**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ**
**КАФЕДРА КОМП'ЮТЕРНИХ НАУК**

**Секція інформаційно-комунікаційних технологій**

# ВИПУСКНА РОБОТА

## на тему:

## «Android-додаток для спільного просування каналів соціальних мереж, сторінок та брендів»

**Завідувач**
**випускаючої кафедри** **Довбиш А.С.**

**Керівник роботи** **Власенко О.В.**

**Студент групи ІНз - 71с** **Василишин В.Р.**

**СУМИ 2021**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Затверджую_____
Зав. кафедрою Довбиш А.С.
"_____" _____2021 р.

**ЗАВДАННЯ**
**до випускної роботи**

Студента п'ятого курсу, групи ІНз-71с спеціальності «122 – Комп'ютерні науки» заочної форми навчання Василишина Віктора Руслановича.

**Тема:** «Android-додаток для спільного просування каналів соціальних мереж, сторінок та брендів»

Затверджена наказом по СумДУ
№_____ від_____2021 р.

Зміст пояснювальної записки: 1) аналітичний огляд потреби в створенні Android-додатку для колаборації та спільного просування каналів соціальних мереж, сторінок та брендів; 2) постановка завдання та формування завдань розробки; 3) опис основних методів, та правил розробки мобільного додатку для операційної системи Android; 4) розробка інформаційного та програмного забезпечення Android-додатку; 5) аналіз результатів розробки.

Дата видачі завдання "_____"_____2021 р.
Керівник випускної роботи _____ Власенко О.В.
Завдання прийняв до виконання _____ Василишин В.Р.

# РЕФЕРАТ

**Записка:** 34 стор., 7 рис., 6 додатків, 13 джерел.

**Об’єкт дослідження** — сучасні методи розробки мобільних додатків для операційної системи Android.

**Мета роботи** — розробка mvp версії додатку для колаборації користувачів з метою спільного просування каналів соціальних мереж, сторінок та брендів.

**Методи дослідження** — функціонально статистичні дослідження.

**Результати** — розроблено алгоритм та програмне забезпечення мобільного додатку для пошуку власників соціальних каналів, сторінок та брендів, для спільних маркетингових та рекламних кампаній. Створений програмний продукт максимально задовольняє сучасні вимоги програмування додатків для операційної системи Android. Розроблений програмний продукт створено за допомогою середовища розробки Android Studio, з використанням мови програмування Kotlin.

СОЦІАЛЬНІ МЕРЕЖІ КАНАЛИ ТА БРЕНДИ, АНДРОЇД ДОДАТОК, ПЕРЕХРЕСНЕ ПРОСУВАННЯ, АРХІТЕКТУРНІ РІШЕННЯ МОБІЛЬНИХ ДОДАТКІВ, MBaaS СЕРВІСИ, ХМАРНЕ ЗБЕРЕЖЕННЯ ДАНИХ GOOGLE FIREBASE

# ЗМІСТ

ВСТУП ..... 5
1 АНАЛІЗ ПРОБЛЕМИ, ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ ..... 6
1.1 Аналітичний огляд потреби створення Android-додатку для колаборації у просуванні каналів, сторінок та брендів ..... 6
1.2 Сучасні стандарти розробки Android-додатків ..... 7
1.3 Постановка задачі ..... 9
2. MBaaS СЕРВІСИ ДЛЯ СТВОРЕННЯ БЕКЕНДУ ТА ХМАРНОГО ЗБЕРЕЖЕННЯ ДАНИХ КОРИСТУВАЧІВ ..... 10
2.1 Аналіз MBaaS сервісу Microsoft Azure. ..... 10
2.2 Аналіз MBaaS сервісу Google Firebase ..... 13
2.3 Аналіз MBaaS сервісу AWS Amplify ..... 16
3 РОЗРОБКА MVP ДОДАТКУ ТА ІНТЕГРАЦІЯ CLOUD FIRESTORE ..... 19
3.1 Основна активність, реєстрування та авторизація користувача. ..... 19
3.2 Створення та інтеграція консолі Google firebase. ..... 21
3.3 Тестування та виправлення багів. ..... 24
ВИСНОВКИ ..... 26
СПИСОК ЛІТЕРАТУРИ ..... 27
ДОДАТОК 1
ДОДАТОК 2
ДОДАТОК 3
ДОДАТОК 4.1
ДОДАТОК 4.2
ДОДАТОК 4.3

# ВСТУП

З кожним роком просування свого бренду в соціальних мережах стає важчим і дорожчим. Переважна більшість власників блогів, каналів чи сторінок соціальних мереж, не мають можливості вкладати достатньо грошей у їх розвиток, щоб бути конкурентоспроможними. Існує, досить, вдале рішення для вирішення цієї проблеми - cross-promotion просування бренду, каналу, сторінки тощо[1]. Суть цього підходу в колаборації власників суміжних брендів, що можуть доповнити один одного, та організувати спільну рекламну кампанію. Таким чином, витрати на просування бренду можуть бути зменшені, або зовсім відсутні, а обидва бренди здобудуть нових користувачів, підписників тощо.

Переважна більшість сервісів, для розвитку діяльності у соціальних мережах, пропонують платні послуги адміністрування чи удосконалення каналів, SMM, рекламу, та різноманітні курси для навичок розширення аудиторії свого бренду. Такий сервіс як пошук бажаючих організувати перехресне просування майже не представлений серед послуг. Є припущення, що цей вид діяльності розглядається компаніям, що заробляють на послугах з SMM, не тільки як безприбутковий, але й такий, що може прибутки скоротити. Адже перехресне просування може відбуватися, безпосередньо, між адміністраторами каналів чи сторінок, без будь яких посередників і давати достойні результати.

Оскільки сервіс колаборації для спільного перехресного просування контенту не вимагає спеціалізованої роботи з робочого місця, то саме мобільний додаток може стати зручним способом вирішення організаційних питань - пошук бажаючих, встановлення первинного контакту, вирішення деталей та тонкощів.

# 1. АНАЛІЗ ПРОБЛЕМИ, ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ

## 1.1 Аналітичний огляд потреби створення Android-додатку для колаборації у просуванні каналів, сторінок та брендів

Соціальні мережі об’єднують мільйони людей і є привабливим місцем для просування бізнесу чи приватної діяльності, бо вся ця аудиторія є потенційними клієнтами чи покупцями. Замість об’яв у газетах, або вуличної реклами, переважна більшість підприємців, брендів, авторів чи творців іншого контенту, знаходять своїх прихильників в соціальних мережах. Розвиток власної сторінки чи каналу потребує багато часу, інтелектуальних ресурсів, та, у багатьох випадках, фінансових інвестицій.

Велика інформаційна насиченість та конкуренція створює певні складнощі у залученні нових прихильників, для більшості творців контенту. Адміністратори соціальних каналів та груп, постійно знаходяться в процесі залучення нових прихильників своєї творчості. Існує багато платних варіантів просування каналу чи бренду. Можно скористатися послугами маркетологів, чи пропозиціями реклами тієї платформи, на якій ведеться діяльність. Та є певні складнощі – такі рекламні акції потребують значних фінансових інвестиції, але жодних гарантій відповідного притоку нових прихильників дати не зможуть.

Вдалим рішенням для заощадження коштів та залучення нових прихильників, є співпраця з іншими творцями чи брендами організована у формі перехресного просування (Cross-Promotion). В основі цього прийому лежить колаборація між двома, або більше, компаніями, брендами, каналами соціальних мереж чи адміністраторами сторінок, для створення спільного контенту чи рекламних акцій, спрямованих на стимулювання збуту, підвищення впізнаваності, залучення нових підписників.

Головні переваги перехресного просування:

- учасники отримують доступ до, вже наявної, аудиторії один одного, яка буде більш лояльною до нових пропозицій;

● фінансові витрати можуть бути нижчими за рахунок їх розділення між двома учасниками акції, або відсутні взагалі;

● спільна робота над контентом чи рекламною акцією дозволить розділити обов'язки, що заощаджує час та інтелектуальні можливості;

● обмін досвідом та новими ідеями між суміжними командами дозволяють знайти максимально цікаві ідеї для спільного просування.

