Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту)

Кафедра обчислювальної техніки
(повна назва кафедри)

Пояснювальна записка

до магістерської кваліфікаційної роботи

магістр

(освітньо-кваліфікаційний рівень)

на тему: Мультимедійний програмний засіб контролю особистого бюджету для ОС Android

Виконав: студент 2 курсу, групи КІ —19м
спеціальності:
123 «Комп'ютерна інженерія»
(шифр і назва напряму підготовки)

Ящук Сергій Сергійович
(прізвище та ініціали)

Керівник: к.т.н., доц. Гарнага В.А.
(прізвище та ініціали)

м. Вінниця — 2020 рік

АНОТАЦІЯ

Дана магістерська кваліфікаційна робота присвячена розробці мультимедійного програмного засобу для контролю особистого бюджету для ОС Android. Розроблений продукт збільшує ефективність процесу ведення обліків особистих фінансів.

У даній роботі виконаний літературний огляд існуючих методів та засобів створення мобільних застосунків, обрано оптимальне програмне середовище для розробки програмної частини та користувацького інтерфейсу, розроблена структурна і функціональна частина додатку. Спроектовано інтуїтивно-зрозумілий UI/UX інтерфейс. Проведено аналіз та порівняння вже існуючих на ринку аналогів.

Результатом роботи є розроблений програмний застосунок, що наділений функціоналом для забезпечення простого та ефективного контрою особистого бюджету.

Перевірка працездатності здійснена шляхом практичних випробувань.

ANNOTATION

This master's thesis is devoted to the development of a multimedia software for personal budget control for Android. The developed product increases the efficiency of the process of accounting for personal finances.

This paper provides a literature review of existing methods and tools for creating mobile applications, selected the optimal software environment for software development and user interface, developed the structural and functional part of the application. An intuitive UI / UX interface has been designed. The analysis and comparison of already existing analogues on the market is carried out.

The result is a developed software application that is endowed with functionality to provide simple and effective control of personal budget.

Performance testing is carried out through practical tests.

ЗМІСТ

ВСТУП

Популярність використання мобільних пристроїв у всьому світі продовжує зростати. Сьогодні користувачі витрачають більше часу на свої смартфони в різних цілях (соціальні мережі, електронна пошта, карти, новини, відео, комерційні додатки та інші). У таких умовах господарювання вимагає від фахівців з економічного управління всебічного використання новітніх інформаційних технологій. Широкі можливості мобільних засобів в питаннях збору, обробки та видачі необхідної інформації здатні значно підвищити якість економічних розрахунків, зробити більш ефективним процес обґрунтування економічних рішень.

Таким чином, процес розробки мобільних додатків стає актуальним напрямком у IT-індустрії [1]. Сучасні компанії, такі, як Google, Apple, Microsoft та інші, розробили мобільні платформи, що включають мобільні операційні системи та засоби розробки (SDK, Software Developer Kit). Важливою особливістю мобільних пристроїв є те, що вони мають обмежене джерело живлення, невеликій розмір екрана та набір різноманітних сенсорів. Процес розробки мобільних додатків є достатньо технологічним і потребує певних компетенцій з об’єктноорієнтованого програмування, проектування баз даних та UI/UX, розуміння мережевої взаємодії, тестування програмного забезпечення.

Найбільш росповсюдженою мобільною платформою є Android. Вона складає понад 82 % від усіх засобів на 2020 р., що підтверджує її універсальніть і перспективність для подальшого вивчення.

Використання мобільних застосунків дозволяє ефективно організувати процес контролю особистого бюджету, мати безпечний доступ до банківських рахунків , що є вкрай важливим для людей з обмеженим бюджетом. Отже, розробка даного додатку є **актуальною**.

Метою роботи є покращення якості ведення обліку фінансів за рахунок реалізації у мобільному додатку інструментів для ведення та аналізу особистого бюджету.

Задачі дослідження магістерської роботи:

1) аналіз існуючих аналогів, та засобів розробки мобільних застосунків для ОС Android.

2) розробка користувацького інтерфейсу мобільного мультимедійного додатку.

3) розробка бази даних, та структури застосунку.

4) програмна реалізація мультимедійного програмного застосунку для контролю особистого бюджету.

5) тестування розробленого програмного забезпечення.

Об’єкт дослідження магістерської кваліфікаційної роботи — процес розробки та оптимізації мультимедійного додатку під OS Android.

Предмет дослідження магістерської роботи — методи та засоби реалізації мобільних застосунків.

Методи дослідження магістерської роботи: технології для розробки мобільного програмного забезпечення під ОС Android, та методи роботи з базою даних Firebase Realtime. У роботі використано принципи об’єктно-орієнтованого програмування для реалізації запропонованого підходу.

Наукова новизна отриманих результатів магістерської роботи полягає у реалізації подальшого розвитку мобільних застосунків для контролю особистого, який відрізняється від існуючих, використано сучасної бази данних в режимі реального часу Firebase, якій не потрібно мати власний сервер адже вона працює на серверах Google, що дозволяє підвищити продуктивність програми. Проаналізовано недоліки аналогів на ринку, та впроваджено їх вирішення у кінцевому програмному засобі.

Практичне значення одержаних результатів магістерської роботи: розроблений мультимедійний програмний застосунок орієнтований на використання мобільних пристроїв під ОС Android. Розробка призначена для спрощення процесу контролю бюджету, фіксації витрат, доходів та транзакцій, встановлення обмежень на певні періоди часу (місяць, рік, тощо).

Апробація результатів дослідження була здійснена у процесі подання тези доповіді, яка була опублікована:

Ящук С. С., «Мультимедійний програмний засіб контролю особистого бюджету для ОС Android» в Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2020)», Вінниця, 2020. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2021/schedConf/presentations. Дата звернення: Листопад 2020.

1 АНАЛІЗ МЕТОДІВ І ТЕХНОЛОГІЙ РОЗРОБКИ ПРОГРАМНИХ ЗАСОБІВ ДЛЯ ОС ANDROID

1.1 Обґрунтування ролі та важливості мобільних програмних засобів для контролю особистого бюджету.

Ведення обліку особистих коштів є важливою частиною життя у людей з обмеженим бюджетом. Планування та контроль бюджету полягає у відслідковуванні щоденних витрат та доходів, аналізі необхідних та зайвих покупок, планування заощаджень ціль яких — забезпечення фінансової допомоги у випадках непередбачуваних ситуацій.

У процесі ведення особистого бюджету виникають труднощі, наприклад:

— не кожна операція із витратою або отриманням грошей записується, що дає велику похибку при кінцевому підрахунку фінансів;
— економія на певних категоріях товарів або послуг потребує числової та графічної статистики;
— формування заощаджень вимагають деякі обчислення, на основі даних поточного фінансового стану, та статистику засновану на попередніх витратах та доходах.

Програмний засіб для контролю бюджету вирішує ці проблеми, так як:

— вся інформація знаходиться в смартфоні користувача;
— зберігання даних про бюджет здійснюється через хмарну базу даних у реальному часі;
— кожен користувач має особистий обліковий запис для доступу до своїх даних;
— графічний інтерфейс зручно та лаконічно зображує усі операції та статистику;
— користувачу потрібно лише вводити дані про витрати та доходи, усі необхідні обрахунки виконує додаток.

За допомогою мобільного додатку користувач впорається з підрахунком власних коштів набагато якісніше і легше, за рахунок швидкого виконання арифметичних операцій.

1.2 Методи, етапи розробки та типи мобільних програмних застосунків

Розробка мобільних застосунків для планшетів, смартфонів та інших мобільних пристроїв це процес написання програмного коду кінцевим результатом якого є створення програм, що будуть працювати на конкретних мобільних платформах (Android, iOS, Windows Phone тощо).

Розроблене програмне забезпечення та додатки можуть встановлюватися користувачами на пристрій в процесі його використання, або вони можуть бути попередньо завантажені на будь який мобільний пристрій (смартфони, планшети тощо).

Для того щоб почати проектування мобільного додатку, потрібно з’ясувати певні речі:

— пристрої, для яких призначений додаток;

— середовище розробки застосунку;

— платформа експлуатування додатку;

— методи які доцільно використовувати при проектуванні застосунку.

Форм-фактор пристроїв під які пишуться мобільні програми, це головна особливість розробки цих програм, тому що ці девайси це — сматфони та планшети, які мають невеликий дисплей. Наприклад, якщо взяти смарт-годинник, він має екран розміром 2x2 сантиметри, якщо екран прямокутний, в інших випадках екран круглий. У зв’язку з цим дуже важливим моментом розкробки мобільного застосунку стає UX (User Experiense) — простими словами це зручність, логічність та простота використання користувацького інтерфейсу, він включає в себе різні компоненти: інформаційну архітектуру, проектування взаємодій, графічний дизайн та контент.

Процес розробки мобільного програмного забезпечення складається з певних етапів:

— вибір основної концепції та тематики додатку, проводиться аналіз уже існуючих аналогових додатків на ринку;

— визначення базового функціоналу, у більшості випадків мобільні додатки починають свій випуск з демо-версії (версія з базовим функціоналом),

зібравши відгуки користувачів, розробник розуміє на що приділити більше уваги в подальшій розробці;

— позначення базової моделі поширення, адже від цього також залежать основні сторінки самого додатка;

— розробка UI/UX (користувацького інтерфейсу), на цьому етапі окремий розробник починає працювати з конкретним функціоналом, знаючи яким чином клієнт буде користуватись додатком, створюється інформаційна система та навігаційна структура програми;

— безпосередньо проектування мобільного застосунку, та його допоміжних сервісів, у процесі створення додатку, він повинен постійно тестуватись та покращуватись, адже часто в процесі розробки приходять нові ідеї на рахунок додаткового функціоналу та дизайну які можуть вплинути на тривалість часу, необхідного на розробку;

— наступний етап — це тестування функціональності застосунку, вирішуються всі помилки та недоліки в роботі додатку;

— якщо тестування додатку дає позитивний результат, то додаток завантажують у магазин мобільних застосунків.

Ще один важливий пункт в розробці мобільного програмного забезпечення це великий спектр пристроїв під які ми бажаємо створити додаток. Створений додаток може розраховуватись тільки на конкретний вид пристрою (тільки на смартфон або тільки на планшет, тощо).

Є два методи створення мобільних програмних застосунків [2]:

— нативна розробка;

— кросплатформена (або гібридна).

Нативна розробка — процес створення мобільного програмного забезпечення для конкретної платформи, на конкретній мові. Додатки створені цим методом не мають обмежень в розробці, а їх продуктивність на досить високому рівні (для iOS — Swift, для Android — Java, Kotlin). Плюсом такого методу є відсутність застримки на виконання дії користувача, клієнт має прямий доступ до апаратної частини, для

користувача певної платформи можна розробити найбільш звичний для нього інтерфейс. Висока ціна розробки і підтримки, та тривалий час виділений на створення застосунку даним методом є його недоліком.

Кросплатформена розробка — відбувається за допомогою web-технологій — HTML, CSS і JavaScript, які дають можливість розробити застосунок на кілька платформ відразу. Мобільний застосунок потрібно перевести на зрозумілу мову для тієї платформи на якій він має експлуатуватись. Низька вартість розробки та не тривалий час для її виконання дозволяють іноді задіювати лише одного фахівця, що є великою перевагою. А от його мінусом є затримка реагування функціоналу додатку на дії користувача, можливі виникнення труднощів у роботі всіх функцій. Інтерфейс буде вимагати додаткового опрацювання, адже він буде занадто простим.

Мобільні застосунки є трьох типів (рисунок 1.1) [3].

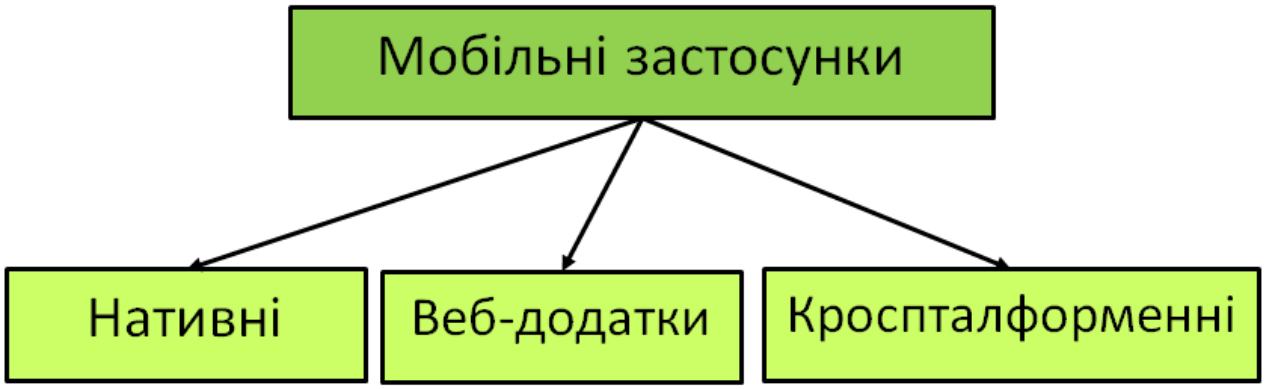


Рисунок 1.1 — Типи мобільних додатків

Застосунок розроблений нативним методом — це додаток, що розроблений на рідній мові програмування для певної платформи (Java, Kotlin для Android, C# для Windows Phone). Платформа для, якої розробляється додаток, дає розробнику певний набір інструментів (SDK — software development kit), він дає можливість отримати доступ практично до всіх сервісів пристрою, та також призначений для спрощення процесу створення мобільних програмних засобів.

Веб-додаток — це програмний засіб, розроблений за допомогою HTML, JavaScript, CSS. Для свого виконання він потребує встановленого і налаштованого браузера з виходом в Інтернет. В даного застосунку клієнтом виступає браузер, а сервером — вебсервер. Основна функція браузера — відображення інформації, що завантажена мережею з сервера, та передача даних назад користувачу. Основним плюсом такого типу додатків є те що вони не залежать від операційної системи пристрою користувача, тому вони є кросплатформеними сервісами.

Кожен тип мобільних програмних застосунків має свої недоліки та переваги один над одним, їх порівняльна характеристика наведено у таблиці 1.1.

Таблиця 1.1 — Порівняльна характеристика типів мобільних додатків

	Нативні	Кросплатформені	Веб-додатки
Встановлення	потребує	потребує	не потребує
Доступ до Інтернету	не обов'язковий	обов'язковий	обов'язковий
Оновлення	Повторне розміщення	Повторне розміщення	Не потрібне
Платформи	Розробка під кожну	кросплатформені	кросплатформені
Доступ до ресурсів	повний	обмежений	відсутній
Публікація	В магазинах застосунків	В магазинах додатків	Не потребує

Кросплатформений (гібридний) додаток — це додаток, який використовує частково функціональні можливості як нативних так і веб-додатків можливість часткового доступу до ресурсів пристрою — від нативних; підтримка HTML і робота в браузері — від веб-застосунків). Тому деякі гібридні додатки можуть частково функціонувати без доступу

до інтернету. З точки зору маркетингу, ці додатки є вигіднішими для розробки, так як можуть публікуватись у магазинах додатків різних операційних систем (Google Play, App Store). Будь-які оновлення випущені для додатку будуть впроваджуватись одразу на всіх операційних системах.

1.3 Варіантний вибір засобів для розробки мультимедійного додатку під ОС Android

Для створення додатку під ОС Android потрібно обрати інтегроване середовище розробки, яке буде підтримувати мови Java, Kotlin, C++ (мови які використовуються для створення програм під ОС Android). Таких середовищ є досить багато, одними з найоптимальнішими є Eclipse та Android Studio [4].

1.3.1 Огляд інтегрованих середовищ розробки

Eclipse — це інтегроване середовище для проектування мобільного програмного забезпечення (рисунок 1.2).

