СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра програмування та математики

ПОЯСНЮВАЛЬНА ЗАПИСКА
до кваліфікаційної випускної роботи

освітній ступінь бакалавр
спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)
спеціалізація „Інженерія програмного забезпечення”

на тему „Розробка мобільного додатку для обліку особистих фінансів”

Виконав: студент групи ІПЗ-16бд _____ _В.В. Мікляєв_____
(підпис) (ініціали і прізвище)

Керівник _____ _В.О. Лифар_____
(підпис) (ініціали і прізвище)

Завідувач кафедри _____ _В.О. Лифар_____
(підпис) (ініціали і прізвище)

Рецензент_____

Сєвєродонецьк – 2020

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра програмування та математики

Освітній ступінь бакалавр
спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)
спеціалізація „Інженерія програмного забезпечення”
(назва спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри ПМ,
д.т.н., доцент
_____ Лифар В.О.
“____” _____2020 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

Мікляєву Владиславу Володимировичу
(прізвище, ім’я, по батькові)

1.Тема роботи Розробка веб-платформи для створення сучасних односторінкових сайтів.

Керівник роботи доцент, д.т.н.Лифар Володимир Олексійович,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджений наказом університету від “___”_____20__року №___

2. Строк подання студентом роботи 20 травня 2020 р.

3. Вихідні дані до роботи Об'єктом даної роботи є процес розробки мобільного додатку.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Вступ. Аналіз предметної області, з висвітленням наступних питань: економічна характеристика, аналіз та характеристика аналогів, технологія розробки, мови програмування, постановка задачі. Основна частина, в якої висвітлити: збір вимог та проектування, етап даталогічного проектування. Висновки. Перелік використаних джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслеників)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання__30 березеня 2020 року._____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1	Одержання завдання на виконання роботи	30.03.20	
2	Укладання і погодження з керівником плану і етапів виконання роботи	06.04.20	
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної області»	13.04.20	
4	Аналіз шляхів виконання завдання. Вибір і погодження керівником оптимального шляху	20.04.20	
5	Укладання та тестування програмного продукту	27.04.20	
6	Укладання, оформлення та погодження пояснювальної записки з керівником	04.05.20	
7	Здача готової пояснювальної записки на кафедру	12.05.20	
8	Укладання доповіді і презентації	30.05.20	

Студент _____ ___В.В. Мікляєв_____
(підпис) (ініціали і прізвище)

Керівник роботи _____ _____В.О. Лифар_____
(підпис) (ініціали і прізвище)

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студента гр. ІПЗ-16бд Мікляєв В.В.

Науковий керівник

Доцент, д.т.н. _____ Лифар В.О.

Оцінка наукового керівника: _____

Рецензент _____

ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:

_____

Голова ЕК

_____ Лифар В.О.

підпис

РЕФЕРАТ

Пояснювальна записка до дипломного проекту (роботи) бакалавра: _88_ с., _9_ рис., _17_ бібліографічних джерел посилань , _1_ додатків.

В кваліфікаційній роботі розроблено автоматизовану систему ведення сімейного бюджету як мобільний додаток. Продукт вирішує проблему ведення обліку сімейних і особистих витрат і доходів. Програма значно полегшує контроль за фінансовим станом користувача, так як за потребою надає доступ до інформації про поповнення бюджету, щоденні та щомісячні витрати, заплановані покупки.

Для реалізації мобільного додатку розроблено модель роботи додатку, структуру бази даних, ієрархічну модель автоматизованої системи. Також спроектовано зрозумілий інтерфейс.

Для реалізації автоматизованої системи обрано мову Java та середовище програмування Android Studio.

ЗМІСТ

ВСТУП

Актуальність теми. Проблема управління особистим бюджетом є актуальною для кожної людини. Люди отримують зарплату та інші доходи, здійснюють різні покупки, зберігають заощадження. Людина яка грамотно розпоряджається своїм бюджетом постійно стежить за тим, скільки грошей у нього є, розраховує, скільки грошей йому потрібно витратити, і приймає рішення, де він може заощадити і від чого він може взагалі відмовитися. Можна, багаторазово спростити дані процеси, розробивши систему стеження та управління бюджетом для смартфону.

Розробка мобільних додатків на сучасному етапі є вкрай популярною послугою. Кількість користувачів мобільних пристроїв на різних платформах зростає з кожним днем. Зараз люди багато часу проводять у своїх смартфонах: читають новини, грають в відеоігри та звичайно здійснюють online покупки. Тому було б відмінно мати додаток в котрому можно стежити за історією особистих витрат.

Об'єкт дослідження: Процес розробки мобільних додатків.

Предмет досліджень: Методи та засоби реалізації мобільних додатків, моделі та алгоритми системи ведення сімейного бюджету.

Мета дослідження: Підвищення ефективності ведення сімейного бюджету за рахунок створення сучасних інструментів обліку та аналізу особистих фінансів, їх реалізація у мобільному додатку.

Завдання дослідження:

1. Аналіз існуючих аналогів, а також особливостей реалізації мобільних додатків.
2. Розробка моделей системи ведення сімейного бюджету.
3. Розробка бази даних та реалізація програмного забезпечення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Економічна характеристика предметної області

Для того, щоб ефективно використовувати свої доходи, людина повинна правильно скласти свій бюджет, ретельно продумати купівлі та робити заощадження для досягнення своїх цілей. Для складання особистого або сімейного бюджету необхідно складання списку всіх джерел доходів членів сім'ї. Це зарплата, соціальні виплати і відсотки на заощадження. У статті витрат потрібно перерахувати все, за що треба заплатити протягом місяця: квартплата і послуги, харчування, проїзд, сплата податків і внесків. У плановані витрати так само включаються і заощадження на майбутнє.

Якщо доходи дорівнюють витратам, то це збалансований бюджет. Якщо передбачувані витрати перевищують доходи, то цей бюджет має дефіцит. Бюджет, в якому доходи перевищують витрати, буде мати надлишок. Якщо доходи перевищують витрати, необхідно виключити з планів зайві покупки, щоб збалансувати бюджет.

Сімейний бюджет [1] - це фінансовий план, який підсумовує доходи і витрати (сім’ї) за певний період часу. Іншими словами – це вимірювання суми доходів і витрат сім’ї.

Матеріальне становище людини, стан її фінансів характеризуються особистим (сімейним) бюджетом, що показує величину і збалансованість всіх доходів і витрат людини.

Бюджет (від французького) - грошовий гаманець.

Баланс доходів і витрат людини - це розрахунок і співвідношення особистих чи сімейних витрат з одержуваними доходами.

Баланс доходів і витрат сім'ї, що складається за звітний минулий період (зазвичай місяць, квартал, рік), називається звітним балансом, а складається на майбутні періоди - плановим балансом.

В результаті складання звітного або планового балансу доходів і витрат сім'ї виявляється дефіцит або накопичення сімейного бюджету.

1.2. Аналіз та характеристика аналогів

Для обліку фінансів існує безліч мобільних додатків, які не завжди повною мірою виконують функції, потрібні користувачеві. Для розробки додатку для ведення сімейного бюджету з оптимальним набором функцій проаналізовано переваги та недоліки найвідоміших аналогів.

Основним конкурентом є додаток Toshl Finance [2] – найбільш просте і цікаве рішення для контролю особистих витрат. Щоб почати роботу, необхідно зареєструватися, вибрати основну валюту та синхронізуватися з сервером. Кожний календарний день в додатку являє собою закладку з фінансовими операціями. Додаток має тільки базові функції і включає такі розділи як витрати, дохід, статистика і бюджет. Замість звичних категорій витрат і доходів тут використовуються мітки, що істотно спрощує введення даних. З особливостей програми варто відзначити інтуїтивно зрозумілий інтерфейс (рисунок 1.1), наочні інфографіки та автосинхронізація у фоновому режимі. Toshl Finance доступний безкоштовно, проте є і платна підписка, яка дозволяє додавати необмежену кількість доходів і бюджетів, експортувати звіти в PDF, Excel і Google Docмs.

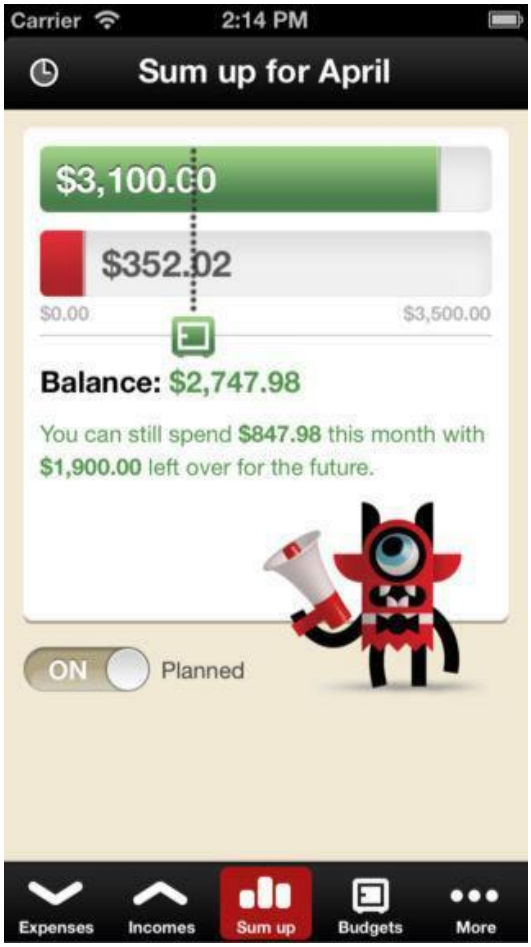

Рисунок 1.1 - Додаток Toshl Finance

Наступним аналогом розробки є додаток MoneyLover [3]. Він має інтуїтивно простий дизайн і зрозумілий користувачеві набір функцій (рисунок 1.2). Цей додаток буде зручним для тих, хто є фанатом яскравої графіки, і кому не потрібен поглиблений статистичний аналіз. Програма дозволяє створювати бюджети, планувати особисті витрати, додавати власні категорії транзакцій, використовувати календар.

Також є функція створення плану заощаджень. Гроші, які планують заощадити, не показуються в загальному балансі, що дозволяє «забути» про них не розраховувати при плануванні витрат на найближчий час. Крім того, можна встановлювати бюджет на окрему категорію чи на всі категорії в гаманці. Якщо ви перевищите бюджет, то декілька разів на день отримуватимете про це сповіщення. Вхід в програму можна захистити пін-кодом, а отже, зберегти витрати та доходи користувача від допитливих очей.

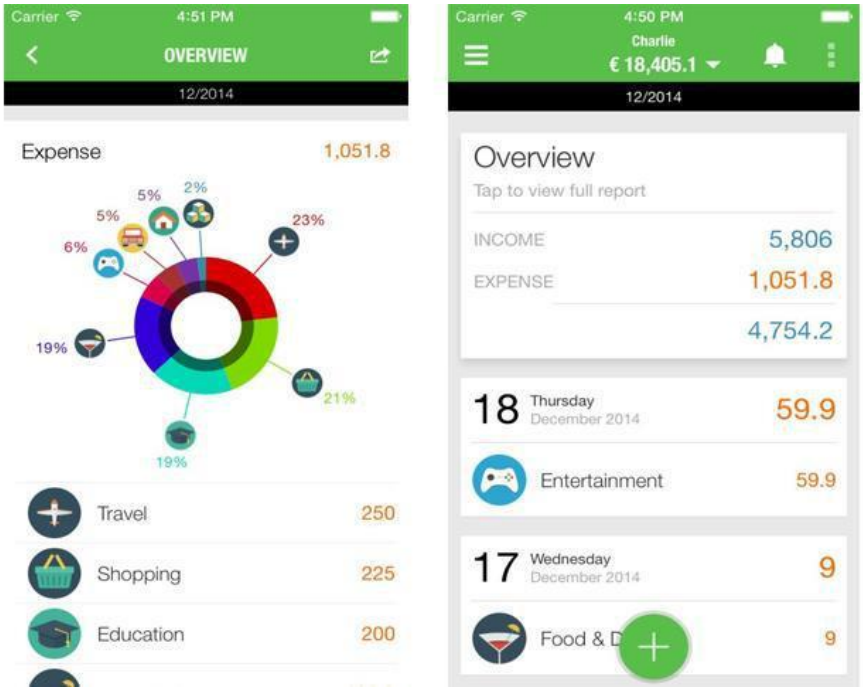

Рисунок 1.2 – Додаток MoneyLover

Monefy [4] – додаток, який також тішить гарною графікою та простотою у використанні. Всі витрати та доходи подані у вигляді великої кольорової діаграми (рисунок 1.3). Розробники створили велику кількість категорій, за якими можна розділити витрати, наприклад, розваги, походи в кафе, шопінг, витрати на продукти, товари для дому, предмети гігієни, таксі та громадський транспорт та ін. Крім того, категорії можна змінювати на власний смак, видаляючи непотрібні та додаючи нові. Всі дані можна синхронізувати з Dropbox, тому їх можна переглядати на комп'ютері та інших пристроях.

Monefy також можна захистити пін-кодом для особистого користування, або ж, навпаки, створити декілька акаунтів, тоді додатком зможуть користуватись декілька людей з одного пристрою.

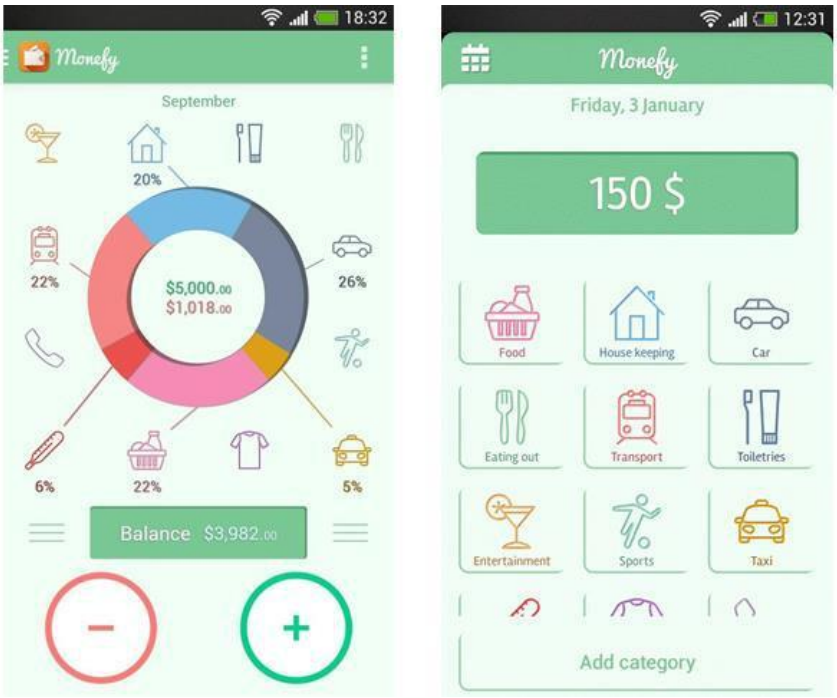

Рисунок 1.3 – Додаток Monefy

Додаток FinancePM розроблений для Android та Windows Phone [5]. Однією з переваг цього додатку є можливість ведення обліку фінансів одразу у кількох валютах. Це досить зручно для тих, хто багато подорожує за кордоном, адже не потрібно кожного разу конвертувати витрати в національну валюту. Крім того, є можливість створювати багато гаманців та підлаштовувати категорії витрат та доходів під себе .

Програма може встановлювати обмеження на бюджетні витрати і попереджувати, коли користувач наблизився до критичного рівня витрат. Також присутня функція налаштування постійних виплат (мобільний зв'язок, комунальні послуги, спортзал) і створення шаблонів платежів з повторюваною сумою.

Рисунок 1.4 – Додаток FinancePM

1.3. Особливості класифікації мобільних додатків

Перед тим, як починати розробку мобільних додатків, слід чітко визначитися з головними функціями майбутньої розробки. Від того, які функції має виконувати мобільний додаток, залежить тривалість його розробки, рівень складності. Загалом визначають чотири основні типи мобільних додатків:

1. Корпоративні. Їх мета – спрощення роботи компанії, швидка передача даних між працівниками або отримання корпоративної інформації. Цільовою аудиторією таких мобільних додатків є, насамперед, працівники компанії, а також реальні та потенційні клієнти та партнери. Розробка мобільних додатків такого типу простіша, для їх дизайну використовуються кольори та елементи корпоративного стилю компанії.

2. Контентні. Це мобільні додатки, основна мета яких надавати різного роду інформацію (текстову або у відео чи аудіоформаті). Головним чином, така розробка мобільних додатків здійснюється для засобів масової інформації, радіо, телеканалів чи порталів.

3. Сервісні. Їх мета випливає з назви, а саме – надавати певні сервісні послуги, тобто виконувати завдання, які ставить користувач, у режимі

реального часу. Розробка мобільних додатків такого типу є складною, адже вони повинні працювати, як годинник: починаючи від калькулятора або будильника і закінчуючи програмами для роботи з великими обсягами тексту або графіки.

