Державний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЄКТ

на тему:

«Веб-орієнтованийдодаток на платформі Android з використанням технологій Retrofit, WebView та RemoteGist»

Студента 4 курсу, _ групи, спеціальності 121 «Інженерія програмного забезпечення» освітньої програми «Інженерія програмного забезпечення» _____ підпис студента — Валентенка Олександра Сергійовича

Науковий керівник кандидат технічних наук, доцент кафедри інженерії програмного забезпечення такібербезпеки _____ підпис керівника — Рзаєва Світлана Леонідівна

Гарант освітньої програми кандидат технічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки _____ підпис гаранта — Рзаєва Світлана Леонідівна

КИЇВ – 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь бакалавр
Спеціальність 121 «Інженерія програмного забезпечення»

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
«14» листопада 2022 р.

Завдання
на випускний кваліфікаційний проєкт студентові

Валентенку Олександру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проєкту «Веб – орієнтований додаток на платформі Android з використанням технологій Retrofit, WebView та Remote Gist

Затверджена наказом ректора від «6» грудня 2022 р. № 3288

2. Строк здачі студентом закінченого проєкту 5 червня 2023

3. Цільова установка та вихідні дані до проєкту

Мета проєкту створення веб - орієнтованого додатку на операційній системі Android

Об'єкт дослідження операційна система Android технології Retrofit, WebView та Remote Gist

Предмет дослідження веб орієнтовані технології Retrofit, WebView та Remote Gist

4. Консультанти проєкту із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускного кваліфікаційного проєкту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЗАСТОСУВАННЯ МОБІЛЬНОГО ДОДАТКУ

1.1. Вибір платформи та мови програмування

1.2. Сфери застосування Android додатку

1.3. Організація програми

1.4. Технічне завдання

1.5. Висновки до розділу 1

РОЗДІЛ 2. ВИБІР ПРОГРАМНИХ ЗАСОБІВ ТА ПРОЄКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ

2.1. Аналіз технологій

2.2. Бібліотека Retrofit

2.3. Технології для візуалізації

2.4. Технологія WebView

2.5. Висновки до розділу 2

РОЗДІЛ 3. РОЗРОБКА ДОДАТКУ

3.1. Налаштування платформи для розробки

3.2. Створення проекту

3.3 Інтеграція бібліотек, налаштування WebView

3.4. Вибір кольору

3.5. Висновки до розділу 2

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

6. Календарний план виконання проєкту

№ пор.	Назва етапів випускного кваліфікаційного проєкту	Строк виконання етапів проєкту	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускного кваліфікаційного проєкту*	*21.09.2022*	*21.09.2022*
2.	*Розробка та затвердження завдання на проєкт*	*14.11.2022*	*14.11.2022*
3.	*Вступ та перелік літературних джерел*	*23.12.2022*	*23.12.2022*
4.	*Розділ 1. Аналіз предметної області застосування мобільного додатку*	*27.01.2023*	*27.01.2023*
5.	*Розділ 2. Вибір програмних засобів та проєктування мобільного додатку*	*03.03.2023*	*03.03.2023*
6.	*Розділ 3. Розробка додатку*	*14.04.2023*	*14.04.2023*
7.	*Висновки*	*28.04.2023*	*28.04.2023*
8.	*Здача випускного кваліфікаційного проєкту на кафедру (перша перевірка)*	*17.05.2023*	*17.05.2023*
9.	*Підготовка автореферату та презентації доповіді*	*26.05.2023*	*26.05.2023*
10.	*Попередній захист випускного кваліфікаційного проєкту*	*29.05.2023 – 02.06.2023*	*02.05.2023*
11.	*Зовнішнє рецензування випускного кваліфікаційного проєкту*	*05.06.2023*	*05.06.2023*
12.	*Здача прошитого випускного кваліфікаційного проєкту на кафедру*	*05.06.2023*	*05.06.2023*
13.	*Публічний захист випускного кваліфікаційного проєкту*	*19.06.2023*	*19.06.2023*

7. Дата видачі завдання «14» листопада 2022 р.

8. Науковий керівник випускного кваліфікаційного проєкту Рзаєва С.Л.
(прізвище, ініціали, підпис)

9. Гарант освітньої програми Рзаєва С.Л.
(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Валентенко О.С.
(прізвище, ініціали, підпис)

11. Відгук керівника випускного кваліфікаційного проєкту

Науковий керівник випускного кваліфікаційного проєкту

_____

(підпис, дата)

Відмітка про попередній захист _____

(ПІБ, підпис, дата)

12. Висновок про випускний кваліфікаційний проєкт

Випускний кваліфікаційний проєкт студента Валентенко О.С.

(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми Рзаєва С.Л.

(прізвище, ініціали, підпис)

Завідувач кафедри Криворучко О. В.

(підпис, прізвище, ініціали)

«____»_____20____р.

АНОТАЦІЯ

Відповідно до мети дослідження робота присвячена створенню веб-орієнтованого додатка на платформі Android з використанням технологій Retrofit, WebView та RemoteGist, в результаті розробки створення додатку, виявлено зручність та гнучкіть такого сервісу як RemoteGist, який має перевагу перед

Ключові слова: додаток, технологія, залежність, клас, інтеграція Retrofit, WebView, RemoteGist, Android

ABSTRACT

According to the purpose of the research, the work is devoted to the creation of a web-oriented application on the Android platform using Retrofit, WebView and RemoteGist technologies, as a result of the development of the application, the convenience and flexibility of such a service as RemoteGist, which has an advantage over

Keywords: application, technology, dependency, class, Retrofit integration, WebView, RemoteGist, Andro

ЗМІСТ

ВСТУП

У сучасному світі веб-орієнтовані додатки набувають все більшої популярності, оскільки вони дозволяють користувачам взаємодіяти з інформацією та сервісами з будь-якого місця та в будь-який час. Для розробки таких додатків на платформі Android існує багато технологій, які допомагають розробникам швидко та легко розробляти високоякісні додатки. Для досягнення поставленої мети, у роботі розглянуто наступні завдання:

Аналіз можливостей технологій Retrofit, WebView та RemoteGist для розробки мобільних додатків на базі Android.

Проектування мобільного додатку, який буде включати в себе функції, що задовольнять потреби користувачів.

Розробка мобільного додатку з використанням технологій Retrofit, WebViewtaRemoteGist. Для розробки проекту було обрано програмне забезпечення IntelliJ IDEA,

На мою думку це найкраще програмне забезпечення на ринку, яке дає змогу інтегрувати Android Studio. Також варто зазначити що, для розробки даного проекту буде достатньо безкоштовної версії цієї програми, а саме IntelliJ IDEA Community, ця версія містить у собі весь необхідний функціонал, та інструменти.

Мета дослідження: взаємодія операційної системи Android з технологіями Retrofit, WebView, Remote Gist

Об'єкт дослідження: веб спроможності WebView та Android

*Предмет дослідження***:** технології Rertofit WebView

У відповідності з метою дослідження поставлені наступні *завдання*:

- проаналізувати предметну область застосування мобільних додатків;
- розробка технічного завдання;
- проаналізувати технології Retrofit, WebView та RemoteGist з подальшим їх впровадженням у розробку мобільного додатку на базі Android;
- створення веб орієнтованого додатку з використанням технологій Retrofit та RemoteGist.

Практичне значення дослідження: спростити вівідування сайту, та можливість реклами одного продукту за рахунок іншого.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЗАСТОСУВАННЯ МОБІЛЬНОГО ДОДАТКУ

1.1 Вибір платформи та мови програмування

Android - це операційна система для мобільних пристроїв, розроблена компанією Google. Вона базується на ядрі Linux та має відкритий код, що дозволяє розробникам створювати додатки та розширення для різних пристроїв, що працюють на Android.

Щодо мов програмування, Android підтримує кілька мов програмування, серед яких:

Java - це офіційна мова програмування для розробки додатків Android. Java використовується для створення більшості додатків та бібліотек для Android.

Kotlin - це новітня мова програмування для розробки додатків Android. Вона є альтернативою мові Java і надає додаткові можливості та переваги, такі як більша безпека та чистота коду.

C/C++ - ці мови використовуються для створення високопродуктивних компонентів, таких як бібліотеки та драйвери.

Python - ця мова програмування використовується для розробки скриптів та інструментів, які підтримують розробку додатків Android.

JavaScript - ця мова програмування використовується для розробки веб-додатків та гібридних додатків на платформі Android.

Оскільки Android підтримує різні мови програмування, розробники можуть вибирати мову, яка найкраще підходить для їх потреб.

Мова програмування Java була обрана тому що для неї використовуються найбільша кількість користувацьких бібліотек, які закривають майже всі потреби розробника, деякі з них були використанні для створення програмного забезпечення.

Метою мого додатку є швидкий доступ до сайту ДТЕУ, та можливість не нудьгувати коли сервера сайту будуть перевантажені або вимкнуті.

Додаток повинен швидко проводити переадресацію на сайт буквально за одне натискання на ярлик додатку на смартфоні який підтримує операційну систему Android, також як було сказано раніше в момент коли сайт буде недосупний користувач мав змогу пограти в гру яка буде розробленна для цього проекту.

Додаток буде використовувати бібліотеки які находяться в відкритому доступі.

Оскільки вибір пав саме на мову програмування Java для розробки додатку знадобиться середовище розробки для написання коду. Оди з найпопулярніших це Android Studio, котрий в свою чергу спеціально було створено для зручного, та швидкого програмування додатків для операційної системи Android. Також одною із головних переваг цього середовища можна виділити легке та зручне інтегрування бібліотек, які є у вільному доступі, або навіть розробка власних бібліотек, для майбутнього використовування. До того ж Android Studio має зручний та зрозумілий інтерфейс, та низький ступінь входу, а також, не вибагливі системні вимогу до пристрою на якому він буде використовуватись. Що до особливостей цього серидовища можна віднести:

- Консоль розробника, або підсказки по оптимізації програми, або вказівки на помилки при створенні коду, також допомога до перекладу, та навіть метрики Google аналітики.

- Живі макети(loyaut), або як його ще називають “живе кодування” тобто подавання програми в реальному часі, насправді дуже цікава та зручна функція за допомогою якої можливе відстеження зовнішнього вигляду програми, задля розуміння того який результат можна очікувати.
- Базування на Gradle, завдяки якому можливе під`єднання сторонніх бібліотек, та налаштування програми для версії пристроїв.
- Шаблони для створення дизайнів та компонентів.
- А також перевірка орфографії та синтаксису коду, із заміною його на актуальний, новітній варіант. Зазвичай використовується в програмування мовою Kotlin. А також лаконічне відтворення та спростування деяких частин коду.