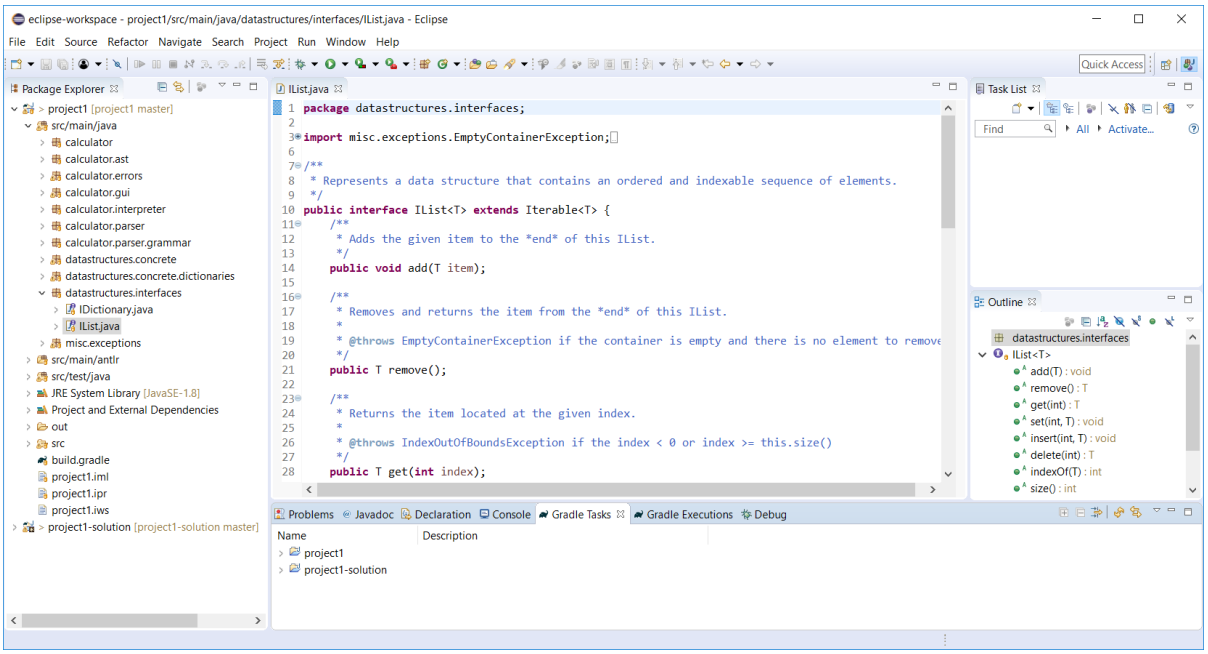


Рисунок 1.2 — Інтерфейс Eclipse

Дане середовище являє собою фреймворк для створення кросплатформових мобільних додатків, воно має свої певні особливості:

— програмне забезпечення можна розробляти на багатьох мовах програмування;

— це інтегроване середовище розробки є гібридним (кросплатформенним);

— сирцевий код є відкритим.

Середовище розробки це повноцінна Java IDE, націлена на групову розробку, а беручи до уваги її наявність у безкоштовному доступі, багато організацій використовує її як стандарт для розробки програмного забезпечення на Java.

Eclipse — це універсальне середовище розробки, що підтримує багато мов програмування. Для опису інтерфейсу додатків використовується ще одна мова — XML (розширювана мова розмітки).

Це середовище розробки також служить платформою для розробки нових розширень, тобто будь-який розробник може розширити середовище написавши якись модуль. Завдяки цьому можлива розробка на інших мовах програмування, (C/C, Python, PHP, та інші). Оскільки Eclipse написана на Java, то вона є не залежною від платформи. Відомі компанії такі як: IBM, Nokia, Borland та інші, інвестують кошти у розвиток даного середовища розробки.

Проте не зважаючи на всі плюси Eclipse, дане середовище вже є досить застарілим, оптимальною альтернативою стала Android Studio.

Android Studio — середовище розробки мобільних програм для Android від компанії Google на основі IntelliJ IDEA (рисунок 1.3). Воно є оптимальною заміною плагіну ADT для платформи Eclipse, та побудоване на базі вихідного коду IntelliJ IDEA Community Edition [5].

Дане середовище дає можливість створювати мобільне програмне забезпечення не тільки для смартфонів і планшетів, але і для телевізорів (Android TV), окулярів Google Glass і автомобільних систем (Android Auto). Android Studio надає інстумент, щоб автоматично імпортувати існуючий проект який був розроблений на Eclipse і ADT плагіні.

Це інтегроване середовище розробки дає на вибір два стилі інтерфейсу, темний та світлий.

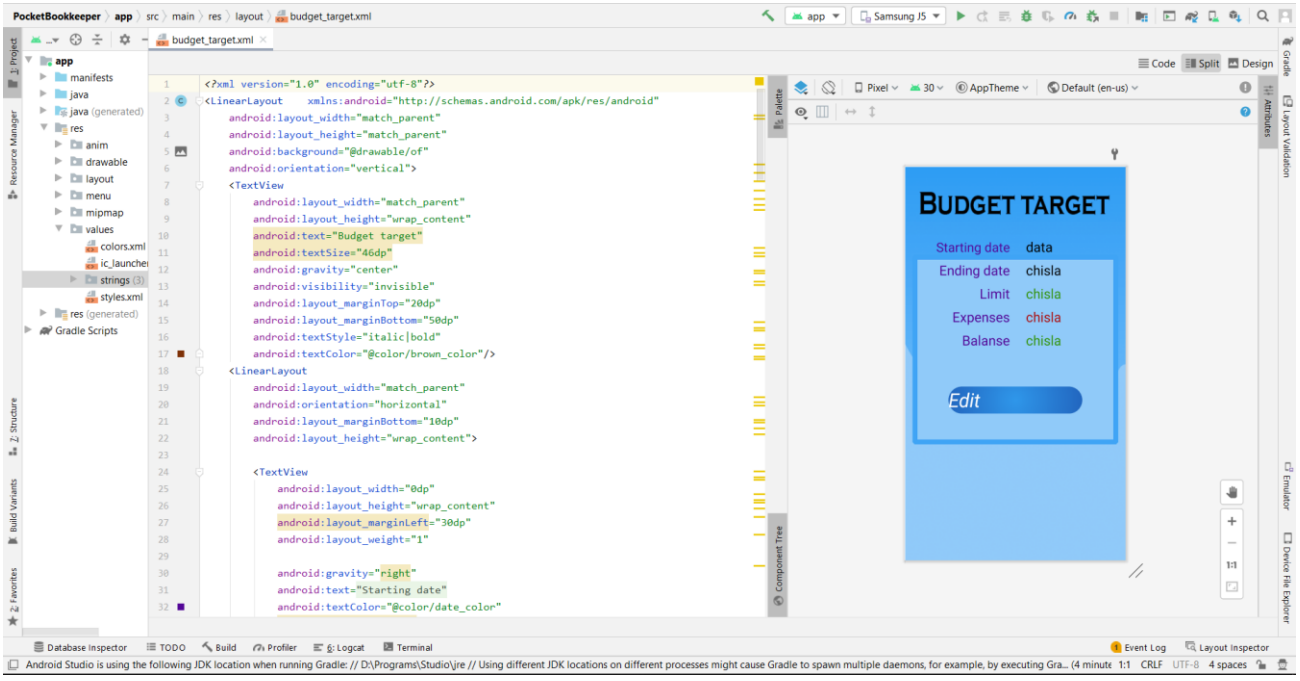


Рисунок 1.3 — Інтерфейс Android Studio

Дане середовище адаптоване для виконання типових завдань, що вирішуються в процесі розробки застосунків для платформи Android. Включені засоби які дозволяють спростити тестування програм на сумісність з різними версіями платформи. Для проектування застосунків що мають працювати на пристроях з різною роздільною здатністю екрана, інтегровано окремі інструменти. В Android Studio реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування і розгортання застосунків, заснована на складальному інструментарії Gradle і підтримуюча використання засобів безперервної інтеграції.

Доступна вибірка основних та базових елементів інтерфейсу і візуальний редактор для їхнього компонування, що надає зручний попередній перегляд різних станів інтерфейсу застосунку, що зменшує час витрачений на розробку додатку. Проте, можна створити нестандартний інтерфейс, для цього присутній майстер створення власних елементів. Типові приклади коду можна завантажувати з GitHub [6].

Присутні інструменти для:

— перевірки сумісності з минулими випусками;

— виявлення проблем з продуктивністю;

— моніторингу споживання пам'яті;

— оцінки зручності використання;

— розширені модулі рефракторингу.

Android API підтримує систему підсвічування, для статичного аналізу і виявлення помилок, це по суті є режимом швидкого внесення правок. Окрім цього наявний ProGuard (оптимізатор коду). Додано можливість генерувати цифровий підпис, та інтерфейс для керування перекладами на інші мови.

Android Studio надає емулятор — це віртуальний пристрій який являє собою аналог реального смартфону чи планшету з дещо обмеженим функціоналом. Його призначення — тестування розроблених мобільних програмних забезпечень.

1.3.2 Аналіз мови програмування

Для написання програми Android Studio дає на вибір дві мови програмування а саме Kotlin та Java. Найчастіше використовують саме мову Java, так як вона полегшує роботу розробникам завдяки своїм перевагам, а саме:

— вона є незалежною від платформи на якій пишуться програми, тобто один код можна запускати на Windows, Linux та інших;

— її синтаксис дуже подібний до мови C/C++;

— мова є майже повністю об’єктно-орієнтованою, всі сутності Java — це об’єкти;

— мова містить засоби виділяти найбільш поширені помилки, що робить її надійною;

— вона містить механізм збірки “сміття”, тобто пам'ять в ній звільняється та виділяється автоматично, що страхує розробника від помилок з неправильним використанням пам’яті;

— оператор присвоювання не буде переплутуватись з оператором порівнянням на вірність;

— множинне наслідування — відсутнє, замість нього є нове поняття — інтерфейс, він надає розробнику майже все, що можна отримати від множинного наслідування, при цьому унакаючи складнощів.

Незалежність від платформи на якій пишуться програми означає, що програма, написана на мові Java, буде функціонувати на будь-якій платформі без змін у початковому коді [7].

Оскільки, Java є повністю об’єктно-орієнтованою мовою, то всі її дані та дії групуються в класи об'єктів, окрім виключення примітивних типів як int чи float. Дане рішення було прийняте для збільшення швидкості виконання поставлених задач.

В цій мові всі об'єкти є похідними від головного об'єкта з якого вони успадковують базову поведінку і властивості.

На відмінну від C++, у Java можливе тільки одинарне успадкування, завдяки чому відпадає можливість конфліктів між членами класу, що успадковуються від базових класів.

Беручи до уваги всі переваги цієї мови, вирішено виконувати розробку мобільного програмного застосунку саме на ній.

1.3.3 Огляд організації бази даних

На сьогодні день жодна інформаційна технологія не обходиться без баз даних. Вони дозволяють ефективно упорядковувати та розміщувати дані, що спрощує їх пошук для подальшої взаємодії.

В розробленому мультимедійному програмному застосунку потрібно ефективно зберігати та взаємодіяти з інформацією про особистий бюджет, тому у ньому буде застосовано концепцію баз даних.

База даних — це сукупність даних, які організовані відповідно до концепції, яка описує їх характеристику і взаємозв’язки між їхніми елементами. Базою даних можна вважати будь-який упорядкований набір даних. У сучасних системах для забезпечення роботи з БД використовують СКБД (системи керування баз даних). Це системи, які забезпечують визначення, створення, контроль, використання та

маніпулювання базами даних. Вони дозволяють ефективно працювати з БД, так як їх обсяг не можливо опрацювати вручну.

Для вирішення задач мобільного додатку для керування особистого бюджету, найбільш доцільно використовувати базу даних реального часу. Такою базою даних є Firebase Realtime (рисунок 1.4).

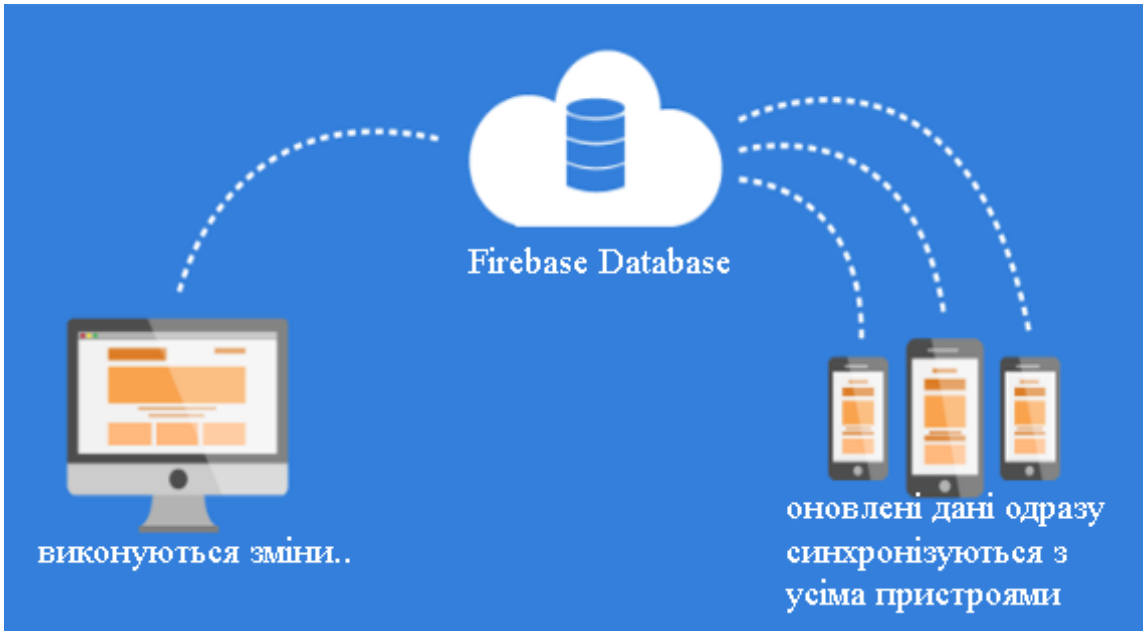


Рисунок 1.4 — Зберігання та синхронізація даних у Firebase Realtime

Firebase Realtime Database — це хмарна NoSQL база даних. В ній дані зберігаються у форматі JSON та синхронізуються у режимі реального часу з кожним підключеним користувачем [8].

Підключені особливості інтеграції з додатків під операційні системи Android та iOS, також реалізовані API для застосунків на Java, JavaScript, Node.js. Дана база даних передбачає API для шифрування інформації.

При створення гібридних (кросплатформенних) додатків, за допомогою SDK для iOS, Android всі клієнти використовують один екземпляр бази даних в реальному часі, та автоматично отримують оновлення з найновішими даними.

Замість типових HTTP-запитів, дана база даних використовує синхронізацію даних, щоразу коли дані міняються, тому будь-який під'єднаний пристрій отримає це оновлення одразу ж. Доступ до Firebase Realtime можна

отримати з мобільного приладу або веб-браузера. Є можливість оптимізувати аутентифікацію за допомогою Firebase Authentication.

База даних Firebase Realtime дозволяє створювати багатофункціональні програми для спільної роботи, забезпечуючи безпечний доступ до бази даних. Дані зберігаються локально, і навіть в автономному режимі події в реальному часі продовжують спрацьовувати, надаючи користувачу повноцінний відгук. Коли пристрій відновлює з'єднання, база даних реального часу синхронізує локальні зміни даних з віддаленими оновленнями, які відбулися, коли клієнт знаходився в автономному режимі, автоматично об'єднуючи будь-які зміни.

База даних реального часу надає гнучкий мову правил на основі виразів, званий правилами безпеки баз даних в реальному часі Firebase, для визначення того, як ваші дані повинні бути структуровані і коли дані можуть зчитуватися або записуватися. При інтеграції з аутентифікації Firebase розробники можуть визначити, хто має доступ до яких даними і як вони можуть отримати до них доступ.

База даних реального часу є базою даних NoSQL і, тому має інші види оптимізації і функціональність у порівнянні з реляційною базою даних. API-інтерфейс бази даних реального часу дозволяє виконувати тільки ті операції, які можуть бути виконані швидко. Це дозволяє вам проводити обробку даних в реальному часі, обслуговуючи мільйони користувачів без шкоди для швидкості відгуку. У зв'язку з цим важливо продумати те, як користувачі повинні отримувати доступ до ваших даних, і потім відповідним чином структурувати їх.

1.4 Аналіз існуючих аналогів

Мобільних додатків для ведення обліку особистих фінансів існує досить багато, але вони не завжди повною мірою надають функції, потрібні користувачеві. Деякі з них мають багато зайвого функціоналу, який лише займає більше пам’яті пристрою, а деякі навпаки мають не доситатню кількість функцій, що обмежує користувача у якості ведення обліку особистих фінансів. Для

розробки оптимального варіанту додатку, проведено аналіз недоліків та переваг вже існуючих аналогів.

Одним із найвідоміших додатків у данній категорії є Moneyfy (рисунок 1.5). Щоб почати роботу, не потрібно реєструватися, що є мінусом, аже відсутність авторизації означає, що ваші дані може переглянути будь-хто, кому потрапить у руки ваш смартфон. Кожен календарний день в даному додатку являє собою сторінку з фінансовими операціями. Цей програмний засіб має лише базові функції, а саме доходи, витрати, бюджет і статистика. Додаток доступний у безкоштовному доступі, проте є і платна підписка, яка надає певний перелік додаткових функцій. Данні функції ніяк не покращують якість ведення особистого бюджету, а лише надають змогу користувачу змінювати інтерфейс. До недоліків можна віднести відсутність деяких додаткових функцій, таких як планування покупок, сповіщення при перевищенні ліміту витрат на ту чи іншу категорію товарів та послуг. Доходи та витрати зображені у вигляді кольорових діаграм, що візуально дає зрозуміти яка категорія товарів чи послуг є найбільш затратною.