4. Ігрові. Основне завдання таких мобільних додатків – розважати, але це не означає, що воно єдине. Отже, розробка мобільних додатків такого типу ускладнюється необхідністю вдало розмістити контекстну або пряму рекламу.

Для розробників мобільних додатків більш важливою є класифікація з точки зору їх структури:

1) веб-додатки можна назвати мобільною версією сайту з розширеним інтерактивом. Веб-додатки не розміщуються в спеціалізованих магазинах, а використовують для роботи браузер. Швидкість роботи таких додатків залежить від якості Інтернет-з'єднання, крім того вони характеризуються низькою вартістю і швидкими термінами реалізації, є кроссплатформенними;

2) гібридні додатки або генератори мобільних додатків – щось середнє між нативними і веб-додатками, встановлюються через офіційні магазини, мають обмежений доступ до апаратної частини пристрою, за функціоналом і якістю наближені більше до нативних додатків, проте дешевше їх, залежать від фреймворку, який використовувався розробником цього додатку;

нативні додатки пишуться на мовах програмування під конкретну платформу і вбудовуються в операційну систему, працюють швидко і коректно, володіють перевагою як по функціоналу, так і за швидкістю роботи інших мобільних додатків. Дані програми також мають доступ до апаратної частини пристроїв: камери, мікрофону, акселерометру, телефонною книги і т.п., економно витрачають ресурси, працюють повністю або частково при відключеному Інтернет-з'єднанні. Звичайно, мають високу вартість і великі витрати за часом написання програми, як наслідок того, що

розробник повинен володіти спеціальними знаннями в середовищі розробки, а також з тієї причини, що кожній платформі відповідає своя мова програмування.

Розроблена автоматизована система ведення сімейного бюджету є сервісним нативним мобільним додатком, оскільки вона написана під конкретну платформу (Android) та надає користувачу сервісні послуги, а саме – можливість обліку його фінансового стану.

1.4. Аналіз та обґрунтування вибору засобів реалізації системи

1.4.1. Аналіз технологій розробки мобільних додатків

Розробка додатків для мобільних пристроїв – це процес, при якому додатки розробляються для невеликих портативних пристроїв таких як КПК, смартфони, стільникові телефони. Ці додатки можуть бути предустановлені на пристрій в процесі виробництва, завантажені користувачем за допомогою різних платформ для розповсюдження ПЗ або являтися веб-додатками, які обробляються на стороні клієнта або сервера.

Сьогодні існує великий вибір мов програмування для розробки мобільних додатків [6]. Це пов'язано з тим, що для різних мобільних пристроїв доводиться використовувати різні мови програмування, що обумовлене тим, що мобільні пристрої мають різні операційні системи. Цільова платформа (або платформи) – iOS, Android, Windows Phone, BlackBerry – буде мати значний вплив на мову розробки, яка буде використовуватися. Наприклад, можна розробляти рідні додатки для кожної платформи або використовувати сторонній інструмент для оптимізації своїх додатків на різних платформах. Другий підхід може заощадити час і зусилля, хоча це може вплинути на зручність використання. Сучасні мобільні пристрої пропонують широкий спектр варіантів розробки.

Проаналізуємо основні технології, які використовуються для розробки додатків для мобільних телефонів. Перша технологія – це Java 2 Micro Edition (J2ME). Це набір специфікацій і технологій, призначених для різних типів портативних пристроїв. Існують два основні напрями: Connected Device

Configuration (CDC) і Connected Limited Device Configuration (CLDC). Напрям визначає тип конфігурації центральних бібліотек Java, а так само параметрів віртуальної машини Java (в якій будуть використовуватися додатки). Пристрої, які використовують технологію CDC будуть більш розвиненими, в якості прикладу можна навести комунікатори. До пристроїв CLDC відносяться звичайні мобільні телефони, які апаратно володіють більш скромними можливостями (ресурсами). Спеціальні режими дозволяють визначати функціональність конфігурацій для різних типів пристроїв. Режим Mobile Information Device Profile (MIDP) призначений для CLDC портативних пристроїв з можливістю спілкування. CLDC і MIDP закладають основу реалізації J2ME. Java-код інтерпретується безпосередньо самим пристроєм за допомогою так званої Java Virtual Machine. Цей механізм робить можливим вільне розповсюдження Java-додатків, так як вони працюють на всіх пристроях з аналогічною Java-платформою. Програмування Java-додатків і на сьогоднішній день займає більшу частину, так як більшість мобільних пристроїв (в основному мобільні телефони) в світі мають вже встановлену Java-систему.

Наступна популярна сьогодні технологія – це технологія Qt. Вона в основному використовується в якості крос-платформного середовища, яке дозволяє використовувати написані з її допомогою додатки на різних пристроях і операційних системах, у тому числі Windows, Mac OS X, Linux, Symbian, Android та інших. Із зростаючою користувальницькою базою Qt, зростає і потреба у вбудованих, мобільних додатках і UI-розробників. Qt є однією з найбільш вдалих бібліотек для C++. Налагодження додатків, розроблених для мобільних пристроїв, відбувається за допомогою емулятора, який міститься в середовищі розробки. Таким чином, можна писати складні програми для мобільних пристроїв з використанням бібліотек C++ і підтримкою платформ. Для роботи Qt на мобільних пристроях необхідна установка певного фреймворку.

Ще однією технологією є технологія розробки Windows Phone SDK. Код додатка, що розроблюється, описується на мові XAML. Хоча насправді – це просто XML файли з мовою розмітки XAML. Платформа Windows Phone не просто чергова платформа для мобільних пристроїв. Вона містить у собі не тільки технологічну складову, але і повністю опрацьовану концепцію дизайну інтерфейсу і взаємодії з користувачем під назвою Metro-дизайн або стиль Metro. Вся розробка під Windows Phone ведеться в середовищі Visual Studio. Для мобільних додатків під Windows Phone відладка та тестування відбувається за допомогою емулятора Windows Phone у середовищі розробки Windows Phone.

Наступна технологія розробки – це iPhone SDK. Розробка під операційну систему iOS можлива тільки з використанням Mac OS X. Але в Інтернеті можна знайти статті, як можна проводити розробку безпосередньо на Macintosh і навіть на VM. Варто зауважити, що Apple надає інструменти безкоштовно, але платити доведеться за підписку розробника. Для написання програм під iPhone пропонується використовувати Objective-C. При цьому є можливість писати так само і на C, і C++ (для цього необхідно змінювати розширення файлів з .M на .Mm). Правда при цьому повністю звільнитись від Obj-C не вдасться, майже весь API розрахований саме на Obj-C, виключення складають наприклад OpenGL (хоча для його ініціалізації доведеться використовувати кілька рядків коду на Obj-C), так само повністю доступні стандартні бібліотеки C/C++. Налагодження додатка відбувається за допомогою середовища XCode і емулятора iPhone встановленого в ній.

Для написання додатків під Symbian можна використовувати мову програмування C++. Розробка для Symbian OS (якщо говорити про C++) зазвичай ведеться на ПК. Середа розробки – звична для багатьох програмістів Visual Studio, це також можуть бути IDE Metrowerks CodeWarrior Development Studio, Borland C++ BuilderX Mobile Edition, Carbide. C++ (відносно нова IDE, створена компанією Nokia на базі Eclipse), забезпечена додатковими інструментальними пакетами (SDK). Розробнику

доступні практично всі звичні можливості, як створення програмне забезпечення (ПЗ), так і налагодження (трасування, перегляд змінних, стека викликів, структур класів). Відлагодження програми запускається в емуляторі Symbian OS. Цю підсистему правильніше було б назвати симулятором, оскільки імітуються не апаратні засоби, а лише програмне оточення (відповідні API операційної системи, реалізовані поверх API Win32). При цьому програмні модулі, які завантажуються в емулятор, являють собою виконані файли для архітектури x86, відповідне ПЗ для цільової платформи формується після підсумкової компіляції. Це передбачає певну специфіку, але сьогодні емулятор забезпечує досить високу ступінь подібності та проблеми виникають лише при створенні програм, які нестандартно використовують API.

Для розробки під Android можна використовувати середовище Android Studio або Eclipse з встановленим плагіном ADT. Розробка ведеться на мові програмування Java. Є можливість налагодження з використанням емулятора вбудованого в Android Studio або безпосередньо на мобільному пристрої з ОС Android. Існують різні версії SDK, які використовуються для написання коду для різних версій Android. Підтримується майже повна зворотна сумісність версій. Крім розробки на мові Java підтримується можливість більш низькорівневої розробки з використанням Android NDK (Native Development Kit) на мові C/C++. На частку мобільних пристроїв на базі Android припадає 50 відсотків всіх користувачів. Створення додатків для цієї платформи відрізняється від розробки сайту. В даному випадку також приділяється увага інтерфейсу – дружньому та інтуїтивно зрозумілому. Щоб не зникало бажання використовувати додаток кожен день, важливо опрацювати прототипи і дизайн програми. Розробка додатків для Android вимагає від розробників глибоких знань і розуміння принципів ефективної взаємодії користувача з додатком, законів дизайну.

Платформа Android надає розробникам найбільшу свободу вибору ОС, на якій розробляти додатки. Android розробники можуть використовувати

Windows, Mac або Linux. BlackBerry користувачі можуть вибирати між Mac, Windows і Linux. Якщо ви хочете розробити для iOS і пристроїв Windows, єдиним варіантом є Mac і Windows відповідно.

Основні принципи проектування програмного забезпечення для мобільних пристроїв незмінні незалежно від платформи та операційної системи, встановленої на ній. Головними відмінностями є технічні особливості, такі як інструментарій, API і SDK, парадигми проектування інтерфейсу.

Для розробки автоматизованої системи ведення сімейного бюджету обрано платформу Android, так як на сьогоднішній день вона є найбільш популярною серед користувачів мобільних додатків.

1.4.2. Порівняння мов програмування

На даний момент існує багато мов програмування, на яких можна вести розробку додатків для мобільних ОС: C#, C++, Java, Objective-C. Також існує доволі багато різноманітних додатків, що дозволяють інтегрувати модулі, написані іншими мовами програмування, але це не є повноцінною розробкою. До таких мов належить php, JavaScript.

Зважаючи на популярність і простоту Java, переважна кількість розробок проводиться саме з її допомогою. Java [7] – об'єктно-орієнтована мова програмування, випущена компанією Sun Microsystems у 1995 році як основний елемент платформи Java. Синтаксис дуже схожий на синтаксис мов сімейства C, що дозволяє програмістам на C, C++ або C# дуже швидко перейти на Java без особливих труднощів. Розробка на Java є найпопулярнішою, бо ця мова найбільш повно реалізована під Android, має багато додаткових модулів, методичного матеріалу і дозволяє створювати додатки будь-якого рівня. Для роботи можна обрати будь-яку середу розробки – підтримка Java реалізована усюди. Також мова не є складною для вивчення і вдосконалення навичок. Серед недоліків є низька продуктивність коду на цій мові а також дуже великі системні вимоги до продуктів, що вимушує програміста або в певних питаннях чимось жертвувати в бік

продуктивності, що стає помітно в ігрових проектах, або підвищувати вимоги для пристроїв, що звужує коло потенціальних користувачів. Також треба зазначити, що Java дозволяє програмістам використовувати багато різноманітних засобів для підвищення ефективності: робота с потоками, автоматичне очищення ресурсів і доволі велика кількість бібліотек для розширення функціоналу або ж спрощення певних дій. Також варто зазначити, що ПЗ, написане на Java під ОС Android також працюватиме на ОС WindowsPhone і BlackBerry, така підтримка з'явилася відносно нещодавно, але певні додатки вже можуть бути встановлені без додаткових налаштувань. Власне, це є перевагою саме мови Java – програми, написані на Java, будуть працювати на будь-якому пристрої з Java Virtual Machine, а розробка інтерфейсу за допомогою XML файлів доволі широко підтримується в різноманітних ОС. Таким чином, розробляючи додаток на Java + XML, отримуємо повністю робочий і функціонуючий додаток на 3-х популярних ОС.

Для дуже ресурсоємних проектів використовується більш «легка» і адаптована для цих задач мова C#. Розробку можна вести за допомогою Unity, яка дозволить ефективно програмно управляти ресурсами і цим досягають вражаючих результатів в ігровій індустрії навіть на мобільних пристроях. Також програми, написані за допомогою C# легко модифікуються модулями на C++, JavaScript і C, що дозволяє значно розширити можливості, а в певних ситуаціях порівнювати ефективність того чи іншого способу, обираючи найефективніший. Проте, розробка на C# під Android має дуже мало документації, методичних матеріалів і підручників, що значно ускладнює розробку, вимушуючи витрачати багато часу на вивчення мови. Також доволі незручним фактором є те, що програма, розроблена на C, працюватиме на обмеженій кількості пристроїв навіть серед пристроїв на Android – лише на нових версіях ОС додатки будуть працювати стабільно і без труднощів.

Також серед мов програмування варто зазначити Objective-C – мову для програмування на iOS. На жаль, підтримка розробки на Objective-C під ОС Android мінімальна – є лише певні засоби, які зможуть запустити додаток написаний на Objective-C на пристрої з Android, проте це не гарантує нормального функціювання додатку і значно знижує його продуктивність, чим підвищує вимоги до пристрою. Також серед недоліків Objective-C варто зазначити доволі слабку реалізацію роботи в потоках, і синтаксис, що доволі сильно відрізняється від C (незважаючи на те, що Objective-C розроблений на базі мови C). Усі ці фактори роблять розробку на Objective-C під ОС Android неефективною і майже неможливою, тому розглядати цю мову як мову для програмування під ОС (принаймні, на сьогодні) не має сенсу.

Варто також зазначити, що розробка на Java займає значно менше часу, в порівнянні з C#, що є дуже важливим фактором. І це зумовлено не характером задач, що розв'язуються, а саме зручністю і рівнем адаптованості мови до задачі: Java має багато вже написаних бібліотек, компонентів, додатків, які на C# відсутні і їх доводиться писати власноруч або ж шукати інші шляхи розв'язку задачі.

Обираючи мову програмування треба зважати на подальші плани: робота з мережею, експортування даних в інші додатки, адаптація на інші платформи, тощо. Кожна мова дасть певні переваги в певних аспектах, а деякі ускладнить, саме тому треба одразу встановити вимоги. Дуже важливо одразу вирішити, чи буде ПЗ переноситися на інші платформи, чи будуть інтегруватися якісь специфічні модулі – іноді вибір тієї чи іншої мови робить такі моменти неможливими і значно знижує потенціал майбутнього додатка.

Також часто залежить вибір мови в залежності від наявних середовищ розробки: іноді добре вивчена програмістом мова не дає змогу вести розробку через нестачу середовищ, що примушує обирати іншу мову, яка має необхідне середовище. Написання на ОС Android не має такої проблеми – найкращі IDE доступні безкоштовно і доволі швидко розвиваються і надають користувачу нові функції і полегшують розробку, а кількість матеріалу по

налаштуванню і поліпшення середи для власних потреб дозволять кожному працювати так, як йому зручно і з максимальним комфортом.

Зважаючи на предметну область, було вирішено вести розробку додатку на Android за допомогою мови програмування Java. Велика кількість етапів розробки там проходить інтуїтивно зрозуміло, дуже велика кількість навчального матеріалу, зрозумілий і знайомий інтерфейс також позитивно впливають на швидкість адаптації до нової мови програмування. Усі необхідні елементи програмуються мовою Java, інтерфейс розроблюється в форматі XML, що є дуже поширеною формою і дозволить експортувати цей інтерфейс далі і працювати с ним під різними ОС на різних пристроях. Робота з мережею в такому випадку не є дуже складною, а деякі функції реалізовані в самій мові, що також позитивно впливатиме на терміни розробки. Практично відсутні ресурси (усі дані зберігаються на віддаленому сервері і отримуютсья за допомогою невеличких php функцій) і невеликі об'єми розрахунків дають змогу не хвилюватися за пам'ять і навантаження, повністю зробивши акцент на швидкості і функціональності розробки.

1.4.3. Вибір технології розробки

Кожна сучасна мобільна платформа надає інструментарій для розробників – SDK (software development kit), який дозволяє отримати доступ практично до всіх сервісів пристрою. Розробники SDK прагнуть спростити процес створення додатків шляхом вирішення типових завдань, з якими сторонні розробники стикаються в повсякденній практиці.

Технології, які лежать в основі кожної з SDK, як правило, розрізняються досить сильно. Наприклад, для програми на Android SDK розробляються на мові програмування Java та C або C#.

Вибір розробників SDK для Android на користь Java цілком виправданий, тому що ця мова на даний момент є, очевидно, найпопулярнішою. Разом з тим, вона багата можливостями і може з успіхом застосовуватися в областях від програмування простих пристроїв до розробки величезних корпоративних додатків. Багатство можливостей

пред'являє розробнику підвищені вимоги до знань мови, але також представляє користувачу широкі можливості розробки і значно спрощує його задачу.

