МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ
(підпис)

_____ червня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук,

освітньо-професійної програми «Інформатика»

на тему: **«Інформаційне та програмне забезпечення навчального мобільного додатку «Енциклопедія трейдера»»**

здобувача групи ІН -91 Сумцова Данила Едуардовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Данило СУМЦОВ
(підпис)

Керівник, доцент,
кандидат фізико-математичних наук Сергій ШАПОВАЛОВ _____
(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-91 Сумцова Данила Едуардовича

1. Тема роботи: «Інформаційне та програмне забезпечення навчального мобільного додатку «Енциклопедія трейдера»»
затверджую наказом по СумДУ від *«01» червня 2023 р. № 0475-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 09 червня 2023 року*
3. Вхідні дані до кваліфікаційної роботи
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз предметної області, постановка й формування завдання
2) Огляд методики розробки та інструментів 3) Програмна реалізація продукту, інформаційне забезпечення і тестування 4) Аналіз результатів розробки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ (підпис) Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	Аналіз предметної області, постановка й формування завдання		
2	Огляд методики розробки та інструментів		
3	Програмна реалізація продукту, інформаційне забезпечення і тестування		
4	Аналіз результатів розробки		
5	Оформлення пояснювальної записки до кваліфікаційної роботи		

Здобувач вищої освіти _____ (підпис) Керівник _____ (підпис)

АНОТАЦІЯ

Записка: 72 стор., 37 рис., 1 табл., 1 додаток, 11 використаних джерел.

Обґрунтування актуальності теми роботи – У сучасному житті все більше людей використовують мобільні додатки для вирішення безлічі власних питань. Використовують їх для спілкування, отримання послуг, пошуку інформації, а також для навчання. Одними з найпопулярніших категорій мобільних додатків є саме фінансові та освітні платформи. Враховуючи тенденцію зростання їх популярності, а також такого роду діяльності, як трейдинг, актуальним на даний момент є створення саме навчального мобільного додатку тематики трейдингу.

Об’єкт дослідження — мобільний навчальний додаток

Мета роботи — інформаційна забезпечення і програмна реалізація мобільного навчального додатку

Методи дослідження — технології розробки мобільних додатків.

Результати — розроблено мобільний навчальний додаток для пристроїв під управлінням ОС Android, який має необхідний набір функцій для зручного використання. В процесі розробки використовувалась мова програмування Kotlin, мова розмітки XML, середовище розробки Android Studio, програма для створення дизайну Figma, а також набір хмарних рішень платформи Firebase.

МОБІЛЬНИЙ ДОДАТОК, НАВЧАЛЬНА ПЛАТФОРМА, KOTLIN, XML, ANDROID STUDIO, FIREBASE

ЗМІСТ

ВСТУП

Актуальність: У сучасному світі мобільні пристрої стали невід'ємною частиною нашого життя. Все більше людей делегують свої побутові та робочі справи на мобільні додатки на своїх смартфонах. Використовують їх для спілкування, отримання послуг, пошуку інформації, а також для *навчання*. Однією з найпопулярніших категорій мобільних додатків є саме освітні платформи [1]. Враховуючи тенденцію зростання їх популярності, було вирішено обрати темою роботи саме *розробку навчального мобільного додатку*.

Все більше молодих людей останнім часом починають захоплюватись темою трейдингу. Пробують різні підходи у цій сфері, в якості джерел тимчасового прибутку або ж взагалі, маючи певний успіх, поєднують своє життя з цим родом діяльності[2]. Тому однією з актуальних *тематик* для навчальної платформи було обрано саме *трейдинг*.

Методом пошуку та огляду необхідної інформації, було вирішено питання вибору типу розробляємого додатку*(кроссплатформа чи нативний)*, а саме нативна розробка під одну з двох домінуючих мобільних ОС*(IOS та Android)*. Оскільки за показниками багатьох відомих моніторингових ресурсів, кількість пристроїв під управлінням операційної системи Android переважає у декілька разів, було обрано саме її.

Таким чином було сформовано **мету кваліфікаційної роботи**, а саме: *розробка навчального мобільного додатку "Енциклопедія трейдера" під ОС Android*

Предмет дослідження - мобільні додатки під ОС Android

Об’єкт дослідження - функціональність додатків під ОС Android

Методи дослідження - аналіз літератури за темою роботи і практична реалізація додатку

Структура роботи:

- Інформаційний огляд за темою роботи
- Вибір методів та інструментів для реалізації задачі
- Проектування архітектури та функціоналу додатку
- Розробка програмного забезпечення
- Інформаційне забезпечення додатку
- Тестування готового продукту

1. ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1. Актуальності теми роботи

Мобільні додатки пропонують безліч різноманітних послуг, які спрощують наше повсякденне життя та розширюють можливості нашого смартфону. Від ігор та розваг до освіти та бізнесу. За даними за 2022-й рік, до семи найпопулярніших категорій мобільних додатків входять фінансові(третє місце) та освітні(п’яте місце) [1]. Через глобальну пандемію COVID-19 у 2020-му році відсоток зростання популярності мобільних навчальних додатків сягнув 148% за місяц. Багато людей тоді відкрили для себе освітні платформи і не розлучаються з ними до сьогоднішнього часу, адже це дуже зручний спосіб для навчання. В свою чергу фінансові платформи показують щорічне зростання популярності в 115%. Тому було прийнято рішення поєднати ці дві популярні категорії. Для багатьох людей, які цікавляться фінансовими ринками та хочуть розширити свої знання в цій галузі, може бути актуальною і корисною наявність саме таких мобільних-довідників як *"Енциклопедія трейдера"*, де користувачі можуть швидко та безперешкодно отримати доступ до інформації, що стосується їхньої сфери зацікавленості. Основна актуальність такого додатка полягає в таких факторах, як:

- **Зростаючий інтерес до трейдингу**. Останнім часом спостерігається збільшений інтерес до фінансових ринків та трейдингу. Багато людей бажають зрозуміти, як працюють валютні ринки, як аналізувати ціни акцій та інші фінансові інструменти.
- **Зручний доступ до навчання**. Мобільні додатки надають зручний доступ до інформації та навчальних матеріалів в будь-який час і в будь-якому місці.

- **Оновлення та актуальність**. У світі трейдингу швидко змінюються тренди, стратегії та умови. Важливо, щоб додаток надавав своєчасну інформацію та враховував актуальні тенденції. Це можна забезпечити за допомогою регулярних оновлень і постійної актуалізації інформації.
- **Зручний інтерфейс та персоналізація**. У додатку має бути простий і зрозумілий інтерфейс, який дозволяє користувачу легко знайти необхідну інформацію.

Підсумовуючи, можна зробити висновок, що розробка такого додатку є цілком затребуваною та зможе залучити необхідний прошарок користувачів.

1.2. Порівняння аналогів на ринку

Одним з основних кроків перед розробкою будь-якого продукту є аналіз ринку та порівняння аналогів. Для цього було обрано декілька найпопулярніших представників навчальних додатків у сфері трейдингу(рис. 1.1), та складено характеристику їх функціональності(табл. 1.1).

Основні функції / Додаток	**Investmate** від Capital.com	**How To Invest: Trader's Course** від INVESTING.com	**Шлях трейдера - Інвестиції** від Trade Republic
Уроки, навчальні матеріали	+	+	+
Інструменти для аналізу ринку	+	+	+

Особистий тренер для надання порад	-	-	+
Навігація через зручний та простий інтерфейс	+	+	+

Таблиця 1.1 Порівняння функцій аналогів

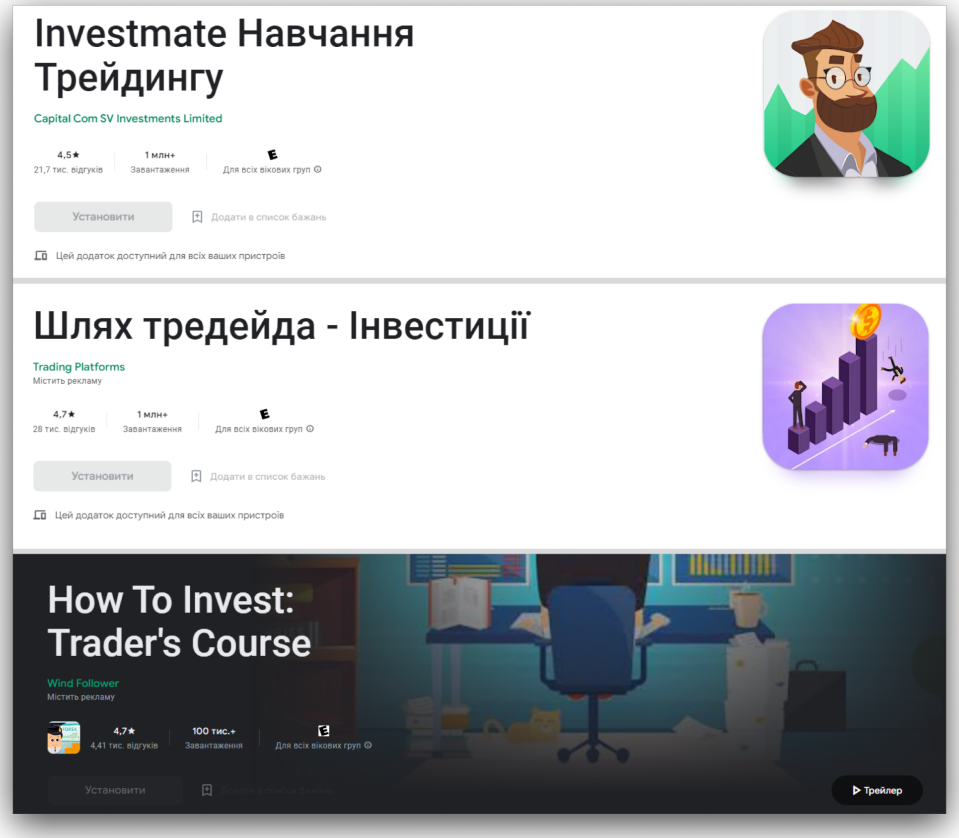


Рисунок 1.1 - Додатки-аналоги, що розглядаються

Провівши порівняння додатків-аналогів, основними функціями можна виділити: навчальні матеріали, інструменти для аналізу ринку і наявність зручної та швидкої навігації завдяки простому інтерфейсу. Оскільки

реалізація інструментів аналізу ринку зазвичай потребує інтеграції в додаток різних типів графіків та розробки алгоритмів їх симуляції, було прийнято рішення за основу мінімально життєздатного продукту(MVP) [6] взяти наявність навчальних матеріалів та швидкої, зручної навігації. Функціональність відстеження ринку залишити в статусі майбутнього оновлення.

1.3. Типи мобільних додатків

Зазвичай, на етапі проектування мобільних додатків розглядають два основних їх типи - нативні та кросплатформенні(гібридні) [3].

- Нативні додатки розробляються для конкретної платформи, найрозповсюдженішими з яких є iOS та Android. При розробці використовуються мови програмування та фреймворки, що специфічні для кожної платформи. Розробники зазвичай використовують такі мови програмування, як Swift або Objective-C для iOS, і Java або Kotlin для Android. Основною перевагою нативних додатків є повний доступ до функціональних можливостей платформи і найкраща продуктивність. Однак, розробка таких додатків відразу для декількох платформ вимагає окремого коду для кожної з них.
- Кросплатформенні додатки розробляються з використанням універсальних технологій та фреймворків, які надають можливість виконувати один і той же код на різних платформах. Це дозволяє розробникам використовувати одну базу програмного коду для створення додатків, які працюють як на iOS, так і на Android. Найпопулярнішими гібридними фреймворками є React Native, Xamarin, Flutter та Ionic. Кросплатформенні додатки можуть забезпечити більш швидку

розробку, через потенційно меншу кількість коду, але можуть мати деякі обмеження у доступі до специфічних функцій платформи або в продуктивності.

З урахуванням переваг та недоліків кожного типу, найпривабливішим варіантом є саме розробка нативного мобільного додатку. Це має розширити спектр доступним функцій і оптимізувати їх виконання.

1.4. Мобільні операційні системи

Двома найпоширенішими операційними системами на мобільних пристроях у сучасному світі є ОС iOS та Android.

- iOS є операційною системою, розробленою компанією Apple. Ця система працює виключно на продуктах компанії(iPhone, iPad та iPod Touch). Вона відома своєю безпекою та оптимізацією під пристрої компанії Apple.
- Android є операційною системою, розробленою компанією Google. Вона використовується на різноманітних мобільних пристроях, включаючи смартфони та планшети від різних виробників, таких як Samsung, Huawei, Xiaomi, LG та інші.

Ця ОС є більш відкритою та дозволяє розробникам створювати індивідуалізовані додатки.

За даними на 2022 рік, Android має найбільшу частку на ринку операційних систем у світі. Це близько 75% ринку мобільних пристроїв. В свою чергу iOS займає приблизно 25% [4]. Більша розповсюдженість Android пов'язана з різноманітністю пристроїв, доступних за різними цінними діапазонами. Це означає, що для охоплення більшої бази потенційних користувачів, доцільним рішенням є розробка додатку саме під цю систему.

1.5. Мета та задачі розробляємого додатку

Метою кваліфікаційної роботи є розробка навчального мобільного додатку під ОС Android з інформаційним контентом за тематикою трейдингу. Спроектувати зручний і сучасний UI/UX дизайн інтерфейсу. Програмно реалізувати функціональність, використовуючи мови Kotlin та XML. Інформаційно забезпечити додаток контентом на українській мові. Готовий продукт має включати в себе наступні способи використання:

- можливість перегляду стратегій та порад для трейдингу
- можливість перегляду словника трейдера
- можливість пошуку кожного типу інформації за назвою та описом
- доступ без реєстрації
- можливість збереження окремих пунктів до “Обраних”

Функціональними вимогами додатку є: підтримка пристроїв версії ОС Android 6.0 та вище, наявність актуальної інформації українською, можливість оновлення та актуалізації контенту додатку. **Нефункціональні вимоги**: простота і зрозумілість інтерфейсу та навігації додатку, наявність темного оформлення для зручного використання у вечірній час. Для реалізації продукту у найякіснішому стані, необхідно грамотно обрати відповідні інструменти та технології розробки Android додатків.

Поставлена мета потребує виконання наступних задач:

- аналізу предметної області розробляємого додатку
- огляд актуальних технологій та інструментів
- проектування інтерфейсу
- проектування моделі зберігання інформаційного контенту
- розширене тестування останньої версії продукту

2. МЕТОДИКА ВИРІШЕННЯ ПОСТАВЛЕНИХ ЗАДАЧ

2.1. Інструменти проектування дизайну інтерфейсу

Розробка дизайну інтерфейсів є важливою частиною процесу створення мобільних додатків. Інструментарій включає в себе багато цільових програм. Найпоширенішими з них є Adobe Photoshop, Adobe XD, Figma, Sketch.

- Photoshop є одним з найпоширеніших програмних засобів для роботи з графікою. Він надає широкі можливості для створення дизайну інтерфейсу, включаючи редакцію зображень, створення графічних елементів та компонентів
- Sketch є високопродуктивним інструментом розробки дизайну інтерфейсу, спеціально розробленим для роботи з веб-дизайном та мобільними додатками для macOS.
- Figma: Figma є онлайн-інструментом для дизайну та спільної роботи над проектами. Він дозволяє дизайнерам створювати векторні графіки, розробляти інтерактивні прототипи, а також спілкуватися та співпрацювати з командою в режимі реального часу.
- Adobe XD: Adobe XD є інструментом, спеціально розробленим для дизайну інтерфейсу та прототипування. Він надає можливості для створення дизайну, інтерактивних прототипів, анімацій та спільної роботи з командою.

Найоптимальнішим засобом для вирішення поставленої задачі є онлайн-інструмент Figma. Оскільки навіть базова(безкоштовна) його версія надає усе необхідне для розробки прототипу якісного, красивого та зручного інтерфейсу. Також його “хмарність” дозволяє працювати над проектом з різних пристроїв навіть у браузері.

