МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Комп'ютерних наук, управління та адміністрування

Кафедра Інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: Розробка мобільного застосунку для моніторингу екологічного стану на платформі андроїд

Виконав студент 2 курсу групи МІС-20
спеціальності 122 Комп'ютерні науки
Кириловський Олександр Олександрович

Керівник д.т.н., професор
Казакова Надія Феліксівна

Рецензент регіональний координатор програми «EGAP» Копиченко І.Ю.

Одеса 2021

АНОТАЦІЯ

на магістерську кваліфікаційну роботу
«Розробка мобільного застосунку для моніторингу екологічного стану на платформі андроїд»,
студента Кириловського Олександра Олександровича

Актуальність вирішення проблем моніторингових досліджень полягають в тому, що хоча й існує низка відомчих спостережень систем за станом довкілля, але вони не зведені в єдиний комплекс і не можуть ефективно виконувати узагальнюючу функцію оцінки стану і рівня використання ресурсів.

Мета роботи – розробка мобільного додатку на платформі андроїд для моніторингу екологічного стану.

Об’єкт дослідження – мобільні застосунки, призначені для моніторингу екологічного стану.

Методи дослідження – теоретичне ознайомлення з існуючими мобільними додатками, призначеними для моніторингу екологічного стану, моделювання макету дизайну застосунку, методи об’єктно-орієнтованого програмування.

Проведено аналіз мобільних застосунків з подібним функціоналом, здійснено порівняльну характеристику, виявлено переваги та недоліки.

Магістерська кваліфікаційна робота містить 70 сторінок, 43 рисунки та 25 джерел.

Ключові слова: ANDROID, МОБІЛЬНИЙ ДОДАТОК, XML, JAVA, ANDROID STUDIO, GRADLE.

SUMMARY

for a master's degree

«Development of a Mobile Application on the Android Platform for Monitoring of the Environmental Situation», students of Oleksandr Kyrylovskyi

The relevance of solving the problems of monitoring research is that although there are a number of departmental observation systems for to the state of the environment, but they are not integrated into a single complex and can not effectively perform a generalizing function of assessing the state and level of resource use.

The purpose of the work is development of a mobile application on the Android platform for monitoring of the environmental situation.

The object of research is mobile applications designed for monitoring of the environmental situation.

Research methods – theoretical acquaintance with the existing mobile applications designed for monitoring of the environmental situation, modeling of the application design layout, methods of object-oriented programming.

The analysis of mobile applications with similar functionality is carried out, the comparative characteristic is carried out, advantages and lacks are revealed.

The master's thesis contains 70 pages, 43 figures and 25 sources.

Keywords: ANDROID, MOBILE APP, XML, JAVA, ANDROID STUDIO, GRADLE.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

AQI (Air Quality Index) – індекс якості повітря

IDE (Integrated Development Environment) – інтегроване середовище розробки

SDK (Software Development Kit) – набір засобів для розробки програмного забезпечення

JVM (Java Virtual Machine) – віртуальна машина Java

JIT (Just-in-Time) – динамічна компіляція

API (Application Programming Interface) – інтерфейс програмування

XML (Extensible Markup Language) – розширювана мова розмітки

URF (Uniform Resource Identifier) – уніфікований ідентифікатор ресурсу

ПЗ – програмне забезпечення

TCP/IP – набір протоколів мережі інтернет

JDK (Java Development Kit) – набір засобів для розробки програм на мові Java

DSL (Domain Specific Programing Language) – предметно-орієнтована мова

AVD (Android Virtual Device) – емулятор, віртуальний девайс андроїд

CVS (Concurrent Versions System) – система одночасних версій

Git – система управління версіями з розподіленою архітектурою

SWT (Standart Widget Toolkit) – бібліотека для розробки графічних інтерфейсів користувача на мові Java

GUI (Graphical user interface) – тип інтерфейсу, що призначений для взаємодії з електронними пристроями через графічні зображення

JSON (JavaScript Object Notation) – текстовий формат обміну даними

APK (Android Package) – формат архівних файлів-додатків для «Android»

UML (Unified Modeling Language) – уніфікована мова моделювання

ВСТУП

Для кожної людини сфера екології є найбільш чутливою. Вона безпосередньо впливає на наше здоров’я та якість життя.

З погіршенням глобального екологічного стану зусилля спільноти все більше спрямовано на покращення та збереження екологічної ситуації. Зокрема, значна увага приділяється моніторингу стану довкілля.

Під моніторингом навколишнього середовища розуміють комплексну систему спостережень, прогноз та оцінка під впливом антропогенних факторів. Зараз моніторинг визначають як сукупність спостережень за визначеними компонентами біосфери та комплекс методів екологічного прогнозування.

Промислові підприємства є головними забруднювачами. Збільшення шкідливих викидів через велику кількість автомобілів також має значний вплив[1][1)].

Сервісів, що надають дані про забруднення у відкритому доступі, стає все більше, що дає змогу провести оцінку більш детально. Основна задача мережі спостережень для моніторингу за станом довкілля полягає в екологічному районуванні території, яке дозволяє виявити і оцінити фактори, що впливають на розповсюдження забруднюючих речовин.

Система моніторингу антропогенних змін природного середовища не є новою. Вона не потребує організації мережі нових станцій спостереження, ліній та телекомунікації, центрів обробки даних та та. ін.

Основною метою даної роботи є аналіз існуючих технологій та методів спостереження, а також розробка мобільного додатку на платформі андроїд для моніторингу стану навколишнього середовища, який може значно спростити інформування пересічних громадян про стан забруднення.

[1)][1]Особливості екологічного та біологічного моніторингу. URL: https://works.doklad.ru/view/3NelM8XFMXs/2.html (дата звернення 12.08.2021).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У даному розділі будуть розглянуті схожі додатки, а також буде проведено аналіз із визначення переваг та недоліків.

Актуальність і невідкладність вирішення проблем моніторингових досліджень полягають в тому, що хоча й існує низка відомчих спостережень систем за станом довкілля, але вони не зведені в єдиний комплекс і не можуть ефективно виконувати узагальнюючу функцію оцінки стану і рівня використання ресурсів.

Готовий програмний продукт має забезпечувати наступні функціональні можливості:

− відображати індекс якості повітря та прогноз погоди за різні періоди часу;

− відображати дані про забруднюючі гази;

− креслити графік, який показує зміни погоди та забруднення протягом часу;

− відображати дані на карті.

Державною гідрометеорологічною службою здійснюються спостереження за забрудненням атмосферного повітря у 53 містах України

Ведуться спостереження за хімічним складом атмосферних опадів та за кислотністю опадів.

Деякі станції здійснюють спостереження за додатковими забруднюючими речовинами. Проводиться аналіз наявності забруднюючих речовин в опадах та сніговому покриві.

Державна екологічна інспекція здійснює вибірковий відбір проб на джерелах викидів[2][1)].

[1)][2] Экологический мониторинг в Украине: какие данные открыты. URL: https://www.epravda.com.ua/rus/columns/2018/07/17/638718/ (дата звернення 12.08.2020)

1.1. Аналіз ринку подібних проектів

Серед існуючого програмного забезпечення, спрямованого на моніторинг екологічних проблем є програми IQAir AirVisual[3][1)], BreezoMeter[4][2)], AirQuality-AirCare[5][3)], AirMatters[6][4)].

1.1.1 Огляд застосування IQAir AirVisual

IQAir AirVisual – це мобільний додаток, орієнтований на прогноз погоди та моніторинг забруднення в режимі реального часу.

Додаток показує дані для більш ніж 10000 міст та 80 країн. В додатку присутні рекомендації по охороні здоров'я та відповідна інформація для чутливих груп.

Особливістю є моніторинг 6 основних забруднювачів та відображення історичних даних на графіках.

На рис. 1.1 наведений загальний вигляд інтерфейсу застосування.

[1)][3] Приложения в Google Play – IQAir Airvisual | Air Quality. URL: https://play.google.com/store/apps/details?id=com.airvisual (дата звернення 15.08.2021).

[2)][4] Приложения в Google Play – Индекс качества воздуха и пыльца – BreezoMeter. URL: https://play.google.com/store/apps/details?id=app.breezometer (дата звернення 15.08.2021).

[3)][5] Приложения в Google Play – Air Quality – AirCare. URL: https://play.google.com/store/apps/details?id=com.gorjan.airquality (дата звернення 15.08.2021).

[4)][6] Приложения в Google Play – Air Matters. URL: https://play.google.com/store/apps/details?id=com.freshideas.airindex (дата звернення 15.08.2021).

Рисунок 1.1 – Головна сторінка додатку

Після переходу на карті відображаються дані AQI та інші показники (див. рис. 1.2).

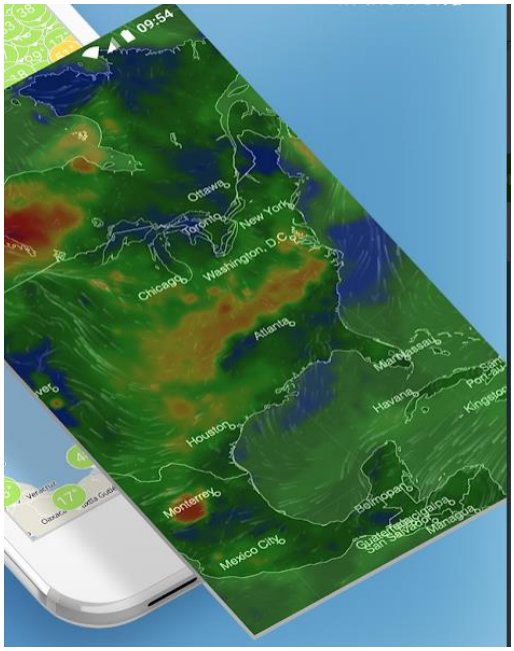

Рисунок 1.2 – Карта додатку

На рис. 1.3 зображено інформацію про кількість різних речовин у повітрі. Також є графік їх погодинної та щоденної кількості в різних кольорах відповідно до рівня безпеки.

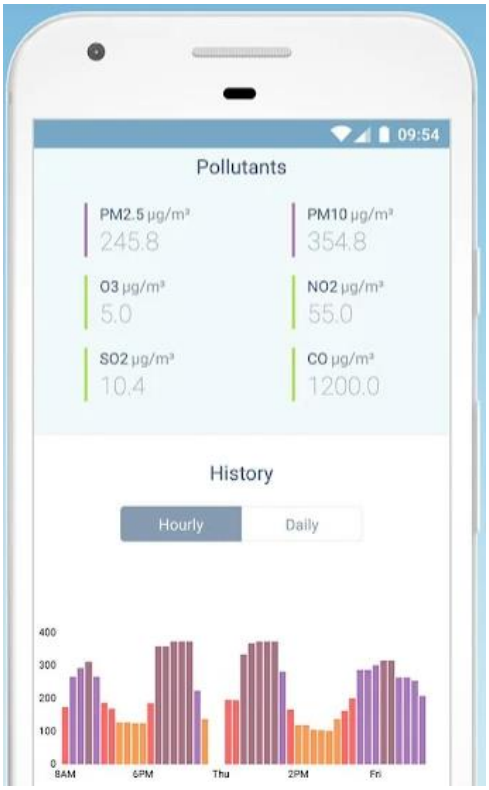

Рисунок 1.3 – Графік речовин

На жаль, дані не завжди відображаються коректно.

1.1.2 Огляд застосування BreezoMeter

BreezoMeter призначений для відстеження якості повітря поряд з вами та зменшення впливу забрудненості.

У додатку присутні рекомендації для здоров’я на основі AQI. А також можливо дізнатися місцевий індекс якості та його проноз (доступний для 94 країн) та побачити на глобальній карті.

На рис. 1.4 наведено головну сторінку додатка з показниками забруднення, їх рівнями у числових та кольорових значеннях. Також присутня інформація про пожежі.

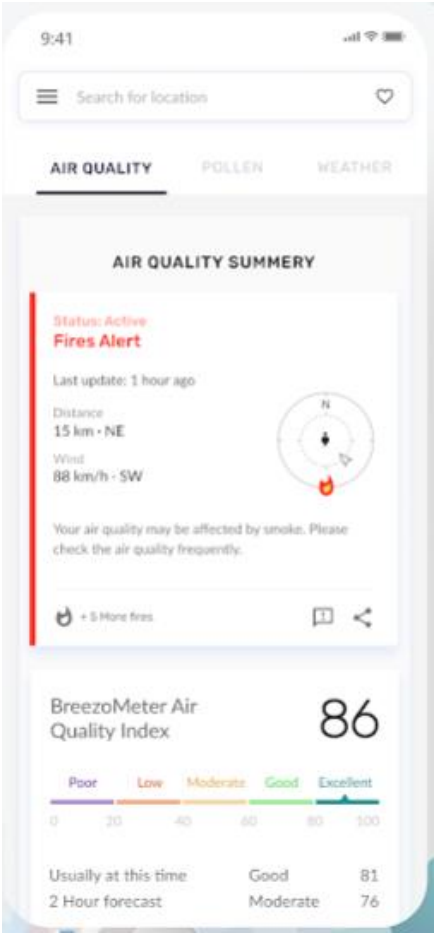

Рисунок 1.4 – Головний екран

Також на цій сторінці можна переглянути прогноз забруднення на 5 годин (див. рис. 1.5).

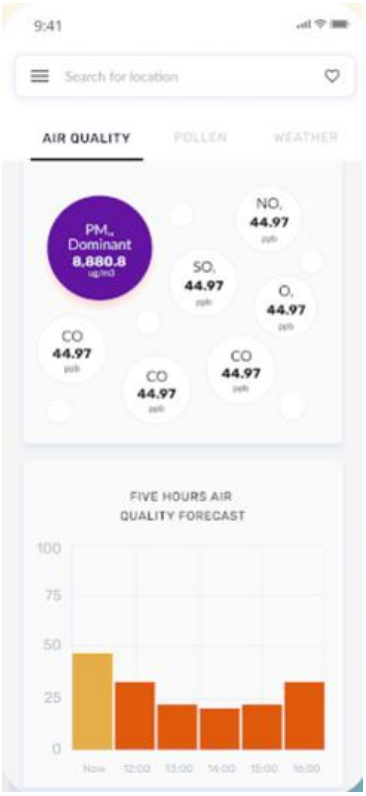

Рисунок 1.5 – Прогноз на 5 годин

На рис. 1.6 показано рівень AQI.

