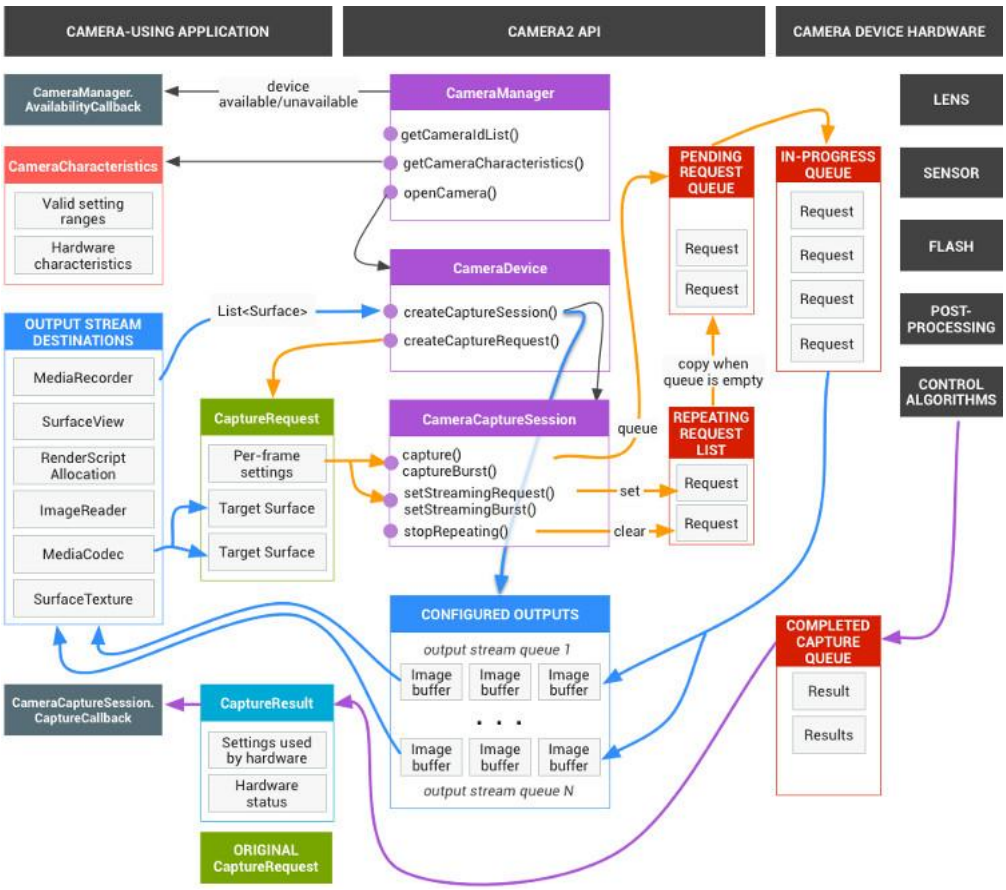


Рис. 1.4 – Схема роботи камери з Camera2 API

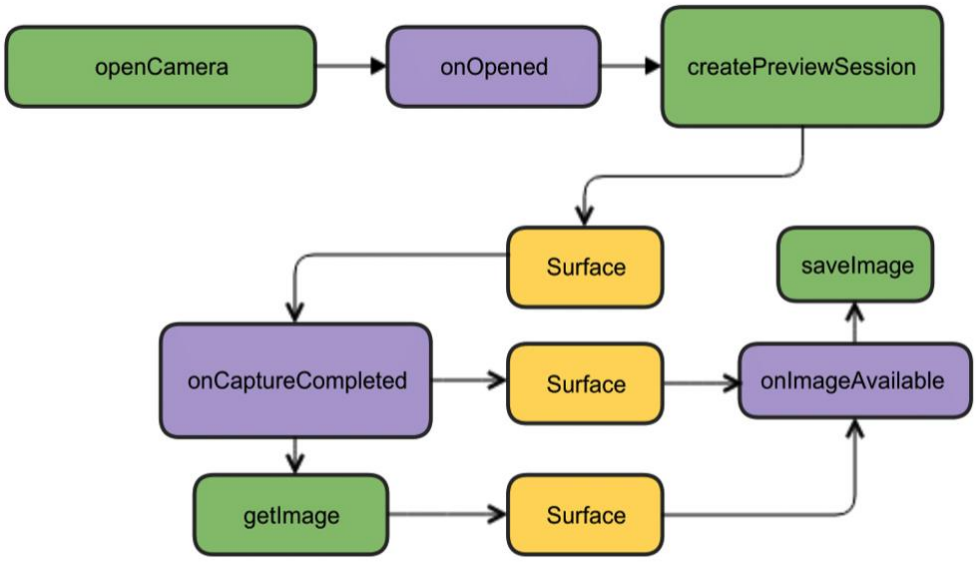


Рис. 1.5 – Схема відкриття камери

Режим ліхтарика стає недоступним, коли пристрій камери, якому він належить, стає недоступним або інші ресурси камери, які йому потрібні, стають зайнятими через інші дії камери з вищим пріоритетом. Режим ліхтарика вимикається, коли його було вимкнено або коли пристрій камери, якому він належить, більше не використовується, а інші необхідні ресурси камери більше не зайняті. Режим ліхтарика камери вимикається, коли програма вимагає CameraManager.setTorchMode(String, boolean) вимкнути режим ліхтарика камери або коли програма вмикає режим ліхтарика іншої камери, якщо не підтримується увімкнення кількох режимів ліхтарика одночасно. Режим ліхтарика стає активним, коли його ввімкнено за допомогою CameraManager.setTorchMode(String, boolean).[4, с 345]

Режим ліхтарика доступний для налаштування CameraManager.setTorchMode(String, boolean) лише тоді, коли він увімкнений або вимкнений.

Камера в основному використовується для зйомки фотографій і відео. Ми можемо керувати камерою за допомогою методів Camera API.

Android надає можливість працювати з камерою двома способами:

1. За призначенням камери
2. Через Camera API

При роботі з камерою і спалахом треба обговорити в першу чергу чотири класи.

1. Intent

За допомогою 2 констант класу MediaStore ми можемо знімати зображення та відео без використання екземпляра класу Camera.

ACTION_IMAGE_CAPTURE

ACTION_VIDEO_CAPTURE

2. Camera

Це основний клас API камери, який можна використовувати для фотографування та відео.

3. SurfaceView

Він являє собою вид поверхні або попередній перегляд камери в реальному часі.

4. MediaRecorder

Використовується для запису відео за допомогою камери. Його також можна використовувати для запису аудіофайлів, як ми бачили в попередньому прикладі медіафреймворку. [4, с.563]

1.3.3 Приклади реалізації мобільних додатків

Розглянемо реалізацію декількох програм-ліхтариків для мобільних телефонів Android.

При розміщенні в Android Store кілька значна частина програм можуть мати внутрішню систему мікроплатежів, яка дозволяє отримати доступ до преміум-контенту (наприклад, до розширених функцій та ексклюзивних функцій).

Розглянемо декілька варіантів додатків з функціями ліхтарика.

Вариант1 (Студія легких програм). В цьому варіанті програма-ліхтарик підтримує жести, тому світло можна вмикати та вимикати, легко та швидко струшуючи з боку в бік. Крім того, є віджет, який ви можете розмістити в будь-якому місці на головному екрані, щоб активувати ліхтарик

в будь-який час, без зайвих слів. Крім того, йдеться про надлегкий додаток, оскільки він важить трохи більше 3 МБ. Нарешті, він є одним з найбільш завантажуваних і використовуваних, з більш ніж 5 мільйонами завантажень і досить респектабельним рейтингом 4.7 зірки, що робить його одним із найкращих для мобільних пристроїв Android; не дарма ми помістили його на перше місце у цій збірці.[10]

Вариант 2. Ліхтарик високої потужності (High Power Flashlight). Це програма з приємним інтерфейсом, з кнопкою в центрі, при натисканні на яку ліхтарик вмикається або вимикається. Крім цього, поставляється з вбудованим компасом, за допомогою якого ви завжди можете орієнтуватися по сторонах світла, особливо якщо немає навіть найменшого світла, ідеально підходить для випадків відключення електроенергії або коли ви загубилися в лісі або іншому місці. Ще одна особливість цієї програми - наявність сигналу SOS та режиму стробування з 10 різними частотами.[10]

Вариант 3. Ліхтарик (Світлана Дев). Це ще одна дуже хороша програма для ліхтарика, яка, хоча і не виділяється безліччю функцій, але є однією з найфункціональніших (вас є те, що ви шукаєте: простий ліхтарик, на який можна розраховувати, коли всюди панує темрява). Інтерфейс цього додатку є одним із найпростіших, і в ньому ви знайдете кнопку по всьому центру, за допомогою якої ви можете увімкнути або вимкнути ліхтарик простим дотиком. Він також має вбудований режим стробоскопа та різні дизайни кнопок живлення, які ви можете налаштувати відповідно до бажаного кольору.