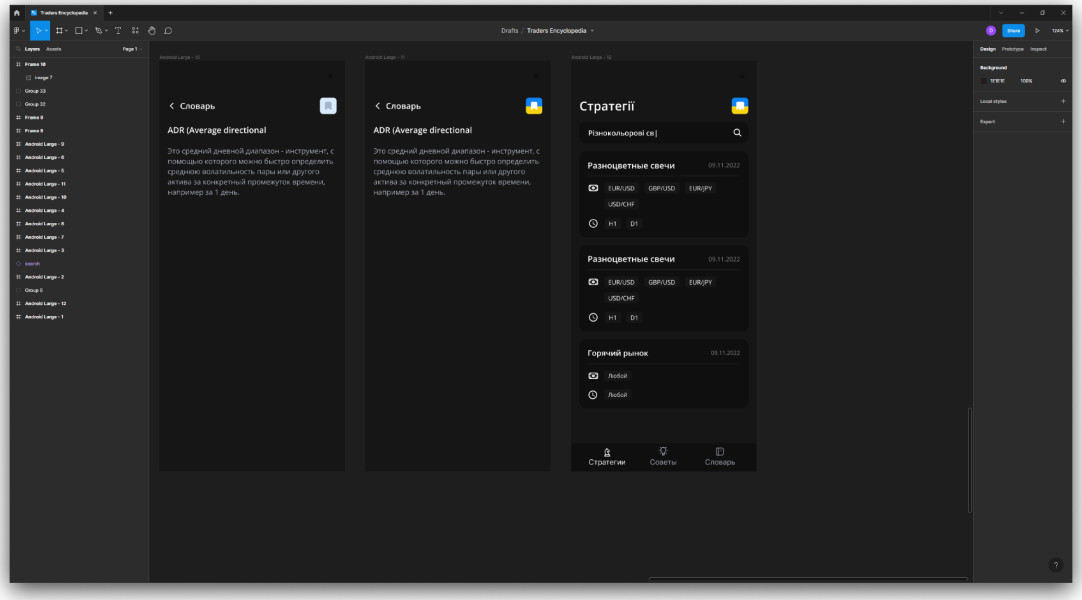


Рисунок 2.1 - Інтрефейс середовища Figma

2.2. Інструменти розробки Android додатків

Наразі для розробки нативних Android додатків актуальним середовищем є Android Studio. Це інтегроване середовище розробки (IDE) для створення мобільних додатків під платформу Android, розроблена компанією Google [5]. Воно надає розробникам широкий набір інструментів, які спрощують процес розробки, налагодження та тестування додатків.

- Android Studio надає потужний редактор коду, який підтримує різні мови програмування та формати. Він має функції автодоповнення коду, перевірку синтаксису, вбудований дебагер та багато інших інструментів, що полегшують розробку додатків.

- Середовище має інтуїтивно зрозумілий макетний редактор, який дозволяє створювати користувацький інтерфейс додатка. Розробники можуть переглядати макети в режимі дизайну та попереднього перегляду для візуалізації вигляду додатка.

- Android Studio має набір інструментів, які допомагають розробникам управляти проектами, збудованими на платформі Android. Це включає систему збирання та управління залежностями Gradle, інструменти для керування версіями, аналізу коду, створення та керування емуляторами та фізичними пристроями для тестування

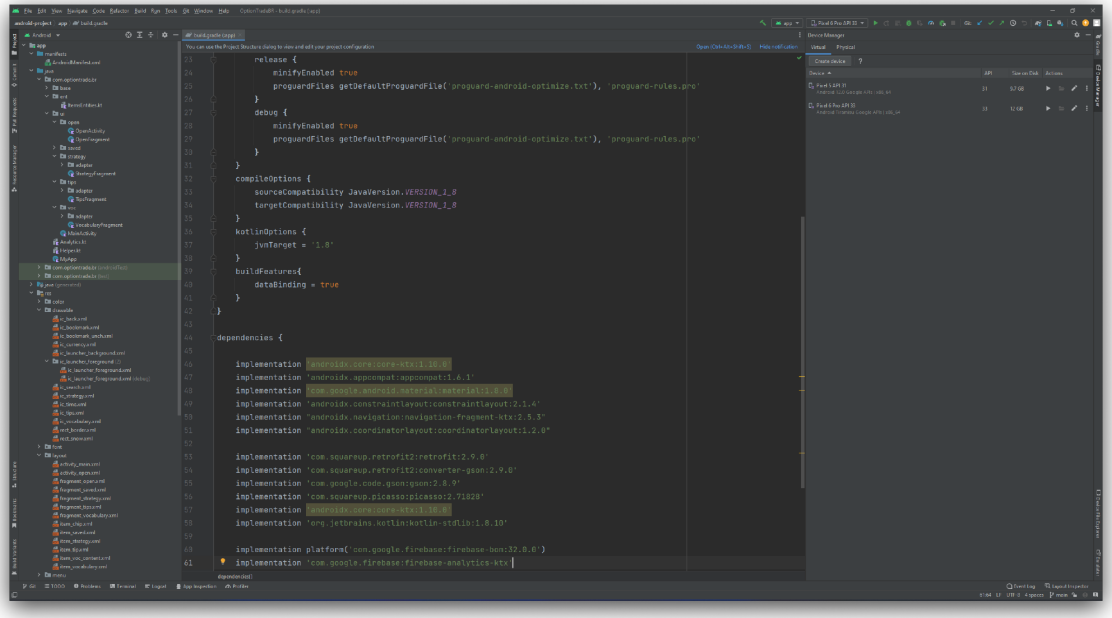


Рисунок 2.2 - Інтрефейс середовища Android Studio

2.3. Інструменти інтеграції хмарних послуг

Хмарні сервіси надають розробникам додатків доступ до потужних інфраструктурних ресурсів та інструментів, без необхідності створювати та управляти власними серверами. Серед популярних сервісів хмарних послуг можна перерахувати: Amazon Web Services, Microsoft Azure, Google Cloud Platform, Firebase.

- Amazon Web Services є одним з найбільших та найбільш розповсюджених сервісів хмарних послуг. Він надає широкий набір послуг, включаючи обчислення, зберігання даних, аналітику та багато інших
- Azure є сервісом хмарних послуг, розробленим компанією

Microsoft. Він надає гнучкі рішення для розробки та розгортання додатків. Пропонує такі послуги, як обчислення, доступ до баз даних, можливості штучного інтелекту, аналітика та інші.

- Google Cloud Platform є сервісом хмарних послуг від компанії Google. Він дозволяє розробникам отримувати доступ до інфраструктури Google, включаючи обчислення, зберігання в базах даних, інтеграцію з технологіями машинного навчання, аналітику та інші послуги.
- Firebase є платформою розробки мобільних та веб-додатків, яка належить компанії Google. Вона надає наступні послуги: аутентифікація користувачів, бази даних в режимі реального часу, зберігання даних, відправка повідомлень та аналітика. Firebase дозволяє розробникам швидко створювати та впроваджувати функціональні додатки з мінімальними зусиллями.

Серед усього списку, найоптимальнішим варіантом для розробки Android додатку є платформа Firebase, адже вона належить компанії Google і має широкий спектр наданих послуг, навіть з базовим(безкоштовним) планом.

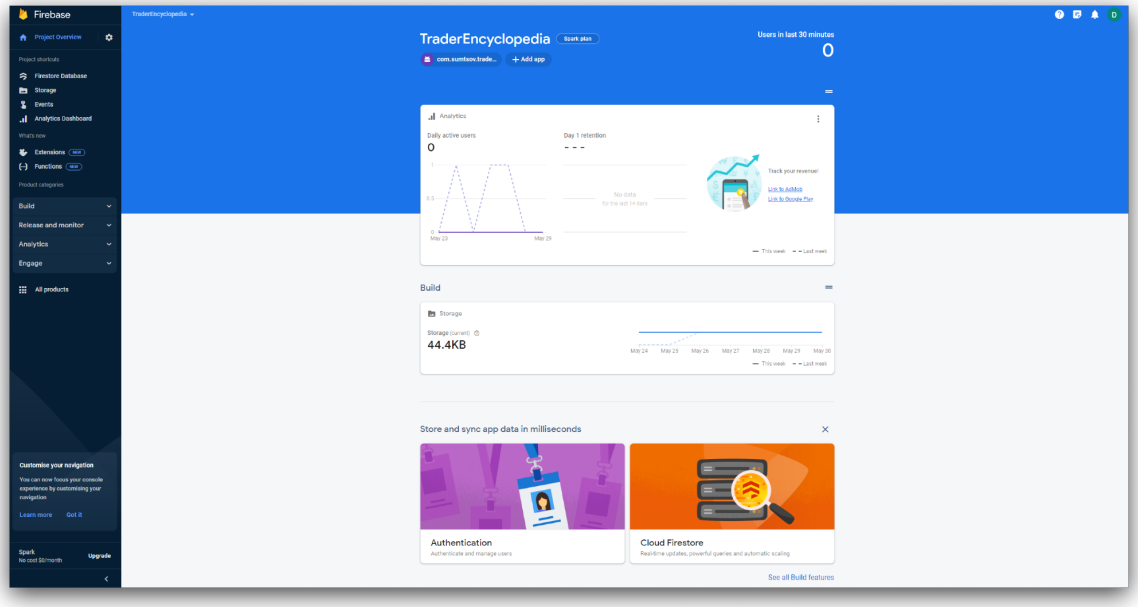


Рисунок 2.3 - Інтрефейс дашборду хмарної платформи Firebase

2.4. Проектування структури додатку

Для початку роботи над проектом, необхідно мати чітке уявлення про те, як додаток має функціонувати. Було створено візуалізацію бачення структури та послідовності дій користувача. На початковому етапі запуску додатку, має відбуватися завантаження даних з хмарного та локального сховищ, після цього користувач має отримати доступ до списків відповідних категорій інформаційних довідок(стратегій, порад і термінів зі словника), а також до розділу “збережених”. Для більш зручного і швидкого знаходження необхідної інформації, користувач матиме можливість пошуку за назвою та описом відповідних позицій у списку. Після вибору цікавлячої статті або терміну, він потраплятиме на екран огляду інформації, де також буде мати можливість додати її до “обраних”. Збережений контент відразу зберігається до локального сховища Android пристрою.

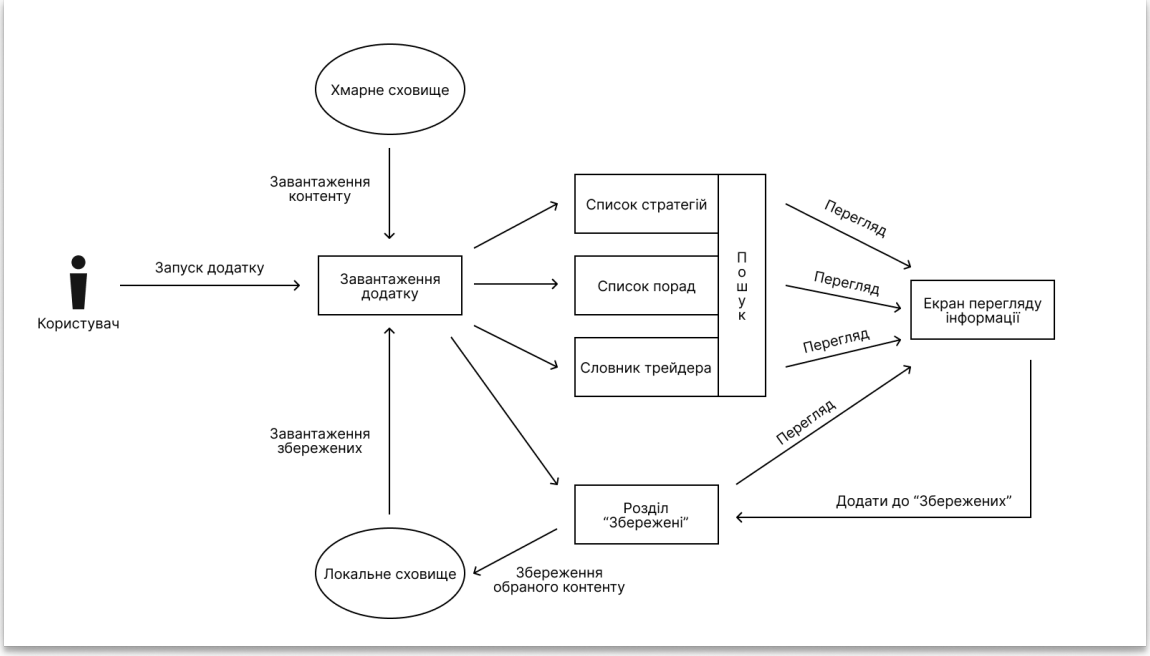


Рисунок 2.4 - Схема-візуалізація структури мобільного додатку

2.5. Проектування моделей даних

Для подальшого процесу необхідно спроектувати моделі даних. Для цього, спочатку, конкретизується перелік необхідної інформації, що міститиметься у кожному типі наявного контенту. Візуалізація представлена на рисунку 2.5. Після цього було спроектовано моделі даних, у вигляді Kotlin Data-класів, які повністю відповідають кожному з типів інформаційних блоків. Зображено на рисунку 2.6.

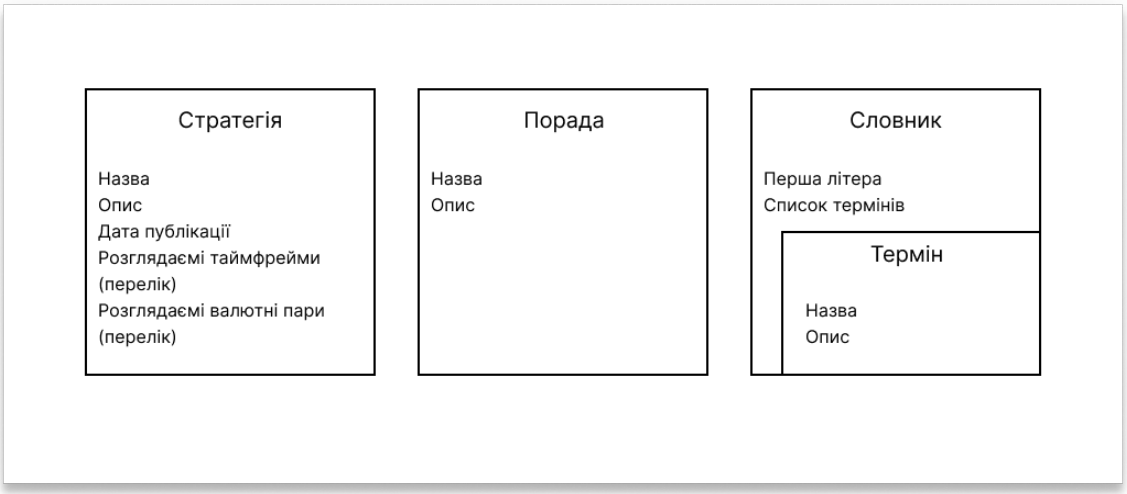


Рисунок 2.5 - Схема моделей інформаційного контенту

```
data class Strategy(
    val id: Int,
    val title: String,
    val date: Long,
    val text: String,
    val cur: ArrayList<String>,
    val tf: ArrayList<String>
){
```

```
data class Vocabulary(
    val letter: String,
    val content: ArrayList<VocContent>
)

data class VocContent(
    val id: Int,
    val title: String,
    val desc: String
){
```

```
data class Tip(
    val id: Int,
    val title: String,
    val text: String
){
```

Рисунок 2.6 - Схема моделей інформаційного контенту

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ

3.1. Розробка дизайну мобільного додатку

Розробка дизайну відбулася в раніше обраній програмі Figma. З урахуванням необхідних зручності в навігації та сучасності зовнішнього вигляду, в першу чергу відбувалося створення макетів екранів світлої теми додатку, як базової. На рисунку 3.1 зображені три основі екрани: стратегії, поради, словник. На них розміщено тулбар з назвою категорії та кнопкою переходу до розділу “Збережені”, текстове поле пошуку, список інформаційних позицій у вигляді карток, а також блок нижньої навігації з трьома розділами, що відповідає гайдлайнам UI/UX дизайну від Google.