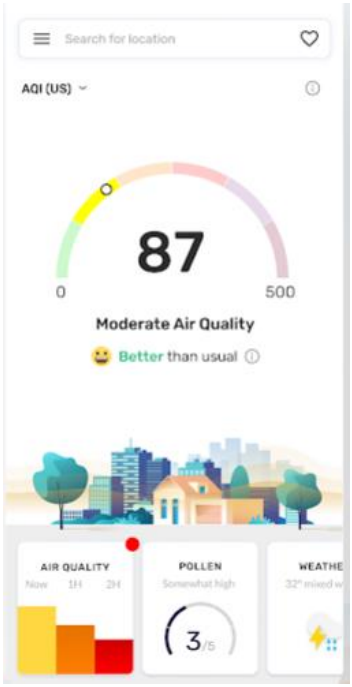

Рисунок 1.6 – Рівень AQI

Сторінка з картою, на якій можна переглянути дані про потрібний вам населений пункт або місцезнаходження (див. рис. 1.7).

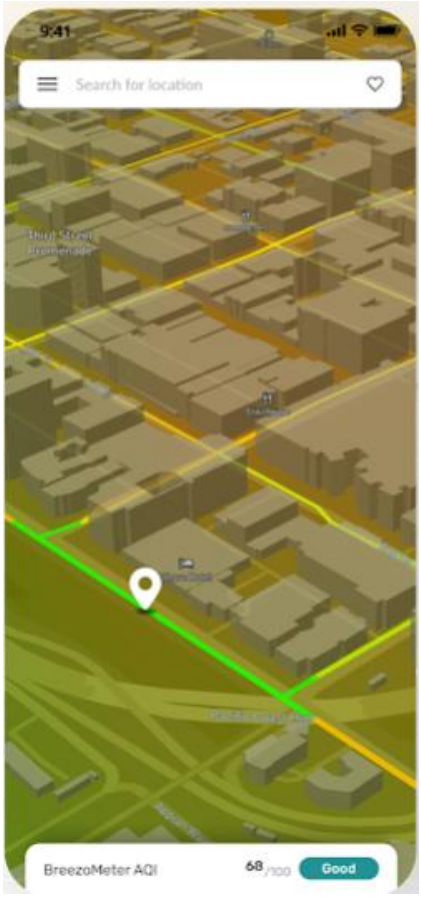

Рисунок 1.7 – Сторінка з картою

Додаток має деякі недоліки. Дані забруднень іноді показує неправильні, також виникають проблеми з роботою додатку.

1.1.3 Огляд застосування AirQuality - AirCare

Основною функцією додатку є представлення даних про якість повітря і такі забруднювачі як PM10, PM2,5, O3, CO, NO2, SO2 в режимі реального часу. Також користувачу представлені дані про пилок для окремих країн. Візуалізація даних у графіках, діаграмах та картах.

На головній сторінці (див. рис. 1.8) показано індекс якості повітря (AQI) та інші дані, що можна вибрати з випадаючого списку для вказаного населеного пункту або місцезнаходження.

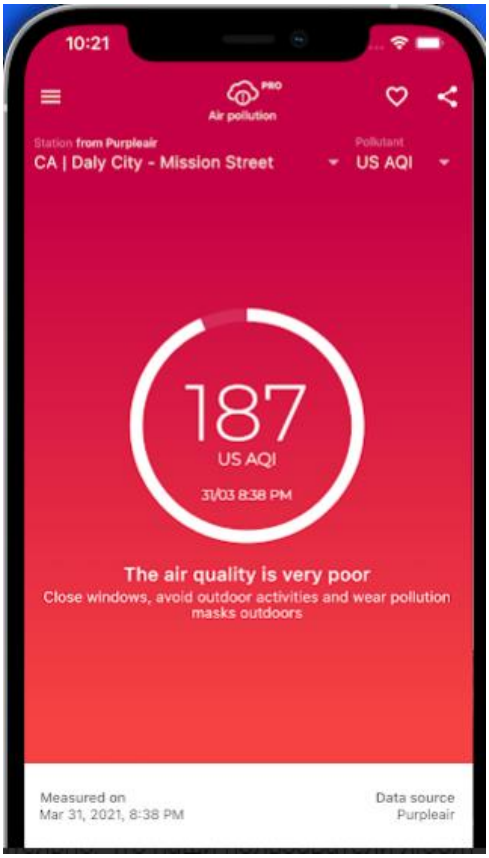

Рисунок 1.8 – Головна сторінка

Наступна сторінка призначена для показу різних слоїв карти. Серед них рівень забруднення , пожежі та вітер (див.рис.1.9).

Рисунок 1.9 – Вкладка з картою

На наступній сторінці побудовано гістограму рівня AQI протягом різного періоду часу (див.рис.1.9).

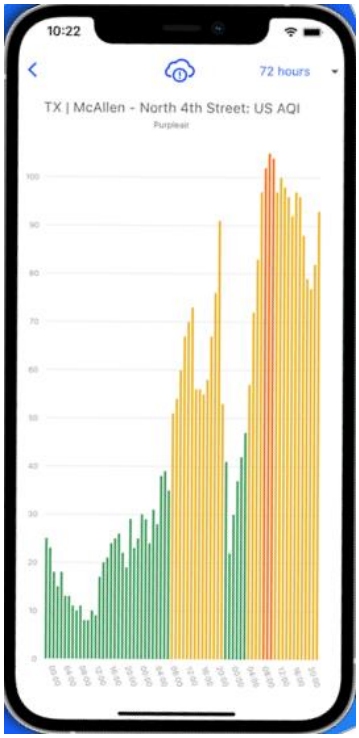

Рисунок 1.10 – Графік AQI

Недоліком цього додатку є підтримка невеликої кількості країн.

1.1.4 Огляд застосування Air Matters

Air Matters – це застосування, призначене для представлення користувачу інформації про якість повітря у більш ніж 180 країнах. Дані надходять з найближчих до вас станцій моніторингу.

Перемикається стандарт між США, Китаєм, Нідерландами, Європою, Великобританією та Індією.

На рис. 1.11 зображено головний екран, на якому показано AQI, домінуючий забруднювач у повітрі та його рівень.

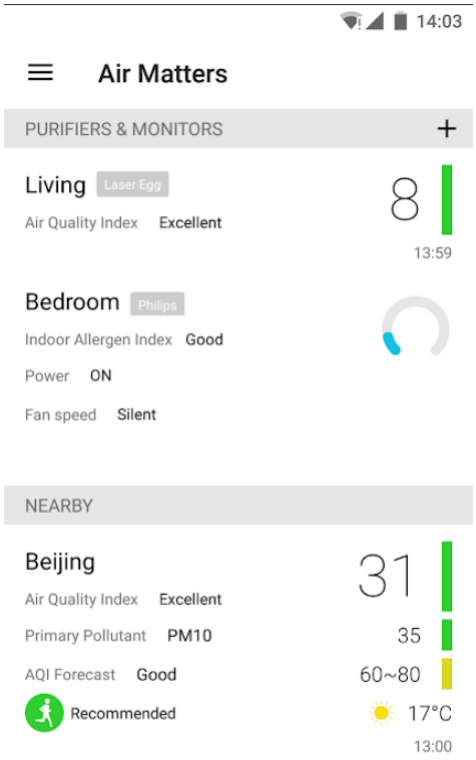

Рисунок 1.11 – Стартовий екран застосування

Також ви можете отримати всі необхідні дані про обраний населений пункт, такі як рівень AQI та скорочений прогноз погоди (див. рис. 1.12).

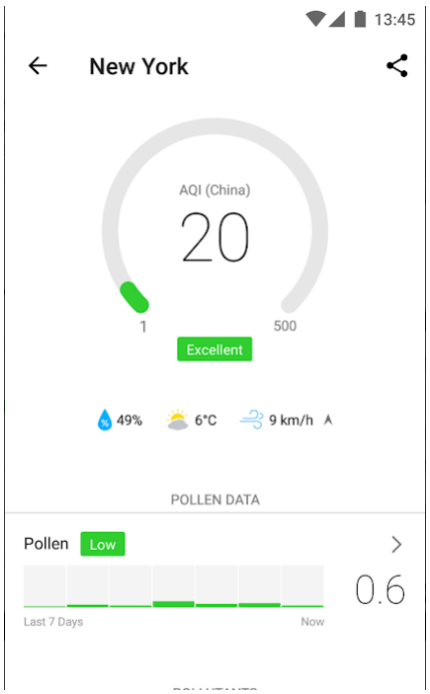

Рисунок 1.12 – Екран даних про обране місто

На наступній вкладці (див. рис. 1.13) представлено список міст з рівнем забруднення від найгіршого до найкращого та навпаки.

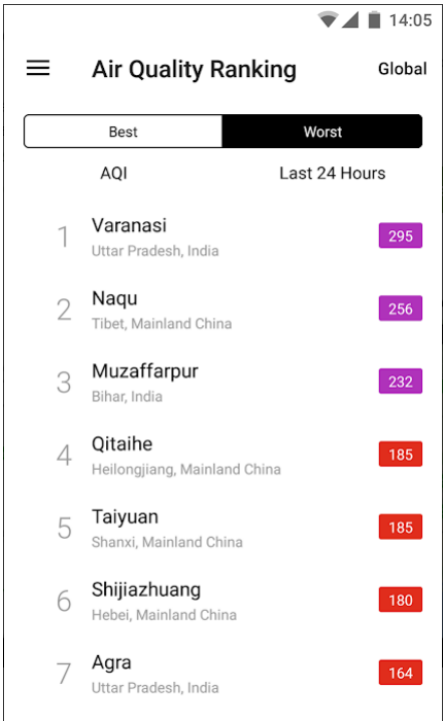

Рисунок 1.13 – Екран ранжування забруднення

На наступній вкладці (див. рис. 1.14) відображається карта із рівнем забруднення.

Рисунок 1.14 – Вкладка з картою

На останній сторінці розміщено глобальну карту з рівнем AQI. На жаль, в додатку не має змоги переглянути кількість інших основних забруднювачів.

2 ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНІЧНИХ ЗАСОБІВ

Вибір середовища розробки для програмування – один з основних етапів розробника. Інтегрована середа розробки (IDE), в якій ви пишете свій код, визначає не тільки його якість, але навіть структуру і стиль написання.

2.1 Платформа Android

Платформа «Android» є продуктом групи “Open Handset Alliance” , метою якої є створення досконалої мобільної системи. У 2008 році біло випущено перший пристрій з такою системою. Послідовні версії пакета розробки для програмного забезпечення (SDK) були єдиним інструментом для розробки. Відкритий код Android дозволяє розробникам змінювати та публікувати свої версії для вдосконалень.

Сервіси, що базуються на акселерометрах та визначенні місцезнаходження підтримуються стеком програмного забезпечення так само, як і сервіси для роботи з користувальницьким інтерфейсом. Мобільні пристрої поступалися настільним графікою та способом збереження даних. У Android проблему з графікою вирішено за допомогою вбудованої бібліотеки «OpenGL», а збереження даних стало робити простіше завдяки наявності SQLite в платформі.

За специфікою Android багаторівнева система в основі якої знаходиться Linux ядро з великим функціоналом. Додатки пишуться на мові програмування Java та Kotlin і виконуються в віртуальній машині. Замість Java Virtual Machine (JVM) роботу виконує «Dalvik Virtual Machine». Компанія Google вклала багато ресурсів в розвиток та ефективність. Віртуальна машина Dalvik була розроблена для дотримання продуктивності. Dalvik заснований на регістрі не підтримує JIT компіляцію, що відрізняє його від JVM. Кожен окремий додаток виконується в окремому екземплярі віртуальної машини Dalvik і пам’ять розподіляється між цими екземплярами для зменшення витрат. Використовуючи інший підхід для розробки мобільних додатків, ви можете досягти

високий рівень однорідності, завдяки чому більшість додатків без подальших змін буде працювати на будь-якому пристрої на базі Android.

Важливою особливістю, яка відрізняє Android від інших платформ є те, що Android офіційно підтримує фонові процеси в сторонніх додатках. Вони реалізовані за допомогою сервісного компонента Android. Сервіси – це операції без заголовка, які виконуються в фоновому режимі протягом тривалого часу. Нижче (див. рис. 2.1) наведено архітектуру системи.

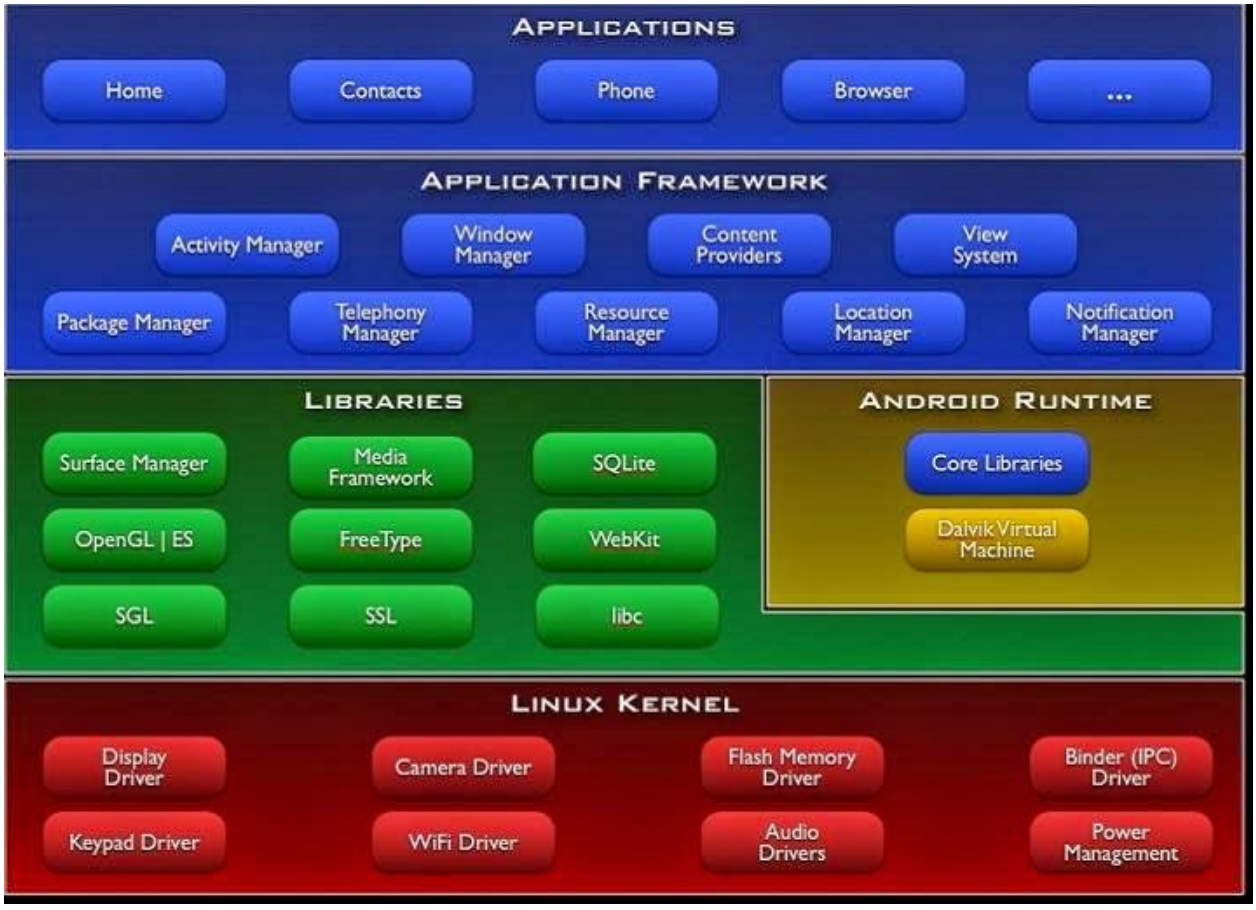

Рисунок 2.1 – Схема архітектури системи

Для створення додатків та використання основних функцій платформи Android надає великий асортимент високорівневих інтерфейсів програмування (API). Такі інтерфейси забезпечують високий рівень абстракції, завдяки чому їх легко можна використовувати у процесі розробки.