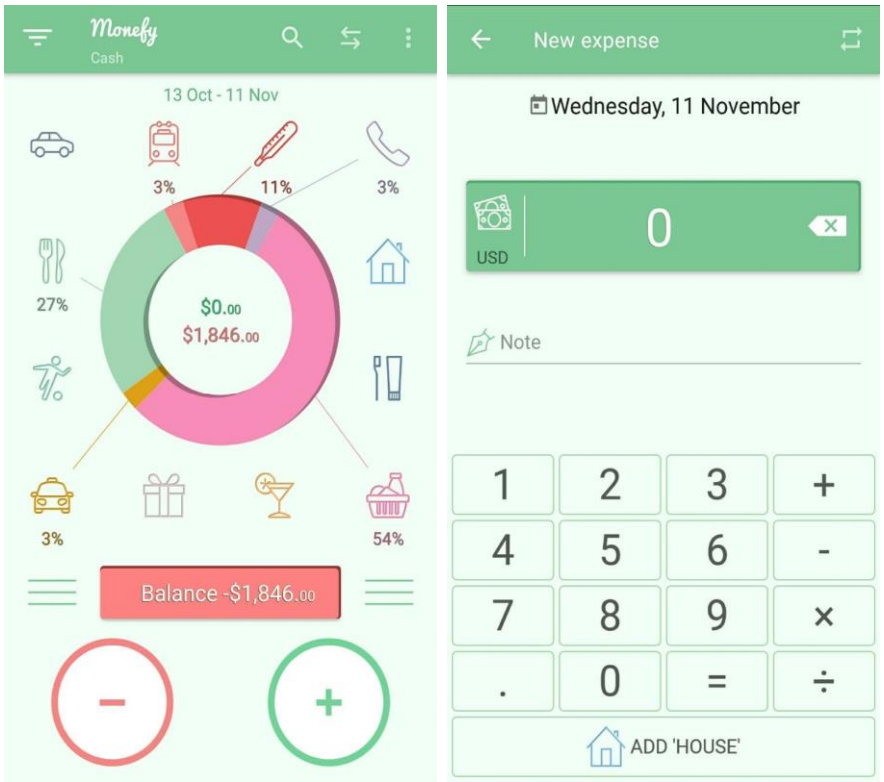


Рисунок 1.5 — Додаток Moneyfy

Наступним аналогом є не менш відомий додаток Money Manager Expense&Budget (рисунок 1.6). Даний додаток має досить великий спектр функціоналу. Витрати та доходи зображені у вигляді кругової діаграми. Цей додаток буде зручним для тих кому потрібен поглиблений статистичний аналіз. Є можливість створення плану заощаджень, тобто ті кошти які плануються заощадитись не будуть відображатись в загальному балансі, що доволяє не розраховувати на них у найближчий час. Також можна встановити обмеження витрат на певну категорію товарів чи послуг.

Проте для більшості користувачів він буде мати багато зайвих функцій які будуть заважати. Даний мобільний засіб має не найкращий інтерфейс більшість складових якого є списковими екранами. Найбільшим недоліком даного аналога є його платна підписка яка відкриває весь необхідний список функцій для користувача, немає можливості купити лише необхідний функціонал, або купуєш все, або нічого.

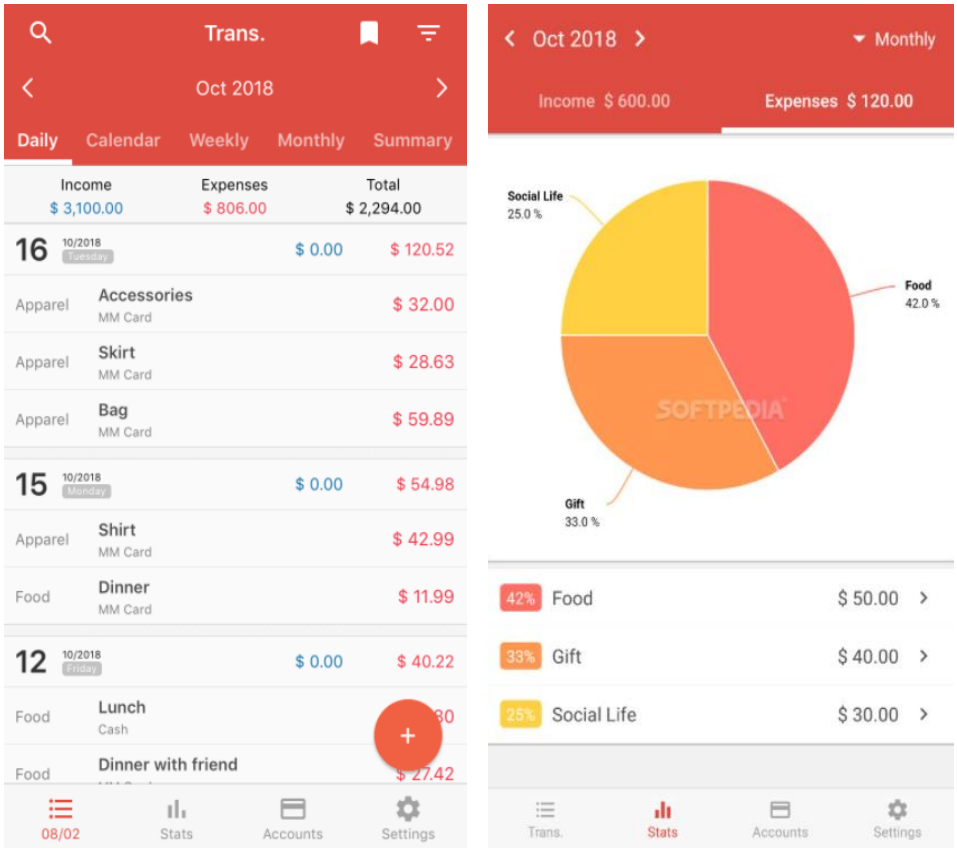


Рисунок 1.6 — Додаток Money Manager Expense&Budget

Додаток Wallet: Personal Finance, Budget&Expense Tracker розроблений для Android та Windows Phone (рисунок 1.7). Відмінність цього екземпляру від попередніх прикладів є наявність можливості ведення обліку фінансів в декількох валютах одночасно. Дана функція буде досить корисною для користувачів які багато подорожують за кордоном, так як не потрібно конвертувати валюту щоразу як змінюється геолокація. Досить корисним функціоналом є присутність можливості налаштування постійних витрат (комунальні послуги, інтернет, тощо). Окрім звичайної авторизації, вхід у програму можна захистити пін-кодом, що надасть додатковий захист вашим фінансовим записам, на випадок якщо ваш мобільний пристрій загубиться. Wallet має приємну графіку користувацького інтерфейсу, та радує простотою його використання, все інтуітивно зрозуміло.

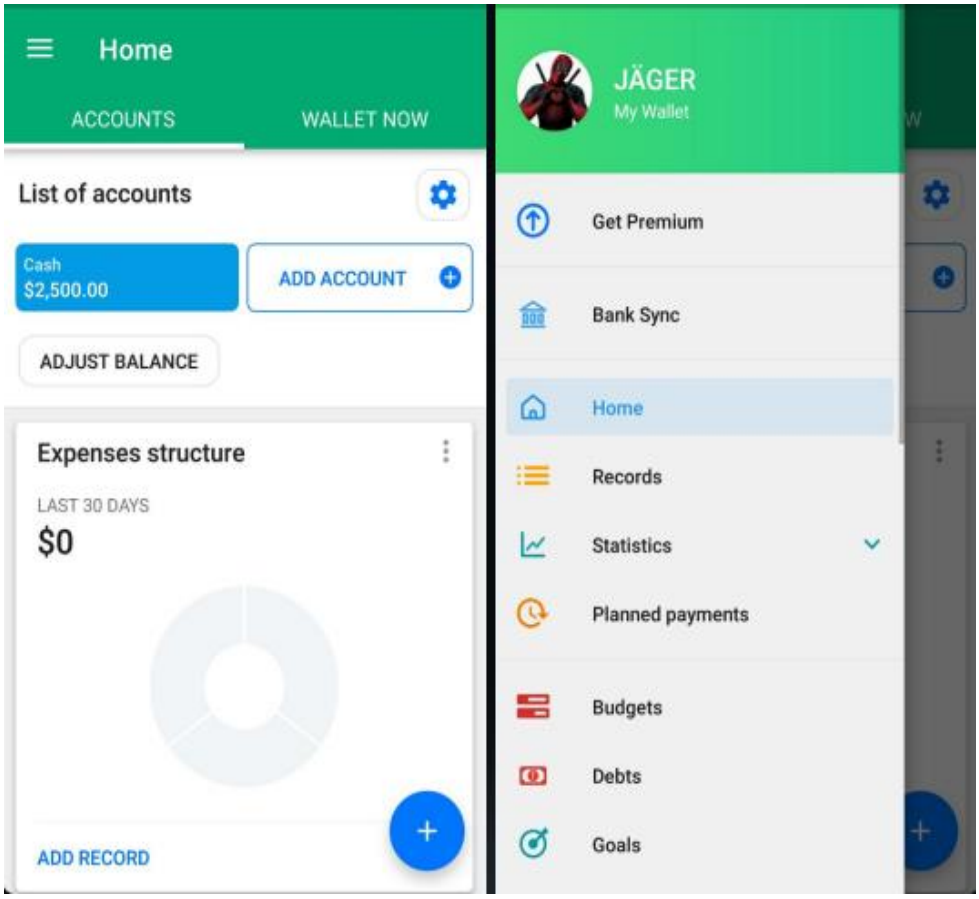


Рисунок 1.7 — Додаток Wallet: Personal Finance, Budget&Expense Tracker

Звичайно, кожен з наведених прикладів має свої відмінності від своїх аналогів, в чомусь вони виграють у інших, в чомусь програють. Для того щоб

зрозуміти яким чином нова розробка може обійти аналогові застосунки на ринку, потрібно реалізувати виправлення їх недоліків у своїй розробці, та додати вже готові переваги аналогів. Провівши огляд передових аналогових мобільних додатків, можна проаналізувати їхні недоліки та переваги один над одним, та на основі цієї інформації зробити висновки щодо концепцій які необхідно включити або не задіювати при виконанні задачі даної магістерської кваліфікаційної роботи. Порівняльна характеристика переваг та недоліків аналогових мобільних застосунків наведена у таблиці 1.2. На основі даних цієї таблиці можна впроваджувати конкретні пункти у проектований додаток.

Таблиця 1.2 — Переваги та недоліки альтернативних додатків

Назва додатку	Переваги	Недоліки
Moneyfy	— Витрати і доходи у вигляді кругової діаграми; — Безкоштовний доступ; — Зручний інтерфейс;	— Малий спектр функціональних можливостей; — Відсутня ПК версія;
Money Manager	— План заощаджень; — Функція обмеження витрат на категорію товарів чи послуг;	— Велика кількість зайвих функцій ; — Весь функціонал доступний лише у платній версії додатку;
Wallet	— Ведення обліку в декількох валютах; — Захист пін-кодом;	— Відсутня можливість планування заощаджень; — Відсутня ПК версія;

Отже, було обгрунтовано роль та важливість мультимедійних програмних застосункуів для ведення обліку особистих фінансів. Проведено огляд осонових етапів та методів розробки мобільного програмного забезпечення Здійснено аналіз існуючих аналогів з проведенням їх порівняння,та на основі цього зроблено варіантний вибір інструментів розробки.

2 ПРОГРАМНА РЕАЛІЗАЦІЯ МУЛЬТИМЕДІЙНОГО ПРОГРАМНОГО ЗАСОБУ КОНТРОЛЮ ОСОБИСТОГО БЮДЖЕТУ

Програмним середовищем для розробки мобільного програмного засобу було обрано інтегроване середовище розробки Android Sudio [9]. Щоб створити проект, потрібно задати його характеристики, а саме:

1) ім’я проекту;
2) унікальне ім’я пакету (яке необхідне для публікації проекту у магазинах додатків;
3) мова, на якій буде писатись код;
4) мінімальний рівень API;
5) тип головної Activity.

Зазначення вищевказаних пунктів зображено на рисунку 2.1.

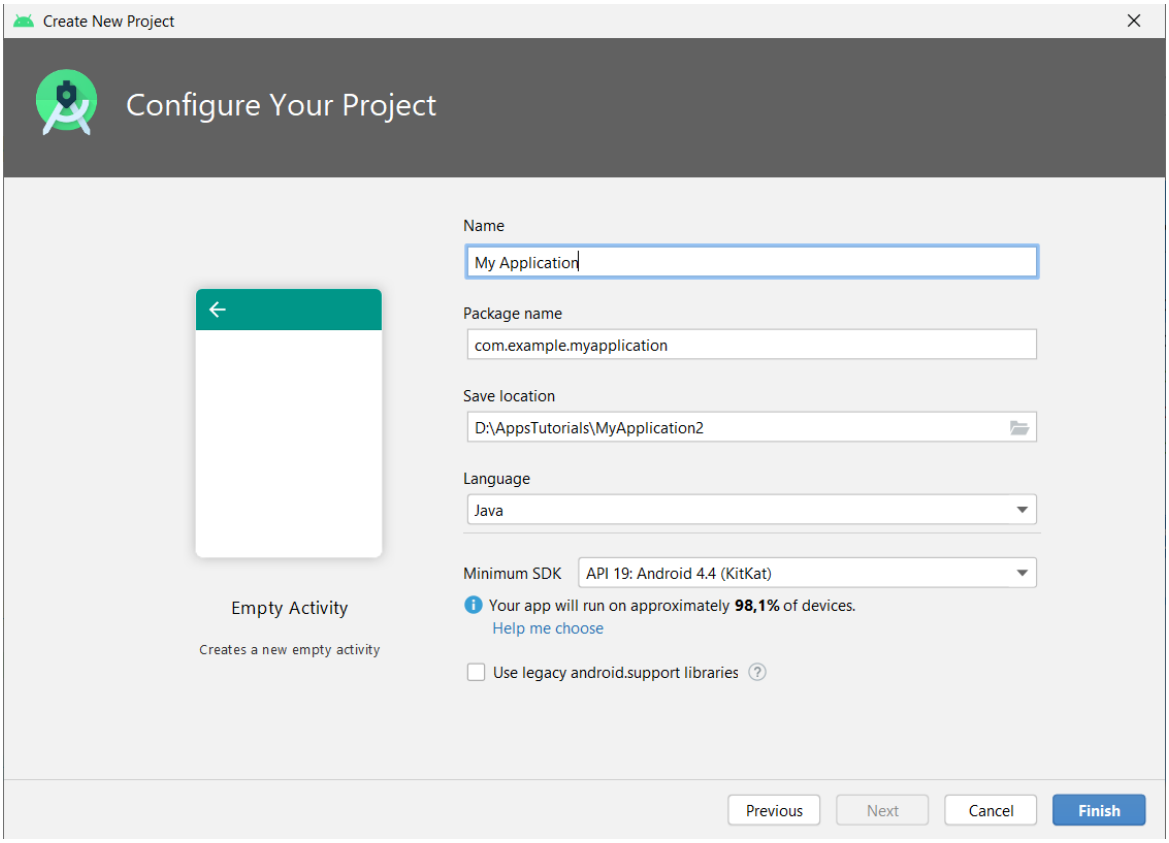


Рисунок 2.1 — Початкове налаштування проекту

Створення програм в даному середовищі передбачає роботу з так званими Activity [10]. Це ключовий компонент для створення візуального інтерфейсу в

мобільних застосунках. Зазвичай Activity асоціюється з окремим екраном або вікном додатку, а переключення між цими вікнами буде відбуватись як перехід від одної Activity до іншої. Застосунок може мати одну або декілька активностей. Усі її об’єкти являють собою об’єкти класу android.app.Activity, який містить базовий функціонал для всіх активностей.

Кожен додаток Android мають свій життєвий цикл (рисунок 2.2). Коли користувач запускає додаток, система дає йому високий пріоритет.

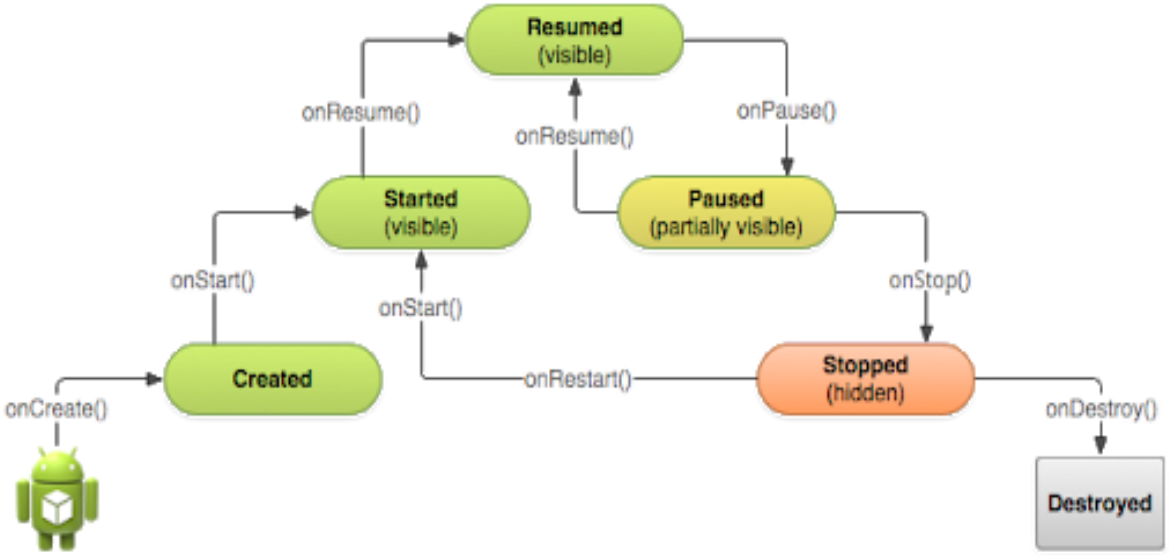


Рисунок 2.2 — Життєвий цикл Activity

2.1 Реалізація вікон авторизації та реєстрації

Після того як пустий проект створено, за замовчуванням з’являються файли activity_main.xml та MainActivity.js. Файли формату xml містять у собі код який відповідає за розміщення віджетів інтерфейсу на екрані, а файл формату js відповідає за функціонал цих елементів.

