Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет *інфокомунікацій*
(повна назва)

Кафедра *інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти *другий (магістерський)*
(рівень вищої освіти)

«*Розробка Android додатку з використанням Web бібліотеки Retrofit*»
(тема)

Виконав:

студент *2* курсу, групи *ІМІм-19-2*

Литовченко В.В.
(прізвище, ініціали)

Спеціальність *172 «Телекомунікації та радіотехніка»*
(код і назва спеціальності)

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

Освітня програма *«Інформаційно-мережна інженерія»*
(повна назва освітньої програми)

Керівник *доц. Кривенко С.А.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Безрук В.М.
(підпис) (прізвище, ініціали)

2021 р.

Не містить відомостей, заборонених до відкритого публікування

Студент _____

Керівник _____

Харківський національний університет радіоелектроніки

Факультет _____*інфокомунікацій*_____
(повна назва)

Кафедра _____*інформаційно-мережної інженерії*_____
(повна назва)

Рівень вищої освіти _____*другий (магістерський)*_____
(рівень вищої освіти)

Спеціальність _____*172 «Телекомунікації та радіотехніка»*_____
(код і назва спеціальності)

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

Освітня програма _____*«Інформаційно-мережна інженерія»*_____
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____» _____ 20____ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові _____*Литовченко Віталію Віталійовичу*_____
(прізвище, ім'я, по батькові)

1. Тема роботи «*Розробка Android додатку з використанням Web бібліотеки Retrofit*»

Затверджена наказом університету від 12 березня 2021 р. № 350 Ст

2. Термін подання студентом роботи до екзаменаційної комісії *21 травня 2021 р.*

3. Вихідні данні до роботи: *мова програмування Java, мова програмування XML, мова програмування Gradle, середовище розробки Android Studio, мережева бібліотека Retrofit, мережева бібліотека Okhttp, тип даних JSON.*

4. Перелік питань, що потрібно опрацювати в роботі:

1. Операційна система Android

2. Аналіз Структури Android додатків

3. Розробка додатку

4. Робоча документація проекту

Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускної кафедри) *слайди у форматі PowerPoint (назва роботи, мета роботи, операційна система Android, мови програмування Java та Kotlin, переваги та недоліки мов, сучасні вимоги до Android додатків, бібліотека Retrofit, розробка мобільного додатку, посилання на GitHub, висновок)* _____

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	доц. Кривенко С.А.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ.*	*12.03.2021*	
2	*Підбір літератури за темою роботи.*	*15.03.-21.03.2021*	
3	*Виконання розділу 1*	*22.03-31.03.2021*	
4	*Виконання розділу 2*	*01.04.-11.04.2021*	
5	*Виконання розділу 3*	*12.04.-20.04.2021*	
6	*Виконання розділу 4*	*21.04.-02.05.2021*	
7	*Оформлення презентаційного матеріалу, підготовка до захисту у ЕК*	*03.05.-14.05.2021*	

Дата видачі завдання 15 березня 2021 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Кривенко С.А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 60 с., 11 рис., 14 джерел.

Об’єкт дослідження – операційна система Android та бібліотека Retrofit.

Мета роботи – розробка мобільного додатка під операційну системи Android, з використанням мережної бібліотеки Retrofit.

В роботі було розроблено проект на мові програмування Java, який демонструє принцип роботи мобільних додатків із Rest API серверів, за допомогою мережної бібліотеки Retrofit.

THE ABSTRACT

Explanatory note: 60 p., 11 fig., 14 sources.

The object of the study is the Android operating system and the Retrofit library.

The purpose of the work is develop a mobile application for the Android operating system, using the Retrofit network library.

The project was developed at the Java programming language, which demonstrates the principle of mobile applications from Rest API servers, using the Retrofit network library.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ООП – (Об’єктно-орієнтоване програмування) методологія програмування, заснована на представленні програми у вигляді сукупності об'єктів, кожен з яких є екземпляром певного класу, а класи утворюють ієрархію спадкування.

DEX – (Dalvik Executable) реєстрова віртуальна машина для виконання програм, написаних на мові програмування Java, створена групою розробників Google.

IDE – (Integrated Development Environment) також єдине середовище розробки, ECP – комплекс програмних засобів, який використовується програмістами для розробки програмного забезпечення

JVM – (Java Virtual Machine) віртуальна машина Java – основна частина виконує системи Java, так званої Java Runtime Environment (JRE). Віртуальна машина Java виконує байт-код Java, попередньо створений з вихідного тексту Java-програми компілятором Java (javac).

OHA – (Open Handset Alliance) бізнес-альянс 84 компаній з розробки відкритих стандартів для мобільних пристроїв, що включає Google Inc., HTC, Intel, Motorola, Asus, Qualcomm, Samsung, LG Electronics, T-Mobile, Nvidia, Wind River Systems і інші компанії.

SDK – (Software Development Kit) набір засобів розробки, що дозволяє фахівцям з програмного забезпечення створювати додатки для певного пакету програм, програмного забезпечення базових засобів розробки, апаратної платформи, комп'ютерної системи, ігрових консолей, операційних систем і інших платформ.

REST – (Representation State Transfer) архітектурний стиль взаємодії компонентів розподіленого додатка в мережі. REST є узгоджений набір обмежень, що враховуються при проектуванні розподіленої гіпермедіа-системи.

ВСТУП

Мобільні телефони стали невід'ємною частиною життя кожної сучасної люди. Це комунікації між людьми за допомогою мессенджерів, прослуховування улюбленої музики, переглядання відео, замітки, ігри та ще багато різноманітних функції. Мобільний смартфон став копією комп'ютера, яку постійно можна мати при собі. Говорючі про будь-яку комп'ютерну систему, ми повинні пам'ятати про її оболонку – операційну систему. Серед мобільних телефонів найпопулярнішою операційною системою є Android. Ця система підтримує велику кількість пристроїв: смартфони, планшети, електронні книжки, цифрові програвачі, наручні годинники, фітнес-трекери, ігрові приставки, ноутбуки, нетбуки, смартбуки та телевізори різних виробників. Розробка під операційну систему Android у сучасному світі є дуже актуальною темою. Є багато інструментів та мов програмування для створення мобільних додатків. Серед найпопулярніших мов можна виділити Java та Kotlin. Майже кожний сучасний додаток повинен вміти працювати із інтернет мережею. Сервер відповідає за схоронність та актуалізацію даних з додатку серед усіх користувачів, цього додатка. Кожний додаток, мобільний або ж на комп'ютері, спілкується із сервером за допомогою API цього серверу. Існують спеціальні бібліотеки, для мобільної розробки, за допомогою яких додаток може запитувати, або посилати дані на сервер. Найпопулярнішою бібліотекою, для розробки під операційну систему Android, є Retrofit.

Ця робота показує процес створення мобільного додатку, який буде відправляти запит на сервер та приймати від нього відповідь, саме завдяки цій бібліотеці.

1 ОПЕРАЦІЙНА СИСТЕМА ANDROID

1.1 Створення ОС Android

ОС Android та IoS – це дві найпоширеніші операційні системи для мобільних телефонів. Згідно з даним World Mobi за 2020 рік, 70% ринку мобільних операційних систем займає саме Android.