Методи перехресного просування досить активно використовуються у соціальних мережах. Наприклад на YouTube, творці створюють спільні відео, або можуть згадати один одного у черговому випуску. На таких платформах як Telegram і Facebook, адміністратори діляться посиланнями на групи своїх партнерів тощо.

Не дивлячись на це, при аналізі мережі Internet, та Google Play не вдалося знайти жодного додатку, що позиціонував би себе як платформу для колаборації для спільних Cross-Promotion акцій. Натомість існують багато сервісів, що пропонують платні послуги розвитку та просування каналів чи сторінок.

Виходячи з цього, зроблено висновок, що ідея додатку для комунікації творців та адміністраторів соціальних калів, з метою колаборації і спільного просування своїх брендів має певні перспективи.

### 1.2 Сучасні стандарти розробки Android-додатків

Існують певні стандарти що визначають якість мобільних Android-додатків. Якість роботи програмного продукту впливає на досвід користування ним, що формується у користувачів. Позитивний досвід використання додатку сприяє зростанню кількості постійних користувачів, та залученню нових.

Додаток має бути максимально стабільний на всіх, або хоча б, на більшості, Android пристроях. Усі "слабкі місця", що можуть призвести до помилок та екстреного завершення роботи додатку, мають бути оброблені спеціальними запобіжними кодом. Інтерфейс додатку має бути простим, та інтуїтивно зрозумілим для користувача, без додаткових інструкцій. При натисканні на певні

елементи інтерфейсу, мають відбуватися цілком очікувані дії. Також важливими вимогами є асинхронна робота процесів, запитів до сервера чи локальної бази даних, прослуховування команд користувача. В процесі обробки даних, чи тимчасових запитів на сервер, ні в якому разі, не повинно відбуватися блокування інтерфейсу до момента отримання відповіді чи результату. Додаток має максимально швидко здійснювати заявлені обчислення чи процеси, та займати мінімальний об’єм пам’яті.

Java – об’єктно орієнтована мова, котра за, більш ніж, 25 років існування, зайняла доволі міцні позиції у створенні програмного забезпечення для великої кількості операційних систем, у тому числі і для Android. На думку спеціалістів, попри всі свої сильні сторони, Java не має динамічності та гнучкості у програмуванні під Android, тому розробка відбувається повільно, а тестування та виправлення багів потребує додаткового часу.

Kotlin – досить молода мова програмування, котра була створена, з урахуванням сучасних потреб у програмуванні. З 2018 р. Google офіційно представив Kotlin як найбільш рекомендовану мову для програмування Android додатків. Kotlin має конкурентоспроможні, порівняно з Java, теоретичну та практичну базу. Більш того, Kotlin набагато динамічніша, та гнучкіша що скорочує процес написання додатків [2, 4]. Після детального аналізу завдання, наявного часу, та ресурсів, було вирішено використати мову програмування Kotlin для створення прототипу Android-додатку.

Для того щоб отримати якісний і стабільний продукт, існують певні патерни програмування та архітектурні рішення. Один з них – Dagger2, що бере на себе відповідальність, постачання залежностей у потрібні місця, без постійного створення об’єктів. Це дуже важливе архітектурне рішення, що робить проект Android-додатку гнучкішим, підтримуваним та простим для розширення. Об’єкти можуть бути перевикористані, та не займають ресурси пристрою без потреби. Також зменшується об’єм пам’яті необхідний для інсталювання та використання додатку [5].

Корпорація Google, хоч і з невеликим запізненням, але усвідомила важливість єдиного набору інструментів розробки під платформу, власником якої вона є. Google Architecture Components значно спрощують, та прискорюють розробку додатків. Цей комплекс бібліотек та архітектурних рішень полегшує інтеграцію роботи з картами, створення графічного інтерфейсу, навігації, роботи з локальною базою даних, тощо [3].

Майже вся робота Android-додатку є асинхронною. Ніколи не відомо, коли система отримує сигнал від користувача, дані з бази даних чи результати обчислень з паралельного потоку. Тому використання асинхронних механізмів роботи, які не будуть блокувати роботу додатку протягом очікуванням тих чи інших даних, але будуть “слухачами” і реагувати на вхідні дані, залежно від їх типу. RxKotlin інкапсулює в собі механізми, що дозволять зробити роботу Android-додатку чуйною та зручною для користувача.

### 1.3 Постановка задачі

Головною задачею роботи є проектування та створення minimum viable product (MVP) додатку, інтеграція хмарного збереження даних користувачів, та публікація додатку до Google Play. У MVP додатку мають бути реалізований наступні функціональні можливості:

- реєстрація та авторизація користувачів з e-mail та паролем;
- відображення профайлу зареєстрованого та авторизованого користувача;
- редагування інформації про канал чи сторінку яка має бути використана для просування;
- перегляд матеріалів доданих іншими користувачами.

Для задоволення функції хмарного збереження даних мають були створені та інтегровані у додаток бекенд можливості за допомогою одного з наявних MBaaS сервісів.

# 2. MBAAS СЕРВІСИ ДЛЯ СТВОРЕННЯ БЕКЕНДУ ТА ХМАРНОГО ЗБЕРЕЖЕННЯ ДАНИХ КОРИСТУВАЧІВ

Переважна більшість мобільних додатків потребує створення серверної частини для своєї роботи. Особливо, якщо має відбуватися обмін даними між користувачами. Розробка та підтримка бази даних потребує не менше, а часами навіть більше уваги і ресурсів ніж фронтенд частина. У випадку, якщо створюваний додаток буде перебувати якийсь час у тестовому режимі, або не передбачає велику кількість трафіку на початку існування, можна скористатись послугами сервісів Mobile Backend-as-a-Service (MBaaS), котрі надають послуги зі зберігання даних, та можливість доступу до них.

Процеси створення бекенд за допомогою MBaaS спрощені, в порівнянні з розробкою “з нуля”, що заощаджує час та кошти. Провайдер MBaaS бере на себе всі питання, пов'язані зі стабільністю серверів, балансуванням навантаження, масштабованістю та іншими інфраструктурами складнощами. Більшість MBaaS провайдерів надають певну кількість трафіку безкоштовно, що цілком достатньо для початкового, або тестового етапу розвитку додатку.

Для порівняння проаналізовано три великих MBaaS сервісів що добре зарекомендували себе: Microsoft Azure, Google Firebase, AWS Amplify. Критерії, за якими розглянемо сервіси: функціональність бекенд і аналітика, складність інтеграції сервісу, надійність і стабільність роботи, цінова політика.

## 2.1 Аналіз MBaaS сервісу Microsoft Azure.

Microsoft Azure – Infrastructure-As-A-Service (IaaS) сервіс, який містить в собі повноцінну BaaS функціональність і допомагає при створенні бекенд для мобільних додатків.

Microsoft Azure має у своєму розпорядженні повний набір функціональності для створення бекенд мобільного додатку – обробка push-повідомлень, автоматичне масштабування, синхронізація даних, інтеграція з соціальними мережами і багато іншого [6].

Важлива особливість Azure – географічне положення серверів. Вони розташовані в 54 регіонах світу, що підвищує ймовірність серверу з максимально швидкою передачею даних (рис. 2.1). Також допускаємо, що велика сегментація регіонів, сприяє ймовірність справної роботи серверу. Як стверджують Microsoft, у них більше регіонів, ніж у будь-якого іншого постачальника хмарних рішень. Це, безсумнівно, плюс.


**Рисунок 2.1** – Розташування серверів Azure

Сервіс надає можливість в реальному часі слідкувати за працездатністю додатків і збирати звіти про їх роботу на пристроях користувачів. Це дозволяє динамічно локалізовувати та вирішувати можливі проблеми.