Перед тим як описувати функціонал, необхідно розробити алгоритм, за яким буде працювати авторизація у додаток. Авторизація — це надання доступу та певних повноважень користувачу на виконання деяких дій у системі обробки данних Має бути присутня перевірка на наявність акаунту, якщо його немає, то у алгоритм має бути включена послідовність операцій для проходження реєстрації нового акаунта для користувача. Якщо користувач уже убв зареєстрований, то

повинна бути присутня можливість повернутись з вікна реєстрації, до вікна авторзації. Розроблений алгоритм зображено нижче на рисунку 2.3 [11].

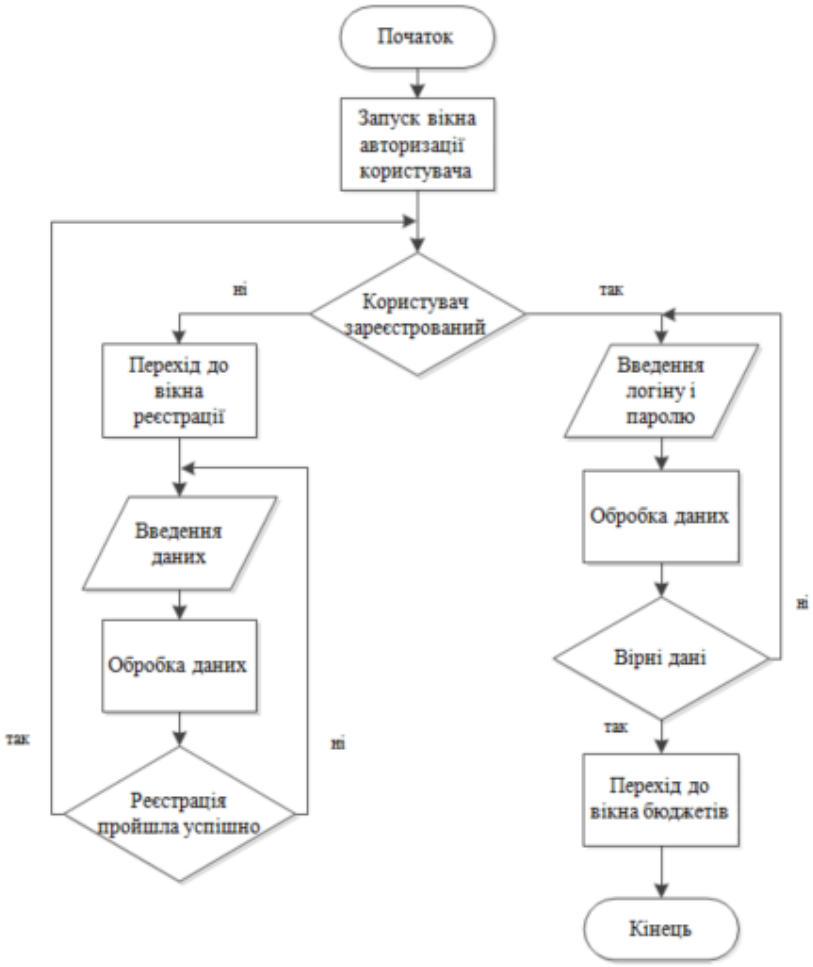


Рисунок 2.3 — Алгоритм авторизації

Для того щоб писати функціонал, потрібно мати xml файл, до якого він буде привязаний, тому інтерфейс розробляється у першу чергу.

Вікна авторизації та реєстрації, є практично ідентичними з точки зору зовнішнього вигляду, та коду написаного у xml файлі. Розміщення всіх віджетів інтерфейсу та задання їх характеристик здійснюється у Linear Layout, це елемент, який вирівнює усі об’єкти в середині себе вертикально або горизонтально. В даному випадку в ньому розміщуються такі елементи як:

— scrollView — елемент інтерфейсу який дозволяє прокручувати його вміст;

— textView — елемент інтерфейсу що містить у собі статичний текст;

— editText — віджет у який зберігається текст який вводить користувач;

— button — кнопка яка при натиску виконує задану їй функцію.

Фрагмент коду наведено нижче:

```
<ScrollView
 android:layout_width="match_parent"
 android:layout_height="wrap_content">
    <TextView
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:gravity="center"
       android:text="@string/log_in"
       android:textAppearance="?android:textAppearanceLarge"
       android:textSize="30sp"
       android:textStyle="bold|italic" />
    <EditText
       android:id="@+id/email_login"
       android:layout_width="350dp"
       android:layout_height="wrap_content"
       android:background="@drawable/edittext_background"
       android:hint="@string/email"
       android:inputType="textEmailAddress"/>
    <Button
       android:id="@+id/btn_login"
       android:layout_width="250dp"
       android:layout_height="wrap_content"
       android:layout_marginBottom="10dp"
       android:background="@drawable/edittext_background"
       android:text="@string/sign_in"
       android:textAppearance="?android:textAppearanceMedium"
       android:textSize="30sp"
          </ScrollView>
```

У результаті було отримано інтерфейс який зображено на рисунку 2.4.

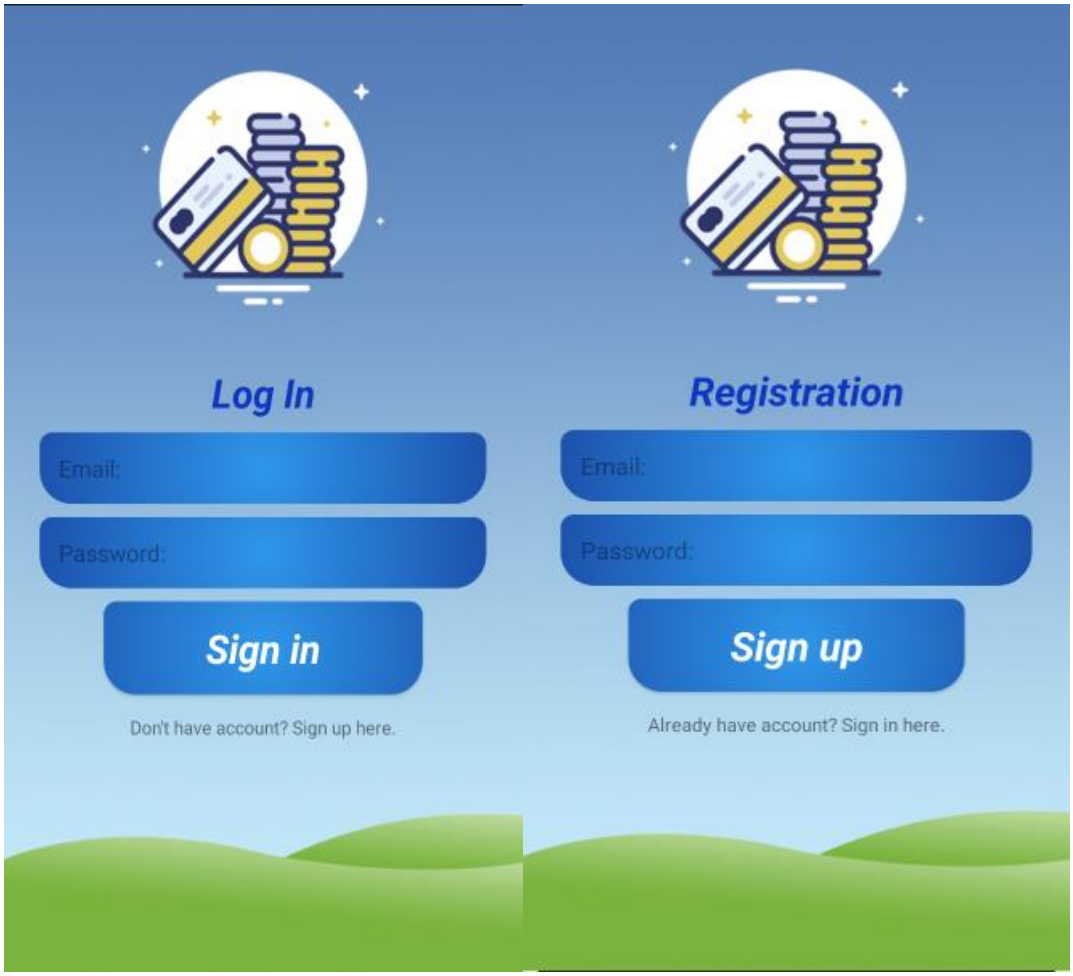


Рисунок 2.4 — Інтерфейс вікон авторизації та реєстрації

Для того зв’язати інтерфейс з файлом у якому буде описуватись його функціонал, використовується команда яку оголошують в методі onCreate() в файлі MainActivity.js : setContentView(R.layout.activity_main);

Далі потрібно налаштувати роботу кнопок та EditText. Дані, які будуть вводитись користувачем у поля пошти та паролю, зберегтись у базі даних, яку буде інтегровано та організовано пізніше. Данні які вводяться у поле для паролю, будуть прихованими крапками (базове введення паролю). Після натисненя на кнопку буде здійснено перехід до наступного вікна додатку. Для цьог потрібно налаштувати прослуховувач для кнопки, та оголосити змінні які будуть містити дані, що було введено у пусті поля. Код для виконання даних операцій наведено нижче:

```
btnLog.setOnClickListener(new View.OnClickListener() {
   public void onClick(View v) {
      String email=mEmail.getText().toString().trim();
      String pass=mPass.getText().toString().trim();
   };
```

Для того щоб не було необхідності під час кожної авторизації у додаток вводити логін та пароль, можна додати режим зберігання користувача. Тобто при відкритті застосунку, користувач одразу перейде до головного меню. Ця можливість буде працювати до тих пір поки користувач не вийде з свого облікового запису:

```
if (mAuth.getCurrentUser()!=null){
   startActivity(new Intent(getApplicationContext(),HomeActivity.class));
}
```

2.2 Створення навігаційного меню

Меню навігації — це меню, що з’являється ліворуч або праворуч від головного екрану. Воно дозволяє користувачу побачити інформаційну структуру додатку, переключатися між вікнами, наборами даних, вкладками та функціональними аспектами мобільного застосунку.

Під час розробки мобільного додатку для контролю бюджету, було вирішено розмістити навігаційне меню ліворуч. Оскільки, основна задача даного меню — лаконічний показ усіх функцій додатку, меню складаєтсья з двох частин:

— nav_header.xml — файл що служить шапкою навігаційного меню;

— navmenu.xml — файл що містить перелік доступних вікон з функціоналом.

Файл nav_header.xml по суті представляє додаток, це невеличке поле яке містить назву додатку та його емблему. Для його розробки використовувалось 3 елементи графічного інтерфейсу : Linear layout, TextView та ImageView.

ImageView — це віджет інтерфейсу який дозволяє описувати графічне зображення (картинки) [12]. В якості заднього фону було використано градієнт по

типу того який використовувався для конопок у меню авторизації. Код для реалізації меню навігації наведено нижче:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="200dp"
   android:background="@drawable/dashboard_style"
   android:orientation="vertical">
   <ImageView
      android:id="@+id/imageView"
      android:gravity="top|left"
      app:srcCompat="@drawable/main_icon" />
   <TextView
      android:id="@+id/textView"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:startColor="#2F97EA"
      android:text="Pocket Bookkeeper"
      android:textSize="30sp"
      android:textStyle="bold|italic" />
</LinearLayout>
```

Наступним етапом розробки меню навігації, є написання navmenu.xml. Даний файл містить список усіх вікон, які будуть доступні для використання у подальшому. Для кожного елементу списку потрібно оголосити:

— id — номер по якому можна викликати даний елемент в Activity;

— title — назва цього елементу яка буде відображатись на екрані користувача;

— icon — обрати створену, або наявну іконку для цього елементу.

У розробленому мультимедійному мобільному застосунку, навігаційне меню містить у собі п’ять елементів. Команди, для опису одного із елементів меню наведено нижче:

```
<item android:id="@+id/dashboard"
   android:title="@string/dashboard_fragment"
   android:icon="@drawable/dashboard"/>
```

Маючи дві розроблені частини меню, їх можна об’єднати у одне вікно, та задати їм певні графічні характеристики (колір, шрифт тощо). Для цього у папці проекту layout, створюється новий файл activityhome.xml, та у ньому описується та під’єднуються розроблені частини навігаційного меню. Замість звичного LinearLayout, для опису меню використовують DrawerLayout так як він дозволяє підключити NavigationView (елемент який дозволяє відображати вікно як меню):

```
<androidx.drawerlayout.widget.DrawerLayout
   tools:context=".HomeActivity">
   <include layout="@layout/appbar_layout"/>
   <com.google.android.material.navigation.NavigationView
      app:headerLayout="@layout/nav_header"/>
if (getArguments() != null) {
         mParam1 = getArguments().getString(ARG_PARAM1);
         mParam2 = getArguments().getString(ARG_PARAM2);
mParam1 = getArguments().getString(ARG_PARAM1);
         mParam2 = getArguments().getString(ARG_PARAM2);
      }
</androidx.drawerlayout.widget.DrawerLayout>
```

Меню навігації у даному мобільному застосунку містить такі кнопки:

— DashBoard — перехід до головного меню додатку;

— Income list — перехід до вікна з списком доходів;

— Expense list — перехід до вікна з списком витрат;

— Budget Target — перехід до вікна для контролю особистого бюджету;

— LogOut — вихід із облікового запису.

У результаті проведених дій, було розроблене навігаційне меню, яке є важливою частиною у будь-якому мобільному програмному засобі (рисунок 2.5).

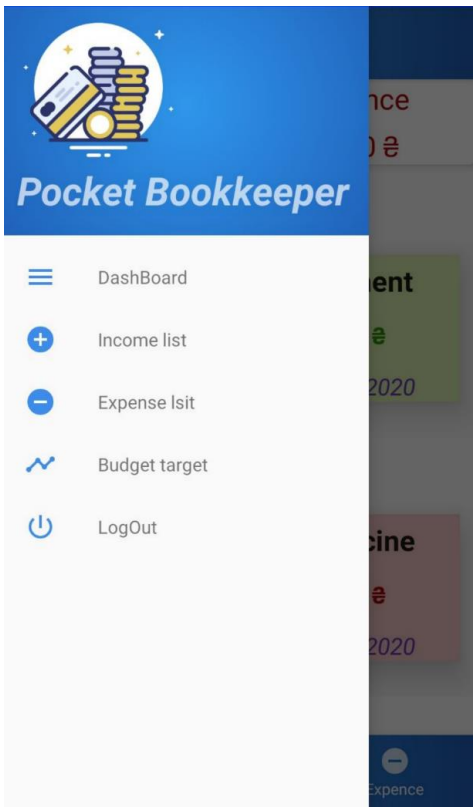


Рисунок 2.5 — Навігаційне меню мобільного застосунку

2.3 Реалізація вікон для ведення контролю особистого бюджету

Написання основного функціоналу є найбільш об’ємною частиною розробки, так об’єднує у собі велику кількість вікон, арифметичних операцій та графічної роботи. На даному етапі розробки мобільного застосунку відбувається написання таких частин додатку як :

— Dashboard — панель головного меню додатку, на якій відображається перелік витрат та доходів, їх загальна сума, кнопки для вводу інформації про нові надходження або витрати фінансів;

— Income/Expense list — вікна які дають детальну інформацію про транзакції у вигляді списку;

— Budget Target — вікно у якому користувач може бюджетний ліміт який він не повинен перевищувати, та календарний термін на який цей ліміт оголошується;

— LogOut — функція виходу з облікового запису користувача.

2.3.1 Розробка фрагменту Dashboard

Даний фрагмент головного меню було розроблено використовуючи віджет CardView. Це елемент, який подальшим розвитком FrameLayout, він дозволяє створити гарну карточку з тінню та заокругленими кутами. Основне його призначення — служити контейнером для інших компонентів. При розробці панелі для головного меню CardView використовується двічі, а саме для зони в якій відображається загальні числа усіх витрат, та для комірок у яких буде міститись детальна інформація про транзакції. Для пілключення необхідного фрагменту, використовується наступна команда:

View myview=inflater.inflate(R.layout.*fragment_dash_board*, container, false);

Для зручного відображення даних про фінансовий облік, усі ці картки були поміщенні у RecyclerView. Даний елемент дозволяє пролистувати горизонтальний список певних елементів, в нашому випадку це картки (CardView). Картки мають особливість яка відсутня у звичайних LinearLayout, а саме їм можна задати висоту та кут, що дозволить створити тінь на графічному інтерфейсі користувача. Для підключення RecyclerView використано наступні команди:

mRecyclerIncome=myview.findViewById(R.id.*recycler_income*);
mRecyclerExpense=myview.findViewById(R.id.*recycler_expense*);

Нижня частина Dashboard представляє собою меню, яке містить три кнопки: “панель”, “доходи” та “витрати”. Кнопки “доходи” та “витрати” при натисканні на них, відкривають вікно у якому потрібно заповнити інформацію про відповідну транзакцію.

Це вікно містить три поля:

— Сума — число фінансів яке було витрачено або отримано;
— Тип — на що саме витратились фінанси або отримались (Заробітна плата, Купівля продуктів, тощо);

— Примітка — Додаткове поле у якому користувач може більш обширно описати транзакцію (наприклад конкретніше вказати що він купив у магазині, або за що він отримав надходження коштів) .

Після того як користувач заповнив пусті поля інформацією, потрібно натиснути кнопку “Зберегти”, тоді данні про гроші будуть занесені до бази даних:

```
mIncomeDatabase=
FirebaseDatabase.getInstance().getReference().child("IncomeData").child(uid);
mExpenseDatabase=
FirebaseDatabase.getInstance().getReference().child("ExpenseData").child(uid);
```

Якщо користувач одразу натисне на кнопку “Зберегти”, при цьому не ввівши дані до пустих полів, система видасть помилку та підказку у якій буде сказано про необхідність заповнення обов’язкових полів. Після того як користувач зберіг внесені дані потрібно підключити графічні елементи що відповідають за виведення загальних сум витрат та доходів:

```
totalIncomeResult=myview.findViewById(R.id.income_set_result);
totalExpenseResult=myview.findViewById(R.id.expence_set_result);
```

Для відображення загальної суми доходів та витрат у верхній частині головного меню, необхідно додати нову змінну, яка буде додавати своє значення до кожної величини яку буде введено:

```
mIncomeDatabase.addValueEventListener(new ValueEventListener() {
public void onDataChange(@NonNull DataSnapshot dataSnapshot) { int totalsum=0;
      for(DataSnapshot mysnap:dataSnapshot.getChildren()){
         Data data=mysnap.getValue(Data.class); totalsum+=data.getAmount();
         String stResult=String.valueOf(totalsum); totalIncomeResult.setText(stResult);
      }    }
   public void onCancelled(@NonNull DatabaseError error) {    }
public void onCreated(@NonNull DatabaseError error) {    }
});
```

Наступним етапом є передача введених даних до карточки (CardView), для відображення їх на інтерфейсі користувача. Так як ці дані вже беруться із бази даних, тому для того щоб дістати їх звідти потрібно задіяти адаптер бази даних Firebase. Він дозволяє переміщати данні у відповідні їм комірки з даними:

```
FirebaseRecyclerAdapter<Data,ExpenseViewHolder>expenseAdapter=new
FirebaseRecyclerAdapter<Data, ExpenseViewHolder>
      (      Data.class,
             R.layout.dashboard_expense,
             DashBoardFragment.ExpenseViewHolder.class,
             mExpenseDatabase ) {
   protected void populateViewHolder(ExpenseViewHolder viewHolder, Data model,
int i) {  viewHolder.setExpenseType(model.getType());
      viewHolder.setExpenseAmount(model.getAmount());
viewHolder.setExpenseAmount(model.getAmount());
viewHolder.setExpenseNote(model.getNote());
viewHolder.setExpenseType(model.getType());
      viewHolder.setExpenseDate(model.getDate());
}
};
mRecyclerExpense.setAdapter(expenseAdapter);
```

У картки CardView педається лише два з трьох полів, а саме сума та тип. Поле з приміткою буде відображатись у наступних вікнах додатку. Замість них у картці показується дещо важливіша інформація, а саме дата виконання транзакції. Додаток показує число, місяць та рік, які бере у відповідності до тієї дати яка налаштована в системі телефону у даний момент часу. Картки розміщені у горизонтальній послідовності в два ряди, та мають свої кольори (зелений для доходів, червоний для витрат).

У результаті розробки, було отримано наступний вигляд панелі головного меню, який зображено на рисунку 2.6.

Рисунок 2.6 — Результат розробки фрагменту Dashboard

2.3.2 Розробка списків витрат та доходів

У навігаційному меню оголошено такі елементи як Income list Expense list. Це вікна які будуть містити у собі детальнішу інформацію про доходи чи витрати.

Створення графічного вигляду даних вікон було здійснено використовуючи такий інструмент як RelativeLayout. У цьому віджеті зручно розміщувати елементи інтерфейсу, так як можна вказати який елемент після якого слідує. На відміну від LinearLayout, де всі елементи йдуть тільки один за одним тільки горизонтально або тільки вертикально, RelativeLayout дозволяє розміщувати елементи в довільному порядку і саме так як того захоче розробник.

Данні списки будуть містити у собі загальну суму фінансів певної категорії, та повний детальний список усіх транзакцій. Окрім цього додано можливість оновити або видалити дані. Для реалізації цих функцій необхідно налаштувати прослуховувачі на кнопки, приклад коду програми наведено нижче та у додатках:

```
btnUpdate.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        type=edtType.getText().toString().trim();
        note=edtNote.getText().toString().trim();
        String mdamount=String.valueOf(amount);
        mdamount=edtAmount.getText().toString().trim();
        int myAmount=Integer.parseInt(mdamount);
        String mDate= DateFormat.getDateInstance().format(new Date());
        Data data=new Data(myAmount,type,note,post_key,mDate);
        mIncomeDatabase.child(post_key).setValue(data);
        dialog.dismiss();
    }});
btnDelete.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        mIncomeDatabase.child(post_key).removeValue();
        dialog.dismiss();
    }
btnDelete.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        mIncomeDatabase.child(post_key).removeValue();
        dialog.dismiss();
    }
});
```

Для того щоб обрати якусь дію над будь-яким записом зі списку, потрібно на нього натиснути, після чого виведеться вікно аналогічне до вікна для заповнення данних про фінанси. Ввівши нові данні та натиснувши кнопку “Оновити”, старий запис буде стерто з бази данних, та перезаписано під новим id-номером, та тією датою коли користувач здійснив оновлення. Відповідно кнопка “Видалити” — знищує запис із бази даних без можливості повернути його. У результаті розробки було отримано вікна зображені на рисунку 2.7.

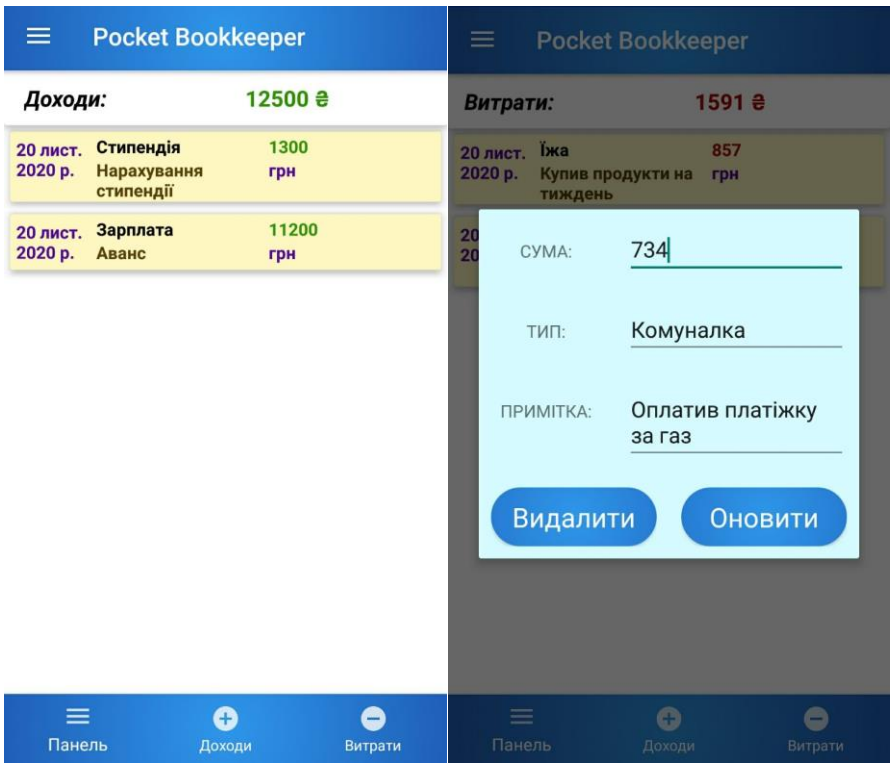


Рисунок 2.7 — Списки доходів та витрат

2.3.3 Розробка вікна для контролю бюджету

Заключним етапом розробки мобільного програмного застосунку для контролю особистого бюджету є створення вікна для визначення ліміту витрачених коштів, та його календарних меж. Функція даного фікна — встановити ліміт коштів який не можна перевищувати за певний період часу. При перевищенні зазначеної суми, користувач отримує сповіщення про те, що він не справляється з виставленою задачею.

Вікно містить у собі такі дані:

— дата початку дії ліміту;

— дата кінця дії ліміту;

— ліміт — сума грошей яку не можна перевищувати за вказаний вище період часу;

— витрати — сума усіх витрат за даний період;

— залишок — число яке дорівнює різниці ліміту та витрат, вказує на суму коштів які доступні для використання.

Розробивши графічну частину вікна, потрібно під’єднати до неї функціонал. В першу чергу фрагмент коду під’єднано до бази даних Firebase:

```
mAuth=FirebaseAuth.getInstance();
FirebaseUser mUser=mAuth.getCurrentUser();
String uid=mUser.getUid();
mBudgetDatabase=
FirebaseDatabase.getInstance().getReference().child("BudgetData").child(uid);
mExpenseDatabase=
FirebaseDatabase.getInstance().getReference().child("ExpenseData").child(uid);
```

У вікні для контролю бюджету присутня одна кнопка — “Змінити”. При натисненні на неї, користувач побачить на екрані панель де має ввести календарні дані та суму ліміту. Щоб зберегти отриману інформацію потрібно налаштувати прослуховувач на кнопку “Зберегти”:

```
btnEdit.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      editDateItem();
   }
});
```

Наступним кроком буде налаштування відображення дати. Так як дані про календарний день, місяць та рік беруться з системи мобільного пристрою, необхідно задіяти такий перелік команд:

```
DatePickerDialog.OnDateSetListener() {
   public void onDateSet(DatePicker view, int year, int monthOfYear,                int
dayOfMonth) {       myCalendar.set(Calendar.YEAR, year);
      myCalendar.set(Calendar.MONTH, monthOfYear);
      myCalendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);
      sdf = new SimpleDateFormat("dd/MM/yyyy", Locale.UK);
```

Після того як кнопку “Зберегти” налаштовано, потрібно зберегти введене у базі даних, для цього потрібно налаштувати її, щоб інформація про контроль бюджету зберігалась в окремій папці та з унікальним id-номером:

```
mBudgetDatabase.addValueEventListener(new ValueEventListener() {
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        for (DataSnapshot mysnapshot:dataSnapshot.getChildren()) {
            BData data = mysnapshot.getValue(BData.class);
            StdateText.setText(data.getStdate());
            endateText.setText(data.getEndate());
            balance=limitR-DashBoardFragment.expenseR;
            BalanseText.setText(balance+ " ₴");
            ExpensesText.setText(DashBoardFragment.expenseR+" ₴");
            limitText.setText(data.getLimit()+" ₴");
});
```

У результаті виконання вище перечислених дій, було отримано графічне вікно з функціями контролю бюджету (рисунок 2.8).

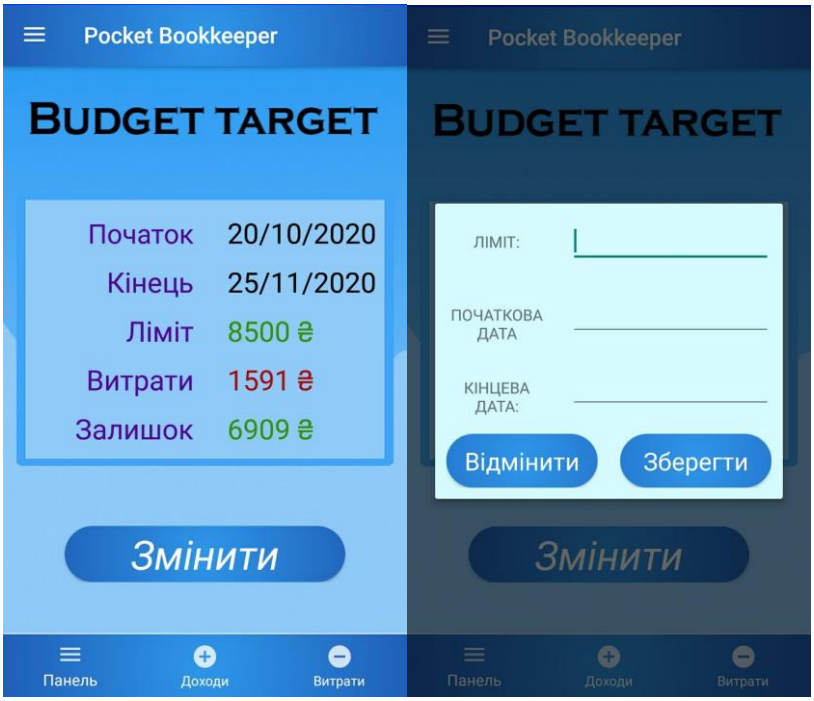


Рисунок 2.8 — Вікна для контролю бюджету

2.4 Інтегрування бази даних Firebase Realtime

Для подальшої розробки мобільного застосунку, необхідно організувати підлючення бази даних. Для виконання задачі даної магістерської роботи було обрано базу даних реального час Firebase Realtime.

Першим кроком є реєстрація на офіційному сайті — firebase.google.com, після чого можна створювати новий проект який буде звязаний з вашим додатком. Процес створення нового проекту вимагає виконати певну послідовність дій, а саме:

— вказати ваше унікальне і’мя пакету;

— назвати проект;

— вказати сертифікат підписання налагодження SHA-1 (даний сертифікат знаходиться у Android Studio, та знайти його можна перейшовши у Gradale projects>android>signingReport) ;

— скачати файл google-services.json, та перенести його у кореневий каталог програми;

— додати Firebase SDK (набір засобів розробки), ввевши певні рядки коду у файл build.gradle, та синхронізувавни нововведення у середовищі розробки.

Тепер над створеним проектом можна виконувати різні маніпуляції. Так як нам необхідно виконати здійснення авторизації, у розділі Authentication>Sign-in method, можа обрати засоби завдяки яким можна авторизуватись у розроблений мобільний застосунок (рисунок 2.9). В нашому випадку це адреса електронної пошти та пароль.

Виконавши вище перечисленні дії, базу даних реального часу інтегровано у середовище розробки, але вона знаходиться у виключеному стані. Щоб включити базу даних потрібно у середовищі розробки Android Studio перейти по шляху: Tools > Firebase > Realtime Database > Save&Retrieve data, та нажати на кнопку Connect to Firebase. Підключивши базу даних реального часу до середовища розробки, можна починати взаємодіяти з нею безпосередньо через код у файлах типу js. Для виконання авторизації написано фрагмент коду, який міститься у прослуховувачі кнопки входу, цей код відповідає за передачу даних

від додатку до хмари бази даних, після чого виконується перехід до наступного екрану (Activity).

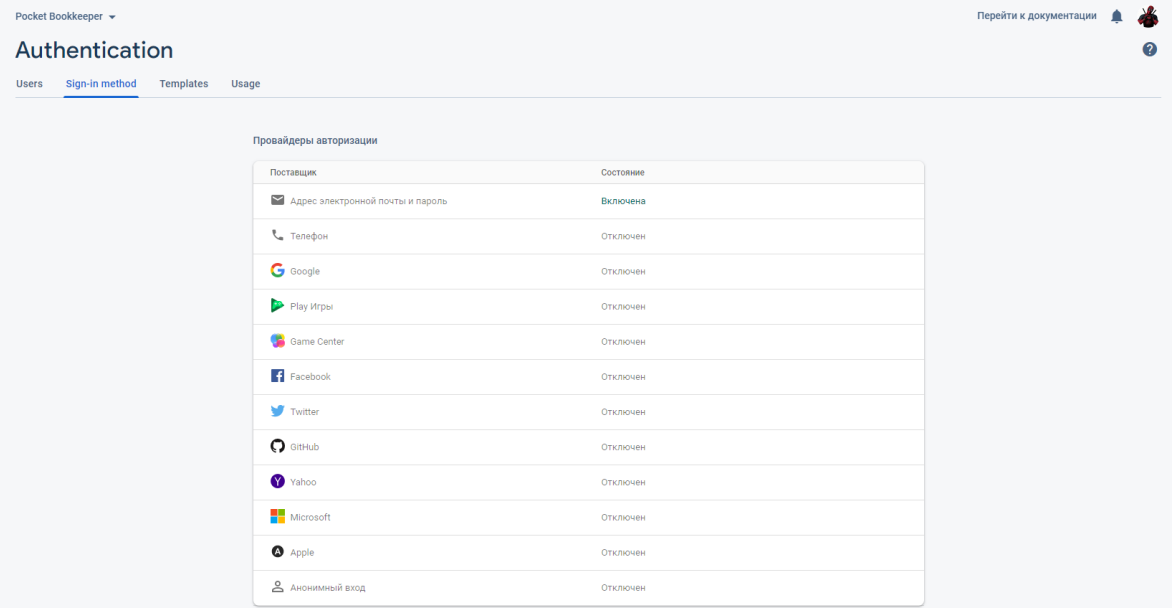


Рисунок 2.9 — Засоби для авторизації

Фрагмент коду, для запису введених користувачем даних у необхідні поля, наведено нижче та у додатках:

```
mAuth.signInWithEmailAndPassword(email,pass).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
   public void onComplete(@NonNull Task<AuthResult> task) {
      if (task.isSuccessful()){
         mDialog.dismiss();
         startActivity(new Intent(getApplicationContext(),HomeActivity.class));
Toast.makeText(getApplicationContext(),R.string.login_s,Toast.LENGTH_SHORT).sho
w();
Toast.makeText(getApplicationContext(),R.string.login_f,Toast.LENGTH_SHORT).sho
w(); });
```

Окрім даних про акаунт користувача, у базу даних заноситься уся інформація про доходи, витрати та встановлений ліміт користувача. Після того як

користувач натиснув кнопку для збереження інформації у базу даних, на екрані його мобільного пристрою має з’явитись повідомлення про те що дані збережено, або при збереженні виникла помилка (у разі не збереження інформації). Інформація про зареєстрованих користувачів зберігається у окремому розділі від даних про облік фінансів. Кожен запис розміщується у конкретну папку (наприклад записи про витрати коштів зберігаються у папці Витрати, записи про надходження коштів — у папці Доходи). Усі нові записи, що надходять до бази даних, отримують свій унікальний id-номер. Далі вони зберігаються у папку, назва якої відповідає id-користувача який цей запис додав (рисунок 2.10).

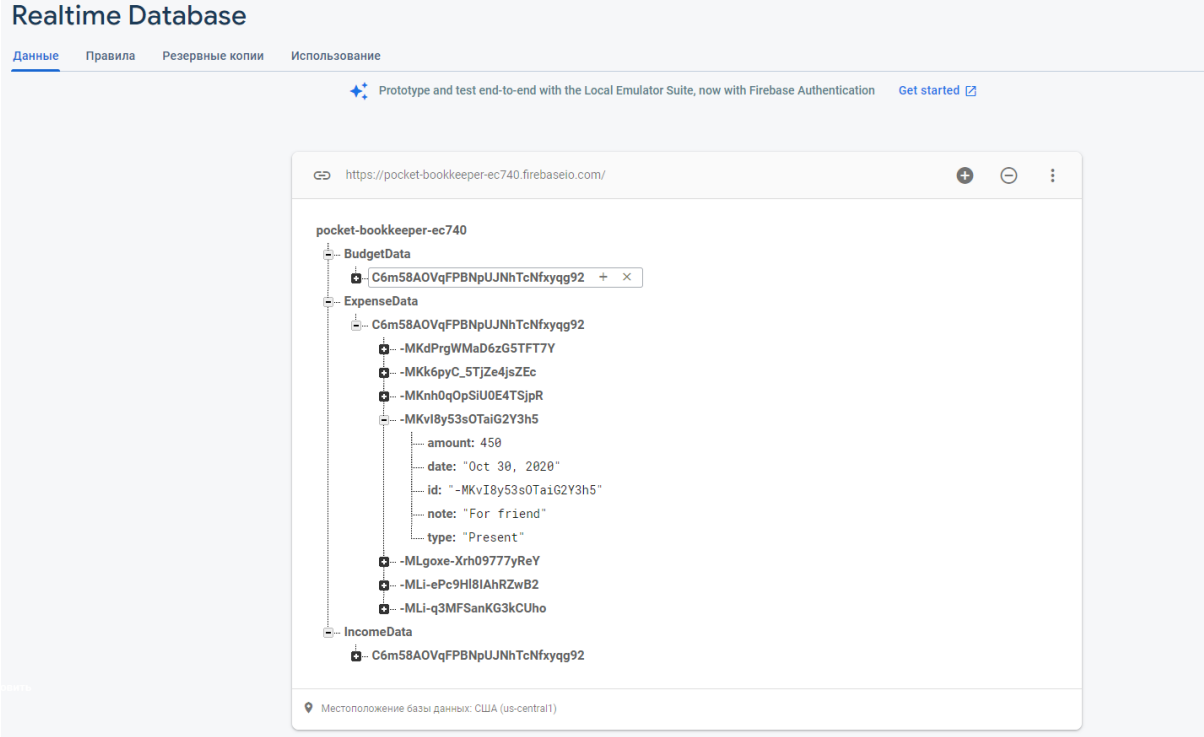


Рисунок 2.10 — Структура записів бази даних

Дана особливість дозволяє побачити кому належать ті чи інші записи у базі даних. Щоб отримати доступ до перегляду записів, на сторінці Firebase потрібно перейти до Realtime Database у вкладку Данні.

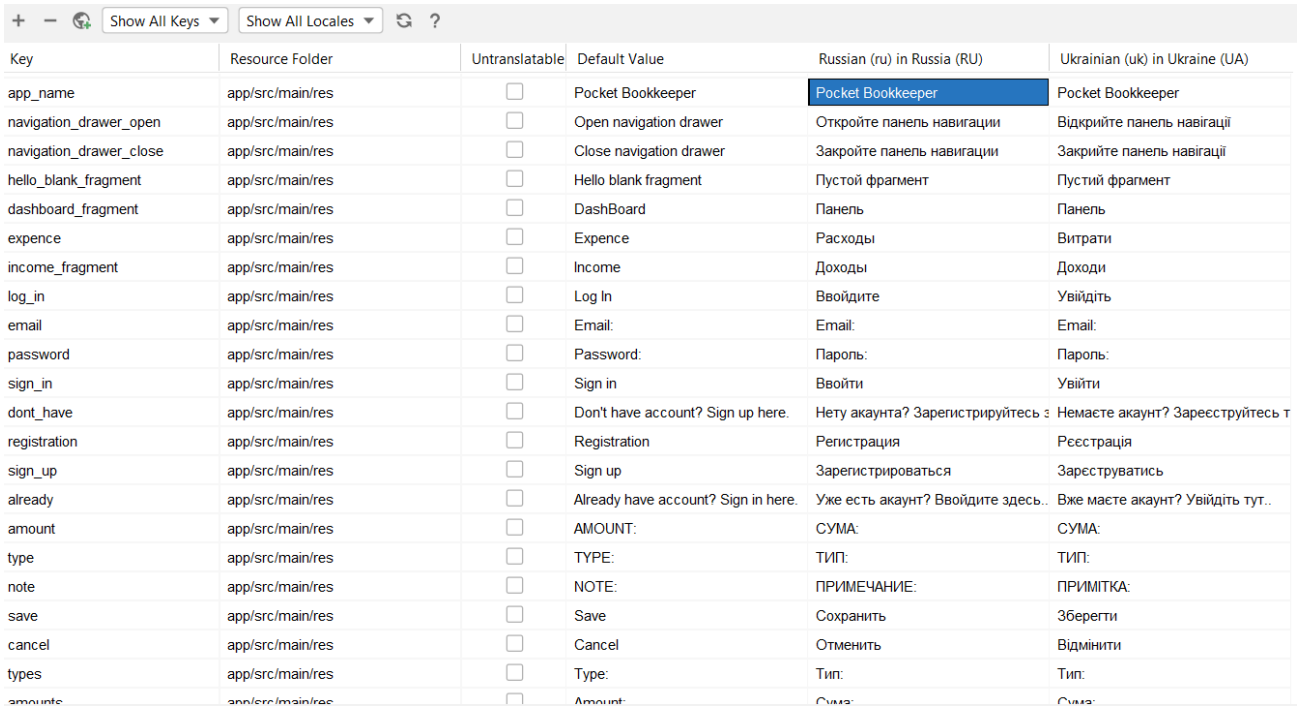
2.5 Реалізація локалізації та дизайну інтерфейсу

Мовна локалізація — це адаптація та переклад певного “продукту” до особливостей певних країн, регіонів або груп населення. Під “продуктом”

розуміється будь який товар або послуга.

Для впровадження локалізації, необхідно кожному текстовому елементу надати свій id, та оголосити їх у файлі string.xml що знаходиться в папці values. Після виконання вище вказаних дій, потрібно правою клавішою миші натиснути на файл string.xml, та зайти у редактор переводів. Він містить список слів або виразів які були додані у string.xml, англійською мовою за замовчуванням. Щоб додати якусь мову потрібно натиснути на кнопку Add Locate, та обрати будь-яку із списку доступних (понад 200 мов). Додавши якусь мову, необхідно вручну заповнити поля, які будуть містити їх переклад, напроти кожного слова (рисунок 2.11). При розробці даного мультимедійного програмного застосунку, було виконано локалізацію на трьох мовах (Англійська, Українська та Російська).

Мова у додатку буде відповідати мові системи телефону. Проте якщо в налаштуваннях мови телефону, обрати ту якої немає у редакторі перекладу, то застосунок виставить мову за замовчуванням, тобто англійську.



Key	Resource Folder	Untranslatable	Default Value	Russian (ru) in Russia (RU)	Ukrainian (uk) in Ukraine (UA)
app_name	app/src/main/res	☐	Pocket Bookkeeper	Pocket Bookkeeper	Pocket Bookkeeper
navigation_drawer_open	app/src/main/res	☐	Open navigation drawer	Откройте панель навигации	Відкрийте панель навігації
navigation_drawer_close	app/src/main/res	☐	Close navigation drawer	Закройте панель навигации	Закрийте панель навігації
hello_blank_fragment	app/src/main/res	☐	Hello blank fragment	Пустой фрагмент	Пустий фрагмент
dashboard_fragment	app/src/main/res	☐	DashBoard	Панель	Панель
expence	app/src/main/res	☐	Expence	Расходы	Витрати
income_fragment	app/src/main/res	☐	Income	Доходы	Доходи
log_in	app/src/main/res	☐	Log In	Ввойдите	Увійдіть
email	app/src/main/res	☐	Email:	Email:	Email:
password	app/src/main/res	☐	Password:	Пароль:	Пароль:
sign_in	app/src/main/res	☐	Sign in	Ввойти	Увійти
dont_have	app/src/main/res	☐	Don't have account? Sign up here.	Нету акаунта? Зарегистрируйтесь з	Немаєте акаунт? Зареєструйтесь т
registration	app/src/main/res	☐	Registration	Регистрация	Реєстрація
sign_up	app/src/main/res	☐	Sign up	Зарегистрироваться	Зарєструватись
already	app/src/main/res	☐	Already have account? Sign in here.	Уже есть акаунт? Ввойдите здесь..	Вже маєте акаунт? Увійдіть тут..
amount	app/src/main/res	☐	AMOUNT:	СУМА:	СУМА:
type	app/src/main/res	☐	TYPE:	ТИП:	ТИП:
note	app/src/main/res	☐	NOTE:	ПРИМЕЧАНИЕ:	ПРИМІТКА:
save	app/src/main/res	☐	Save	Сохранить	Зберегти
cancel	app/src/main/res	☐	Cancel	Отменить	Відмінити
types	app/src/main/res	☐	Type:	Тип:	Тип:
amounts	app/src/main/res	☐	Amount:	Сума:	Сума:

Рисунок 2.11 — Редактор перекладів мобільного застосунку

За зовнішній вигляд усіх елементів інтерфейсу та віджетів (кнопки, текст, задній фон, тощо) відповідає репозиторій res (ресурси). У ньому зберігаються всі

графічні елементи які служать так званою “обгорткою” для об’єктів інтерфейсу. При розробці мобільного застосунку для контролю бюджету, робота велась з двома елементами репозиторію ресурсів, а саме з папкою drawable та файлом colors.xml.

Папка drawable містить у собі майже усі графічні елементи які будуть відображатись на інтерфейсі користувача, а саме :

— іконки головного меню та кнопок;

— файли з картинками;

— градієнти;

— значки;

Вікна реєстрації, авторизації та ліміту бюджету у якості заднього фону мають картинки. Для того щоб зробити картинку фоном якогось вікна, необхідно інтегрувати його, перетянувши мишею у папку drawable [13].

Деякі кнопки та поля меню, за фон мають градієнт. Градієнт можна створити окремо для будь якого об’єкту інтерфейсу, для цього потрібно у папці drawable створити новий файл з розширенням .xml, та дати йому назву. Всередині цього файлу можна описати тип градієнта, його кольори та розміри. Наприклад, для полів вводу елктронної пошти та паролю на екрані входу, було створено окремий файл edittext_background.xml з градієнтом лише ддя цих полів, у якому було виначено два різних відтінки синього кольору які будуть переливатись один в одного, та радіус заокруглення кутів, код даного фрагменту наведено нижче та у додатках:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <gradient android:type="radial"
      android:gradientRadius="200dp"
      android:startColor="#2F97EA"
      android:endColor="#1A49A6"/>
   <corners android:bottomLeftRadius="25dp"
      android:bottomRightRadius="25dp"
```

```
    android:topLeftRadius="10dp"
    android:topRightRadius="10dp"/>
</shape>
```

Кнопки навігаційного меню, та панелі інструментів мають свої іконки (значки). Іконки — це невеличка картинка яка позначає додаток, каталог, файл, кнопку. Натиснення мишею або іншим вказівним приладом на іконку запускає відповідну дію (відкритя, закриття, тощо).

Щоб додати іконку потрібно клікнути правою кнопкою миші по папці drawable, та обрати New > Image Asset. Відкриється відповідне вікно для для налаштування об’єкту зображення (рисунок 2.12).

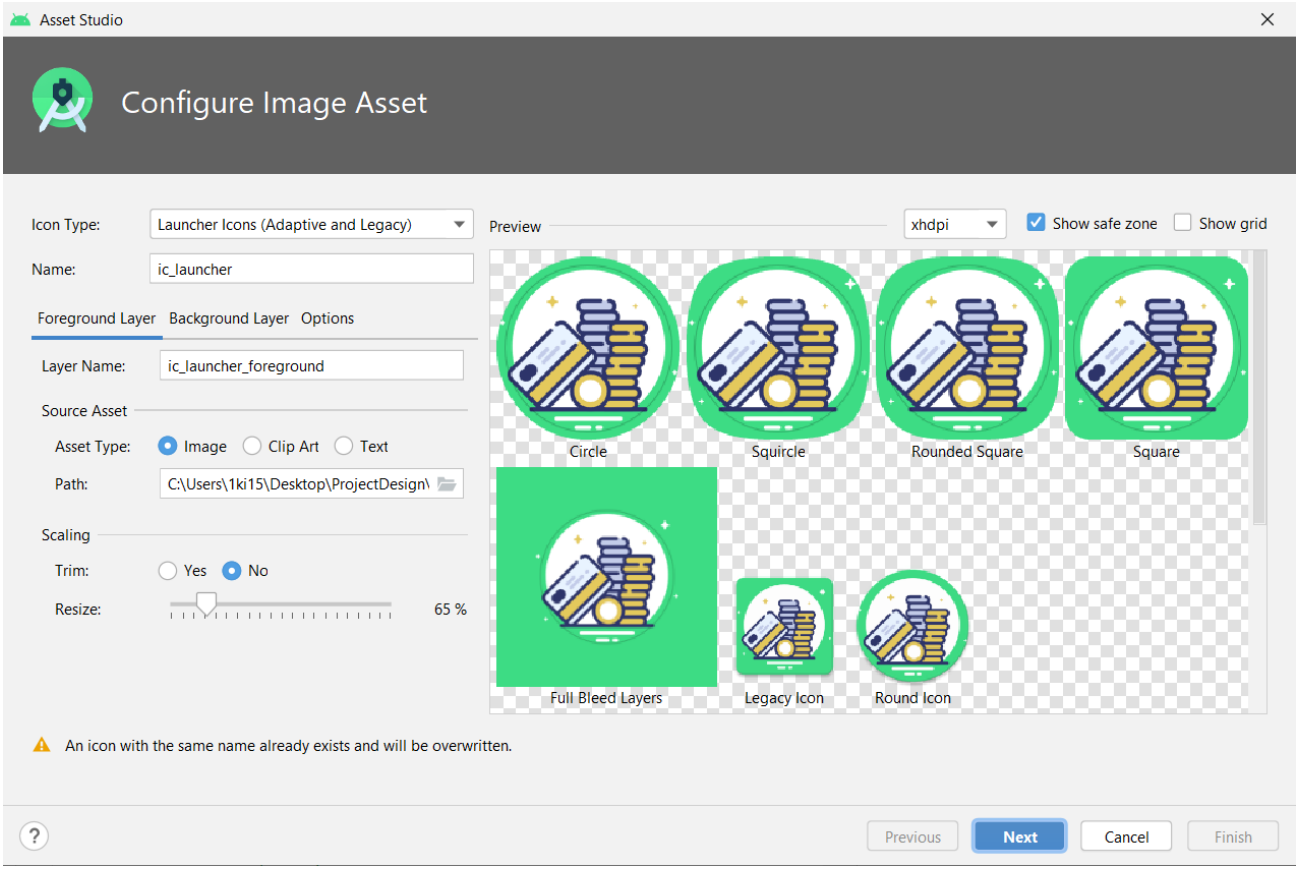


Рисунок 2.12 — Створення та налаштування нової іконки

В даному вікні можна вибрати тип значка, та виконувати над ним певні маніпуляції. Android Studio має інтегровану бібліотеку, яка містить досить обширний набір базових значків, з яких можна обрати будь-який. Також є

можливість створити свій значок, завантаживши необхідне зображення у відповідну папку. Будь-якій іконкі необхідно надати ім’я, для того щоб її можна було оголосити у коді для опису певного графічного елементу.