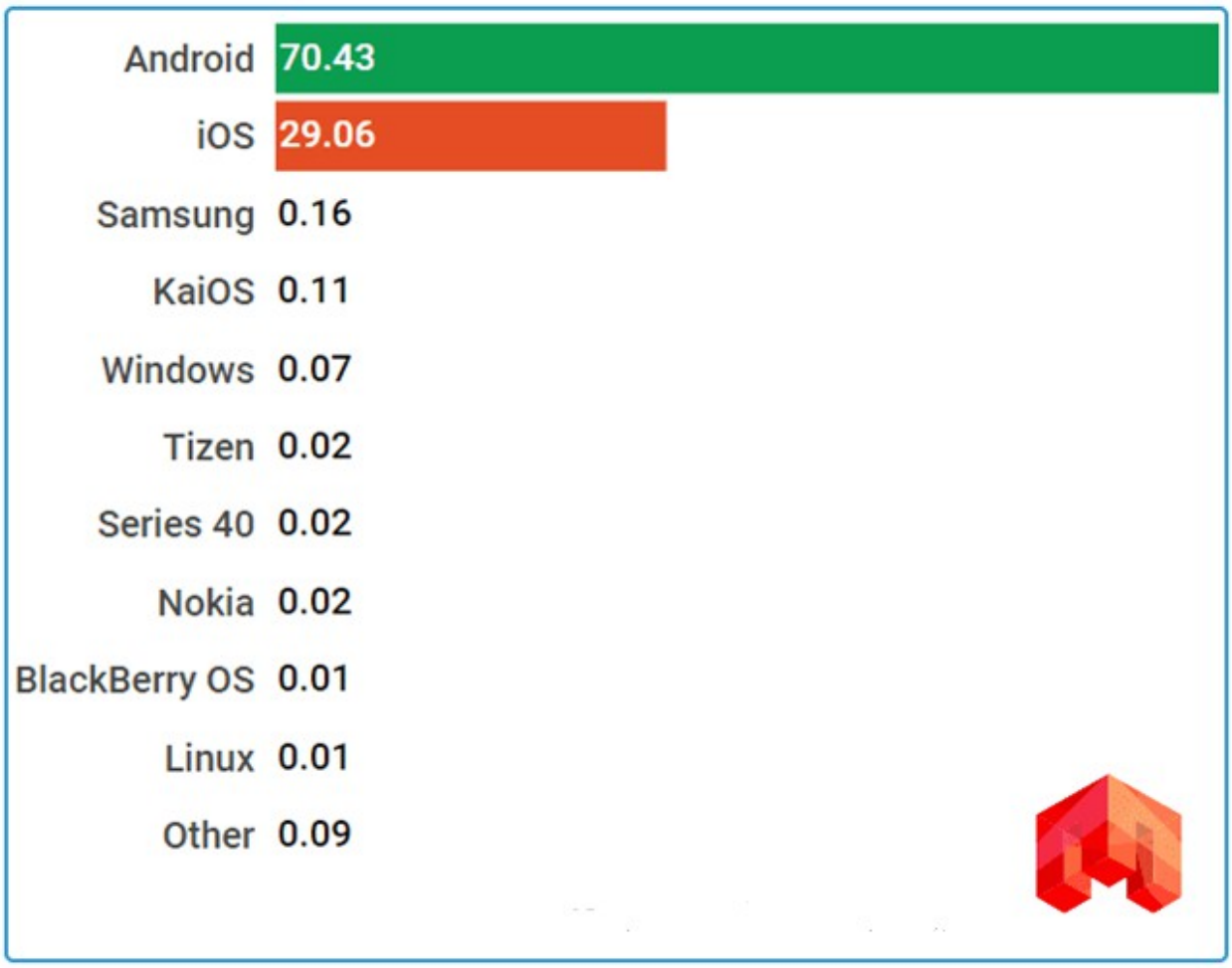

Рисунок 1.1 – Статистика встановлених мобільних операційних систем

На початку свого шляху операційна система Android була створена компанією Android Inc. Починаючи з 2003 року вони почали розробку операційної системи для телефонів та фотоапаратів. У 2005 році кампанія Google за 50 мільйонів доларів придбала Android Inc і вже через два роки

оголосила про створення Open Handset Alliance, який повинен був займатися розвитком операційної системи. Перший SDK та емулятор Android для комп'ютера з'явились вже через дві неділі після створення альянсу.

Першими учасниками консорціуму OHA стали 34 компаній, серед яких більшість були оператори мобільного зв'язку (NTT DoCoMo, T-Mobile , Sprint Nextel, Sprint Nextel, T-Mobile China Mobile), виробники мобільних телефонів (LG, Samsung, HTC, Motorola) та виробники напівпровідників (Intel, Broadcom, Nvidia, SiRF, Synaptics, Qualcomm, Texas Instruments). Пізніше до Open Handset Alliance приєдналися інші великі компанії, такі як ASUSTek, Sony Ericsson, Vodafone, Garmin, Ericsson, Toshiba, Huawei, Acer, Alcatel, Lenovo, MediaTek. На даний момент у цьому альянсі знаходяться вже 84 компанії. Перші версії операційної системи Android мали такі складні назви, як «m3-rc20a», «m3-rc37a», «m5-rc14», але пізніше їх змінили на кодові імена. Сама схема кодування була придумана трохи пізніше, тому перший білд мав назву CRA29. Для Android 1.0 була дана назва - невеликого десерту Petit-Four. Для версії 1.1 використовувалась таж сама назва. Смартфон T-Mobile G1 був першим операторським смартфоном, який використовував операційну систему Android 1.0. Виробником цього смартфону була компанія HTC Dream. Коли все ж таки було прийнято рішення перейти до нової системи кодувань імен, її розпочали з третьої букви латинського алфавіту – C, тому що це була вже третя версія операційної системи Android.

1.2 Хронологія виходу версій Android.

– 23.09.2008 – Вихід першої версії операційної системи Android, яка мала назву Base. На той час вже були розроблені додатки для YouTube, Gmail та Google Maps.

– 10.02.2009 – Вийшло перше оновлення для Base версій, його назвали Base_1.1, у цій версії була надана можливість зберігати вкладення MMS.

– 30.04.2010 – Вийшла нова версія Android 1.5 CupCake. Одном із нововведень цієї версії стала власна екранна клавіатура та автоматичний перехід між альбомною та портретною орієнтацією екрана.

– 15.09.2009 – Стала доступна версія Android 1.6, вона мала назву Donut, в цій версії з'явилась можливість конфігурувати власні VPN мережі.

Починаючи з версії Android 2.0 не кожне оновлення системи мало випуск нової основної версії. Замість цього починаючи з жовтня 209 року, після кожного оновлення для основної версії почали з'являтися декілька підверсій, проте ім'я основної версії не змінювалось. Для кожної з нових версій розробники вводили нові технічні можливості, змінювали дизайн та інтерфейс користувача.

– 26.10.2009 – Вийшла нова оновлена версія Android 2.0 з назвою Éclair. В ній була додана підтримка світолдіодного спалаху та презентована можливість цифрового приближення.

– 13.12.2009 – Була випущена версія Android 2.0.1, в якій виправили декілька помилок.

– 17.01.2010 – З'явилось оновлення до версії 2.1. Основною новинкою стала можливість використання анімованих шпалер.

– 20.05.2010 – Вийшов Android 2.2, в цій версії була додана можливість переносити мобільні додатки на CD-карту, а також об'єм оперативної пам'яті став більше ніж 256 Мбайт.

– 06.12.2010 – З'явилась нова версія Android 2.3 – Gingerbread. Починаючи саме з цієї версії була додана підтримка NFC, а також був додан власний операційний менеджер завантажень.

– 23.02.2011 – Вийшла версія Android 2.3.3, в якій була додана можливість оснащення двоядерним процесором Android пристроїв.

– 29.04.2011 – Разом із виходом оновлення 2.3.4 з'явилась можливість відео та голосового чату завдяки додатку Google Talk.

– 23.05.2011 – Вийшла глобальна версія Android 3.0, її назва – Honeycomb. У цій новій версії, було покращено інтерфейс системи та

оптимізована його робота для планшетів. Крім того з’явились нові функції в браузері Google Chrome.

– 10.05.2011 – На систему Honeycomb вийшло перше оновлення, у якому був доданий режим USB Host.

– 19.10.2011 – Стала доступна система версії Android 4.0 під кодовою назвою Ice Cream Sandwitch. У цій версії була додана система розпізнавання обличчя для розблокування.

– 27.06.2011 – Вийшла Jelly Bean, вона була версією Android 4.1. Це оновлення значно покращило інтерфейс користувача, крім того у систему було додано платіжну службу Google – Google Wallet. А через декілька тижнів була додана можливість за допомогою жестів керувати повідомленнями.

– 31.10.2013 – Була випущена версія Android 4.4, яка мала назву KitKat. Разом з цією версією була додана можливість користуватися замість стандартних смс додатком Google+ Hangouts.

– 03.11.2014 – З’явилась нова глобальна версія Android 5.0 із назвою Lollipop. У ній були додані зміни панелі повідомлень, а також ця версія досі використовується для більшості розумних годинників.

– Android 6.0 був презентований на телефонах моделі Nexus, а потім і на телефонах інших виробників. В цій версії була покращена автономна робота акумулятора: система автоматично завершала фонові процеси, завдяки цьому батарея працювала довше. Також починаючи з цієї версії була реалізована швидкий заряд пристрою за допомогою USB-стандарту Type C.

– 22.08.2016 – Офіційно стала доступна нова версія Android 7.0. Ця версія принесла нововведення в яких було здійснено покращення безпеки, продуктивності, була додана можливість оновлення у фоновому режимі. Також був доданий режим Splitscreen (розділення екрану) – це коли 2 додатка можуть працювати незалежно один від одного поділяючи екран телефону навпіл, тільки якщо додаток підтримує такий режим. Буда додана можливість відповідати на повідомлення мессенджерів з панелі повідомлень, не завантажуючи при цьому сам додаток.

– В серпні 2017 року – з’явилася можливість завантажити версію Android 8.

– 6.08.2018 - дев'ятою версією Android став Pie, який вийшов, на таких смартфонах, як Essential Phone та Google Pixel. Трохи пізніше Google зробили заяву, що випустять більш легку версію цієї системи під назвою Android Pie Go Edition. Станом на вересень 2019 року, згідно з даними компанії Google, 27,77% всіх пристроїв працюють саме на версії Android Pie, що робить її доволі популярною.

– Наступною версією Android стала десята версія, яка привнесла у собі деякі зміни. Вона отримала назву Android Q, а також компанія заявила, що більше не буде використовувати літери латинського алфавіту, а залишать лише цифрові значення.

1.3 Мови програмування для платформи Android

Операцій система Android дозволяє розробляти додатки на великій мові програмувань. Це такі мови, як Basic, Python, PhoneGap, Kotlin, Java та C/C++.

Проте незважаючи на таку велику варіативність можна виділити лише три основних мови програмування це – Kotlin, Java та C/C++. На мову Java посилається більшість офіційної документації від компанії Google, велика кількість Android додатків вже розроблена саме на цій мові, на якій і надалі буде йти їх підтримка. Знайти бібліотеки під Java також дуже просто. Проте оскільки Java є об'єктно-орієнтованою мовою в ней є декілька таких особливостей як конструктори, виключення, що доволі часто призводить до падіння програми під час її роботи, та інші особливості, на які треба звертати уваги при програмуванні. Втім у Java доволі легко читається та структурується її код, особливо якщо при його розробці дотримуватися норм доброго коду. При розробці на Java під операційну систему Android, використовуються не тільки класи Java, але і певні файли на мові XML. Зазвичай ці файли несуть у собі основну інформацію про додаток, та про системи автоматичної збірки

проекту, такі як Ant, Maven або ж Gradle. В цих збірках використовується такі мови програмування, як XML, POM та Groovy. За замовченням у проектах використовують саме Gradle. Для розробки візуального інтерфейсу використовується мова XML.