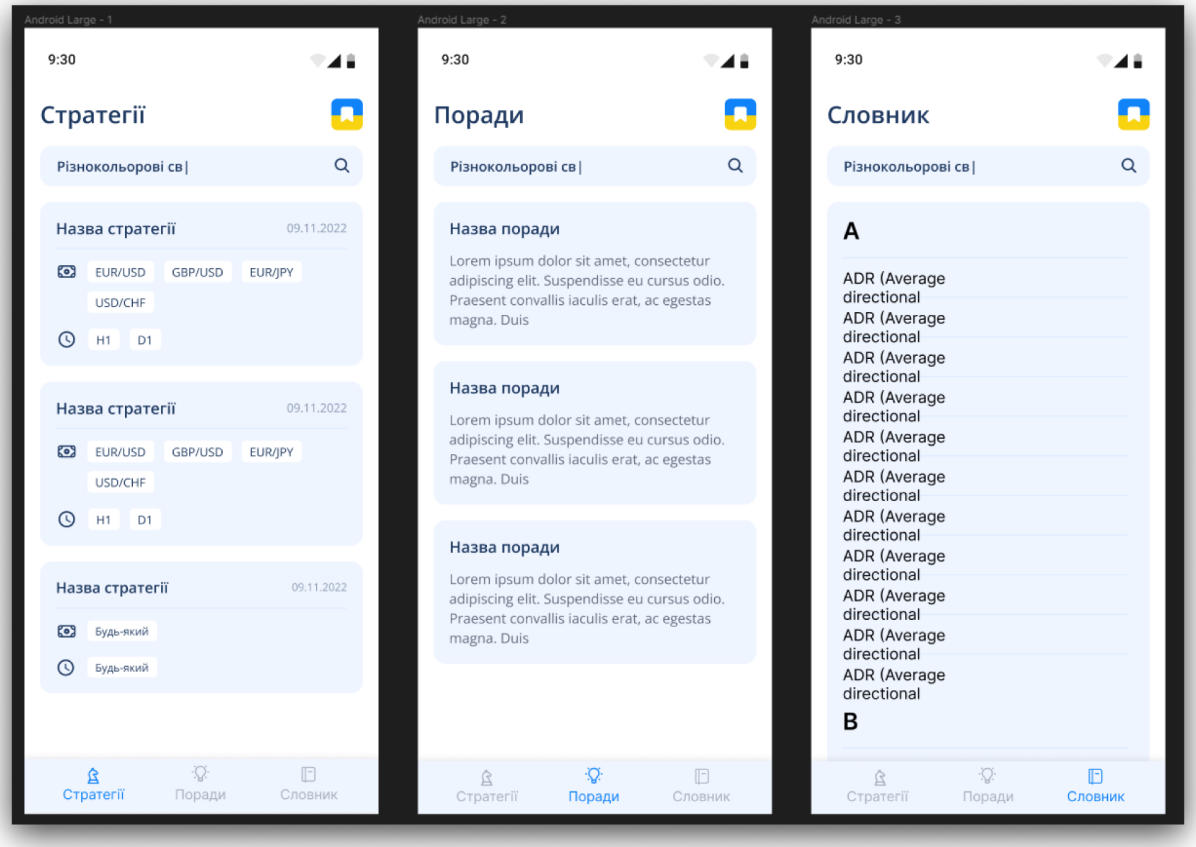


Рисунок 3.1 - Основні екрани додатку у світлому оформленні

Далі було розроблено два макети екрану “Збережені”, у різному стані(з

пустим і не пустим списком). На них розміщено заголовок розділу, інтерактивний перемикач вкладинок та список збереженого контенту.

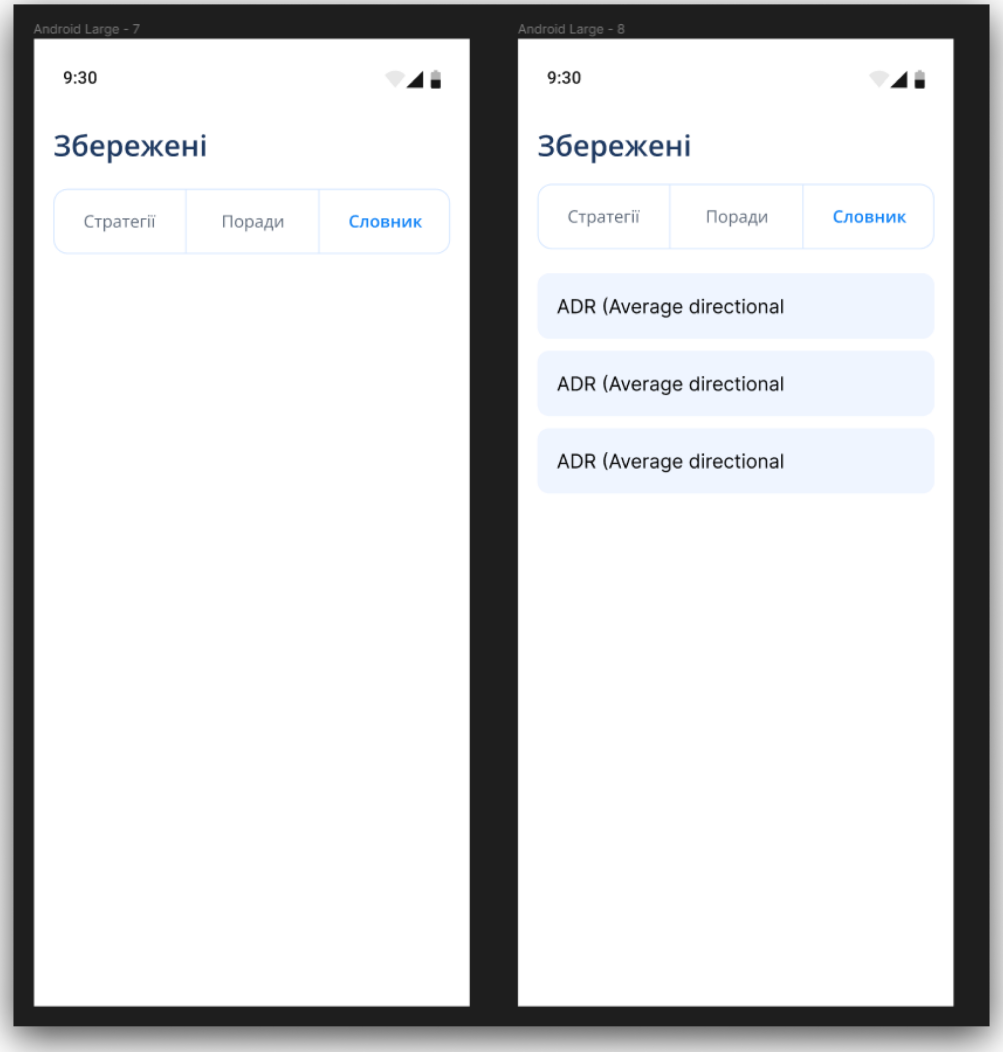


Рисунок 3.2 - Екран розділу “Збережені” у різному стані

Останнім етапом розробки інтерфейсу у світлому оформленні було створення макетів екрану перегляду інформації для кожної з трьох категорій. Для порад та термінів на екрані розміщено: назву категорії, кнопку збереження, заголовок та опис. У випадку зі стратегіями, також додано блок з переліком таймфреймів та валютних пар, що стосуються обраної стратегії. На цих макетах кнопка збереження зображена у двох різних станах.

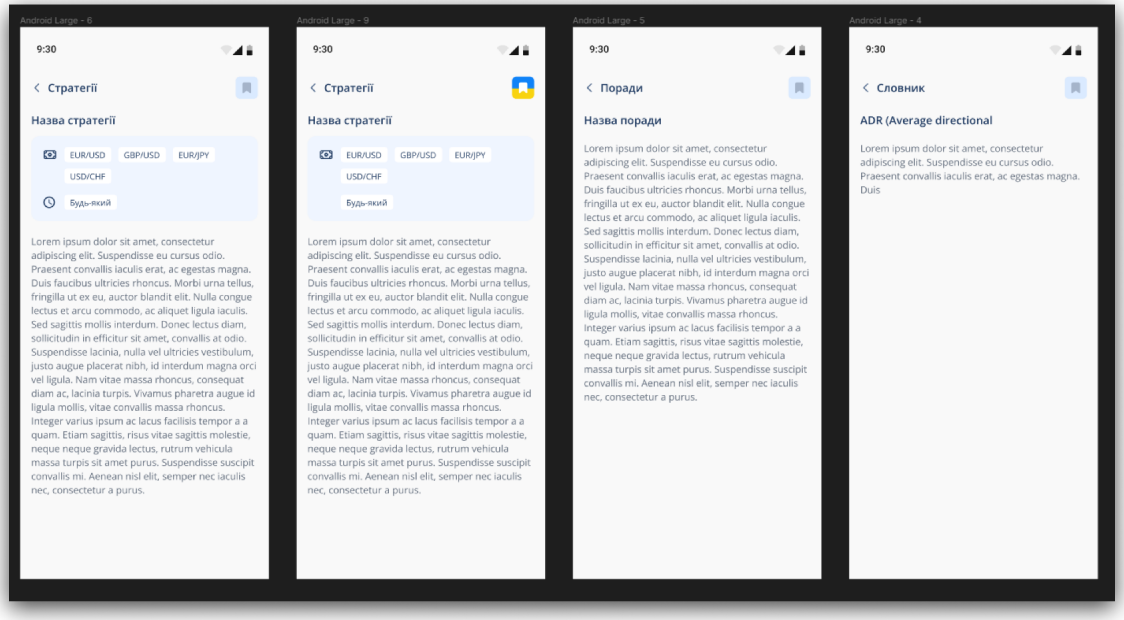


Рисунок 3.3 - Екран перегляду різних категорій інформації

Для розробки дизайну темної теми достатньо було створити декілька макетів існуючих екранів зі зміненими кольорами на основних елементах інтерфейсу. Це дозволило сформувати палітру для усього темного оформлення. На рисунку 3.4 зображено макети основних екранів з оптимальним зовнішнім виглядом, який дозволить більш зручно використовувати додаток у темний час.

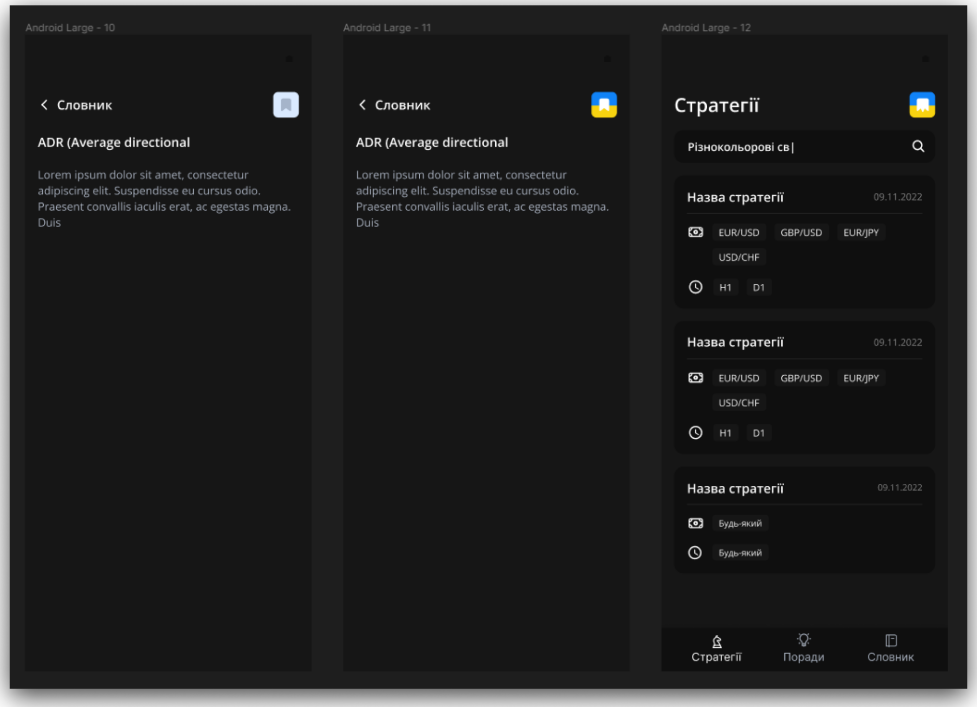


Рисунок 3.4 - Приклади макетів декількох екранів у темному оформленні

Завершальним етапом стала розробка зовнішнього вигляду іконки мобільного додатку. Для логотипу дуже важливо передавати суть продукту, для якого він створений. Саме тому було одночасно поєднано векторні зображення:

- графіку, адже додаток стосується саме тематики трейдингу;
- закладки, оскільки однією з основних функцій є збереження необхідної інформації для швидкого доступу до неї;
- українського прапору, адже інформаційний контент представлений саме українською мовою;

Фінальна версія логотипу зображена на рисунку 3.5

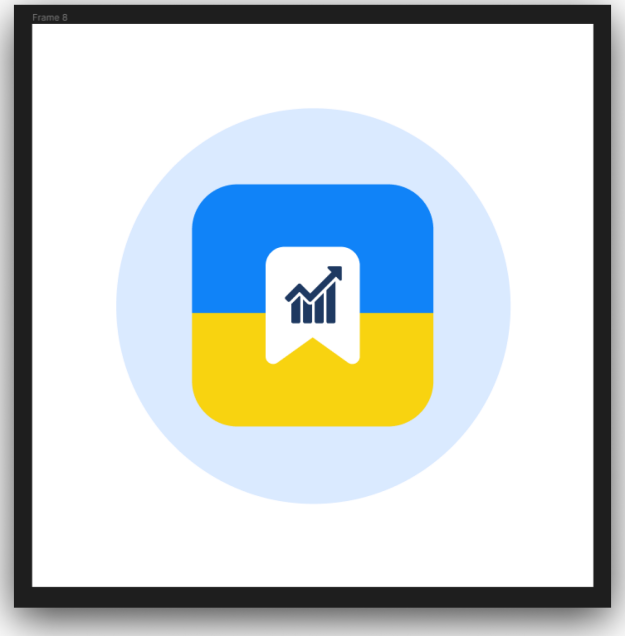


Рисунок 3.5 - Дизайн іконки мобільного додатку “Енциклопедія трейдера”

3.2. Розробка програмної частини додатку

Обравши основним середовищем розробки офіційне рішення від компанії Google - Android Studio, було розпочато процес розробки самого Android додатку. Для початку необхідно створити новий проект у середовищі.

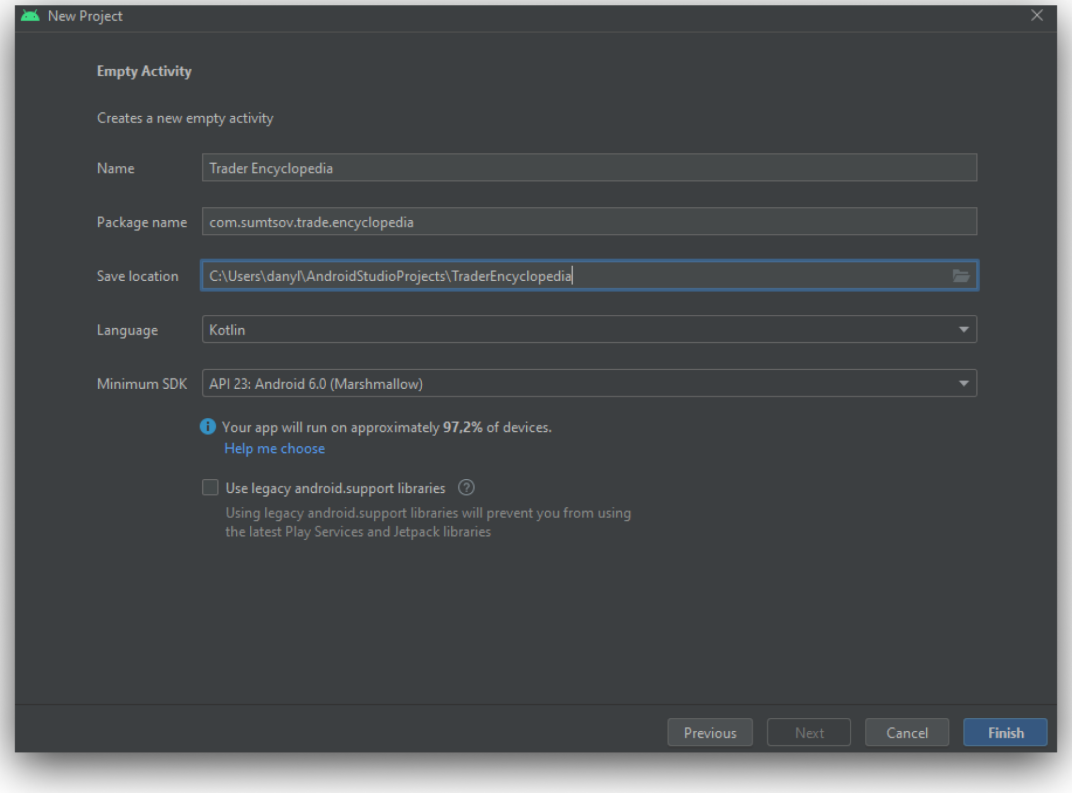


Рисунок 3.6 - Створення нового Android проекту

Наступним кроком стане імплементація усіх необхідних бібліотек та інструментів через додавання відповідних команд до файлу build.gradle, а саме: інструмент для спрощеної роботи з форматом JSON (Gson), інструмент для асинхронного завантаження зображень(Picasso), комплект розробки(Firebase SDK), тощо. На рисунку 3.7 зображено блок імплементації