Вариант 4. Flash Flashlight (від RV AppStudios). Цей додаток має надзвичайно цікаву функцію, а саме спалах та масштабування відео, що дозволяє активувати ліхтарик мобільного телефону, в той же час, коли відео дозволяє бачити все через екран, який фокусується. камеру, дуже корисний варіант для пошуку об'єктів у темряві, не наближаючись. Ця функція, яка також діє як збільшувальне скло , звик читати в темряві через екран. Але в такому додатку ефекту стробоскопічного спалаху з регулюванням швидкості

не було. Крім того, ліхтарик Flash Flashlight має світло для економії заряду батареї, параметри налаштування таймера освітлення та регулювання яскравості світлодіодного спалаху, який можна налаштувати за допомогою повзунка. Крім цього, має індикатор заряду батареї. Всі ці параметри та налаштування можна налаштувати через головну панель програми, яка має дуже простий та зрозумілий інтерфейс із кількома кнопками. [10]

Вариант5. Ліхтарик від Programs Splend. Цей додаток має кілька корисних та цікавих функцій: можливість активації/деактивації ліхтарика або заднього світлодіодного спалаху на свій смак; використання екрану як ліхтарика з максимальною доступною яскравістю; присутність віджету, який ви можете розмістити в будь-якому місці на головному екрані, щоб отримати доступ до ліхтарика або увімкнути його; можливість вмикання звуку звук під час увімкнення або вимкнення ліхтарика. [10]

1.4 Постановка та формулювання завдання розробки додатку

Наведений аналіз дозволяє зробити висновок, що розробка крос-платформного комплексного додатку, який має функції ліхтарика та компаса і якийсь інші, є актуальним завданням.

Тому при виконанні дипломної роботи, яка присвячена темі «Технології та методи крос-платформного обміну даними та управлінням пристроєм», був створений додаток, який надає наступні можливості:

- Ліхтарик на базі світлодіодного спалаху камери мобільного пристрою:
- Компас з використанням датчику магнітного поля;
- Контроль орієнтації пристрою з використанням гіроскопу та датчика прискорення;
- Інформаційну сторінку.

Таким чином, функціональність розробленого додатку відповідає сучасним зразкам.

2. Засоби і технології створення МОБІЛЬНОГО ДОДАТКУ

2.1 Опис концепції розроблюваного програмного забезпечення

2.1.1 Використання Activity

Функціональність додатку, що розроблюється, потрібує створення декількох екранів з різними варіантами графічного інтерфейсу користувача (вмикання/вимикання ліхтарику – один варіант інтерфейсу; робота з компасом – інший варіант, робота з екраном орієнтації – третій). Обрання варіанту використання і інформаційної сторінки також вимагає або створення додаткового екрану, або меню з декількома пунктами.

Кожен варіант інтерфейсу базується на концепції активності.

Активність — це єдина цілеспрямована річ, яку може зробити користувач. Майже всі додатки взаємодіють з користувачем, тому клас Activity піклується про створення вікна, у якому розміщуються елементи інтерфейсу користувача за допомогою методу setContentView(View). Можливі дії додатку часто представлені користувачеві як повноекранні вікна, але їх також можна використовувати іншими способами: як плаваючі вікна (через тему з набором R.attr.windowIsFloating), у багатовіконному режимі або вбудовані в інші вікна. [7, с. 257]

Майже всі підкласи Activity реалізують два методи:

- **onCreate(Bundle)** де ви ініціалізуєте свою діяльність. Найважливіше те, що тут ви зазвичай викликаєте setContentView(int) ресурс макета, який визначає ваш інтерфейс користувача та використовує findViewById(int) для отримання віджетів у цьому інтерфейсі користувача, з якими вам потрібно взаємодіяти програмним шляхом.
- **onPause()** де ви маєте справу з користувачем, який призупиняє активну взаємодію з діяльністю. Будь-які зміни, внесені користувачем, повинні бути закріплені на цьому етапі (зазвичай для зберігання ContentProviderданих). У цьому стані діяльність усе ще видно на екрані.

Для використання з Context.startActivity(), усі класи діяльності повинні мати відповідну <activity> декларацію у своєму пакеті AndroidManifest.xml. [5. с.389]

2.1.2 Виклик активності за допомогою інтенту

Intent— це об’єкт обміну повідомленнями, який можна використовувати для запиту дії від іншого компонента програми. Хоча intents (наміри) полегшують зв’язок між компонентами декількома способами, існує три основних випадки використання:

1. Початок роботи Activity. Один екран у програмі відповідає одній Activity. Тому почати новий екземпляр Activity можна, передавши Intent в startActivity().Intent описує дію, яку потрібно розпочати, і містить усі необхідні дані. Якщо необхідно отримати результат від діяльності, коли вона закінчиться, треба використовувати метод startActivityForResult(). Activity отримує результат як окремий об’єкт Intent у зворотному виклику onActivityResult().[9, с.213]
2. Запуск служби. Service— це компонент, який виконує операції у фоновому режимі без інтерфейсу користувача. З Android 5.0 (рівень API 21) і новіших версій ви можете запустити службу за допомогою JobScheduler. Якщо служба розроблена з інтерфейсом клієнт-сервер, ви можете прив’язатися до служби з іншого компонента, передавши Intent до bindService().[2, с.76]
3. Ведення трансляції. Трансляція – це повідомлення, яке може отримати будь-який додаток. Система надає різні трансляції про системні події, наприклад, коли система завантажується або пристрій починає заряджатися. Ви можете відправити трансляцію в інші програми, передавши Intent або sendBroadcast() або sendOrderedBroadcast(). [8, с.432]

Є два типи інтентів:

- Явні інтенти вказують, яка програма задовольнить інтент, надаючи назву пакета цільової програми або повне ім’я класу компонента. Зазвичай ви використовуєте явний інтент, щоб запустити компонент у своїй програмі, оскільки знаєте ім’я класу активності чи служби, яку хочете запустити.
- Неявні наміри не називають конкретний компонент, а натомість оголошують загальну дію для виконання, яка дозволяє компоненту з іншої програми обробляти її. Наприклад, якщо ви хочете показати користувачеві місцезнаходження на карті, ви можете використати неявний намір, щоб попросити іншу відповідну програму показати вказане місце на карті.

На рисунку 2.1 показано, як інтент використовується під час початку діяльності. Коли Intent об’єкт явно називає певний компонент діяльності, система негайно запускає цей компонент.[5, с 569]

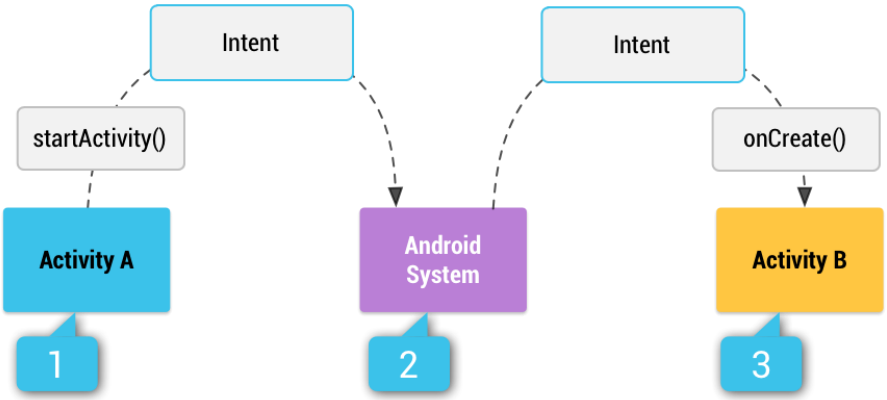


Рисунок 2.1. Використання інтентів для початку іншої активності

На рисунку 1 *дія А* створює Intentз описом дії та передає його до startActivity(). [2] Система Android шукає в усіх програмах фільтр наміру, який відповідає наміру. Коли збіг знайдено, система починає пошук відповідності (*діяльність В*), викликаючи свій onCreate()метод і передаючи йому Intent.