1.3.1 Використання Java в Android

Java - це об'єктно-орієнтована мова програмування, строго типізована, загального призначення, була розроблена компанією SunMicrosystems, яку потім придбала компанія Oracle. Подальшим розвитком та розробкою Java займається Java Community Process.

Зазвичай усі програми Java транслюються в спеціальний байт-код, завдяки цьому була реалізована кросплатформеність, тому що код Java можна виконати на будь-якій машині, на якій встановлена реалізація віртуальної Java-машини.

На сам перед основною перевагою Java-розробників, над іншими розробниками під Android, є те, що API Android дуже схожий на API самої Java. Операційна система Android підтримає майже всі доступні у J2SE SDK класи та бібліотеки. Хоча Google і надає API Java для розробки коду, який потім компілюється у файли класів, проте на цьому схожість закінчується. Операційна система Android не використовує віртуальну машину Java (JVM). Для виконання коду в Android є своя віртуальна машина – Dalvik. Вона не є істинною JVM, проте має велику схожість. Dalvik не працює з Java -байткодом, вона компілює файли класів до формату DEX (Dalvik EXecutable).

Після того як файли з кодом перетворюються до формату DEX, вони об’єднуються, разом із іншими ресурсами проекту, та перетворюються в пакети Android (APK). Файли з розширенням .apk використовуються для інсталяції та поширення на різних пристроях. Головне що слід пам’ятати, так це те що в базовій бібліотеці класів, на яких працює віртуальна машина Dalvik, заходиться підмножина Apache Harmony, внаслідок чого Dalvik не може підтримувати увесь API J2SE.

Операційна система Android, за замовчуванням, надає кожному із додатків, у своїй системі, унікальний ідентифікатор. Це робиться для того щоб після запуску додатків, кожен з них виконується у своєму власному процесі, та у своїй власній віртуальній машини. Завдяки цьому всі додатки Android працюють ізольовано один від одного, у міру необхідності операційна система сама керує процесом запуску, або ж зупинкою процесів додатка. Це дуже схоже на принцип розробки додатків в J2ME, де є певні права доступу.

При першому запуску програми, вона запитає права, які необхідні їй для коректної роботи, це можуть бути запити для доступу до інтернету, системним ресурсам або телефонній книзі. Всі ці права доступу повинні бути описані у файлі маніфесту Android вашого додатка. Цей файл представляє собою файл з кодом на мові XML, у якому перераховані компоненти програми, та їх налаштування.

У кожного Android додатка є чотири основних компонента, це – сервіси, активності, широкомовні приймачі та постачальники контенту. Найчастіше використовувані з них це активності. Вони відповідають окремій екранній формі із вашого Android додатка.

Операційна система Android, як і Java, створює об’єкти активностей сама. За їх організацію відповідає, такий клас як System. Активність запускається за допомогою метода startActivity(), у який треба передати об’єкт Intent у якості параметру. Після цього виклику, клас System створить новий об’єкт активності, або ж якщо він уже був створений, просто повторно використає старий.

Як і у Java, в операційній системі Android є автоматична збірка сміття, яка відповідає за повторне використання пам’яті системи. Система Android надає можливість перевизначити події життєвого циклу, завдяки яким можна втручатися в автоматичний процес запуска, зупинки або знищення додатка.

1.3.2 Мова програмування Kotlin

У травні 2017 року, на Google I/O була офіційно представлена мова програмування Kotlin. Через 2 роки Google визнали Kotlin кращою мовою

програмування для розробки Android-додатків. Після чого мова програмування Java відійшла на другий план. На сам перед це означало що розробка нових інструментів, таких як бібліотеки та функції Android Studio, буде націлена саме на мову Kotlin. Kotlin отримав краще від таких мов, як Scala, Java та TypeScript. Серед його основних переваг можна виділити автоматичне виділення типів даних, функції-розширення та підтримка, окрім об'єктно-орієнтованої, функціональної парадигми. Проте найголовнішою перевагою Kotlin над мовою Java, є захист від виключення NullPointerException. Саме через це його популярність різко виросла, згідно з даними опитування на StackOverflow. В 2019 році Kotlin увійшов до п'ятірки найбільш улюблених мов програмування серед спільноти. Зазвичай дуже великі проект, з великою кількістю написаного legacy коду, розроблені на мові Java, проте нові проекти все частіше і частіше пишуться саме на Kotlin, тому для сучасного Android-розробника необхідно знати ці дві мови програмування.

1.3.3 Розробка нативних библиотек на C/C++

Мови C/C ++ - це більш низько рівневі мови програмування, на відміну від Java та Kotlin, проте вони також підтримуються Android Studio з використанням Java NDK. Завдяки цьому можна писати нативні додатки, що використовується при створенні ігор або інших ресурс затратних програм. Android Studio підримує розробку на C / C ++ через Android NDK (Native Development Kit). Це означає, що код, написаний на C/C++ не буде запускатися на Java Virtual Machine, а безпосередньо через сам девайс, що дає вам набагато більше контролю над такими елементами системи, як жести, пам'ять, сенсори і т.д., а також це дає можливість використання Android-пристрої на максимальні ресурси.

1.4 Взаємодія Android додатків із мережею

Одною з основних вимог до сучасних мобільних додатків є синхронізація своєї роботи із сервером. Він зберігає у собі дані різних користувачів додатка та відповідає за їх актуальне оновлення у інших користувачів. Це реалізовано завдяки централізації бізнес логіки додатка на стороні сервера. Сходячи з цього можна зробити висновок що Android додаток повинен вміти працювати з даними серверу та обмінюватись з ним інформацією.

На сьогоднішній день існує багато різноманітних технологій, які дозволяють проводити обмін даними з сервером. Процес синхронізації можна домогтися за допомогою socket з'єднань або HTTP запитів. Перший підхід використовується під час розробки додатків для персональних комп'ютерів, коли необхідно підключитись до серверу. Другий підхід більш звичний для веб-розробки. Для забезпечення взаємодії роботи додатка із API серверу було вироблено архітектурний стиль Rest, який забезпечує взаємодію компонентів розподіленого додатку у мережі.

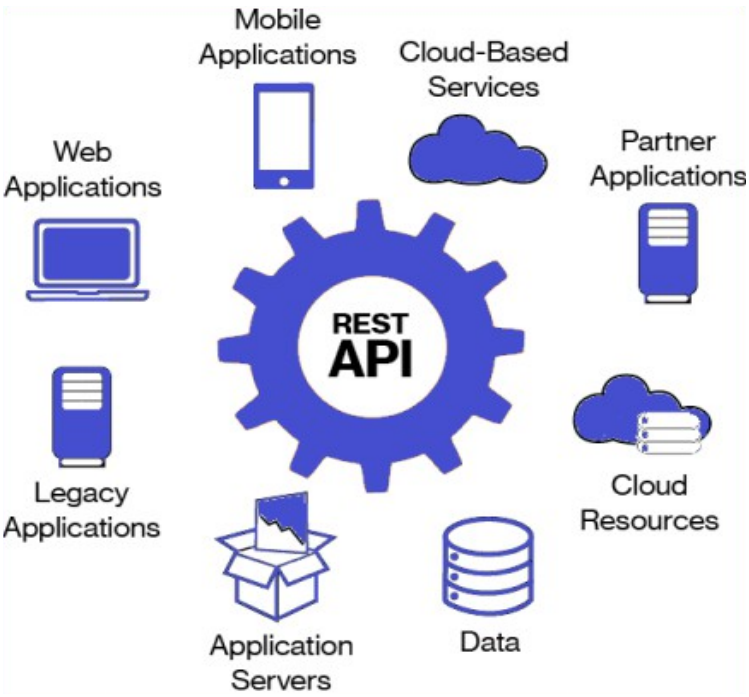

Рисунок 1.2 – Можливості архітектури Rest API

1.4.1 Постановка задачі

Метою даної роботи є показ взаємодії Android додатка із API серверу для отримання даних та їх коректного відображення у додатку, за допомогою мережної Android бібліотеки Retrofit. Для реалізації даної мети було обрано розробити простий додаток, який буде відправляти запит на сервер про погоду в обраному користувачем регіоні (городі або країни) та отримувати на нього відповідь, яка буде відображатися в візуальному інтерфейсі користувача. Для цього був обраний сайт Openweathermap (його сайт https://openweathermap.org/), який надає відкритий API з даними про погодні умови для мобільних додатків. У якості основної мови програмування була обрана Java. XML служить для розробки графічного інтерфейсу користувача. За мережну взаємодію додатка та API сайту відповідає бібліотека Retrofit2. Середою розробки була обрана Android Studio, тому що вона є офіційною середою розробки під операційну систему Android від компанії Google.

2 АНАЛІЗ СТРУКТУРИ ANDROID ДОДАТКІВ

2.1 Компоненти Android додатка

Додатки для Android в своїй роботі використовують вікна, аналогічно вікна Windows, однак у цій системі вікна носять іншу назву – Activity. Як і в операційній системі Windows, кожне вікно має свої особливості та життєвий цикл. Зазвичай Activity асоціюється з окремим екраном або вікном додатка, перехід між вікнами буде відбуватися як перехід від однієї Activity до іншої. Додаток може мати одну та більш Activity. Усі Android додатки мають строго типізований системою життєвий цикл роботи. Коли користувач запускає додаток система дає цьому додатку високий пріоритет. Кожен окремий додаток запускається у вигляді окремого процесу, що дозволяє системі надавати одним додаткам більш високий пріоритет ніж іншим. Завдяки цьому, наприклад, при роботі з одними додатками Android дозволяє не блокувати вхідні дзвінки. Після закінчення роботи з додатком система звільнює усі пов’язані з цим додатком ресурси, та надає йому низько пріоритетний статус, та за необхідністю закриває його. Усі об’єкти Activity, які є у додатку, керуються системою у вигляді стеку, який має назву back stack. Коли йде запуск нової Activity, вона розміщується на поверх стеку та виводиться на екран пристрою, доки не з’явиться нова Activity. Коли користувач закінчує роботу з Activity, наприклад закриває додаток, вона видаляється із стеку, а Activity яка була другою у стеку поновлює свою роботу.