Інструменти для розробки під Android доступні на всіх популярних ОС (Windows, Linux, Mac, Oracle). Тестувати програми можна на будь-якому підтримуваному пристрої без будь-яких обмежень або ж емуляторі, що має підтримку необхідного функціоналу.

До переваг такої технології розробки можна віднести високу продуктивність написаного ПЗ – усе, що писалося програмістом, можна було оптимізувати і воно працює на пристрої, не залежить від сторонніх факторів і надає користувачу найбільш швидку (порівняно з WEB) і стабільну (порівняно з гібридною технологію) роботу додатку.

Також варто зазначити, що такі додатки доволі легко тестувати і перевіряти на різноманітні покажчики: безпосередня розробка може бути миттєво перевірена вбудованими засобами, доволі легке і швидке розповсюдження дозволить протестувати ПЗ в режимі online і знайти його слабкі і сильні сторони.

Дуже великою перевагою є наявність дуже великої кількості програм для розробки додатків за допомогою Android SDK: Eclipse [8], Android Studio, Unity, Visual Studio і так далі. Кожне середовище розробки передбачає певні переваги для програміста, розраховані саме на його вид діяльності і спеціально під це налаштовані, так що середовище розробки також впливає на подальшу розробку.

Розглядаючи обране мною середовище розробки додатків на Android – Android Studio [9] – можна зазначити, що ця IDE, по-перше, безкоштовна, по-друге, дуже потужна і забезпечує усім необхідним (починаючи від вбудованого емулятора для тестування розробки, закінчуючи модулями для роботи з мережею), що потрібно для розробки в обраній області.

Розробка Android Studio передбачала швидку адаптацію до роботи навіть для початківців: усі елементи макетів можна швидко розмістити на

екрані, с цього-ж екрану почати писати обробники на цей елемент, і, в решті решт, протестувати отриманий елемент на емуляторі Android-пристрою, що дозволяє не тільки швидко вести розробку, але й проводити налагодження програми. Велика кількість автоматично доданих конструкцій, які програміст може відредагувати «під себе», або залишити в такому виді, також економлять багато сил і часу, що також є дуже великою перевагою цієї IDE над іншими середовищами розробки цього ж класу.

Сама по собі мова Java дає великі можливості з урахування того, що вона є кросплатформовою, а Android Studio працює с кодом на Java і макетами на XML ми отримуємо універсальну, потужну і дуже функціональне середовище для розробки на будь-який пристрій на ОС Android. Також приємною особливістю Java є те, що вона дуже легко інтегрується с елементами коду C#, що дозволяє використовувати готові блоки при розробці ПЗ – таким прикладом є робота с гео-локацією, камерою, акселерометром.

Варто також зазначити, що дана технологія розробки виграє ще й великою кількістю різноманітної документації, що дозволить в найкоротші терміни вивчити необхідні модулі і моментально повернутися до праці, не сповільнюючи термінів розробки і не знижуючи якість продукту.

1.5. Постановка задачі

Задля досягнення поставленої мети роботи необхідно вирішити такі задачі:

– проаналізувати особливості розробки бази даних для мобільних додатків;

– розробити структуру бази даних;

– розробити моделі мобільного додатку;

– розробити структуру мобільного додатку;

– розробити базу даних автоматизованої системи ведення сімейного бюджету;

– розробити алгоритм функціонування мобільного додатку;

– розробити інтерфейс автоматизованої системи;

– здійснити програмну реалізацію мобільного додатку;

1.6. Висновки

В розділі проаналізовано обрану предметну область, виявлено проблему, яка потребує розв’язання, а також наведено особливості класифікації мобільних додатків та визначено, до якого виду належить додаток, що розробляється. Проведено аналіз засобів системи ведення сімейного бюджету.

Проведено технічний аналіз основних аспектів, пов’язаних з розробкою мобільного додатку, що дозволяє зробити висновок, що розробка програмного продукту є придатною на сьогодні.

2 РОЗРОБКА МОДЕЛЕЙ І СТРУКТУР МОБІЛЬНОГО ДОДАТКУ

2.1 Аналіз принципів розробки мобільних додатків

Android використовує спеціальні механізми опису дій засновані на Intent. При необхідності виконати дію (виклик, відправка пошти, SMS) викликається Intent. Для обміну даними між додатками використовується Content Providers. Зазвичай для створення проекту використовуються Android SDK, Java і IDE.

Кожна мова програмування – унікальна і копіює методи і класи. У парадигмі програмування, кожна з мов пов'язана одна з однією концепціями, принципами і абстракціями, що визначають фундаментальний стиль програмування. Рідними мовами для Android є мови розмітки XML, C ++ і C #.

Ефективні методи програмування під Android [10]:

- програмування під Android – це Java, додатки якої транслюються в байт-код, що виконується віртуальною Java-машиною JVM;

- використання розмітки RelativeLayout і властивість «fill_parent» для цієї розмітки;

- використання порожних елементів розмітки TextField з нульовою висотою і шириною з параметром «centerInParent» рівним «true» для вирівнювання елементів по центру екрану;
- використання елегантних способів доступу до даних, таких як values [SensorManager.DATA_X];
- використання методів onPause()/onResume() для збереження або закриття всього того, що цього вимагає;
- використання кольорової розмітки для об'єктів. Встановлення кольору фону для деяких об'єктів. Це дозволяє виділити помилки і знаходити їх швидше;
- використання IDE, яка відповідає особистим уподобанням і вимогам розробника.
- використання кількох робочих моніторів для програмування. Додавання вікна додатків на кількох екранах (вікно IDE, емулятор, прев'ю і документація);
- використання Intents за допомогою окремого методу;
- уникнення створення безлічі об'єктів. Використання існуючих об'єктів;
- використання пулів потоків (thread pools), вбудованих класів AsyncTask;
- зберігання даних в SQLite, якщо вони займають великий об'єм;

Існують принципи розробки продуктивних додатків під Android. Основні з них такі: необхідно економити апаратні ресурси, ефективно працювати з виділенням пам'яті, протоколювати і аналізувати хід виконання додатку, уникати зайвих об'єктів і створювати методи статичними; для констант, класів необхідно використовувати модифікатор static final і не використовувати enum там, де модифікатор не вписується.

2.2 Аналіз особливостей розробки бази даних

Сучасні інформаційні технології побудовані на принципах взаємодії з базами даних. Бази даних грають велику роль у житті більшості людей. Вони

поширені всюди, починаючи від невеликих підприємств, які утримують власні бази товарів, працівників, партнерів, постачальників і закінчуючи базами даних глобального масштабу – держави чи кількох держав, наприклад, реєстраційні дані про людину чи автомобіль. Бази даних дозволяють ефективно розмістити та упорядкувати дані, що дозволить отримати ефективний пошук та взаємодію.

Тому вони займають провідне місце на ринку програмних продуктів.

В розробленій автоматизованій системі необхідно ефективно зберігати інформації про фінансовий облік. Для вирішення цієї задачі у мобільному додатку необхідно використати концепцію баз даних.

База даних – сукупність даних, що зберігаються у відповідності зі схемою даних і відображають стан об'єктів та їх взаємозв'язків, маніпулювання якими виконуються відповідно до правил засобів моделювання даних.

Бази даних є потужним і зручним способом зберігання і користування інформацією. Вони життєво необхідні у величезній кількості учбових закладів, установ, організацій, підприємств. Важливість баз даних важко переоцінити.

Реляційна база даних – колекція пов'язаних таблиць. Різниця між базою даних і реляційної базою даних знаходиться в будові таблиць. Якщо вести облік компанії на серії таблиць Excel, то робота йде тільки з базою даних. Можна отримувати інформацію з кожної таблиці, але тоді не буде можливості використовувати інформацію з однієї таблиці в якості основи для запиту інформації, що міститься на іншому аркуші. У реляційній базі даних, таблиці побудовані так, що є логічний зв'язок між ними. На підставі інформації, яку знайдено в одній таблиці, можна отримати відповідну інформацію з іншої таблиці.

В Android використовується база даних SQLite – це досить популярна, легка і швидка реляційна база даних [11]. Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер, тобто движок SQLite не є

окремим процесом, з яким взаємодіє додаток, а надає бібліотеку, з якою програма компілюється і движок стає складовою частиною програми. Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому пристрої, на якому виконується додаток. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції весь файл, що зберігає базу даних, блокується; ACID-функції досягаються зокрема за рахунок створення файлу-журналу.

Якщо перед розробником постає питання зберігання великого обсягу даних, то відповідь буде цілком очевидна - потрібно використовувати базу даних. Для цих цілей в андроїд є підтримка sqlite засобами якої, можна вирішити переважна більшість завдань, проте використовувати її часто не зовсім зручно. Навіть для самих простих операцій вставки / отримання даних, доводиться писати багато однотипного коду, що може привести не тільки до незручностей, але і до помилок. І ось тут нам на допомогу приходить бібліотека Room [12] - деякий прошарок між API для взаємодії з базою і вашим кодом.

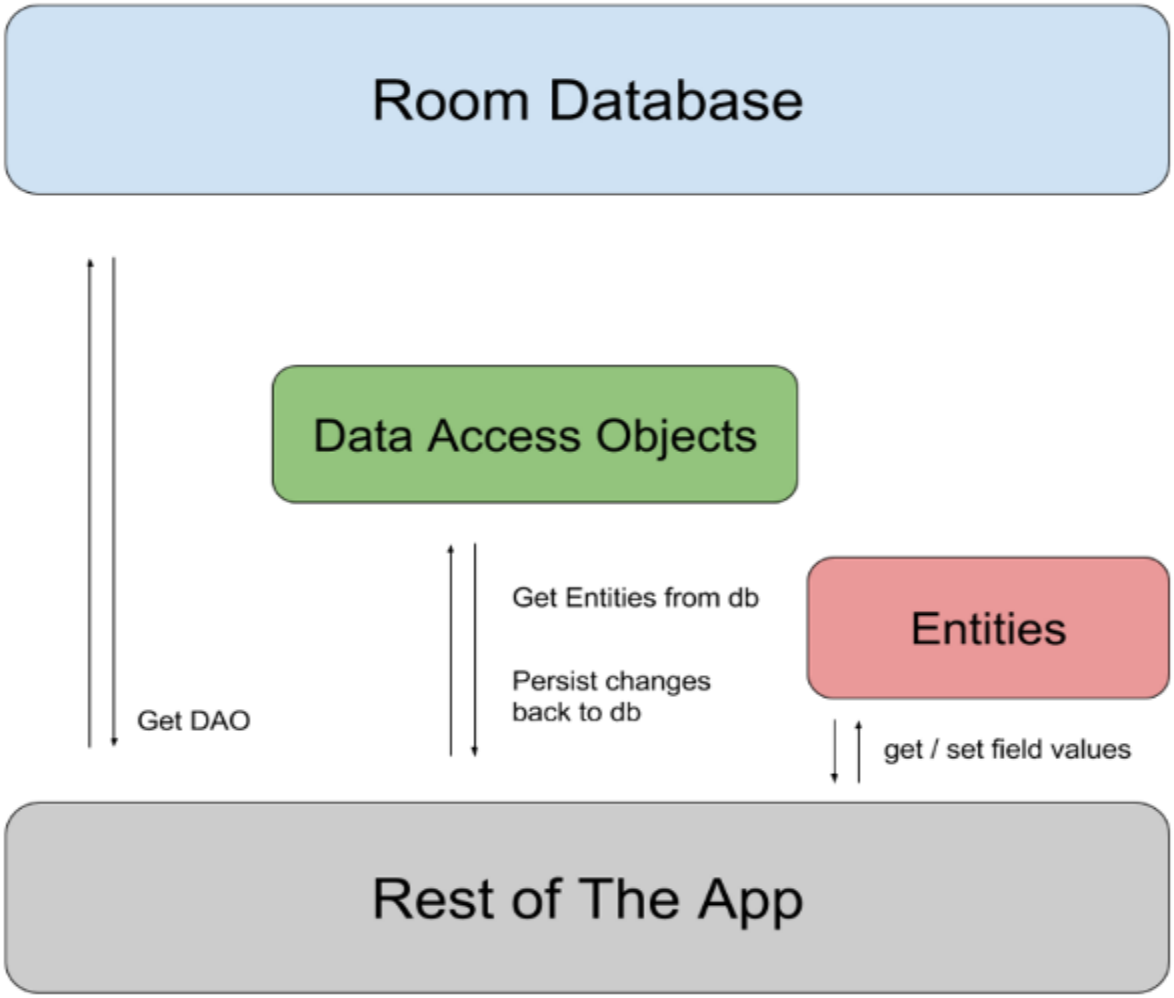

Рисунок 2.1 – Схема ахітектури Room

Якщо раніше доводилося використовувати незручні API (деякі вимагають по аж сім параметрів), заповнювати даними ContentValues, робити обхід записів витягуючи дані з Cursor, то тепер все спростилося. Ці рутинні речі зробить Room, причому ще на стадії компіляції проекту і плюс до того, зробить деякі перевірки дозволяють уникнути помилок. При цьому, ніхто не забороняє програмісту користуватися всією міццю мови запитів якщо це раптом стане необхідно.

Існують три основні типи компонентів які входять до складу Room: Entity - об'єкт який необхідно зберігати в базі даних. Для кожного Entity створюється окрема таблиця в базі даних. DAO - об'єкт використовується для будь-яких маніпуляцій з Entity. Використовується для додавання, видалення, зміни та отримання даних з бази. Database - клас бази даних, в ньому

визначені Entity і DAO які відносяться до даної базі, інформація про версії, правила поновлення і т.д.

2.3 Розробка моделей мобільного додатку

Для роботи мобільного додатку необхідна база даних для зберігання інформації про баланс та операції користувача. За допомогою мобільного додатку відображується інформація користувача, і він може маніпулювати своїми даними.

2.3.1 Розробка структури бази даних

Однією з обов'язкових умов при розробці баз даних є її нормалізація. Нормалізація схеми бази даних — покроковий процес розбиття одного відношення (на практиці: таблиці) відповідно до алгоритму нормалізації на декілька відношень на базі функціональних залежностей.

Нормальна форма – властивість відношення в реляційної моделі даних, що характеризує його з точки зору надмірності, яка потенційно може призвести до логічно помилкових результатів вибірки або зміни даних. Нормальна форма визначається як сукупність вимог, яким має задовольняти відношення.

Таким чином, схема реляційної бази даних переходить у першу, другу, третю і так далі нормальні форми. Якщо відношення відповідає критеріям нормальної форми n, та всім попереднім нормальним формам, тоді вважається, що це відношення знаходиться у нормальній формі рівня n.

У роботі базу даних приведено до третьої нормальної форми, тобто, тоді вона відповідає вимогам першої та другої нормальних форм, а саме:

- перша нормальна форма:

1) кожна таблиця повинна мати основний ключ: мінімальний набір колонок, які ідентифікують запис;

2) уникнення повторень груп (категорії даних, що можуть зустрічатись різну кількість разів у різних записах), правильно визначаючи не-ключові атрибути;

3) роздільність: кожен атрибут повинен мати лише одне значення, а не множину значень;

- друга нормальна форма:

1) схема бази даних повинна відповідати вимогам першої нормальної форми;

2) дані, що повторно з'являються в декількох рядках, виносяться в окремі таблиці;

- третя нормальна форма:

1) схема бази даних повинна відповідати всім вимогам другої нормальної форми;

2) будь-яке поле, що залежить від основного ключа та від будь-якого іншого поля, має виноситись в окрему таблицю.

2.3.2 Розробка ієрархічної моделі мобільного додатку

У плані обробки взаємодії між інтерфейсом користувача і його логікою Android слідує архітектурному шаблону «Model-View-ViewModel» (MVVM) [13].

Model-View-ViewModel – це шаблон проектування додатків для розділення коду інтерфейсу користувача і іншого коду. За допомогою MVVM декларативно визначається інтерфейс користувача (у нашому випадку, використовуючи XML) і використовується розмітка прив'язки даних, щоб пов'язати його з іншими рівнями, що містять дані і команди користувача.

Шаблон MVVM організовує код так, що можна змінювати окремі його частини, не впливаючи на інші. Це дає багато переваг, серед яких:

- можливість використання ітеративного, довільного стилю написання коду;
- спрощене тестування модулів;
- більш ефективне використання інструментів проектування;
- підтримка взаємодії в команді.

При використанні шаблону MVVM Android-додаток ділиться на такі частини:

1. Інтерфейс, який розробляється за допомогою технології XML.
2. Логіка користувацького інтерфейсу реалізується розробником як компонент ViewModel.
3. Функціональні зв'язки між інтерфейсом користувача і ViewModel реалізуються через Біндинг (bindings). Біндинги можуть бути написані в коді або визначені декларативним шляхом (Android використовує обидва типи).

View – базовий клас для всіх віджетів користувацького інтерфейсу. Інтерфейс Android-додатку являє собою дерево екземплярів нащадків цього класу.