Коли ви використовуєте неявний намір, система Android знаходить відповідний компонент для запуску, порівнюючи вміст наміру з *фільтрами намірів*, заявленими у файлі маніфесту інших програм на пристрої. Якщо намір відповідає фільтру намірів, система запускає цей компонент і доставляє йому об’єкт Intent. Якщо кілька фільтрів намірів сумісні, система відображає діалогове вікно, щоб користувач міг вибрати, яку програму використовувати.

Фільтр намірів — це вираз у файлі маніфесту програми, який визначає тип намірів, які компонент хоче отримати. Наприклад, оголошуючи фільтр намірів для дії, ви дозволяєте іншим програмам безпосередньо розпочинати вашу діяльність із певним типом наміру. Подібним чином, якщо ви *не* оголошуєте жодних фільтрів намірів для дії, тоді її можна розпочати лише з явним наміром.

2.2 Системні вимоги до розробки додатків на ОС Android.

AndroidStudio – це інтегроване середовище розробки для роботи з платформою Android. ПЗ було анонсовано 16 травня 2013 року компанією Google. Софт допомагає розробляти різноманітні програми та ігри під Android.

Це офіційне середовище програмування, яке знаходиться під підтримкою Google. В основі лежить IntelliJ IDEA від JetBrains. [1, с.47]

AndroidStudio має наступні особливості:

- наявність вбудованого емулятора;
- потужний функціонал та інструментарій для розробників;
- вбудований відладчик;
- зрозумілий і добре продуманий інтерфейс;
- наявність документації та багато туторіалів та уроків, за допомогою яких можливо швидко засвоїти платформу;
- присутність гарячих клавіш;
- можливість настроїти Android Studio за кілька хвилин;

- сумісність із більшістю популярних ОС.

Встановлення софту Android Studio проводиться без істотних труднощів. Впоратися з цим завданням максимально швидко та точно допоможуть інструкції, наведені нижче. За рахунок цього інструменту Google вдалося отримувати на виході кросплатформні програми.

Системні вимоги Android Studio до операційних систем наведено в таблиці 2.1. [3, с. 89]

Таблиця 2.1 – Системні вимоги для використання Андроїд студіо

Властивість	**Windows**	**OS X**	**Linux**
Версія OS	Microsoft Windows 11/10/8/7/Vista (64-bit)	Apple macOS 10.8.5 або вище, до 10.13 (High Sierra)/10.14 (Mojave)	GNOME або KDE
Процесор	x86-64 Intel з підтримкою VT-x, або AMD з підтримкою AMD-V, або ARM (для Apple)		
Оперативна пам'ять	8 ГБ (мінімум), 16 ГБ (рекомендується)		
Вільне місце на диску	8 ГБ мінімум (2,5 ГБ для IDE + 5.5 ГБ для Android SDK та образу системи емулятора), 32 ГБ SSD (рекомендується)		
Версія JDK	Java Development Kit 8 або вище		
Роздільна здатність екрану	1280 x 800 (мінімум)		
Додатково	-	Java Runtime Environment (JRE)	GNU C Library (glibc) 2.31 або вище

2.3 Засоби розробки додатку

2.3.3 Система збірки Gradle

Android Studio використовує Gradle як основу системи збірки, а плагін Android Gradle надає додаткові можливості для Android . Ця система збірки працює як інтегрований інструмент із меню Android Studio та незалежно від

командного рядка. Ви можете використовувати функції системи збирання, щоб зробити наступне:

Використовуючи гнучкість Gradle, ви можете досягти всього цього, не змінюючи основні вихідні файли програми.

Файли збірки Android Studio називаються, build.gradle.kts якщо ви використовуєте Kotlin (рекомендовано) build.gradle або Groovy. Це звичайні текстові файли, які використовують синтаксис Kotlin або Groovy для налаштування збірки з елементами, наданими плагіном Android Gradle. Кожен проект має один файл збірки верхнього рівня для всього проекту та окремі файли збірки рівня модуля для кожного модуля. Коли ви імпортуєте наявний проект, Android Studio автоматично створює необхідні файли збірки. [4, с.67]

Система збірки може допомогти вам створити різні версії однієї програми з одного проекту. Це корисно, якщо у вас є як безкоштовна, так і платна версії програми або якщо ви хочете поширювати кілька файлів .apk для різних конфігурацій пристроїв у Google Play.

Підтримка кількох файлів APK дозволяє ефективно створювати кілька файлів APK на основі щільності екрана або ABI. Наприклад, ви можете створити окремі файли .apk програми для щільності екрана hdpіта mdpi, водночас розглядаючи їх як єдиний варіант і дозволяючи їм спільно використовувати параметри тестового файлу .apk, javac, dx і ProGuard.

Процес створення включає в себе багато інструментів і процесів, які перетворюють ваш проект на пакет Android Application Package (APK) або Android App Bundle (AAB).[3, с.167]

Плагін Android Gradle виконує більшу частину процесу збирання за вас, але він може бути корисним для розуміння певних аспектів процесу збирання, щоб ви могли налаштувати збірку відповідно до своїх вимог.

Різні проекти можуть мати різні цілі створення. Наприклад, збірка для бібліотеки третьої сторони створює бібліотеки AAR або JAR. Однак програма є найпоширенішим типом проекту, і збірка для проекту програми

створює файл APK або AAB для налагодження або випуску вашої програми, який можна розгортати, тестувати або випускати для зовнішніх користувачів.

Ця сторінка зосереджена на розробці додатків, але багато етапів і понять є загальними для більшості типів збірок.

2.3.1 Мова програмування Java

Java - це мова, програми якою компілюються та інтерпретуються, і водночас вона має просту структуру мови високого рівня. Написана програма компілюється в проміжну форму - *байткод*. Пізніше ця програма виконується, тобто інтерпретується виконавчим середовищем *Java*. Байткод дуже відрізняється від машинного коду, який є послідовністю нулів та одиничок. Байткод - це набір інструкцій, які подібні до команд *Асемблера*. Машинний код комп'ютер виконує безпосередньо, а байткод потрібно інтерпретувати. Тому машинний код можна використати тільки на конкретній платформі, для якої його скомпільовано. Байткод можна виконувати на довільній платформі, на якій є виконавче середовище *Java*. Саме ця можливість і робить програми на *Java* незалежними від архітектури. Так як байткоди є проміжною формою програми, то його інтерпретація вимагає незначних витрат. [6, с.156]

Байткод створено для машини, яка реально не існує. Цю машину називають *віртуальною Java-машиною* (*JVM*), вона існує тільки в пам'яті комп'ютера. Створення компілятором *Java* байткоду для неіснуючої машини - це тільки половина процесу, який забезпечує незалежність від архітектури. Другу частину виконує інтерпретатор *Java*, який виконує роль посередника між віртуальною *Java*-машиною та реальним комп'ютером. [7, с.278]

Переваги мови програмування Java:

Перевага 1. Java легко вивчається новачками

Синтаксис цієї мови програмування схожий на звичайну англійську мову. В ній є мінімальна кількість складних для запам'ятовування символів.

По суті, після встановлення JDK, настройки PATH і вивчення особливостей Classpath спеціаліст вже може створювати елементарні програми на Java.[9, с.75]

Перевага 2. У Java прийняті концепції хорошого програмування

Java — об'єктноорієнтована мова, причому це саме «об’єктне» в Java реалізоване просто на відмінно! Разом з ООП Ви вивчите концепції наслідування, абстракції, поліморфізму і так далі. Java навчить концепціям, які можна застосовувати в більшості інших мов, наприклад, в Python. [8, с.109]

Перевага 3. Java така мова програмування рідко змінюється

Для новачків це ВЕЛИКИЙ плюс, що не потрібно відволікатися і постійно слідкувати за новими тенденціями IT. Новий функціонал Java інколи навіть занадто довго вводиться… наприклад, подібне відбувалося із замиканнями.

Перевага 4. Користувачі Java мають доступ до великої колекції бібліотек з відкритим кодом.

Найкращі програмісти світу розробили шаблони, які роблять процедуру розробки більш простою. Підтримка з боку таких гігантів, як Google і Apache, дає зрозуміти, що ця мова програмування ще довго буде використовуватися в проектах. Java має гарно пророблений API, великий вибір інструментарію, велику кількість фреймворків.