1.2 Сфери застосування Android додатку

На ринку технологій, додатки для смартфонів отримають визнання вже досить тривалий час, адже на сьогоднішній людство має змогу використовувати смартфон у ролі кишенькового комп`ютера, котрий в деяких випадках має такіж самі характеристики як і домашній ПК, тільки значно менший та простіший у використовуванні. Тим самим забезпечує швидкий доступ з будь - якої точки світу, а користуються ними як і школярі так і відомі світові лідери. Завдяки цьому ринок додатків для смартфонів набирає оберти кожного дня. На сьогоднішній день людство має змогу завдяки смартфону як оплачувати комунальні послуги, так і вести торги на самих відомих біржах, або просто відпочивати в іграх, чи проводити час з онлайн друзями за переглядом фільму, або навіть мати доступ до державних послуг, оплати штрафів, якщо брати інноваційний додаток ДІЯ. Отже, головною перевагою для людста стала портативність, та простота.

Якщо брати до уваги світові бренди, то наявність мобільного додатку котрий надає користувачам доступ до сервісу компанії, або навіть замовлення їжі у ресторані, став відміткою якості та клієнтоорієнтованості компанії, адже завжди зручно замовити продукти для дому, або новеньке авто находячись у метро, або прямуючи з роботи додому. Також, до цих критеріїв можна віднести факт того що не у кожної родини є персональний комп`ютер, навідміну від смартфону, не володіє такою портативністю та простотою у використанні.

1.3 Організація програми

Додатки Android організовані як набір компонентів. Існує чотири типи компонентів, і додатки можуть складатися з одного або більше компонентів кожного типу. Динамічний екземпляр компонента відповідає підмножині програми, яка може виконуватися незалежно від інших. Таким чином, багато в чому додаток для Android можна розглядати як набір взаємодіючих компонентів. Компоненти Android-додатків бувають двох типів:

- Користувацькі компоненти, які реалізують відображення та захоплення даних.
- Фонові компоненти, які працюють незалежно від будь-якої видимої користувачеві активності.

1.4 Технічне завдання

Щодо технічного завдання веб – орієнтованого додатку на платформі Android з використанням технологій Retrofit, WebView та RemoteGist, необхідно взяти до уваги швидкість використання, та малу вагу додатку.

Тобто у розробці будуть використані наступні дії:

1. Огляд:

- Назва додатку: DTEU Reader
- Опис: Додаток дозволяє користувачам переглядати веб-сторінки та завантажувати вміст з віддалених Gist-файлів.

2. Функціональні вимоги:

- Відображення домашньої сторінки зі списком доступних Gist-файлів.
- Можливість вибрати конкретний Gist-файл для перегляду.
- Завантаження та відображення вмісту обраного Gist-файлу в WebView.
- Підтримка офлайн-режиму для раніше завантажених Gist-файлів.
- Можливість оновлення вмісту Gist-файлів з сервера.
- Збереження обраного Gist-файлу як вибраного для швидкого доступу.

3. Нефункціональні вимоги:

- Дизайн інтерфейсу повинен бути привабливим і зручним для користування.
- Додаток повинен бути оптимізований для роботи на різних пристроях з платформою Android.
- Використання технології Retrofit для взаємодії з веб-сервером та отримання списку Gist-файлів.
- Використання WebView для відображення вмісту Gist-файлів.
- Використання RemoteGist для завантаження вмісту Gist-файлів з сервера.
- Збереження обраного Gist-файлу в локальне сховище для доступу в офлайн-режимі.

4. Технічні вимоги:

- Мінімальна версія платформи Android: Android 5.0 (Lollipop) або новіше.
- Використання мови програмування Java.

- Використання бібліотеки Retrofit для роботи з HTTP-запитами до сервера.
- Використання бібліотеки WebView для відображення вмісту веб-сторінок.
- Використання бібліотеки RemoteGist для завантаження вмісту Gist-файлів з сервера.
- Збереження обраного Gist-файлу в локальну базу даних або в файлову систему пристрою.

5. Розширення можливостей:
   - Можливість додавання коментарів до Gist-файлів.
   - Підтримка авторизації користувача для доступу до приватних Gist-файлів.
   - Налаштування для кастомізації зовнішнього вигляду та поведінки додатку.
   - Використання Retrofit для взаємодії з веб-сервером і отримання списку Gist-файлів за допомогою API-запитів.
   - Використання WebView для відображення HTML-контенту Gist-файлу з можливістю взаємодії з вмістом.
   - Використання RemoteGist для завантаження вмісту Gist-файлів з сервера та їх збереження в локальне сховище.
   - Розділення логіки додатку на різні шари (наприклад, даних, доменної логіки, презентації) для забезпечення масштабованості та легкості розробки.

Насправді, я вважаю що деякі з цих пунктів є обов`язковими для використання у розробці більшості додатків, так як принципи розробки залишаються майже не змінними. Однак варто зазначити, що конкретні вимоги та пріоритети можуть відрізнятися залежно від типу програми, функції та призначення.

Наприклад, програма, яка дуже піклується про безпеку, може вимагати використання HTTPS і шифрування даних, тоді як інша програма, яка не обробляє конфіденційну інформацію, може поставити ці вимоги на нижчий пріоритет. Крім того, розробка додатків для платформи Android залежить від специфіки проекту, обраної архітектури, використовуваних технологій і бібліотек. Наприклад, використання Retrofit, WebView і RemoteGist – це лише деякі з можливих варіантів, існує багато інших інструментів і бібліотек для розробки додатків Android. Тому важливо враховувати контекст конкретного проекту та потреби користувачів, щоб визначити обов'язкові вимоги та вибрати найкращі практики розробки.

Але так як веб – орієнтований додаток не вимагатиме для користувачі доступу до їх даних таких як: гелерея, файли, паролі до системи. Тому додаток буде максимально безпечний для використання.

Приймачі широкомовлення. Як вже обговорювалося раніше, загальносистемні широкомовні події можуть генеруватися системним програмним забезпеченням або додатками. Компоненти, які слухають ці трансляції від імені програм, є приймачами трансляцій. До складу програми може входити декілька приймачів трансляцій, які прослуховують оголошення. У відповідь приймач трансляції може ініціювати інший компонент, наприклад, активність, для взаємодії з користувачем або використання загальносистемного менеджера сповіщень. Наприклад, активність, для взаємодії з користувачем або використання загальносистемного менеджера сповіщень наприклад, активність, для взаємодії з користувачем або використання загальносистемного менеджера сповіщень.

Провайдери контенту. Компоненти, які надають доступ до даних програми, є постачальниками контенту. Базові класи надаються в Android SDK як для постачальника контенту (тобто компонент постачальника контенту повинен розширювати базовий клас), так і для компонента, який шукає доступ. Постачальник контенту може зберігати дані у будь-якому внутрішньому представленні, яке він обирає, чи то файлова система, служба SQLite, чи якесь специфічне для програми представлення (включно з тими, що реалізовані через віддалені веб-служби).

Додатки Android складаються з комбінацій цих типів компонентів. Виклик компонентів керується за допомогою загальносистемного механізму трансляції на основі намірів.

1.5 Висновок до розділа 1

Операційна система Android досить гнучка, та має досить цікаві можливості як для розробника так і для користувача. Не дивлячись на свою простоту ця ОС дарує досить серйозні а головне актуальні додатки, які покривають більшість потреб користувачів, починаючи від державних послуг(якщо взяти до уваги додаток ДІЯ), закінчуючи переглядом фільмів та іграми. Веб – орієнтований додаток, мусить мати лаконічний вигляд, та функціонал який він буде пропонувати не має загружати систему та пристрій користувача, аби він не втрачав своєї швидкодійності, та мав можливіть використання у багатовікненному режимі. Для того аби користувач мав змогу нотувати інформацію з сайту, або не відриватися від інших додатків котрими він користувався, до запуску веб – орієнтованого додатку.

РОЗДІЛ 2.

ВИБІР ПРОГРАМНИХ ЗАСОБІВ ТА ПРОЄКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ

2.1 Аналіз технологій

Для розробки мобільного додатку використовуються різні технології та бібліотеки. У даній роботі використовуються технології Retrofit, WebView та RemoteGist.

Retrofit - це бібліотека для роботи з мережевими запитами в Android-додатках. Вона дозволяє розробникам створювати клієнтів для RESTful API швидко та легко, використовуючи мінімальний код. Retrofit дозволяє використовувати різні формати серіалізації даних, такі як JSON, XML та інші.

WebView - це компонент Android, який дозволяє відображати веб-сторінки безпосередньо в додатку. Він використовує внутрішній браузер, щоб завантажувати та відображати веб-сторінки, але також надає можливість контролювати взаємодію користувача з веб-сторінкою.

RemoteGist - це система контролю версій, яка дозволяє зберігати та керувати версіями програмного коду на віддаленому сервері. Вона надає можливість розробникам працювати з кодом в різних гілках та версіях, а також дозволяє керувати правами доступу до коду.

Кожна з цих технологій має свої унікальні особливості та використовується для різних завдань в розробці програмного забезпечення.

Retrofit допомагає розробникам створювати клієнтів для взаємодії з сервером API, WebView дозволяє відображати веб-сторінки у додатку, а RemoteGist дозволяє керувати версіями коду та співпрацювати з іншими розробниками у роботі з кодом.

Розглянемо принцип роботи додатку та взаємодії класів на прикладі діаграми(рисунок 1)

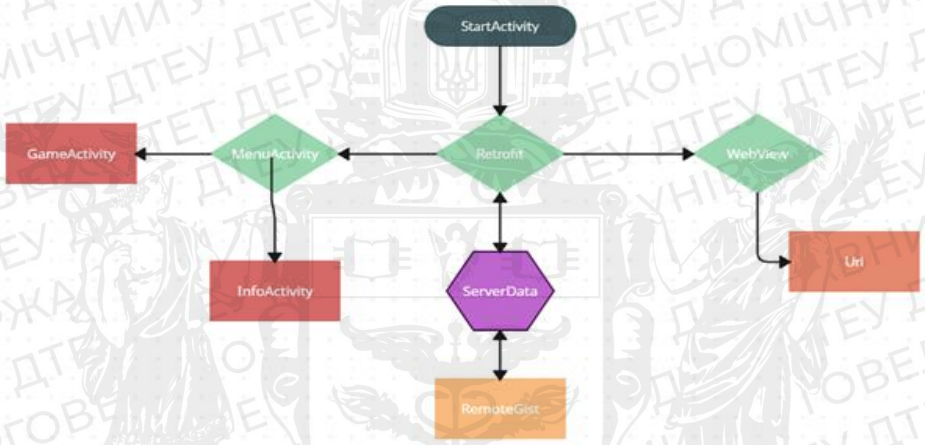

Рис. 2.1. Взаємодія класів мобільного додатку

Проаналізувавши діаграмму можемо зрозуміти що старт додатку починається з класу StartActivity, який являє собою коротку анімацію під час якого завдяки бібліотеці Retrofit а також його методу проходить перевірка на перемикач який знаходиться у ServerData, який у свою чергу через JSON файл надає запит до сервісу Remotegist, на якому знаходяться керування перемикачем та url сайту. Він відправляє данні про нього в ServerData і методи Retrofit показують інформацію яка залежить від перемикача, тобто демонструє нам через WebView сайт ДТЕУ, або запускає клас MenuActivity, котрий в свою чергу надає нам можливість пограти в гру або прочитати інформацію про додаток.