Клас Activity і його підкласи містять логіку, що реалізує інтерфейс користувача. Цей клас відповідає ViewModel в архітектурному шаблоні Model-View-ViewModel (MVVM). Відношення між підкласом Activity і інтерфейсом користувача – це відношення один до одного; зазвичай кожен підклас Activity має тільки один пов'язаний з ним користувацький інтерфейс, і навпаки.

У роботі мобільний додаток містить чотири рівнів моделей представлення підкласів класу Activity. Перший рівень (екран-заставка) – зазвичай перше що бачить користувач на екрані (у нашому випадку мобільного телефону). Це зображення яке з'являється під час завантаження програми. другий рівень додатку – рівень бюджету користувача. Тут він може додавати і видаляти транзакції або переглянути баланс. Наступний рівень – рівень обліку фінансів. Користувач може додавати здійснені ним витрати або отримані доходи. Обравши одну з цих функцій, користувач переходить до останнього п’ятого рівня, де він додає детальну інформацію про витрати, доходи.

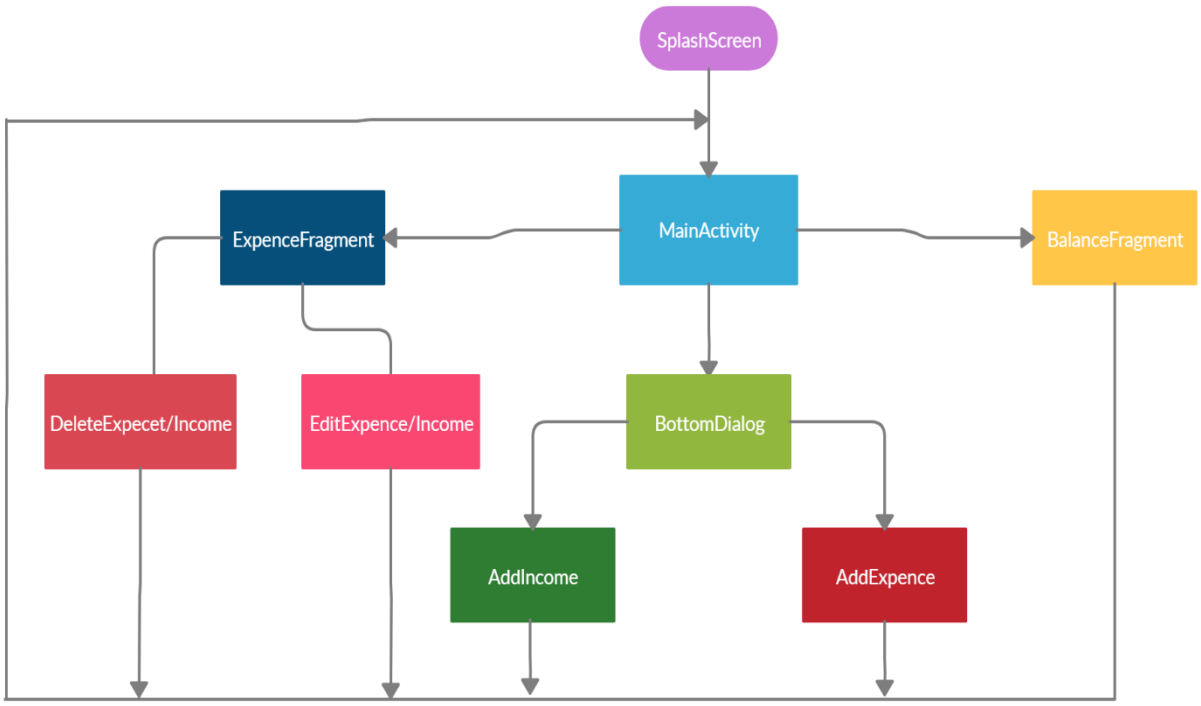

Рисунок 2.2 – Ієрархічна модель мобільного додатку

2.3.3 Розробка алгоритму роботи мобільного додатку

Алгоритм – набір інструкцій, що описують порядок дій виконавця для досягнення результату рішення задачі за визначене число дій.

Для створення алгоритму необхідно знати:

- повний набір вихідних даних завдання (початковий стан об'єкта);
- мета створення алгоритму (кінцевий стан об'єкта);
- систему команд виконавця (тобто набір команд, які виконавець розуміє і може виконати).

Існує два методи розробки складних алгоритмів.

Метод послідовної деталізації завдання («зверху-вниз») полягає в тому, що вихідна складна задача розбивається на підзадачі. Кожна з підзадач розглядається і вирішується окремо. Якщо які-небудь з задач складні, вони також розбиваються на підзадачі. Процес продовжується до тих пір, поки підзадачі не зведуться до елементарних. Вирішення окремих підзадач потім збираються в єдиний алгоритм розв'язання вихідної задачі. Метод широко використовується, тому що дозволяє вести розробку загального алгоритму

одночасно кільком програмістам, які вирішують локальні підзадачі. Це необхідна умова швидкої розробки програмних продуктів.

Складальний метод («знизу-вверх») полягає у створенні безлічі програмних модулів, що реалізують рішення типових задач. При вирішенні складного завдання програміст може використовувати розроблені модулі в якості допоміжних алгоритмів (процедур).

Кожний алгоритм повинен відповідати ряду загальних вимог [14], а саме:

1) дискретність – алгоритм повинен представляти процес вирішення завдання як послідовне виконання деяких простих кроків. При цьому для виконання кожного кроку алгоритму потрібно кінцевий відрізок часу, тобто перетворення вихідних даних у результат здійснюється в часі дискретно;

2) детермінованість (визначеність) – у кожен момент часу наступний крок роботи однозначно визначається станом системи. Таким чином, алгоритм видає один і той же результат (відповідь) для одних і тих же початкових даних.

У сучасному трактуванні у різних реалізацій одного і того ж алгоритму має бути ізоморфний граф. З іншого боку, існують імовірнісні алгоритми, в яких наступний крок роботи залежить від поточного стану системи і випадкового числа, що генерується. Однак при включенні методу генерації випадкових чисел в список «початкових даних», імовірнісний алгоритм стає підвидом звичайного;

3) зрозумілість – алгоритм повинен включати тільки ті команди, які доступні виконавцю і входять в його систему команд;

4) завершеність – при корректно заданих початкових даних алгоритм повинен завершувати роботу і видавати результат за кінцеве число кроків.

5) масовість (універсальність) – алгоритм повинен бути застосовний до різних наборі початкових даних;

6) результативність – завершення алгоритму певними результатами;

7) алгоритм містить помилки, якщо призводить до отримання неправильних результатів або не дає результатів зовсім;

8) алгоритм не містить помилок, якщо він дає правильні результати для будь-яких допустимих початкових даних.

Алгоритми в залежності від мети, початкових умов завдання, шляхів її вирішення, визначення дій виконавця поділяються наступним чином:

– механічні алгоритми – задають певні дії, позначаючи їх в єдиній і достовірної послідовності, забезпечуючи тим самим однозначний необхідний або шуканий результат, якщо виконуються ті умови процесу, завдання, для яких розроблений алгоритм;

– гнучкі алгоритми, наприклад стохастичні, тобто імовірнісні та евристичні;

– імовірнісний алгоритм дає програму рішення задачі кількома шляхами або способами, що призводять до ймовірного досягненню результату;

– евристичний алгоритм (від грецького слова «еврика») – алгоритм, що використовує різні розумні міркування без строгих обґрунтувань;

– лінійний алгоритм – набір команд (вказівок), виконуваних послідовно в часі один за одним;

– розгалужений алгоритм – алгоритм, який містить хоча б одну умову, в результаті перевірки якої може здійснюватися поділ на кілька паралельних гілок алгоритму;

– циклічний алгоритм – алгоритм, що передбачає багаторазове повторення однієї і тієї ж дії (одних і тих же операцій) над новими вихідними даними. Цикл програми – послідовність команд, яка може виконуватися багато разів (для нових початкових даних) до задоволення деякої умови;

Алгоритм розробленої автоматизованої системи є лінійним, так як виконується послідовно один за одним.

2.4 Висновки

У розділі проаналізовано принципи розробки мобільних додатків та особливості розробки бази даних під платформу Android. Також розроблено моделі мобільного додатку і проаналізовано його архітектуру з використанням шаблону проектування Model-View-ViewModel, головною перевагою якого є розділення коду інтерфейсу користувача і іншого коду. Здійснено розробку структури бази даних та наведено алгоритм роботи, який належить до лінійного типу.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОБІЛЬНОГО ДОДАТКУ

3.1 Розробка бази даних Room Database

За замовчуванням, для кожного поля в класі, буде створюватися свій стовпець у відповідній таблиці. Ім'я стовпця буде збігатися з ім'ям поля, а тип даних стовпця з типом даних поля. Все що потрібно, це вказати яке поле є ключем, за допомогою анотації @PrimaryKey. Всі поля які будуть зберігатися в таблиці, повинні бути доступні для запису ззовні. Наприклад можна додати конструктор який дозволить їх задати або зробити сетер на кожен з них. Ну або зовсім зробити їх публічними (що з точки зору інкапсуляції звичайно не завжди вітається). Якщо буде використовуватися конструктор, то імена його параметрів повинні відповідати іменам полів які зберігаються в таблиці. Типи даних зрозуміло теж повинні збігатися, порядок параметрів не важливий, саме вміст конструктора теж ролі не грає (код скомпілюється навіть якщо він буде порожнім). Так само необхідно надати можливість читати поля за допомогою відповідних геттеров, без них буде помилка компіляції. Також, необхідно додати в додаток DAO за допомогою якого ми будемо читати / писати дані в таблицю.

Частина коду DAO наведено нижче:

```
@Dao
public interface TransactionDao {
    @Query("select * from transactionTable order by date DESC")
    LiveData<List<TransactionEntry>> loadAllTransactions();
    @Query("select * from transactionTable where id = :id")
    LiveData<TransactionEntry> loadExpenseById(int id);
    @Query("select sum(amount) from transactionTable where
transactionType =:transactionType")
    int getAmountByTransactionType(String transactionType);
    @Query("select sum(amount) from transactionTable where
transactionType =:transactionType and  date between :startDate and
:endDate")
    int getAmountbyCustomDates(String transactionType,long
startDate,long endDate);
```

```
    @Query("select sum(amount) from transactionTable where
category=:category")
    int getSumExpenseByCategory(String category);
    @Query("select sum(amount) from transactionTable where
category=:category and date between :startDate and :endDate")
    int getSumExpenseByCategoryCustomDate(String category,long
startDate, long endDate);
    @Query("select min(date) from transactionTable ")
    long getFirstDate();
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void insertExpense(TransactionEntry transactionEntry);
    @Delete
    void removeExpense(TransactionEntry transactionEntry);
    @Update(onConflict = OnConflictStrategy.REPLACE)
    void updateExpenseDetails(TransactionEntry transactionEntry);
}
```

Після компіляції проекту, буде створено клас всередині якого знаходиться реалізація всіх цих методів і залишиться лише отримати його і можна читати / писати в таблицю з транзакціями. Тепер необхідно створити клас бази даних:

```
@Database(entities = TransactionEntry.class,version = 1,exportSchema =
false)
@TypeConverters(DateConverter.class)
public abstract class AppDatabase extends RoomDatabase {
    private static final String LOG_TAG =
AppDatabase.class.getSimpleName();
    private static final Object LOCK = new Object();
    private static final String DATABASE_NAME = "TransactionDb";
    private static AppDatabase sInstance;
    public static AppDatabase getInstance(Context context){
        if (sInstance == null){
            synchronized (LOCK){
                Log.d(LOG_TAG,"Creating new database instance");
                sInstance =
Room.databaseBuilder(context.getApplicationContext(),
                        AppDatabase.class,AppDatabase.DATABASE_NAME)
                        .build();
            }
```

```
        }
        Log.d(LOG_TAG,"Getting the database instance");
        return sInstance;
    }
    public abstract TransactionDao transactionDao();
```

Клас повинен бути абстрактним і успадковуватися від RoomDatabase. В анотації @Database вказані entities котрі входять до складу бази даних, а так само її версію. У сам клас необхідно додати абстрактний публічний метод який повертає посилання на AppDataBase. Тепер, коли класи створені, можна починати роботу з базою даних.

Room намагається зберегти кожне поле Entity класу в таблицю бази даних, але часто може виникати ситуація, коли це не потрібно. Для цього необхідно анотувати таке поле за допомогою @Ignore. Нижче наведена реалізація таблиці:

```
@Entity(tableName = "transactionTable")
public class TransactionEntry {
    @PrimaryKey(autoGenerate = true)
    private int id;
    private int amount;
    private String category;
    private String description;
    private Date date;                    // COMPLETED: 13-09-2018 Add
appropriate type converter
    private String transactionType; //to decide whether income or
expense
    @Ignore
    public TransactionEntry(int amount,String category,String
description,Date date,String transactionType){
        this.amount = amount;
        this.category = category;
        this.description = description;
        this.date = date;
        this.transactionType=transactionType;
    }
    public TransactionEntry(int id,int amount,String category,String
description,Date date,String transactionType){
```

```
        this.id = id;
        this.amount = amount;
        this.category = category;
        this.description = description;
        this.date = date;
        this.transactionType=transactionType;
    }
    public int getAmount() {
        return amount;
    }
    public void setAmount(int amount) {
        this.amount = amount;
    }
    public String getCategory() {
        return category;
    }
    public void setCategory(String category) {
        this.category = category;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
    public Date getDate() {
        return date;
    }
    public void setDate(Date date) {
        this.date = date;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getTransactionType() {
        return transactionType;
```

```
    }
    public void setTransactionType(String transactionType) {
        this.transactionType = transactionType;
    }
}
```

3.2 Розробка мобільного додатку

3.2.1 Реалізація логіки додатку

При розробці додатку під Android головним класом є клас Activity. Він представляє візуальну активність додатку і визначає дії, які може виконувати користувач. У процесі роботи класу Activity, спочатку створюється об’єкт класу Activity, потім він запускається, відпрацьовує та знищується, а користувач смартфона переходить до нового об’єкта.

Протягом життєвого циклу Activity може перебувати в одному з трьох станів [15]:

1) Активне і виконується – цей користувацький інтерфейс знаходиться на передньому плані (на вершині стека Activity).

2) Призупинене – якщо даний інтерфейс користувача втратив фокус, але все ще видимий. У такому стані ніякий код не виконується.

3) Завершене – якщо інтерфейс користувача невидимий. У такому стані код не виконується.

При створенні нової активності, наприклад, при запуску програми, Android викликає метод OnCreate. У цьому методі проводиться початкове налаштування activity. Зокрема, створюються об'єкти візуального інтерфейсу. Цей метод отримує об'єкт Bundle, який містить колишній стан activity, якщо він був збережений. Якщо activity заново створюється, то даний об'єкт має значення null. Якщо ж activity вже раніше була створена, але перебувала у загальмованому стані, то bundle містить зв'язану з activity інформацію.

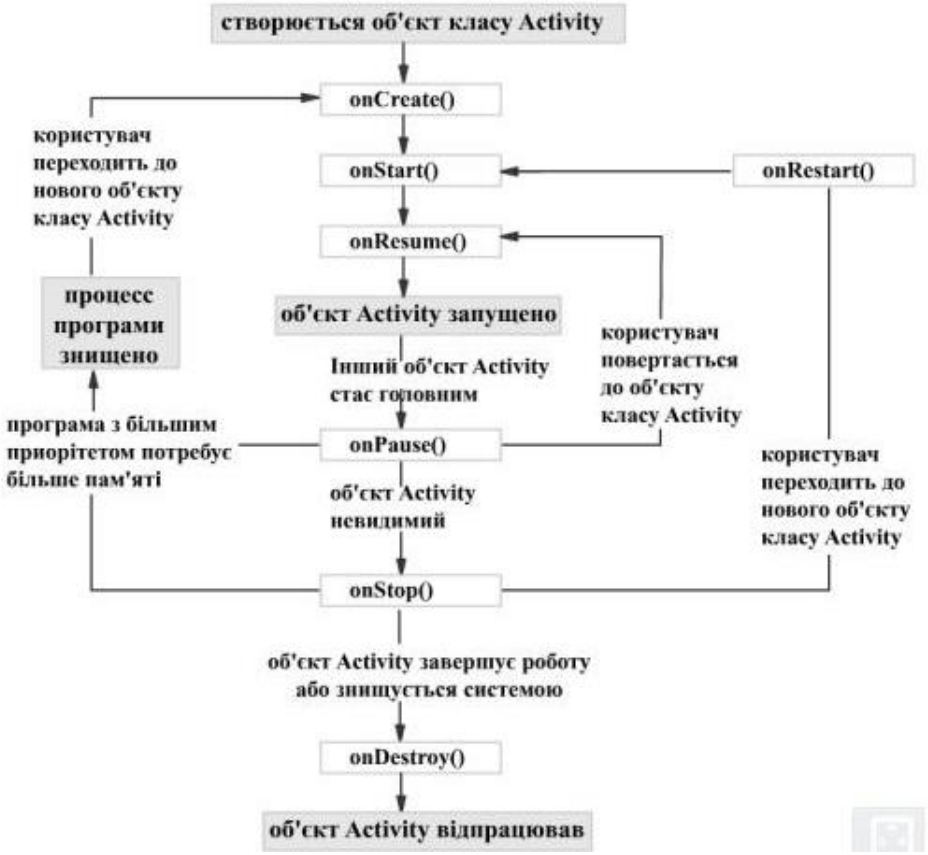

Рисунок 3.1 – Процес роботи класу Activity

Потім викликається метод OnStart, а activity переходить у "видимий" стан.