Також в Azure надається власна бібліотека для збирання аналітики в додатках: основні метрики (інформація про пристрій, сесії, активність користувачів тощо), створення власних подій для відстеження. Всі зібрані дані відразу експортуються в Azure, дозволяючи проводити з ними аналітичну роботу в зручному форматі

Існують цікаві функції типу тестування збірок додатків на реальних пристроях, налаштування CI / CD для автоматизації процесу розробки, та інструментарій для відправки збірок додатків на бета-тестування, або відразу в App Store та Google Play

Azure дозволяє використовувати фреймворк «з коробки» призначений для роботи з картами і геопросторовими даними, що спрощує роботу з подібним форматом.

Особливо цікаво виглядає можливість вирішення завдань з використанням штучного інтелекту, за допомогою якого можна прогнозувати різні аналітичні показники і використовувати готові до роботи інструменти для комп'ютерного зору, розпізнавання мови, тощо.

Сервіс Microsoft Azure надає SDK для основних мобільних платформ (iOS та Android) та кросплатформенних рішень (Xamarin і PhoneGap).

В цілому, користувачі скаржаться на складний інтерфейс і високий поріг входження в користування сервісом, що говорить про можливі проблеми його інтеграції на початковому етапі. Але високий поріг входження – не окремий випадок з Azure, а загальна проблема для IaaS. Наприклад, Amazon Web Services, який буде розглянуто далі, також схильний до даної особливості.

Стабільність сервісу від Microsoft виглядає гідно. Можна побачити, що в середньому, раз на місяць можуть відбуватися короткочасні технічні проблеми в різних регіонах (рис. 2.2). Але, порівняно із загальною картиною, це вважається нормою. Більш того, проблеми швидко вирішуються, дозволяючи сервісу тримати гідний uptime [11].

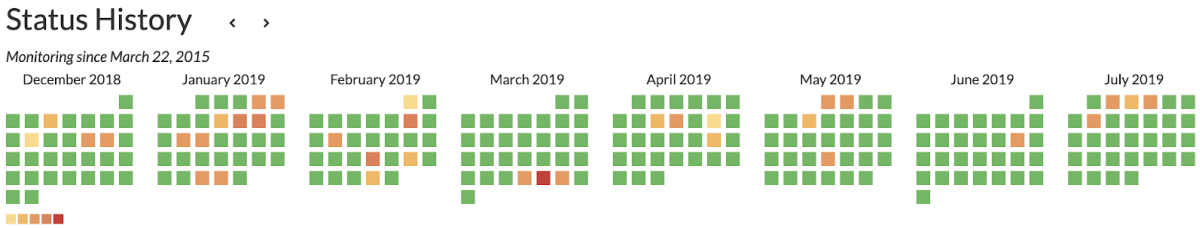



**Рисунок 2.2** – Відображення технічних проблем

У ціновій політиці Microsoft Azure існують різні тарифи оплати сервісу. Серед них – безкоштовний план з певними лімітами, якого має вистачити для тестування додатку.

Важливо пам'ятати, що Azure – IaaS сервіс, більшість з яких через свою специфіку і складність підрахунку відпрацьованих ресурсів, мають складнощу пов’язані з прогнозуванням вартості користування. Багато з них не можуть наперед розрахувати потужності, що будуть використані, тому реальний рахунок за користування послугами може значно відрізнятися запланованого. У багатьох відгуках користувачі скаржаться на складну цінову політику і неможливість прогнозування вартості послуг сервісу. Запропонований Microsoft калькулятор називають марним, а сам сервіс вкрай дорогим.

Також у Azure, крім цих планів, є окремі платні послуги: App Service Domain, Azure App Service Certificates і SSL Connections. Всі вони відносяться до адміністрування інфраструктури додатку, на даному етапі розвитку якого, потреби в таких послугах немає.

Сервіс Azure від Microsoft – функціональний і стабільний інструмент для використання в якості основного MBaaS провайдера. Він з самого початку надає повноцінну інфраструктуру, відкриває безліч можливостей для подальшого розвитку бекенд частини проекту поза рамками мобільного додатку. Велика кількість серверів і регіонів, де вони розташовані, допомагає підібрати більш вдалі для конкретного додатку. Позитивні відгуки користувачів це підтверджують. З негативних моментів – високий поріг входження і складності з прогнозуванням вартості роботи сервісу.

### 2.2 Аналіз MBaaS сервісу Google Firebase

Сервіс Firebase від Google є одним з найцікавіших варіантів як MBaaS сервісу для застосування. Він зарекомендував себе в якості корисного інструменту і є таким для багатьох відомих додатків: Shazam, Duolingo, Lyft та інших [9].

Firebase забезпечує всіма функціями, що можуть знадобитися мобільному додатку. Сервіс поєднує в повноцінні бекенд-можливості (рис. 2.3), такі як зберігання даних, синхронізація, аутентифікація, хмарні функції (виконання бекенд коду), Machine Learning Kit що реалізується в додатку різна функціональність на основі машинного навчання (розпізнавання тексту, об'єктів на фотографіях і багато іншого).

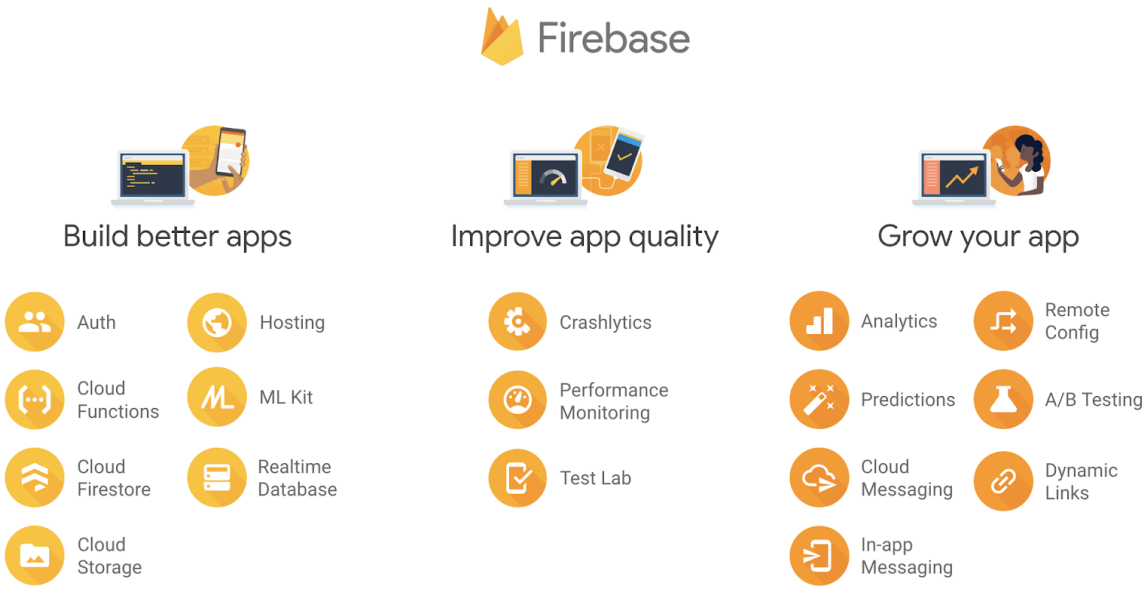



**Рисунок 2.3** – Функції сервіса Firebase від Google

Важлива особливість Firebase в тому, що крім бекенд функціональності, сервіс пропонує широкий спектр можливостей для аналітики додатку. Вбудована Google Analytics, сегментування призначеної для користувача бази і робота з push-повідомленнями. Також широко поширений сервіс Fabric інтегрований в Firebase поряд з Crashlytics, є вкрай корисним інструментом для відстеження, збору статистики і звітів про помилки у додатку, що сталися на пристроях користувачів.

Firebase надає інструмент Firebase Dynamic Links для обробки динамічних посилань на ваш контент, за допомогою цього інструменту можна генерувати посилання, які ведуть в певний додаток, якщо він встановлений на пристрої,

якщо ж ні – відправляють користувача в App Store або Google Play для встановлення додатку.