Аби підключити бібліотеку Retrofit нам необходно потрібно внести наступну залежність у файл build.gradle проекту (рису. 2.2):

```
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
```

Рис. 2.2. Фрагмент програмного коду – build.gradle проекту

Перша залежність відповідає за саму бібліотеку, друга дає можливість конвертувати JSON файли а також читати їх зміни.

Файл JSON потрібно створити на сайті GitHubGist а також прописати прості функції які і будуть нашими перемикачами наприклад (ри. 2.3):

```
1  {
2  "switcher":"true",
3  "load":"https://knute.edu.ua"
4  }
```

Рис. 2.3. Фрагмент програмного коду – GitHubGist

Саме swither буде керувати проектом, змінну якої можна перемикати з true на false та навпаки. Load в свою чергу відповідає за url який буде нам демострувати WebView.

Що до бази даних проекту, було вирішено використовувати сервіс OneSignal

OneSignal - це сервіс повідомлень для мобільних та веб-додатків, який дозволяє розсилати повідомлення користувачам з різних платформ через один API. З OneSignal можна надсилати сповіщення про оновлення, новини, пропозиції та іншу інформацію безпосередньо на пристрої користувачів.

Але не дивлячись на свою спеціалізацию в розсиланні повідомлень, ций сервіс дає нам можливість моніторити інформацію про активних користувачів, а також їх пристрої країну та мову.

Для демонстації використую дані з іншого проекту

Рис. 2.4. Створений проект OneSignal

На даному рисунку ми бачимо таблицю яка містить у собі назву проекту, кількість пристроїв які встановили додаток(TotalSubs), а також активні користувачі(ActiveSubs) і платформу пристроя(Platforms).

Також якщо ми перейдемо по назві проекту ми побачимо більш детальну статистику про додаток наприклад : Рисунок 2

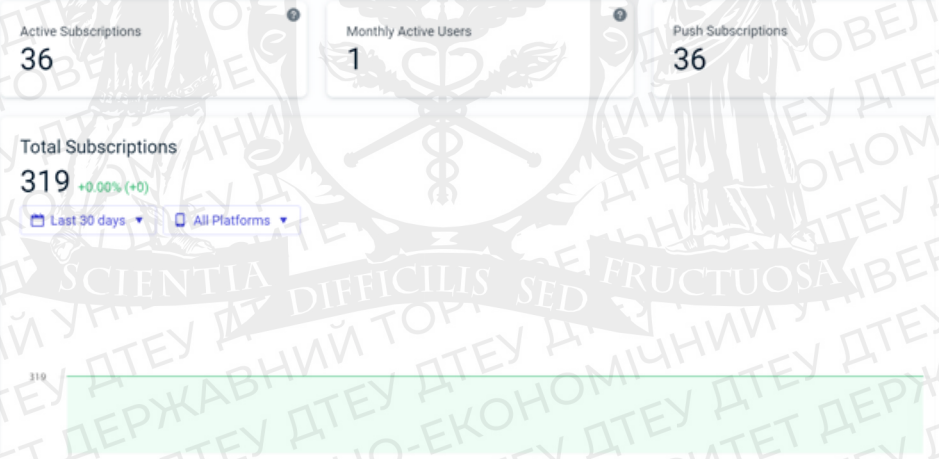

Рис. 2.5. Статистика One Signal

Але нам потрібна таблиця з базою даних користувачів, і звичайно її маємо (рис. 2.3., 2.4)

Channel	Subscribed	Last Unsubscribed	Last Active	Usage Duration	Sessions	First Session	Device
	Not Subscribed	03/25/22, 1:01:30 pm	03/24/22, 9:23:42 pm	0	2	03/24/22, 8:20:45 pm	TECNO CD7 10
	Not Subscribed	03/19/22, 1:01:15 pm	03/18/22, 7:40:54 pm	0	1	03/18/22, 7:40:54 pm	SM-A307FN 10
	Not Subscribed	01/01/22, 7:01:10 pm	01/01/22, 6:27:03 pm	0	1	01/01/22, 6:27:03 pm	M2006C3LG 10
	Outdated Google Play Services App		10/11/21, 10:28:12 pm	0	1	10/11/21, 10:28:12 pm	Pixel 3 XL 9
	Outdated Google Play Services App		10/10/21, 9:40:22 pm	0	1	10/10/21, 9:40:22 pm	Pixel 3 XL 9
	Outdated Google Play Services App		10/09/21, 9:24:47 pm	0	1	10/09/21, 9:24:47 pm	Pixel 3 XL 9
	Outdated Google Play Services App		10/08/21, 9:46:48 pm	0	1	10/08/21, 9:46:48 pm	Pixel 3 XL 9
	Not Subscribed	10/24/21, 1:01:35 pm	08/02/21, 1:21:34 pm	0	2	06/23/21, 6:42:16 pm	DUA-L22 8.1.0
	Not Subscribed	08/09/21, 1:01:52 pm	06/10/21, 5:07:56 pm	0	1	06/10/21, 5:07:56 pm	Redmi 8 10
	✔		10/14/21, 8:39:51 pm	0	2	06/03/21, 6:58:20 am	SM-A107F 9

Рис. 2.6. База даних користувачіа

Player ID	Phone Number	Country	Location Point	Language Code	Tags	Rooted
1f7dfdf4-ab9f-11ec-9898-9e43b631eefd	-	UA		ru		No
8f722764-a6e2-11ec-b7bf-a6d8e0688000	-	UA		ru		No
a7330e12-6b1f-11ec-b0fa-d21783443f9d	-	UA		ru		No
84741912-6bb3-4832-8640-5483276e2dd0	-	CN		zh-Hans		Yes
b8336b12-9abf-4252-b01c-7ee83ca1e7fa	-	CN		zh-Hans		Yes
846076e8-b507-4d60-9351-152511cd6cdf	-	CN		zh-Hans		Yes
1cfc2611-dd1b-4058-96c8-894d5f7f07f1	-	CN		zh-Hans		Yes
1e51ebcd-4e1f-4ec4-9dd6-10a9200fb4b6	-	UA		ru		No
31fb078d-f5be-4480-8158-898e46b7a5e0	-	UA		uk		No
1c8055c5-87e7-421c-ba0e-c7de53eacf6f	-	UA		uk		No

Рис. 2.7. База даних країн

Як і мовилось раніше ми маємо інформацію про те чи використовую користувач сервіси GoogleApp(Subscribed), коли користувач видалив додаток(LastUnsuscribed), його останню активність(Last Active), кількість сессій(Sessions), першу сессію(FirstSession), те на який пристрій встановлено додаток(Device), уникальний код користувача, для відправки персональних

пуш повідомлень(PlayerID), країну в якій находиться пристрій(Country), а також мову яка вікористовується пристроєм. Можливо це і забогато інформоції яку збирає ця база, але якщо розробляти додаток наприклад магазину або гру, то для приваблення користувачів які давно не користуються додатком не обхідно використовувати всю інформацію для правильного налагодження пуш повідомлень, наприклад більшостікористувачам буде зручніше читати повідомлення мовою якою вони користуються, а також час в який будуть надходити повідомлення.

Тим паче, якщо розглядати таку базу даних в екосистемі великих компаній, які мають майже безмежні потужності, така детальна інформація про користувачів допоможе іншим відділам компанії точніше налагоджувати таргетовану рекламу, яка у свою чергу привабить нових користувачів, тим самим збільшить прибуток для фірми.

Щоб додати OneSignal до додатку Android, потрібно виконати наступні кроки:

Створити обліковий запис OneSignal і додаток в OneSignal Dashboard.

Додати залежність OneSignal SDK до вашого проекту. Це можна зробити, додавши наступну залежністькод у файл build.gradle (Module: app рис. 2.9).

```
implementation 'com.onesignal:OneSignal:4.3.5'
```

Рис. 2.8. Залежність OneSignal

Ініціалізувати OneSignal SDK у вашому додатку, зазвичай в методі onCreate() вашої Activity. Для цього необхідно викликати метод OneSignal.initWithContext(), передавши контекст вашої додатку та ключ реєстрації OneSignal. Наприклад (рисунок 8):

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Ініціалізуємо OneSignal SDK
        OneSignal.initWithContext(this);
        OneSignal.setAppId("YOUR_ONESIGNAL_APP_ID");
    }
}
```

Рис. 2.9. Ініціалізація OneSignal

Налаштувати прийом повідомлень в вашому додатку. Для цього необхідно створити клас-сповіщення (NotificationReceiver), який наслідується від класу BroadcastReceiver. У методі onReceive() цього класу ви можете здійснювати потрібні дії з повідомленням. Наприклад, показувати сповіщення на екрані. Прийняти повідомлення можна за допомогою методу OneSignal.setNotificationOpenedHandler(), передавши в нього екземпляр вашого класу-сповіщення. Наприклад:

```
public class NotificationReceiver extends BroadcastReceiver {

    @Override

    public void onReceive(Context context, Intent intent) {

        String title = intent.getStringExtra("title");