А при виклику методу OnResume activity відображається на екрані, і користувач може з нею взаємодіяти.

Якщо користувач вирішить перейти до іншої активності, то система викликає метод OnPause. Після цього, якщо користувач вирішить повернутися до колишньої активності, то система викличе знову метод OnResume, і activity знову з'явиться на екрані. Інакше, якщо activity більше невидима, то викликається метод OnStop.

Якщо після виклику методу OnStop користувач вирішить повернутися до колишньої activity, тоді система викличе метод OnRestart. Ну і завершується робота активності викликом методу OnDestroy, який виникає або, якщо система вирішить вбити activity, або при виклику методу finish().

При проектуванні та розробці програмного забезпечення має відзначатися його швидкість та ефективність роботи. Дуже важливо постійно

приділяти увагу оптимізації програмного забезпечення на всіх етапах його розробки.

Запустивши розроблений додаток, першим користувач бачить Splash Screen (екран-заставка) - Зазвичай перше що бачить користувач на екрані Це зображення яке з'являється під час завантаження програми. Splash screen потрібен, тому що додатку необхідно кілька секунд на організацію, перш ніж користувач зможе рухатися далі. Незалежно від того, як швидко додаток запускається, майже завжди існує кілька невеликих завдань, які операційна система, або код додатка повинні виконати, перш ніж користувач зможе почати користуватися додатком. Частина коду реалізації SplashScreen та MainActivity наведена нижче:

```
public class SplashScreen extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        startActivity(new Intent(SplashScreen.this,
MainActivity.class));
        // close splash activity
        finish();
    }
}
public class MainActivity extends AppCompatActivity {
    private ViewPager mViewPager;
    public static FloatingActionButton fab;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mViewPager=findViewById(R.id.container);
        setupViewPager(mViewPager);
        TabLayout tabLayout=findViewById(R.id.tabs);
        tabLayout.setupWithViewPager(mViewPager);
         fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
```

```
                new
CustomBottomSheetDialogFragment().show(getSupportFragmentManager(),
"Dialog");
            }
        });
    }
    private void setupViewPager(ViewPager viewPager){
        SectionsPageAdapter adapter=new
SectionsPageAdapter(getSupportFragmentManager());
        adapter.addFragment(new ExpenseFragment(),"Операции");
        adapter.addFragment(new BalanceFragment(),"Баланс");
        viewPager.setAdapter(adapter);
    }
}
```

Після SplashScreen користувача переносить до фрагменту ExpenseFragment (Вікно операцій),де можна переглянути історію доходів/витрат а також додавати, редагувати чи видаляти транзакції. Свайпнувши вправо користувач переходить до інформативного графіку, де можливо передивитися залишок коштів та витрати за обраний період.

Реалізація коду ExpenseFragment:

```
public class ExpenseFragment extends Fragment {
        private static final String LOG_TAG =
ExpenseFragment.class.getSimpleName();
        private RecyclerView rv;
        private List<TransactionEntry> transactionEntries;
    private CustomAdapter customAdapter;
    public TransactionViewModel transactionViewModel;
    private AppDatabase mAppDb;
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
@Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        final View
view=inflater.inflate(R.layout.fragment_expense,container,false);
        rv=view.findViewById(R.id.transactionRecyclerView);
        rv.setHasFixedSize(true);
        transactionEntries = new ArrayList<>();
```

```
        rv.setLayoutManager(new LinearLayoutManager(getActivity()));
        mAppDb = AppDatabase.getInstance(getContext());
        new ItemTouchHelper(new ItemTouchHelper.SimpleCallback(0,
ItemTouchHelper.RIGHT) {
            @Override
            public boolean onMove(RecyclerView recyclerView,
RecyclerView.ViewHolder viewHolder, RecyclerView.ViewHolder target) {
                return false;
            }

            @Override
            public void onSwiped(final RecyclerView.ViewHolder
viewHolder, int swipeDir) {
                AppExecutors.getInstance().diskIO().execute(new
Runnable() {
                    @Override
                    public void run() {
                        int position =
viewHolder.getAdapterPosition();
                        List<TransactionEntry> transactionEntries =
customAdapter.getTransactionEntries();

mAppDb.transactionDao().removeExpense(transactionEntries.get(position)
);
                    }
                });
                Snackbar.make(view,"Транзакция
удалена",Snackbar.LENGTH_LONG).show();
            }
        }).attachToRecyclerView(rv);
        setupViewModel();
        return view;
    }
    public void setupViewModel(){
        transactionViewModel = ViewModelProviders.of(this)
                .get(TransactionViewModel.class);
        transactionViewModel.getExpenseList()
                .observe(this, new Observer<List<TransactionEntry>>()
{
```

```
                    @Override
                    public void onChanged(@Nullable
List<TransactionEntry> transactionEntriesFromDb) {
                        Log.i(LOG_TAG,"Actively retrieving from DB");
                        transactionEntries = transactionEntriesFromDb;
                        for (int i =0 ; i < transactionEntries.size()
; i++){
                            String description =
transactionEntries.get(i).getDescription();
                            int amount =
transactionEntries.get(i).getAmount();
                        }

                        customAdapter=new
CustomAdapter(getActivity(),transactionEntries);
                        rv.setAdapter(customAdapter);
                    }
                });
    }
}
```

3.2.2 Реалізація розмітки Android-додатку

Під час розробки Android-додатку, візуальна частина вікон реалізується окремо від логіки програми. Код розмітки екрану знаходиться у файлах з розширенням .xml у папці проекту, яка має назву «layout».

Розмітка для Android програми будується з використанням ієрархії View і ViewGroup об'єкти [16]. View об'єкти це віджети для інтерфейсу користувача, такі як кнопки або текстові поля і ViewGroup це невидимий вид контейнерів, які визначають розташування дочірніх представлень, як наприклад, в сітці або вертикальному списку.

Існує кілька стандартних типів розміток:

1. FrameLayout – є найпростішим типом розмітки. Зазвичай це порожній простір на екрані, який можна заповнити тільки дочірніми об'єктами View або ViewGroup. Усі дочірні елементи FrameLayout прикріплюються до верхнього лівого кута екрана. У розмітці FrameLayout не можна визначити різне розташування для дочірнього об'єкта. Наступні

дочірні об'єкти View будуть просто малюватися поверх попередніх компонентів, частково або повністю затінюючи їх, якщо об'єкт, що знаходиться зверху, непрозорий, тому єдиний дочірній елемент для FrameLayout зазвичай розтягнутий до розмірів батьківського контейнера.

2. LinearLayout – вирівнює всі дочірні об'єкти в одному напрямку – вертикально або горизонтально. Напрямок задається за допомогою атрибута орієнтації android: orientation. Усі дочірні елементи поміщаються в стек один за іншим, так що вертикальний список представлень буде мати тільки один дочірній елемент в рядку незалежно від того, наскільки широким він є. Горизонтальне розташування списку буде розміщувати елементи в один рядок з висотою, рівній висоті найвищого дочірнього елемента списку.

3. TableLayout – позиціонує свої дочірні елементи в рядки і стовпці, як це звикли робити веб-майстри в тезі table. TableLayout не відображає лінії обрамлення для їх рядків, стовпців або комірок. TableLayout може мати рядки з різною кількістю комірок. При формуванні розмітки таблиці деякі комірки при необхідності можна залишати порожніми. При створенні розмітки для рядків використовуються об'єкти TableRow, які є дочірніми класами TableLayout (кожен TableRow визначає єдиний рядок в таблиці). Рядок може не мати комірок або мати одну і більше комірок, які є контейнерами для інших об'єктів. У комірку допускається вкладати інший TableLayout або LinearLayout.

4. RelativeLayout – дозволяє дочірнім компонентам визначати свою позицію щодо батьківського компонента або відносно сусідніх дочірніх елементів (ідентифікатора елемента). У RelativeLayout дочірні елементи розташовані так, що якщо перший елемент розташований по центру екрана, інші елементи, вирівняні відносно першого елемента, будуть вирівняні відносно центру екрана. При такому розташуванні, при оголошенні розмітки в XML-файлі, елемент, на який будуть посилатися для позиціонування інші об'єкти представлення, має бути оголошений раніше, ніж інші елементи, що звертаються до нього за його ідентифікатором.

5. GridLayout – На перший погляд він може здатися схожим на TableLayout. Але насправді він набагато зручніший і функціональніший. Розмітка відноситься до класу android.widget.GridLayout і має колонки, рядки, клітинки як в TableLayout, але при цьому елементи можуть гнучко налаштовуватися. У GridLayout для будь-якого компонента можна вказати рядок колонку, і в цьому місці таблиці він буде розміщений. Вказувати елемент для кожної комірки не знадобиться, це потрібно робити тільки для тих комірок, де дійсно повинні бути компоненти. Компоненти можуть розтягуватися на кілька елементів таблиці. Більш того, в одну комірку можна помістити кілька компонентів.

Всі описані розмітки є підкласами ViewGroup і успадковують властивості, визначені в класі View.

Для підключення створеної розмітки використовується код в методі onCreate():

```
setContentView(R.layout.main),
```

де main – назва головної розмітки.

Код розмітки класу MainActivity наведений нижче:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main_content"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context=".activities.MainActivity">

    <android.support.design.widget.AppBarLayout
        android:id="@+id/appbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="@dimen/appbar_padding_top"
        android:theme="@style/AppTheme.AppBarOverlay">
```

```
        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:layout_weight="1"
            android:background="?attr/colorPrimary"
            app:layout_scrollFlags="scroll|enterAlways"
            app:popupTheme="@style/AppTheme.PopupOverlay"
            app:title="@string/app_name">

        </android.support.v7.widget.Toolbar>

        <android.support.design.widget.TabLayout
            android:id="@+id/tabs"
            android:layout_width="match_parent"
            android:layout_height="wrap_content">
        </android.support.design.widget.TabLayout>
    </android.support.design.widget.AppBarLayout>

    <android.support.v4.view.ViewPager
        android:id="@+id/container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
/>

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center|bottom"
        android:layout_margin="@dimen/fab_margin"
        app:backgroundTint="#ffffff"
        app:srcCompat="@android:drawable/ic_input_add" />