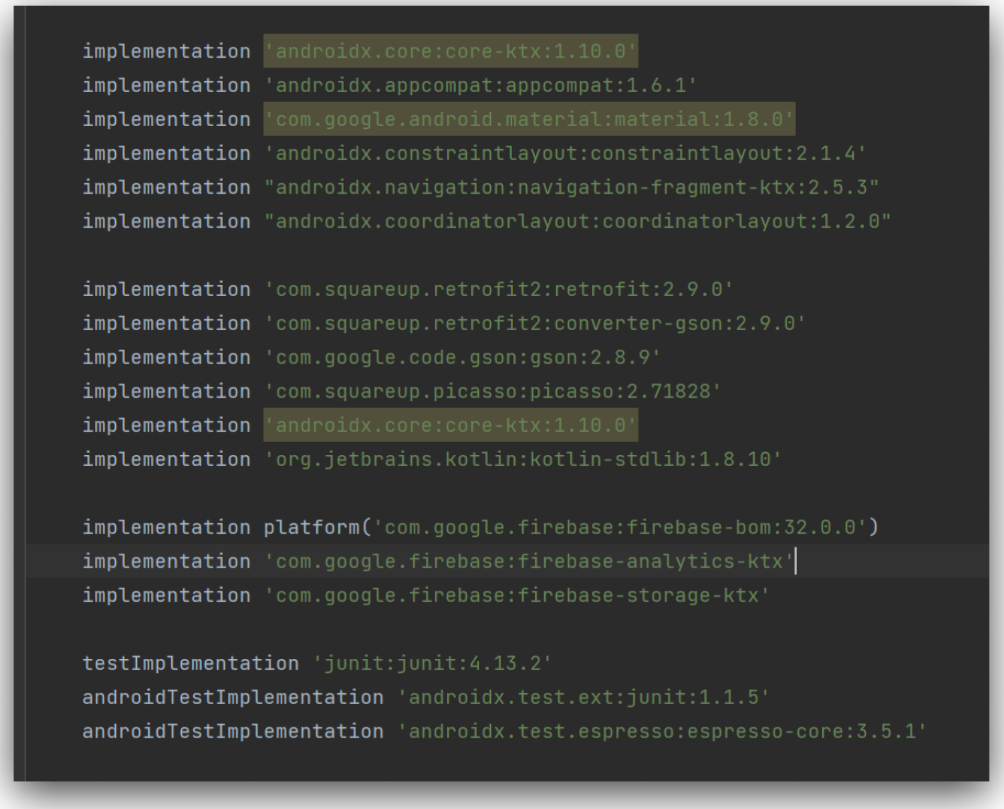
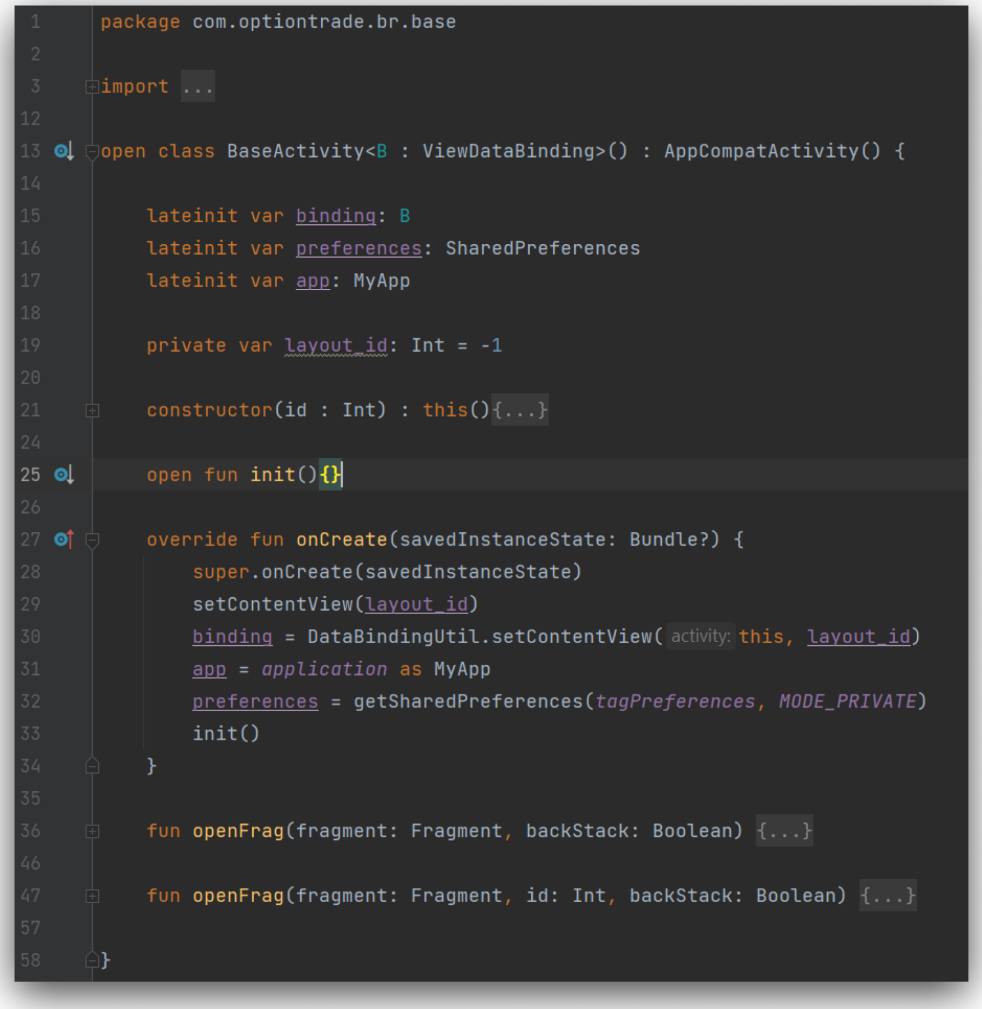


Рисунок 3.7 - Gradle-імплементація бібліотек до проекту

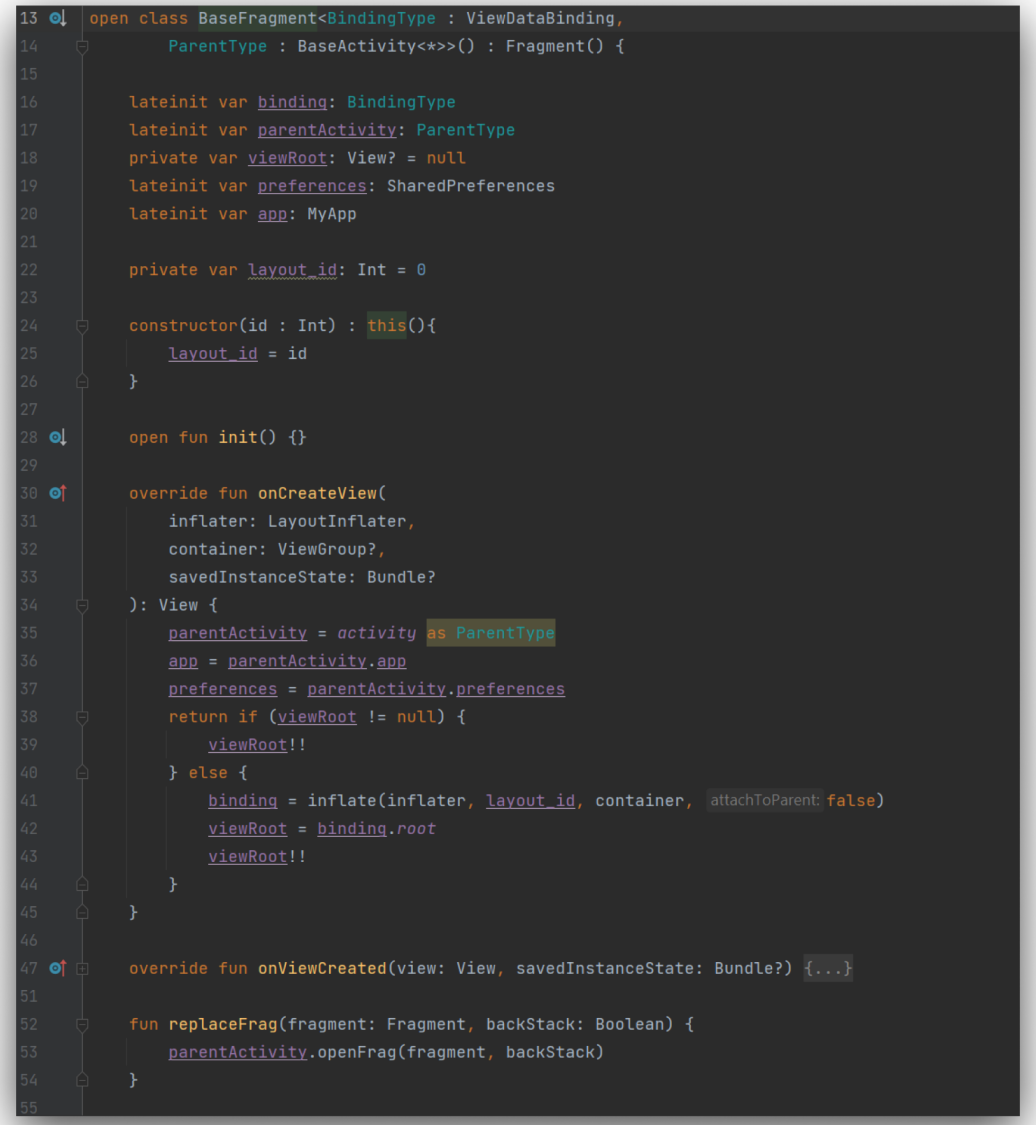
Для подальшого написання програмного коду та реалізації необхідних функцій, було створено набір базових класів. Клас BaseActivity.kt створено та налаштовано як батьківський клас для наступних активіті. В ньому реалізовано необхідні методи на конструктори для швидкого і компактного підключення макетів через AndroidBinding, перемикання фрагментів, доступу до локального сховища, тощо. Програмний код зображено на рисунку 3.8.



```
package com.optiontrade.br.base

import ...

open class BaseActivity<B : ViewDataBinding>() : AppCompatActivity() {

    lateinit var binding: B
    lateinit var preferences: SharedPreferences
    lateinit var app: MyApp

    private var layout_id: Int = -1

    constructor(id : Int) : this(){...}

    open fun init(){}

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(layout_id)
        binding = DataBindingUtil.setContentView( activity: this, layout_id)
        app = application as MyApp
        preferences = getSharedPreferences(tagPreferences, MODE_PRIVATE)
        init()
    }

    fun openFrag(fragment: Fragment, backStack: Boolean) {...}

    fun openFrag(fragment: Fragment, id: Int, backStack: Boolean) {...}

}
```

Рисунок 3.8 - Програмний код класу BaseActivity.kt

Другий базовий клас створений для успадкування наступними фрагментами. На рисунку 3.9 зображено програмний код класу BaseFragment.kt. Він також служить для полегшення реалізації деяких базових функцій.



```
open class BaseFragment<BindingType : ViewDataBinding,
        ParentType : BaseActivity<*>>() : Fragment() {

    lateinit var binding: BindingType
    lateinit var parentActivity: ParentType
    private var viewRoot: View? = null
    lateinit var preferences: SharedPreferences
    lateinit var app: MyApp

    private var layout_id: Int = 0

    constructor(id : Int) : this(){
        layout_id = id
    }

    open fun init() {}

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        parentActivity = activity as ParentType
        app = parentActivity.app
        preferences = parentActivity.preferences
        return if (viewRoot != null) {
            viewRoot!!
        } else {
            binding = inflate(inflater, layout_id, container,  attachToParent: false)
            viewRoot = binding.root
            viewRoot!!
        }
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {...}

    fun replaceFrag(fragment: Fragment, backStack: Boolean) {
        parentActivity.openFrag(fragment, backStack)
    }

```

Рисунок 3.9 - Програмний код класу BaseFragment.kt

Далі було створено всі необхідні моделі даних у вигляді дата-класів з функціями equal() і contains(s : String), для подальшої інтеграції пошуку. Зображено на рисунку 3.10.

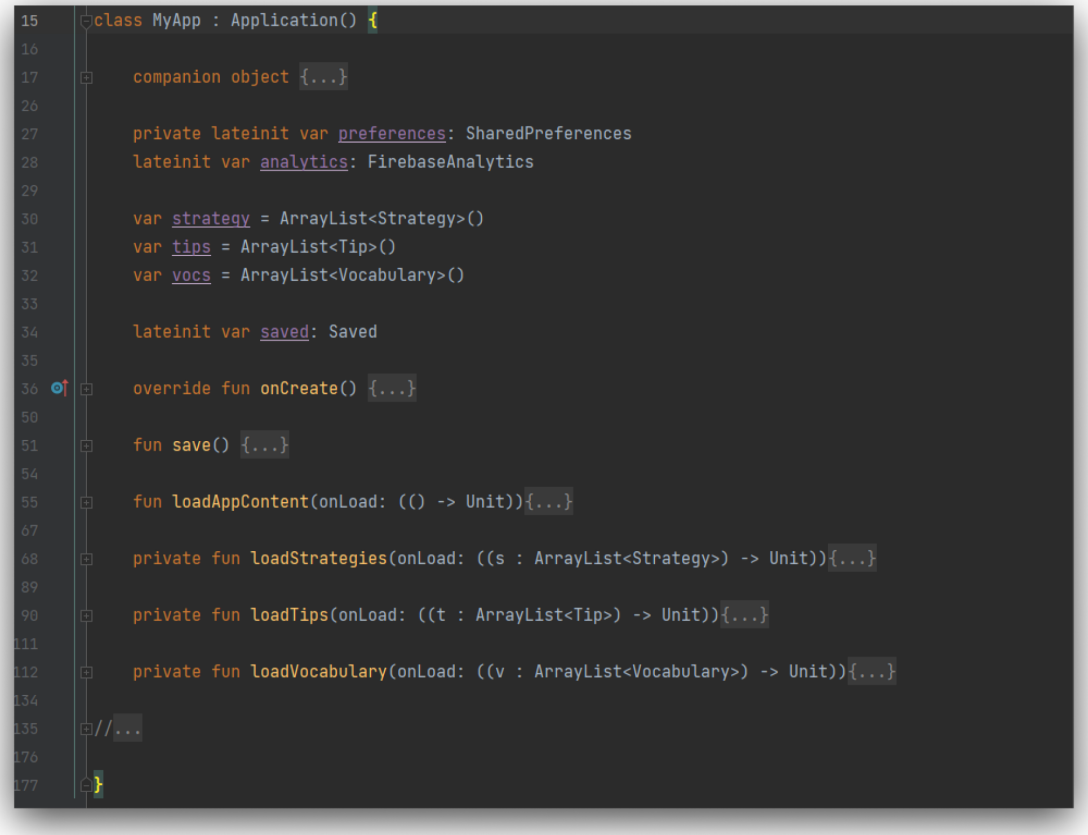
```
data class Strategy(
    val id: Int,
    val title: String,
    val date: Long,
    val text: String,
    val cur: ArrayList<String>,
    val tf: ArrayList<String>
){

    fun contains(s : String) : Boolean{
        return (this.title.contains(s, ignoreCase: true)
                || this.text.contains(s, ignoreCase: true))
    }

}

data class Tip(
    val id: Int,
    val title: String,
    val text: String
){

    fun contains(s : String) : Boolean{
        return (this.title.contains(s, ignoreCase: true)
                || this.text.contains(s, ignoreCase: true))
    }

}
```

```
data class Vocabulary(
    val letter: String,
    val content: ArrayList<VocContent>
)

data class VocContent(
    val id: Int,
    val title: String,
    val desc: String
){

    fun equal(v : VocContent) : Boolean{
        return (this.title == v.title
                && this.desc == v.desc)
    }

    fun contains(s : String) : Boolean{
        return (this.title.contains(s, ignoreCase: true))
    }

}

data class Saved(
    val str: ArrayList<Strategy>,
    val tips: ArrayList<Tip>,
    val vocs: ArrayList<VocContent>
)
```

Рисунок 3.10 - Моделі даних додатку

Наступним кроком було створення центрального класу MyApp.kt, що розширює системний клас Application. У ньому знаходяться необхідні константи, універсальні об’єкти необхідних типів даних, до яких матиметься доступ з будь-якого місця в додатку. Реалізовано функції завантаження та збереження даних, ініціалізація необхідних сервісів.