        String message = intent.getStringExtra("message");
```

Що ж після додавання бази OneSignal ми маємо наступну діаграму (рис. 2.10)

Тобто при запуску додатку бібліотека та метод onCreate() від OneSignal одразу починає збір статистики на пересилає данні на сайт OneSignal App в якому ми вже можемо відслідковувати статистику додатку та нових користувачів.

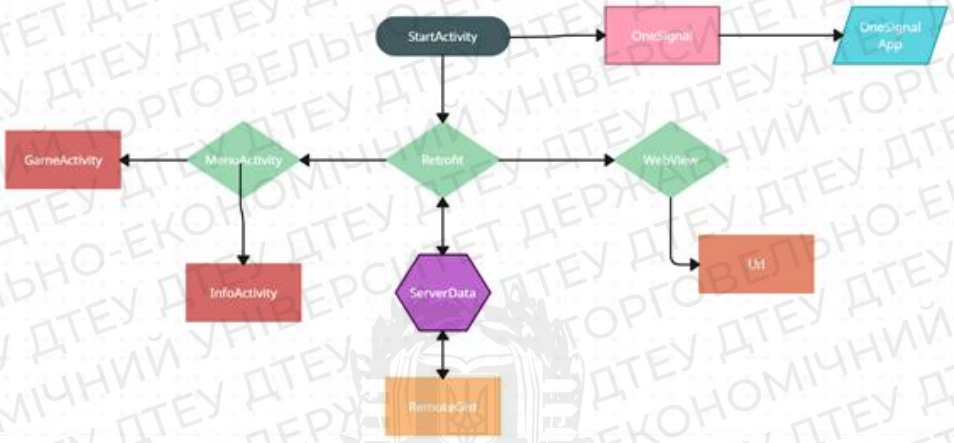

Рис. 2.10. Запуск OneSignal

Саме тому і була обрана база даних OneSignal тому ще це сучасне рішення яке використовують передові компанії. Завдяки цьому ми можемо не лише слідкувати за користувачами але й правильно налагодити повідомлення які будуть приваблювати користувачів.

2.2 Бібліотека Retrofit

Багато сайтів мають власні API для зручного доступу до своїх даних. Наразі найпоширеніший варіант - це JSON. Також можуть зустрічатися дані у вигляді XML та інших форматів. Бібліотека Retrofit спрощує взаємодію з REST API сайту, беручи на себе частину рутинної роботи. Авторами бібліотеки Retrofit є розробники з компанії "Square", які написали безліч корисних бібліотек, наприклад, Picasso, Okhttp, Otto.

За описом на сайті розробника бібліотеки це типізований http клієнт для Android. Офіційна документація доступна тут. Для простоти сприйняття необхідно описати деякі поняття. Усі пристрої в мережі обмінюватися даними між собою через протоколи обміну (TCP/IP, Http, Https, Ftp тощо).

Протокол - це загально узгоджені правила пакування, адресації, надсилання та отримання даних. За допомогою цих протоколів клієнт і сервер (Web сервер) обмінюватися даними між собою. Rest - це архітектура web серверів, яка використовується для обміну даними на протоколах http і https. Rest являє собою набір правил для виконання методів HTTP.

Retrofit це API на мові java, що спрощує спілкування з rest веб серверами. Дозволяє виконувати методи HTTP безпосередньо мовою java.

Http методи - стандартизовані формати процедур, які клієнт запитує web-сервер виконати. Методи http: Get, Head, Post, Put, Delete, Trace, Options, Connect і Patch. Retrofit підтримує тільки перші 5

HTTP методи

1. Get і Head методи вважаються найбезпечнішими методами, оскільки отримують дані, не змінюючи їх де-небудь. Перший метод отримує заголовок і тіло відповіді тоді як другий тільки заголовок. Відповіді від сервера приходять у вигляді колекцій значень. Під час запиту ми можемо вказати додаткові параметри для відбору

На приклад у запиті adress.ua?list=10&name=retro ми передаємо параметри відбору list = 10 і name = retro

2. Delete - метод виконує однойменну функцію, видаляє записи на сервері. Для методу не потрібне тіло запиту

3. Post і Put методи створюють або змінюють дані на сервері. На відміну від інших цих методах наявність тіло запиту обов'язкова. Put метод створить запис на сервері, якщо такого немає, якщо запис присутній, то він його змінить. Post створює записи і запускає процедури на сервері. Можна запам'ятати такі вирази: Put - відправляє дані і Post - Відправляє повідомлення на сервер. За приклад візьмемо XML метод:

```
<root>
<id>1234</id>
<name>Sanobe</name>
</root>
```

Щоб перенести об’єкти Java на сервер, який може працювати будь-якою мовою, модернізація використовує конвертери даних. Ці конвертери також відомі як бібліотеки серіалізації даних. Retrofit включає 6 конвертерів, які використовують найпопулярніші бібліотеки серіалізації: Gson, Jackson, Moshi, Protobuf, Wire і Simple XML. Використовуваний конвертер залежить від форматів, які приймає сервер. Наприклад, я використовую конвертер GSon, тому що він простий у використанні

2.3 Технології для візуалізації.

Найголовнішу роль в додатку бере на себе саме його візуальна частина. Тому саме дизайну необхідно надати особливу увагу. Дизайн додатку - це візуальне оформлення програми, а також створення структури, заснованої на логіці користувача поведінки. Іншими словами, це не лише зовнішній вигляд, а й зручність використання.

Проаналізувавши цей термін, ми розуміємо що:

- Додаток повинен мати простий, зрозумілий дизайн.
- Повинен мати мінімум зайвої інформації та зайвих функцій.
- Мусить мати приємні кольори оформлення.
- Повинен використовувати заокругленні кути(наприклад кнопок), для того аби користувач був розслаблений при використанні додатку.
- Мати розвантажений інтерфейс

Інтерфейс - сукупність засобів, методів і правил взаємодії (керування, контролю, тощо) між елементами системи

Для використання основних правил, та поставлених задач щодо розробки дизайну, в проекті ввикористано технологію Material Design.

Material Design - це зовнішній вигляд програмного забезпечення та програм операційної системи Android від компанії Google.

Технологія дає нам змогу створення тактильної поверхності. Тобто, кожний дотик користувача до елементу додатку, супроводжується тою чи іншою анімацією, зміною яскравості елементу, або додавання тіней.

Як пояснюється в Mobile Design Trends 2015 & 2016р[1] Робота з тактильними поверхнями є контейнерами для зберігання контенту та інформації. Він має плоский дизайн, але має світлий відтінок, що виділяє його серед інших контейнерів і шарів. Цей підхід не вимагає використання інших технік, таких як текстури або градієнти, щоб створити візуальні відмінності між шарами.

Тобто дана технологія надає розробнику можливість розділяти додаток на декілька шарів, за приклад візьмемо роботу Material Design в одному із встановлених додатків на пристрої рис. 2.11.

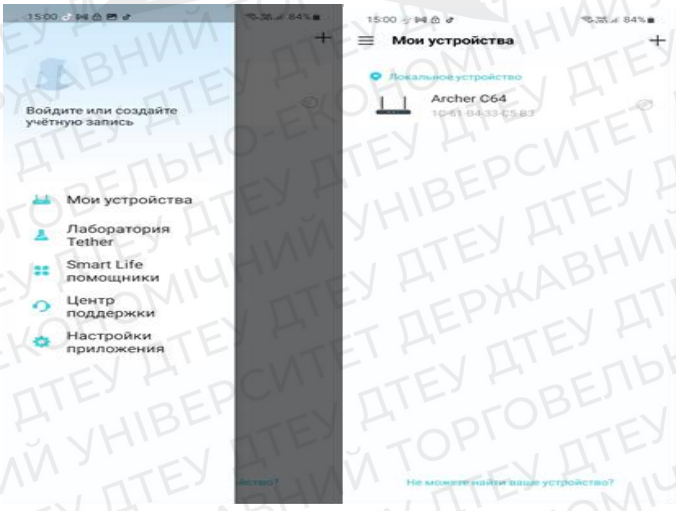

Рис. 2.11. Знімок з екрану

Звернемо увагу на рис. 2.7, це головна сторінка додатку, але при натисканні на три горизонтальні лінії в лівому куті екрану, з лівої частини інтерфейсу відкривається вікно(рисунок 6), та тим самим перекриває головну сторінку додатку. Саме це перекриття і являє роботу з розділенням шарів у додатку, тобто, меню зображене на рисунку 1, і є верхнім шаром програми.

До цього всього технологія Material Design чітко визначає відношення і функції як продемонстровано у дослідженні Android Lollipop UI Kit[2] тактильні поверхні чітко визначають взаємозв'язок і функцію вмісту в дизайні (кожен блок зазвичай виконує певну функцію, наприклад посилання або відеоплеєр). Цей підхід також створює глибину, оскільки елементи з інших блоків накладаються шарами, щоб створити, здавалося б, тривимірний світ. За приклад візьмемо наступний рис. 2.8

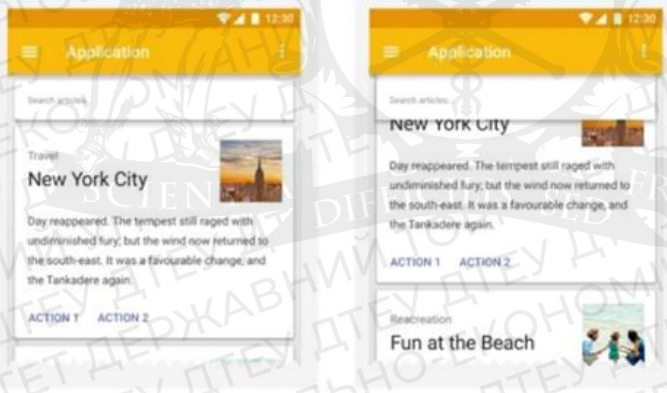

Рис. 2.12. Технологія Material Design

На даному рисунку демонструється згадана раніше шаровість, та ефект створення глибини. Котрий надає об`єму в дизайні, та позитивно впливає на користувача.

Отже спектор можливостей Material Design, майже безмежний, дана технологія надає можливість роботи від чогось глобального наприклад шаровість інтерфейсу, до мінімальної і простої кнопки, після натискання на яку, змінюється її форма, колір, або додаються тіні.

Цікаво, що одна з останніх рекомендацій щодо дизайну в рамках керівництва з Material Design просить веб-дизайнерів перейти від навігації вгорі екрана до навігації внизу екрана - те, що Apple застосовує вже багато років. В одному зі своїх останніх бренд-переходів Google оновлює інтерфейс YouTube, роблячи його більш пласким і обтічним. В інтерфейсі "Вхідних від Gmail" користувачі можуть побачити характерні риси матеріального дизайну Google. Плоскі кнопки, шрифт і графіка мають чітко виражений фірмовий стиль. Якщо ви перейдете на сайт, ви також побачите анімацію, характерну для матеріального дизайну.

Ще один сайт, натхненний матеріальним дизайном, Nimbus9, чітко демонструє характерні риси дизайну на головній сторінці. Стрілка вниз точно така ж, як і стрілка в інтерфейсі "Вхідні від Gmail". Ці два продукти відрізняються один від одного, але підхід до дизайну в них знайомий.

Тобто вся чарівність Material Design полягає в тому що ніщо не говорить про те, як потрібно дослівно слідувати інструкціям з матеріального дизайну. Для кожного додатку є ті чи інші методи дизайну, і взагалі немає якихось виважених норм вигляду додатку, я вважаю що основне що має бути у дизайні це простота, та легкість, тим самим правилом и корегуються такі гіганти як Google або Apple. Якщо завантажити та встановити будь який додаток від Google ми не побачимо у ньому загромадженості, та не відчуємо дискомфорту, навпаки спостерігаючи за всіма їх анімаціями, відчуватимемо лише відпочинок та комфорт, а також що є дость важливо, такими дидайнами знижується навантаження на очі.

2.4 Технологія WebView

WebView - це вбудований браузер, який нативний додаток може використовувати для відображення веб-вмісту. Тут слід виділити два набори слів:

Переше - це нативний додаток (також відомий як додаток). Нативний додаток - це додаток, написаний мовою та фреймворком інтерфейсу, розробленими спеціально для певної платформи:

Іншими словами, додаток не є крос-платформним веб-додатком, що запускається в браузері. Натомість, думайте про свої програми як про такі, що написані переважно мовою Swift, Objective-C, Java, C++, C# і т.д., які тісніше взаємодіють з системою. Щоб пояснити це в контексті, більшість програм, які ви використовуєте на своєму мобільному пристрої, будуть нативними програмами. Багато популярних програм, таких як Microsoft Office на вашому настільному комп'ютері/ноутбуці, також є нативними.

Друге слово, на яке слід звернути увагу - це вбудований браузер. Ми всі знаємо, що таке браузер. Це окрема програма, яку ми можемо використовувати для серфінгу в інтернеті.

Якщо уявити браузер як дві частини, то одна з них - це інтерфейс (адресний рядок, кнопки навігації тощо), а інша - рушій, який перетворює розмітку і код на пікселі, які ми можемо бачити і з якими взаємодіяти.

WebView - це просто частина движка браузера, яку ви можете вставити як iframe у свій додаток і програмно вказати йому, який веб-вміст завантажувати.

Якщо зібрати все це разом і з'єднати деякі крапки над "i", то WebView - це просто візуальний компонент/елемент керування/віджет/тощо, який ми будемо використовувати для створення візуальних елементів нашого власного додатку.

WebView зазвичай завантажує веб-вміст віддалено з сайту http:// або https://. Це означає, що ви можете взяти частину (або весь) веб-додаток, який знаходиться на вашому сервері, і покластися на WebView для його відображення у нативній програмі.

За замовчуванням, будь-який веб-код, що виконується у WebView або веб-браузері, ізольований від решти програми. Це робиться з міркувань безпеки, які зводяться до мінімізації шкоди, яку може завдати зловмисний JavaScript. Якщо браузер або WebView вийдуть з ладу, це прикро, але нічого страшного. Для довільного веб-вмісту такий рівень безпеки має великий сенс. Ви ніколи не можете повністю довіряти веб-вмісту, який завантажується. Це не стосується WebViews. Для сценаріїв WebView розробник зазвичай має повний контроль над вмістом, який завантажується. Ймовірність потрапляння туди шкідливого коду, який спричинить хаос на вашому пристрої, досить низька.

Ось чому для WebViews розробники мають безліч підтримуваних способів перевизначити поведінку безпеки за замовчуванням і налагодити взаємодію між веб-кодом і кодом власне програми. Ця взаємодія зазвичай відбувається за допомогою так званого моста. На попередній схемі ви можете побачити цей міст у складі Native Bridge і JavaScript Bridge.

Тобто, WebView можна вважати безпечним браузером, котрий був інтегрований в додаток, для перегляду тої чи іншої інформації. Тобто якщо сказати досить грубо, ця технологія є ретранслятором між сайтом, та пристроєм.

Ранні версії WebView були інтегровані як частина основної ОС. Користувачі могли оновлювати WebView лише через великі оновлення системи.

З виходом Android 5.0 компанія Google відокремила WebView від основної ОС.

Завдяки цій зміні оновлення WebView можна було поширювати через додаток Google Play Store, і користувачі могли оновлювати WebView незалежно від ОС.Якщо у компоненті WebView було виявлено проблему, Google випускав виправлення, яке користувачі могли встановити без необхідності повністю оновлювати Android. Google поєднував WebView з Google Chrome у версіях 7.0, 8.0 і 9.0. Однак в Android 10.0 він знову став окремим компонентом. Користувачі можуть оновити WebView в Android 10 через Google Play Store. Іноді для цього потрібно видалити збережені дані з кешу програми перед завантаженням оновлення.

Google більше не надає патчі для вразливостей, знайдених у старіших версіях Android - 4.3 і раніше. Щоб захистити пристрої від атак, які можуть використовувати можливості WebView, компанія рекомендує користувачам Android запускати останню версію ОС і оновлювати WebView, коли з'являється відповідний запит.

Зазвичай для оновлення програмного коду або того ж самого WebView використовують технологію RemoteGist. Котра в свою чергу вистуває невеликим сервером, запит до якого програми може отримувати інформацію, або посилання для WebView

2.5 MVVM Додатку

MVVM (Model-View-ViewModel) — це архітектурний шаблон, який часто використовується в розробці додатків для Android. Веб-додатки на платформі Android також можна розробляти за допомогою підходу MVVM.

MVVM дозволяє відокремити логіку програми від її відображення, що полегшує тестування та підтримку коду. Веб-орієнтовані програми на платформі Android, створені за допомогою MVVM, можуть ефективно використовувати веб-сервіси, отримувати дані, відображати дані та відповідати на події користувача. Крім того, за допомогою технологій Retrofit і WebView, мережеві запити та веб-сторінки можна легко інтегрувати в архітектуру MVVM. Retrofit дозволяє робити HTTP-запити до веб-серверів, отримувати дані та оновлювати моделі. WebView надає можливість відображати веб-сторінки, які можна використовувати як частину відображення в поданні, або керувати ViewModel через інтерфейс JavaScript.

Загалом MVVM сприяє структурованій та ефективній веб-орієнтованій розробці додатків на платформі Android, забезпечуючи розділення відображення та бізнес-логіки та полегшуючи обробку мережевих запитів і веб-сторінок. Розглянемо принципи роботи на наступному рисунку 2.6

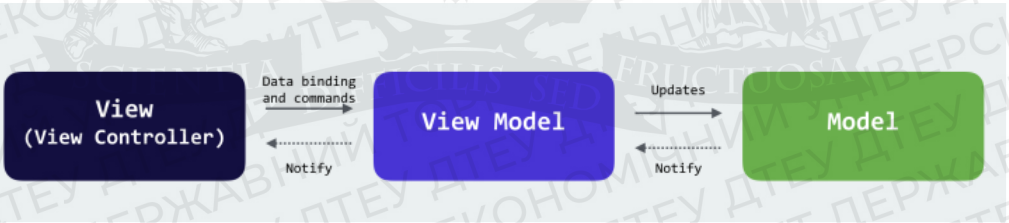

Рис. 2.13. Принципи роботи технології

2.6 Висновки до розділу 2

Отже, технології описані у розділі, мають широкий спектор використань, якщо брати до уваги саме технологію Material Design, завдяки її широкому функціоналу та гнучкості, у втіленні ідей, можна створювати додатки різних сфер, за її допомоги створюються як і віртуальні гаманці, так і додатки для відстеження здоров`я.

В свою чергу WebView дає можливість втілити до додатку буквально будь який сайт, який можна контролювати технологією Retrofit.

Якщо використовувати ці технології разом то користувач, буквально стає свідком командної роботи цих технологій, які гармонічно доповнюють одна одну. Що до дизайну додатку, все ж таки слід розробляти його так аби користувачам і розробнику було приємно користуватись тими чи іншими функціями додатку. Особливу увагу слід приділяти кольору, та відображення інформації в самій програмі, аби вона була легкою, та зрозумілою. Того ж самого потрібно притримуватись при налагоджуванні WebView. Та особливо не забувати указувати дії стандартних кнопок пристрою, аби користувач інтуітивно міг розраховувати на них.

РОЗДІЛ 3.

РОЗРОБКА ДОДАТКУ

3.1 Налаштування платформи для розробки.

Для того аби почати розробку проекту, необхідно налаштувати нашу платформу для розробки. У моєму випадку необхідно завантажити IntelliJ IDEA з офіційного сайту та інтегрувати Android Studio.

Для використання Android Studio необхідно:

1.Відкрити IntelliJ IDEA

2.Перейти до розділу Plugins

3. В пошуку ввести Android

4. Увімкнути натиснувши на нього

Пропоную розібрати ці дії (рис. 3.1)

Рис. 3.1. Android Studio

3.2 Створення проекту.

Для того аби створити проект необхідно натиснути кнопку New Project (рис. 3.2)

Рис. 3.2. New Project

Після натискання у спеціальному вікні необхідно обрати операційну систему Android, тип пристрою, у нашому випадку Phone and Tablet (рисунок 15), та натиснути кнопку Next.

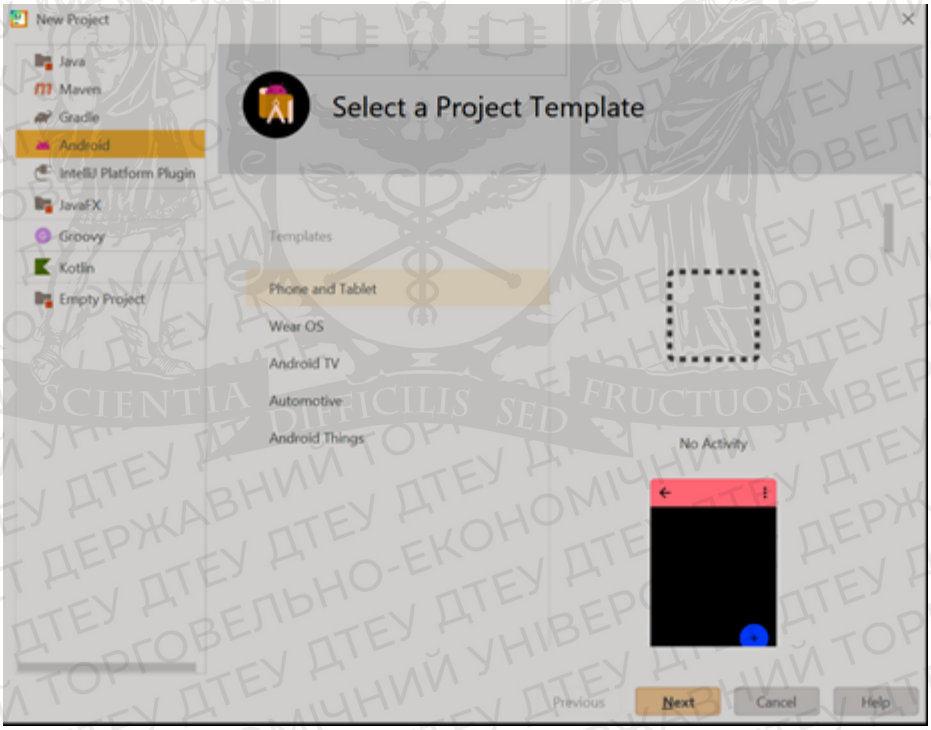

Рис. 3.3. Обрання типу пристрою

Далі потрібно обрати назву проекту, назву пакетів проекту, а також місце збереження, та натиснути кнопку Finish (рис. 3.4)

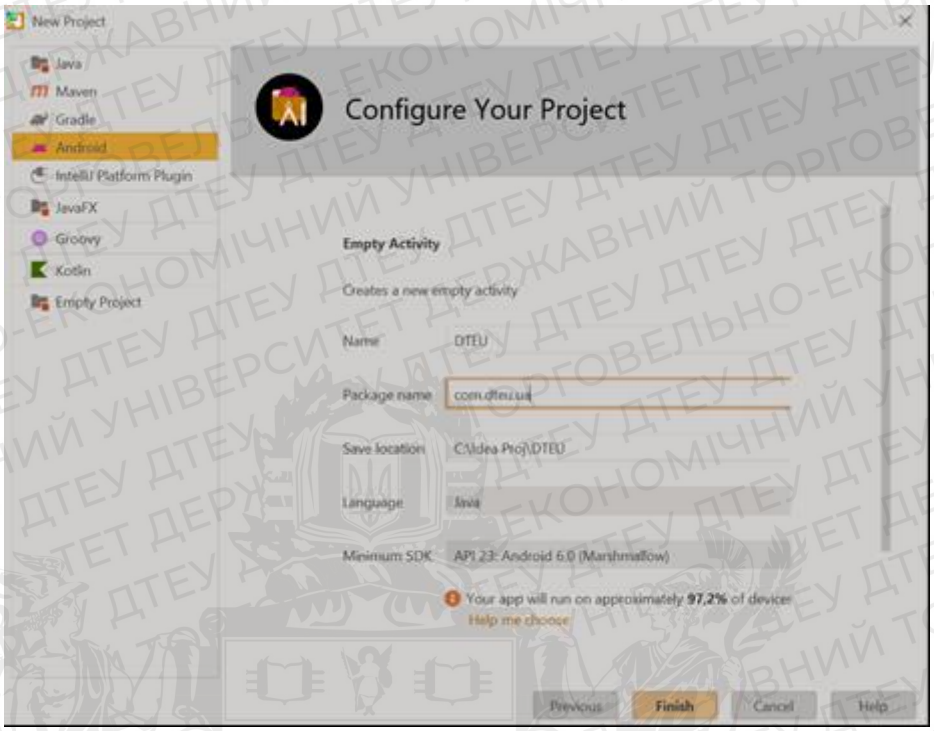

Рис. 3.4. Конфигурація системи

Після виконаних дій буде створено пустий проект в якому буде одна активність MainActivity, а також один XML файл для неї, в якому можна опрацьовувати візуальну частину проекту.

3.3 Інтеграція бібліотек, налаштування WebView.

Пропоную розпочати роботу над проектом саме з інтеграції необхідних бібліотек для нього. Для цього достатньо перейти в файл build.gradle, і під’єднати repositories наступним чином:

```
buildscript {
repositories {
maven { url 'https://plugins.gradle.org/m2/' }
}
dependencies {
classpath 'gradle.plugin.com.onesignal:onesignal-gradle-plugin:0.13.4'
}
}
apply plugin: 'com.onesignal.androidsdk.onesignal-gradle-plugin'
apply plugin: 'com.android.application'
```

Після додавання данного коду у нас з'являєтьтя можливість використовувати бібліотеки OneSignal.

Також для інтеграції бібліотек Retrofit достатньо в блоці dependencies додати насупне:

implementation 'com.onesignal:OneSignal:4.6.0' - підтримка OneSignal

implementation 'com.squareup.retrofit2:retrofit:2.9.0' - інтеграція Retrofit

implementation 'com.squareup.retrofit2:converter-gson:2.9.0'- конвертація gson для роботи з Retrofit

Для взаємодії додатку з INTERNET необхідно прописати дозвіл в AndroidManifest:

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

Також аби додаток міг підлаштовувати орієнтацію екрану не обхідно указати android:configChanges="keyboardHidden|orientation|screenSize"/> в блоці application перед активностями.

Також пропоную одразу додати до проекту технології WebView аби користувачі додатку мали змогу переходити на сайт, тобто створюю окрему

активність та XML файл для WebView, і одразу прописую код для налаштування мобільного браузеру. Варто звернути увагу на те що, також необхідно указати метод керування браузером через стандартні кнопки пристрою, особливо кнопка яка відповідає про повернення на попередню сторінку. Звернення до якої викликається наступним чином :

```
@Override
public void onBackPressed() {
    if (webView.canGoBack()) {
        webView.goBack();
```

3.4 Вибір кольору

Коли користувач бере в руки телефон, він хоче вирішити виконати якусь задачу, або навіть перевірити час, погоду, тощо. Більшість задач вирішують додатки, завантажені на його пристрій. Але, яким би не був функціональний додаток, якщо у нього поганий дизайн, користувач втрачає цікавість до нього. Отже, дизайн додатку це ледь не найважливіша частина його розробки, тому що перше що бачить, до чого він доторкається це саме візуальна частина.

Пропоную почати розробку інтерфейсу з самого основного, тобто, вибір кольору. На цьому етапі найскладніше за все буде вибрати один із мільйона інших кольорів. Після вибору одного кольору, за допомогою платформи Adobe Colors(7), ми зможемо підібрати підкольори які будуть гармонічно доповнювати один одного, для того аби користувачу було приємно користуватись додатком.

Для вибору кольору звернувся до статті Creativebloq, в якій описано психологію кольору.

Психологія кольору: Психологія кольору — це галузь науки, яка вивчає кольори та те, як різні відтінки кольорів можуть викликати певні емоції.

Після ознайомлення зі статтею, було прийнято рішення за оснву взяти синій колір, тому що він викликає емоції зваженості та відпочинку. Потім після обраного кольору hex код якого є #48C0D6, звернувся до платформи Adobe Color, дізнався гармонічні відтінки цього кольору, власне на самій платформі це виглядає наступним чином (рис. 3.5)

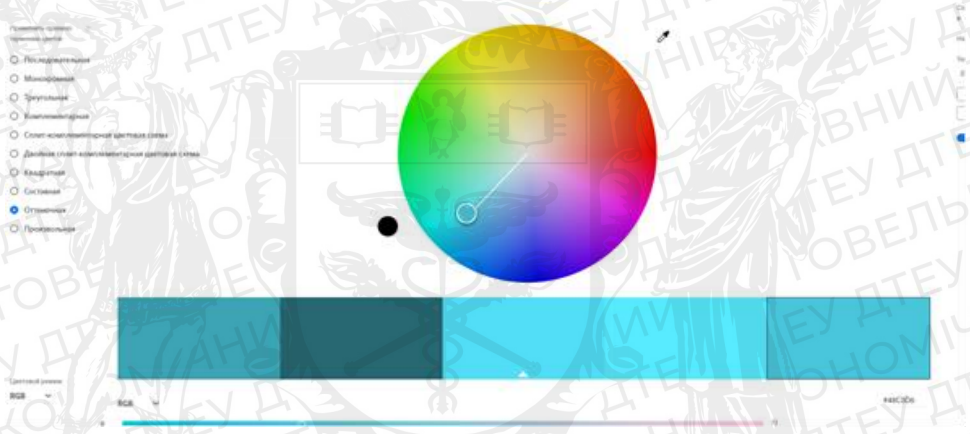

Рис. 3.5. Вибір кольорової гами

Отримано 5 відтінків темні з яких буду використовувати для рамки кнопок, світліші для тла, та кольору заповнення самих кнопок. Наступним етапом для розробки дизайну є саме інтеграція бібліотеки до проекту. Як і раніше для цього достатньо буде вписати в файл build.gradle

```
dependencies {
  // ...
  implementation 'com.google.android.material:material:<version>'
  // ...
  }
```

Тобто в блок dependencies вписуємо залежність implementation та назву залежності 'com.google.android.material:material:<version>'

Після цих дій все готово для використання технології Material Design.

Наступним кроком, в дизайні додатку додам анімацію для головної активності MainActivity, для цього в ньому ж прописати наступне

```
private void spin() {
    int angle = Random.nextInt(3600 - 360) + 360;
    float pivotX = wheel.getWidth() / 2;
    float pivotY = wheel.getHeight() / 2;
    final Animation animation = new RotateAnimation(lastAngle == -1 ? 0 : lastAngle, angle, pivotX,
pivotY);
    lastAngle = angle;
    animation.setDuration(2500);
    animation.setFillAfter(true);
     wheel.startAnimation(animation);
  }
```

А також в XML файлі activity_main завантажити необхідну картинку яку ми будемо анімувати, та прописати розміри положення, тло екрану завантаження, назвемо цю анімацію wheel.

Після описаних дій при запуску активності Main Activity буде відображення анімації завантаження сторінки.

3.4. Висновок до розділу 3

Після дослідження технологій описаних у роботі, отримали простий додаток, та функціональний додаток який знаходиться завжди поруч. Також отримали тандемну роботу технологій. Не дивлячись на багатий функціонал будь якого, додатку, користувач звертає увагу саме на візуальну частину.

Отже для того аби захопити користувачів, необхідно вести статистику пристроїв якими вони користуються, час, мову, завдяки технології OneSignal ми маємо таку можливість. А за допомогою Remote Gist та Retrofit, ми зможемо змінювати направлення та роботу додатку, слідкуючи за відгуками користувачів.

Технологія OneSignal надає нам можливість здійснювати такий моніторинг. Крім того, за допомогою Remote Gist та Retrofit, ми зможемо вносити зміни до роботи додатку, враховуючи відгуки та побажання користувачів.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Під час розробки проекту було створено швидкий та функціональний веб-орієнтований додаток для операційної системи Android. Використання таких технологій, як Retrofit, WebView та RemoteGist, дозволило досягти поставлених завдань і забезпечити лаконічний дизайн.

Додаток надає користувачам можливість швидко перевіряти посилання та їх переадресацію. Завдяки Retrofit, можна легко виконувати HTTP-запити до веб-сервера і обробляти отримані відповіді. WebView дозволяє відображати веб-сторінки безпосередньо в додатку і взаємодіяти з ними.

RemoteGist забезпечує систему контролю, що дозволяє зберігати і синхронізувати дані між різними пристроями. Це може бути корисно для забезпечення доступу до перевірених посилань та налаштувань користувача на різних пристроях.

Під час розробки проекту була врахована гнучкість та простота використаних технологій. Використання Material Design дозволяє створити привабливий дизайн, що сподобається користувачам. Такі типи додатків можуть бути використані для реклами продукції або веб-сайту на різних платформах.

Результатом роботи над проектом є функціональний та зручний додаток, який допомагає користувачам перевіряти посилання та контролювати їх переадресацію. Використання сучасних технологій та дизайну дозволяє забезпечити зручну роботу з додатком і привернути увагу користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 1.Технологія Retrofit [Електронний ресурс] – Режим доступу: https://square.github.io/retrofit/
2. 2. Платформа Android [Електронний ресурс] – Режим доступу: https://www.android.com/intl/ru_ru/
3. 3. Технологія Web View [Електронний ресурс] – Режим доступу:
4. https://developer.android.com/reference/android/webkit/WebView
5. Mobile Design Trends 2015 & 2016 [Електронний ресурс] – Режим доступу: https://www.uxpin.com/studio/ebooks/mobile-ui-ux-design-trends-2015-2016/
6. Android Lollipop UI Kit [Електронний ресурс] – Режим доступу:

https://www.uxpin.com/material-design-ui-kit

					ДТЕУ 121 **06-04**.БР		
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*			
Зав. каф.		Криворучко О.В.		23.12.22	Веб-орієнтованого додатка на платформі Android з використанням технологій Retrofit, WebView та RemoteGist	*Стадія*	*Аркуш* *Аркушів*
Керівник		Рзаєва С.Л.		23.12.22		*СВД*	*46* *46*
Гарант		Рзаєва С.Л.		23.12.22		Факультет інформаційних технологій *4 курс*, 6 *група*	
Розробив		Валентенко О.С		23.12.22	Список використаних джерел		

Додаток А

Class CastCamping