Сторонні додатки мають смогу працювати майже з усіма основними компонентами мобільної платформи. На рис. 2.2 зображено приклад використання API.

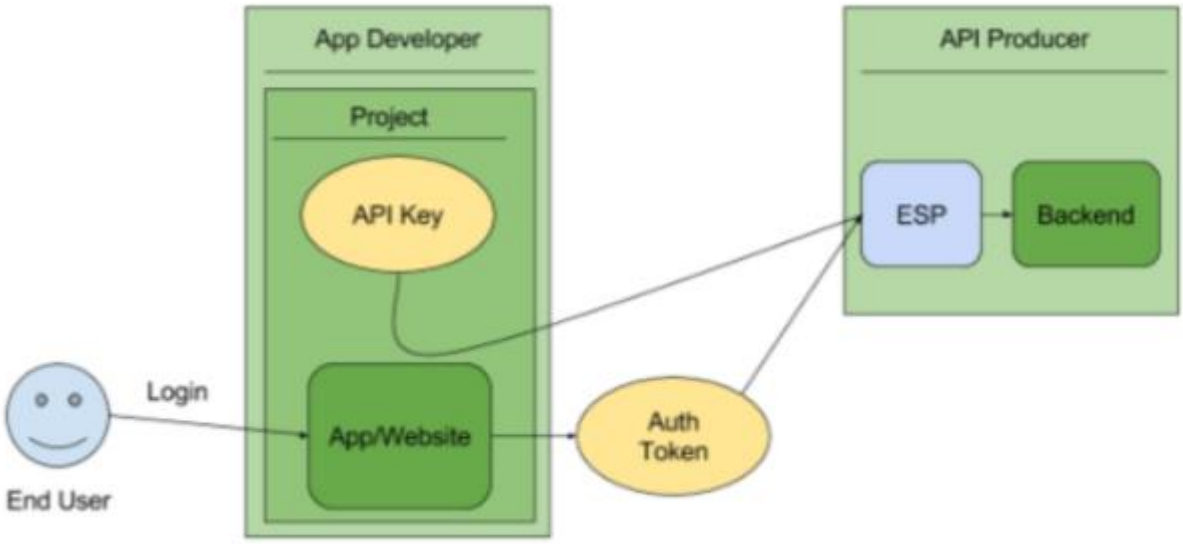

Рисунок 2.2 – Приклад використання API

Android надає широкий набір інструментів для побудови інтерфейсу. Для його опису використовується розширювана мова розмітки (XML). На окремі віджети можна посилатися по встановленому в макеті id. Віджети можна створювати не тільки за допомогою мови опису, а і програмно. Android SDK не завжди працює злагоджено, але дає розробнику змогу маніпулювати дизайном у редакторі.

Додатки складаються з активностей, приймачів, служб та провайдерів. Провайдери призначені для взаємодії з різними джерелами даних та їх обміну. Вони надають потрібну інформацію через стандартизований інтерфейс запитів. Синтаксисом URI (уніфікований ідентифікатор ресурсу) можна описувати запити, але зазвичай розробники використовують багаторівневі класи оболо-

нки, що генерують рядки запитів і керують ними. Провайдер контекста у відповідь на запит повертає об’єкт-курсор. Якщо вам потрібні повідомлення про зміну даних, ви можете підключити механізм моніторингу.

Система провайдера контенту пропонує розробникам Android додатків декілька суттєвих переваг:

– високий рівень взаємодії, що дозволяє проводити обмін даними уніфікованим способом;
– доступ за замовчування до основних джерел даних, такі як системи збереження мультимедіа, контакти та інше.

Важливою особливістю, Android від інших платформ є те, що офіційно підтримуються фонові процеси в сторонніх додатках. Їх реалізація відбувається за допомогою сервісного компоненту. Операції, що виконуються в фоновому режимі протягом тривалого часу.

Вбудовані широкомовні приймачі, що визначають коли з’являються системні повідомлення і у відповідь виконують певні дії, становлять собою частину системи повідомлень.

Взаємодія всіх компонентів регулюється об’єктами Intents, які містять ідентифікатор дії та URI даних. Найпоширеніший спосіб їх використання – це запуск нової активності. На рисунку 2.3 зображено загальну схему компонентів Android додатку.

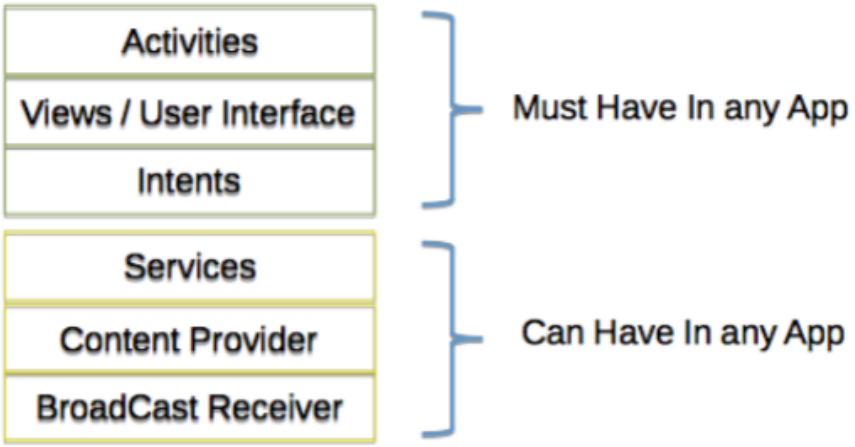

Рисунок 2.3 – Схема компонентів Android

Головною перевагою операційної системи Android є відкритий вихідний код. Кожен розробник може використовувати його для своїх цілей, покращувати або використовувати для поглибленого вивчання роботих[7][1].

2.2 Вибір мови програмування

При виборі мови програмування були розглянуті Java та Kotlin.

2.2.1 Огляд Java

Java – це об'єктно-орієнтована мова програмування, розроблена компанією «Sun Microsystems», яка зараз належить компанії «Oracle». Основним мотивом для створення Java була потреба в мові програмування, яку можна було б використовувати для створення програмного забезпечення (ПЗ), що вбудовується в різноманітні побутові електронні прилади.

Основою синтаксису послугували мови програмування C і C++. Особливості мови Java формувалися одночасно з появою мережі Інтернет. Саме тоді вона набула великої популярності серед розробників, адже з’явилися прикладні програми нового типу та було вирішено дві найбільші, на той час, проблеми програмування − безпека та кросплатформеність[8][2].

Мова Java призначена для розподіленого середовища Інтернету, бо підтримує мережеві протоколи TCP/IP. По суті, звернення до ресурсу по уніфікованому вказівнику інформаційного ресурсу (URL) мало чим відрізняється від звернення до файлу. Java також підтримує віддалений виклик методів програми, що дозволяє викликати їх через мережу. Програми містять значний об’єм даних динамічного типу для перевірки повноважень і дозволу доступу до об’єктів під час виконання програми. Це в свою чергу дозволяє безпечно та

[1] [7] Android – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Android (дата звернення 19.08.2021).

[2] [8] Java – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Java (дата звернення 21.08.2021).

раціонально виконувати динамічну зв’язку коду. Це дуже важливо для стійкості середовища Java, де невеликі фрагменти байт-коду можуть динамічно оновлюватися в діючій системі.

Основною особливістю є те, що при компіляції ми отримуємо набір інструкцій для виконання в JVM замість виконуваного коду. Такий підхід спростовує виконання на різних пристроях, адже на них потрібно тільки реалізувати віртуальну машину. Операції, що перевищують встановлені повноваження програми, негайно перериваються.

Компіляція байт-коду в машинозалежний виконується оперативно, що підвищує продуктивність, не зважаючи на те, що мова програмування Java інтерпретована. Згодом поява технології HotSpot надала динамічний компілятор, і якщо він присутній в JVM, то фрагменти байт-коду компілюються в виконуваний по частинах. Одночасна компіляція всієї програми у виконуваний код недоцільна, оскільки в Java проводяться перевірки у процесі виконання.

Динамічна компіляція під час виконання відбувається у разі виникнення необхідності. Фрагменти байт-коду компілюються за пріоритетом. Тож при такій компіляції характеристики безпеки та кросплатформеності зберігаються, оскільки JVM відповідає за цілісність виконуваного середовища. Процес компіляції зображено на рис. 2.4.

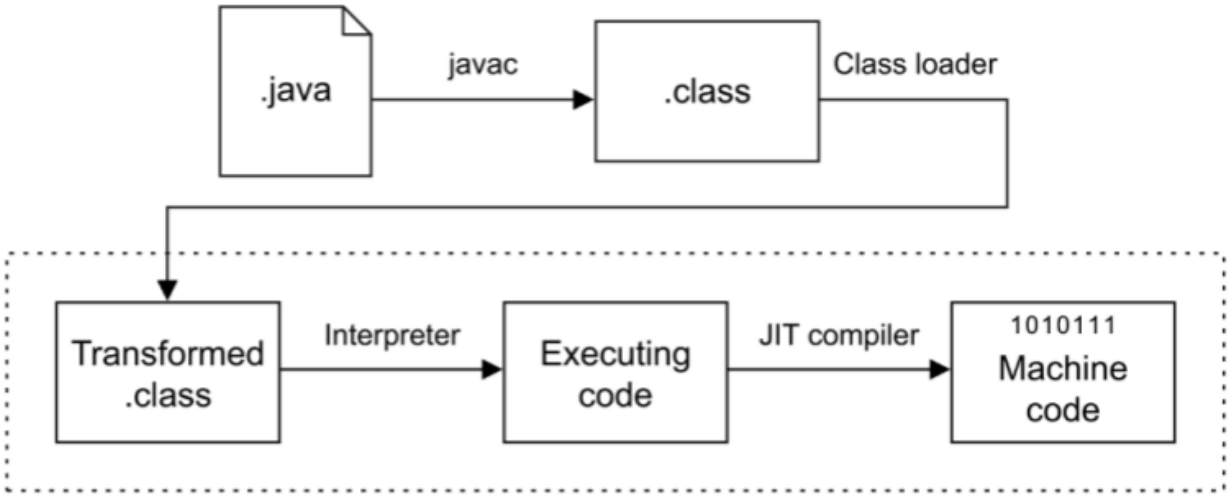

Рисунок 2.4 – Компіляція програмного коду в Java

В Java підтримується написання багатопоточних програм, здатних одночасно виконувати декілька дій. Приклад роботи потоків зображено на рис. 2.5. Синхронізація декількох процесів дуже складна, але вона надає можливість розробникам покращити оптимізацію програми. Приклад багатопоточності зображено на рис. 2.6

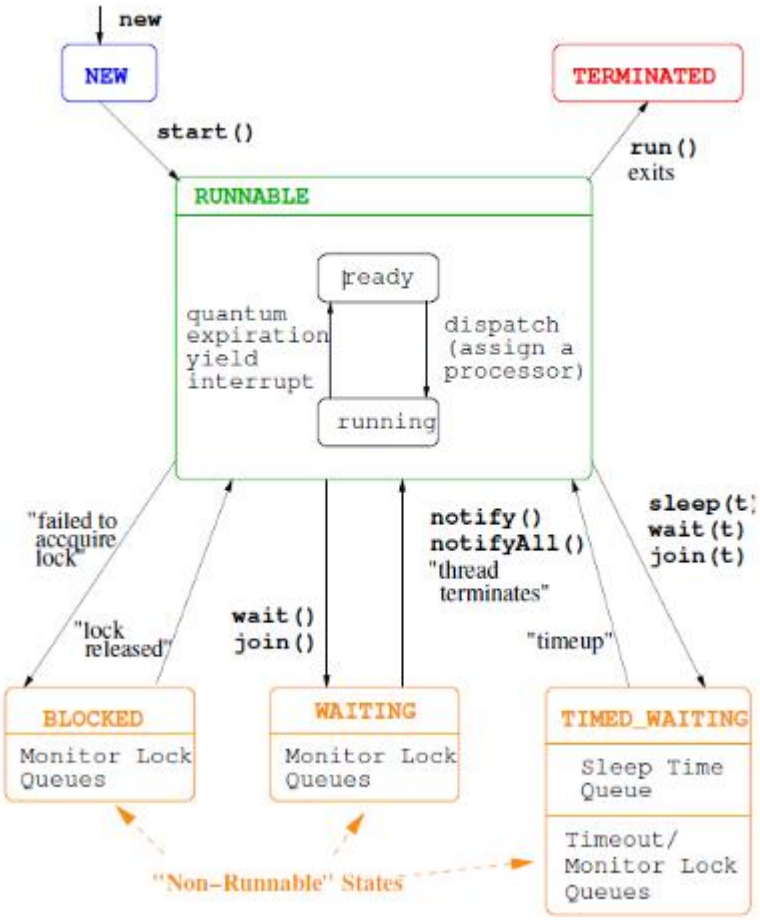

Рисунок 2.5 – Приклад роботи потоків в Java

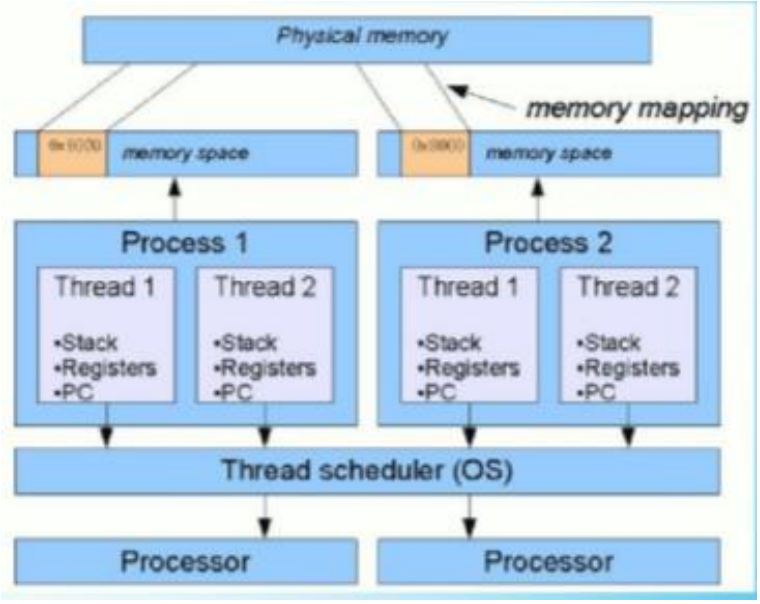

Рисунок 2.6 – Приклад багатопоточності в Java

Можна було б вважати недоліком JVM зниження продуктивності, але завдяки оптимізації байт-коду це не дуже помітно[9][1)]. Деякі удосконалення збільшили швидкість виконання програм:

- трансляція байт-коду в машинний код під час роботи програми з можливістю збереження версій класу в машинному коді;
- широке застосування платформено-орієнтованого коду в стандартних бібліотеках;
- апаратні засоби, що забезпечують прискорену обробку байт-коду.