Перевага 5. Java має гігантську спільноту по всьому світу

Існує маса форумів, тематичних порталів та інших ресурсів, де обмінюються знаннями користувачі Java. Там спілкуються досвідчені програмісти, переймають досвід новачки, обговорюють новини, пов’язані з мовою програмування. Ви можете бути впевнені в тому, що ніяких проблем з пошуком інформації точно не буде.

Велика частина сучасних стартапів не використовує Java, через те, що на цей момент існують більш швидкі шляхи виконання того ж обсягу роботи. Проте ми навчаємо вивчати те, що приносить найбільше задоволення, так Ви

станете справжнім експертом в обраній справі, зокрема кодуванні улюбленою мовою програмування.

Архітектура мови для розподіленого мережевого середовища. Головною вимогою щодо мови для роботи в розподіленому просторі комп'ютерів (наприклад, в Internet) − це можливість працювати на різнорідних і розподілених платформах. Мова Java є пристосованою до перенесення завдяки підтримці стандартів IEEE для структур даних, наприклад, цілих чисел, чисел з плаваючою комою і рядків.

До мови Java зачислено безпосередньо підтримку таких розповсюджених протоколів як FTP, HTTP, що забезпечує сумісність під час роботи в мережі.

Java забезпечує розподілену роботу за допомогою механізму виклику віддалених методів (RMI), тобто дає змогу використовувати об'єкти, розташовані на локальних і віддалених машинах.

Багатопоточність. У багатопоточних операційних системах для кожного застосування (процесу) надається окрема захищена область пам'яті, в якій зберігаються коди програми і дані. А час одного процесора квантується між цими процесами. З метою запуску процесу або переключення з одного на інший на рівні операційної системи необхідно виконати значний об'єм роботи.

Тому для розробників прикладних програм спеціально створили "полегшену" версію системного процесу − потік. Найбільшою проблемою, пов'язаною з процесами і потоками, є їхнє функціонування під керівництвом конкретної операційної системи. [8]

Спеціалісти компанії Sun зробили потоки частиною мови програмування. Тому багатопоточне застосування, написане мовою *Java*, працюватиме і в операційних середовищах *Windows, Unix, MacOs*.

Висока продуктивність. Інтерпретатор *Java* може виконувати байткоди зі швидкістю, яка наближається до швидкості виконання коду, відкомпільованого до машинного формату, що досягається завдяки

використовування інтерпретатором багатьох потоків виконання. Наприклад, доки комп'ютер чекає на введення даних, фонові потоки можуть зайнятися очищенням пам'яті. [9]

Стійкість до помилок. *Java* - це мова строгого використання типів, що зумовлює зменшення числа помилок під час написання програми. У мові Java відбувається автоматична перевірка виконання граничних умов під час роботи з масивами і стрічками, які в *Java* є класами.[7]

У *Java* немає арифметики покажчиків, а керування пам'яттю здійснюється автоматично. Програмний код, написаний мовою *Java* не може зіслатися на пам'ять поза простором програми або зробити помилку внаслідок вивільнення пам'яті і тим самим вичерпати всю пам'ять. Зазначимо, що у *Java* організовано процес автоматичного збирання сміття, тобто об'єктів, на які більше ніхто не вказує.

Безпека. Функції забезпечення безпеки дуже важливі для розподілених мереж з безліччю вірусів, "троянських коней" і т. п. Для реалізації цієї мети розробники мови *Java* створили механізм, який отримав назву пісочниці (*sandbox*): - перевірку на рівні *JVM*; - захист на рівні мови; - інтерфейс *Java Sucurity* (цифрового підпису).[5]

Простір імен. Під час написання складних програм інколи важко забезпечити унікальність імен змінних і класів. У мові *Java* використовують систему декількох рівнів вкладеності імен:

0 - простір імен пакета;

1 - простір імен одиниці компіляції (файл класу);

2 - простір імен типу (клас у класі);

3 - простір імен методу;

4 - простір імен локального блоку;

5 - простір імен вкладеного локального блоку.

За підтримку і перетворення просторів імен відповідає компілятор *Java*. Імена, пов'язані з кожним рівнем, відокремлюють від імен інших рівнів крапкою. Наприклад, Java.awt.BorderLayout. [8, с.490]

Файли програми. Файл з вихідним текстом програми мовою *Java* є звичайним текстовим файлом з розширенням .java. Після компіляції (за допомогою javac) вихідних текстів отримується по одному файлу для кожного класу, оголошеного в тексті програми. Імена файлів збігаються з іменами класів (з урахуванням регістра) і мають розширення .class.

Пакети. У багатьох мовах програмування набір зв'язаних класів або функцій називають бібліотекою. *Java* надає поняттю бібліотеки певний відтінок, використовуючи для описання набору зв'язаних класів термін *пакет.* Наприклад, базові функції *Java* розташовані в пакеті java.lang.

Оператори імпорту. З метою використання класів з існуючих пакетів необхідно до тексту програми (першими) додати оператори *import java.awt.*;* Це означає, що всі класи пакета java.awt можна використовувати при написанні програмного коду безпосередньо без посилання на пакет. ***Оголошення класів.*** Усі класи в мові *Java* є похідними від системного класу *Object*. У *Java* допускається тільки одинарне наслідування, тобто в ієрархії перед класом є тільки один базовий клас. [5]

Оголошення інтерфейсу. Інтерфейсом у мові *Java* називають абстрактний клас. Його введено для реалізації наслідування від декількох класів.[9]

2.3.2 Крос-платформні компоненти кода і середовища розробки

Значна частина кода додатку є нативною за рахунок взаємодії активностей компонентів додатку з сенсорами, спалахом і камерою.

Інтерфейс звертання до компонентів – крос-платформна частина додатку.

Цілком крос-платформним є середовище розробки – Android Studio, яке має версії для Windows і Linux (в роботі були використані обидві, тому що при розробці під Windows значно частіше був отриманий пакет із зараженням вірусами, при збиранні пакета під Ubuntu віруси були відсутні).

Крім того, в додатку був використаний Cross device SDK, який полегшує розробникам створення програм, сумісних на кількох пристроях. SDK спрощує розробку різноманітних і привабливих можливостей для кількох пристроїв, поєднуючи різні технології підключення в один інструментарій. Раніше розробникам доводилося самостійно працювати з інфраструктурами підключення, як-от Bluetooth і Wi-Fi, щоб створювати можливості для кількох пристроїв. Тепер розробники можуть зосередитися на найважливіших частинах взаємодії з користувачем, тоді як SDK обробляє ці технології нижчого рівня.

Цей SDK є частиною нашого більшого набору інструментів розробки для кількох пристроїв, який включає підтримку емулятора, профілювання тощо. Cross device SDK забезпечує наступні основні функції: [7]

- Виявлення та авторизація пристрою;
- Безпечні з’єднання та передача даних;
- Сеанси на кількох пристроях.

2.3.3 Ресурси мови XML

Ресурси — це додаткові файли та статичний вміст, який використовує ваш код, наприклад растрові зображення, визначення макета, рядки інтерфейсу користувача, інструкції щодо анімації тощо.

Завжди використовуйте зовнішні ресурси програми, такі як зображення та рядки, із вашого коду, щоб ви могли підтримувати їх незалежно. Крім того, надайте альтернативні ресурси для певних конфігурацій пристроїв, згрупувавши їх у спеціально названих каталогах ресурсів. Під час виконання Android використовує відповідний ресурс на основі поточної конфігурації. Наприклад, ви можете надати інший макет інтерфейсу користувача залежно від розміру екрана або різні рядки залежно від налаштувань мови.[3]

Після того, як ви виведете ресурси програми назовні, ви зможете отримати до них доступ за допомогою ідентифікаторів ресурсів, створених у R-класі вашого проекту. У цьому документі показано, як згрупувати ресурси

у вашому проекті Android. Тут також показано, як надати альтернативні ресурси для певних конфігурацій пристроїв, а потім отримати доступ до них із коду програми чи інших XML-файлів.

Кожен тип ресурсу розміщується в певному підкаталозі каталогу res/ проекту, який розроблюється. Наприклад, нижче наведена ієрархія файлів для простого проекту MyProject: [2]

```
src/
   MyActivity.java
res/
   drawable/
      graphic.png
   layout/
      main.xml
      info.xml
   mipmap/
      icon.png
   values/
      strings.xml
```

Каталог res/ містить усі ресурси у своїх підкаталогах: ресурс зображення, два ресурси макета, каталог mipmap/для піктограм запуску та файл рядкового ресурсу. Імена каталогів ресурсів є важливими та описані в таблиці 2.2. []

Таблиця 2.2 - Каталоги ресурсів, які підтримуються в res/каталозі проекту.