На рисунку 3.11 зображено програмний код створеного класу.



```
15  class MyApp : Application() {
16
17      companion object {...}
26
27      private lateinit var preferences: SharedPreferences
28      lateinit var analytics: FirebaseAnalytics
29
30      var strategy = ArrayList<Strategy>()
31      var tips = ArrayList<Tip>()
32      var vocs = ArrayList<Vocabulary>()
33
34      lateinit var saved: Saved
35
36      override fun onCreate() {...}
50
51      fun save() {...}
54
55      fun loadAppContent(onLoad: (() -> Unit)){...}
67
68      private fun loadStrategies(onLoad: ((s : ArrayList<Strategy>) -> Unit)){...}
89
90      private fun loadTips(onLoad: ((t : ArrayList<Tip>) -> Unit)){...}
111
112     private fun loadVocabulary(onLoad: ((v : ArrayList<Vocabulary>) -> Unit)){...}
134
135 //...
176
177 }
```

Рисунок 3.11 - Головний Application-клас додатку

Наступним кроком стала верстка макетів, за розробленим раніше дизайном, безпосередньо в середовищі Android Studio. За допомогою вбудованого інструменту Layout Editor, створено необхідні екрани додатку. На рисунку 3.12 зображено один із створених макетів, а саме fragment_strategy.xml.

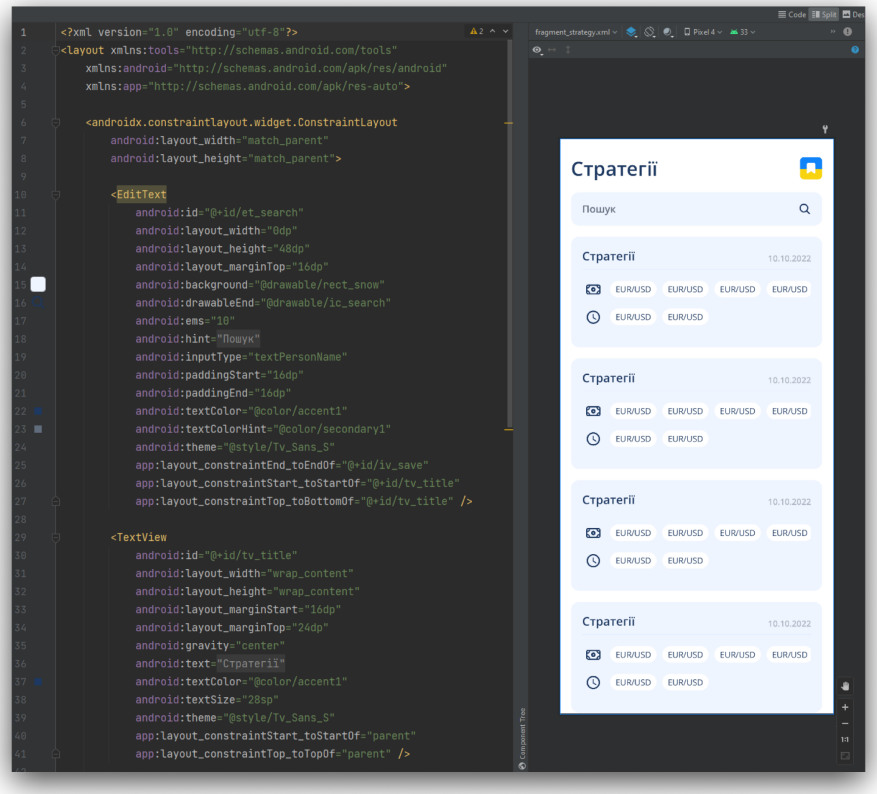


Рисунок 3.12 - layout-файл екрану стратегій

Для функціонування нічної теми додатку створено окремий ресурс файл з кольоровою палітрою темного оформлення.

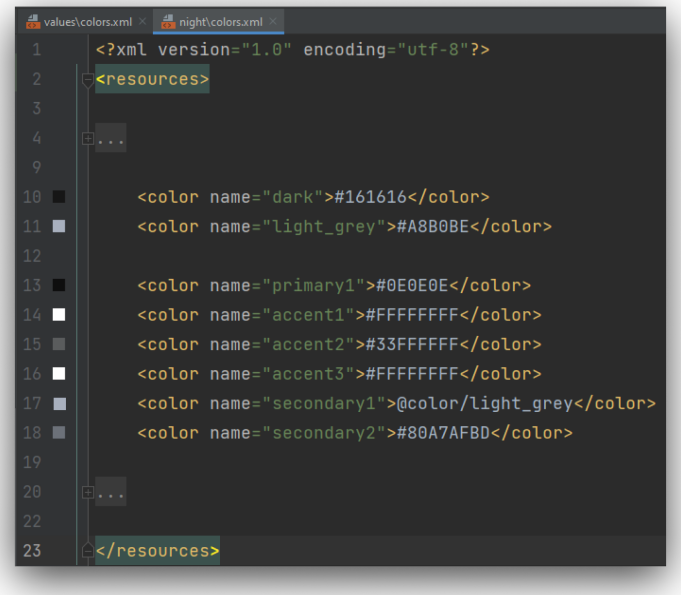


Рисунок 3.13 - values-файл кольорів нічної теми

Для кожного з макетів необхідно створити клас фрагменту, що буде його обробляти. Для екрану стратегій керуючим класом є StrategyFragment.kt, в методі init() виконуємо копіювання списку стратегій в окремий об'єкт, налаштовуємо віджет списку та завантажуємо дані в StrategyAdapter, додаємо обробник натискання для кнопки збереження. Програмний код функції init() зображено на рисунку 3.13

Для коректної роботи списку стратегій необхідно створити клас StrategyAdapter.kt, що розширює системний RecyclerView.Adapter, з внутрішнім класом MyViewHolder. В функції bind(strategy: Strategy) реалізовано відображення даних з моделі у графічному інтерфейсі та виклик зворотнього дзвінку на подію натискання на пункт списку. Код зображено на рисунку 3.14.

Додатково було розроблено алгоритм пошуку у вигляді функції search(), яка використовує попередньо прописані, в моделях даних, методи перевірки наявності за назвою та описом. На рисунку 3.15 представлений програмний код функції. Аналогічно було реалізовано класи фрагментів для порад і словника.

```
@SuppressLint("SetTextI18n")
override fun init() {
    with(binding) { this: FragmentStrategyBinding
        str = ArrayList(app.strategy)
        rvStrategy.adapter = StrategyAdapter(str) { it: Strategy
            val intent = Intent(requireContext(), OpenActivity::class.java)
            intent.action = OpenActivity.actionStrategy
            intent.putExtra(OpenActivity.keyContent, toJson(it))
            startActivity(intent)
        }
        ivSave.setOnClickListener { it: View!
            val intent = Intent(requireContext(), OpenActivity::class.java)
            intent.action = OpenActivity.actionSaved
            startActivity(intent)
        }
        search()
    }
}
```

Рисунок 3.14 - Функція init класу StrategyFragment.kt

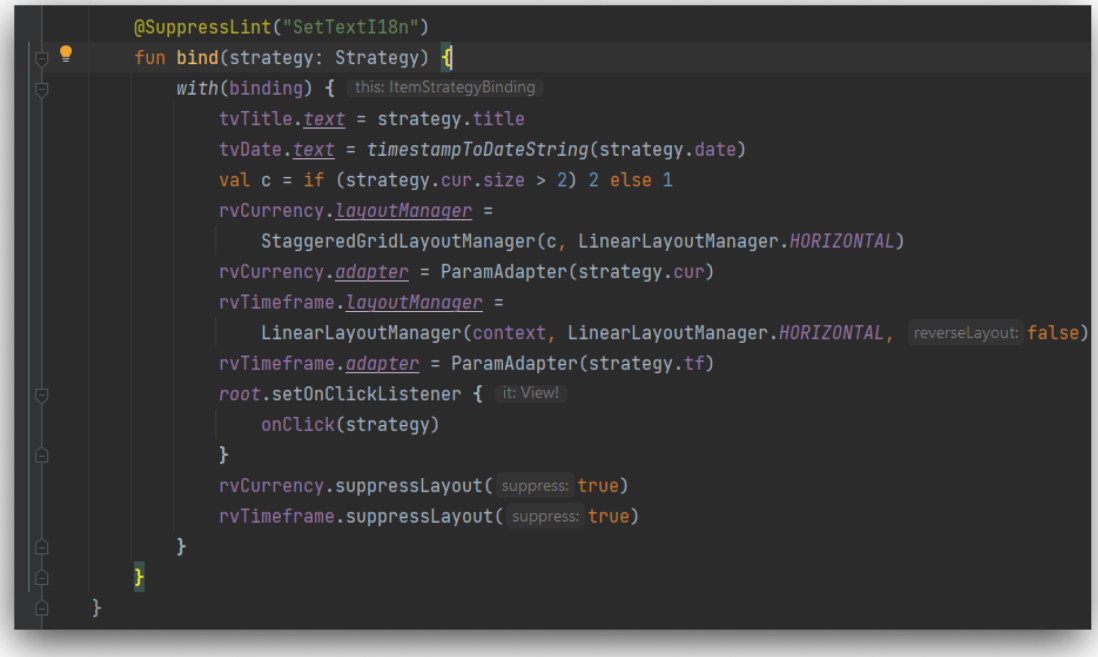


Рисунок 3.15 - Функція bind адаптера StrategyAdapter.kt

Рисунок 3.16 - Функція search в класі StrategyFragment.kt

Розділ “Збережені” було реалізовано за допомогою класу SavedFragment.kt. В фукнції init() для кожної вкладинки підключено об’єкт

ClickListener, що перемикає вибірку збережених даних і керує графічним інструментом в залежності від елементу. Код класу зображено на рисунку 3.16.

Для обраних стратегій та порад використовуються раніше створені адаптери. Проте у випадку з словником необхідний додатковий, без групування термінів. SavedVocAdapter.kt зображений на рисунку 3.17.

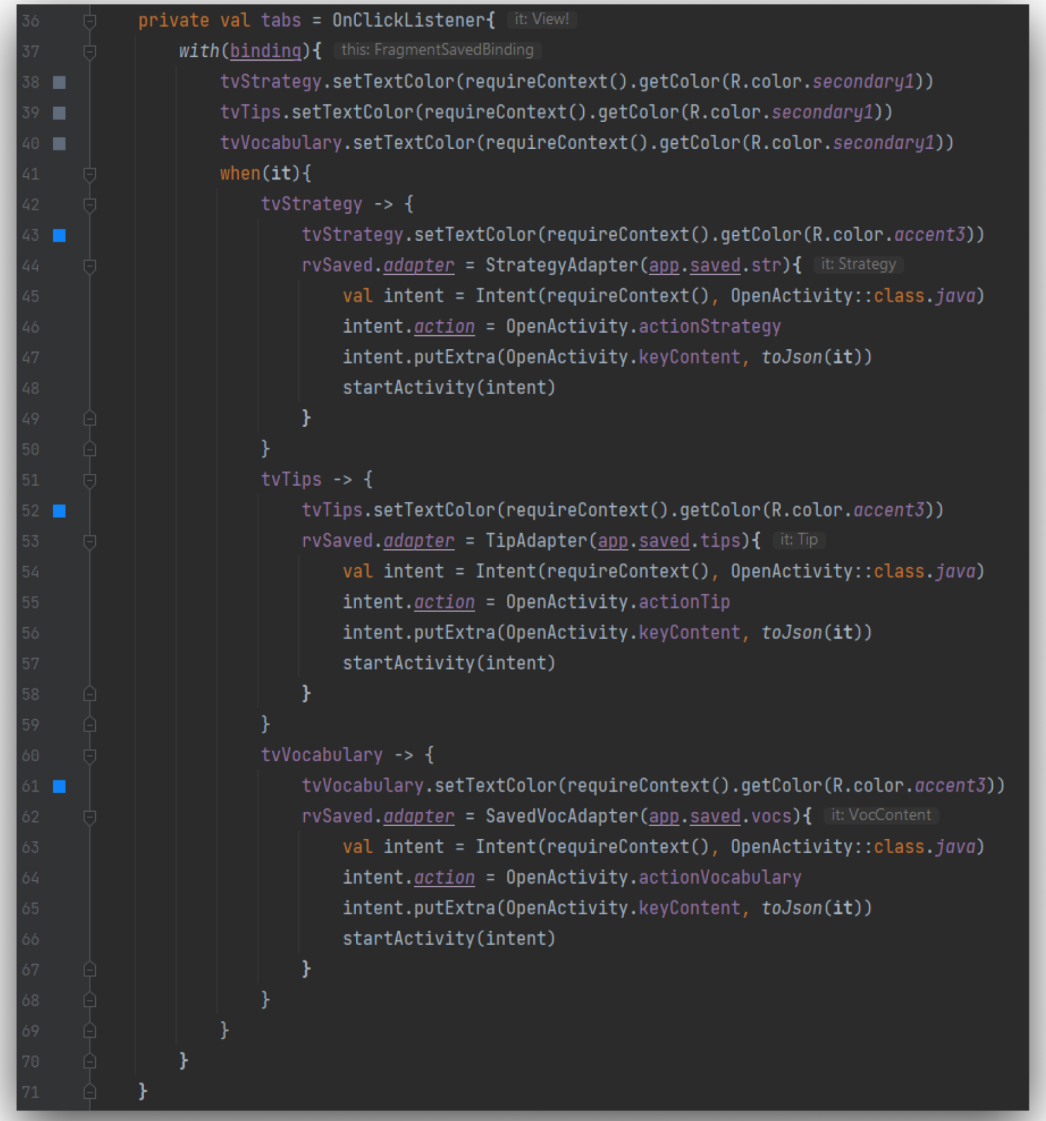


Рисунок 3.17 - Об’єкт-Listener в класі SavedFragment.kt

```
class SavedVocAdapter(
    private val items: ArrayList<VocContent>,
    private val onClick: ((vocContent: VocContent) -> Unit)
) :
    RecyclerView.Adapter<SavedVocAdapter.MyViewHolder>() {
```

Рисунок 3.18 - SavedVocAdapt.kt

Наступним кроком було створення класу OpenFragment.kt, що відповідає за перегляд відкритого інформаційного блоку різних категорій. У ньому реалізовано три функції відображення контенту через графічний інтерфейс. Код преставлений на рисунку 3.18.

На рисунку 3.19 зображено програмний код функції init(), в якій відбувається виклик функцій content(), а також обробка кнопки збереження, в залежності від категорії відкритого контенту.

```
private fun content(s: Strategy) {
    with(binding) {
        tvTitle.text = "Стратегії"
        tvContentTitle.text = s.title
        tvDesc.text = s.text
        clParams.visibility = View.VISIBLE
        val c = if (s.cur.size > 2) 2 else 1
        rvCurrency.layoutManager =
            StaggeredGridLayoutManager(c, LinearLayoutManager.HORIZONTAL)
        rvCurrency.adapter = ParamAdapter(s.cur)
        rvTimeframe.layoutManager =
            LinearLayoutManager(requireContext(), LinearLayoutManager.HORIZONTAL, false)
        rvTimeframe.adapter = ParamAdapter(s.tf)
        app.saved.str.forEach {
            if (it.id == s.id)
                ivSave.tag = true
        }
    }
}
```

```
private fun content(t: Tip) {
    with(binding) {
        tvTitle.text = "Поради"
        tvContentTitle.text = t.title
        tvDesc.text = t.text
        clParams.visibility = View.GONE
        app.saved.tips.forEach {
            if (it.id == t.id)
                ivSave.tag = true
        }
    }
}
```

```
private fun content(v: VocContent) {
    with(binding) {
        tvTitle.text = "Словник"
        tvContentTitle.text = v.title
        tvDesc.text = v.desc
        clParams.visibility = View.GONE
        app.saved.vocs.forEach {
            if (it.equal(v))
                ivSave.tag = true
        }
    }
}
```

Рисунок 3.19 - Програмний код функцій відображення контенту на екрані перегляду

Рисунок 3.20 - Програмний код функцій відображення контенту на екрані перегляду

Перед етапом інформаційного забезпечення необхідно налаштувати хмарне сховище Firebase Storage, для подальшого завантаження контенту [8]. У відповідні константи записано адреси до файлів у сховищі. В класі MyApp.kt для кожної категорії інформаційного контенту реалізовано функції завантаження і парсингу даних.

На рисунку 3.20 зображено інтерфейс розділу хмарного сховища консолі Firebase, на рисунку 3.21 - константи з адресами. Програмний код функції завантаження стратегій представлений на рисунку 3.22.