І хоча Java в порівнянні з C++ повільніше виконує програми та споживає більше пам’яті, вона має деякі переваги:

- поєднання простого синтаксису з зручним в роботі середовищем розробки дозволяє швидше створювати нові програми;
- велику кількість бібліотек;
- програми, написані на Java, працюють на будь-яких пристроях що мають JVM.

[1)][9] Java – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Java#Основные_особенности_языка (дата звернення 21.08.2021).

2.2.2 Огляд Kotlin

Kotlin – це статично типізована мова програмування, розроблена компанією JetBrains. Метою авторів було створення більш лаконічної мови, ніж Java та простішої, ніж Scala, яку можна буде використовувати в змішаних проектах, а також повинна працювати всюди, де працює Java. Наслідками спрощення, стали швидша компіляція та краща підтримка[10][1)].

В 2017 році Kotlin став офіційною мовою для розробки застосунків для платформи Android. Отримавши підвищену продуктивність, програмний код на ньому в середньому на 40% коротше, ніж на інших мовах, а також він сумісний з Java (див. рис. 2.7).

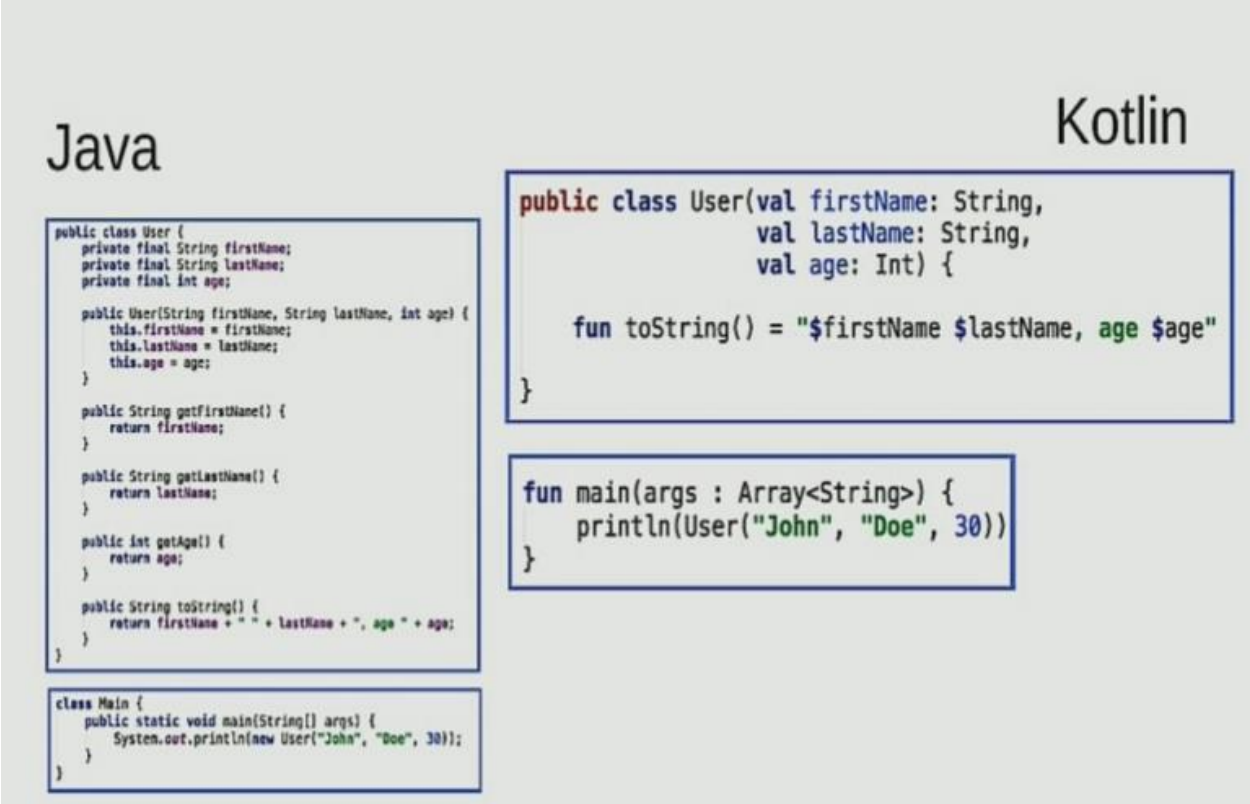

Рисунок 2.7 – Порівняння Java та Kotlin

[1)][10] Kotlin – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Kotlin (дата звернення 23.08.2021).

Kotlin чудово підходить для розробки додатків Android, що приносить всі переваги сучасної мови для платформи Android без введення нових обмежень:

– сумісність: Kotlin повністю сумісна з Java development kit 6 (JDK), що без проблем дозволяє працювати на старих пристроях. Інструмент Kotlin повністю підтримується в Android Studio і сумісний з системою Android build;

– продуктивність: додаток на Kotlin працює так само швидко, як еквівалентний Java, завдяки дуже схожим структурам байт-кодів;

– оперативна сумісність: Kotlin повністю сумісний з Java, що дозволяє використовувати всі існуючі бібліотеки Android у програмі Kotlin. включно обробку анотацій;

– час компіляції: підтримка комбінованої поетапної компіляції.

У підсумку: простота дозволяє використовувати мову майже будьякому Java-розробнику, зворотна сумісність дозволяє використовувати мову у вже існуючому проекті.

2.3 Розширювана мова розмітки XML

Стандарт XML (вперше виданий 10 лютого 1998, останнє, четверте видання 29 вересня 2006) визначає набір базових лексичних та синтаксичних правил для побудови мови описання інформації шляхом застосування простих тегів. Метою XML є визначеність і структурованість елементів дизайну незалежно від дизайну. Приклад файлу на рис. 2.8.

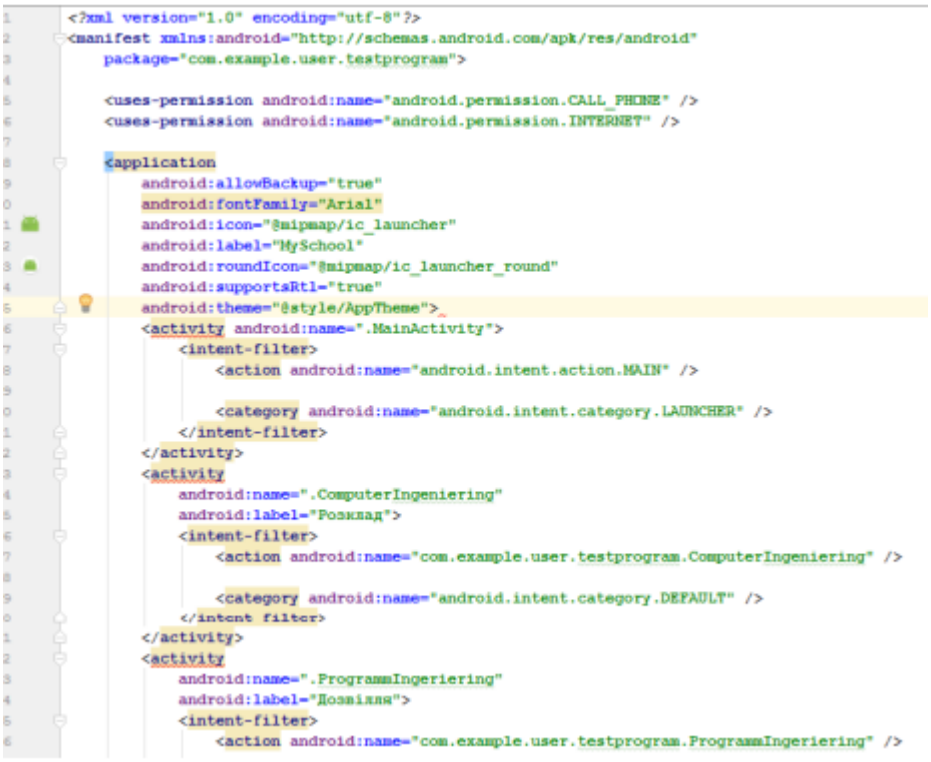

Рисунок 2.8 – Приклад XML файлу

При створенні XML- документів, розробники повинні дотримуватися таких вимог:

- мова розмітки, її номер та версія вказуються в оголошенні XML у заголовку документа;
- кожний відкриваючий тег повинен обов'язково мати відповідний закриваючий тег;
- повинен враховуватися регістр символів;
- всі значення атрибутів, повинні бути взяті в лапки.

XML-документ можна назвати формально-правильним, якщо в ньому дотримані всі приведені вище правила, і тоді аналізатори зможуть коректно працювати з таким документом[11][1)].

Є три рівні коректності:

[1)][11] Что такое XML. URL: https://javarush.ru/groups/posts/2287-chto-takoe-xml (дата звернення 25.08.2021).

- правильно побудований. Елементи такого документу структуровані у вигляді дерева, а теги відкриття та закриття відповідають нормам.
- діючий – містить теги, що відповідають оголошенню типу документа. Він містить тільки елементи і значення атрибутів, що відповідають правилам мови.
- синтаксично коректний не контролюється XML. За логічну структуру такого документа відповідає розробник.

2.4 Автоматизація побудови із Gradle

Створення програмного продукту без засобів автоматизації підвищує ризик помилок. Кожний етап в процесі розробки від написання коду до релізу потрібно робити вручну.

Автоматизація проекту знижує кількість ручного втручання в проект, та робить розробку більш ефективною. Вона включає такі дії як компіляція сирцевого коду в бінарний та його складання, виконання тестів, написання документації.

Основною метою була автоматизація викликів компіляторів і компонувальника. Згодом, із зростання процесів, почали додаватися нові дії, як наприклад, перевірка версій. На даний час процес автоматизації включає в себе дії з компіляцією та компонуванням (див. рис. 2.9).

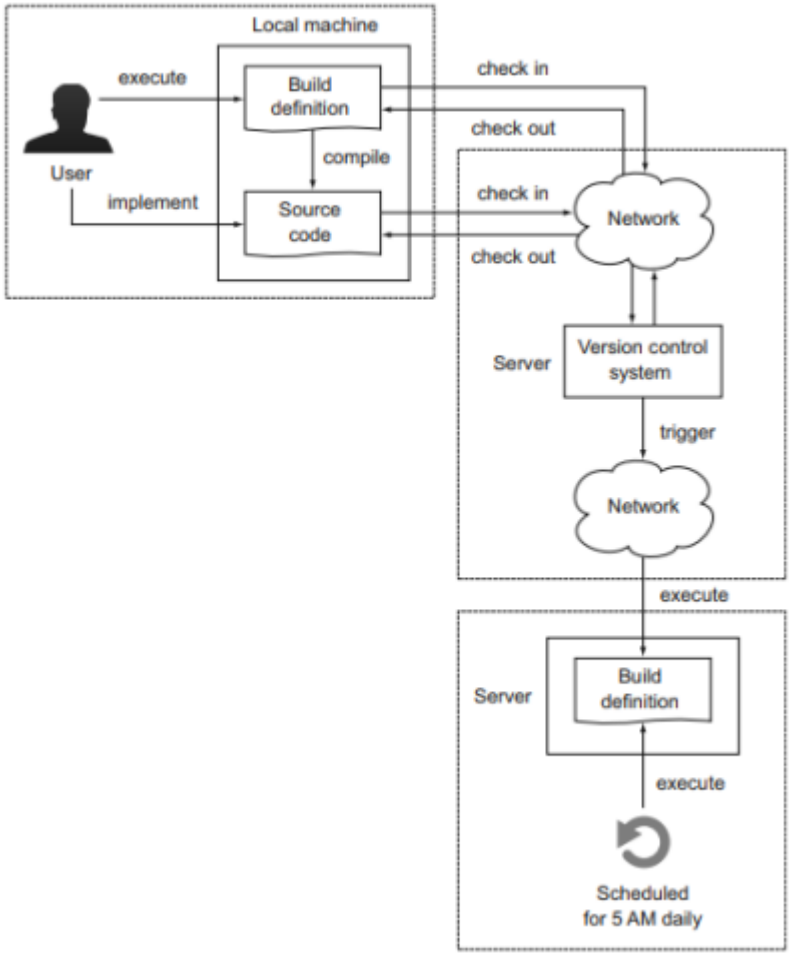

Рисунок 2.9 – Приклад запланованої збірки

Автоматизована збірка Gradle була випущена в 2007 році зробила крок евоюції в інструментах побудови JVM. Опираючись на Maven і Ant вона вті-лила їх найкращі ідеї та покращила їх.

Gradle дозволяє декларативно моделювати вашу предметну область за допомогою потужної доменної мови DSL, на основі Groovy замість XML (див. рис. 2.10). Gradle був розроблений для створення багато-проектних збірок. Підтримка додаткових збірок не заважає визначити, які частини оновлюються.