Довідник	**Тип ресурсу**
animator/	XML-файли, які визначають властивості анімації.
anim/	Файли XML, які визначають анімацію Tween . Анімації властивостей також можна зберігати в цьому каталозі, але цей каталог є кращим для анімацій властивостей, щоб розрізняти два типи.animator/
color/	Файли XML, які визначають список кольорів стану. Щоб отримати додаткові відомості, треба переглянути ресурс списку стану кольору.

drawable/	Растрові файли (PNG, .9.png, JPG або GIF) або XML-файли, скомпільовані в такі підтипи ресурсів, які можна малювати: • Растрові файли, растрові зображення змінного розміру • Анімації • Інші малюнки
mipmap/	Файли для малювання для різної щільності значків панелі запуску.
layout/	Файли XML, які визначають макет інтерфейсу користувача.
menu/	Файли XML, які визначають меню програми, наприклад меню параметрів, контекстне меню або підменю.

Продовження таблиці 2.2

raw/	Довільні файли для збереження в необробленому вигляді. Якщо потрібен доступ до оригінальних імен файлів та ієрархії файлів, треба використовувати збереження ресурсів у каталозі assets/ замість res/raw/. Файли в assets/ не мають ідентифікатора ресурсу, тому ви читати їх можливо лише за допомогою AssetManager.
values/	XML-файли, які містять прості значення, такі як рядки, цілі числа та кольори. Файли в каталозі values/ описують кілька ресурсів. Деякі правила імен файлів для ресурсів, які можна створити в цьому каталозі: • arrays.xml для масивів ресурсів • colors.xml для значень кольору • dimens.xml для значень Dimension • strings.xml для рядкових значень • styles.xml для стилів
xml/	Довільні файли XML, які можна прочитати під час виконання, викликавши.
font/	Файли шрифтів із такими розширеннями, як TTF, OTF або TTC, або файли XML, які містять елемент <font-family>.

Ресурси, зібрано в підкаталогах, визначених у таблиці 2.2, є ресурсами за замовчуванням. Тобто ці ресурси визначають дизайн і вміст за замовчуванням для вашої програми. Однак різні типи пристроїв на базі Android можуть потребувати різних типів ресурсів.

Наприклад, ви можете надати різні ресурси макета для пристроїв, які мають більші за звичайні екрани, щоб скористатися перевагами додаткового простору на екрані. Ви також можете надати різні рядкові ресурси, які перекладатимуть текст у вашому інтерфейсі користувача на основі мовних

налаштувань пристрою. Щоб надати ці різні ресурси для різних конфігурацій пристроїв, вам потрібно надати альтернативні ресурси на додаток до ресурсів за замовчуванням.

3. РОЗРОБКА МОБІЛЬНОГО-ДОДАТКУ ДЛЯ РОБОТИ ЗІ СПАЛАХОМ І СЕНСОРАМИ

3.1 Основні компоненти додатку

Розроблений додаток містить сім класів на мові Java:

About – виводить інформаційну сторінку;

Accelerometer – відповідає за роботу з датчиком положення;

Compass – відповідає за роботу з датчиком магнітного поля і інтерфейс компаса;

FlashLight відповідає за роботу зі спалахом камери, увімкнення та вимикання ліхтарика;

Gyroscope – відповідає за роботу з датчиком гіроскопа;

Orientatyon – відповідає за отримання показників положення пристрою і зміну кольорів екрану при нахилах і поворотах пристрою;

MainActivity – організує вибір необхідній активності.

Приклад інтерфейсу активності MainActivity наведено на рис. 3.1.

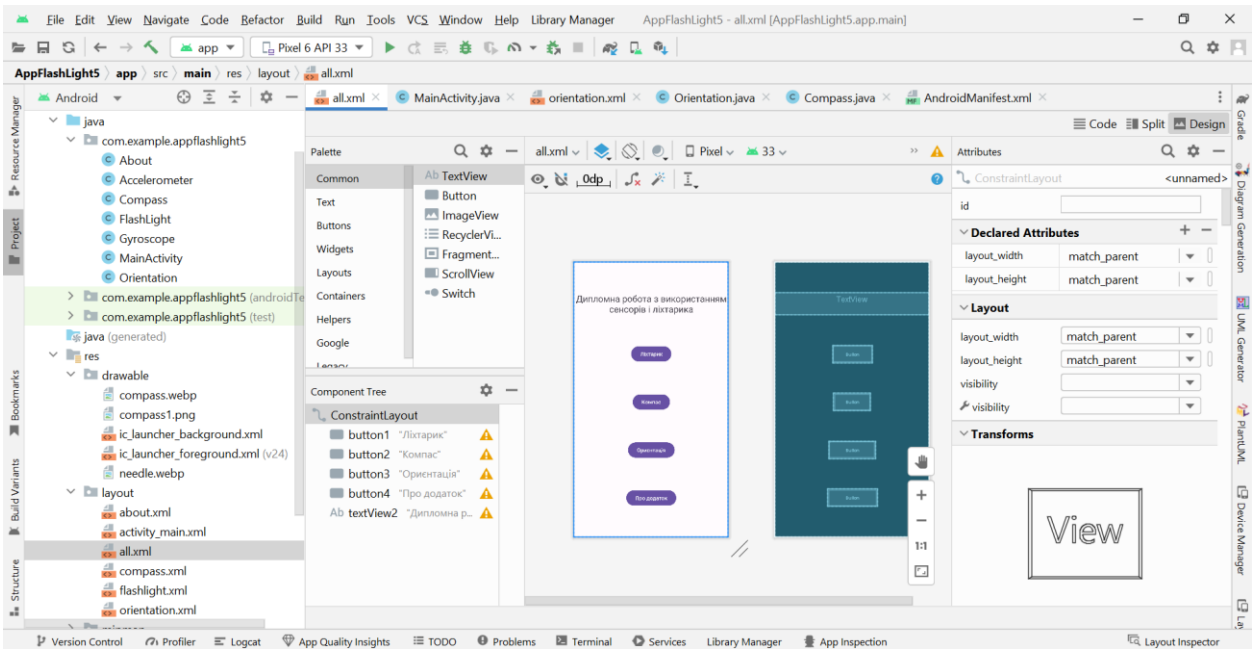


Рис. 3.1 – Розробка інтерфейса користувача MainActivity

Основний екран містить чотири кнопки, натискання на кожну з кнопок призводить до виклику відповідній активності.

Кожна дочірня активність містить кнопку повертання до головного вікна (приклад розробки інтерфейсу ліхтарика див. рис. 3.2).