Google дозволяє проводити A/B тестування програм за допомогою Firebase A/B Testing і налаштовувати віддалену конфігурацію з інструментом Remote Config.

Цей сервіс поєднує в собі вкрай велику кількість можливостей. Для інтеграції Firebase варто використовувати SDK необхідної платформи, серед яких iOS, Android, JavaScript, а також для C++ і Unity. Важливо відзначити, що у Firebase досить докладна документація і широка база користувачів-розробників, і як наслідок, велика кількість допоміжного контенту в мережі – відповіді на питання, оглядові статті тощо.

З одного боку, у Google Firebase це високостабільний провайдер, але все одно можна знайти безліч повідомлень про проблеми з аптаймом [11]. Наприклад, цитата одного з користувачів: «Downtime happens. In the case of Firebase, you might say that «uptime» happens». І дійсно, якщо подивитися на статистику щодо подій з сервісами Firebase, побачимо, що бувають як невеликі простої, так і повноцінні відключення на 5-7 годин (рис. 2.4). Це може бути критично для будь якого додатку, особливо, якщо на цих сервісах працює життєво важливий для продукту код.

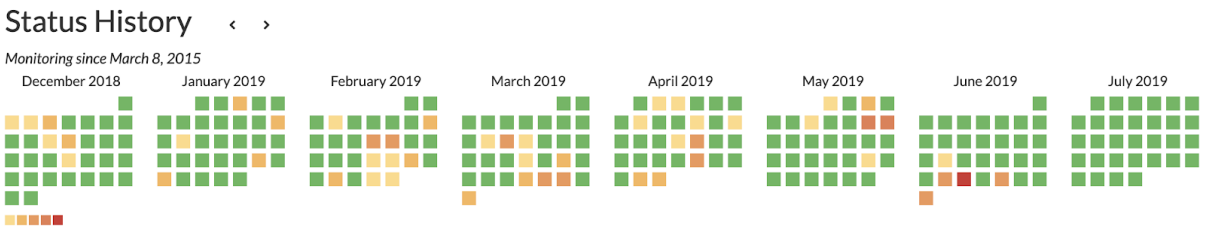



**Рисунок 2.4** – Технічні збої сервіса Firebase

Цінова політика Firebase зрозуміла і проста, є три тарифних плани: Spark, Flame і Blaze. Вони ідеологічно відрізняються один від одного. У той час як Spark – безкоштовний план з лімітами, що дозволяють розгорнути і протестувати значну частину функціональності платформи. Плани Flame і Blaze припускають

платне використання. Вартість Flame фіксована 25 $ в місяць. По суті ви отримуєте той же Spark, тільки зі значно більшими лімітами [10].

Blaze відрізняється від інших. Він дозволяє використовувати можливості платформи в необмежених кількостях, а оплата здійснюється пропорційно ресурсів, які використовуються. Це вкрай гнучкий план, в якому сплачується тільки та функціональність, яка використовується. Якщо, наприклад, використовувати платформу тільки для тестування додатку – то і сплачується лише перевищення безкоштовних лімітів по тестуванню. Загалом, ціноутворення Firebase досить прозоре і прогнозоване (рис. 2.5).

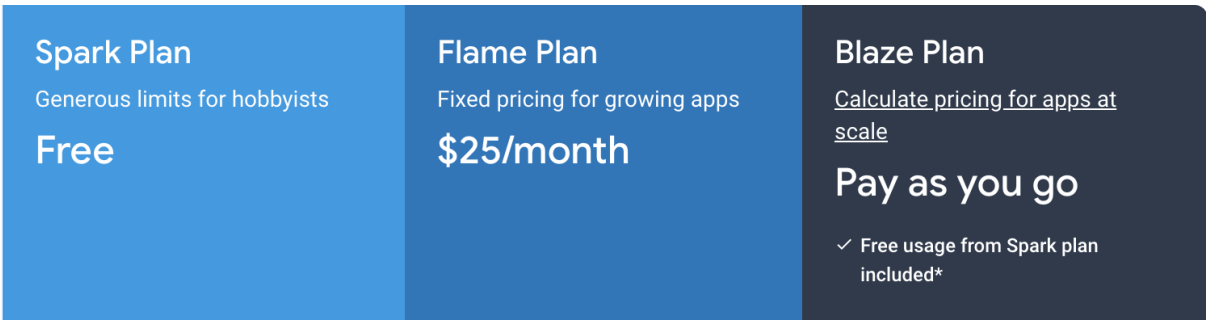



**Рисунок 2.5** – Цінова політика сервіса Firebase від Google

Сервіс Firebase від Google представляє собою повноцінний MBaaS провайдер, який обмежує від інфраструктурних складнощів, з якими безпосередньо пов'язані AWS і Azure. Присутній весь необхідний для розробки хмарного бекенду функціонал. Досить широкі можливості для аналітики, відносна простота інтеграції, низький поріг входження і прозоре ціноутворення. З негативних сторін – порівняно нижча стабільність сервісу.

### 2.3 Аналіз MBaaS сервісу AWS Amplify

Amazon Web Services (AWS) – постачає величезну кількість сервісів і цікавий тим, що у нього за аналогією з Microsoft Azure існує виділений набір функціональності під назвою AWS Amplify, який по суті і є мобільним бекендом. Раніше ви могли чути назву AWS Mobile Hub, який довгий час був основним

сервісом, що надає MBaaS функціональність. Як пишуть самі Amazon, Amplify це доопрацьований і вдосконалений Mobile Hub, в якому вирішені основні проблеми попередника. Якщо вірити Amazon, то сервісу Amplify довіряє безліч великих компаній, серед яких Netflix, Airbnb і багато інших [7].

Мобільне рішення від Amazon дозволяє в короткі терміни конфігурувати всю необхідну функціональність для мобільного застосування: серверну логіку, зберігання даних, авторизацію користувачів, обробку і постачання контенту, повідомлення та аналітику.

Amazon також надає всі необхідні умови з точки зору інфраструктури, такі як масштабування, балансування навантаження і багато іншого.

За аналітику відповідає окремий сервіс Amazon Pinpoint, в якому можна сегментувати аудиторію і проводити масштабні таргетингові кампанії через різні канали (push повідомлення, смс та електронну пошту) по залученню користувачів. Pinpoint надає дані в режимі реального часу. Можна створювати динамічні сегменти аудиторії, аналізувати їх і оптимізувати маркетингову стратегію на їх основі.

Amazon Amplify надає доступ до сервісу AWS Device Farm для тестування білдів додатку на реальних пристроях. Сервіс дозволяє проводити мануальне тестування, чи автоматичне паралельне на безлічі фізичних пристроїв [8]. Сервіс AWS Amplify Console є інструментом для деплоя і хостингу як серверних ресурсів, так і веб додатків з можливістю настройки CI / CD для автоматизації процесу розробки. Також незвично виглядає можливість впровадження в мобільні додатки «з коробки» голосових і текстових ботів в якості інтерфейсу для взаємодії з користувачем. Все це працює на сервісі Amazon Lex.

Цікаво, що AWS Amplify надає також і невелику бібліотеку готових UI компонентів для React Native, що може трохи прискорити процес розробки, чи використовуватися в прототипі або MVP проекту.

Сервіс Amazon Amplify надає SDK для iOS, Android, JavaScript і React Native і досить докладну документацію. Важливо відзначити що крім REST, сервіс підтримує ще й GraphQL.

Високий поріг входження є агальною проблемою для всіх IaaS. Amazon не виняток, бо на думку багатьох розробників – є одним з найскладніших сервісів для входження. Освоєння AWS з нуля займе чимало часу. Але якщо обмежитися тільки Amplify – можна реалізувати робоче рішення в адекватні терміни.

Сервіс від Amazon за статистикою виглядає менш стабільним, ніж Azure. Але повноцінних відключень – відносно мало (рис. 2.6). Переважна більшість проблем це попередження і нестабільність в роботі деяких сервісів[11].