</android.support.design.widget.CoordinatorLayout>
```

MainActivity реалізовано за допомогою розмітки CoordinatorLayout, так як у нашому випадку її найзручніше використовувати.

Оголошення інтерфейсу користувача в файлах XML дозволяє відокремити інтерфейс додатку від коду, що означає, що можна змінювати визначення інтерфейсу без зміни і перекомпіляції коду. Наприклад, у додатку можуть бути визначені розмітки у файлах XML для різних орієнтацій екрану, різних розмірів пристроїв, різних мов і т.д. Крім того, оголошення розмітки в XML дозволяє легше візуалізувати структуру інтерфейсу і полегшує налагодження.

3.3 Розробка інтерфейсу програмного продукту

Розробка дизайну і інтерфейсу мобільного додатку – найважливіший етап створення програмного продукту. Від даної роботи залежить те, як користувач сприйме додаток, чи сподобається воно йому, чи буде додаток зручним в експлуатації, і чи стане він популярним.

Інтерфейс – це зовнішня оболонка додатка разом із програмами керування доступом і іншими схованими від користувача механізмами керування, що дає можливість працювати з документами, даними й іншою інформацією, що зберігається в комп’ютері чи за його межами. Головна мета будь-якого додатка – забезпечити максимальну зручність і ефективність роботи з інформацією: документами, базами даних, графікою, зображеннями.

Дизайн мобільного додатку повинен бути максимально простий і зрозумілий. З огляду на те, що мобільні пристрої мають невеликий за розмірами екран, при проектуванні додатків для них не можна керуватися тими ж правилами, що і для ПК. Основні вимоги до дизайну інтерфейсу мобільних додатків [17]:

- мінімум елементів. Не слід перевантажувати невеликий простір дисплея мобільного пристрою великою кількістю об'єктів. Необхідно намагатися вмістити максимум функціоналу в лаконічний і дружній інтерфейс.

Ця вимога є головною для розробки будь-якого ПЗ, а не тільки мобільного;

- управління сучасними телефонами з сенсорними екранами здійснюється за допомогою чуттєвого дотику. З огляду на те, що площа дотику пальця значно більше розмірів покажчика комп'ютерної миші, а також стілуса, інтерфейс не повинен містити дрібних елементів. По-перше, вони погано помітні на невеликому екрані. По-друге, торкаючись деякого ділянки дисплея, користувач може натиснути не той елемент, який йому потрібен, що в кращому випадку може призвести до зниження зручності використання програми, а в гіршому – до небажаних наслідків;

- розмір всіх написів повинен бути достатнім для того, щоб користувач міг прочитати їх з відстані не менше 30 см;

- найбільш важливі і часто використовувані елементи інтерфейсу повинні знаходитися в центрі екрану і мати достатній розмір для того, щоб виділятися серед інших;

- при перенесенні додатків з ПК на мобільний платформу можна обмежитися створенням зменшеної копії додатка. Необхідно оптимізувати весь інтерфейс, прибрати всі зайві елементи, згрупувавши схожі по функціоналу.

При великій кількості об'єктів слід зробити додаткові «вікна», що змінюють один одного на дисплеї. На відміну від ПК, на мобільних платформах під вікнами розуміються елементи інтерфейсу, що займають весь простір екрану пристрою. Користувач здійснює переходи між такими вікнами за допомогою графічних елементів- навігаторів, або перетягуючи їх за допомогою пальця (в залежності від тієї або іншої платформи і переваг творців додатка).

При проектуванні дизайну мобільного додатку може виникнути необхідність враховувати культурні особливості регіону, для якого воно призначене (читання справа наліво або правильний підбір колірної гами). У

правильно спроектованому мобільному додатку повинні поєднуватися три основні властивості:

1. Зручність у використанні (інтуїтивний дизайн, об'єднання та використання всіх можливостей мобільного пристрою). Найбільш популярними мобільними платформами є iPhone і Android. Вони мають багато спільного.

Уміло спроектований додаток буде включати в себе той функціонал, який використовує особливості кожної з них.

2. Мобільний додаток повинен бути цікавим користувачу.

3. Корисність. Максимальний рейтинг мають ті додатки, які здатні бути дійсно потрібними.

Під час розроблення мобільного додатку було дотримано всіх вище наведених правил та спроектовано оптимальний на вигляд і для застосування інтуїтивно зрозумілий інтерфейс. На рисунку 3.2 наведено вікно додатку «Баланс». На рисунку 3.3 зображено сторінку витрат мобільного додатку, де користувач може додавати витрати, які він здійснив.

Рисунок 3.2 – Баланс

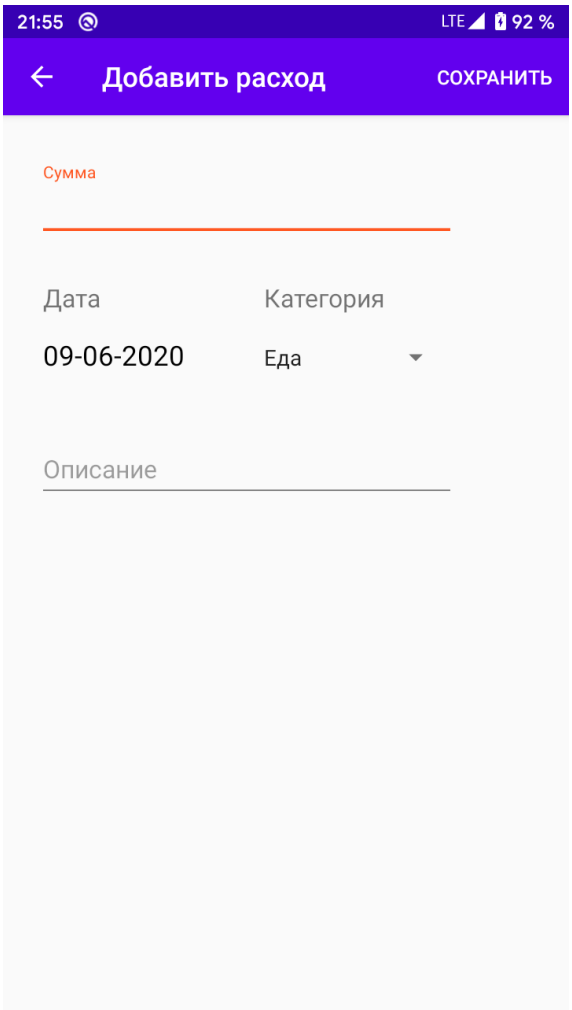

Рисунок 3.3 – Додавання витрат

Під час проектування інтерфейсу користувача необхідно визначити структуру діалогу, який буде відбуватися між користувачем і мобільним пристроєм. Розглянуті нижче чотири варіанти структури діалогу є різновидами структури типу «питання – відповідь», проте кожна з них має свої особливості і найбільш зручна для певного класу задач.

Діалог типу «питання – відповідь». Структура діалогу типу «питання – відповідь» (Q&A) заснована на аналогії із звичайним інтерв'ю. Система бере на себе роль інтерв'юера і отримує інформацію від користувача у вигляді

відповідей на запитання. Це найбільш відома структура діалогу; всі діалоги, керовані пристроєм, в тій чи іншій мірі складаються з питань, на які користувач відповідає. Однак у структурі Q&A цей процес виражений явно. У кожній точці діалогу система виводить як підказку одне питання, на яке користувач дає одну відповідь. Залежно від отриманої відповіді система може вирішити, яке наступне питання задавати. Структура Q&A не гарантує мінімального обсягу введення, оцінюваного за кількістю натискань клавіш, однак при відповідному підборі скорочень можна зменшити будь-яку надмірність. Разом з тим, структура Q&A володіє одним істотним недоліком. Навіть якщо введення відбувається досить швидко, для людини, яка вже знає, які питання задає система, і які відповіді потрібно на них давати, відповідати на всю серію питань досить утомливо.

Діалоги на основі меню. Меню є найбільш популярним варіантом організації запитів на введення даних під час діалогу, керованого комп'ютером. Існує кілька основних форматів представлення меню на екрані:

– список об'єктів, обираних прямою вказівкою, або вказівкою номера (або мнемонічного коду);

– меню у вигляді блоку даних;

– меню у вигляді рядка даних;

– меню у вигляді піктограм.

Меню у вигляді рядка даних може з'являтися вгорі чи внизу екрану і часто залишається в цій позиції протягом всього діалогу. Таким чином, за допомогою меню зручно відображати можливі варіанти даних для введення, доступних в будь-який час роботи з системою. Додаткові меню у вигляді блоків даних «спливають» на екрані в позиції, яка визначається поточним положенням покажчика, або «випадають» безпосередньо з рядка меню верхнього рівня. Ці меню зникають після вибору варіанта. Меню у вигляді піктограм являє собою безліч об'єктів вибору, розкиданих по всьому екрану; часто об'єкти вибору містять графічне представлення варіантів роботи. Якщо діалог складається виключно з меню, можна реалізувати послідовний

інтерфейс, при якому користувач застосовує тільки пристрої для вказівки; проте така постійність рідко досягається на практиці.

Меню – це найбільш зручна структура діалогу для непідготовлених користувачів; жорстка черговість відкриття і ієрархічна вкладеність меню може викликати роздратування професіонала, уповільнювати його роботу. Традиційна структура меню недостатньо гнучка й не повною мірою узгоджується з методами адаптації діалогу, такими, наприклад, як випереджаюче введення, за допомогою якого можна прискорити темп роботи підготовленого користувача.

Діалог на основі екранних форм. Як структура типу «питання – відповідь», так і структура типу меню припускають обробку єдиної відповіді на кожному кроці діалогу. Діалог на основі екранних форм допускає обробку на одному кроці діалогу декількох відповідей. На практиці форми використовуються в основному там, де облік якоїсь діяльності вимагає введення достатньо стандартного набору даних. Людина, що заповнює форму, може вибирати послідовність відповідей, тимчасово пропускати деяке питання, повертатися назад для корекції попередньої відповіді і навіть «порвати бланк» і почати заповнювати новий. Вона працює з формою до тих пір, поки не заповнить її повністю і не передасть системі. Програмна система може перевіряти кожну відповідь безпосередньо після введення або почекати і вивести список помилок лише після заповнення форми повністю.

Структура діалогу на основі екранної форми забезпечує високий рівень підтримки користувача: для кожного питання форми можуть бути передбачені повідомлення про помилки та довідкова інформація. Користувачеві можна також надати допомогу, включивши деякі елементи формату відповіді в питання або в поле відповіді. Ця структура дозволяє підвищити швидкість введення даних в порівнянні зі структурою типу «питання – відповідь» і маніпулювати більш широким діапазоном вхідних даних, ніж меню; крім того, з нею можуть працювати користувачі будь-якої кваліфікації.

Діалог на основі командної мови. Структура діалогу на основі командної мови настільки ж поширена, як і структура типу меню. Вона дуже часто використовується в операційних системах і розташовується на іншому кінці спектру структур діалогу стосовно структури типу меню. Історично це перша з реалізованих структур діалогу. При такій організації діалогу програмна система не виводить нічого, крім постійної підказки (запрошення на введення команди), яка означає готовність системи до роботи. Кожну команду вводять з нового рядка і зазвичай закінчують натисканням клавіші «Enter». Відповідальність за правильність задавання команд лягає на користувача. Система інформує про неможливість виконання невірної команди, як правило, не пояснюючи причин.

Подібно меню, діалог на базі команд зручний для введення керуючих повідомлень, проте він забезпечує більш широкі можливості вибору в будь-якій точці діалогу і не вимагає ієрархічної організації обслуговуючих його програм.

Структура на основі мови команд за своїми можливостями найшвидша і найбільш гнучка з усіх структур діалогу. Більшість користувальницьких інтерфейсів на базі «природної» мови реалізується за допомогою мов команд з дуже великим набором ключових слів. Підготовлений користувач відчуває задоволення від відчуття того, що він керує системою, а не навпаки. Однак ця структура не забезпечує користувача підтримкою, тому навіть підготовлені користувачі вважають, що дуже складно використовувати всі закладені в ній можливості. Більшість же користувачів добре знайомі тільки з вельми обмеженим набором засобів, з яким вони працюють регулярно.

Інтерфейс розробленої автоматизованої системи ведення сімейного бюджету поєднує в собі два типи діалогу: діалог типу меню і «питання – відповідь».

3.4 Висновки

У розділі розроблено базу даних мобільного додатку та наведено частину коду вікна авторизації користувача. Також розроблено розмітку

екрану Android-додатку та наведено частину коду реалізації розмітки вікна LoginActivity. Розроблено інтуїтивно-зрозумілий інтерфейс користувача та визначено, що він належить до діалогу типу меню.

ВИСНОВКИ

У дипломній роботі розроблено автоматизовану систему ведення сімейного бюджету як мобільного додатку. Сьогодні, коли людина постійно прагне бути на зв'язку, розробка мобільних додатків є дуже актуальною.

Для розробки мобільного додатку проаналізовано найбільш популярні існуючі аналоги. Також зроблено огляд особливостей класифікації мобільних додатків і визначено, що розроблений додаток належить до сервісних нативних додатків. Проведено варіантний аналіз засобів реалізації автоматизованої системи і обґрунтовано вибір середовища програмування Android Studio та мови програмування Java.

Проаналізовано принципи розробки мобільних додатків і обрано платформу Android для розробки автоматизованої системи, а також здійснено аналіз особливостей розробки бази даних.

Для реалізації мобільного додатку розроблено модель роботи автоматизованої системи, структуру бази даних та ієрархічну модель мобільного додатку. Також наведено алгоритм роботи, який належить до лінійного типу.

Розроблено базу даних та розмітку вікон Android-додатку. Розроблено інтуїтивно-зрозумілий інтерфейс користувача та визначено, що він належить до діалогу типу меню.

Додаток вирішує проблему ведення обліку сімейних і особистих витрат і доходів. Програма значно полегшує контроль за фінансовим станом користувача, так як за потребою надає доступ до інформації про поповнення бюджету, щоденні та щомісячні витрати, заплановані покупки.

Розроблений додаток сприятиме ефективному веденню бухгалтерського обліку сімейного бюджету.

ПЕРЕЛІК ПОСИЛАНЬ

1. Мобільна бухгалтерія. [Електронний ресурс] // URL: https://itc.ua/articles/mobilnaya-buhgalteriya-android-prilozheniya-dlya-ucheta-lichnyih-finansov/ (дата звернення: 09.05.2020).

2. ToshlFinance. [Електронний ресурс] // URL: http://itc.ua/articles/mobilnaya-buhgalteriya-android-prilozheniya-dlya-ucheta-lichnyih-finansov/ (дата звернення: 09.05.2020).

3. MoneyLover. [Електронний ресурс] // URL: http://www.monefy.me/ (дата звернення: 10.05.2020).

4. Monefy. [Електронний ресурс] // URL: https://monefy.me/ (дата звернення 10.05.2020)

5. FinancePM. [Електронний ресурс] // URL: http://finance.uramaks.com/login.html (дата звернення: 10.05.2020).

6. Технологія розробки мобільних додатків. [Електронний ресурс] // URL: http://lektsii.net/2-50017.html (дата звернення: 15.05.2020).

7. Мови програмування. [Електронний ресурс] // URL: http://zei.narod.ru/Comparison_C__Java_Cpp_3.pdf (дата звернення 17.05.2020).

8. Eclipse. [Електронний ресурс] // URL: https://uk.wikipedia.org/wiki/Eclipse (дата звернення 17.05.2020).

9. Android Studio. [Електронний доступ] // URL: https://uk.wikipedia.org/wiki/Android_Studio (дата звернення 17.05.2020).

10. Методологія розробки додатків. [Електронний ресурс] // URL: http://android.mobile-review.com/articles/22580/ (дата звернення 18.05.2020).

11. SQLite. [Електронний ресурс] // URL: https:// uk.wikipedia.org/wiki/SQLite (дата звернення 20.05.2020).

12. Room DataBase. [Електронний ресурс] // URL: https://developer.android.com/training/data-storage/room (дата звернення 20.05.2020)

13. Архітектура Android-додатків. [Електронний ресурс] // URL: http://http://android.mobile-review.com/articles/22580/ (дата звернення 21.05.2020).

14. Алгоритми. [Електронний ресурс] // URL: https:// uk.wikipedia.org/wiki/Алгоритм (дата звернення 22.05.2020).

15. Activity. [Електронний ресурс] // URL: http:// http://metanit.com/java/android/2.1.php (дата звернення 22.05.2020)

16. Layout. [Електронний ресурс] // URL: http:// developer.alexanderklimov.ru/android/theory/layout.php (дата звернення 25.05.2020).

17. Методологія проектування та інструментарій для створення мобільних додатків. [Електронний ресурс] // URL: http:// http://elar.tsatu.edu.ua/bitstream/123456789/1199/1/1733.pdf (дата звернення 30.05.2020).

ДОДАТОК А

Лістинг мобільного додатку

AddExpenseActivity.java

```
public class AddExpenseActivity extends AppCompatActivity {
    TextInputEditText amountTextInputEditText;
    TextInputEditText descriptionTextInputEditText;
    TextInputLayout amountTextInputLayout;
    TextInputLayout descriptionTextInputLayout;
    TextView dateTextView;
    LinearLayout dateLinearLayout;
    Spinner categorySpinner;
    ArrayList<String> categories;
    Calendar myCalendar;
    String description;
    Date dateOfExpense;
    private DatePickerDialog datePickerDialog;
    private static AppDatabase appDatabase;
    private static final String LOG_TAG =
AddExpenseActivity.class.getSimpleName();
    int amount;
    String categoryOfExpense;
    String categoryOfTransaction;
    String intentFrom;
    TransactionViewModel transactionViewModel;
    int transactionid;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_expense);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        amountTextInputEditText =
findViewById(R.id.amountTextInputEditText);
        descriptionTextInputEditText =
findViewById(R.id.descriptionTextInputEditText);
        amountTextInputLayout =
findViewById(R.id.amountTextInputLayout);
        descriptionTextInputLayout =
findViewById(R.id.descriptionTextInputLayout);
        dateTextView = findViewById(R.id.dateTextView);
        dateLinearLayout = findViewById(R.id.dateLinerLayout);
        categorySpinner = findViewById(R.id.categorySpinner);
        appDatabase =
AppDatabase.getInstance(getApplicationContext());
        transactionViewModel = ViewModelProviders.of(this)
                .get(TransactionViewModel.class);
        categories = new ArrayList<>();
        myCalendar = Calendar.getInstance();
        setDateToTextView();
        Intent intent = getIntent();
        intentFrom = intent.getStringExtra("from");
        if (intentFrom.equals(Constants.addIncomeString)) {
```

```
            categoryOfTransaction = Constants.incomeCategory;
            setTitle("Добавить доход");
            categories.add("Доход");
            categorySpinner.setClickable(false);
            categorySpinner.setEnabled(false);
            categorySpinner.setAdapter(new
ArrayAdapter<>(AddExpenseActivity.this,
android.R.layout.simple_list_item_1, categories));
        } else if
(intentFrom.equals(Constants.addExpenseString)) {
            categoryOfTransaction = Constants.expenseCategory;
            setTitle("Добавить расход");
            categories.add("Еда");
            categories.add("Одежда");
            categories.add("Здоровье");
            categories.add("Автомобиль");
            categories.add("Путешествия");
            categories.add("Ребенок");
            categories.add("Другое");
            categorySpinner.setAdapter(new
ArrayAdapter<>(AddExpenseActivity.this,
                    android.R.layout.simple_list_item_1,
categories));
        } else if
(intentFrom.equals(Constants.editIncomeString)) {
            setTitle("Редактировать доход");

amountTextInputEditText.setText(String.valueOf(intent.getIntExtr
a("amount", 0)));

amountTextInputEditText.setSelection(amountTextInputEditText.get
Text().length());

descriptionTextInputEditText.setText(intent.getStringExtra("desc
ription"));

descriptionTextInputEditText.setSelection(descriptionTextInputEd
itText.getText().length());
            transactionid=intent.getIntExtra("id",-1);
            SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-
yyyy");
            try {
                Date date =
sdf.parse(intent.getStringExtra("date"));
                myCalendar.setTime(date);
            } catch (ParseException e) {
                e.printStackTrace();
            }
            dateTextView.setText(intent.getStringExtra("date"));
            categoryOfTransaction = Constants.incomeCategory;
            categories.add("Доход");
            categorySpinner.setClickable(false);
            categorySpinner.setEnabled(false);
```

```
            categorySpinner.setAdapter(new
ArrayAdapter<>(AddExpenseActivity.this,
android.R.layout.simple_list_item_1, categories));
        } else if
(intentFrom.equals(Constants.editExpenseString)) {
            categoryOfTransaction = Constants.expenseCategory;
            setTitle("Редактировать расход");

amountTextInputEditText.setText(String.valueOf(intent.getIntExtr
a("amount", 0)));

amountTextInputEditText.setSelection(amountTextInputEditText.get
Text().length());

descriptionTextInputEditText.setText(intent.getStringExtra("desc
ription"));

descriptionTextInputEditText.setSelection(descriptionTextInputEd
itText.getText().length());
            dateTextView.setText(intent.getStringExtra("date"));
            transactionid=intent.getIntExtra("id",-1);
            SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-
yyyy");
            try {
                Date date =
sdf.parse(intent.getStringExtra("date"));
                myCalendar.setTime(date);
            } catch (ParseException e) {
                e.printStackTrace();
            }
            categories.add("Еда");
            categories.add("Одежда");
            categories.add("Здоровье");
            categories.add("Автомобиль");
            categories.add("Путешествия");
            categories.add("Ребенок");
            categories.add("Другое");
            categorySpinner.setAdapter(new
ArrayAdapter<>(AddExpenseActivity.this,
android.R.layout.simple_list_item_1, categories));

categorySpinner.setSelection(categories.indexOf(intent.getString
Extra("category")));
        }
        dateLinearLayout.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showDatePicker();
            }
        });
    }
    public void showDatePicker() {
```

```
        DatePickerDialog.OnDateSetListener dateSetListener=new
DatePickerDialog.OnDateSetListener() {
            @Override
            public void onDateSet(DatePicker view, int year, int
month, int dayOfMonth) {
                myCalendar.set(Calendar.YEAR, year);
                myCalendar.set(Calendar.MONTH, month);
                myCalendar.set(Calendar.DAY_OF_MONTH,
dayOfMonth);
                setDateToTextView();
            }
        };
        DatePickerDialog datePickerDialog=new
DatePickerDialog(AddExpenseActivity.this,dateSetListener,
                myCalendar.get(Calendar.YEAR),
myCalendar.get(Calendar.MONTH),
myCalendar.get(Calendar.DAY_OF_MONTH));

datePickerDialog.getDatePicker().setMaxDate(System.currentTimeMi
llis());
        datePickerDialog.show();
    }
    public void setDateToTextView() {
        Date date = myCalendar.getTime();
        SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-
yyyy");
        String dateToBeSet = sdf.format(date);
        dateTextView.setText(dateToBeSet);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater menuInflater = getMenuInflater();
        menuInflater.inflate(R.menu.add_expense_activty_menu,
menu);
        return super.onCreateOptionsMenu(menu);
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
            case R.id.saveButton:
                if
(amountTextInputEditText.getText().toString().isEmpty()
                        ||
descriptionTextInputEditText.getText().toString().isEmpty()) {
                    if
(amountTextInputEditText.getText().toString().isEmpty())
                        amountTextInputEditText.setError("Сумма
не введена");
                    if
(descriptionTextInputEditText.getText().toString().isEmpty())
```