Отже, у даному розділі було покроково розроблено мультимедійний програмний засіб для контролю особистого бюджету. Створено алгоритм роботи авторизації та реєстрації облікового запису користувача, розроблено основну логіку та функціонал програми, наведено частковий код для реалізації їх у мобільному додатку. Інтегровано базу даних реального часу, та розроблено інтуїтивно-зрозумілий інтерфейс користувача.

3 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Процес тестування будь-якого програмного забезпечення, це перш за все перевірка того, чи відповідає продукт вимогам які йому заявлені [14]. Будь-який програмний продукт вимагає тестування перед випуском його у публіку, а особливо мобільні додатки. Тестування мобільних застосунків дає відповіді на те, як відреагує створена програма в певних умовах, та деякі інші аспекти а саме:

— чи працює додаток через бездротову мережу та в комп'ютерній мережі;
— як буде працювати додаток за умови відсутності підключення до інтернету;
— як відреагує додаток якщо його роботу перерве телефонний дзвінок;
— що станеться якщо під час роботи програми прийде стороннє повідомлення;
— реакція додатку на недостачу пам'яті у пристрої;
— мінімальна конфігурація яка необхідна для нормальної роботи розробленої програми;
— чи налаштовані сповіщення;
— захист додатку;
— реакція програмного продукту на обертання пристрою;
— правильність роботи заявленого функціоналу програмного забезпечення.

3.1 Тестування мобільного додатку

Виконати тестування мобільного застосунку, розробленого у середовищі Android Studio, можна двома способами:

— використовуючи емулятор;
— використовуючи реальні мобільні пристрої.

Оскільки обране середовище розробки надає емулятор, процес тестування було проведено використовуючи його. Емулятор на відміну від реального мобільного пристрою, має дещо обмежений функціонал. Проте, у даному випадку, функції які відсутні у емуляторі не мають необхідності використовуватись, тому емулятора буде достатньо [15].

Для того щоб провести тестування створеного програмного забезпечення у емуляторі, його потрібно спочатку створити (рисунок 3.1). Створення

віртуального аналогу приладу для тестування здійснюється у Android Virtual Device Manager. Дана програма дає на вибір список доступних приладів, для яких можна створити емулятор (телевізор, смарт-годинник тощо). В даному випадку створюється віртуальний аналог смартфону.

Наступним етапом створення є вибір характеристик смартфону, а саме:

— назва (Samsung G5) ;

— розмір екрану (5.0″);

— розширення дисплею (1080 x 1920) ;

— щільність (420 dpi).

Після цього потрібно обрати рівень API на якому буде запускатись програма (API 21), та початкова орієнтація пристрою (вертикальна).

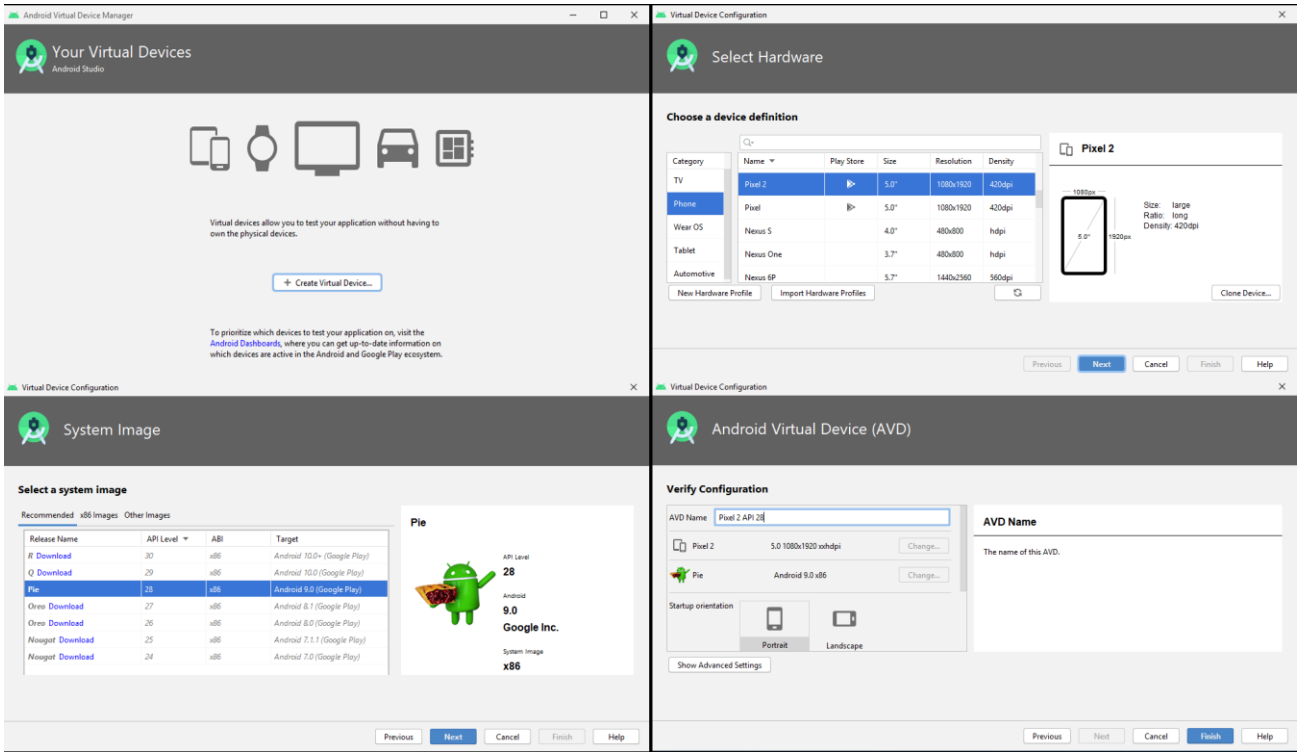


Рисунок 3.1 — Процес створення віртуального пристрою для тестування додатку

Створивши віртуальний пристрій для перевірки роботи розробленого мобільного застосунку, у середовищі розробки програми потрібно скористатись комбінацією клавіш Shift+F10, або натиснути на кнопку “Run App”, що знаходиться у верхній частині на панелі інструментів, для запуску емулятора (рисунок 3.2) [16].

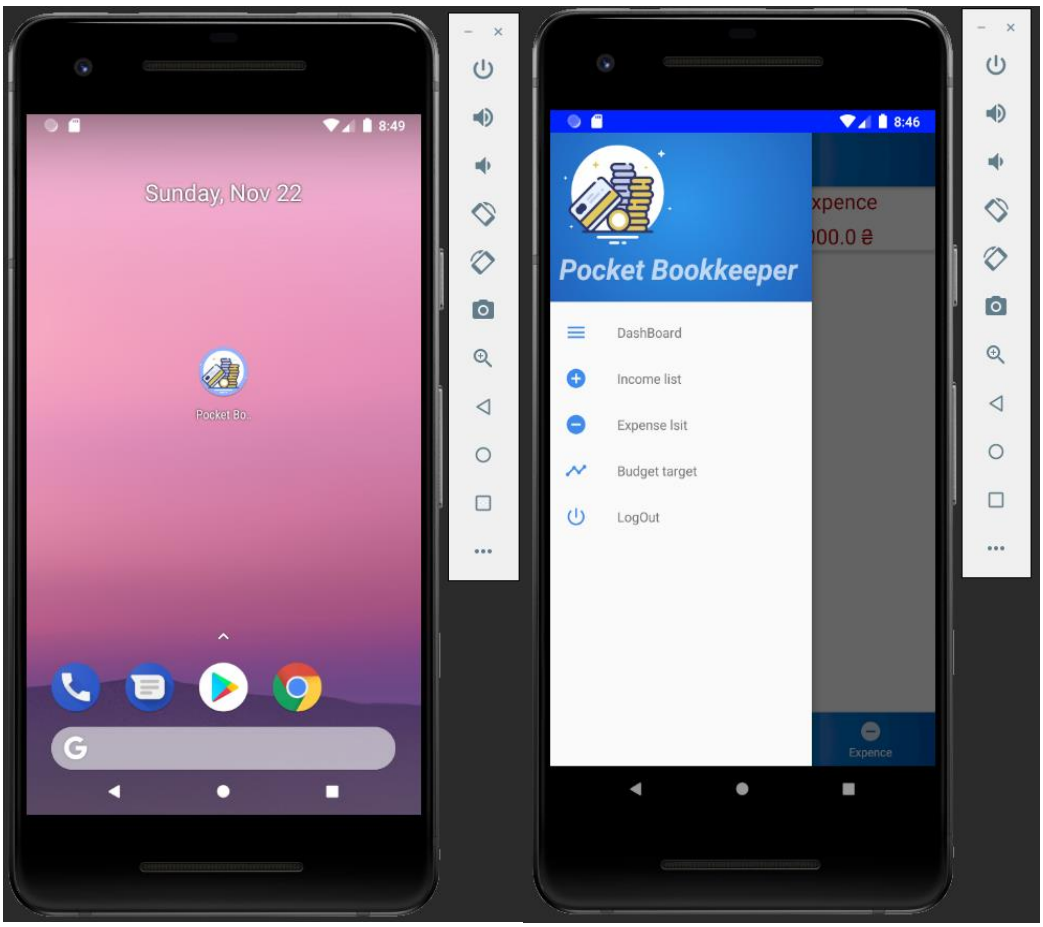


Рисунок 3.2 — Android емулятор

Відкривши емулятор смартфону, можна проводити тестування мобільного застосунку. Цей процес здійснюється шляхом перевірки відповідності функціонування усіх кнопок, вікон, вкладок та полів (чи відповідає функція певного елементу інтерфейсу — функціям які заявлені додатком).

Провівши тестування розробленого програмного засобу для контролю бюджету, критичних помилок у роботі функціоналу застосунку не було виявлено. У мобільному застосунку відсутня зміна локазілації через меню налаштувань, тобто мова додатку буде відповідати мові системи мобільного пристрою користувача. Так як розроблений програминй засіб використовує базу даних релального часу Firebase Realtime, то він не може працювати без доступу до мережі Інтернет.

Знайдені дефекти в роботі не є критичними, та не впливають на головну роботу мобільного додатку, проте потребують виправлення.

3.2 Тестування роботи бази даних

Перевірка правильності роботи бази даних перевіряється простим чином. Після того як користувач зберіг якісь внесенні дані у базу даних реального часу, на його екрані з’являється повідомлення про те що дані успішно збережено, або при їх зберіганні виникла помилка. Якщо з’явилося повідомлення про успішне збереження даних, для перевірки достовірності цієї інформації потрібно зайти у консоль бази даних Firebase Realtime. Щоб перевірити чи було зареєстровано новий обліковий запис користувача, потрібно перейти у вкладку Authentication, де одразу буде відображено які облікові записи зареєстровано у даному додатку (рисунок 3.3).

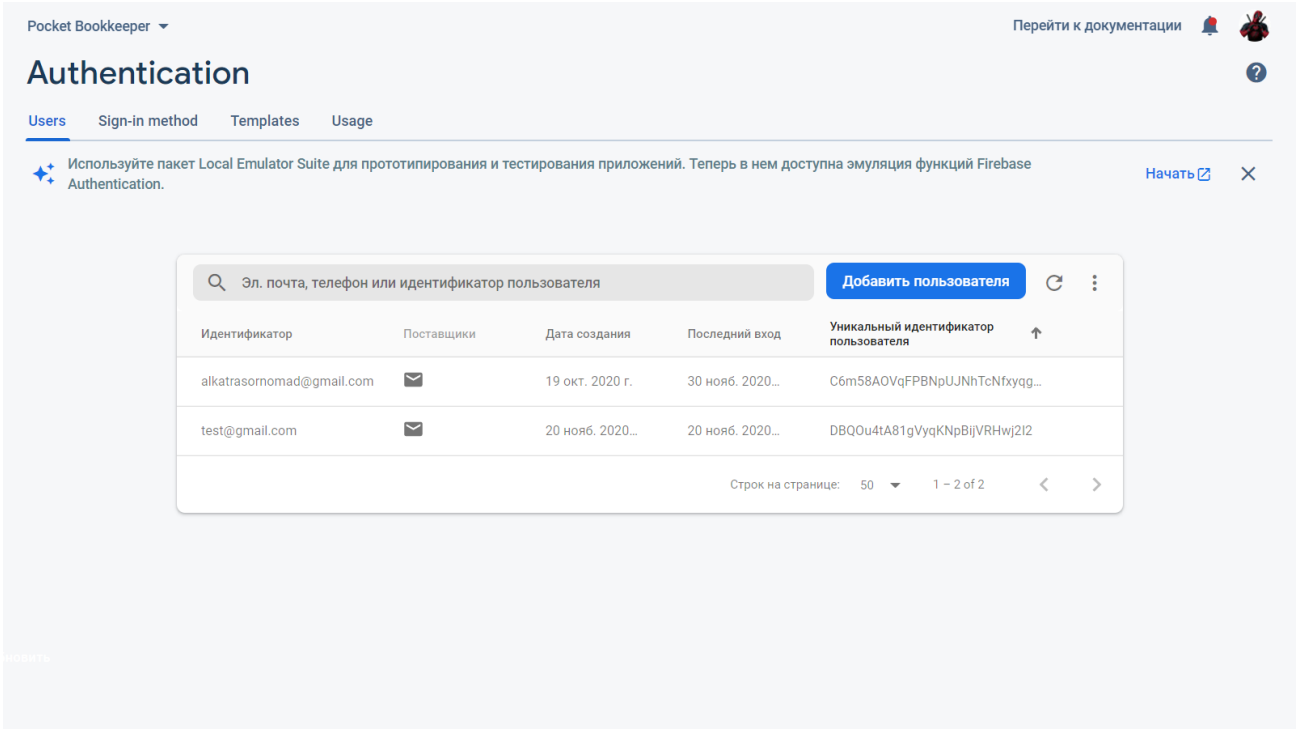


Рисунок 3.3 — Перелік зареєстрованих користувачів

Наступним кроком тестування бази даних, є перевірка того чи записуються дані у сховищі бази даних. Для цього потрібно перейти у вкладку Realtime Database, де у вікні “Дані” повинно відображатись кожен запис у відповідній папці, та з унікальним id-кодом, який не повинен збігатись із іншими записами (рисунок 3.4).