2.2 Життєвий цикл Activity

Загальна схема роботи Activity представлена на рисунку 2.1.

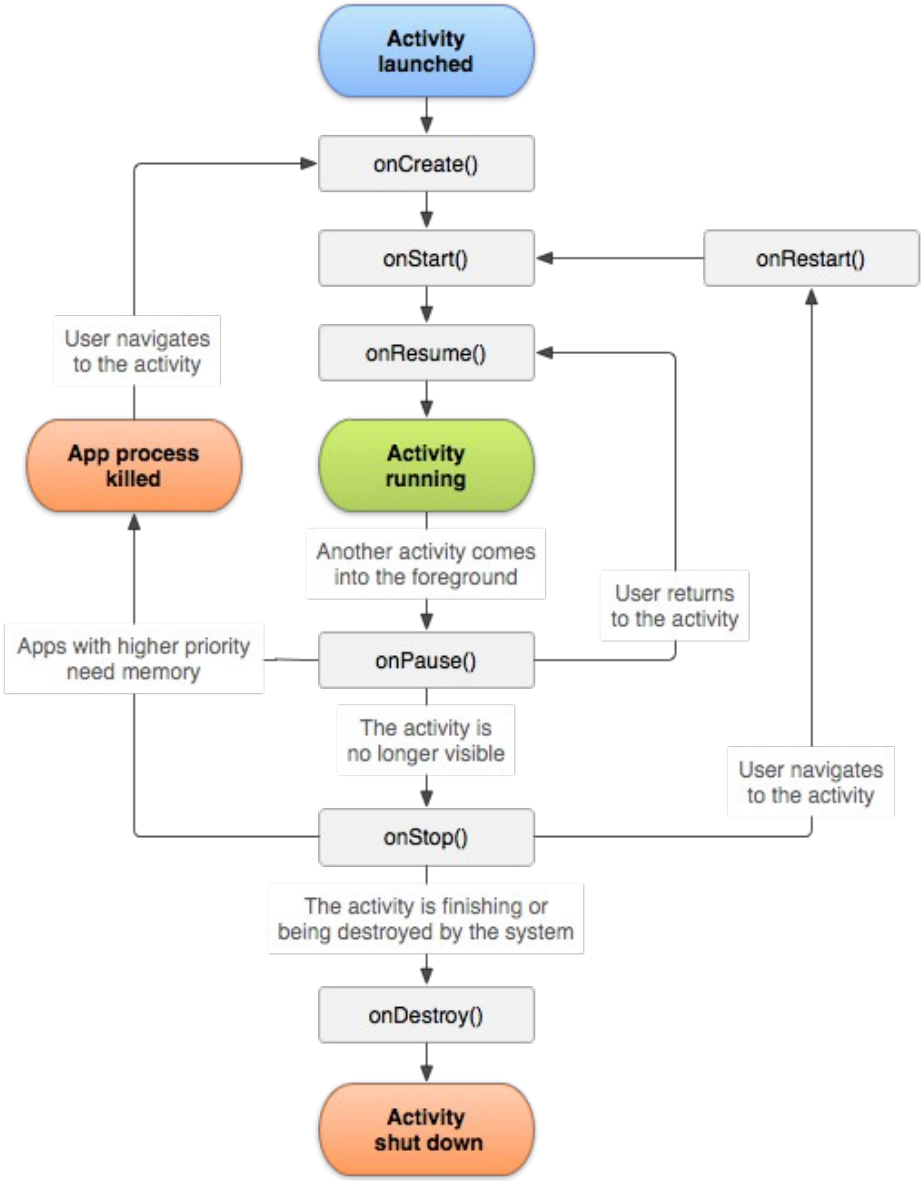

Рисунок 2.1 – Життєвий цикл Activity

Після запуску activity проходить через цілий ряд подій, які повинна обробити система. Для того щоб контролювати цей процес обробки існують спеціальні методи.

onCreate() - це перший метод з якого і починається життєвий цикл кожної activity. У результаті виконання цього методу активність переходить у стан

Created. Цей метод повинен обов’язково бути перевизначений у класі, який відповідає за activity. У цьому методі також роблять першу настройку додатка, а саме створюють об’єкти візуального інтерфейсу, та пов’язують їх унікальними ідентифікаторами. Цей метод приймає, у якості параметра, об’єкт типу Bundle, який зберігає у собі попередній стан activity, якщо цей стан було збережено. Якщо activity буде створена наново, то цей об’єкт буде мати значення null. Якщо ж activity вже була створена та знаходиться в призупиненому стані, то об’єкт bundle буде зберігати у собі інформацію про стан activity.

Після закінчення роботи методу onCreate(), activity переходить у наступний свій стан, а саме у стан Started та викликає метод onStart(). В цьому методі здійснюється підготовка до виводу activity на екран пристрою. За звичай, цей метод не має потреби щоб його перевизначали, тому що майже всю роботу робить влаштований код. Після завершення роботи цього методу activity може бути відображена на екрані користувача, після цього викликається метод onResume(), а сама activity переходить до стану Resumed.

Після того я цей метод почав свою роботу activity з’являється на екрані, а користувач вже може починати з нею взаємодіяти. У цьому стані activity залишається доти поки не втратить свій фокус.

Якщо користувач вирішить перейти до іншої activity то система автоматично викликає метод onPause(), а сама активність переходить до стану Paused. У цьому стані можна почати призупиняти процеси, наприклад такі як анімацій, роботу камери, аудіо або ж відео. Також можна звільняти ресурси які використовує додаток, для підвищення продуктивності системи. Проте треба брати до уваги, що у цьому стані activity все ще видно на екрані, тому на обробку цього методу залишається дуже мало часу. У цьому методі не треба зберігати якісь дані, особливо якщо вони пов’язані з робою мережі, або з базою даних. Такі дії найкраще виконувати у методі onStop().

Після виконання методу onStop() activity стає невидимою, вона не відображується на екрані, проте вона все ще активна. Якщо користувач все ж

таки вирішиться повернутися до цієї activity то система Android викличе метод onResume() і activity знову з’явиться на екрані.

Проте система іноді може і сама завершити роботу activity, якщо вона побачить що для роботи активних додатків треба виділити більше пам’яті.

У стані Stopped activity залишається у пам’яті пристрою, вона зберігає стан усіх своїх елементів інтерфейсу.

Якщо після виклику методу onStop() користувач знову вирішить повернутися до цієї activity то система Android викличе метод onRestart(). Якщо activity завершить свою роботу то викричеться метод onDestroy().

Цей метод завершує життєвий цикл activity. Він викликається також якщо система вирішила завершити роботу activity через конфігураційні причини.

Також слід пам’ятати, що при зміні орієнтацій екрану, система завершує activity, а потім створює її заново викликаючи перший метод – onCreate().

2.3 Передача даних за допомогою формату JSON

JSON (Java Script Object Notation) – це найпоширеніший формат обміну даним. Він дуже зручний для написання та читання як людиною так і комп’ютером. JSON був оснований на підмножині такої мови програмування, як JavaScript визначеного у стандарті ECMA-262 3rd Edition. У 1999 році JSON був остаточно визнаний незалежним мів мовної реалізації. Він використовує угоди, які нагадують С-подібні мови програмування, такі як С, С++, Perl та Java. Саме ці властивості роблять JSON ідеальною мовою для обміну даними.

Незважаючи на походження від мови JavaScript, формат JSON вважається незалежним та може використовуватися майже з усіма мовами програмування. Для багатьох мов існує готовий код, для створення даних у форматі JSON та їх подальшої обробки.

Завдяки своїй лаконічності, у порівняні із мовою XML, формат JSON може бути більш вигідним для сериализации складних структур. Йому також

знайшлось використання у веб-додатках, його використовують для обмін даними між сервером та браузером, а також проста між серверами.

У JSON значення повинні бути одним з таких типів:

- String
- Number
- Object (Json object)
- Array
- Boolean
- Null

2.4 Мережева бібліотека Retrofit

Retrofit – це безпечний HTTP-клієнт для Android розробки. Його модернізація дуже легко дозволяє підключитися до веб-служби REST за допомогою перекладу API сайту у інтерфейси Java або Kotlin. Retrofit дуже потужна бібліотека, вона дозволяє з легкістю використовувати такі типи даних як XML або JSON, які потім можна перетворити у прості POJO об’єкти для Java. Бібліотека дозволяє виконувати такі запроси як GET, POST, PUT, DELETE та PATCH.

Бібліотеку Retrofit розробили автори з компанії «Square», вони також написали ще багато дуже корисних бібліотек, серед яких є такі як Okhtp та Picasso.