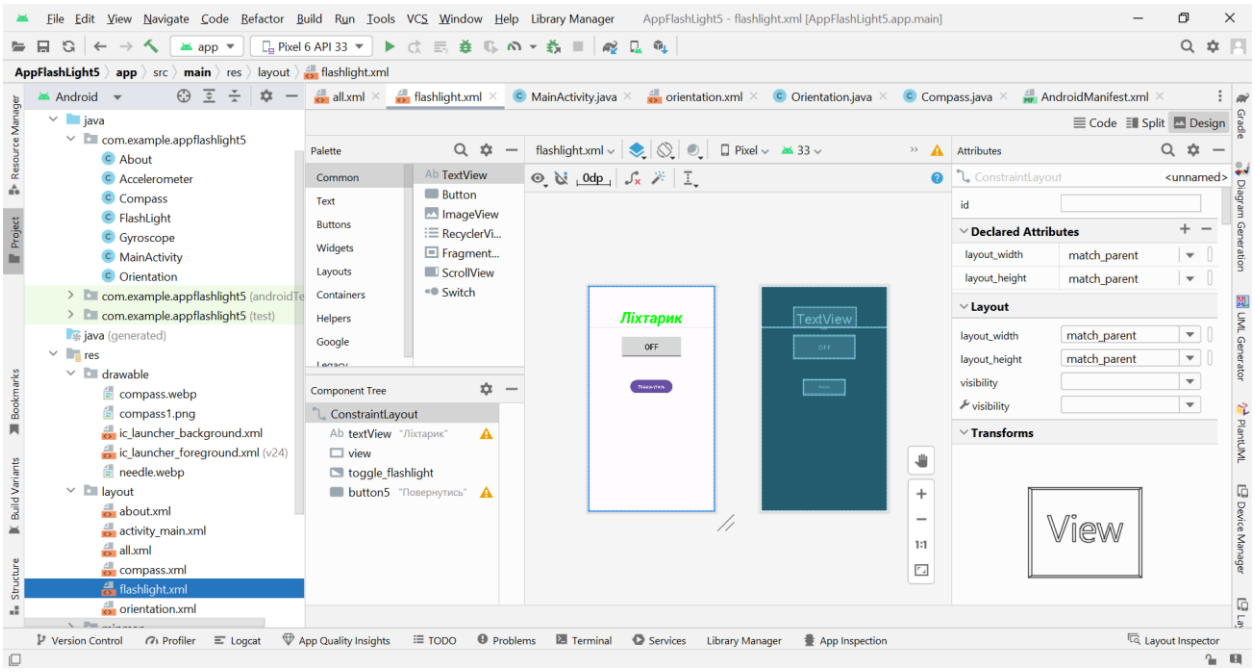


Рис. 3.2 – Приклад розробка інтерфейсу ліхтарика активності для FlashLight

Кожна із кнопок повернення у дочірніх активностях містить блок коду виклику відповідного інтенту. Приклад коду виклика MainActivity для ліхтарика наведено нижче:

```
button5.setOnClickListener(new View.OnClickListener()  {
        @Override
        public void onClick(View view) {
            Intent intent=new Intent(FlashLight.this, MainActivity.class);
            startActivity(intent);
        }
    });
```

За аналогічною схемою працюють інші активности. Головна активність MainActivity звертається до інтентів виклику дочірніх активностей (частина коду наведена нижче (метод onClick() – це обробник натискань на кнопки):

```
public void onClick(View v) {
    if (v.getId()==R.id.button1) {
       //Создаем переход:
       Intent intent=new Intent(MainActivity.this, FlashLight.class);
       //Запускаем его при нажатии:
       startActivity(intent);
    }
    else if (v.getId()==R.id.button2) {
       Intent intent=new Intent(MainActivity.this, Compass.class);
       startActivity(intent);
    }
    else if (v.getId()==R.id.button3) {
       Intent intent=new Intent(MainActivity.this, Orientation.class);
       startActivity(intent);
    }
    else if (v.getId()==R.id.button4) {
       Intent intent=new Intent(MainActivity.this, About.class);
       startActivity(intent);
    }
```

3.2 Об'єктно-орієнтована модель додатку

Об’єктно-орієнтовна модель компонентів додатку була розроблена за допомогою плагіну UML Generator (див. приклад на рис. 3.3).

Для основних активностей було згенеровано діаграми класів, які наочно ілюструють структуру і зв’язки компонентів додатку. Діаграма класів для основної активності MainActivity наведена на рис. 3.4. Цей клас є нащадком базового класу AppCompatActivity і реалізує інтерфейс обробки подій натискання на кнопки View.OnClickListener. Діаграма класів активності About наведена на рис. 3.5. Діаграма класів активностей, які пов’язані з орієнтацією пристрою, наведена на рис. 3.6.