```
descriptionTextInputEditText.setError("Пожалуйста, напишите
описание");
                } else {
                    amount =
Integer.parseInt(amountTextInputEditText.getText().toString());
                    description =
descriptionTextInputEditText.getText().toString();
                    dateOfExpense = myCalendar.getTime();
                    if
(intentFrom.equals(Constants.addIncomeString)
                            ||
intentFrom.equals(Constants.editIncomeString))
                        categoryOfExpense = "Доход";
                    else
                        categoryOfExpense =
categories.get(categorySpinner.getSelectedItemPosition());
                    final TransactionEntry mTransactionEntry =
new TransactionEntry(amount,
                            categoryOfExpense,
                            description,
                            dateOfExpense,
                            categoryOfTransaction
                    );

if(intentFrom.equals(Constants.addIncomeString)||intentFrom.equa
ls(Constants.addExpenseString)) {

AppExecutors.getInstance().diskIO().execute(new Runnable() {
                            @Override
                            public void run() {

appDatabase.transactionDao().insertExpense(mTransactionEntry);
                            }
                        });
                        InputMethodManager inputManager =
(InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);

inputManager.hideSoftInputFromWindow(getCurrentFocus().getWindow
Token(), InputMethodManager.HIDE_NOT_ALWAYS);

Snackbar.make(getCurrentFocus(),"Транзакция
добавлена",Snackbar.LENGTH_LONG).show();
                    }
                    else{
                        mTransactionEntry.setId(transactionid);

AppExecutors.getInstance().diskIO().execute(new Runnable() {
                            @Override
                            public void run() {

appDatabase.transactionDao().updateExpenseDetails(mTransactionEn
```

```
try);
                                }
                            });
                            InputMethodManager inputManager =
(InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);

inputManager.hideSoftInputFromWindow(getCurrentFocus().getWindow
Token(), InputMethodManager.HIDE_NOT_ALWAYS);

Snackbar.make(getCurrentFocus(),"Транзакция
обновленна",Snackbar.LENGTH_LONG).show();
                        }
                        new Handler().postDelayed(new Runnable() {
                            @Override
                            public void run() {
                                finish();
                            }
                        }, 1000);
                    }
                    break;
            }
            return true;
        }
}
```

CustomAdapter.java

```
public class CustomAdapter extends
RecyclerView.Adapter<CustomAdapter.ViewHolder> {
    Context context;
    private List<TransactionEntry> transactionEntries;
    private AppDatabase appDatabase;
    public CustomAdapter(Context context, List<TransactionEntry>
transactionEntries){
        this.context=context;
        this.transactionEntries=transactionEntries;
    }
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        View view =
LayoutInflater.from(context).inflate(R.layout.list_item,parent,f
alse);
        return new ViewHolder(view);
    }
    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int
position) {
        String amount;

holder.categoryTextViewrv.setText(transactionEntries.get(positio
n).getCategory());
```

```
if(transactionEntries.get(position).getTransactionType().equals(
Constants.incomeCategory)) {

amount="+"+transactionEntries.get(position).getAmount();
            holder.amountTextViewrv.setText(amount);

holder.amountTextViewrv.setTextColor(Color.parseColor("#aeea00")
);
        }
        else {
            amount="-
"+transactionEntries.get(position).getAmount();
            holder.amountTextViewrv.setText(amount);

holder.amountTextViewrv.setTextColor(Color.parseColor("#ff5722")
);
        }
        SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
        String
dateToBeSet=sdf.format(transactionEntries.get(position).getDate(
));
        holder.dateTextViewrv.setText(dateToBeSet);

holder.descriptionTextViewrv.setText(transactionEntries.get(posi
tion).getDescription());
    }
    @Override
    public int getItemCount() {
        if (transactionEntries == null ||
transactionEntries.size() == 0){
            return 0;
        } else {
            return transactionEntries.size();
        }
    }
    public List<TransactionEntry> getTransactionEntries() {
        return transactionEntries;
    }
    public class ViewHolder extends RecyclerView.ViewHolder {
        TextView categoryTextViewrv;
        TextView amountTextViewrv;
        TextView descriptionTextViewrv;
        TextView dateTextViewrv;
        public ViewHolder(View itemView) {
            super(itemView);

categoryTextViewrv=itemView.findViewById(R.id.categoryTextViewrv
);

amountTextViewrv=itemView.findViewById(R.id.amountTextViewrv);

descriptionTextViewrv=itemView.findViewById(R.id.descriptionText
```

```
Viewrv);

dateTextViewrv=itemView.findViewById(R.id.dateTextViewrv);
            appDatabase = AppDatabase.getInstance(context);
            itemView.setOnClickListener(new
View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    Intent intent=new
Intent(context,AddExpenseActivity.class);
                    SimpleDateFormat sdf=new
SimpleDateFormat("dd-MM-yyyy");
                    String
date=sdf.format(transactionEntries.get(getAdapterPosition()).get
Date());
                    if
(transactionEntries.get(getAdapterPosition()).getTransactionType
().equals(Constants.incomeCategory)) {
                        intent.putExtra("from",
Constants.editIncomeString);

intent.putExtra("amount",transactionEntries.get(getAdapterPositi
on()).getAmount());

intent.putExtra("description",transactionEntries.get(getAdapterP
osition()).getDescription());
                        intent.putExtra("date",date);

intent.putExtra("id",transactionEntries.get(getAdapterPosition()
).getId());
                    }
                    else {
                        intent.putExtra("from",
Constants.editExpenseString);

intent.putExtra("amount",transactionEntries.get(getAdapterPositi
on()).getAmount());

intent.putExtra("description",transactionEntries.get(getAdapterP
osition()).getDescription());
                        intent.putExtra("date",date);

intent.putExtra("category",transactionEntries.get(getAdapterPosi
tion()).getCategory());

intent.putExtra("id",transactionEntries.get(getAdapterPosition()
).getId());
                    }
                    context.startActivity(intent);
                }
            });
        }
```

```
    }
}
```

SectionPageAdapter.java

```
public class SectionsPageAdapter extends FragmentPagerAdapter {
    private final List<Fragment> FragmentList=new ArrayList<>();
    private final List<String> FragmentTitleList=new
ArrayList<>();
    public void addFragment(Fragment fragment,String title){
        FragmentList.add(fragment);
        FragmentTitleList.add(title);
    }
    public SectionsPageAdapter(FragmentManager fm) {
        super(fm);
    }
    @Nullable
    @Override
    public CharSequence getPageTitle(int position) {
        return FragmentTitleList.get(position);
    }
    @Override
    public Fragment getItem(int position) {
        return FragmentList.get(position);
    }
    @Override
    public int getCount() {
        return FragmentList.size();
    }
}
```

BalanceFragment.java

```
public class BalanceFragment extends Fragment implements
AdapterView.OnItemSelectedListener{
    private AppDatabase mAppDb;
    PieChart pieChart;
    Spinner spinner;
    private TextView balanceTv,incomeTv,expenseTv;
    private TextView dateTv;
    private int balanceAmount,incomeAmount,expenseAmount;
    private int
foodExpense,travelExpense,clothesExpense,childExpense,heathExpen
se,otherExpense,carExpense;
    long firstDate;
    ArrayList<ExpenseList> expenseList;
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
@Nullable ViewGroup container, @Nullable Bundle
savedInstanceState) {
        View
view=inflater.inflate(R.layout.fragment_balance,container,false)
;
        pieChart= view.findViewById(R.id.balancePieChart);
        spinner = view.findViewById(R.id.spinner);
```

```
        spinner.setOnItemSelectedListener(this);
        mAppDb = AppDatabase.getInstance(getContext());
        balanceTv = view.findViewById(R.id.totalAmountTextView);
        expenseTv =
view.findViewById(R.id.amountForExpenseTextView);
        incomeTv =
view.findViewById(R.id.amountForIncomeTextView);
        dateTv = view.findViewById(R.id.dateTextView);
        expenseList=new ArrayList<>();
        getAllBalanceAmount();
        setupPieChart();
        return view;

    }
    private void setupSpinner() {
        ArrayAdapter<CharSequence> arrayAdapter =
ArrayAdapter.createFromResource(getContext(),
                R.array.date_array,
                android.R.layout.simple_spinner_item);

arrayAdapter.setDropDownViewResource(android.R.layout.simple_spi
nner_dropdown_item);
        spinner.setAdapter(arrayAdapter);
    }
    @Override
    public void setUserVisibleHint(boolean isVisibleToUser) {
        super.setUserVisibleHint(isVisibleToUser);
        Log.i("fragment", String.valueOf(isVisibleToUser));
        if (isVisibleToUser){
            setupSpinner();
            fab.setVisibility(View.GONE);
        } else{
            fab.setVisibility(View.VISIBLE);
        }
    }
    private void setupPieChart() {
        AppExecutors.getInstance().diskIO().execute(new
Runnable() {
            @Override
            public void run() {
                if(spinner.getSelectedItemPosition()==0)
                    getAllPieValues();
                else if(spinner.getSelectedItemPosition()==1) {
                    try {
                        getWeekPieValues();
                    } catch (ParseException e) {
                        e.printStackTrace();
                    }
                }
                else if(spinner.getSelectedItemPosition()==2){
                    try {
                        getMonthPieValues();
                    } catch (ParseException e) {
```

```
                    e.printStackTrace();
                }
            }
            expenseList.clear();
            if(foodExpense!=0)
                expenseList.add(new
ExpenseList("Еда",foodExpense));
            if(clothesExpense!=0)
                expenseList.add(new
ExpenseList("Одежда",clothesExpense));
            if(heathExpense!=0)
                expenseList.add(new
ExpenseList("Здоровье",heathExpense));
            if(carExpense!=0)
                expenseList.add(new
ExpenseList("Автомобиль",carExpense));
            if(travelExpense!=0)
                expenseList.add(new
ExpenseList("Путешествия",travelExpense));
            if(childExpense!=0)
                expenseList.add(new
ExpenseList("Ребенок",childExpense));
            if(otherExpense!=0)
                expenseList.add(new
ExpenseList("Другое",otherExpense));
            }
        });
        AppExecutors.getInstance().mainThread().execute(new
Runnable() {
            @Override
            public void run() {
                List<PieEntry> pieEntries = new ArrayList<>();
                for(int i = 0 ; i <expenseList.size(); i++){
                    pieEntries.add(new
PieEntry(expenseList.get(i).getAmount(),expenseList.get(i).getCa
tegory()));
                }
                pieChart.setVisibility(View.VISIBLE);
                PieDataSet dataSet = new
PieDataSet(pieEntries,null);

dataSet.setColors(ColorTemplate.COLORFUL_COLORS);
                PieData pieData = new PieData(dataSet);
                pieData.setValueTextSize(16);
                pieData.setValueTextColor(Color.WHITE);
                pieData.setValueFormatter(new
PercentFormatter());
                pieChart.setUsePercentValues(true);
                pieChart.setData(pieData);
                pieChart.animateY(1000);
                pieChart.invalidate();
                pieChart.getDescription().setText("");
                Legend l=pieChart.getLegend();
```

```
l.setPosition(Legend.LegendPosition.LEFT_OF_CHART);
                //l.setXEntrySpace(8f);
                //l.setYEntrySpace(1f);
                //l.setYOffset(0f);
            }
        });
    }
    @Override
    public void onItemSelected(AdapterView<?> adapterView, View
view, int i, long l) {
        if(adapterView.getSelectedItemPosition()==0){
            getAllBalanceAmount();
            setupPieChart();
        }
        else if (adapterView.getSelectedItemPosition() == 1){
            //This week
            try {
                getWeekBalanceAmount();
                setupPieChart();
            }
            catch (ParseException e) {
                e.printStackTrace();
            }
        }
        else if(adapterView.getSelectedItemPosition()==2){
            //This month
            try {
                getMonthBalanceAmount();
                setupPieChart();
            } catch (ParseException e) {
                e.printStackTrace();
            }
        }
    }
    @Override
    public void onNothingSelected(AdapterView<?> adapterView) {
    }
    private void getAllPieValues(){
        foodExpense
=mAppDb.transactionDao().getSumExpenseByCategory("Еда");

clothesExpense=mAppDb.transactionDao().getSumExpenseByCategory("
Одежда");

heathExpense=mAppDb.transactionDao().getSumExpenseByCategory("Зд
оровье");

carExpense=mAppDb.transactionDao().getSumExpenseByCategory("Авто
мобиль");

travelExpense=mAppDb.transactionDao().getSumExpenseByCategory("П
утешествия");
```

```
childExpense=mAppDb.transactionDao().getSumExpenseByCategory("Ре
бенок");

otherExpense=mAppDb.transactionDao().getSumExpenseByCategory("Др
угое");
    }
    private void getWeekPieValues() throws ParseException {
        Calendar calendar;
        calendar=Calendar.getInstance();
        DateFormat df = new SimpleDateFormat("dd/MM/yyyy",
Locale.getDefault());
        String startDate = "", endDate = "";
        calendar.set(Calendar.DAY_OF_WEEK, Calendar.MONDAY);
        startDate = df.format(calendar.getTime());
        Date sDate=df.parse(startDate);
        final long sdate=sDate.getTime();
        calendar.add(Calendar.DATE, 6);
        endDate = df.format(calendar.getTime());
        Date eDate=df.parse(endDate);
        final long edate=eDate.getTime();
        foodExpense
=mAppDb.transactionDao().getSumExpenseByCategoryCustomDate("Еда"
,sdate,edate);

clothesExpense=mAppDb.transactionDao().getSumExpenseByCategoryCu
stomDate("Одежда",sdate,edate);

heathExpense=mAppDb.transactionDao().getSumExpenseByCategoryCust
omDate("Здоровье",sdate,edate);

carExpense=mAppDb.transactionDao().getSumExpenseByCategoryCustom
Date("Автомобиль",sdate,edate);

travelExpense=mAppDb.transactionDao().getSumExpenseByCategoryCus
tomDate("Путешествия",sdate,edate);

childExpense=mAppDb.transactionDao().getSumExpenseByCategoryCust
omDate("Ребенок",sdate,edate);

otherExpense=mAppDb.transactionDao().getSumExpenseByCategoryCust
omDate("Другое",sdate,edate);
    }
    private void getMonthPieValues() throws ParseException{
        Calendar calendar;
        calendar=Calendar.getInstance();
        DateFormat df = new SimpleDateFormat("dd/MM/yyyy",
Locale.getDefault());
        String startDate = "", endDate = "";
        calendar.set(Calendar.DAY_OF_MONTH,1);
        startDate = df.format(calendar.getTime());
        Date sDate=df.parse(startDate);
        final long sdate=sDate.getTime();
```

```
        calendar.set(Calendar.DAY_OF_MONTH,
calendar.getActualMaximum(Calendar.DAY_OF_MONTH));
        endDate = df.format(calendar.getTime());
        Date eDate=df.parse(endDate);
        final long edate=eDate.getTime();
        foodExpense
=mAppDb.transactionDao().getSumExpenseByCategoryCustomDate("Еда"
,sdate,edate);

clothesExpense=mAppDb.transactionDao().getSumExpenseByCategoryCu
stomDate("Одежда",sdate,edate);

heathExpense=mAppDb.transactionDao().getSumExpenseByCategoryCust
omDate("Здоровье",sdate,edate);

carExpense=mAppDb.transactionDao().getSumExpenseByCategoryCustom
Date("Автомобиль",sdate,edate);

travelExpense=mAppDb.transactionDao().getSumExpenseByCategoryCus
tomDate("Путешествия",sdate,edate);

childExpense=mAppDb.transactionDao().getSumExpenseByCategoryCust
omDate("Ребенок",sdate,edate);

otherExpense=mAppDb.transactionDao().getSumExpenseByCategoryCust
omDate("Другое",sdate,edate);
    }
    private void getAllBalanceAmount(){

       AppExecutors.getInstance().diskIO().execute(new
Runnable() {
           @Override
           public void run() {
               firstDate=mAppDb.transactionDao().getFirstDate();
           }
       });
        SimpleDateFormat df = new
SimpleDateFormat("dd/MM/yyyy");
        String first = df.format(new Date(firstDate));
        Date today=Calendar.getInstance().getTime();
        String todaysDate=df.format(today);
        String Date=first+" - "+todaysDate;
        dateTv.setText(Date);
        AppExecutors.getInstance().diskIO().execute(new
Runnable() {
            @Override
            public void run() {
                int income =
mAppDb.transactionDao().getAmountByTransactionType(Constants.inc
omeCategory);
                incomeAmount = income;
                int expense =
mAppDb.transactionDao().getAmountByTransactionType(Constants.exp
```