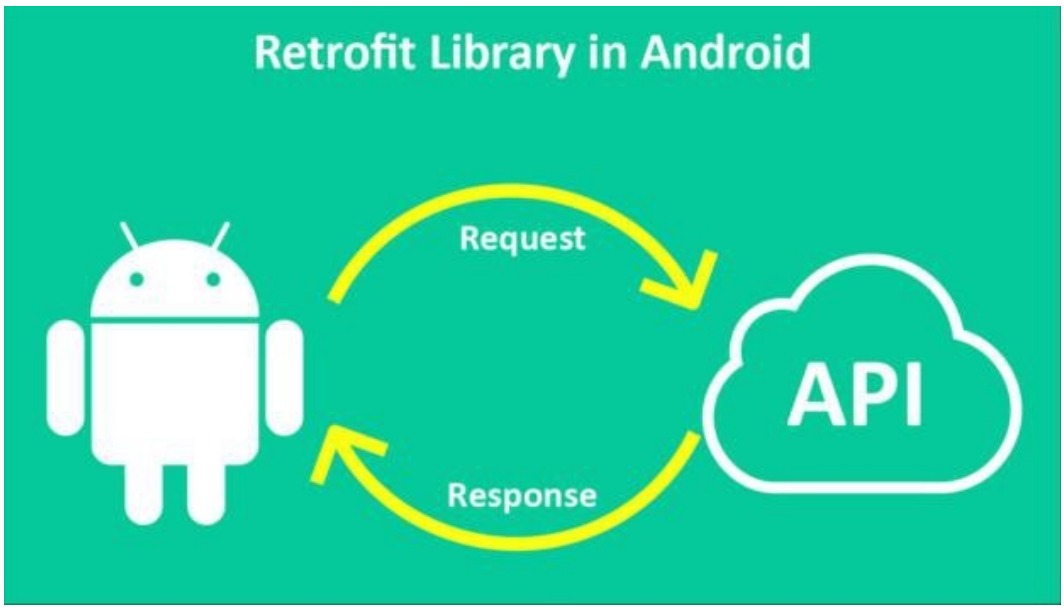

Рисунок 2.2 – Мережева бібліотека Retrofit

2.5 Принцип роботи бібліотеки

Retrofit також може працювати у синхронному режимі та асинхронному, що дозволяє легко працювати з потоками. Okhttp автоматично підключається до бібліотеки Retrofit тому її не треба власноруч прописувати. Бібліотека може працювати з XML та GSON за допомогою використання спеціальних конверторів, які треба вказувати окремо у файлі build.gradle. Конвертори потрібні щоб було легше перетворювати данні у спеціальних форматах у прості Java об’єкти. Найпоширеніші конвертори це Gson, Moshi, Simple XML, Wire та Jackson. Проте можна також створювати свій конвертор за допомогою абстрактного класу Converter.Factory. Для основної роботи з Retrofit знадобляться три основних класа:

– POJO (Plain Old Java Object) або його ще називають як Model Class. Цей клас потрібен щоб створити на основі відповіді від сайту Java клас, з об’єктами аналогічними відповіді сайту. POJO класи можна створювати власноруч, або за допомогою готових веб-сервісів, які працюють в автоматичному режимі. Цей клас зазвичай використовує анотацій, які

допомагають уникнути помилок. Список анотацій залежить від використовуваного вами типу контейнеру.

– Для роботи з адресою за допомогою команд GET, POST та інші, треба зроби інтерфейс, який буде містити у метод, за допомогою якого і буде реалізовано звернення до API сайту. Запроси розміщуються в узагальненому класі Call<>, з вказаним типом відповіді яка вам потрібна.

– Третій обов'язковий клас для коректної роботи, це клас який буде оброблювати результати відповіді серверу на наш запит. Для синхронного запиту треба використовувати метод Call.execute(), для асинхронного – метод Call.enqueue(). У результаті бібліотека Retrofit, робить запит, та якщо отримає відповідь то буде проведено розбір відповіді. Далі треба буде лише викликати методи потрібних вам класів для отримання даних.

Основна частина роботи буде відбуватися в методі onResponse(), а обробка помилок в методі onFailure(). До помилок можна віднести неправильну адресу сервера, неправильний формат моделі класу, або некоректний формат даних. HTTP-коди серверу не відносять до помилок.

Метод onResponce викликається завжди, навіть якщо запит був неуспішний. Клас Response має метод isSuccessful(), який використовується для обробки успішного запросу. У помилкових ситуаціях обробку можна здійснити у методі errorBody() класу ResponseBody.

3 РОЗРОБКА ДОДАТКУ

3.1 Початкове налаштування середовища розробки проекту

Для демонстрації взаємодій Android додатків із API сайтів за допомогою мережної бібліотеки Retrofit було обрано написати погодний додаток. Цей додаток повинен буде відправляти запит користувача з обраним містом або країною на сервер та повертати результат відповіді в форматі JSON, який потім буде перетворений в Java об’єкти та виведений у графічну частину додатка. Середою розробки буде IDE Android Studio. Для створення проекту потрібно обрати його назву, вказати package name (зазвичай не зворотній URL сайту компаній), далі обираємо місце знаходження проекту на диску.

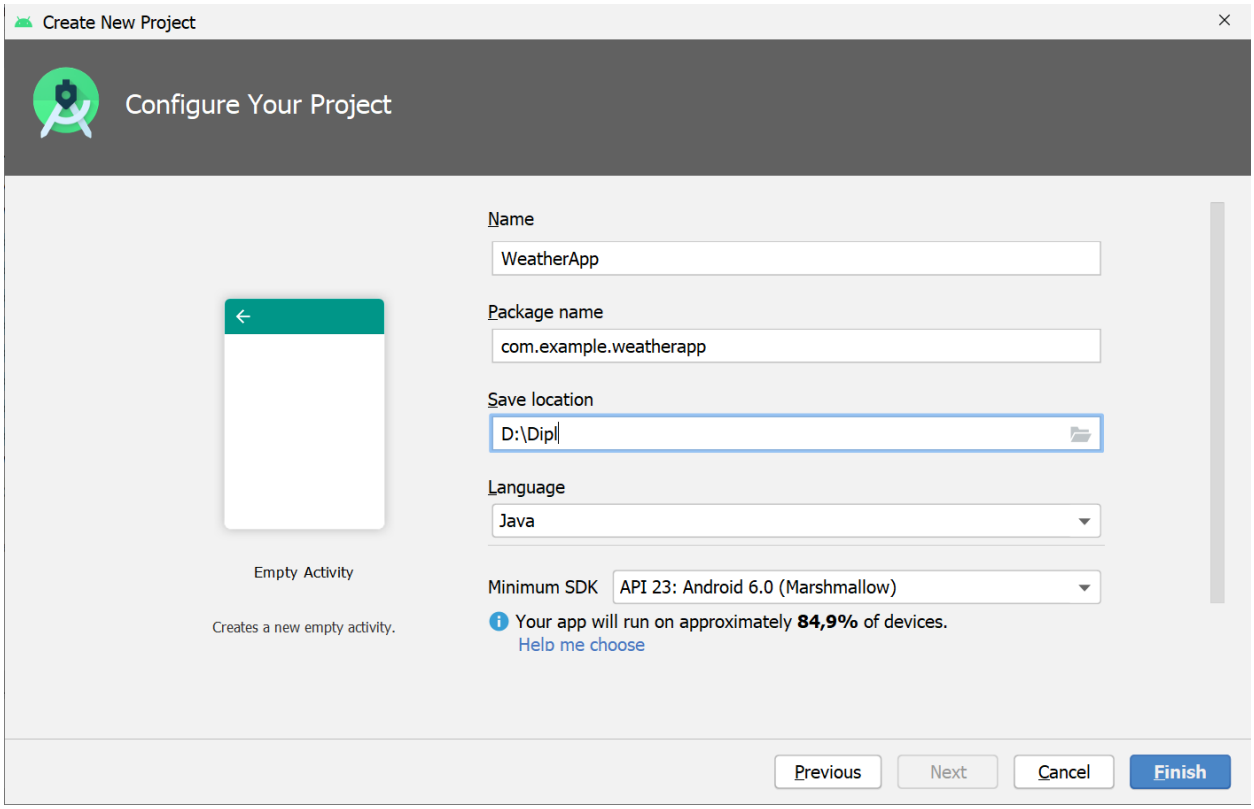

Рисунок 3.1 – Створення нового проекту в Android Studio

Також Android Studio дозволяє одразу обрати мову програмування, на цій мові буде автоматично створено та налаштовано перший та основний клас

додатку – MainActivity, а також до нього буде створено візуальну частину activity_main.xml. Наступним кроком треба обрати мінімальну версію SDK проекту. SDK це набір бібліотек за допомогою який відбувається розробка додатків. Зазвичай нова SDK це додавання нового функціоналу до більш старої версії, тому мобільний телефон з більш новою SDK буде запускати додатки, які написані на старшій версії, проте стара версія не зможе запускати більш нову. Тому обирати SDK треба виходячи з статистики їх популярності серед носіїв мобільних телефонів. Google публікує данні про статистику популярності SDK. Обираємо API Level 23, ця версія покриває 84,9% усіх мобільних телефонів. Далі Android Studio почне білдити проект.