Переваги:

– поліпшення якості продукту

– прискорення процесу компіляції і компонування

– ведення історії складання і релізів для розбору випусків

```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'

android {
    compileSdkVersion 25
    buildToolsVersion '27.0.3'
    defaultConfig {
        applicationId "com.example.user.testprogram"
        minSdkVersion 15
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    api fileTree(dir: 'libs', include: ['*.jar'])
    androidTestImplementation('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    //noinspection GradleDynamicVersion
    implementation 'com.android.support:appcompat-v7:25.3.1+'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    implementation 'com.squareup.picasso:picasso:2.5.2'
    implementation 'com.android.support:support-v4:25.3.1'
    testImplementation 'junit:junit:4.12'
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
}
repositories {
    mavenCentral()
}
```

Рисунок 2.10 – Gradle файл

Опис залежності від каталогів, модулів, завдань та ін. є досить гнучким. Наприклад, в Gradle можна вказати, що один модуль залежить від скомпільованих класів, а не від результату (jar) збірки іншого (див. рис. 2.11).

Побудова Gradle розпочинається із сценарію, що за стандартом має назву build.gradle. При виконанні базової команди систему веде пошук файлу саме з такою назвою та за його відсутності отримаєте повідомлення про помилку[12][1)].

[1)][12] Gradle – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Gradle (дата звернення 28.08.2021).

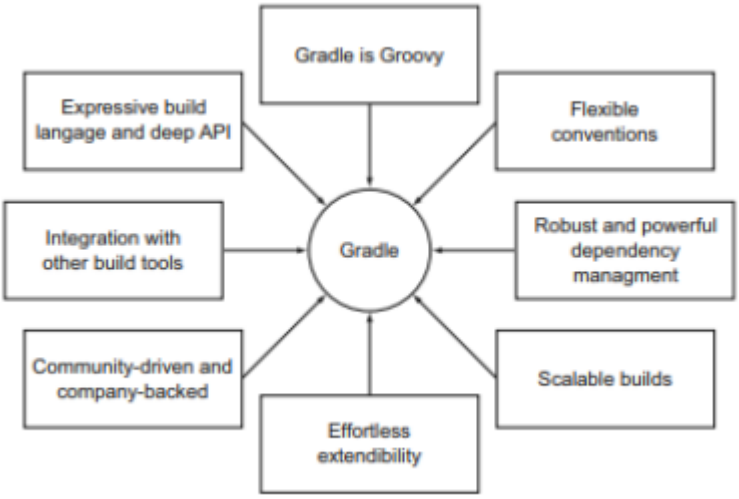

Рисунок 2.11 – Набір функцій системи Gradle

Підводячи підсумок, можна сказати що Gradle досить гнучка система автоматизованої збірки. Зважаючи на всі переваги, описані вище, вона займає пріоритетне місце для побудови багатьох проектів.

2.5 Файл маніфесту

Кожний Android додаток містить файл AndroidManifest.xml у кореневій папці проекту. Цей файл можна редагувати вручну, змінюючи XML-код, або за допомогою візуального редактору. Тим не менш, файл маніфесту є дуже важливим, адже він інкапсулює всю архітектуру, конфігурацію та функціонал додатку[13][1)].

Файл маніфесту призначений для наступного:

- оголошує ім’я пакета додатку, що є унікальним ідентифікатором;
- містить список потрібних для додатка дозволів;
- описує компоненти програми, тобто її діяльності, служби та контент-провайдери;

[1)][13] Android: Файл манифеста AndroidManifest. URL: http://developer.alexanderklimov.ru/android/theory/AndroidManifestXML.phpдата звернення 01.09.2021).

− перераховує пов’язані бібліотеки;

− оголошує мінімальний рівень API, необхідний для роботи програми.

Елемент <manifest> являється кореневим. Обов'язковими елементами є теги <application> і <uses-sdk>. Основний елемент <application> містить дочірні елементів, які обумовлюють оботу та структуру. Через атрибути елементів встановлюються всі значення. Крім обов'язкових елементів, у маніфесті за потребою використовуються інші елементи (див. рис. 2.12).

Рисунок 2.12 – Приклад AndroidManifest

2.6 Пакети Android SDK

Розглянемо структуру пакетів Java, щоб мати більше уявлення про мобільну платформу. Android SDK відрізняється від стандартного, тому нижче наведено важливі пакети.

Android.app −відповідає за реалізацію моделі додатку.

Android.Bluetooth − містить класи для роботи з Bluetooth. Основні класи BluetoothDevise, BluetoothAdapter, BluetoothServerSocket, BluetoothSocket та BluetoothClass. Клас BluetoothDevise представляє собою віддалений пристрій Bluetooth. BluetoothAdapter – це клас, що використовується для управління адаптером, який встановлений на комп'ютері.

Android.content − призначений для роботи з постачальниками контенту, які сумувати обмін даних та зберігають їх.

Android.content.pm – Представляє класи, що пов'язані диспетчером пакетів. Інформація про компоненти та дозволи знаходиться в цьому пакеті.

Android.database – створений для абстрагування бази даних. Основний інтерфейс називається Cursor;

Android.database.sqlite − реалізує концепцію пакета бази даних, в якості якої є фізична SQLite. Основні класи SQLiteQuery, SQLiteQueryBuilder, SQLiteCursor, SQLiteDatabase.

Android.graphics − містить класи Color, Canvas, Camera, Matrix, Movie, Path, Paint, Rasterizer.

Android. hardware − дозволяє використовувати класи, призначені для роботи з камерою.

Android.location містить класи Address, GeoCoder, Location, LocationManager і LocationProvider.

Android.media − містить класи MediaPlayer, MediaRecorder, Ringtone, AudioManager і FacebookDetector. Клас MediaPlayer призначений для потокової підтримки аудіо та відео. Клас Ringtone використовується для відтворення коротких аудіовідліків. AudioManager відповідає за регулювання гучності. FaceDetector може бути використаний для виявлення людських облич.

Android.net −реалізує базову мережу на рівні сокетів API. Основні класи включають Uri, ConnectivityManager, локальний сокет та місцевий ServerSocket. Слід також відзначити, що Android підтримує рівень HTTPS-браузера та мережевий рівень. Крім того, Android підтримує JavaScript в браузері.

Android.net.WiFi − керує підключенням Wi-Fi. Основні класи WifiManager і WifiConfiguration. Клас WifiManager відповідає за складання списку налаштованих мереж і працює з поточною активною мережею Wi-Fi;

Android.OpenGL −містить допоміжні класи, які використовуються при виконанні операцій OpenGL ES.

Android.os−служба операційної системи, доступ до якої здійснюється за допомогою мови Java. Деякі важливі класи − BatteryManager, FileObserver, Handler, Looper та PowerManager. FileObserver веде облік змін у файлах. Клас Handler використовується для виконання завдань у потоці повідомлень, і Looper починається саме повідомлення потоку;

Android.preference − дозволяє додаткам надати користувачам можливість керувати своїми налаштуваннями для цієї програми в єдиній формі. Основні класи PreferenceActivity, PreferenceScreen;

Android.provider − включає набір готових контент-провайдерів, пов'язаних з android.content. ContentProvider. Серед постачальників контенту −Контакти, MediaStore, браузер та налаштування. Цей набір інтерфейсів і класів містить методи для опису базової структури даних;

Android.sax −забезпечує ефективний набір простих API для XML (SAX), допоміжні класи, призначені для синтаксичного аналізу. 27за. Основні класи Element, RootElement деякі ElementListener інтерфейси;

Android.speech −містить константи для розпізнавання мови. Цей пакет входить тільки у версії 1.6 і вище;

Android.speech.tts − забезпечує підтримку для перетворення тексту на мову. Основний клас −TextToSpeech. В Android є механізм PICO TTS (перетворення тексту в мову, синтезатор мови) виробництва SVOX;

andmid.tekphony − містить класи CellLocation, PhoneNumberUtils і TelephonyManager. TelephonyManager клас для визначення місцезнаходження, з якого було здійснено виклик, номер телефону, ім'я постачальника послуг, тип мережі, тип телефону та серійний номер модуля ідентифікації абонента (Subscriber Identity Module, SIM);

Android.tekphony. GSM − дозволяє збирати інформацію про адреси осередків, засновані на розташування даних вишок стільникового зв'язку, але також містить класи, відповідальні за роботу з SMS-повідомленнями. визначає глобальну систему мобільного зв'язку в ім'я цього пакета називається GSM, як оригінальні короткі стандарти обміну повідомленнями (SMS) (Глобальна система мобільного зв'язку); CDMA-підтримує CDMA-телефонію;

Android.text − містить класи для обробки тексту;

Android.Text. method −надає класи для введення тексту в різні елементи управління;

Android.Text. style −забезпечує різноманітність методів обробки тексту;

Android.Utils −містить класи, DebugUtils, TimeUtils і Xml;

Android.view−містить класи, меню View, ViewGroup, а також деякі з процесів слухачів та зворотних викликів;

Android.view.animation −забезпечує підтримку анімації в структурі проміжних кадрів;

Android.view.InputMethod −реалізує введення-виведення системної архітектури. У цьому пакеті міститься лише у версіях 1.5 та вище;

Android.WebKit містить класи, пов'язані з веб-браузером. Серед основних класів WebView, CacheManager та CookieManager;

Android.widget −містить всі класи елементів управління інтерфейсу користувача, які отримані головним чином з точки зору класу. Основні віджети −Кнопка, Галочка, хронометр, AnalogClock, DatePicker, EditText, ListView, FrameLayout, GridView, ImageButton, MediaController, ProgressBar, RadioButton, RadioGroup, RatingButton, скроллер, Scrollview, Spinner, TabWidget.

com.google.android.maps − містить клас MapView, MapController і MapActivity, необхідні для роботи з Google Maps[14][1)].

[1)][14] Android SDK – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Android_SDK (дата звернення 03.09.2021).

Android SDK містить в собі більше 40 пакетів та 700 класів, тож вище перераховані пакети лише невелика частина важливих для мобільної платформи пакетів.

2.7 Вибір інтегрованого середовища розробки

Для створення програми на ОС Android найбільш популярністю користуються такі середовища розробки:

– «Android Studio»
– «Eclipse»
– «IntelliJ IDEA»

2.7.1 Огляд Android Studio

У наш час програмне забезпечення створюється з використанням інтегрованого середовища розробки. В IDE автоматизований процес компіляції, збірки та запуску додатків полегшує роботу. Підсвічення синтаксису та нумерація рядків значним чином допомагають зорієнтуватися в програмного коді. Майже кожне IDE має автоматичний відладник додатків. Також дуже важливою є інтеграція з системою контролю версій[15][1)].

Android Studio – це інтегроване середовище розробки, побудоване на основі IntelliJ IDEA, яке можна встановити на різні операційні системи та використовувати його інструменти для створення додатків на Android.

Зазвичай робота над створенням розпочинається із створення користувальницького інтерфейсу. Для цього в Android Studio присутні різні макети. Е середовище розробки є віконним. Проекти Android складаються з багатьох ка-

[1)][15] Android Studio: среда разработки мобильных приложений. URL: https://arduinoplus.ru/android-studio/ (дата звернення 08.09.2021).

талогів та файлів. Така фрагментація зумовлена кращою класифікацією файлів. Крім фізичного пристрою, готовий додаток можна встановити та перевірити якість його роботи на віртуальному пристрої Andorid (AVD). Тестування додатку на AVD, зручніше та швидке, ніж на фізичного завдяки пристроям з різними екранами, співвідношеннями сторін та різними версіями операційної системи[16][1)].

Додатки розробляються в Android Studio з використанням Java, C ++ або Kotlin , яку Google анонсував 17 травня 2017, як офіційну мову програмування. В основі робочого процесу Android Studio закладений концепт безперервної інтеграції, що дозволяє відразу ж виявляти наявні проблеми.

2.7.2 Огляд Eclipse

Eclipse – це середовище розробки кросплатформових додатків, що підтримується некомерційною організацією Eclipse Foundation.

В більшості Eclipse використовують для розробки розширень. Вони збільшують можливості середовища розробки.

Eclipse JDT − модуль, націлений на групову розробку: середовище інтегроване із системами управління версіями CVS, GIT є вбудованим, для інших систем існують плагіни. Також пропонує підтримку зв'язок між IDE та системою управління завданнями (помилками). Підтримка трекера помилок Bugzilla та безліч інших розширень для підтримки різних трекерів також вбудовані за стандартом.

Eclipse написана на Java, тому є платформо-незалежним продуктом, крім бібліотеки для розробки графічних інтерфейсів SWT, яка розробляється всім поширених платформ. Замість стандартної бібліотеки Java Swing використо-

[1)][16] Android Studio IDE от Google. URL: http://wnfx.ru/android-studio-ide-ot-google/ (дата звернення 08.09.2021).

вується бібліотека SWT. Вона забезпечує швидкість і простий зовнішній вигляд інтерфейсу користувача, але іноді викликає на різних платформах проблеми сумісності[17][1)].

Платформа розширеного клієнта, яка є основою для Eclipse, має такі компоненти:

− ядро платформи, призначення якого – запуск модулів;

− OSGi (стандартне середовище постачання комплектів);

− SWT (портований інструментарій віджетів);

− JFace (файлові буфери, робота з текстом, текстові редактори);

− робоче середовище Eclipse (панелі, редактори, проекції, майстри).

Завдяки додатковим модулям, програми можна розроблювати на таких мовах програмування як Groovy, Ruby, C/C++, Perl, Python, PHP та ін.[18][2)]

Завдяки активному розвитку та постійній підтримці з’явилось багато нових інструментів та функцій Eclipse має наступні переваги:

– офіційна русифікація інтерфейсу і документації

– велике число доповнень

– можливість підключення модулів

– можливість групової розробки

На жаль недоліків для більше, ніж переваг. До основних можна віднести велике використання системних ресурсів, сильне навантаження на процесор, неповна інтеграція з Maven і Gradle та громіздкість деяких конфігурацій.

Eclipse була дуже популярна кілька років тому. Однак у зв'язку з виходом Android Studio, в 2014 році Google перестала підтримувати Eclipse як основне середовище для розробки додатків під ОС Android.

[1)][17] IDE Eclips: за и против от ведущих программистов.
URL: https://proglib.io/p/ide-eclipse-za-i-protiv-ot-vedushchih-programmistov-2019-11-02 (дата звернення 12.09.2021).

[2)][18] ANDROID STUDIO VS ECLIPSE: ПРЕИМУЩЕСТВО КАЖДОЙ ИЗ СРЕД. URL: https://kitapp.pro/android-studio-vs-eclipse-preimushhestvo-kazhdoj-iz-sred/ (дата звернення 12.09.2021).