```
package com.aeusgoruner.tohate.appswheel;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.provider.Settings;

import com.aeusgoruner.tohate.DeviceWheelUniqueId;
import com.aeusgoruner.tohate.MenuActivity;
import com.aeusgoruner.tohate.view.WheelViewActivity;
import com.onesignal.OneSignal;

import java.util.Map;


public class CustCamping {
    private Activity activity;
    private Context context;
    private String secondSwitcher;
    private String domen;
    private Map<String, Object> map;

    public CustCamping(String secondSwitcher, String domen, Activity activity, Context context, Map<String, Object> map) {
        this.secondSwitcher = secondSwitcher;
        this.domen = domen;
        this.activity = activity;
        this.context = context;
        this.map = map;
    }

    public boolean isContainKeyCampaign(Map<String, Object> map) {
        return map.containsKey("campaign");
    }

    public boolean isCampaignNull(Map<String, Object> map) {
        if (isContainKeyCampaign(map)) {
            return String.valueOf(map.get("campaign")).equalsIgnoreCase("null")
                    && String.valueOf(map.get("campaign")).equalsIgnoreCase("none")
                    && String.valueOf(map.get("campaign")).equalsIgnoreCase("");
        } else {
            return true;
        }
    }

    public void checkCampaign() {
        if (secondSwitcher.equals("false")) {
            if (isCampaignNull(map)) {
                openZaglushka();
            } else {
                createUrl();
            }
        } else {
            if (isCampaignNull(map)) {
                if (isAdbOn()) {
                    openZaglushka();
                } else {
                    createUrlWithoutCampaign();
                }
            } else {
                createUrl();
            }
        }
    }

    private void createUrlWithoutCampaign() {
        DeviceWheelUniqueId transitionHelper = new DeviceWheelUniqueId();
        Intent intent = new Intent(activity, WheelViewActivity.class);
        intent.putExtra("adress",
                domen.concat(transitionHelper.adUniqueId(context))
                        .concat(transitionHelper.deviceUniqueId(context)));
        activity.startActivity(intent);
        activity.finish();
```

```
        }

    public void createUrl() {
        sendPush();
        DeviceWheelUniqueId transitionHelper = new DeviceWheelUniqueId();
        Intent intent = new Intent(activity, WheelViewActivity.class);
        intent.putExtra("adress",
                domen.concat(transitionHelper.adUniqueId(context))
                        .concat(transitionHelper.deviceUniqueId(context)).concat("&sub_id_1=").concat(String.valueOf(map.get("campaign"))));
        activity.startActivity(intent);
        activity.finish();
    }
    public void sendPush(){
        String[] strings = String.valueOf(map.get("campaign")).split("&");
        for (String data : strings) {
            if (data.startsWith("push")) {
                OneSignal.sendTag("sub_app", data);
                break;
            }
        }
    }

    public void openZaglushka() {
        activity.startActivity(new Intent(activity, MenuActivity.class));
        activity.finish();
    }


    public Boolean isAdbOn() {
        int adb = Settings.Global.getInt(context.getApplicationContext().getContentResolver(), "adb_enabled", 0);
        return adb == 1;
    }
}
```

Class FlyerWheel

```
package com.aeusgoruner.tohate.appswheel;


import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.provider.Settings;

import com.aeusgoruner.tohate.DeviceWheelUniqueId;
import com.aeusgoruner.tohate.MenuActivity;
import com.aeusgoruner.tohate.view.WheelViewActivity;
import com.appsflyer.AppsFlyerConversionListener;

import java.util.List;
import java.util.Map;


public class FlyerWheel implements AppsFlyerConversionListener {
    private Activity activity;
    String secondOpen = "first";
    private List<String> params;
    private Context context;

    public FlyerWheel(List<String> params, Context context,Activity activity) {
        this.params = params;
        this.activity = activity;
        this.context = context;
    }

    @Override
    public void onConversionDataSuccess(Map<String, Object> map) {
        switch (secondOpen) {
            case "first":
                secondOpen = "second";
                CustCamping customCamping = new CustCamping(params.get(1), params.get(3), activity, context, map);
                customCamping.checkCampaign();
                break;
            case "second":
                break;
        }
    }

    @Override
```

```
public void onConversionDataFail(String s) {
    if (params.get(1).equals("false")) {
        openZaglushka();
    } else {
        if (isAdbOn()) {
            openZaglushka();
        } else {
            createUrlWithoutCampaign();
        }
    }
}

@Override
public void onAppOpenAttribution(Map<String, String> map) {

}

@Override
public void onAttributionFailure(String s) {

}

private void createUrlWithoutCampaign() {
    DeviceWheelUniqueId transitionHelper = new DeviceWheelUniqueId();
    Intent intent = new Intent(activity, WheelViewActivity.class);
    intent.putExtra("adress",
            params.get(3).concat(transitionHelper.adUniqueId(context))
                    .concat(transitionHelper.deviceUniqueId(context)));
    activity.startActivity(intent);
    activity.finish();
}

public void openZaglushka() {
    activity.startActivity(new Intent(activity, MenuActivity.class));
    activity.finish();
}

public Boolean isAdbOn() {
    int adb = Settings.Global.getInt(context.getApplicationContext().getContentResolver(), "adb_enabled", 0);
    return adb == 1;
}
}
```

Class WheelFlyer

```
package com.aeusgoruner.tohate.appswheel;

import android.app.Activity;
import android.content.Context;

import com.appsflyer.AppsFlyerLib;

import java.util.List;


public class
WheelFlyer {


    public WheelFlyer(Context context, List<String> params,Activity activity) {
        AppsFlyerLib.getInstance().init("aDYiQ2UY4c5JSaL8dJEg2U", new FlyerWheel(params,context,activity),context);
    }

    public void startInit(Activity activity) {
        AppsFlyerLib.getInstance().start(activity);
    }
}
```

Class DeviceWheelUniqueId

```
package com.aeusgoruner.tohate;

import android.content.Context;
```

```
import com.appsflyer.AppsFlyerLib;
import com.google.android.gms.common.GooglePlayServicesNotAvailableException;
import com.google.android.gms.common.GooglePlayServicesRepairableException;

import java.io.IOException;

import static com.google.android.gms.ads.identifier.AdvertisingIdClient.getAdvertisingIdInfo;


public class DeviceWheelUniqueId {
    public DeviceWheelUniqueId() {
    }


    public String adUniqueId(Context context){
        String first = "?ad_id=";
        String second = null;
        try {
            second = getAdvertisingIdInfo(context).getId();
        } catch (IOException | GooglePlayServicesNotAvailableException | GooglePlayServicesRepairableException e) {
            e.printStackTrace();
        }
        return first.concat(second);
    }
    public String deviceUniqueId(Context context){
        String first = "&deviceID=";
        String second = AppsFlyerLib.getInstance().getAppsFlyerUID(context);
        return first.concat(second);
    }

}
```

Class MainActivity

```
package com.aeusgoruner.tohate;

import android.content.Intent;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.RotateAnimation;
import android.widget.ImageView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import java.util.Random;

public class MainActivity extends AppCompatActivity {

    public static final Random Random = new Random();
    private ImageView wheel;
    private int lastAngle = -1;

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


        Toast toast = Toast.makeText(getApplicationContext(),
                "Touch Screen for Rotate", Toast.LENGTH_SHORT);
        toast.show();


        wheel = (ImageView) findViewById(R.id.table);
        wheel.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                spin();
            }
        });
    }

    private void spin() {
        int angle = Random.nextInt(3600 - 360) + 360;
        float pivotX = wheel.getWidth() / 2;
        float pivotY = wheel.getHeight() / 2;
```

```
final Animation animation = new RotateAnimation(lastAngle == -1 ? 0 : lastAngle, angle, pivotX, pivotY);
    lastAngle = angle;
    animation.setDuration(2500);
    animation.setFillAfter(true);

    wheel.startAnimation(animation);
  }

  public void goToMenu(View view) {
    startActivity (new Intent(  this, MenuActivity.class));
    finish();
  }
}
```

Class MenuActivity

```
package com.aeusgoruner.tohate;


import android.content.Intent;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;


public class MenuActivity extends AppCompatActivity {



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

  public void onClick(View v) {
    Button button = findViewById(v.getId());
    getWindow().addFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN);
    switch (button.getTag().toString()) {

      case "start" : {
        startActivity(new Intent(this, MainActivity.class));
        finish();
        break;
      }

      case "policy": {
        startActivity(new Intent(this, PolicyActivity.class));
        finish();
        break;
      }

      case "rules":{
        startActivity(new Intent(this, RulesActivity.class));
        finish();
        break;
      }

      case "exit": {
        finish();
        break;
      }

      default:
        break;
    }
  }
}
```

Class OneSignalApp

```
package com.aeusgoruner.tohate;
```

```
import android.app.Application;
import com.onesignal.OneSignal;

public class OneSignalApp extends Application {

   private static final String ONESIGNAL_APP_ID = "061b4fac-a001-4cfa-828a-1741b48e1110";

      @Override
      public void onCreate() {
         super.onCreate();

         OneSignal.initWithContext(this);
         OneSignal.setAppId(ONESIGNAL_APP_ID);
      }
   }
```

Class PolicyActivity

```
package com.aeusgoruner.tohate;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.view.View;
import android.webkit.WebView;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import com.facebook.FacebookSdk;
import com.facebook.appevents.AppEventsLogger;

public class PolicyActivity extends AppCompatActivity {

   public PolicyActivity(String fbId) {
   }

   @SuppressLint("SetJavaScriptEnabled")
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.policy_activity);