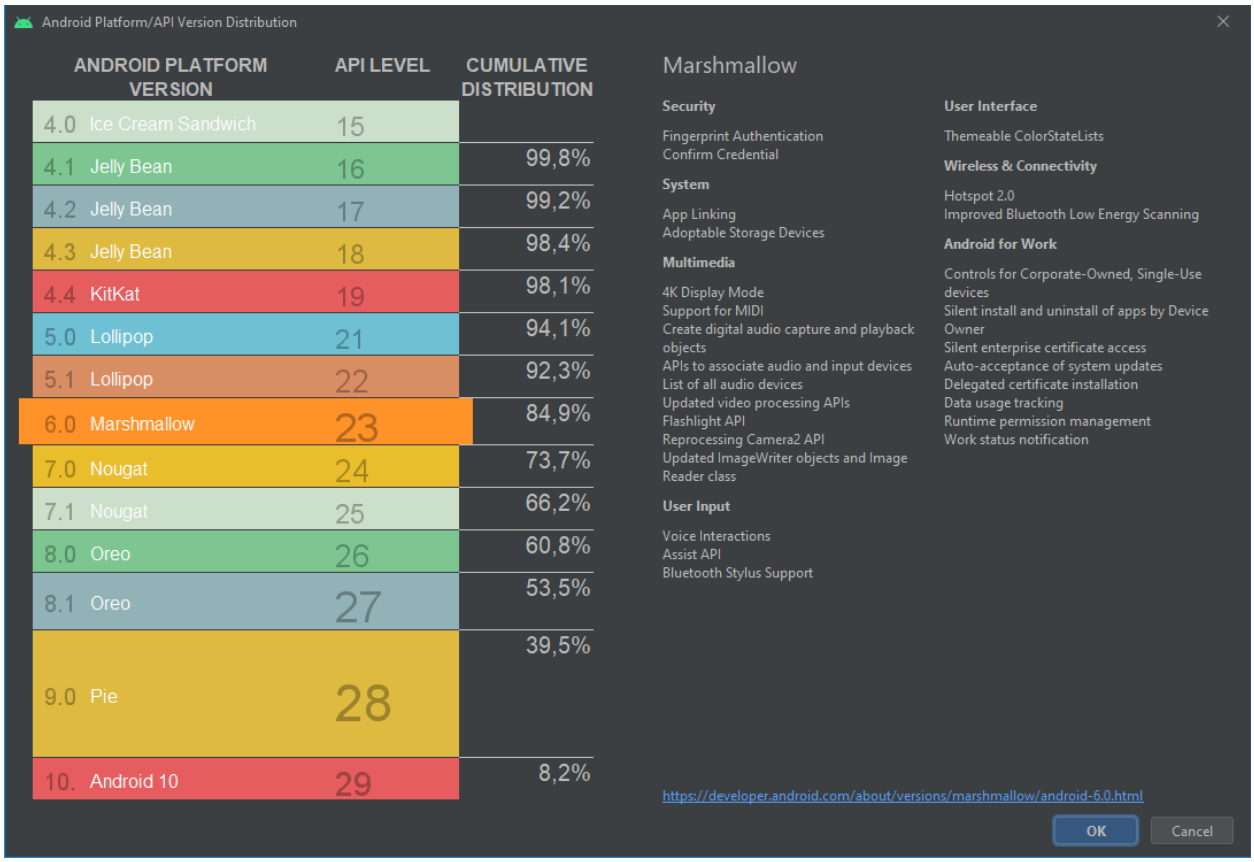

Рисунок 3.2 – Рейтинг популярності систем Android

Android Studio робить структуру папок, та основні файли. У папці Java лежать Java класи. Ці класи зазвичай відповідають за бізнес-логіку додатку, або ж це Java класи які пов’язані з Activity. MainActivity це автоматично створений

клас основної активності додатка, при необхідності його можна змінити на інший. Файл AndroidManifest.xml це необхідний системі Android файл. Він відповідає за детальний опис вашого додатку для операційної системи. Також у цьому файлі визначаються усі елементи з якими буде взаємодіяти ваш додаток. В папці res зберігаються різноманітні ресурси нашого додатка, зазвичай ці ресурси належать до графічної складової. Це можуть бути зображення, кольори проекту, теми які будуть використані, константні значення, меню, параметри, анімацій або інші xml-файли. Система збірки Android компілює ресурси програми та вихідний код та пакує їх у файли .apk, які ви можете перевірити, розгорнути, підписати та розповсюдити. Android Studio використовує Gradle, вдосконалений набір інструментів збірки, для автоматизації та управління процесом збірки, дозволяючи одночасно визначати гнучкі власні конфігурації збірки. Кожна конфігурація збірки може визначити власний набір коду та ресурсів, одночасно використовуючи загальні для всіх версій вашої програми частини. Плагін Android для Gradle працює з набором інструментів збірки, щоб забезпечити процеси та настроюванні параметри, які є специфічними для побудови та тестування програм Android.

Gradle та плагін Android працюють незалежно від Android Studio. Це означає, що ви можете створювати свої програми для Android з Android Studio, командного рядка на вашому комп'ютері або на машинах, де не встановлена Android Studio (наприклад, сервери безперервної інтеграції).

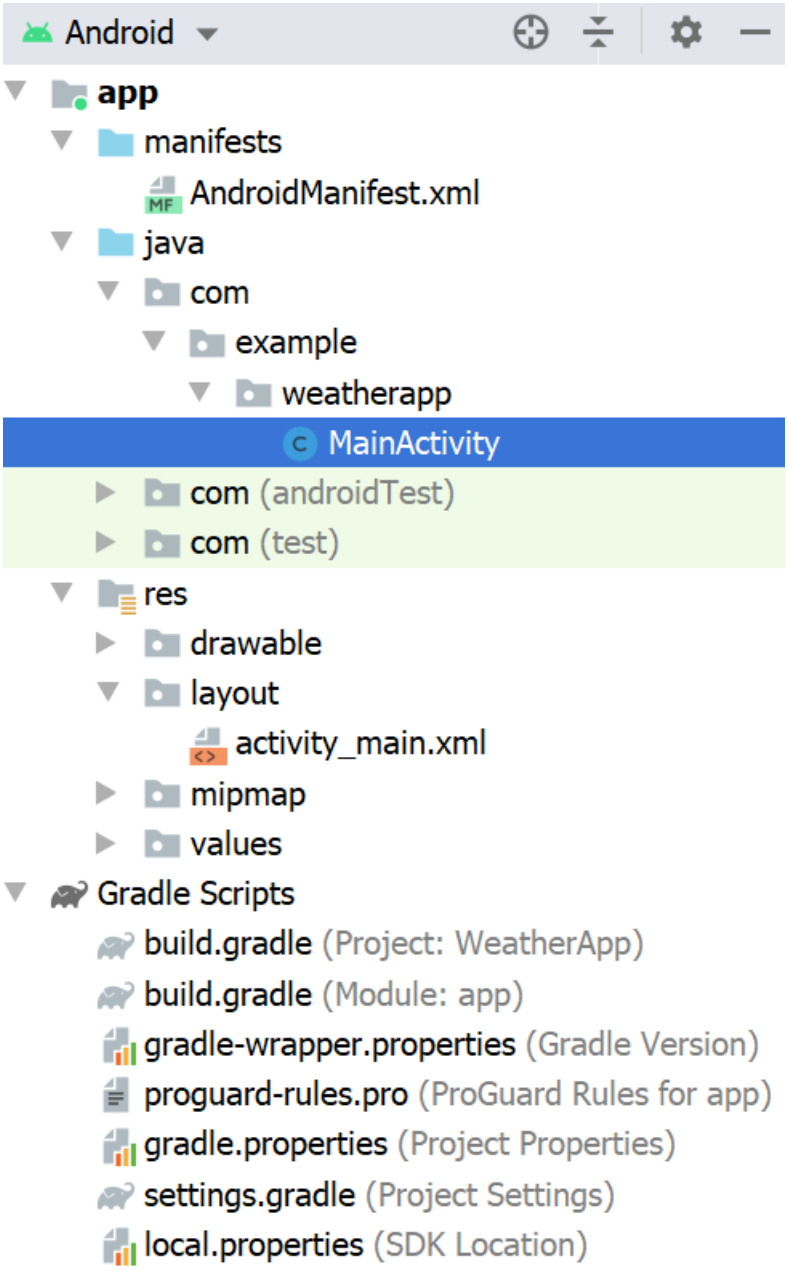

Рисунок 3.3 – Структура створеного проекту

3.2 Розробка графічного інтерфейсу за допомогою XML

Розробка майже кожного Android додатка починається саме з візуальної частини. Android Studio вже створила activity_main.xml файл з яким можна працювати. Для початку треба визначитись з макетом нашого додатку. Макет визначає структуру для користувальницького інтерфейсу у вашому додатку, наприклад, в діяльності. Усі елементи в макеті побудовані з використанням ієрархії об'єктів View і ViewGroup. Вид зазвичай малює те, що користувач може

бачити та взаємодіяти. Тоді як ViewGroup – це невидимий контейнер, який визначає структуру макета для View та інших об’єктів ViewGroup. Існує багато видів лейаутів: FrameLayout, LinearLayout, TableLayout, RelativeLayout, GridLayout та ConstrainLayout. Найпопулярнішими з них є LinearLayout, GridLayout та RelativeLayout. Макет LinearLayout розміщує елементи на екрані в одному напрямку горизонтально, або вертикально. Усі елементи на цьому макеті поміщуються в стек один за одним. Це означає що вертикальний список буде мати лише один елемент в своєму ряді. RelativeLayout дозволяє своїм дочірнім елементам визначати свою позицію на макеті відносно один іншого, або самого макету. Це робиться за допомогою унікальних ідентифікаторів елементів. TableLayout це таблична розмітка, елементи в ній розміщуються у вигляді таблиці. Всі елементи позиціонуються у строки та стовбці, або комірки. GridLayout – це також табличний від розмітки, проте він більш зручний ніж його попередник. У GridLayout для кожного компонента можна вказати строку та колонку і цей елемент буде розміщено саме в ній.