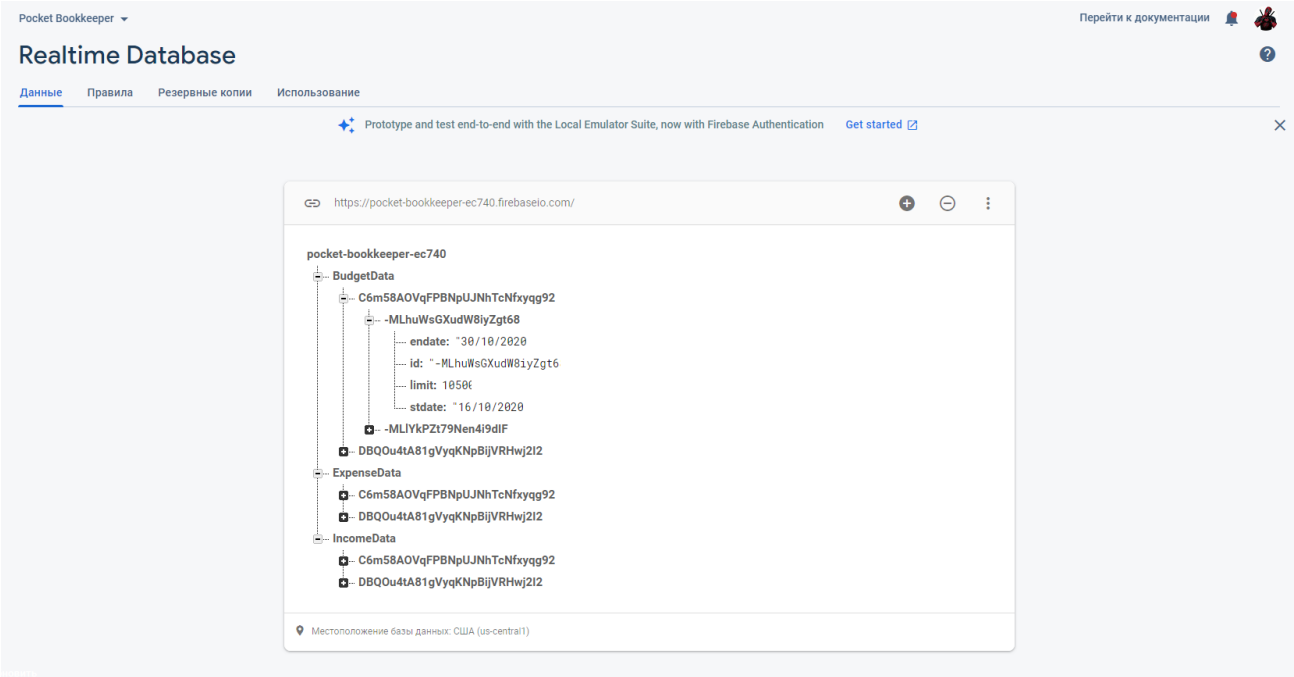


Рисунок 3.4 — Список збережених записів у базі даних

В даному розділі було проведено практичне тестування роботи мобільного програмного застосунку, використовуючи емулятор Android Studio. У результаті тестування було показано стабільність та правильність роботи бази даних реального часу, але виявлено один не значний недолік у роботі системи авторизації, який потребує виправлення.

4 ЕКОНОМІЧНА ЧАСТИНА

Темою магістерської кваліфікаційної роботи є «Мультимедійний програмний засіб контролю особистого бюджету для ОС Android». За цією темою в економічній частині проводяться розрахунки економічних показників на розробку програмного продукту та впровадження його на ринок аналогічних товарів.

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть Гарнага В.А. та Кадук О.В.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями, наведеними у таблиці 4.1, за 5-ти бальною шкалою.

Таблиця 4.1 — Критерії оцінювання комерційного потенціалу розробки та їх бальна оцінка

Бали (за 5-ти бальною шкалою)					
Кри-те-Рій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продук-кту в реаль-них умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно до-рівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження таблиці 4.1

4	Технічні та споживчі властивості продукту значно гірші,ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші,ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі,ніж в аналогів	Технічні та споживчі властивості продукту значно кращі,ніж в аналогів
5	Експлуатаційні витрати значно вищі,ніж в аналогів	Експлуатаційні витрати дещо вищі,ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі,ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамі-Кою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної,так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної,так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси,які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси.Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали,що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10років	Термін реалізації ідеї більший за 5років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х Років

Кінець таблиці 4.1

12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту,що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 4.2.

Таблиця 4.2 — Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Гарнага В.А., к.т.н доц. кафедри ОТ	2. Кадук О.В., к.т.н, доц. кафедри ОТ
	Бали, виставлені експертами:	
1	4	4
2	0	1
3	4	4
4	2	3
5	4	3
6	2	3
7	2	3
8	3	3
9	4	4
10	4	4
11	4	3
12	4	4
Сума балів	$СБ_1 = 37$	$СБ_2 = 39$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 38$	

Отже, з отриманих даних таблиці 4.1 видно, що рівень потенціалу нової розробки — вище середнього. Дана розробка є конкурентоспроможною з аналогами, так як для її розробки було проаналізовано недоліки та переваги аналогових продуктів, та на основі цього впроваджену у розробку. Вона має соціологічний вплив, так як покращує ефективність контролю бюджету користувачів, що є важливим у житті людей з обмеженими фінансами.

4.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько-технологічної роботи

Для розробки нового програмного продукту необхідні такі витрати.
Основна заробітна плата для розробників визначається за формулою (4.1):

$$З_о = \frac{М}{Т_р} \cdot t, \quad (4.1)$$

де М — місячний посадовий оклад конкретного розробника;

$Т_р$ — кількість робочих днів у місяці, $Т_р$ = 20 днів;

t — число днів роботи розробника, t = 37 днів.

Розрахунки заробітних плат для керівника і програміста в таблиці 4.3.

Таблиця 4.3 — Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	7500	375	4	1500
Інженер-програміст	9600	480	33	15840
Всього:				17340

Розрахуємо додаткову заробітну плату, вона розраховується як 10-12% від суми основної заробітної плати всіх розробників:

$$З_{дод} = 0{,}1 \cdot 17340 = 1734 \text{ (грн.)}$$

Нарахування на заробітну плату $Н_{зп}$ для працівників бюджетної сфери становить 22% від суми основної та додаткової заробітної плати:

$$Н_{зп} = (З_о + З_р) \cdot \frac{\beta}{100},$$

$$Н_{зп} = (17340 + 1734) \cdot \frac{22}{100} = 4196{,}28 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А=\frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12},$$

де Ц — балансова вартість обладнання, грн;

$Н_а$ — річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т — Термін використання (Т=2 міс.).

Таблиця 4.4 — Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн
Персональний комп’ютер	12700	25	2	529,1
Мобільний пристрій	4300	15	2	107,5
Всього:				636.6

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і,$$

де n — кількість комплектуючих;

Ні — кількість комплектуючих і-го виду;

Ці — покупна ціна комплектуючих і-го виду, грн;

Кі — коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 4.5 — Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
USB — microUSB кабель	шт.	75	1	75
Пачка паперу	уп.	100	1	100

Ручка	шт.	4	1	4
Всього з урахуванням транспортних витрат				196,9

Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п ;$$

де В — вартість 1кВт-години електроенергії (В=2.35 грн/кВт);

П — установлена потужність комп'ютера (П=0,6кВт);

Ф — фактична кількість годин роботи комп'ютера (Ф=296 год.);

$К_п$ — коефіцієнт використання потужності ($К_п < 1$, $К_п = 0,7$).

$$В_е = 2,35 \cdot 0,6 \cdot 296 \cdot 0,7 = 292,15 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$. Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (\mathbf{1}..\mathbf{3}) \cdot (З_о + З_р).$$

Отже, розрахуємо інші витрати:

$$В_{ін} = \mathbf{1} * (\mathbf{17340} + \mathbf{1734}) = \mathbf{19074} \text{ (грн).}$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

В = 17340+1734+4196,28+636,6+196,9+292,15+19074= 43 469,83 (грн.)

Розрахуємо загальну вартість наукової роботи $В_{заг}$ за формулою:

$$В_{заг} = \frac{В_{ін}}{\boldsymbol{\alpha}}$$

де α — частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{\mathbf{43\,469{,}83}}{\mathbf{1}} = \mathbf{43\,469{,}83}\text{ грн.}$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta}$$

де β — коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{\mathbf{43\,469{,}83}}{\mathbf{0{,}9}} = 48\,299{,}8\text{ грн.}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Спрогнозуємо кількісно, яку вигоду можна отримати у майбутньому від впровадження результатів виконаної наукової роботи.

Виконання наукової роботи та впровадження її результатів буде здійснюватися протягом одного року. Основні позитивні результати від впровадження розробки очікуються протягом 3-х років після її впровадження. Одним із основних позитивних результатів є зростання величини прибутку.

При реалізації результатів наукової розробки покращується якість програмного продукту, що дозволяє підвищити ціну його реалізації на 150 грн. Кількість одиниць реалізації програмного засобу також збільшиться: протягом першого року — на 700 шт., протягом другого року — ще на 450 шт., протягом третього року — ще на 300 шт.

Реалізація продукції до впровадження результатів наукової розробки складала 50 шт., а ціна — 250 грн.

Спрогнозуємо збільшення чистого прибутку підприємства від впровадження результатів наукової розробки у кожному році відносно базового за такою формулою 4.2:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \qquad (4.2)$$

де $\Delta П_о$ — покращення основного оціночного показника від впровадження результатів розробки у даному році. Зазвичай таким показником може бути ціна одиниці нової розробки;

N — основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN — покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$Ц_о$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n — кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

λ — коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт 0,8333.

ρ — коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати = 0,2...0,3;

υ — ставка податку на прибуток. υ = 18%.

Тоді, збільшення чистого прибутку підприємства протягом першого року складе:

$$\Delta П_1 = [150 \cdot 50 + (250 + 150) \cdot 700] \cdot 0{,}8333 \cdot 0{,}2 \cdot \left(1 - \frac{18}{100}\right) = 39290{,}09 \text{ грн}$$

Протягом другого року:

$$\Delta П_2 = [150 \cdot 50 + (250 + 150) \cdot (700 + 450)] \cdot 0{,}8333 \cdot 0{,}2 \cdot \left(1 - \frac{18}{100}\right) = 61\,710 \text{ грн}$$

Протягом третього року:

$$\Delta П_3 = [150 \cdot 50 + (250 + 150) \cdot (700 + 450 + 300)] \cdot 0{,}8333 \cdot 0{,}2 \cdot \left(1 - \frac{18}{100}\right) = 79\,900 \text{ грн}$$

Отже, протягом трьох років підприємство може розраховувати на збільшення чистого прибутку від реалізації наукової розробки.

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $E_{абс}$вкладених інвестицій розраховується за формулою:

$$E_{абс} = (ПП - PV),$$

де ПП — приведена вартість всіх чистих прибутків, які отримає підприємство (організація) від реалізації результатів наукової розробки, грн.;

PV — теперішня вартість інвестицій PV = ЗВ = 48 299,8 грн.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 4.1.

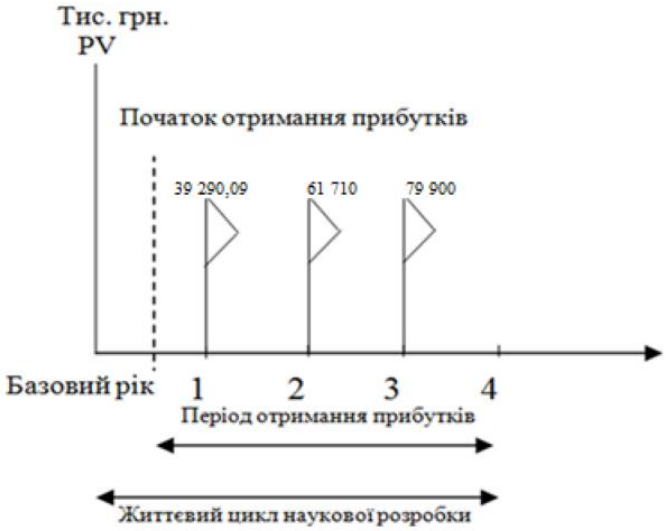


Рисунок 4.1 — Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів наукової роботи

У свою чергу, приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_1^T \frac{\Delta П_i}{(1+\tau)^t},$$

де $\Delta П_i$ —збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т — період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t — період часу (в роках) від моменту отримання чистого прибутку до точки “0”.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{48\,063{,}6}{(1+0{,}1)^0} + \frac{39290{,}09}{(1+0{,}1)^2} + \frac{61710}{(1+0{,}1)^3} + \frac{79900}{(1+0{,}1)^4} = 181\,474{,}87 \text{ грн.}$$

Тоді розрахуємо $E_{абс}$:

$$Е_{абс} = 181\,474{,}87 - 48\,299{,}8 = 133\,175{,}07 \text{ грн.}$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$Е_в = \sqrt[Т]{1 + \frac{Е_{абс}}{PV}} - 1,$$

де $Е_{абс}$ — абсолютна ефективність вкладених інвестицій, грн;

PV — теперішня вартість інвестицій PV = ЗВ, грн;

Т — життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$Е_в = \sqrt[з]{1 + \frac{133\,175{,}07}{48\,063{,}6}} - 1 = 0{,}56 \text{ або } 56\ \%$$

Далі, розраховану величина $Е_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $τ_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $τ_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2020 році в Україні d = 0,2;

f — показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}2 + 0{,}1 = 0{,}3$$

Оскільки $E_в = 56\% > \tau_{мін} = 0{,}3 = 30\%$, то інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в},$$

Якщо $T_{ок} < 3…5$-ти років, то фінансування наукової розробки є доцільним.

$$T_{ок} = \frac{1}{0{,}56} = 1{,}7 \text{ років}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним, оскільки Ток<3 років.

Отже, результати проведених розрахунків дають можливість зробити висновок про доцільність розробки та впровадження нашої наукової роботи. Це підтверджують такі показники як:

— абсолютна ефективність вкладених інвестицій, яка дорівнює 133 175,07 грн., що є більшим 0 і вказує на те, що інвестор може бути зацікавленим у нашій розробці;

— відносна ефективність наукової розробки становить 56%, що є вищим за мінімальну ставку дисконтування (30%), тому вкласти кошти у нашу розробку є вигідніше, ніж покласти кошти на депозит;

— термін окупності вкладених у реалізацію наукового проекту інвестицій складе 1,7 років, що є менше 3-ох і вказує швидку окупність інвестицій.

Крім того, розраховано, що наукова розробка приноситиме підприємству додатковий прибуток протягом 3-х років за рахунок покращення її якості порівняно з існуючими аналогами.Усе це, узяте разом, забезпечує прийняття рішення про доцільність виготовлення нового продукту.

ВИСНОВКИ

Під час виконання даної магістерської кваліфікаційної роботи було проведено детальний літературний огляд методів, етапів та засобів для розв’язання задачі, виконано порівняння з аналогами та обрано оптимальне програмне середовище. .

У ході проектування було побудовано структурну та функціональну частину мобільного програмного засобу. Було проаналізовано методи розробки мобільних застосунків.

Проведено порівняльний аналіз інтегрованих середовищ розробки та визначено, що Android Studio є найоптимальнішою для створення мобільного додатку.

Було проведено тестування правильності роботи бази даних, а також усіх меню та інтерфейсів мобільного додатку, що виявило декілька не значних недоліків, проте на правильну роботу основних функціональних одиниць програмного модуля та готовність його до експлуатації користувачем це не вплинуло.

Отже, в ході написання магістерської кваліфікаційної роботи було виконано усі завдання сформульовані у вступі, отримані навички розробки мультимедійних програмних засобів для ОС Android, що і було покладено в основу мети розробки даної роботи. По темі роботи виконано публікацію.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Розробка мобільних додатків. [Електронний ресурс]. — Режим доступу: http://kiev.itstep.org/razrabotka-mobilnyh-prilozhenij-pod-android
2. Методологія розробки мультимедійних додатків. [Електронний ресурс]. — Режим доступу: http://android.mobile-review.com/articles/22580
3. Класифікація мобільних додатків. [Електронний ресурс]. — Режим доступу: http://voroninstudio.eu/uk/service/razrabotka-mobilnih-prilozheniy.
4. Технології розробки мобільних додатків [Електронний ресурс]. — Режим доступу: http://lektsii.net/2-50017.html
5. Android Studio. [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/Android_Studio
6. Шмидт Г.А. Практическое введение в Android Studio — Оренбург, 2014. 458с.
7. Попов Е.Г. Теоретический курс по Java — Москва, Триумф, 2016. — 523с.
8. Ашок К.С. Mastering Firebase for Android Development — Лондон, 2018.
9. Лаура Томсон, Люк Веллинг. Архітектура Android додатків, 2003. — 872 с.
10. Activity. [Електронний ресурс]. — Режим доступу: http://metanit.com/java/android/2.1.php
11. Мельник Д., Петренко К. Алгоритми роботи додатків, 2014 г. - 378 с.
12. Layout. [Електронний ресурс]. — Режим доступу: http://developer.alexanderklimov.ru/android/theory/layout.php
13. Гизберт Д. UX/UI дизайн - Санкт-Петербург, НТ Пресс, 2015 г. - 320 с.
14. Чедвік Діксон, Роберт Ройс. Тестування мобільних додатків, 2017. — 127 с.
15. Чаффер Д., Шведберг К. Тестирование мобильных приложений - Москва, Символ-Плюс, 2010 г. - 448 с.
16. Емулятори Android. [Електронний ресурс]. — Режим доступу: http://appstips.ru/articles/emulyatory-android.html
17. Гарнага В. А. Модель операції порівняння при порозрядному аналого-цифровому перетворенні з прогресуючим набором тривалостей тактів

урівноваження / О. Д. Азаров, О. О. Решетнік, В. А. Гарнага // Інформаційні технології та комп’ютерна інженерія. — 2008. — № 3 (13). — С. 5—12.

18. Гарнага В. А. Модель операції порівняння при порозрядному аналого-цифровому перетворенні з прогресуючим набором тривалостей тактів урівноваження / О. Д. Азаров, О. О. Решетнік, В. А. Гарнага // Інформаційні технології та комп’ютерна інженерія. — 2008. — № 3 (13). — С. 5—12.
19. Гарнага В. А. Похибки лінійності двотактного симетричного підсилювача постійного струму / О. Д. Азаров , В. А. Гарнага // Інформаційні технології та комп’ютерна інженерія. — 2008. — № 1 (11). — С. 124—132.