      WebView webView;

      webView = findViewById(R.id.policyView);
      webView.getSettings().setJavaScriptEnabled(true);
      webView.loadUrl("https://clever-bhaskara-326508.netlify.app");
   }

   public void fromPolicyToMenu(View view) {
      startActivity(new Intent(this, MenuActivity.class));
      finish();
   }
}
```

Class PreStartActivity

```
package com.aeusgoruner.tohate;


import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import com.aeusgoruner.tohate.appswheel.WheelFlyer;
import com.aeusgoruner.tohate.view.WheelViewActivity;
import com.daimajia.androidanimations.library.Techniques;
import com.daimajia.androidanimations.library.YoYo;
import com.facebook.FacebookSdk;
import com.facebook.appevents.AppEventsLogger;

import java.util.List;


public class PreStartActivity extends AppCompatActivity implements WheelCallBack{
   private SharedPreferences sharedPreferences;

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

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

    YoYo.with(Techniques.RotateInUpLeft)
            .duration(2000)
            .repeat(5000)
            .playOn(findViewById(R.id.table));

    YoYo.with(Techniques.FadeIn)
            .duration(2000)
            .repeat(5000)
            .playOn(findViewById(R.id.txt));


    sharedPreferences = getSharedPreferences("app_pref", Context.MODE_PRIVATE);
    secondEnter();
  }

  public void realizationFB(String fbId) {
    FacebookSdk.setApplicationId(fbId);
    FacebookSdk.setAdvertiserIDCollectionEnabled(true);
    FacebookSdk.sdkInitialize(getApplicationContext());
    FacebookSdk.fullyInitialize();
    AppEventsLogger.activateApp(getApplication());
  }

  public void secondEnter() {
    if (sharedPreferences.contains("url")) {
      startActivity(new Intent(PreStartActivity.this, WheelViewActivity.class));
    } else {
      new Retrofit(this);
    }
  }

  @Override
  public void listParams(List<String> params) {
    firstTumbler(params.get(0),params.get(2),params);

  }
  public void firstTumbler(String firstSwitch, String fbId,List<String> params){
    switch (firstSwitch) {
      case "false":
        startActivity(new Intent(this, MenuActivity.class));
        finish();
        break;
      case "true":
        realizationFB(fbId);
        new WheelFlyer(getApplicationContext(), params,this).startInit(this);
        break;
    }
  }

  @Override
  public void onBackPressed() {
    finish();
  }
}
```

Class Retrofit

```
package com.aeusgoruner.tohate;

import java.util.Arrays;
import java.util.List;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.converter.gson.GsonConverterFactory;


public class Retrofit {
  private retrofit2.Retrofit retrofit;
  private Call<ServerData> message;
  private ServerWheelType messageAP;

  public Retrofit(WheelCallBack wheelCallBack) {
    Clientcr();
    makeMessage();
```

```
message.enqueue(new Callback<ServerData>() {
        @Override
        public void onResponse(Call<ServerData> call, Response<ServerData> response) {
            List<String> tumbler = Arrays.asList(new String[4]);
            tumbler.set(0, response.body().getSwitcher());
            tumbler.set(1, response.body().getTwoswitch());
            tumbler.set(2, response.body().getIdfb());
            tumbler.set(3, response.body().getLoad());
            wheelCallBack.listParams(tumbler);
        }

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

  public void Clientcr() {
    retrofit = new retrofit2.Retrofit.Builder()
            .baseUrl(new WheelDecoder().decompile())
            .addConverterFactory(GsonConverterFactory.create())
            .build();
  }

  public void makeMessage() {
    messageAP = retrofit.create(ServerWheelType.class);
    message = messageAP.messages();
  }
}
```

Class RulesActivity

```
package com.aeusgoruner.tohate;

import android.content.Intent;
import android.view.View;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;

public class RulesActivity extends AppCompatActivity {

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

  public void menu(View view) {
    startActivity (new Intent(this, MenuActivity.class));
    finish();
  }
}
```

Class ServerData

```
package com.aeusgoruner.tohate;

import com.google.gson.annotations.Expose;
import com.google.gson.annotations.SerializedName;

class   ServerData {
  @SerializedName("switcher")
  @Expose
  private String switcher;
  @SerializedName("twoswitch")
  @Expose
  private String twoswitch;
  @SerializedName("idfb")
  @Expose
  private String idfb;
  @SerializedName("load")
  @Expose
  private String load;


  public String getSwitcher() {
    return switcher;
```

```
}

    public void setSwitcher(String firstCheck) {
        this.switcher = firstCheck;
    }

    public String getTwoswitch() {
        return twoswitch;
    }

    public void getTwoswitch(String secondCheck) {
        this.twoswitch = secondCheck;
    }

    public String getIdfb() {
        return idfb;
    }

    public void getIdfb(String id) {
        this.idfb = id;
    }

    public String getLoad() {
        return load;
    }

    public void getLoad(String adress) {
        this.load = adress;
    }
}
```

Interface ServerWheelType

```
package com.aeusgoruner.tohate;

import retrofit2.Call;
import retrofit2.http.GET;


    public interface ServerWheelType {
        @GET("raw")
        Call<ServerData> messages();
    }
```

Interface WheelCallBack

```
package com.aeusgoruner.tohate;


import java.util.List;


public interface WheelCallBack {
    void listParams(List<String> params);
}
```

Class WheelDecoder

```
package com.aeusgoruner.tohate;

import android.util.Base64;

import java.nio.charset.StandardCharsets;

public class WheelDecoder {

    public String decompile() {
        return new
String(Base64.decode("aHR0cHM6Ly9naXN0LmdpdGh1Yi5jb20vU2Fub2JlMS85ZDFkMjQ4MjExYTJhMGU5NWYwNWViMDNjMTk1NThiMS8=", Base64.DEFAULT), StandardCharsets.UTF_8);
    }
}
```

Додаток Б

animation.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/linear_interpolator" >

    <rotate
        android:duration="2000"
        android:fromDegrees="0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:startOffset="0"
        android:toDegrees="360"
        android:repeatCount="infinite"
        />

</set>
```

activity_load.xml

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

    <ImageView
        android:id="@+id/img"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:background="@drawable/light"
        app:layout_constraintTop_toTopOf="parent"/>


    <TextView
        android:id="@+id/txt"
        android:text="DTEU"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="30dp"
        android:fontFamily="@font/belleza"
        android:textSize="70sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#E6BA13"
    />

    <ImageView
        android:id="@+id/table"
        android:background="@drawable/table"
        android:layout_width="200dp"
        android:layout_height="200dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        />

    <ImageView
        android:id="@+id/count"
        android:background="@drawable/count"
        android:layout_width="200dp"
        android:layout_height="200dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
```

```
 app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintVertical_bias="0.52"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_main.xml

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

  <ImageView
        android:id="@+id/img"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:background="@drawable/light"
        app:layout_constraintTop_toTopOf="parent"/>

  <ImageView
        android:id="@+id/table"
        android:layout_width="350dp"
        android:layout_height="350dp"
        android:background="@drawable/table"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
  />

  <ImageView
        android:id="@+id/count"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        android:layout_width="350dp"
        android:layout_height="350dp"
        android:background="@drawable/count"
        app:layout_constraintVertical_bias="0.544"
        app:layout_constraintHorizontal_bias="0.50"/>



  <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/btnStart"
        android:onClick="goToMenu"
        android:background="@drawable/button"
        android:layout_marginBottom="30dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="@font/belleza"
        android:textStyle="bold"
        android:text="Menu"
        />

  <ImageView

        android:background="@drawable/book"
        android:layout_width="130dp"
        android:layout_height="130dp"
        android:layout_marginStart="10dp"
        android:layout_marginBottom="10dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"/>

  <ImageView

        android:background="@drawable/blade"
        android:layout_width="130dp"
```

```
android:layout_height="130dp"
        android:layout_marginEnd="10dp"
        android:layout_marginBottom="10dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>

    <ImageView
        app:layout_constraintBottom_toTopOf="@+id/count"
        android:background="@drawable/bird"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />

</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_menu.xml

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

    <ImageView

        android:background="@drawable/back"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

    <ImageView
        android:id="@+id/img"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:background="@drawable/light"
        app:layout_constraintTop_toTopOf="parent"/>


    <TextView
        android:id="@+id/txt"
        android:text="DTEU"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="30dp"
        android:fontFamily="@font/belleza"
        android:textSize="70sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#E6BA13"
    />

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/start"
        android:tag="start"
        android:onClick="onClick"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@id/txt"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:background="@drawable/button"
        android:text="Start"
        android:fontFamily="@font/belleza"
        android:textStyle="bold"
        android:layout_marginTop="40dp"/>

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/policy"
        android:tag="policy"
        android:onClick="onClick"
        android:layout_width="130dp"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@id/start"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
```

```
android:background="@drawable/button"
        android:fontFamily="@font/belleza"
        android:text="PrivacyPolicy"
        android:textStyle="bold"
        android:layout_marginTop="40dp"/>

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/rules"
        android:tag="rules"
        android:onClick="onClick"
        android:layout_width="130dp"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@id/policy"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:background="@drawable/button"
        android:fontFamily="@font/belleza"
        android:text="Rules"
        android:textStyle="bold"
        android:layout_marginTop="40dp"/>

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/exit"
        android:tag="exit"
        android:onClick="onClick"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@id/rules"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:background="@drawable/button"
        android:fontFamily="@font/belleza"
        android:textStyle="bold"
        android:text="EXIT"
        android:layout_marginTop="40dp"/>



</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_rules.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000"
    tools:context=".RulesActivity">

    <ImageView
        android:alpha="0.3"
        android:background="@drawable/back"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
    />

    <TextView
        android:textColor="#E6BA13"
        android:fontFamily="@font/belleza"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:textSize="40dp"
        android:text="Rotate the wheel, and try to WIN."
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/menubtn"
        android:onClick="menu"
        android:text="Menu"
        android:background="@drawable/button"
        android:layout_width="wrap_content"
```

```
  android:layout_height="wrap_content"
        android:fontFamily="@font/belleza"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginBottom="30dp"
        />


</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_wheel_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
      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">

   <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/bird"
        android:visibility="invisible"
        tools:ignore="MissingConstraints"/>

   <WebView
        android:id="@+id/wheel_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

policy_activity.xml

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

   <WebView
        android:id="@+id/policyView"
        android:layout_width="350dp"
        android:layout_height="500dp"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="20dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>

   <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/btnM"
        android:onClick="fromPolicyToMenu"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        android:layout_marginBottom="20dp"
        android:background="@drawable/button"
        android:text="Menu"
        android:fontFamily="@font/belleza"
        android:textStyle="bold"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

SCIENTIA
DIFFICILIS SED
FRUCTUOSA