Для розміщення елементів у нашому додатку буде використано макет типу LinearLayout, тому що при лінійному розміщенні буде дуже просто розмістити необхідні нам елементи. Базовому LinearLayout буде задана вертикальна орієнтація за допомогою android:orientation = vertical. Також ми будемо використовувати декілька TextView для відображення параметрів на екрані. EditText потрібен для того щоб ввести данні. ImageView буде представляти картинку, на яку буде можливість натиснути. Та ще два LinearLayout для більш камфорної розмітки. Далі буде надано детальний код файлу activity_main.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    android:orientation="vertical"
    tools:context=".MainActivity"
    android:background="@drawable/background">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:padding="10dp"
        android:weightSum="3"
        android:layout_marginTop="20dp">

        <EditText
            android:id="@+id/editTextCity"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="2.5"
            android:textColor="#000"
            android:textSize="16dp"
            android:background="#fff"
            android:padding="10dp"
            android:gravity="center"
            android:layout_gravity="center"
            android:hint="@string/hint_city"
            android:textColorHint="#000"/>

        <ImageView
            android:id="@+id/imageViewSearch"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
```

```
        android:layout_weight="0.5"
        android:src="@drawable/search"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="10dp"
        android:layout_marginTop="10dp">

        <TextView
            android:id="@+id/textViewCity"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="#fff"
            android:textSize="40dp"
            android:layout_gravity="center"
            android:layout_margin="20dp"/>

        <TextView
            android:id="@+id/textViewTemp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Температура: 0 °C"
            android:textColor="#fff"
            android:textSize="25dp"
            android:layout_margin="10dp"/>
```

```
        <TextView
            android:id="@+id/textViewHumidity"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textColor="#fff"
            android:text="Вологість: 0 %"
            android:textSize="25dp"
            android:layout_margin="10dp"/>

        <TextView
            android:id="@+id/textViewWindSpeed"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="#fff"
            android:text="Швидкість повіря: 0 м/с"
            android:textSize="25dp"
            android:layout_margin="10dp"/>

    </LinearLayout>

</LinearLayout>
```

TextView використовуються для відображення на екрані міста, який обрав користувач, відображення температури повітря, скорості повітря та вологості. ImageView буде виконувати роль кнопки, на якій буде висіти обробник. EditText потрібен для роботи з користувачем. На наступному рисунку показано результат роботи цього коду.

Рисунок 3.4 – Результат створення інтерфейсу користувача

Після створення візуального інтерфейсу можна переходити до створення логіки додатку.

3.3 Retrofit клас для роботи з API

Для того щоб Android Studio побачила бібліотеку Retrofit, та мала можливість її завантажити до проекту, треба у файлі build.gradle додати наступні строки коду:

```
implementation 'com.squareup.retrofit2:retrofit:2.1.0'
implementation 'com.google.code.gson:gson:2.6.2'
implementation 'com.squareup.retrofit2:converter-gson:2.1.0'
```

Друга строчка додає до проекту Gson конвертор, щоб у Retrofit була можливість працювати з відповіддю серверу у форматі Json.

Клас WeatherClient буде створювати об’єкт Retrofit. Цей клас розроблений за патерном Singleton, це означає шо ми можемо створити лише один екземпляр цього класу. Метод getRetrofit() буде повертати єдиний екземпляр об’єкту Retrofit. Ми створюємо Retrofit за допомогою білдеру, та вказуємо базовий URL сайту з чим API будемо працювати. Далі додаємо конвертор Gson що працювати із відповіддю у форматі Json.

```
public class WeatherClient {

    private static Retrofit retrofit = null;

    public static Retrofit getRetrofit(){

        if(retrofit == null){

            retrofit = new Retrofit.Builder()
                    .baseUrl("https://api.openweathermap.org/data/2.5/")
                    .addConverterFactory(GsonConverterFactory.create())
                    .build();

        }

        return retrofit;
    }
}
```

Щоб працювати з відповіддю серверу, треба зробити POJO клас, який буде відповідати об’єктам які будуть приходити від API. Це потрібно для перетворення Json змінних в Java об’єкти. Згідно з документацією сайту необхідно буде створити 2 класи. WeatherResponse буде відповідати за отримання таких даних як: температура повітря, вологість та тиск. Для кожного з них треба зробити змінну потрібного типу даних, та get/set методи. Анотація @SerializedName вказує шо ця змінна аналогічна змінній у Json відповіді.

```
public class WeatherResponse {

    @SerializedName("temp")
    public float temp;

    @SerializedName("humidity")
    public float humidity;

    @SerializedName("pressure")
    public float pressure;

    public float getTemp() {
        return temp;
    }

    public void setTemp(float temp) {
        this.temp = temp;
    }

    public float getHumidity() {
        return humidity;
    }
```

```
    public void setHumidity(float humidity) {
        this.humidity = humidity;
    }

    public float getPressure() {
        return pressure;
    }

    public void setPressure(float pressure) {
        this.pressure = pressure;
    }

}
```

Клас WindResponse відповідає за отримання потужності вітру.

```
public class WindResponse {

    @SerializedName("speed")
    public float speed;

    @SerializedName("deg")
    public float deg;

    public float getDeg() {
        return deg;
    }

    public void setDeg(float deg) {
        this.deg = deg;
```

```
    }

    public float getSpeed() {
        return speed;
    }

    public void setSpeed(float speed) {
        this.speed = speed;
    }
}
```

У класі ApiClient треба створити більш глобальні змінні, а саме об’єкти класу WeatherResponse та WindResponse. Цей клас буде прошарком між Json відповіддю та конвертацією її у Java об’єкт.

```
public class ApiClient {

    @SerializedName("main")
    private WeatherResponse weatherResponse;

    @SerializedName("wind")
    private WindResponse windResponse;

    public WeatherResponse getWeatherResponse() {
        return weatherResponse;
    }

    public void setWeatherResponse(WeatherResponse weatherResponse) {
        this.weatherResponse = weatherResponse;
    }
```

```
    public WindResponse getWindResponse() {
        return windResponse;
    }

    public void setWindResponse(WindResponse windResponse) {
        this.windResponse = windResponse;
    }
}
```

Інтерфейс WeatherServiceAPI вказує, який тип запиту буде передано через HTTP запит до сайту. Appid це ключ, який треба отримати на сайті Openweather, він потрібен для доступу даних сайту, та щоб додаток був ідентифікований. Переданий параметр «units» визначає в якій систему будуть повернені данні, «metrics» вказує шо дані будуть у метричній формі. Метод getCurrentWeatherData() приймає один параметр – це назва міста, яке буде вводити користувач. Анотація @Query вказує шо наступний параметр буде запитом який відповідає API сайту. Тип поверненого значення методу значиться як Call <ApiClient>.

```
public interface WeatherServiceAPI {

@GET("weather?appid=0ddd8e43e63ee6299c574365f382708b&units=metric)
    Call<ApiClient> getCurrentWeatherData(@Query("q") String name);
}
```

3.4 Створення Main Activity

Зазвичай кожен клас Activity пов’язує графічні елементи із об’єктами Java. Також цей клас управляє логікою життєвого циклу активності, він також

відповідає за обробку кнопок або ж інших функцій візуального інтерфейсу. Клас MainActivity, окрім своїх основних обов’язків, буде активувати Retrofit та робити запит на сервер. Метод getCurrentWeatherData() це реалізація методу який був у інтерфейсі WeatherServiceAPI, саме через нього буде йти запит. Щоб візуальний інтерфейс коректна працював із графічною частиною треба створити змінні, які будуть мати такий саме тип даних що і елемент інтерфейсу. Також для того щоб об’єкти Java могли бути логічною частиною візуальних об’єктів, їх треба прив’язати за допомогою метода findViewById(), який, у якості аргумента, приймає id елементи візуальної частини. Листнери дозволяють вішати на кнопки, або ж інші елементи, обробник, який при натиску на них буде робити деякий функціонал. ImageViewSearch.setOnClickListener() дозволяє при натиску на картинку використати основний метод, який і передає данні до сайту. editTextCity.getText() отримує назву городу чи країни, які обрав користувач, вони передаються як параметр у методі getCurrentWeatherData(). Метод call.enqueue() означає шо передача даних буде йти у асинхронному режимі. OnResponse() викликається після відповіді серверу, тут можна зробити обробку різних кодів відповіді. Код 200 означає що запит пройшов і відповідь була надана. Далі ці данні треба розпарсити та вивести у нашу візуальну частину, робиться це завдяки get/set методам із відповідних класів.