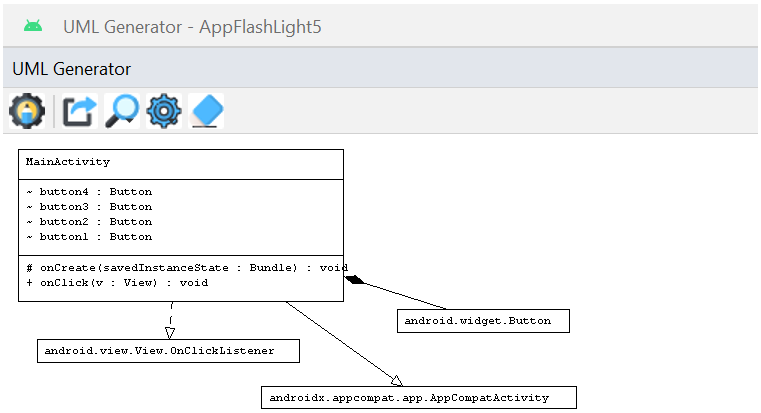


Рис. 3.3 – Генерація діаграм класів для компонентів додатку

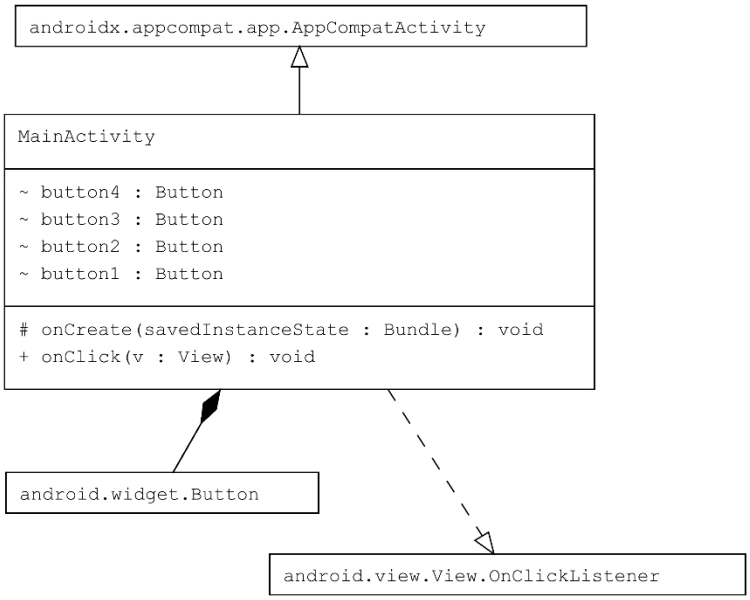


Рис. 3.4 –Діаграм класів для основної активності MainActivity

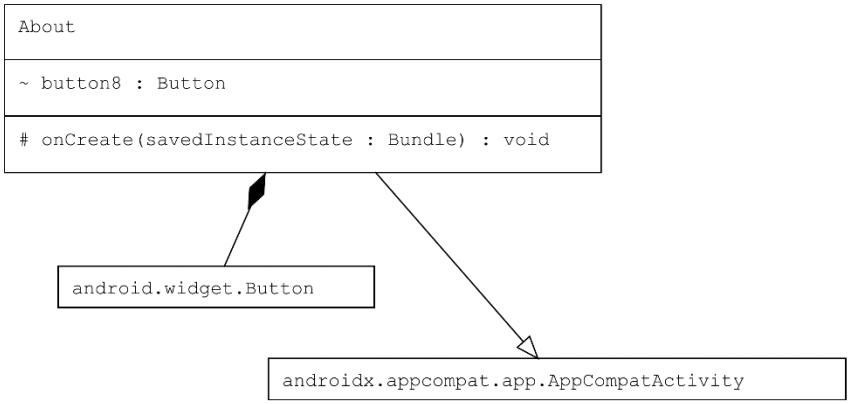


Рис. 3.5 –Діаграм класів для активності About

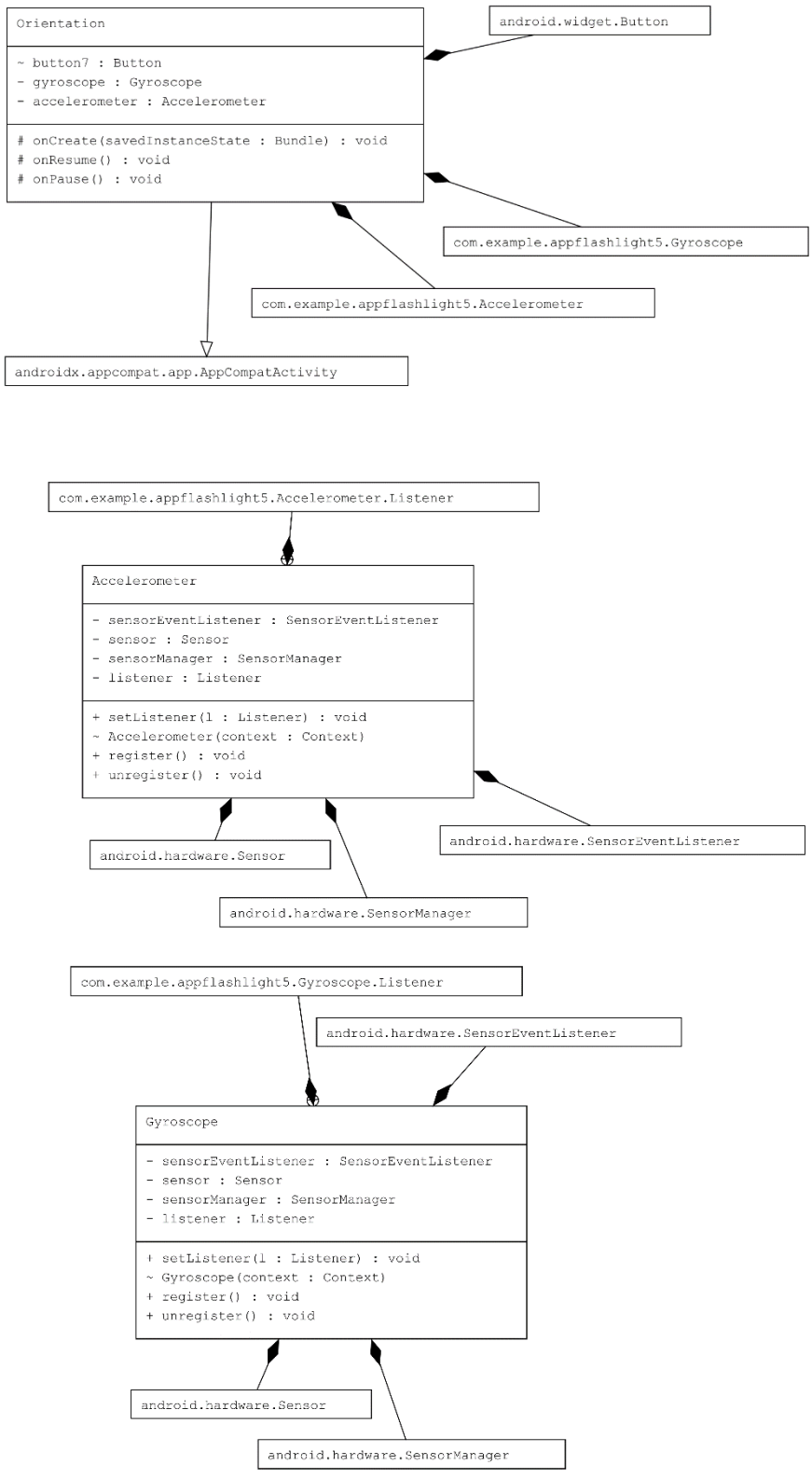


Рис. 3.6 –Діаграм класів для активностей Orientation, Accelerometer, Gyroscope

Діаграма класів активностей, які пов’язані з датчиками компаса і ліхтарика, наведена на рис. 3.7.

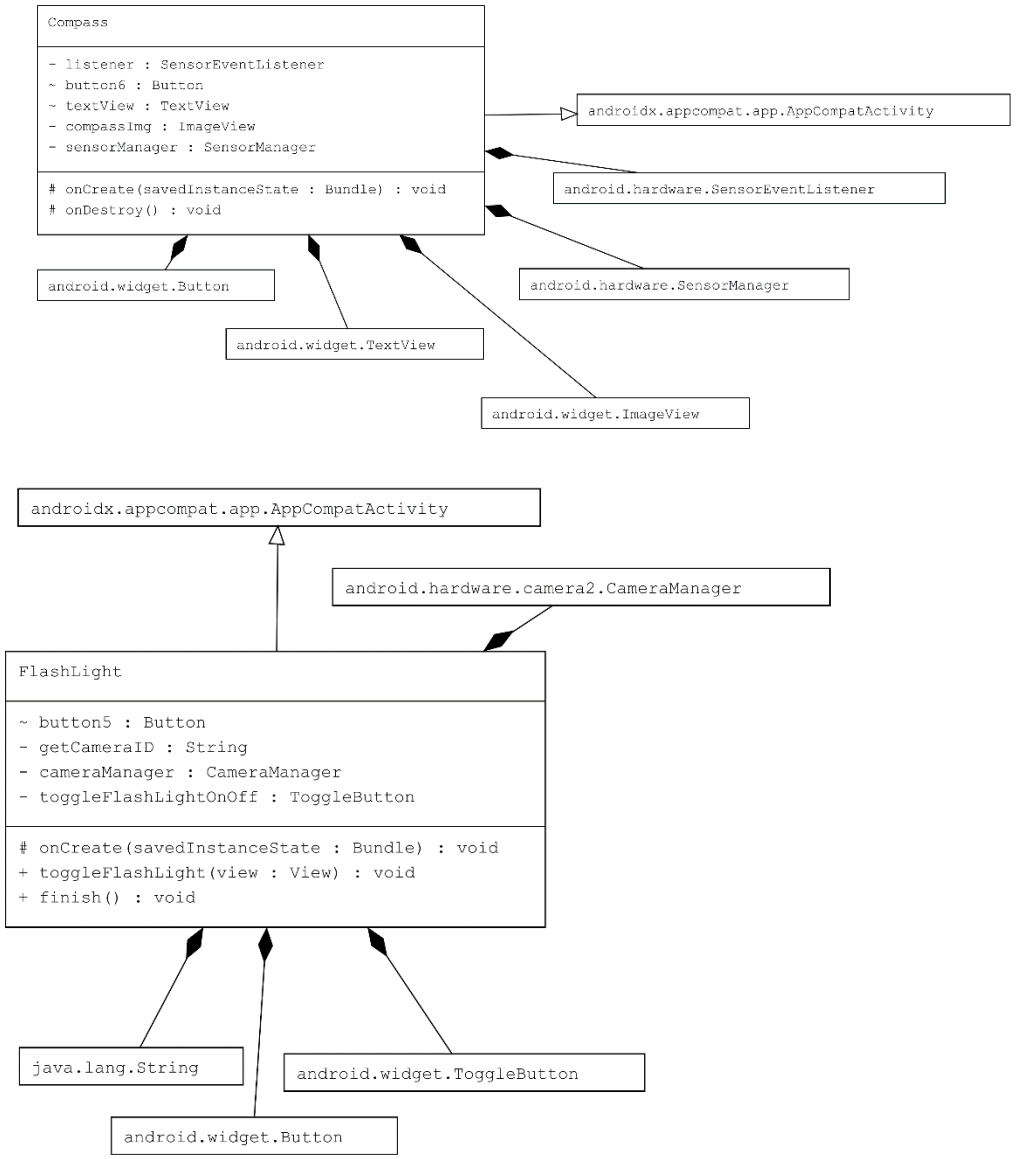


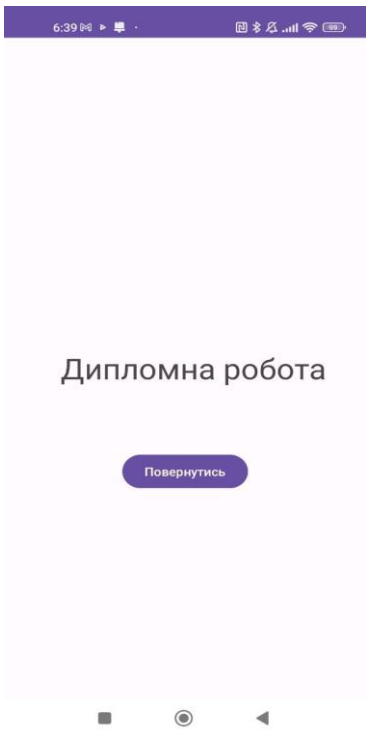
Рис. 3.7 –Діаграми класів для активностей FlashLight і Compass

Як видно з рис. 3.6 і 3.7, класи, які мають візуальний інтерфейс, є нащадками базового класа AppCompatActivity. Робота з сенсорами портібує звертання до SensorManager, робота зі спалахлм – звертання до класу CameraManager.

3.3 Інтерфейс користувача

Робота додатку і реалізація інтерфейсу користувача наведена на рис. 3.8-3.10.