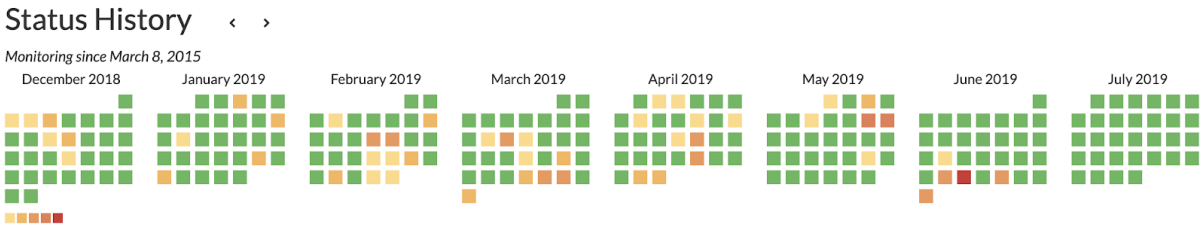



**Рисунок 2.6** – Технічні збої сервісу від Amazon

Цінова політика Amazon Web Services з першого погляду досить проста - платити треба тільки за ті сервіси і об'єм трафіку, що використовується понад безкоштовний ліміт. Але як і в випадку з Microsoft Azure, чим більше сервісів використовується, тим складніше прогнозувати підсумкову вартість роботи. В інтернеті безліч відгуків, які називають AWS занадто дорогим. Навіть існують компанії, які за окрему плату готові оптимізувати ваше використання AWS, мінімізуючи щомісячні рахунки наскільки це можливо.

У цілому ситуація з Amazon Amplify схожа на Azure. Багато в чому аналогічний функціонал для MBaaS, надання повноцінної інфраструктури та можливість розвитку свого бекенду. Позитивно виділяються маркетингові інструменти Amazon, зокрема, Pinpoint. З негативних сторін є високий поріг входу, і такі ж складнощі з прогнозуванням вартості. Додамо до цього менш стабільний сервіс і, судячи з відгуків, не дуже чуйну технічну підтримку.

# 3. РОЗРОБКА MVP ДОДАТКУ ТА ІНТЕГРАЦІЯ CLOUD FIRESTORE

## 3.1 Основна активність, реєстрування та авторизація користувача

Розробка Android-додатку майже завжди починається зі створення класу стартової активності. У випадку наявного додатку, початкова активність MainActivity не має графічного відображення у процесі роботи додатку, але має організаційні значення. Додаток розроблено за принципом Single Activity, що означає використання однієї материнської активності та багатьох фрагментів. Це надало додатку динамічності та гнучкості. Разом з запуском активності MainActivity, активуються постачання необхідних залежностей з Dagger2 та прослуховування натискань значків BottomNavigationView. Код головної активності додатку представлений у Додатку 1.

У процесі розробки створено клас додатку, через який можна буде підключити патерни, чи сторонні бібліотеки, що потребують запуску на рівні додатку і використовуються впродовж всієї роботи. Архітектурний паттерн Dagger2 був підключений саме у цьому класі.

```
class App : Application(), HasAndroidInjector {

  @Inject
  lateinit var dispatchingAndroidInjector: DispatchingAndroidInjector<Any>

  override fun onCreate() {
    super.onCreate()

    DaggerAppComponent.builder()
      .application(this)
      .build()
      .inject(this)
  }

  override fun androidInjector(): AndroidInjector<Any> =
dispatchingAndroidInjector
}
```

Під час розробки додатку було використано багато переваги мови програмування Kotlin. Серед них – розширення, що дозволяють розширювати існуючі класи(навіть ті, що надаються мовою розробки) новими методами. Це одна з багатьох переваг Kotlin над Java. Як один із прикладів, приведено класс String з розширеним методом isValidEmail(), що дозволить перевірити текстове значення на відповідність формату в якому має бути записана e-mail адреса.

```
fun String.isValidEmail() =
  this.isNotEmpty() &&
android.util.Patterns.EMAIL_ADDRESS.matcher(this).matches()
```

Клас View у нативній Android розробці є вершиною ієрархії для всіх інших класів візуальних компонентів, що можуть бути використані у створенні графічного інтерфейсу. Його також було розширено методом, що буде запобігати повторним натисканням на активні елементи інтерфейсу частіше ніж один раз у півтори секунди.

```
fun View.cooldownLarge() {
  //mostly apply for external calls on the server
  freezeView(this, 1500)
}

fun freezeView(view: View, interval: Long) {
  view.isClickable = false
  Handler().postDelayed({
     view.isClickable = true
  }, interval)
}
```

Графічний інтерфейс фрагментів Android-додатку створюється, переважно, за допомогою xml документів з розміткою наявних View елементів. У даному випадку, MainActivity має розмітку в якій розміщені BottomNavigationView панель та контейнер для фрагментів. Код xml документу розміщений у Додатку 2.

### 3.2 Створення та інтеграція консолі Google firebase

Враховуючи потреби наявного проекту та особливості розглянутих сервісів, було обрано Google Firebase у якості MBaaS сервісу. Ключові фактори, які спонукали до такого вибору – вбудована аналітика Fabric\Crashlitics, порівняно легка інтеграція сервісу у проект, докладна документація та велика інформаційна база допоміжного контенту. Також важливу роль зіграла зрозуміла та передбачувана цінова політика сервісу.

Google Firebase пропонує два різновиди баз даних – Cloud Firestore та Realtime Database. Розглянувши плюси, мінуси цих пропозицій, та потреби додатку, було обрано Cloud Firestore. Цей сервіс має більш структуроване зберігання даних, в той час як Realtime Database – є великим JSON деревом, що значно обмежує створення складних структур зберігання даних та формування розгалужених запитів. Сучасна робота мобільного додатку може вимагати завантаження певного документу без потреби тягнути всі підпорядковані йому дані, які можуть знаходитися вниз по дереву JSON. Таким чином Cloud Firestore є зручним, та фінансово більш вигідним рішенням. У разі правильної організації структури зберігання даних, завантаження непотрібних документів зводиться до мінімуму [13].

Під час інтеграції сервісу у додаток, було створено обліковий запис додатку в Google Firebase. Для повноцінної роботи бази даних та додатку, у консолі Google Firebase проведено відповідні налаштування, задані основні параметри, в тому числі SSH ключі для стану дебагу та релізу. Оскільки ці дані є конфіденційними, то графічні представлення реалізації обмежені (рис. 3.1).