2.7.3 Огляд IntelliJ IDEA

IntelliJ IDEA – це інтегроване середовище розробки розроблене компанією JetBrains. Це середовище дозволяє розробляти програми на декількох мовах (Python, PHP, C++, Java та ін.).

IDE доступна в двох версіях «Community Edition» та «Ultimate Edition». Версія Community середовища IntelliJ IDEA надає розробнику інструменти (у вигляді плагінів) системи контролю версій Subversion, Mercuria, CVS і Git, засоби автоматичної збірки Gradle, Ant, Maven, для тестування додатку, мови програмування Java, Scala, Clojure, Groovy і Dart. Також підтримується розробка застосунків для мобільної платформи Android. Такі модулі як XML - редактор, перевірка коректності коду, проектування GUI–інтерфейсу Swing UI Designer та інтеграція проектів з Eclipse.

Комерційна версія «Ultimate Edition» має додаткові функції, а саме використання додаткових мов програмування (JavaScript, SQL, HTML), інструменти для роботи з фреймворками (Spring, Play Framework, Grails, Google Web Toolkit і Hibernate), побудова UML – діаграм, засоби інтеграції з Microsoft Team Foundation Server, Perforce та Rational ClearCase.

Автодоповнення – це дуже важливий функціонал. І хочу він доступний для кожної IDE, але в IntelliJ IDEA штучний інтелект представив його на новому рівні. Зазвичай середовище розробки пропонує декілька варіантів для автодовнення, але в цьому середовищі розробки вони дійсно відповідають вашим потребам.

Мабуть, однією із сильних переваг є підтримка широкого кола технологій. Такі технології мають підтримку зі сторони компанії доти, поки нею користуються.

Також для зручності роботи тут присутній графічний редактор. Автоматична генерація коду відповідно до інтерфейсу економить час, завдяки чому його використовують не тільки новачки.

IntelliJ IDEA має наступні переваги:

- автозаповнювач методів;
- є розширений рефакторінг та форматування;
- зрозумілий та лаконічний інтерфейс;
- швидка відладка значень;
- інструменти для аналізу якості коду;
- дизайнер інтерфейсу;
- інтреграція з автоматизованими системами збірки та наявність інструментів для тестування програм;
- інтеграція із системами контролю версій.

Після аналізу, в якості середи розробки була вибрана IDE Android Studio. Причиною вибору є досить зручний інтерфейс, легкість написання коду та те, що вона спеціально розроблена для створення додатків Android[19][1)].

[1)][19] IntelliJ IDEA – Вікіпедія. URL: https://ru.wikipedia.org/wiki/IntelliJ_IDEA (дата звернення 15.09.2021).

3 ПРОЕКТНА ЧАСТИНА

3.1 Підключення сервісів API

API розшифровується як "Application Programming Interface" (інтерфейс програмування програм). Будь-який сервіс на певному етапі, розроблює API для внутрішнього використання надає для сторонніх програм.

Одним з найпопулярніших сервісів пов’язаних з прогнозом погоди є OpenWeather, а для відстеження забруднення повітря AQICN. Вони дозволяють працювати за допомогою HTTP-запитів. За допомогою API можна додавати, видаляти, змінювати та отримувати інформацію про різні об’єкти. Текст запиту має відповідати текстовому формату обміну даними JSON або XML. В залежності від цього змінюється адреса та значення.

Наприклад, для сервісу OpenWeather після виконання наступного запиту: api.openweathermap.org/data/2.5/weather?q=London&appid={API key}, де API key – це ваш ключ доступу, ми отримаємо відповідь у JSON форматі:


```
{
"coord":{
"lon": 145.77,
"lat": -16.92 },
"weather": [
{
"id": 802,
"main": "Clouds",
"description": "scattered clouds",
"icon": "03n"
}
],
"base": "stations",
"main": {
"temp": 300.15,
"pressure": 1007,
"humidity": 74,
},
"visibility": 10000,
"wind": {
"speed": 3.6,
"deg": 160 }
```

Отримати дані від web-служб можна за допомогою HttpClient сервіс[20][1)]. Будемо використовувати імплементацію java.net.URLConnection, яка доступна в стандартній бібліотеці Java 8. Для отримання сутності URLConnection потрібно використовувати об'єкт класу java.net.URL, його конструктор приймає тип String. Також треба зазначити протокол, у моєму випадку це протокол https. Викликаємо метод openConnection() після отримання сутності URL, результат якого поверне нам сутність HttpsUrlConnection. Тепер потрібно передати тіло запиту в InputStream. Для цього використовуємо InputStreamReader. Для читання використовуємо BufferedReader, і в разі успішної аутентифікації, отримуємо в результаті об’єкт JSON (див. рис 3.1).

```
protected JSONObject getUrlContent(String urlName) throws JSONException {
    StringBuilder content = new StringBuilder();

    try {
        URL url = new URL(urlName);
        URLConnection urlConn = url.openConnection();

        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(urlConn.getInputStream()));
        String line;

        while ((line = bufferedReader.readLine()) != null) {
            content.append(line).append("\n");
        }
        bufferedReader.close();
    } catch (Exception e) {
        System.out.println("Неверный запрос API" + e);
    }
    return new JSONObject(content.toString());
}
```

Рисунок 3.1 – Обробка API запиту

Після цього, вилучаємо дані з JSON об’єкту методом getJSONObject.

[1)][20] Что Такое API. URL: https://developer.mozilla.org/ru/docs/Learn/JavaScript/Client-side_web_APIs/Introduction (дата звернення 20.09.2021).

3.2 UML діаграма проекту

UML (англ. Unified Modeling Language – уніфікована мова моделювання) це – мова графічного опису створення моделей створена для системного проектування та відображення організаційних структур.

На рис. 3.2 зображено UML діаграму класів проекту, на якій показані зв'язки між класами.

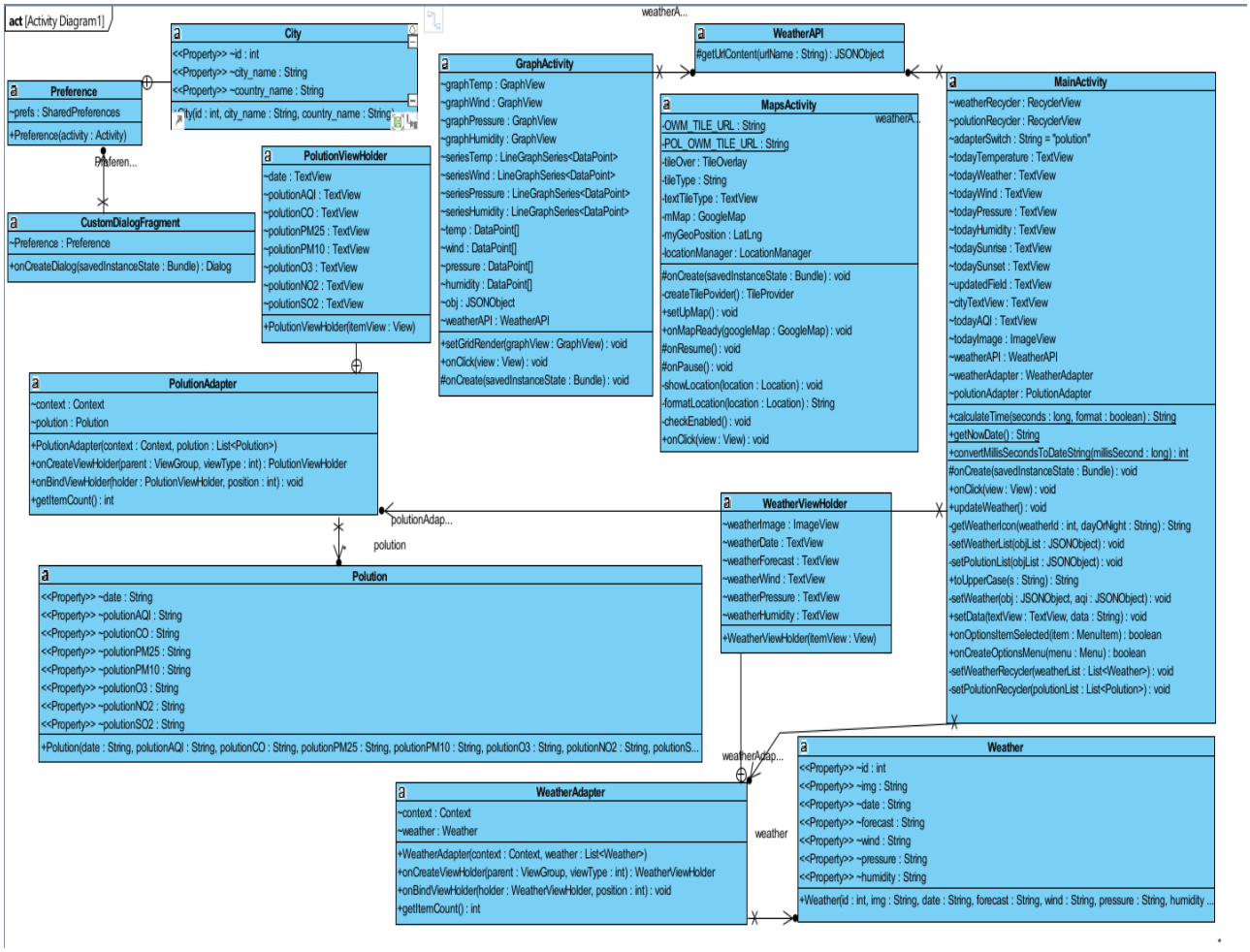

Рисунок 3.2 – UML діаграма класів

UML був створений для візуалізації, визначення та проектування програмних систем. UML не є мовою програмування, але на підставі UML–моделей можлива генерація коду[21][1)].

3.3 Проектування зовнішнього вигляду

Розробка дизайну додатка є найважливішим етапом у розробці. Зручність, простота, зовнішній вигляд та функціонал – це те, на що користувач звертає увагу перш за все.

Дизайн програми – це візуальне оформлення програми, структура, яка створена для подальшої реалізації. Дизайн додатка можна поділити на UX (User Experience, досвід користувача) і UI (User Interface, інтерфейс користувача) частини.

Виявити, проаналізувати, протестувати і впровадити інтуїтивно зрозумілий дизайн, допоможе UX (User eXperience). Ґрунтуючись на досвіді попередніх програм, додатків конкурентів, UX дизайн додатків дозволить виробити найбільш ефективний і інтуїтивно зрозумілий інтерфейс.

Корисний мобільний додаток має бути орієнтованим в першу чергу на клієнта. Допомагаючи вирішити всі необхідні завдання без зайвих зусиль. Тому користувацький інтерфейс повинен бути простим і зрозумілим.

Не менш важливий елемент – це UI дизайн програми. UI визначає кольори і загальне візуальне оформлення додатка, то наскільки зручно користувачеві буде натискати на кнопки і наскільки читабельним буде текст.

Процес розробки дизайну складається з трьох етапів. Визначення призначення додатку – це перший етап. Другим етапом є створення ескізу. Подумати над тим, як буде реалізовано функціонал додатку з таким зовнішнім виглядом для уникнення зайвої роботи при повному відображення екрану.

[1)][21] Діаграма діяльності – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Діаграма_діяльності (дата звернення 26.09.2021).

Останній етап – створення схеми зв’язку елементів різних екранів з функціями.

Екран головної сторінки (див. рис. 3.3) складається із двох фрагментів, що реалізують верхню та нижню частини інтерфейсу та меню, що знаходиться на панелі інструментів. У верхньому фрагменті є перемикач, за допомогою якого користувач зможе змінювати вивід нижнього фрагменту, а саме прогноз погоди або забруднення.

Рисунок 3.3 – Головна сторінка

У випадаючому списку пункту меню головної сторінки (див. рис. 3.4) при виборі «Карта» користувач зможе переглядати різні слої карти (див. рис. 3.5).

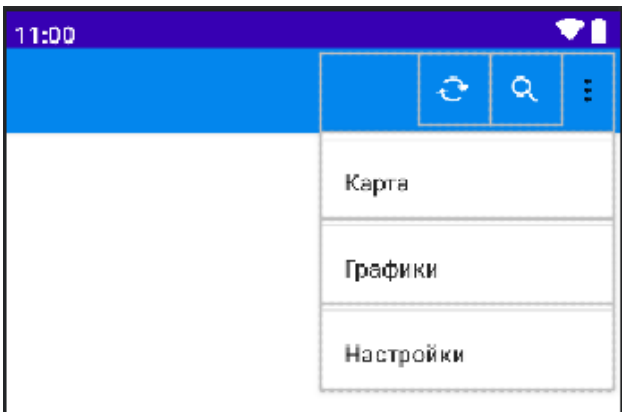

Рисунок 3.4 – Меню головної сторінки

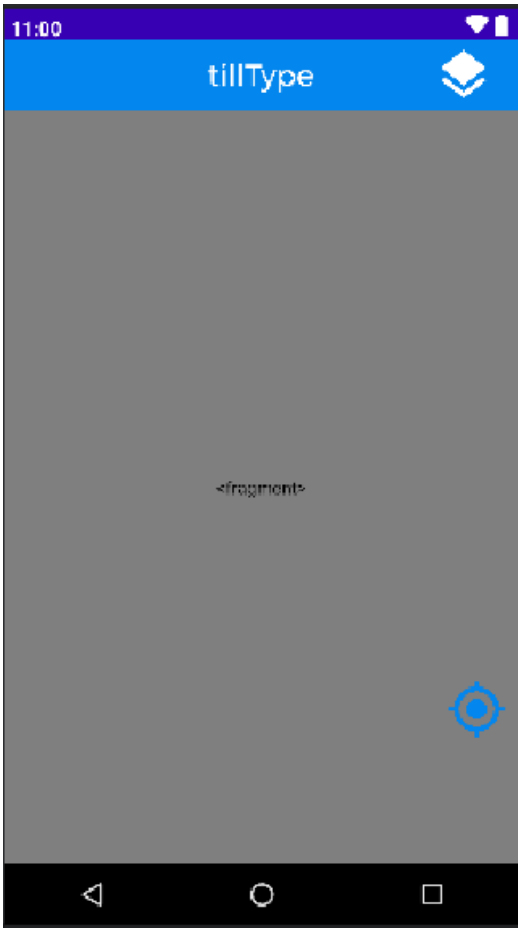

Рисунок 3.5 – Вкладка «Карты»

Якщо натиснути на пошук місцезнаходження, карту користувача збільшить та наблизить до його локації.