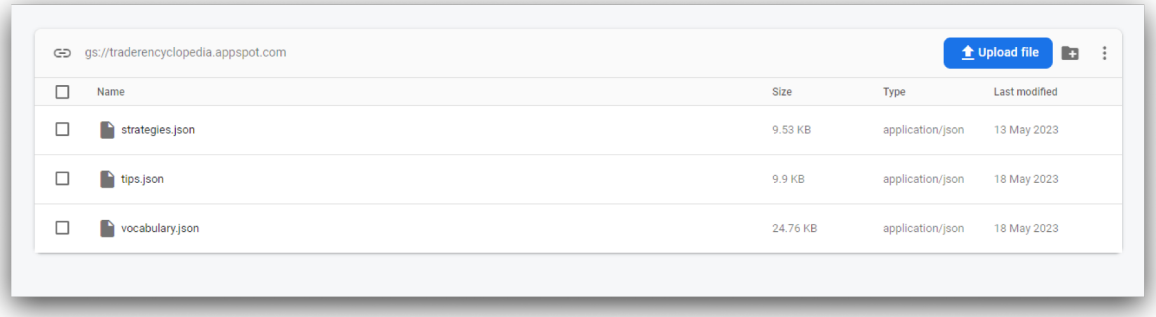


Рисунок 3.21 - Інтерфейс налаштування хмарного сховища Firebase

```
private const val firebaseRootPath = "gs://traderencyclopedia.appspot.com/"
const val strategiesPath = "${firebaseRootPath}strategies.json"
const val tipsPath = "${firebaseRootPath}tips.json"
const val vocabularyPath = "${firebaseRootPath}vocabulary.json"
```

Рисунок 3.22 - Константи з адресами завантаження файлів з контентом

```
Firebase.storage.getReferenceFromUrl(strategiesPath) StorageReference
    .stream StreamDownloadTask
    .addOnCompleteListener { it: Task<StreamDownloadTask.TaskSnapshot!> 
        if (it.isSuccessful) {
            CoroutineScope(Dispatchers.IO).launch { this: CoroutineScope
                val s: Scanner =
                    Scanner(it.result.stream, charsetName: "UTF-8")
                        .useDelimiter( pattern: "\\A")
                val data = if (s.hasNext()) s.next() else ""
                val strategies = fromJson(
                    data,
                    Array<Strategy>::class.java
                )
                onLoad(ArrayList(strategies.toList()))
            }
        } else {
            log("Load is not successful")
        }
    }
```

Рисунок 3.23 - Функція завантаження стратегій з хмарного сховища

3.3. Інформаційне забезпечення додатку

На етапі інформаційного забезпечення було заповнено JSON-файли стратегій, порад і словника, відповідним контентом. Формат об’єктів та їх поля відповідають спроектованим раніше моделям даних, і включають у себе всю необхідну інформацію.

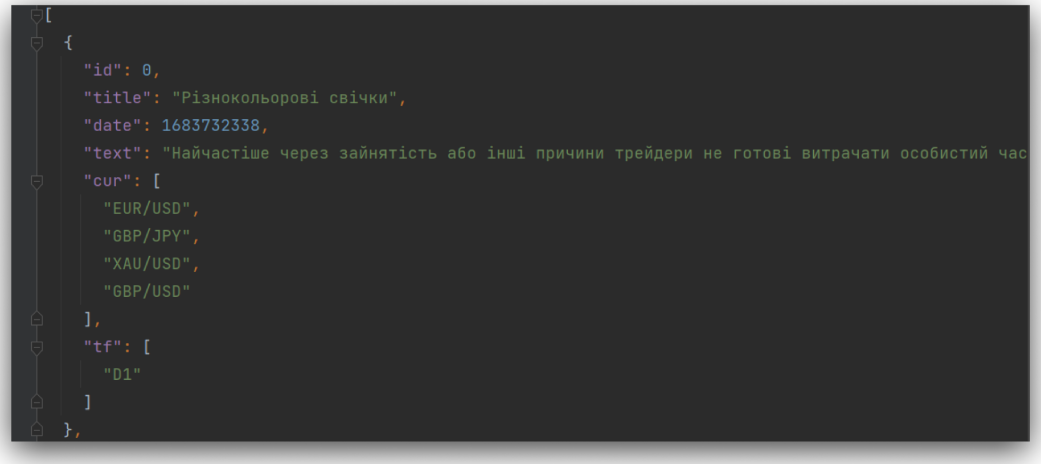


Рисунок 3.24- Фрагмент JSON-файлу зі стратегіями

Заповнені файли завантажено до хмарного сховища. Вони мають можливість оновлюватись в будь-який час, зміни одразу будуть доступні в додатку.

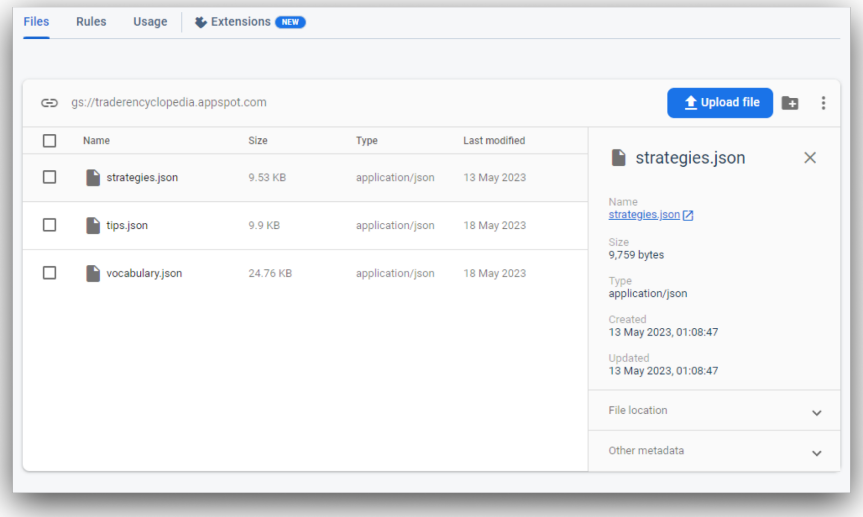


Рисунок 3.25- Завантажені до сховища файли, наповнені контентом

3.4. Тестування фінальної версії додатку

Перед самим процесом тестування, у Firebase проекті було ввімкнено вбудований сервіс аналітики [9].

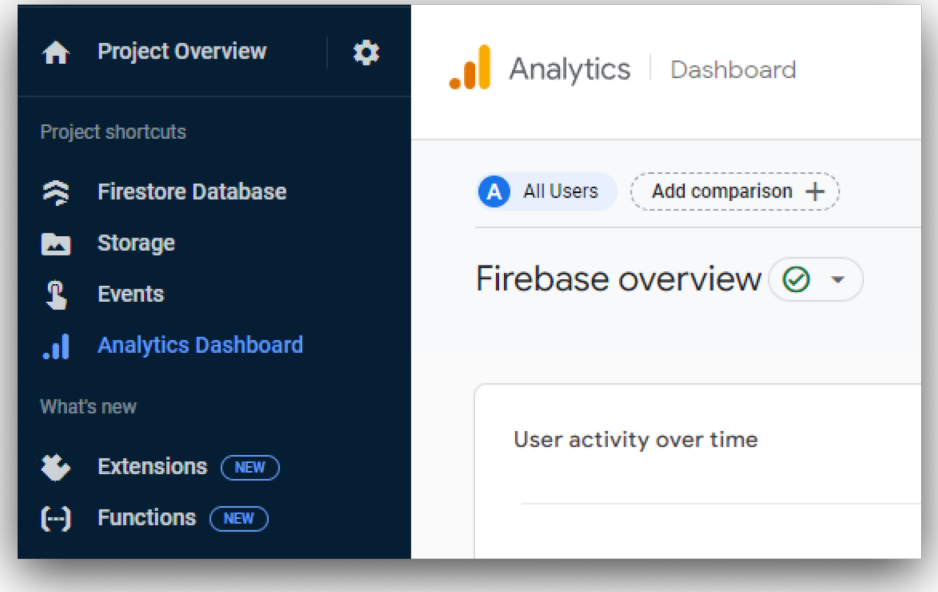


Рисунок 3.26- Розділ аналітики в консолі Firebase

Наступним кроком, на клієнті додатку створено керуючий файл Analytics.kt, в якому прописані необхідні константи з назвами подій, що будуть надсилатися до аналітичної платформи.

```
const val eventStrategyOpen = "eventStrategyOpen"
const val eventTipOpen = "eventTipOpen"
const val eventVocabularyOpen = "eventVocabularyOpen"

const val eventStrategyBookmarked = "eventStrategyOpen"
const val eventTipBookmarked = "eventTipBookmarked"
const val eventVocabularyBookmarked = "eventVocabularyBookmarked"
```

Рисунок 3.27- Константи з назвами моніторингових-подій додатку

У внутрішньому об’єкті Events реалізовано функції відправки подій, які викликаються у відповідних місцях і при відповідних діях користувача у додатку.

```
object Events{

    fun eventStrategyOpen(a: FirebaseAnalytics){
        a.logEvent(eventStrategyOpen, params: null)
    }

    fun eventTipOpen(a: FirebaseAnalytics){
        a.logEvent(eventTipOpen, params: null)
    }

    fun eventVocabularyOpen(a: FirebaseAnalytics){
        a.logEvent(eventVocabularyOpen, params: null)
    }

    fun eventStrategyBookmarked(a: FirebaseAnalytics){
        a.logEvent(eventStrategyBookmarked, params: null)
    }

    fun eventTipBookmarked(a: FirebaseAnalytics){
        a.logEvent(eventTipBookmarked, params: null)
    }

    fun eventVocabularyBookmarked(a: FirebaseAnalytics){
        a.logEvent(eventVocabularyBookmarked, params: null)
    }

}
```

Рисунок 3.28- Функції відправки подій до FirebaseAnalytics

Останнім кроком у налаштуванні перед тестуванням було створення декількох віртуальних пристроїв через вбудований у середовище розробки

інструмент Android Emulator. Для тестування обирались пристрої різних версій операційної системи.

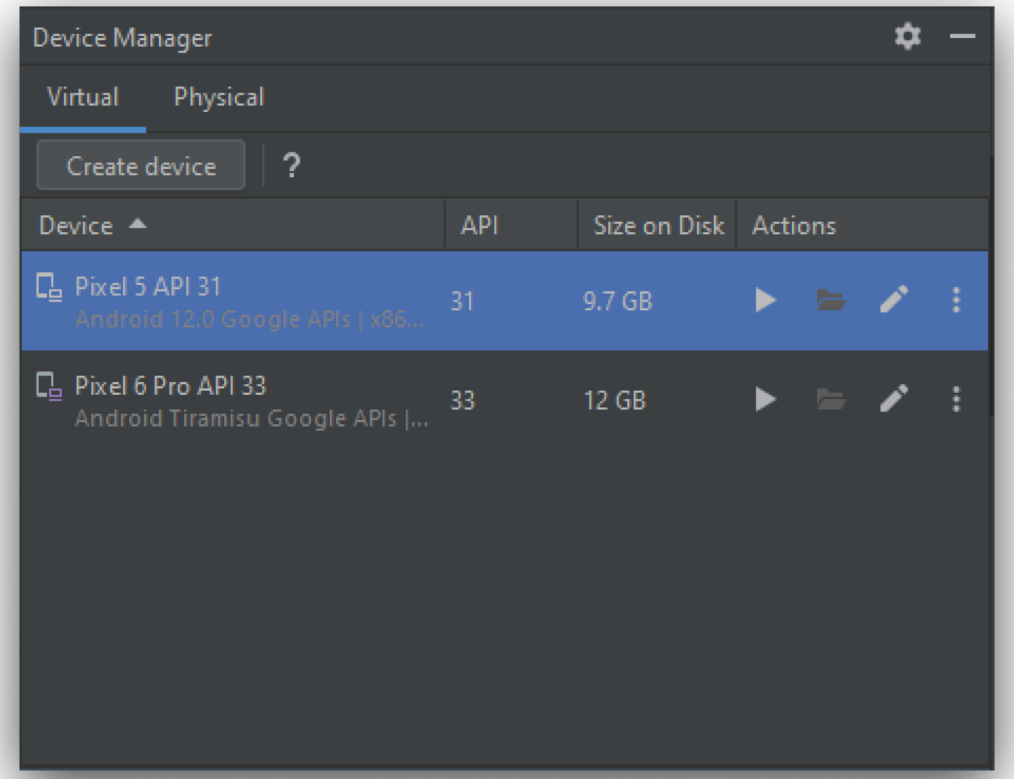


Рисунок 3.29- Список створених емуляторів Android пристроїв

Завершальним кроком стала сама процедура тестування. На створених раніше емуляторах було запущено фінальні збірки проекту. Перевірені основні моменти, такі як: завантаження даних у списки, перегляду інформаційних блоків, пошук контенту по фрагментам назви і опису, додавання позицій до “Збережених”, коректність відображення екранів, можливість перемикання теми інтерфейсу. Додаток працює стабільно і виконує всю закладену в нього функціональність. На рисунку 3.30 зображений процес тестування збірки на різних пристроях.

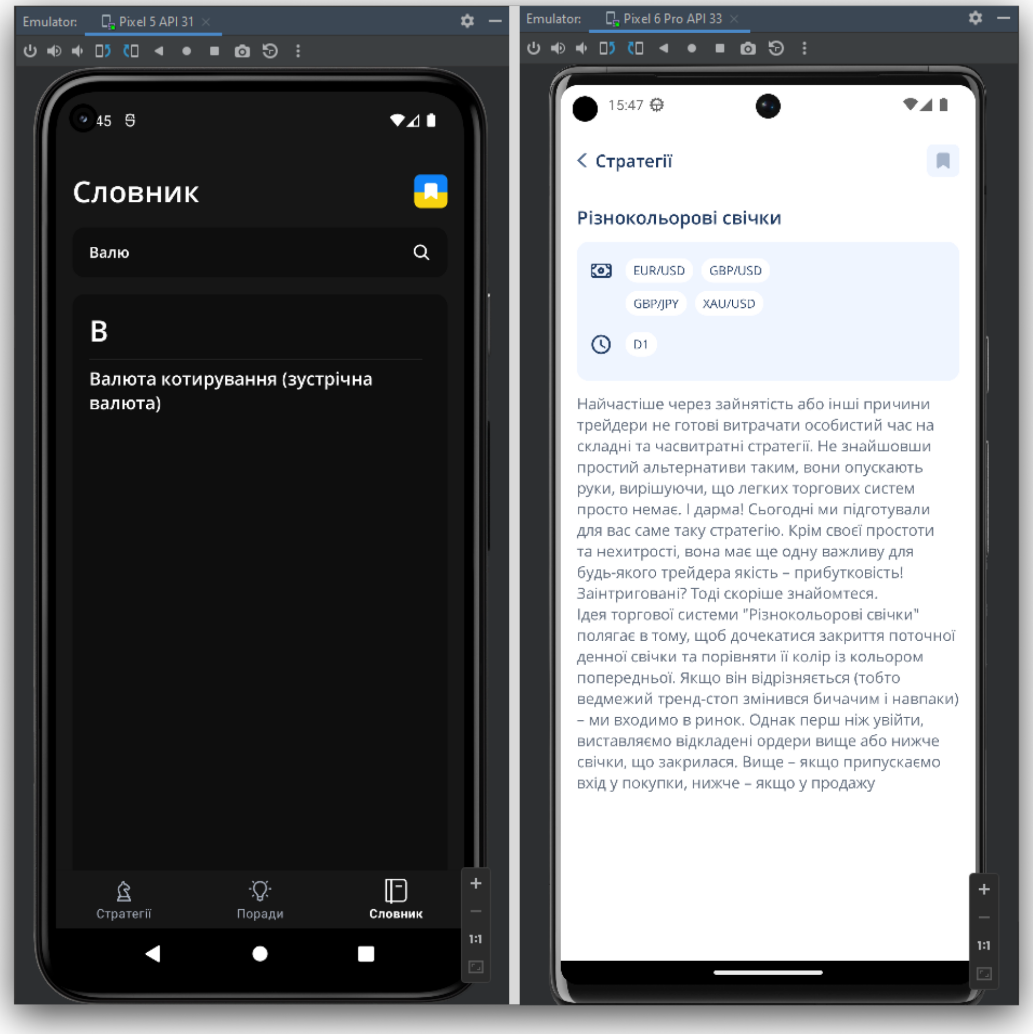


Рисунок 3.30- Процес тестування збірки додатку на віртуальних пристроях

ВИСНОВКИ

Результатом кваліфікаційної роботи бакалавра на тему “Інформаційне та програмне забезпечення навчального мобільного додатку "Енциклопедія трейдера"” є готовий мобільний Android додаток, що представляє собою програмний продукт й має функціональні можливості для подальшого вдосконалення.