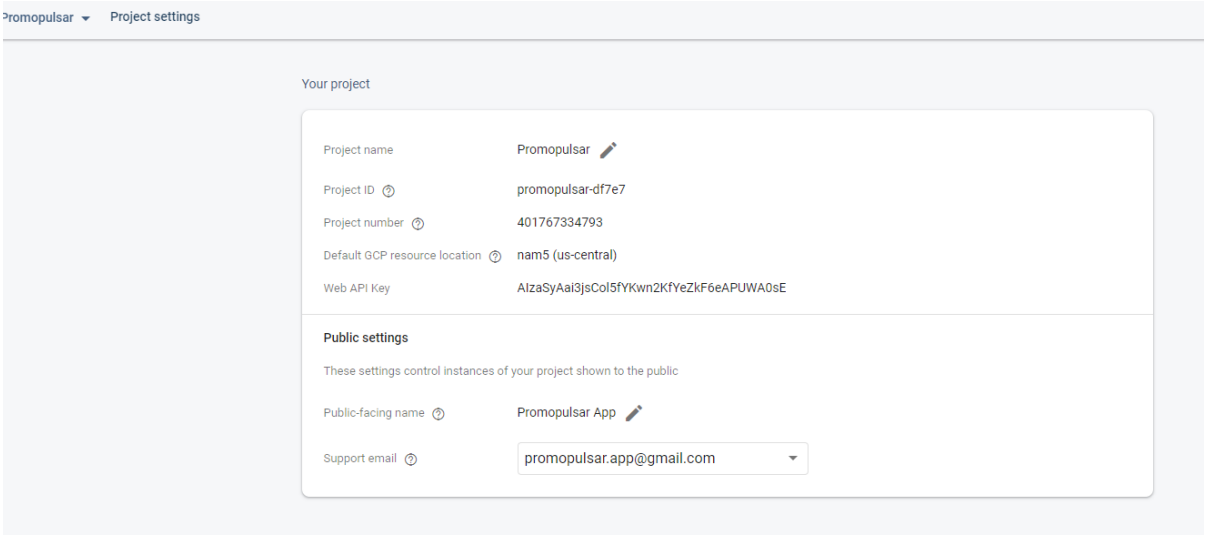



**Рисунок 3.1** – Створення облікового запису додатку в Google Firebase

Захист персональних користувачів, є одним з головних пріоритетів для кожного онлайн продукту. Перша вимога, що надходить до новоствореної бази даних у Cloud Firestore – налаштувати правила запису та зчитування даних для користувачів. Кожен смартфон, що має інстальований додаток Promopulsar, має доступ до бази даних, що його обслуговує. Задача розробника - визначити, які користувачі, які дані можуть читати чи редагувати. Авторизований користувач додатку Promopulsar повинен мати змогу додавати, змінювати, та зчитувати власні персональні дані, та створені ним дані пропозицій спільного просування бренду, що є публічними. Але дані розміщені іншими користувачами, йому можуть бути доступні, винятково, для зчитування в обмеженій кількості. Зі зрозумілих причин, мінювати персональні дані можуть тільки їх власники.

Firebase Firestore надає певні інструменти, у вигляді спеціального розділу у консолі та простої мови програмування, що допомагають налаштувати обмеження та дозволи на зміну чи зчитування даних. Ключові фактори, що перевіряються під час запитів – авторизація користувача, його id, та чи цільові дані були створені ним. Функціонал захисту даних Cloud Firebase значно ширший, але на даному етапі розвитку додатку цілком вистачить основних налаштувать. Нище наведений код реалізації правил читання та редагування однієї з гілок у базі даних. Ці конкретні правила дозволяють створювати,

зчитувати, редагувати та видаляти певні дані, тільки в тому випадку, якщо користувач зареєстрований та авторизований і є автором цих даних.

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /channels/{channel} {
     allow read: if request.auth.uid != null;
    }
    match /channels/{channel} {
     allow create: if request.auth.uid != null
                                        && request.resource.data.user ==
request.auth.uid;
      allow update: if request.auth.uid != null
                                        && request.resource.data.user ==
request.auth.uid;

      allow delete: if request.auth.uid != null
                                        && request.auth.uid ==
resource.data.user;
    }
  }
```

Розробка комунікації додатку з базою даних, в першу чергу, вимагає підключення відповідних залежностей і бібліотек. Серед них – “firebase-core”, яка є ядром майже всіх залежностей, що постачаються сервісом, “firebase-auth” для розробки аутентифікації користувача і збереження його даних на сервері, та firebase-ui-auth – допоміжний функціонал авторизації що надає готову реалізацію створення та входу до облікового запису.

Оскільки проект розробляється з урахуванням асинхронної роботи додатку, було підключено допоміжну бібліотеку “rxfirebase2” для зручної реалізації асинхронних запитів на сервер.

```
implementation "com.google.firebase:firebase-auth:$firebaseAuthVersion"
implementation "com.google.firebase:firebase-core:$firebaseCoreVersion"
implementation 'com.firebaseui:firebase-ui-auth:6.2.0'
implementation "com.androidhuman.rxfirebase2:firebase-
core:$humanFirebaseCoreVersion"
```

У Додатку 3 наведено приклад одного з запитів на мові програмування Kotlin, та з використанням програмування асинхронних запитів за допомогою rxfirebase2. rxfirebase2 – це, по своїй суті, RX обгортка над стандартними запитами, що Google Firebase надає для зв’язку з Cloud Firestore.

У наступному прикладі представлені запити що відповідають за збереження чи видалення даних. У їх створенні також втілені принципи асинхронних запитів за допомогою реактивного програмування та стандартних інструментів від Google Firebase.

```
override fun updateChannel(channel: ChannelDto): Single<String> =
firestoreRef.collection(ConstantsFirebaseFields.CHANNELS).document(channel.id)
        .rxSet(channel)
        .doOnComplete {
            channelsDbStorage.update(channel)
                .subscribeOn(schedulersFacade.io())
                .observeOn(schedulersFacade.main())
                .observe {}
        }
        .toSingle { channel.id }
```

### 3.3 Тестування та виправлення багів

На стадії тестування MVP додатку було виявлено типові помилки: некоректне розміщення та відображення графічних елементів інтерфейсу, помилки у збереженні даних до локальної бази даних, та зчитування даних з Cloud Firebase. В процесі рефакторингу коду, та пошуку помилок активно використовувались інструменти відловлювання багів та помилок, що надає середовище розробки Android Studio. Рефакторинг коду та виправлення помилок зробили додаток стабільним та відповідним до технічного завдання, створеного на етапі проектування. Користувач має можливість створити обліковий запис, редагувати власні дані та оглядати дані створені іншими користувачами.

Таким чином, розроблено продукт з мінімально робочим набором функціоналу, що дозволить користувачу знаходити бажаючих для спільного

просування каналу, бренд, сторінку, тощо. З результатом роботи можно ознайомитися на зображеннях: Додаток 4.1, Додаток 4.2, Додаток 4.3.

Додаток було опубліковано на сервісі Google Play під іменем Promopulsar, за посиланням:

https://play.google.com/store/apps/details?id=com.promopulsar.promopulsar

# ВИСНОВКИ

Створення Android-додатку, що відповідає сучасним стандартам розробки, та вибагливим запитам користувачів, вимагає використання багатьох інструментів. Додаток має бути зрозумілим у використанні, чуйним і ощадним до ресурсів пристрою. Також, важливе значення мають детальне проектування додатку на початковій стадії розробки та його тестування перед релізом. Це значно прискорює процес створення готового продукту, та покращує якість роботи з ним.

Окрема увага має бути приділена формуванню, інтеграції та підтримці бекенду мобільного додатку, що є важливою частиною створення готового продукту. Ця робота потребує значних витрат часу, коштів та інтелектуальних можливостей розробників. Провідні світові компанії, що працюють у сфері інформаційних технологій надають послуги MBaaS сервісів, котрі спрощують розробку та запуск інтернет продуктів. Цим активно користуються компанії з обмеженими бюджетом, інтелектуальними можливостями, приватні розробники чи невеликі команди.

Важливу роль у створенні бази даних грають вибір найбільш підходящого MBaaS сервісу, створення структури бази даних та інтегрування її у додаток, налаштування правил захисту даних користувачів, поточна підтримка та модернізація, тощо.

Головна ціль роботи була реалізована у повному обсязі. Мобільний додаток Promopulsar має хмарну базу даних для збереження та зчитування даних користувачів. Всередині додатку реалізований функціонал збереження, завантаження, та показу даних про користувача, його пропозицій спільного просування брендів. Таким чином користувачі можуть бачити контент створюваний іншими користувачами. Дані захищено стандартними налаштуваннями безпеки, що надаються Cloud Firestore від Google Firebase.

# СПИСОК ЛІТЕРАТУРИ


1. Cross-promotion / https://en.wikipedia.org/wiki/Cross-promotion
2. Жемеров Д. Kotlin в Действии / Жемеров Д., Исакова С. // ДМК Пресс - 2017. - 402 с.
3. Android Developer Guides / https://developer.android.com/guide
4. Kotlin technical documentation / https://kotlinlang.org/docs
5. Dagger2 technical documentation / https://dagger.dev/dev-guide/
6. Документация мобильных приложений Azure / https://docs.microsoft.com/ru-ru/previous-versions/azure/app-service-mobile/
7. Getting started wint AWS Amplify / https://aws.amazon.com/ru/blogs/mobile/aws-mobile-gets-amplified/
8. AWS Device Farm / https://aws.amazon.com/ru/device-farm/?nc=sn&loc=2&dn=3
9. Google Firebase technical documentation / https://firebase.google.com/docs
10. Google Firebase pricing / https://firebase.google.com/pricing
11. Instant cloud status monitoring / https://statusgator.com/
12. Capterra helps you find the right software for your business / www.capterra.com
13. Todd Kerpelman / Cloud Firestore vs the Realtime Database: Which one do I use? / https://firebase.googleblog.com/2017/10/cloud-firestore-for-rtdb-developers.html

**ДОДАТОК 1**