На вкладці «Графики» (див. рис. 3.6) буде побудовано лінійні графіки для різних даних.

На сторінці «Настройки» користувач може змінювати формат даних (див. рис. 3.7). Наприклад, для температури стандартний вивід показує у градусах за Цельсієм, але в налаштуваннях додатку користувач може змінити на показ градусів за Фаренгейтом.

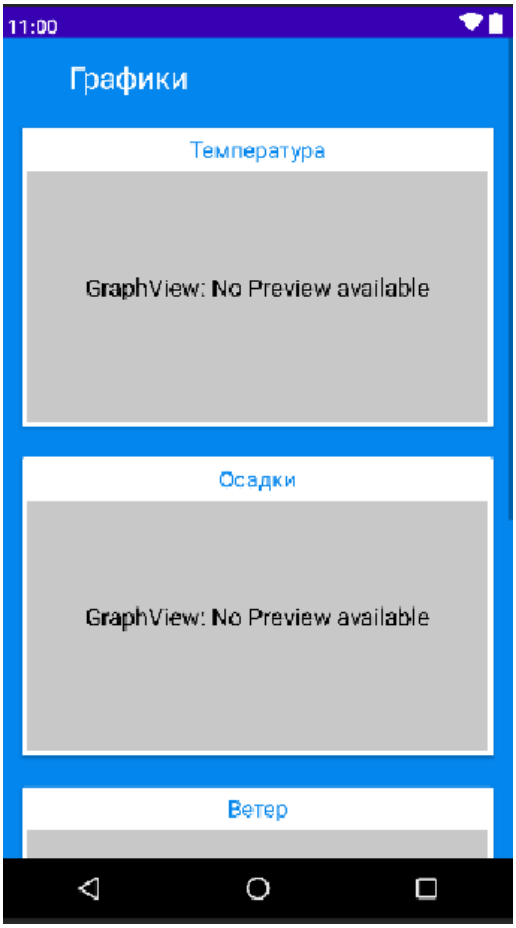

Рисунок 3.6 – Вкладка «Графики»

Рисунок 3.7 – Вкладка «Настройки»

3.4 Розробка окремих програмних елементів

3.4.1 Розробка інструментів для взаємодії з картами

Для роботи з картами були підключені такі пакети як Android.location та com.google.android.gms.maps. До першого пакету включені класи GeoCoder, завдяки роботі з яким можливо дізнатися адресу координат об’єкта, та клас Location, що надаж інформацію про щироту та довготу[22][1)]. Другий пакет призначений безпосередньо для роботи с Google Maps.

Клас, який виконує всю роботу з картами у додатку має назву MapsActivity. Він має наступні основні елементи:

- об'єкт SupprotMapFragment керує життєвим циклом карти і є головним елементом інтерфейсу програми;
- об'єкт GoogleMap надає доступ до даних карти. Це основний клас у Maps SDK для Android;

[1)][22] Maps SDK для Android: краткое руководство. URL: https://developers.google.com/maps/documentation/android-sdk/start?hl=ru (дата звернення 08.10.2021).

- функція moveCamera центрує карту за координатам LatLng, що в нашому випадку є координатами місцезнаходження. Зазвичай, при роботі з картами спочатку потрібно, змінити розташування та налаштування камери: орієнтацію карти, кут огляду та масштаб;
- функція addMrker э стандартною. Вона додає маркер з вказаними координатами на карту. Але в даному випадку він не потрібний.

При роботі з такими сервісами як OpenWeather та AQICN потрібно змінювати слої карти. TileProvider – це інтерфейс для класу, що надаж зображення плитки для TileOverlay. Виклик методи цього інтерфейсу можна робити з декількох потоків, тому його реалізація має бути потокобезпечною[23][1)].

Наступний метод повертає URL адресу зображення, що буде використано для плитки.

```
private TileProvider createTilePovider() {
    return new UrlTileProvider(256, 256) {
        @Override
        public URL getTileUrl(int x, int y, int zoom) {
            @SuppressLint("DefaultLocale") String fUrl =
String.format(OWM_TILE_URL, tileType == null ? "clouds" :
tileType, zoom, x, y); // This is the OWM url you can use yours
            URL url = null;
            try {
                url = new URL(fUrl);
            } catch (MalformedURLException mfe) {
                mfe.printStackTrace();
            }
            return url;
        }
    };
}
```

Наступне, шо потрібно зробити це додати цю плитку до карти

```
public void setUpMap() {
    tileOver = mMap.addTileOverlay(new TileOverlayOptions().
            tileProvider(createTilePovider()));
}
```

[1)][23] Google MAPs API в android или как работать с картами быстрее. URL: https://habr.com/ru/post/341548/ (дата звернення 08.10.2021).

Для пошуку місцезнаходження використовуємо об’єкт класу Location-Manager. Приклад коду зображено на рис. 3.8.

```
@Override
protected void onResume() {
    super.onResume();
    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION)
        //...
        return;
    }
    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
            minTimeMs: 1000 * 10,  minDistanceM: 10, locationListener);
    locationManager.requestLocationUpdates(
            LocationManager.NETWORK_PROVIDER,  minTimeMs: 1000 * 10,  minDistanceM: 10,
            locationListener);
    checkEnabled();
}
@Override
protected void onPause() {...}

private void showLocation(Location location) {
    if (location == null)
        return;
    if (location.getProvider().equals(LocationManager.GPS_PROVIDER)) {
        System.out.println(formatLocation(location));
    } else if (location.getProvider().equals(
            LocationManager.NETWORK_PROVIDER)) {
        System.out.println(formatLocation(location));
    }
}
```

Рисунок 3.8 – Код програми для роботи із місцезнаходженням

3.4.2 Розробка інструментів для роботи з графіками

Для побудови графіку підключаємо просту у використанні бібліотеку com.jjoe64.graphview.GraphView.

Розрахунок точок графіка:

```
JSONArray objList = obj.getJSONArray("list");
System.out.println(objList);
for (int i = 0; i < objList.length(); i++) {
    temp[i] = new
DataPoint(objList.getJSONObject(i).getInt("dt"),
objList.getJSONObject(i).getJSONObject("temp").getDouble("eve"))
;
```

```
    wind[i] = new
DataPoint(objList.getJSONObject(i).getInt("dt"),
objList.getJSONObject(i).getDouble("speed"));
    pressure[i] = new
DataPoint(objList.getJSONObject(i).getInt("dt"),
objList.getJSONObject(i).getInt("pressure"));
    humidity[i] = new
DataPoint(objList.getJSONObject(i).getInt("dt"),
objList.getJSONObject(i).getInt("humidity"));
}
seriesTemp = new LineGraphSeries<>(temp);
seriesWind = new LineGraphSeries<>(wind);
seriesPressure = new LineGraphSeries<>(pressure);
seriesHumidity = new LineGraphSeries<>(humidity);
```

Точки графіка будуються за даними з JSON файла, але для підпису горизонтальної лінії (осі x) даними типу String потрібно викликати метод getGridLabelRenderer. Цей метод форматує час із мілісекунд в день тижня.

Реалізація такого методу:

```
graphView.getGridLabelRenderer().setLabelFormatter(new
DefaultLabelFormatter() {
    @Override
    public String formatLabel(double value, boolean isValueX) {
        // TODO Auto-generated method stub
        if (isValueX) {
            @SuppressLint("SimpleDateFormat") String dayOfWeek =
new SimpleDateFormat("EE").format(new Date((long) (value *
1000L)));

            return dayOfWeek;
        }
        return "" + (int) value;
    }
});
```

І тепер тільки залишилося додати точкові дані до об’єкта GraphView:

```
graphTemp.addSeries(seriesTemp);
setGridRender(graphTemp);
graphWind.addSeries(seriesWind);
setGridRender(graphWind);
graphPressure.addSeries(seriesPressure);
setGridRender(graphPressure);
graphHumidity.addSeries(seriesHumidity);
setGridRender(graphHumidity);
```

3.4.3 Розробка інструментів для роботи з налаштуваннями

Майже у кожній програмі потрібно зберігати частину даних для подальшого використання, наприклад, дані про користувача, формат виводу даних та ін. В мобільній платформі андроїд існує концепція Preferences. Налаштування є групой пар ключ-значення. Значення цих даних можуть бути будь-яким примітивним типом.

За стандартом вони є спільними для всіх активностей, але за бажанням можна зробити їх окремо. Налаштування зберігаються в XML-файлах у незашифрованому вигляді у локальному сховищі.

При роботі з налаштуваннями слід враховувати, що вони не зашифровані, а тому краще не зберігати в них особисті дані. Крім того, вони представляють дані, асоційовані з програмою, і через панель управління програмою в налаштуваннях операційної системи користувач може видалити ці дані[24][1)].

Для роботи з пакетом SharedPreferences використовуємо метод базового класу Context, що називається getSharedPreferences.

```
import android.content.SharedPreferences;<font></font>
<font></font>
//.........................<font></font>
<font></font>
SharedPreferences settings =
getSharedPreferences("PreferencesMyName", MODE_PRIVATE);
<font></font>
```

У вище згаданому коду перший парметр («PreferencesMyName») вказує назву налаштувань. Другий параметр вказує на режим доступу.

Клас android.content.SharedPreferences надає ряд методів для керування налаштуваннями:

– getAll(): повертає всі збережені в установках значення;

[1)][24] SharedPreferences. Сохранение данных в постоянное хранилище Android. URL: https://www.fandroid.info/sharedpreferences-sohranenie-dannyh-v-postoyannoe-hranilishhe-android/ (дата звернення 15.10.2021).

– getFloat(String key, float defValue): повертає значення типу float із ключем key;
– getInt(String key, int defValue): повертає значення типу int із ключем key;
– getBoolean (String key, boolean defValue): повертає значення типу Boolean, яке має ключ key;
– getLong(String key, long defValue): повертає значення типу long із ключем key;
– getString(String key, String defValue): повертає строкове значення з ключем key;
– getStringSet(String key, Set<String> defValues): повертає масив рядків із ключем key;
– contains(String key): повертає true, якщо в налаштуваннях збережено значення з ключем key;
– edit(): повертає об'єкт SharedPreferences.Editor, який використовується для редагування установок.

Об’єкт класу SharedPreferences.Editor використовується для керування. Він визначає такі методи:

– clear(): видаляє всі налаштування;
– remove(String key): видаляє значення з ключем key;
– putFloat(String key, float value): додає в налаштування значення типу float з ключем key;
– putBoolean(String key, boolean value): додає до налаштувань значення типу boolean з ключем key;
– putLong(String key, long value): додає до налаштувань значення типу long з ключем key;
– putString(String key, String value): додає в налаштування рядок з ключем key;
– putInt(String key, int value): додає в налаштування значення int з ключем key;

– putStringSet(String key, Set<String> values): додає в налаштування строковий масив;

– apply(): підтверджує всі зміни в налаштуваннях.

Крім загальних налаштувань, кожна activity може використовувати приватні, до яких доступ з інших activity буде неможливий[25][1)]. Метод getPreferences(MODE_PRIVATE) використовується для отримання налаштувань activity. Нижче наведено приклад коду:

```
import android.content.SharedPreferences;
//.......................
SharedPreferences settings = getPreferences(MODE_PRIVATE);
```

Тобто, на відміну від загальних налаштувань, тут не використовується назва групи налаштувань як перший параметр.

Для пошуку місцезнаходження потрібен дозвіл користувача, тому наступний код викликає вікно із запитом на дозвіл використання GPS.

```
private void requestReadLocationPermission() {
    System.out.println("Calling request location permission");
    if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        if
(ActivityCompat.shouldShowRequestPermissionRationale(this,
                Manifest.permission.ACCESS_FINE_LOCATION)) {
            // Explanation not needed, since user requests this
themself

        } else {
            ActivityCompat.requestPermissions(this,
                    new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},

MainActivity.MY_PERMISSIONS_ACCESS_FINE_LOCATION);
        }
    } else {
        privacyGuardWorkaround();
    }
}
```

[1)][25] Создание и получение настроек SharedPreferences. URL: https://metanit.com/java/android/12.1.php (дата звернення 15.10.2021).

Для зберігання налаштувань використовується метод onSharedPreferencesChanged (див. рис. 3.9).

```
@Override
public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String key) {
    switch (key) {
        case "unit":
        case "lengthUnit":
        case "speedUnit":
        case "pressureUnit":
        case "windDirectionFormat":
            setListPreferenceSummary(key);
            break;
        case "dateFormat":
            setCustomDateEnabled();
            setListPreferenceSummary(key);
            break;

        case "updateLocationAutomatically":
            if (sharedPreferences.getBoolean(key, b: false)) {
                requestReadLocationPermission();
            }
            break;
        case "apiKey":
            checkKey(key);
            break;
        default:
            if (key.equalsIgnoreCase(getString(R.string.settings_enable_notification_key))) {
                if (sharedPreferences.getBoolean(key, b: false)) {
                    requestForegroundServicePermission();
                } else {
                    hideNotification();
                }
            } else if (key.equalsIgnoreCase(getString(R.string.settings_notification_type_key))) {
                setListPreferenceSummary(key);
            }
```

Рисунок 3.9 – Код програми для роботи із налаштуваннями

4 ОПИС РОБОТИ З ПРОГРАМОЮ

4.1 Інсталяція та системні вимоги

Додаток встановлюється на пристрій користувача шляхом встановлення “apk” файлу та його автоматичного налаштування. Оскільки додаток розроблений на платформі “Android”, пристрій користувача повинен цю операційну систему на своєму пристрої. Для коректної роботи додатку необхідно мати версію операційної системи не нижче 5.0.0. Також на пристрої користувача повинен бути доступ до мережі Інтернет.

4.2 Функціонал програми

Головна сторінка програми відображає дані, які отримано за допомогою API сервісів OpenWeather та AQICN. Дані відображає згідно вказаного населеного пункту. Під час першого запуску за стандартом відображає місто London (див. рис. 4.1).