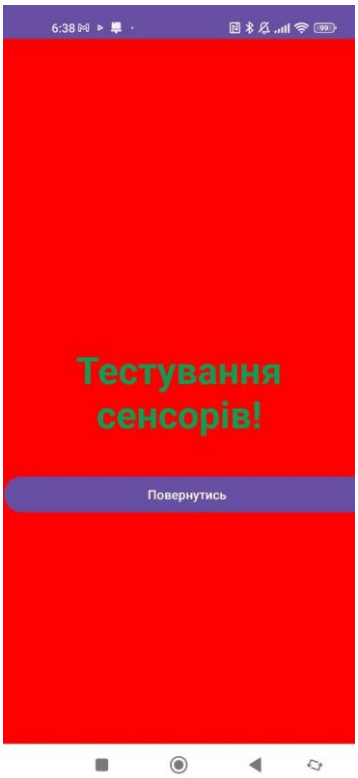
а) головне меню



б) екран About

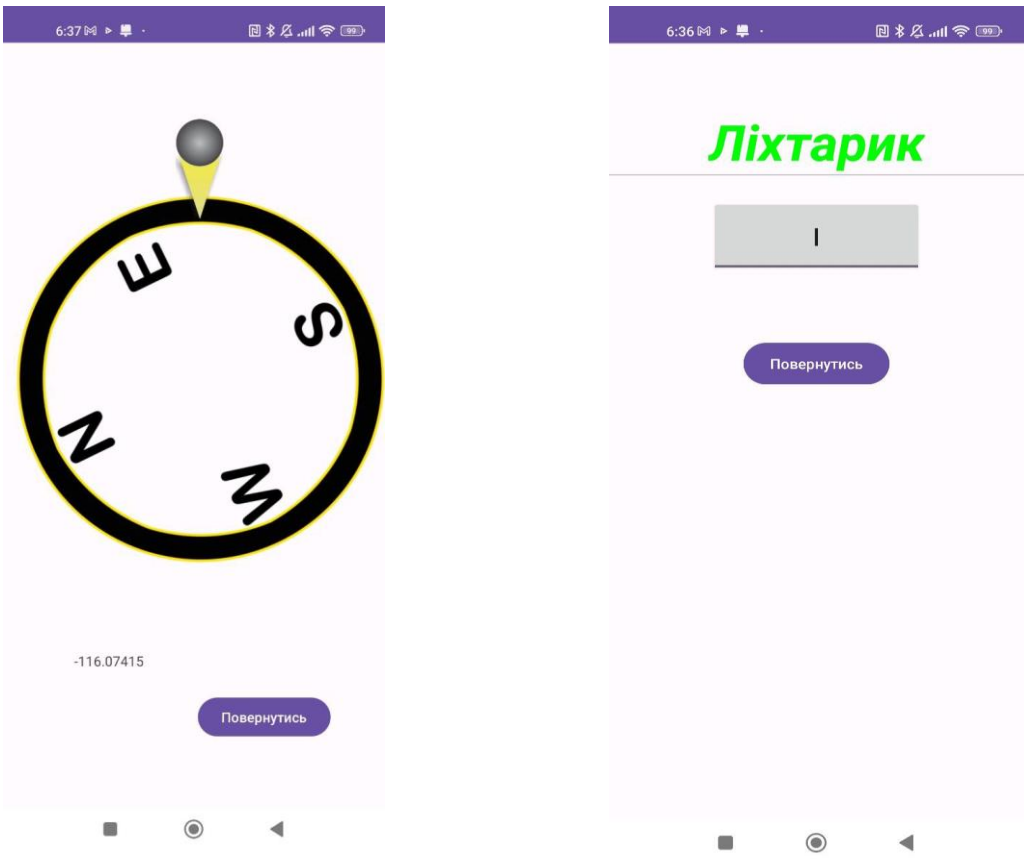
Рис. 3.8 – Скріншоти головного меню і екрану About додатку

а) нормальне положення



б) екран нахилено

Рис. 3.9 – Скріншоти роботи компонента контролю орієнтації пристрою



а) робота компаса б) екран увимкнення/вимикання ліхтарика

Рис. 3.10 – Скріншоти роботи компаса і ліхтарика

Як видно з рис. 3.9, зміна кута нахилу екрана або обертання екрана призводять до зміни його кольору. При відсутності світла можливий контроль положення пристрою і легке підсвічування за допомогою яскравого кольорового екрану.

Більш потужний ліхтарик працює за умовами вмикання світлодіодів спалаху в якості ліхтарика за допомогою відповідній кнопки (рис. 3.10 б).

Роботу компаса ілюструє рис. 3.10 (а). Обертання зображення компасу зроблено за допомогою механізму анімації в Android.

ВИСНОВКИ

Мета даного дипломного проекту – розробка Android-додатку FlashLightComplex мовою програмування Java.

В ході роботи був проведений детальний аналіз існуючих альтернативних рішень проблеми роботи з сенсорами пристроїв Android та інструментів розробки програм і визначено, що доцільно використовувати існуюче середовище розробки Android Studio і можливості Android SDK.

Середовище розробки і обрана мова програмування є крос-платформними і можуть використовуватись на різних платформах – Windows або Linux незалежно від дистрибутива.

Елементи графічного інтерфейсу користувача були реалізовані за допомогою мови розмітки XML і вбудованого набору віджетів Android.

В розробленому додатку створено наступний перелік функцій:

- Потужний ліхтарик з використанням світлодіодів спалаху;
- Легкий ліхтарик зі зміною кольору екрану в залежності від орієнтації екрану;
- Компас з використанням датчика магнітного поля.

Додаток, що розроблено, надає користувачам максимально зручне рішення для використання апаратних можливостей мобільного пристрою.

В проект додатку закладено можливість подальшої модифікації для покращення функціоналу та розширення з додаванням нових функцій.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Dawn Griffiths, David Griffiths. Head First Android Development: A Brain-Friendly Guide Paperback // O'Reilly Vlg. GmbH & Co., 2015, 734 p.
2. John Horton. Android Programming for Beginners // Packt Publishing, 2018. – 766 p.
3. John Horton. Android Programming with Kotlin for Beginners: Build Android apps starting from zero programming experience with the new Kotlin programming language// Packt Publishing, 2019. – 698 p.
4. Michael Burton. Android App Development FD // Packt Publishing, 2015. – 432 p.
5. Bert Bates, Kathy Sierra. Head First Java: Your Brain on Java - A Learner's Guide. // O'Reilly Vlg. GmbH & Co., 2003, 656 p.
6. Herbert Schildt. Java: A Beginner's Guide // McGraw Hill, 2018. – 720 p.
7. Barry A. Burd. Java For Dummies // : John Wiley&Sons, Inc., 2015. – 405 p.
8. Cay S. Horstmann. Core Java Fundamentals: Fundamentals // Prentice Hall, 2018. – 889 p.
9. J. Bloch. Effective Java // Addison-Wesley Professional, 2017. – 412 p.
10. Modern Android Development. URL: https://developer.android.com/ https://medium.com/androiddevelopers

Додаток А

Код MainActivity.java

```
package com.example.appflashlight5;

import android.content.Context;
import android.content.Intent;
import android.hardware.camera2.CameraAccessException;
import android.hardware.camera2.CameraManager;
import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import android.widget.ToggleButton;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity implements
View.OnClickListener{

    Button button1, button2, button3, button4, button_exit;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.all);

        // Register the ToggleButton with specific ID
        button1 = findViewById(R.id.button1);
        button2 = findViewById(R.id.button2);
        button3 = findViewById(R.id.button3);
        button4 = findViewById(R.id.button4);
        button_exit = findViewById(R.id.button_exit);
        button1.setOnClickListener(this);
        button2.setOnClickListener(this);
        button3.setOnClickListener(this);
        button4.setOnClickListener(this);
        button_exit.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        if (v.getId()==R.id.button1) {
            //Создаем переход:
            Intent intent=new Intent(MainActivity.this, FlashLight.class);
            //Запускаем его при нажатии:
            startActivity(intent);
        }
        else if (v.getId()==R.id.button2) {
            Intent intent=new Intent(MainActivity.this, Compass.class);
            startActivity(intent);
        }
        else if (v.getId()==R.id.button3) {
            Intent intent=new Intent(MainActivity.this, Orientation.class);
            startActivity(intent);
        }
        else if (v.getId()==R.id.button4) {
            Intent intent=new Intent(MainActivity.this, About.class);
```

```
            startActivity(intent);
        }
        else if (v.getId()==R.id.button_exit) {
            //Log.d("TestAppClose", "Button Clicked");
            // Завершити роботу цього вікна
            this.finish();
            // on below line we are exiting our activity
            System.exit(0);
        }
    }
}
```

Додаток Б

Код файлу розмітки all.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Ліхтарик"
        app:layout_constraintBottom_toTopOf="@+id/button2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Компас"
        app:layout_constraintBottom_toTopOf="@+id/button3"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button1" />

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Орієнтація"
        app:layout_constraintBottom_toTopOf="@+id/button4"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button2" />

    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Про додаток"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button3" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Дипломна робота з використанням сенсорів і ліхтарика"
```

```
        android:textAlignment="center"
        android:textSize="24sp"
        app:layout_constraintBottom_toTopOf="@+id/button1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button_exit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="30dp"
        android:layout_marginBottom="16dp"
        android:text="Завершити"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```