```
class MainActivity : AppCompatActivity(), HasAndroidInjector {
  @Inject
  lateinit var dispatchingAndroidInjector: DispatchingAndroidInjector<Any>
  @Inject
  lateinit var onNavigationReselectedListener:
BottomNavigationView.OnNavigationItemReselectedListener
  private lateinit var bottomNavigationView: BottomNavigationView
  private lateinit var navController: NavController

  override fun onCreate(savedInstanceState: Bundle?) {
     AndroidInjection.inject(this)
     super.onCreate(savedInstanceState)
     setContentView(R.layout.activity_main)

     bottomNavigationView = findViewById(R.id.bottom_navigation_view)
     navController = Navigation.findNavController(this, R.id.nav_host_fragment)
     initBottomNavigationView()
  }

  private fun initBottomNavigationView() {
     NavigationUI.setupWithNavController(bottomNavigationView, navController)
bottomNavigationView.setOnNavigationItemReselectedListener(onNavigationResele
ctedListener)
  }

  override fun androidInjector(): AndroidInjector<Any> =
dispatchingAndroidInjector
}
```

**ДОДАТОК 2**

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".presentation.MainActivity">

  <com.google.android.material.bottomnavigation.BottomNavigationView
    android:id="@+id/bottom_navigation_view"
    android:layout_width="match_parent"
    android:layout_height="@dimen/bnb_height"
    android:background="?attr/colorPrimary"
    android:enabled="true"
    app:elevation="@dimen/margin_micro"
    app:itemIconSize="@dimen/bnb_icon_size"
    app:itemIconTint="@drawable/selector_menu_icons_color"
    app:layout_constraintBottom_toBottomOf="parent"
    app:menu="@menu/menu_bottom_navigation" />

  <fragment
    android:id="@+id/nav_host_fragment"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    app:defaultNavHost="true"
    app:layout_constrainedHeight="true"
    app:layout_constrainedWidth="true"
```

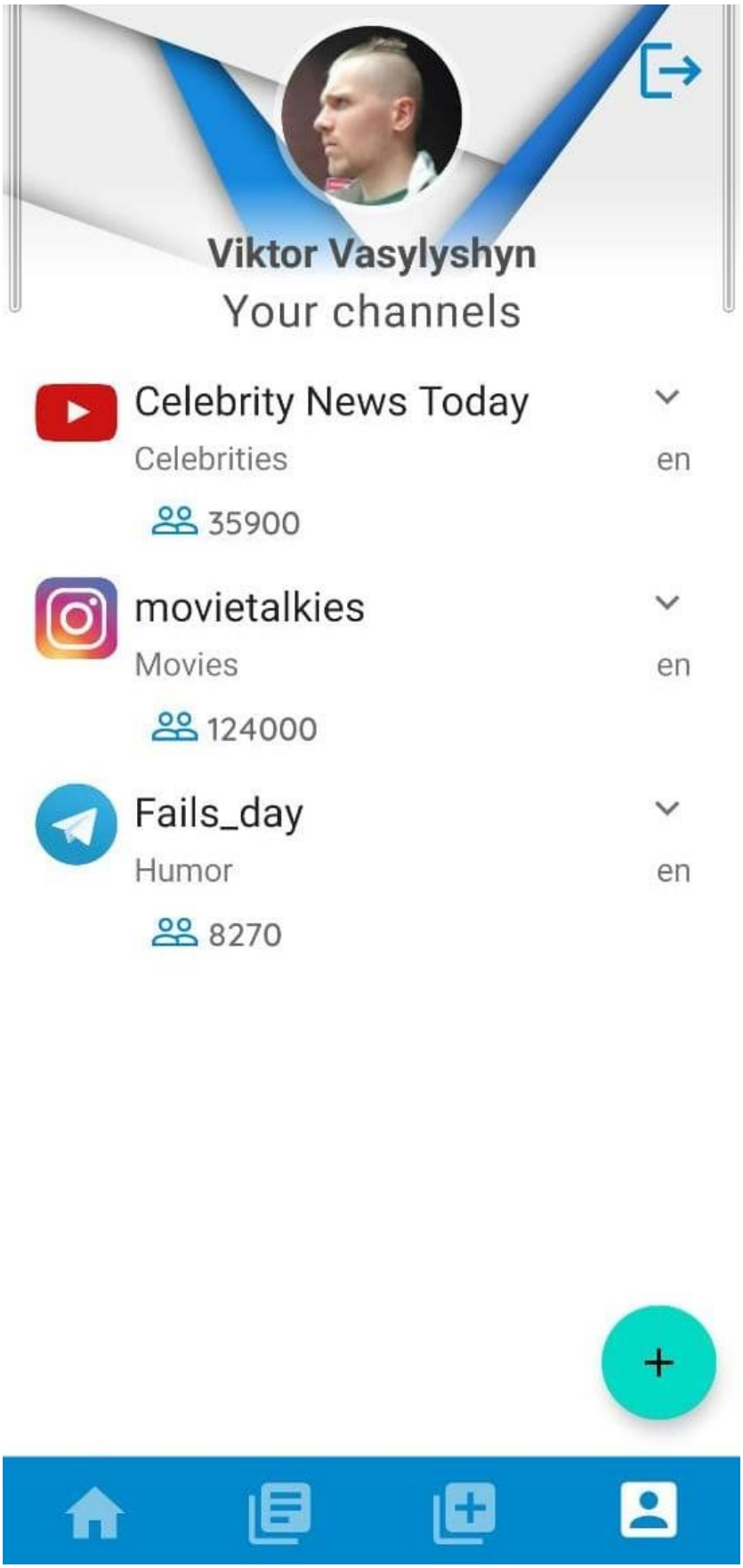
```
        app:layout_constraintBottom_toTopOf="@id/bottom_navigation_view"
        app:layout_constraintTop_toTopOf="parent"
        app:navGraph="@navigation/nav_graph" />

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

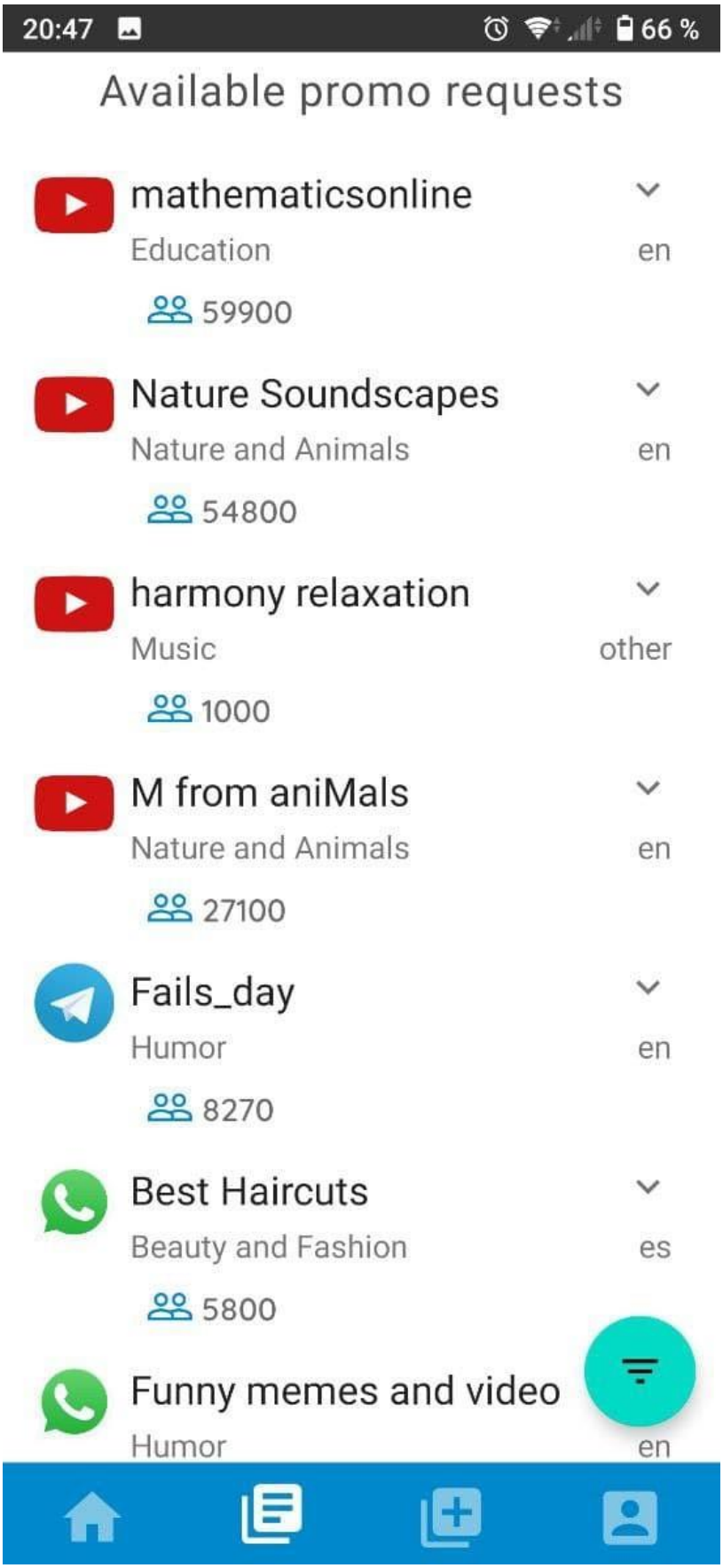
**ДОДАТОК 3**