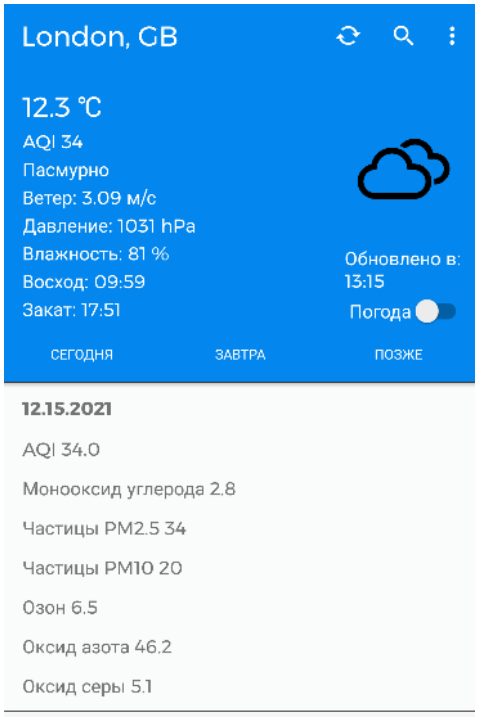

Рисунок 4.1 – Додати нове місто до БД

Після включення Switch «Погода» буде показано прогноз погоди

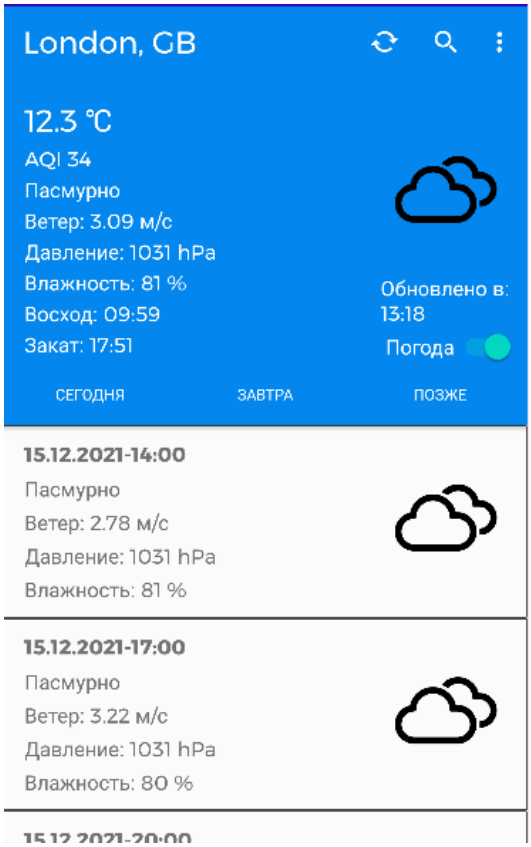

Рисунок 4.2 – Головна сторінка з прогнозом погоди

Під час пошуку, додаток може знайти декілька однакових за назвою населений пунктів. Для того щоб такого не трапилось, після натиску на поле пошуку в панелі інструментів на головній сторінці на екрані користувача з’явиться поле пошуку (див. рис. 4.3). Після, того як користувач натисне кнопку «ОК» з’явиться новий фрагмент карти з маркером на схожому місті, його назвою та країною (див. рис. 4.4).

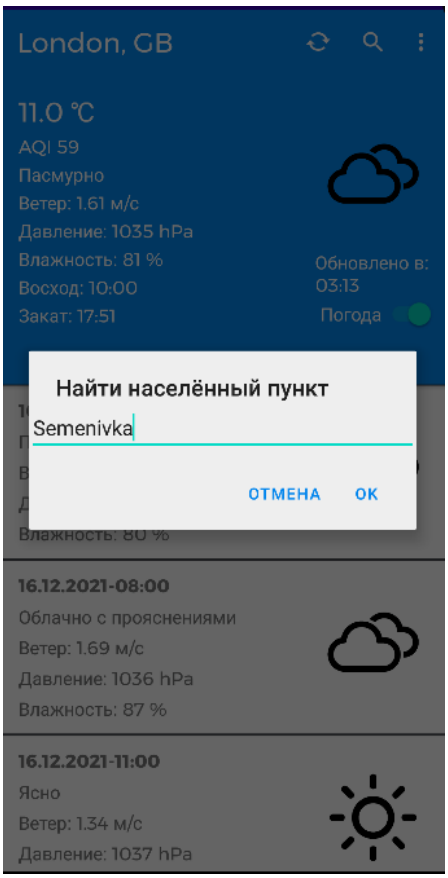

Рисунок 4.3 – Поле пошуку

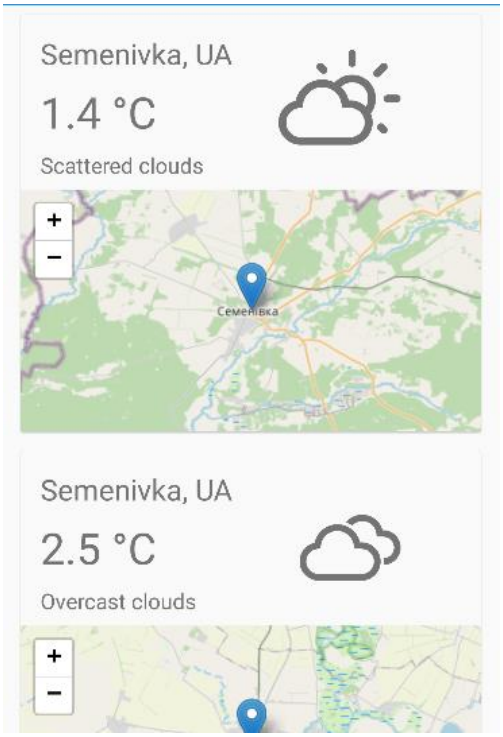

Рисунок 4.4 – Фрагмент пошуку

Після переходу на вкладку «Графики» (див. рис. 4.5) користувач має змогу переглянути зміну даних на 5 днів. Крім основних забруднень повітря, є графіки про температуру, опади, вітер та тиск.

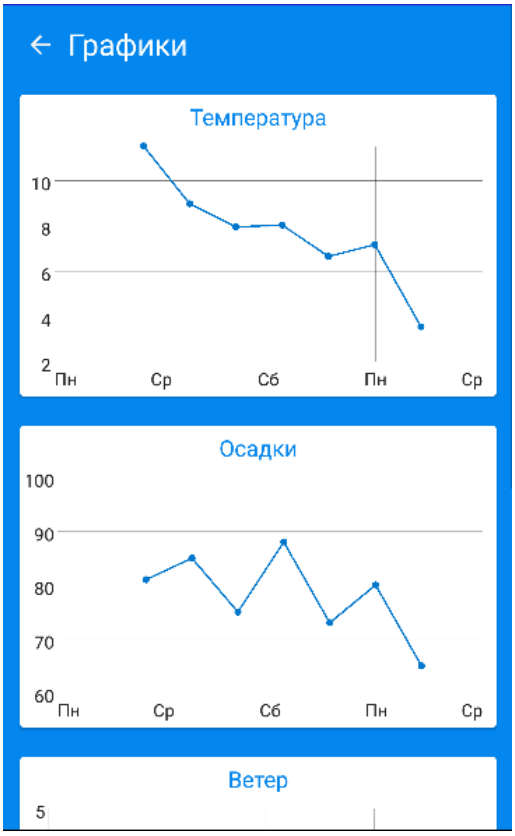

Рисунок 4.5 – Вкладка «Графики»

Побачити всі дані візуально на глобальній карті можна на вкладці «Карта». Якщо натиснути на значок місцезнаходження, то камеру відцентрує на вашої геолокації.

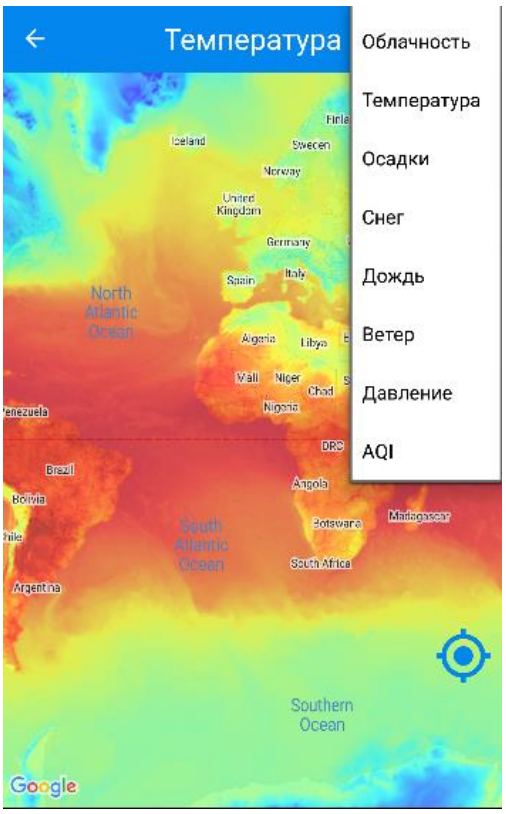

Рисунок 4.6 – Вкладка «Карта»

Рисунок 4.7 – Вкладка «Карта»

Остання вкладка – це «Настройки». Її призначення в зміні формату даних.

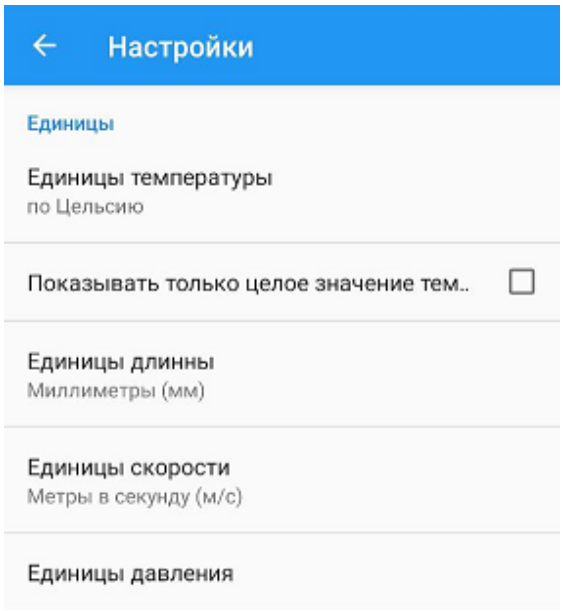

Рисунок 4.8 – Вкладка «Настройки»

ВИСНОВКИ

У магістерській роботі був проведений аналіз існуючих додатків для визначення забрудненості повітря, вивчення їх функціональних можливостей та методів реалізації. У результаті аналізу було виявлено, що існуючі додатки здебільшого мають складну структуру та здійснено їх порівняльну характеристику.

Був спроектований та розроблений програмний продукт, що повністю задовольняє вимоги. Додаток надає змогу користувачу отримати доступ до даних сервісів, що мають необхідну інформацію.

Було досліджено методи і засоби програмної реалізації продукту. В результаті чого було обрано створення додатку для мобільної платформи “Android” на основі об’єктно орієнтованої парадигми програмування. Додаток розроблений на мові “Java” з використанням платформи “Android”. Це дає змогу підвищити гнучкість та зручність системи в процесі розробки а також у процесі використання.

Мобільний додаток було випробувано та протестована і задовольняє всім програмним вимогам. Додаток призначений для людей, які прагнуть слідкувати за забрудненням у своєму місті або країні.

Отже, було розроблено додаток для мобільної платформи Android, що реалізує відображення даних із сервісів OpenWeather та AQICN. Програма вже на даному етапі є доступною для загального користування.

У подальшому є можливості до розвинення проекту. Можливо додати інші мови до інтерфейсу, покращити оптимізацію та розробити додаток для IOS.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Особливості екологічного та біологічного моніторингу. URL: https://works.doklad.ru/view/3NelM8XFMXs/2.html (дата звернення 12.08.2021).
2. Экологический мониторинг в Украине: какие данные открыты. URL: https://www.epravda.com.ua/rus/columns/2018/07/17/638718/ (дата звернення 12.08.2020)
3. Приложения в Google Play – IQAir Airvisual | Air Quality. URL: https://play.google.com/store/apps/details?id=com.airvisual (дата звернення 15.08.2021).
4. Приложения в Google Play – Индекс качества воздуха и пыльца – BreezoMeter. URL: https://play.google.com/store/apps/details?id=app.breezometer (дата звернення 15.08.2021).
5. Приложения в Google Play – Air Quality – AirCare. URL: https://play.google.com/store/apps/details?id=com.gorjan.airquality (дата звернення 15.08.2021).
6. Приложения в Google Play – Air Matters. URL: https://play.google.com/store/apps/details?id=com.freshideas.airindex (дата звернення 15.08.2021).
7. Android – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Android (дата звернення 19.08.2021).
8. Java – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Java (дата звернення 21.08.2021).
9. Java – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Java#Основные_особенности_языка (дата звернення 21.08.2021).
10. Kotlin – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Kotlin (дата звернення 23.08.2021).

11. Что такое XML. URL: https://javarush.ru/groups/posts/2287-chto-takoe-xml (дата звернення 25.08.2021).
12. Gradle – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Gradle (дата звернення 28.08.2021).
13. Android: Файл манифеста AndroidManifest. URL: http://developer.alexanderklimov.ru/android/theory/AndroidManifestXML.phpдата звернення 01.09.2021).
14. Android SDK – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Android_SDK (дата звернення 03.09.2021).
15. Android Studio: среда разработки мобильных приложений. URL: https://arduinoplus.ru/android-studio/ (дата звернення 08.09.2021).
16. Android Studio IDE от Google. URL: http://wnfx.ru/android-studio-ide-ot-google/ (дата звернення 08.09.2021).
17. IDE Eclips: за и против от ведущих программистов. URL: https://proglib.io/p/ide-eclipse-za-i-protiv-ot-vedushchih-programmistov-2019-11-02 (дата звернення 12.09.2021).
18. ANDROID STUDIO VS ECLIPSE: ПРЕИМУЩЕСТВО КАЖДОЙ ИЗ СРЕД. URL: https://kitapp.pro/android-studio-vs-eclipse-preimushhestvo-kazhdoj-iz-sred/ (дата звернення 12.09.2021).
19. IntelliJ IDEA – Вікіпедія. URL: https://ru.wikipedia.org/wiki/IntelliJ_IDEA (дата звернення 15.09.2021).
20. Что Такое API. URL: https://developer.mozilla.org/ru/docs/Learn/JavaScript/Client-side_web_APIs/Introduction (дата звернення 20.09.2021).
21. Діаграма діяльності – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Діаграма_діяльності (дата звернення 26.09.2021).
22. Maps SDK для Android: краткое руководство. URL: https://developers.google.com/maps/documentation/android-sdk/start?hl=ru (дата звернення 08.10.2021).

23. Google MAPs API в android или как работать с картами быстрее. URL: https://habr.com/ru/post/341548/ (дата звернення 08.10.2021).
24. SharedPreferences. Сохранение данных в постоянное хранилище Android. URL: https://www.fandroid.info/sharedpreferences-sohranenie-dannyh-v-postoyannoe-hranilishhe-android/ (дата звернення 15.10.2021).
25. Создание и получение настроек SharedPreferences. URL: https://metanit.com/java/android/12.1.php (дата звернення 15.10.2021).