```
public class MainActivity extends AppCompatActivity {

    private TextView textViewCity, textViewTemp, textViewHumidity,
textViewWindSpeed;
    private ImageView imageViewSearch;
    private EditText editTextCity;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        textViewCity = (TextView) findViewById(R.id.textViewCity);
        textViewTemp = (TextView) findViewById(R.id.textViewTemp);
        textViewHumidity = (TextView) findViewById(R.id.textViewHumidity);
       textViewWindSpeed = (TextView)
findViewById(R.id.textViewWindSpeed);
        imageViewSearch = (ImageView)
findViewById(R.id.imageViewSearch);
        editTextCity = (EditText) findViewById(R.id.editTextCity);

        imageViewSearch.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View v) {

              getCurrentWeatherData(editTextCity.getText().toString().trim());

              textViewCity.setText("" + editTextCity.getText().toString());
           }
        });

        editTextCity.setOnFocusChangeListener(new
View.OnFocusChangeListener() {
           @Override
           public void onFocusChange(View v, boolean hasFocus) {
              editTextCity.setHint("");
           }
        });
     }
```

```
    private void getCurrentWeatherData(String name){

        WeatherServiceAPI weatherServiceAPI =
WeatherClient.getRetrofit().create(WeatherServiceAPI.class);

        Call<ApiClient> call =
weatherServiceAPI.getCurrentWeatherData(name);

        call.enqueue(new Callback<ApiClient>() {
            @Override
            public void onResponse(Call<ApiClient> call, Response<ApiClient>
response) {
                if (response.code() == 200) {
                    textViewTemp.setText("Температура: " +
response.body().getWeatherResponse().getTemp() + " °C");
                    textViewHumidity.setText("Вологість: " +
response.body().getWeatherResponse().getHumidity() + " %");
                    textViewWindSpeed.setText("Швидкість повіря: " +
response.body().getWindResponse().getSpeed() + " м/с");
                }
            }

            @Override
            public void onFailure(Call<ApiClient> call, Throwable t) {
            }
        });
    }
}
```

4 РОБОЧА ДОКУМЕНТАЦІЯ ПРОЕКТУ

4.1 Налаштування файлу AndroidManifest.xml

Файл AndroidManifest містить інформацію та параметри запуску для операційної системи Android. Останнє налаштування додатку буде пов’язане із дозволом Android, на використання інтернет трафіку для додатка. Треба прописати декілька параметрів. Параметр uses-permission описує права, які треба запитати в Android, для коректної роботи додатка. Параметр android:networkSecurityConfig="@xml/security" вказує на xml файл, у якому треба встановити конфіг на використання інтернету.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.weatherapp">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.NoActionBar"
        android:usesCleartextTraffic="true"
        android:networkSecurityConfig="@xml/security">

        <activity android:name=".MainActivity"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
```

```
            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>
</application>

<uses-permission android:name="android.permission.INTERNET"/>

</manifest>
```

Код файлу security.xml переставлений нижче.

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="true" />
</network-security-config>
```

4.2 Огляд та тестування розробленого додатка

Функціонал даного додатку залежить від сайту Openweathermap, тому що він бере інформацію про погоду саме із цього сайту. Я можна побачити API сайту дозволяє вводити назву країни, або городу на різних мовах, проте результат буде однаковий. Додаток коректна приймає та відображає данні про погоду.

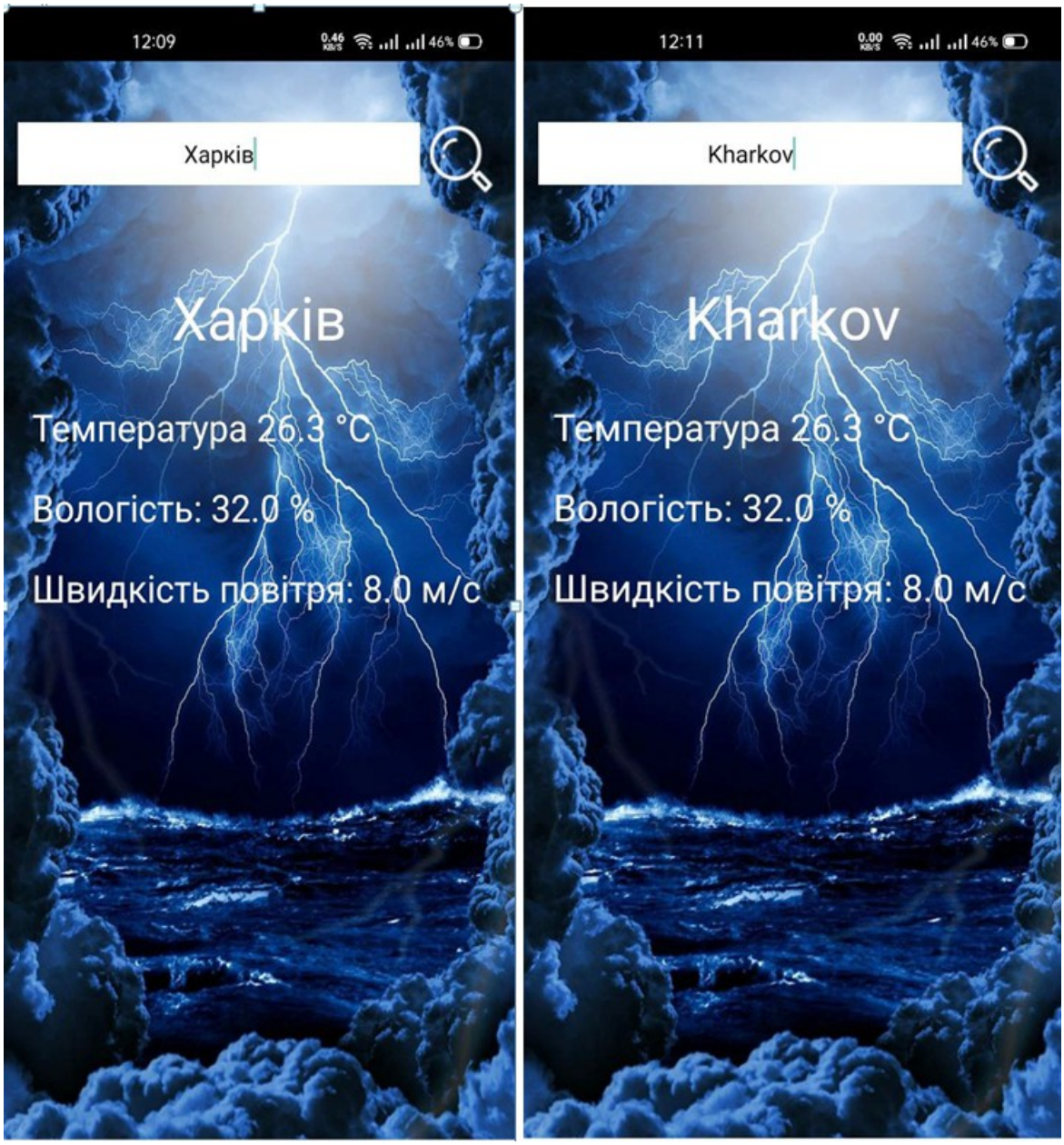

Рисунок 4.1 – Результати роботи програми за запитом погоди

З повною версією коду можна ознайомитись за посиланням - https://github.com/VitalyLitovchenko/WeatherAppDiplom/tree/master.

4.3 Отримання ключа для додатка

Openweathermap надає можливість користуватися їх API для мобільних додатків, проте сайт надає доступ до API лише зареєстрованим додаткам. Щоб сайт міг аутентифікувати додаток треба отримати ключ. Ключ можна створити

після реєстрацій на сайті, далі треба перейти до розділу «My Api Keys» та виконати генерацію ключа.

Рисунок 4.2 – Створенні унікального ключа

Після чого буде генеровано ключ, який потім треба буду використати при створені запиту до API сайту.

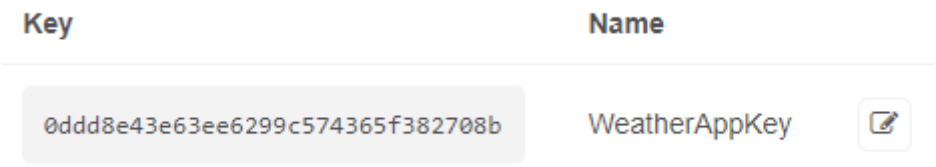

Рисунок 4.3 – Генерований ключ для додатка

За це відповідав інтерфейс WeatherServiceApi, у якому через запит GET було передано раніше отриманий ключ.

@GET("weather?appid=0ddd8e43e63ee6299c574365f382708b&units=metric")

ВИСНОВКИ

У нашому часі мобільна розробка є дуже актуальною темою. Майже кожна людина має свій мобільний смартфон, та користується ним кожного дня. Android займає дуже велику частину ринку, тому його актуальність і надалі буде рости. Майже всі додатки так чи інакше використовують інтернет мережу, тому розробка мобільних додатків завжди потребує вміння правильно працювати із API серверу, на який буде потрібно передавати данні з додатка, або ж запитувати данні з сервера та працювати з ними у додатку. Для того щоб працювати із мережею в Android використовується дуже потужна мережева бібліотека Retrofit. Данна наукова робота показує детальний процес створення Android додатку на об’єктно-орієнтованій мові програмування Java. Підключення та поетапне моделювання та розробку графічного інтерфейсу за допомогою мови XML. Налаштування та розробку компонентів завдяки яким реалізується можливість працювати із різноманітними API сайтів. Мобільний додаток WeatherApp, який було розроблено, має мінімальний необхідний функціонал звичайного погодного додатку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Медникс З., Дорнин Л. Програмування під Android. Видавництво Питер, 2012.

2. Амелин К. С., Граничин О. Н., Кияев В. И., Корявко А. В.. Введення в розробку додатків для мобільних платформ. Видавництво ВВМ, 2011.

3. Дейтел П. Android для програмістів: створюємо додатки. Видавництво Питер, 2012.

4. ГолощаповА.Л. GoogleAndroid. Створення додатків для смартфонів і планшетних ПК. Видавництво Питер 2012.

5. Мельникова О.М.: Смартфони на Android. видавництво Эксмо, 2013.

8. John Wiley & Sons. Reto Meier Professional Android 4 Application Development. Wrox, 2012.

9. Martin Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language (Object Technology Series). Addison Wesley, 2003.

11. Bill Phillips, Brian Hardy. Android Programming: The Big Nerd Ranch Big. NerdRanchGuides, 2013.

12. Square. Retrofit [Электронный ресурс] / square - Режим доступа к ресурсу: https://square.github.io/retrofit/

13. Android Documentation [Электронный ресурс] - Режим доступа к ресурсу: https://developer.android.com/

14. JSON [Электронный ресурс] - Режим доступа к ресурсу: https://www.json.org/json-en.html