```
override fun fetchCurrentUserData(): Single<CurrentUserDto> {another one signed
    currentUserId = FirebaseAuth.getInstance().uid
    return if (currentUserId != null) {
        firestoreRef.collection(ConstantsFirebaseFields.USERS)
            .document(currentUserId as String)
            .data()
            .map { docSnapValue ->
                var currentUserDto = CurrentUserDto("", "", "", null)
                try {if (docSnapValue.value().exists()) {
docSnapValue.value().toObject(CurrentUserDto::class.java)?.let {
                        currentUserDto = it
                    }}
                } catch (e: Exception) {
                    FirebaseCrashlytics.getInstance().recordException(e)
                }
                FirebaseAuth.getInstance().currentUser?.let {
                    currentUserDto = currentUserDto.copy(
                        id = it.uid,
                        email = it.email,
                        name = it.displayName,
                        imageUrl = it.photoUrl
                    )}
                currentUserDto
            }
    } else {
        throw IllegalArgumentException("User id == null")
    }}
```

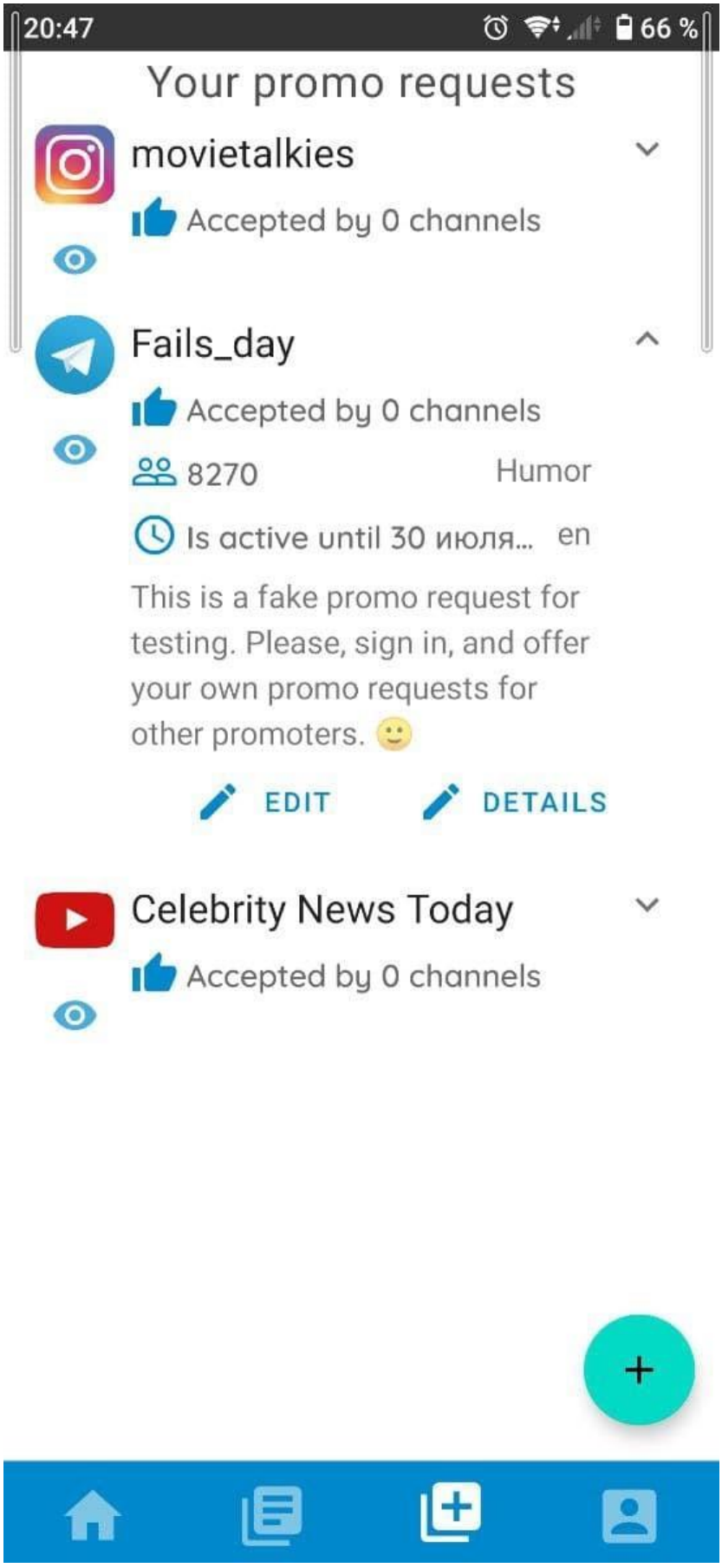
**ДОДАТОК 4.1**


Viktor Vasylyshyn
Your channels
Celebrity News Today
Celebrities
en
35900
movietalkies
Movies
en
124000
Fails_day
Humor
en
8270

**ДОДАТОК 4.2**


20:47
66 %
Available promo requests
mathematicsonline
Education
en
59900
Nature Soundscapes
Nature and Animals
en
54800
harmony relaxation
Music
other
1000
M from aniMals
Nature and Animals
en
27100
Fails_day
Humor
en
8270
Best Haircuts
Beauty and Fashion
es
5800
Funny memes and video
Humor
en

**ДОДАТОК 4.3**


20:47
66 %
Your promo requests
movietalkies
Accepted by 0 channels
Fails_day
Accepted by 0 channels
8270
Humor
Is active until 30 июля...
en
This is a fake promo request for testing. Please, sign in, and offer your own promo requests for other promoters.
EDIT
DETAILS
Celebrity News Today
Accepted by 0 channels