```
enseCategory;
                expenseAmount = expense;
                int balance = income - expense;
                balanceAmount = balance;
            }
        });
        AppExecutors.getInstance().mainThread().execute(new
Runnable() {
            @SuppressLint("SetTextI18n")
            @Override
            public void run() {

balanceTv.setText(String.valueOf(balanceAmount)+" \u20B4");
                incomeTv.setText(String.valueOf(incomeAmount)+"
\u20B4");

expenseTv.setText(String.valueOf(expenseAmount)+" \u20B4");
            }
        });
    }
    private void getWeekBalanceAmount() throws ParseException {
        Calendar calendar;
        calendar=Calendar.getInstance();
        DateFormat df = new SimpleDateFormat("dd/MM/yyyy",
Locale.getDefault());
        String startDate = "", endDate = "";
        calendar.set(Calendar.DAY_OF_WEEK, Calendar.MONDAY);
        startDate = df.format(calendar.getTime());
        Date sDate=df.parse(startDate);
        final long sdate=sDate.getTime();
        calendar.add(Calendar.DATE, 6);
        endDate = df.format(calendar.getTime());
        Date eDate=df.parse(endDate);
        final long edate=eDate.getTime();
        String dateString = startDate + " - " + endDate;
        dateTv.setText(dateString);
        AppExecutors.getInstance().diskIO().execute(new
Runnable() {
            @Override
            public void run() {
                int income =
mAppDb.transactionDao().getAmountbyCustomDates(Constants.incomeC
ategory,sdate,edate);
                incomeAmount = income;
                int expense =
mAppDb.transactionDao().getAmountbyCustomDates(Constants.expense
Category,sdate,edate);
                expenseAmount = expense;
                int balance = income - expense;
                balanceAmount = balance;
            }
        });
        AppExecutors.getInstance().mainThread().execute(new
```

```
Runnable() {
            @SuppressLint("SetTextI18n")
            @Override
            public void run() {

balanceTv.setText(String.valueOf(balanceAmount)+" \u20B4");
                incomeTv.setText(String.valueOf(incomeAmount)+"
\u20B4");

expenseTv.setText(String.valueOf(expenseAmount)+" \u20B4");
            }
        });
    }
    private void getMonthBalanceAmount() throws ParseException {
        Calendar calendar;
        calendar=Calendar.getInstance();
        DateFormat df = new SimpleDateFormat("dd/MM/yyyy",
Locale.getDefault());
        String startDate = "", endDate = "";
        calendar.set(Calendar.DAY_OF_MONTH,1);
        startDate = df.format(calendar.getTime());
        Date sDate=df.parse(startDate);
        final long sdate=sDate.getTime();
        calendar.set(Calendar.DAY_OF_MONTH,
calendar.getActualMaximum(Calendar.DAY_OF_MONTH));
        endDate = df.format(calendar.getTime());
        Date eDate=df.parse(endDate);
        final long edate=eDate.getTime();
        String dateString = startDate + " - " + endDate;
        dateTv.setText(dateString);
        AppExecutors.getInstance().diskIO().execute(new
Runnable() {
            @Override
            public void run() {
                int income =
mAppDb.transactionDao().getAmountbyCustomDates(Constants.incomeC
ategory,sdate,edate);
                incomeAmount = income;
                int expense =
mAppDb.transactionDao().getAmountbyCustomDates(Constants.expense
Category,sdate,edate);
                expenseAmount = expense;
                int balance = income - expense;
                balanceAmount = balance;
            }
        });
        AppExecutors.getInstance().mainThread().execute(new
Runnable() {
            @SuppressLint("SetTextI18n")
            @Override
            public void run() {

balanceTv.setText(String.valueOf(balanceAmount)+" \u20B4");
```

```
                incomeTv.setText(String.valueOf(incomeAmount)+"
\u20B4");

expenseTv.setText(String.valueOf(expenseAmount)+" \u20B4");
            }
        });
    }
}
```

CustomButtomSheetDiaologFragment.java

```
public class CustomBottomSheetDialogFragment extends
BottomSheetDialogFragment {
    Button addIncomeButton;
    Button addExpenseButton;
    @Override
    public View onCreateView(final LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
        View v =
inflater.inflate(R.layout.bottom_sheet_fragment, container,
false);

addIncomeButton=v.findViewById(R.id.bottom_sheet_income_btn);

addExpenseButton=v.findViewById(R.id.bottom_sheet_expense_btn);
        addIncomeButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                dismiss();
                Intent intent =new Intent(getActivity(),
AddExpenseActivity.class);
                intent.putExtra("from",
Constants.addIncomeString);
                startActivity(intent);
            }
        });
        addExpenseButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                dismiss();
                Intent intent =new Intent(getActivity(),
AddExpenseActivity.class);
                intent.putExtra("from",
Constants.addExpenseString);
                startActivity(intent);
            }
        });
        return v;
    }
}
```

AppExecutors.java

```
public class AppExecutors {
    private static final Object LOCK = new Object();
    private static AppExecutors sInstance;
    private final Executor diskIO;
    private final Executor mainThread;
    private final Executor networkIO;
    private AppExecutors(Executor diskIO, Executor networkIO,
Executor mainThread) {
        this.diskIO = diskIO;
        this.networkIO = networkIO;
        this.mainThread = mainThread;
    }
    public static AppExecutors getInstance() {
        if (sInstance == null) {
            synchronized (LOCK) {
                sInstance = new
AppExecutors(Executors.newSingleThreadExecutor(),
                        Executors.newFixedThreadPool(3),
                        new MainThreadExecutor());
            }
        }
        return sInstance;
    }
    public Executor diskIO() {
        return diskIO;
    }
    public Executor mainThread() {
        return mainThread;
    }
    public Executor networkIO() {
        return networkIO;
    }
    private static class MainThreadExecutor implements Executor
{
        private Handler mainThreadHandler = new
Handler(Looper.getMainLooper());
        @Override
        public void execute(@NonNull Runnable command) {
            mainThreadHandler.post(command);
        }
    }
}
```

DateConverter.java

```
public class DateConverter {
    @TypeConverter
    public static Date toDate(Long timestamp) {
        return timestamp == null ? null : new Date(timestamp);
    }
    @TypeConverter
    public static Long toTimestamp(Date date) {
        return date == null ? null : date.getTime();
```

```
    }
}
```

TransactionViewModel.java

```
public class TransactionViewModel extends AndroidViewModel {
    public final LiveData<List<TransactionEntry>> expenseList;
    private AppDatabase appDatabase;
    public TransactionViewModel(@NonNull Application
application) {
        super(application);
        appDatabase =
AppDatabase.getInstance(this.getApplication());
        expenseList =
appDatabase.transactionDao().loadAllTransactions();
    }
    public LiveData<List<TransactionEntry>> getExpenseList() {
        return expenseList;
    }
    public void updateTransaction(TransactionEntry
transactionEntry){
        new
updateTransactionDetails(appDatabase).execute(transactionEntry);
    }
   private static class updateTransactionDetails extends
AsyncTask<TransactionEntry,Void,Void>{
        private AppDatabase mdb;
        public updateTransactionDetails(AppDatabase
appDatabase){
            mdb = appDatabase;
        }
       @Override
       protected Void doInBackground(TransactionEntry...
transactionEntries) {

mdb.transactionDao().updateExpenseDetails(transactionEntries[0])
;
            return null;
        }
   }
```

Constants.java

```
public class Constants {

    public static final String addIncomeString="addIncome";
    public static final String addExpenseString="addExpense";
    public static final String editIncomeString="editIncome";
    public static final String editExpenseString="editExpense";
    public static final String incomeCategory="Income";
    public static final String expenseCategory="Expense";
}
```

ExpenseList.java

```
public class ExpenseList {
    String category;
```

```
    int amount;
    public ExpenseList(String category, int amount) {
        this.category = category;
        this.amount = amount;
    }
    public String getCategory() {
        return category;
    }
    public void setCategory(String category) {
        this.category = category;
    }
    public int getAmount() {
        return amount;
    }
    public void setAmount(int amount) {
        this.amount = amount;
    }
}
```

activity_add_expense.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    tools:context=".activities.AddExpenseActivity">
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <android.support.design.widget.TextInputLayout
        android:id="@+id/amountTextInputLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="35dp"
        android:layout_marginStart="25dp">
        <android.support.design.widget.TextInputEditText
            android:id="@+id/amountTextInputEditText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:hint="@string/amount"
            android:inputType="number"
            android:ems="14"/>
    </android.support.design.widget.TextInputLayout>
    <LinearLayout
        android:layout_marginTop="10dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
            android:id="@+id/dateLinerLayout"
            android:orientation="vertical">
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="@string/date"
                    android:layout_marginTop="15dp"
                    android:layout_marginStart="25dp"
                    android:textSize="18sp"
                    android:padding="4dp"/>
                <TextView
                    android:id="@+id/dateTextView"
                    android:text="11/09/2017"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="8dp"
                    android:layout_marginStart="25dp"
                    android:layout_marginEnd="15dp"
                    android:layout_marginBottom="10dp"
                    android:textSize="20sp"
                    android:textColor="#000000"
                    android:padding="4dp"/>
        </LinearLayout>
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/categoryLinerLayout"
            android:orientation="vertical">
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/category"
                android:layout_marginTop="15dp"
                android:layout_marginStart="35dp"
                android:layout_marginEnd="15dp"
                android:textSize="18sp"
                android:padding="4dp"/>
            <Spinner
                android:id="@+id/categorySpinner"
                android:layout_width="150dp"
                android:layout_height="wrap_content"
                android:layout_marginTop="3dp"
                android:layout_marginStart="23dp"
                android:layout_marginBottom="10dp"
                android:layout_marginEnd="30dp"/>
        </LinearLayout>
    </LinearLayout>
    <android.support.design.widget.TextInputLayout
        android:id="@+id/descriptionTextInputLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="15dp"
        android:layout_marginStart="25dp">
```

```
        <android.support.design.widget.TextInputEditText
            android:id="@+id/descriptionTextInputEditText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:hint="@string/description"
            android:inputType="textPersonName|textCapSentences"
            android:ems="14"/>
    </android.support.design.widget.TextInputLayout>
</LinearLayout>
</ScrollView>
```

bottom_sheet_fragment.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_behavior="@string/bottom_sheet_behavior"
    app:behavior_hideable="true">
    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <TableRow
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:weightSum="2">
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_weight="1"
                android:layout_height="wrap_content"
                android:orientation="horizontal">
                <Button
                    android:layout_width="0dp"
                    android:layout_height="90dp"
                    android:layout_weight="1"
                    android:background="#aeea00"
                    android:drawablePadding="6dp"
                    android:gravity="left|center"
                    android:height="60dp"
                    android:padding="6dp"
                    android:text="@string/add_income"
                    android:textSize="20sp"
                    android:textAllCaps="false"
                    android:id="@+id/bottom_sheet_income_btn"
                    android:textAlignment="center"

android:foreground="?android:attr/selectableItemBackground" />
            </LinearLayout>
            <View android:layout_width="2dp"
                android:layout_height="match_parent"
                android:background="@color/colorAccent"/>
            <LinearLayout
```

```
                android:layout_width="match_parent"
                android:layout_weight="1"
                android:layout_height="wrap_content"
                android:orientation="horizontal">
                <Button
                    android:layout_width="0dp"
                    android:layout_height="90dp"
                    android:layout_weight="1"
                    android:drawablePadding="6dp"
                    android:gravity="left|center"
                    android:height="60dp"
                    android:padding="6dp"
                    android:textSize="20sp"
                    android:text="@string/add_expense"
                    android:textAllCaps="false"
                    android:background="#ff5722"
                    android:id="@+id/bottom_sheet_expense_btn"
                    android:textAlignment="center"

android:foreground="?android:attr/selectableItemBackground" />
            </LinearLayout>
        </TableRow
    </TableLayout>
</LinearLayout>
```

fragment_balance.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".fragments.BalanceFragment">
    <FrameLayout
        android:id="@+id/frameLayout"
        android:layout_width="match_parent"
        android:layout_height="290dp"
        android:layout_margin="4dp"
        android:layout_marginBottom="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/amountForIncomeTextVie
w"
        app:layout_constraintVertical_bias="0.0">
        <com.github.mikephil.charting.charts.PieChart
            android:id="@+id/balancePieChart"
            android:layout_width="match_parent"
            android:layout_height="260dp"
```

```
            android:layout_marginTop="10dp"
            android:visibility="gone">
        </com.github.mikephil.charting.charts.PieChart>
    </FrameLayout>
    <Spinner
        android:id="@+id/spinner"
        android:layout_width="368dp"
        android:layout_height="40dp"
        android:layout_marginBottom="186dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="3dp"
        app:layout_constraintBottom_toTopOf="@+id/frameLayout"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />
    <TextView
        android:id="@+id/dateTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="4dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="4dp"
        android:padding="5dp"
        android:text="01/10/2018 - 30/1/2018"
        android:textAlignment="center"
        android:textSize="16sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.081" />
    <TextView
        android:id="@+id/totalAmountTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="133dp"
        android:layout_marginStart="133dp"
        android:padding="4dp"
        android:text="400"
        android:textAlignment="center"
        android:textColor="@android:color/black"
        android:textSize="26sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.508"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/dateTextView"
/>
    <TextView
```

```
            android:id="@+id/incomeTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginEnd="200dp"
            android:layout_marginStart="125dp"
            android:layout_marginTop="2dp"
            android:text="@string/income"
            android:textSize="18sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/totalAmountTextView"
/>
    <TextView
            android:id="@+id/expenseTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginEnd="8dp"
            android:layout_marginStart="8dp"
            android:layout_marginTop="2dp"
            android:text="@string/expense"
            android:textAlignment="center"
            android:textSize="18sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.664"
            app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/totalAmountTextView"
/>
    <TextView
            android:id="@+id/amountForIncomeTextView"
            android:layout_width="61dp"
            android:layout_height="wrap_content"
            android:layout_marginEnd="201dp"
            android:layout_marginStart="125dp"
            android:layout_marginTop="3dp"
            android:text="600"
            android:textAlignment="center"
            android:textColor="#aeea00"
            android:textSize="18sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/incomeTextView" />
    <TextView
            android:id="@+id/amountForExpenseTextView"
            android:layout_width="70dp"
            android:layout_height="wrap_content"
            android:layout_marginEnd="104dp"
            android:layout_marginTop="3dp"
            android:text="200"
            android:textAlignment="center"
            android:textColor="#ff5722"
```

```
        android:textSize="18sp"
        app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintTop_toBottomOf="@+id/expenseTextView" />
</android.support.constraint.ConstraintLayout>
```

fragment_expense.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bg"
    tools:context=".fragments.ExpenseFragment">
    <android.support.v7.widget.RecyclerView
        android:id="@+id/transactionRecyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginBottom="20dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.025" />
</android.support.constraint.ConstraintLayout>
```

list_item.xml

```
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <android.support.v7.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="2dp"
        android:layout_marginEnd="2dp"
        android:layout_marginStart="2dp"
        android:layout_marginTop="2dp"
        app:cardCornerRadius="6dp"
        android:elevation="7dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="1.0">
```

```
<android.support.constraint.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="2dp">
    <TextView
        android:id="@+id/amountTextViewrv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="6dp"
        android:layout_marginEnd="4dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:padding="2dp"
        android:text="-300"
        android:textSize="28sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.956"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.173" />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0">
        <TextView
            android:id="@+id/categoryTextViewrv"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="2dp"
            android:layout_marginEnd="8dp"
            android:layout_marginStart="10dp"
            android:layout_marginTop="2dp"
            android:padding="2dp"
            android:text="Food"
            android:textColor="#000000"
            android:textSize="20sp" />
        <TextView
            android:id="@+id/dateTextViewrv"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginEnd="8dp"
            android:layout_marginStart="11dp"
            android:padding="2dp"
            android:text="14/09/2018"
            android:textSize="12sp"
             />
    </LinearLayout>
    <TextView
```

```
                android:id="@+id/descriptionTextViewrv"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="3dp"
                android:layout_marginEnd="8dp"
                android:layout_marginStart="11dp"
                android:layout_marginTop="60dp"
                android:fontFamily="sans-serif-condensed-light"
                android:padding="4dp"
                android:text="Dominos"
                android:textSize="14sp"
                app:layout_constraintBottom_toBottomOf="parent"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintHorizontal_bias="0.0"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent"
                app:layout_constraintVertical_bias="0.0" />
        </android.support.constraint.ConstraintLayout>
    </android.support.v7.widget.CardView>
</android.support.constraint.ConstraintLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.justmik.financemanager">
    <application
        android:allowBackup="true"
        android:label="@string/app_name"
        android:icon="@mipmap/ic_launcher"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity

android:name="com.justmik.financemanager.activities.MainActivity
"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar">
        </activity>
        <activity
android:name="com.justmik.financemanager.activities.AddExpenseAc
tivity" />
        <activity
android:name="com.justmik.financemanager.activities.SplashScreen
"
            android:theme="@style/SplashTheme">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

```
        </activity>
    </application>
</manifest>
```