Всі вимоги та задачі описані в постановці завдання до проекту виконані. Результатами виконання роботи є наступне :

- можливість перегляду стратегій та порад для трейдингу
- можливість перегляду словника трейдера
- можливість пошуку кожного типу інформації за назвою та описом
- доступ без реєстрації
- можливість збереження окремих пунктів до “Обраних”

Процес розробки включав в себе декілька етапів:

- аналіз предметної області: розглянуто питання актуальності тематики, типи мобільних додатків, операційні системи, обрані оптимальні рішення, сформовано чіткі мету та задачі розробляємого продукту
- формування методики вирішення поставленої задачі: розглянуто різні інструменти для проектування інтерфейсу, розробки під ОС Android, інтеграції хмарних сервісів, спроектовано структуру додатку і моделі даних.
- практична реалізація: розроблено дизайн інтерфейсу, програмну частину на мові Kotlin, з використанням актуальних технологій створення Android додатків, проведено інформаційне забезпечення контентом та процедуру тестування на декількох пристроях.

Результати тестування готового продукту:

- Коректність відображення графічного інтерфейсу на пристроях різних діагоналей та версій ОС
- Відсутність помилок та вильотів додатку під час використання
- Якісне функціонування згідно початкового плану

В такому вигляді продукт має великі шанси знайти відгук від потенційної аудиторії. Інтегрована аналітична платформа дасть змогу відстежувати бізнес-показники і виявляти нові потреби користувачів, які будуть задовольнятися по мірі оновлень. Наприклад, надалі планується інтеграція інструментів відстеження та аналізу ринку, реалізація симулятора торгівлі й т.п.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 7 Top App Categories for 2022 [Електронний ресурс] - Режим доступу: URL : https://www.eventya.net/eventya-tips-tricks/7-top-app-categories-for-2022/

2. Сучасні реалії інвестування та трейдингу: плюси та мінуси професій [Електронний ресурс] - Режим доступу: URL : https://financy.24tv.ua/novi-mozhlivosti-dlya-kozhnogo-plyusi-minusi-investuvannya-treydingu_n2323692

3. Типи мобільних додатків [Електронний ресурс] - Режим доступу: URL : https://smile-ukraine.com/ua/mobile-apps/mobile-apps-types

4. Частка ринку Android та iOS: оприлюднено статистику 2022 року [Електронний ресурс] - Режим доступу: URL : https://root-nation.com/ua/news-ua/it-news-ua/ua-android-ios-statistika-2022/

5. Android Developers guides [Електронний ресурс] - Режим доступу : URL : https://developer.android.com/docs

6. Мінімально життєздатний продукт: максимальна користь для стартапів [Електронний ресурс] - Режим доступу: URL : https://internetdevels.ua/blog/minimum-viable-product-for-startups

7. Skeen J., Greenhalgh D., Bailey A. Kotlin Programming: The Big Nerd Ranch Guide (2nd edition. Edition). 2021. 541 p.

8. Neil Smyth. Android Studio 4.1 Development Essentials - Kotlin Edition: Developing Android 11 Apps Using Android Studio 4.1, Kotlin and Android Jetpack Paperback. Payload Media, 2020. 822 p.

9. Get started with Cloud Storage on Android [Електронний ресурс] - Режим доступу : URL : https://firebase.google.com/docs/storage/android/start

10. Get started with Google Analytics [Електронний ресурс] - Режим доступу : URL : https://firebase.google.com/docs/analytics/get-started?platform=android

11. STACKOVERFLOW Questions tagged [android] [Електронний ресурс] - Режим доступу : URL : https://stackoverflow.com/questions/tagged/android

ДОДАТОК А

fragment_strategy.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:tools="http://schemas.android.com/tools"
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto">

  <androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
      android:id="@+id/et_search"
      android:layout_width="0dp"
      android:layout_height="48dp"
      android:layout_marginTop="16dp"
      android:background="@drawable/rect_snow"
      android:drawableEnd="@drawable/ic_search"
      android:ems="10"
      android:hint="@string/search"
      android:inputType="textPersonName"
      android:paddingStart="16dp"
      android:paddingEnd="16dp"
      android:textColor="@color/accent1"
      android:textColorHint="@color/secondary1"
      android:theme="@style/Tv_Sans_S"
      app:layout_constraintEnd_toEndOf="@+id/iv_save"
      app:layout_constraintStart_toStartOf="@+id/tv_title"
      app:layout_constraintTop_toBottomOf="@+id/tv_title" />

    <TextView
      android:id="@+id/tv_title"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginStart="16dp"
      android:layout_marginTop="24dp"
      android:gravity="center"
      android:text="@string/strategy"
```

```
        android:textColor="@color/accent1"
        android:textSize="28sp"
        android:theme="@style/Tv_Sans_S"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageView
        android:id="@+id/iv_save"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_marginEnd="16dp"
        app:layout_constraintBottom_toBottomOf="@+id/tv_title"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="@+id/tv_title"
        app:srcCompat="@drawable/ic_bookmark" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rv_strategy"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:clipToPadding="false"
        android:paddingBottom="16dp"
        app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/et_search"
        app:layout_constraintVertical_bias="1.0"
        tools:itemCount="10"
        tools:listitem="@layout/item_strategy" />

  </androidx.constraintlayout.widget.ConstraintLayout>

</layout>
```

fragment_tips.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:tools="http://schemas.android.com/tools"
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto">

  <androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
      android:id="@+id/et_search"
      android:layout_width="0dp"
      android:layout_height="48dp"
      android:layout_marginTop="16dp"
      android:background="@drawable/rect_snow"
      android:drawableEnd="@drawable/ic_search"
      android:ems="10"
      android:hint="@string/search"
      android:inputType="textPersonName"
      android:paddingStart="16dp"
      android:paddingEnd="16dp"
      android:textColor="@color/accent1"
      android:textColorHint="@color/secondary1"
      android:theme="@style/Tv_Sans_S"
      app:layout_constraintEnd_toEndOf="@+id/iv_save"
      app:layout_constraintStart_toStartOf="@+id/tv_title"
      app:layout_constraintTop_toBottomOf="@+id/tv_title" />

    <TextView
      android:id="@+id/tv_title"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginStart="16dp"
      android:layout_marginTop="24dp"
      android:gravity="center"
      android:text="@string/tips"
      android:textColor="@color/accent1"
      android:textSize="28sp"
      android:theme="@style/Tv_Sans_S"
      app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintTop_toTopOf="parent" />

    <ImageView
        android:id="@+id/iv_save"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_marginEnd="16dp"
        app:layout_constraintBottom_toBottomOf="@+id/tv_title"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="@+id/tv_title"
        app:srcCompat="@drawable/ic_bookmark" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rv_tips"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:clipToPadding="false"
        android:paddingBottom="16dp"
        app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/et_search"
        app:layout_constraintVertical_bias="0.0"
        tools:itemCount="10"
        tools:listitem="@layout/item_tip" />

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

fragment_vocabulary.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:tools="http://schemas.android.com/tools"
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto">

  <androidx.constraintlayout.widget.ConstraintLayout
```

```
android:layout_width="match_parent"
android:layout_height="match_parent">

<EditText
    android:id="@+id/et_search"
    android:layout_width="0dp"
    android:layout_height="48dp"
    android:layout_marginTop="16dp"
    android:background="@drawable/rect_snow"
    android:drawableEnd="@drawable/ic_search"
    android:ems="10"
    android:hint="@string/search"
    android:inputType="textPersonName"
    android:paddingStart="16dp"
    android:paddingEnd="16dp"
    android:textColor="@color/accent1"
    android:textColorHint="@color/secondary1"
    android:theme="@style/Tv_Sans_S"
    app:layout_constraintEnd_toEndOf="@+id/iv_save"
    app:layout_constraintStart_toStartOf="@+id/tv_title"
    app:layout_constraintTop_toBottomOf="@+id/tv_title" />

<TextView
    android:id="@+id/tv_title"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginTop="24dp"
    android:gravity="center"
    android:text="@string/vocabulary"
    android:textColor="@color/accent1"
    android:textSize="28sp"
    android:theme="@style/Tv_Sans_S"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

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

```
        android:layout_marginEnd="16dp"
        app:layout_constraintBottom_toBottomOf="@+id/tv_title"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="@+id/tv_title"
        app:srcCompat="@drawable/ic_bookmark" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rv_vocabulary"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="-16dp"
        android:background="@drawable/rect_snow"
        android:clipToPadding="false"
        android:paddingBottom="16dp"
        app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/et_search"
        app:layout_constraintVertical_bias="0.0"
        tools:itemCount="10"
        tools:listitem="@layout/item_voc_content" />

  </androidx.constraintlayout.widget.ConstraintLayout>

</layout>
```

fragment_open.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:tools="http://schemas.android.com/tools"
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto">

  <androidx.core.widget.NestedScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fillViewport="true">
```

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingBottom="16dp">

    <TextView
        android:id="@+id/tv_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="24dp"
        android:drawableStart="@drawable/ic_back"
        android:drawablePadding="8dp"
        android:gravity="center"
        android:text="@string/strategy"
        android:textColor="@color/accent1"
        android:textSize="17sp"
        android:theme="@style/Tv_Sans_S"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/tv_content_title"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="32dp"
        android:layout_marginEnd="16dp"
        android:gravity="start"
        android:text="Title"
        android:textColor="@color/accent1"
        android:textSize="18sp"
        android:theme="@style/Tv_Sans_S"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tv_title" />

    <TextView
        android:id="@+id/tv_desc"
```

```
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginTop="12dp"
    android:gravity="start"
    android:text="Description"
    android:textColor="@color/secondary1"
    android:textSize="15sp"
    android:theme="@style/Tv_Sans_R"
    app:layout_constraintEnd_toEndOf="@+id/iv_save"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/cl_params" />

<ImageView
    android:id="@+id/iv_save"
    android:layout_width="wrap_content"
    android:layout_height="32dp"
    android:layout_marginEnd="16dp"
    app:layout_constraintBottom_toBottomOf="@+id/tv_title"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="@+id/tv_title"
    app:srcCompat="@drawable/ic_bookmark_unch" />

<androidx.constraintlayout.widget.ConstraintLayout
    android:id="@+id/cl_params"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="12dp"
    android:background="@drawable/rect_snow"
    app:layout_constraintEnd_toEndOf="@+id/iv_save"
    app:layout_constraintStart_toStartOf="@+id/tv_content_title"
    app:layout_constraintTop_toBottomOf="@+id/tv_content_title">

    <ImageView
        android:id="@+id/iv_save2"
        android:layout_width="32dp"
        android:layout_height="25dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        app:layout_constraintStart_toStartOf="parent"
```

```
            app:layout_constraintTop_toTopOf="parent"
            app:srcCompat="@drawable/ic_currency" />

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/rv_currency"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            app:layoutManager="androidx.recyclerview.widget.StaggeredGridLayoutManager"
            app:layout_constraintStart_toEndOf="@+id/iv_save2"
            app:layout_constraintTop_toTopOf="@+id/iv_save2"
            tools:itemCount="4"
            tools:listitem="@layout/item_chip" />

        <ImageView
            android:id="@+id/iv_save3"
            android:layout_width="32dp"
            android:layout_height="25dp"
            android:layout_marginTop="8dp"
            android:padding="1dp"
            app:layout_constraintStart_toStartOf="@+id/iv_save2"
            app:layout_constraintTop_toBottomOf="@+id/rv_currency"
            app:srcCompat="@drawable/ic_time" />

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/rv_timeframe"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginBottom="16dp"
            android:orientation="horizontal"
            app:layoutManager="androidx.recyclerview.widget.StaggeredGridLayoutManager"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toEndOf="@+id/iv_save2"
            app:layout_constraintTop_toTopOf="@+id/iv_save3"
            tools:itemCount="2"
            tools:listitem="@layout/item_chip" />

    </androidx.constraintlayout.widget.ConstraintLayout>

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

```
  </androidx.core.widget.NestedScrollView>

</layout>
```

MyApp.kt

```
package com.optiontrade.br

import android.app.Application
import android.content.SharedPreferences
import com.google.firebase.analytics.FirebaseAnalytics
import com.google.firebase.ktx.Firebase
import com.google.firebase.storage.ktx.storage
import com.optiontrade.br.ent.*
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
import java.util.*
import kotlin.collections.ArrayList

class MyApp : Application() {

  companion object {
     const val tagPreferences = "OptionTradeBRSharedPreferences"
     const val keySaved = "keySaved"

     private const val firebaseRootPath = "gs://traderencyclopedia.appspot.com/"
     const val strategiesPath = "${firebaseRootPath}strategies.json"
     const val tipsPath = "${firebaseRootPath}tips.json"
     const val vocabularyPath = "${firebaseRootPath}vocabulary.json"
  }

  private lateinit var preferences: SharedPreferences
  lateinit var analytics: FirebaseAnalytics

  var strategy = ArrayList<Strategy>()
  var tips = ArrayList<Tip>()
  var vocs = ArrayList<Vocabulary>()

  lateinit var saved: Saved
```

```
override fun onCreate() {
    super.onCreate()
    analytics = FirebaseAnalytics.getInstance(this)
    preferences = getSharedPreferences(tagPreferences, MODE_PRIVATE)
    val s = preferences.getString(keySaved, null)
    saved = if (s != null)
        fromJson(s, Saved::class.java)
    else
        Saved(ArrayList(), ArrayList(), ArrayList())
    save()
}

fun save() {
    preferences.edit().putString(keySaved, toJson(saved)).apply()
}

fun loadAppContent(onLoad: (() -> Unit)){
    loadStrategies { s ->
        strategy = ArrayList(s)
        loadTips { t ->
            tips = ArrayList(t)
            loadVocabulary { v ->
                vocs = ArrayList(v)
                onLoad()
            }
        }
    }
}

private fun loadStrategies(onLoad: ((s : ArrayList<Strategy>) -> Unit)){
    Firebase.storage.getReferenceFromUrl(strategiesPath)
        .stream
        .addOnCompleteListener {
            if (it.isSuccessful) {
                CoroutineScope(Dispatchers.IO).launch {
                    val s: Scanner =
                        Scanner(it.result.stream, "UTF-8")
                            .useDelimiter("\\A")
                    val data = if (s.hasNext()) s.next() else ""
```

```
                val strategies = fromJson(
                    data,
                    Array<Strategy>::class.java
                )
                onLoad(ArrayList(strategies.toList()))
            }
        } else {
            log("Load is not successful")
        }
    }
}

private fun loadTips(onLoad: ((t : ArrayList<Tip>) -> Unit)){
    Firebase.storage.getReferenceFromUrl(tipsPath)
        .stream
        .addOnCompleteListener {
            if (it.isSuccessful) {
                CoroutineScope(Dispatchers.IO).launch {
                    val s: Scanner =
                        Scanner(it.result.stream, "UTF-8")
                            .useDelimiter("\\A")
                    val data = if (s.hasNext()) s.next() else ""
                    val tips = fromJson(
                        data,
                        Array<Tip>::class.java
                    )
                    onLoad(ArrayList(tips.toList()))
                }
            } else {
                log("Load is not successful")
            }
        }
}

private fun loadVocabulary(onLoad: ((v : ArrayList<Vocabulary>) -> Unit)){
    Firebase.storage.getReferenceFromUrl(vocabularyPath)
        .stream
        .addOnCompleteListener {
            if (it.isSuccessful) {
                CoroutineScope(Dispatchers.IO).launch {
```

```
                val s: Scanner =
                    Scanner(it.result.stream, "UTF-8")
                        .useDelimiter("\\A")
                val data = if (s.hasNext()) s.next() else ""
                val vocabularies = fromJson(
                    data,
                    Array<Vocabulary>::class.java
                )
                log(vocabularies.json())
                onLoad(ArrayList(vocabularies.toList()))
            }
        } else {
            log("Load is not successful")
        }
    }
}
```

MainActivity.kt

```
package com.optiontrade.br.ui

import android.view.View
import com.optiontrade.br.R
import com.optiontrade.br.base.BaseActivity
import com.optiontrade.br.databinding.ActivityMainBinding
import com.optiontrade.br.ui.strategy.StrategyFragment
import com.optiontrade.br.ui.tips.TipsFragment
import com.optiontrade.br.ui.voc.VocabularyFragment

class MainActivity : BaseActivity<ActivityMainBinding>(R.layout.activity_main) {

    private val strategyFragment = StrategyFragment()
    private val tipsFragment = TipsFragment()
    private val vocsFragment = VocabularyFragment()

    override fun init() {
        super.init()
        with(binding) {
```

```
        progressBar.visibility = View.VISIBLE
        app.loadAppContent {
            runOnUiThread {
                bnv.setOnItemSelectedListener {
                    when (it.itemId) {
                        R.id.strategy -> openFrag(strategyFragment, false)
                        R.id.tips -> openFrag(tipsFragment, false)
                        R.id.vocabulary -> openFrag(vocsFragment, false)
                    }
                    true
                }
                bnv.selectedItemId = R.id.strategy
                progressBar.visibility = View.GONE
            }
        }
    }
  }

}
```

OpenActivity.kt

```
package com.optiontrade.br.ui.open

import com.optiontrade.br.R
import com.optiontrade.br.base.BaseActivity
import com.optiontrade.br.databinding.ActivityMainBinding
import com.optiontrade.br.databinding.ActivityOpenBinding
import com.optiontrade.br.ent.Strategy
import com.optiontrade.br.ent.Tip
import com.optiontrade.br.ent.VocContent
import com.optiontrade.br.fromJson
import com.optiontrade.br.ui.saved.SavedFragment
import com.optiontrade.br.ui.strategy.StrategyFragment
import com.optiontrade.br.ui.tips.TipsFragment
import com.optiontrade.br.ui.voc.VocabularyFragment

class OpenActivity : BaseActivity<ActivityOpenBinding>(R.layout.activity_open) {

  companion object {
    const val actionSaved = "actionSaved"
```

```
        const val actionStrategy = "actionStrategy"
        const val actionTip = "actionTip"
        const val actionVocabulary = "actionVocabulary"

        const val keyContent = "keyContent"
    }

    var content: Any? = null

    override fun init() {
        super.init()
        with(binding) {
            when (intent.action) {
                actionSaved -> openFrag(SavedFragment(), false)
                actionStrategy -> {
                    content =
                        fromJson(
                            intent.getStringExtra(keyContent)!!, Strategy::class.java
                        )
                    openFrag(OpenFragment(), false)
                }
                actionTip -> {
                    content =
                        fromJson(
                            intent.getStringExtra(keyContent)!!, Tip::class.java
                        )
                    openFrag(OpenFragment(), false)
                }
                actionVocabulary -> {
                    content =
                        fromJson(
                            intent.getStringExtra(keyContent)!!, VocContent::class.java
                        )
                    openFrag(OpenFragment(), false)
                }
            }
        }
    }

}
```

StrategyFragment.kt

```
package com.optiontrade.br.ui.strategy

import android.annotation.SuppressLint
import android.content.Intent
import androidx.core.widget.addTextChangedListener
import com.optiontrade.br.R
import com.optiontrade.br.base.BaseFragment
import com.optiontrade.br.databinding.FragmentStrategyBinding
import com.optiontrade.br.ent.Strategy
import com.optiontrade.br.json
import com.optiontrade.br.log
import com.optiontrade.br.toJson
import com.optiontrade.br.ui.MainActivity
import com.optiontrade.br.ui.open.OpenActivity
import com.optiontrade.br.ui.strategy.adapter.StrategyAdapter
import kotlinx.coroutines.*

class StrategyFragment :
  BaseFragment<FragmentStrategyBinding, MainActivity>(R.layout.fragment_strategy) {

  private var str = ArrayList<Strategy>()

  @SuppressLint("SetTextI18n")
  override fun init() {
    with(binding) {
      str = ArrayList(app.strategy)
      rvStrategy.adapter = StrategyAdapter(str) {
        val intent = Intent(requireContext(), OpenActivity::class.java)
        intent.action = OpenActivity.actionStrategy
        intent.putExtra(OpenActivity.keyContent, toJson(it))
        startActivity(intent)
      }
      ivSave.setOnClickListener {
        val intent = Intent(requireContext(), OpenActivity::class.java)
        intent.action = OpenActivity.actionSaved
        startActivity(intent)
      }
      search()
    }
```

```
    }

    @SuppressLint("NotifyDataSetChanged")
    private fun search() {
        with(binding) {
            etSearch.addTextChangedListener { sText ->
                str.clear()
                str.addAll(app.strategy.filter { s ->
                    s.contains(sText.toString())
                })
                log("${sText.toString()} | ${str.size}")
                rvStrategy.adapter?.notifyDataSetChanged()
            }
        }
    }

}
```

TipsFragment.kt

```
package com.optiontrade.br.ui.tips

import android.annotation.SuppressLint
import android.content.Intent
import androidx.core.widget.addTextChangedListener
import com.optiontrade.br.R
import com.optiontrade.br.base.BaseFragment
import com.optiontrade.br.databinding.FragmentStrategyBinding
import com.optiontrade.br.databinding.FragmentTipsBinding
import com.optiontrade.br.ent.Strategy
import com.optiontrade.br.ent.Tip
import com.optiontrade.br.log
import com.optiontrade.br.toJson
import com.optiontrade.br.ui.MainActivity
import com.optiontrade.br.ui.open.OpenActivity
import com.optiontrade.br.ui.tips.adapter.TipAdapter
import kotlinx.coroutines.*

class TipsFragment :
    BaseFragment<FragmentTipsBinding, MainActivity>(R.layout.fragment_tips) {
```

```
    private var tips = ArrayList<Tip>()

    @SuppressLint("SetTextI18n")
    override fun init() {
        with(binding){
            tips = ArrayList(app.tips)
            rvTips.adapter = TipAdapter(tips){
                val intent = Intent(requireContext(), OpenActivity::class.java)
                intent.action = OpenActivity.actionTip
                intent.putExtra(OpenActivity.keyContent, toJson(it))
                startActivity(intent)
            }
            ivSave.setOnClickListener {
                val intent = Intent(requireContext(), OpenActivity::class.java)
                intent.action = OpenActivity.actionSaved
                startActivity(intent)
            }
            search()
        }
    }

    @SuppressLint("NotifyDataSetChanged")
    private fun search() {
        with(binding) {
            etSearch.addTextChangedListener { sText ->
                tips.clear()
                tips.addAll(app.tips.filter { t ->
                    t.contains(sText.toString())
                })
                log("${sText.toString()} | ${tips.size}")
                rvTips.adapter?.notifyDataSetChanged()
            }
        }
    }

}
```

VocabularyFragment.kt

```
package com.optiontrade.br.ui.voc
```

```
import android.annotation.SuppressLint
import android.content.Intent
import androidx.core.widget.addTextChangedListener
import com.optiontrade.br.R
import com.optiontrade.br.base.BaseFragment
import com.optiontrade.br.databinding.FragmentVocabularyBinding
import com.optiontrade.br.ent.Strategy
import com.optiontrade.br.ent.VocContent
import com.optiontrade.br.ent.Vocabulary
import com.optiontrade.br.json
import com.optiontrade.br.log
import com.optiontrade.br.toJson
import com.optiontrade.br.ui.MainActivity
import com.optiontrade.br.ui.open.OpenActivity
import com.optiontrade.br.ui.voc.adapter.VocAdapter
import kotlinx.coroutines.*

class VocabularyFragment :
  BaseFragment<FragmentVocabularyBinding, MainActivity>(R.layout.fragment_vocabulary) {

  private var vocabularies = ArrayList<Vocabulary>()

  @SuppressLint("SetTextI18n")
  override fun init() {
    with(binding) {
      vocabularies = ArrayList(app.vocs)
      rvVocabulary.adapter = VocAdapter(vocabularies) {
        val intent = Intent(requireContext(), OpenActivity::class.java)
        intent.action = OpenActivity.actionVocabulary
        intent.putExtra(OpenActivity.keyContent, toJson(it))
        startActivity(intent)
      }
      ivSave.setOnClickListener {
        val intent = Intent(requireContext(), OpenActivity::class.java)
        intent.action = OpenActivity.actionSaved
        startActivity(intent)
      }
      search()
    }
  }
```

```
@SuppressLint("NotifyDataSetChanged")
private fun search() {
    with(binding) {
        etSearch.addTextChangedListener { sText ->
            if (sText.toString().isEmpty()) {
                vocabularies.clear()
                vocabularies.addAll(app.vocs)
                rvVocabulary.adapter?.notifyDataSetChanged()
                return@addTextChangedListener
            }
            val vCons = ArrayList<VocContent>()
            app.vocs.forEach {
                vCons.addAll(it.content.filter { s ->
                    s.contains(sText.toString())
                })
            }
            vocabularies.clear()
            vocabularies.addAll(groupVocs(vCons))
            rvVocabulary.adapter?.notifyDataSetChanged()
        }
    }
}

private fun groupVocs(vocCs: ArrayList<VocContent>): ArrayList<Vocabulary> {
    if (vocCs.isEmpty()) return ArrayList()

    var l = vocCs.first().title.first()
    val vocs = ArrayList<Vocabulary>()
    var voc = Vocabulary(l.toString(), ArrayList())
    vocCs.forEachIndexed { _, v ->
        if (v.title.first() != l) {
            l = v.title.first()
            vocs.add(voc)
            voc = Vocabulary(l.toString(), ArrayList())
        }
        voc.content.add(v)
    }
    vocs.add(voc)
    log(vocs.json())
```

```
        return vocs
    }

}
```

SavedFragment.kt

```
package com.optiontrade.br.ui.saved

import android.annotation.SuppressLint
import android.content.Intent
import android.view.View.OnClickListener
import com.optiontrade.br.R
import com.optiontrade.br.base.BaseFragment
import com.optiontrade.br.databinding.FragmentSavedBinding
import com.optiontrade.br.toJson
import com.optiontrade.br.ui.open.OpenActivity
import com.optiontrade.br.ui.saved.adapter.SavedVocAdapter
import com.optiontrade.br.ui.strategy.adapter.StrategyAdapter
import com.optiontrade.br.ui.tips.adapter.TipAdapter
import kotlinx.coroutines.*

class SavedFragment :
    BaseFragment<FragmentSavedBinding, OpenActivity>(R.layout.fragment_saved) {

    @SuppressLint("SetTextI18n")
    override fun init() {
        with(binding){
            tvTitle.setOnClickListener { parentActivity.onBackPressedDispatcher.onBackPressed() }
            tvStrategy.setOnClickListener(tabs)
            tvTips.setOnClickListener(tabs)
            tvVocabulary.setOnClickListener(tabs)
            tvStrategy.callOnClick()
        }
    }

    @SuppressLint("NotifyDataSetChanged")
    override fun onResume() {
        super.onResume()
        binding.rvSaved.adapter?.notifyDataSetChanged()
    }
```

```
    private val tabs = OnClickListener{
        with(binding){
            tvStrategy.setTextColor(requireContext().getColor(R.color.secondary1))
            tvTips.setTextColor(requireContext().getColor(R.color.secondary1))
            tvVocabulary.setTextColor(requireContext().getColor(R.color.secondary1))
            when(it){
                tvStrategy -> {
                    tvStrategy.setTextColor(requireContext().getColor(R.color.accent3))
                    rvSaved.adapter = StrategyAdapter(app.saved.str){
                        val intent = Intent(requireContext(), OpenActivity::class.java)
                        intent.action = OpenActivity.actionStrategy
                        intent.putExtra(OpenActivity.keyContent, toJson(it))
                        startActivity(intent)
                    }
                }
                tvTips -> {
                    tvTips.setTextColor(requireContext().getColor(R.color.accent3))
                    rvSaved.adapter = TipAdapter(app.saved.tips){
                        val intent = Intent(requireContext(), OpenActivity::class.java)
                        intent.action = OpenActivity.actionTip
                        intent.putExtra(OpenActivity.keyContent, toJson(it))
                        startActivity(intent)
                    }
                }
                tvVocabulary -> {
                    tvVocabulary.setTextColor(requireContext().getColor(R.color.accent3))
                    rvSaved.adapter = SavedVocAdapter(app.saved.vocs){
                        val intent = Intent(requireContext(), OpenActivity::class.java)
                        intent.action = OpenActivity.actionVocabulary
                        intent.putExtra(OpenActivity.keyContent, toJson(it))
                        startActivity(intent)
                    }
                }
            }
        }
    }

}
```

OpenFragment.kt

```
package com.optiontrade.br.ui.open

import android.annotation.SuppressLint
import android.view.View
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.StaggeredGridLayoutManager
import com.optiontrade.br.*
import com.optiontrade.br.base.BaseFragment
import com.optiontrade.br.databinding.FragmentOpenBinding
import com.optiontrade.br.ent.Strategy
import com.optiontrade.br.ent.Tip
import com.optiontrade.br.ent.VocContent
import com.optiontrade.br.ui.strategy.adapter.ParamAdapter
import kotlinx.coroutines.*

class OpenFragment :
  BaseFragment<FragmentOpenBinding, OpenActivity>(R.layout.fragment_open) {

  @SuppressLint("SetTextI18n")
  override fun init() {
    with(binding) {
      tvTitle.setOnClickListener {
        parentActivity.onBackPressedDispatcher.onBackPressed()
      }
      val action = parentActivity.intent.action
      val content = parentActivity.content
      ivSave.tag = false
      when (action) {
        OpenActivity.actionStrategy -> {
          content(content as Strategy)
          Events.eventStrategyOpen(app.analytics)
        }
        OpenActivity.actionTip -> {
          content(content as Tip)
          Events.eventTipOpen(app.analytics)
        }
        OpenActivity.actionVocabulary -> {
          content(content as VocContent)
          Events.eventVocabularyOpen(app.analytics)
        }
```

```
}
if (ivSave.tag == true)
    ivSave.setImageResource(R.drawable.ic_bookmark)
ivSave.setOnClickListener {
    when (action) {
        OpenActivity.actionStrategy -> {
            if (ivSave.tag == true) {
                var i = -1
                app.saved.str.forEachIndexed { index, s ->
                    if (s.id == (content as Strategy).id)
                        i = index
                }
                log("$i")
                if (i > -1)
                    app.saved.str.removeAt(i)
                ivSave.tag = false
                ivSave.setImageResource(R.drawable.ic_bookmark_unch)
            } else {
                app.saved.str.add(0, content as Strategy)
                ivSave.tag = true
                ivSave.setImageResource(R.drawable.ic_bookmark)
                Events.eventStrategyBookmarked(app.analytics)
            }
            app.save()
        }
        OpenActivity.actionTip -> {
            if (ivSave.tag == true) {
                var i = -1
                app.saved.tips.forEachIndexed { index, t ->
                    if (t.id == (content as Tip).id)
                        i = index
                }
                if (i > -1)
                    app.saved.tips.removeAt(i)
                ivSave.tag = false
                ivSave.setImageResource(R.drawable.ic_bookmark_unch)
            } else {
                app.saved.tips.add(0, content as Tip)
                ivSave.tag = true
                ivSave.setImageResource(R.drawable.ic_bookmark)
```

```
                Events.eventTipBookmarked(app.analytics)
            }
            app.save()
        }
        OpenActivity.actionVocabulary -> {
            if (ivSave.tag == true) {
                var i = -1
                app.saved.vocs.forEachIndexed { index, v ->
                    if (v.equal((content as VocContent)))
                        i = index
                }
                if (i > -1)
                    app.saved.vocs.removeAt(i)
                ivSave.tag = false
                ivSave.setImageResource(R.drawable.ic_bookmark_unch)
            } else {
                app.saved.vocs.add(0, content as VocContent)
                ivSave.tag = true
                ivSave.setImageResource(R.drawable.ic_bookmark)
                Events.eventVocabularyBookmarked(app.analytics)
            }
            app.save()
        }
    }
  }
}

private fun content(s: Strategy) {
  with(binding) {
    tvTitle.text = getString(R.string.strategy)
    tvContentTitle.text = s.title
    tvDesc.text = s.text
    clParams.visibility = View.VISIBLE
    val c = if (s.cur.size > 2) 2 else 1
    rvCurrency.layoutManager =
        StaggeredGridLayoutManager(c, LinearLayoutManager.HORIZONTAL)
    rvCurrency.adapter = ParamAdapter(s.cur)
    rvTimeframe.layoutManager =
        LinearLayoutManager(requireContext(), LinearLayoutManager.HORIZONTAL, false)
```

```
            rvTimeframe.adapter = ParamAdapter(s.tf)
            app.saved.str.forEach {
                if (it.id == s.id)
                    ivSave.tag = true
            }
        }
    }

    private fun content(t: Tip) {
        with(binding) {
            tvTitle.text = getString(com.optiontrade.br.R.string.tips)
            tvContentTitle.text = t.title
            tvDesc.text = t.text
            clParams.visibility = View.GONE
            app.saved.tips.forEach {
                if (it.id == t.id)
                    ivSave.tag = true
            }
        }
    }

    private fun content(v: VocContent) {
        with(binding) {
            tvTitle.text = getString(com.optiontrade.br.R.string.vocabulary)
            tvContentTitle.text = v.title
            tvDesc.text = v.desc
            clParams.visibility = View.GONE
            app.saved.vocs.forEach {
                if (it.equal(v))
                    ivSave.tag = true
            }
        }
    }

}
```