Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Сауляка Яна Максимовича*
(ПІБ)

академічної групи *122-18-2*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка мобільного додатку гаманця для фінансового обліку*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф.Якунін А.О.*			
розділів:				
спеціальний	*проф.Якунін А.О*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент	*проф.Гусєв О.Ю.*			
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

І.М. Удовик
(підпис) **(прізвище, ініціали)**

« » **2022 року**

ЗАВДАННЯ
на кваліфікаційну роботу
бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18-2* ***Сауляка Яна Максимовича***
(група) **(прізвище та ініціали)**
тема кваліфікаційної роботи *Розробка мобільного додатку гаманця Для фінансового обліку*

затверджена наказом ректора НТУ «ДП»від 18.05.2022 № 268-с

Розділ	Зміст виконання	Термін виконання
Спеціальний	*На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.* *Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми*	*13.05.2022 р.*
Економічний	*Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки*	*27.05.2022 р.*

Завдання видав _____ *проф. Якунін А.О.*
(підпис) **(посада, прізвище, ініціали)**

Завдання прийняв до виконання _____ *Сауляк Я.М.*
(підпис) **(прізвище, ініціали)**

Дата видачі завдання: ***14.01.2022 р.***

Термін подання кваліфікаційної роботи до ЕК: ***13.06.2022 р.***

РЕФЕРАТ

Пояснювальна записка: 85 с., 28 рис., 1 табл., 3 дод., 22 джерела.

Об'єкт розробки: інформаційна система для операційної системи Android.

Мета кваліфікаційної роботи: створення мобільного додатку для обліку фінансів для ОС Android.

У вступі уточнюється постановка завдання, конкретизується мета кваліфікаційної роботи актуальність та галузь її застосування.

У першому розділі проведено аналіз предметної галузі, визначено призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі наведено: призначення додатку, опис застосованих математичних методів, опис використаних технологій, опис використаного програмного забезпечення, вибір мов програмування, опис структури програми та алгоритмів її функціонування та детальний опис роботи розробленого програмного продукту.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні застосунка, що надасть можливість користувачу вести точний облік фінансових ресурсів малого бізнесу або власних коштів.

Актуальність інформаційної системи визначається зростанням темпу відкриття малого бізнесу громадянами України.

Список ключових слів: СМАРТФОН, ФІНАНСИ, UI, ІНТЕРФЕЙС КОРИСТУВАЧА, МАЛИЙ БІЗНЕС, UX, ДОСВІД КОРИСТУВАЧА, ДОДАТОК, ГАМАНЕЦЬ, ANDROID, IOS, КЛІЄНТ.

ABSTRACT

Explanatory note: 85 p., 28 figures, 1 table, 3 additings, 22 references.

The object of development: an information system for the Android operating system.

The goal of the qualification work: creation of a mobile add-on for financial accounting for the Android operating system.

The introduction clarifies the statement of the task, specifies the meta-qualification work relevance and scope of its use.

In the first section, an analysis of the subject area was carried out, the purpose of the development was determined, the problem statement was developed, the requirements to the software implementation, technologies and software tools were set.

In the second section is given: the recognition of the addendum, a description of the used mathematical methods, a description of the used technologies, a description of the used software, the choice of programming languages, a description of the program structure and algorithms of its functioning and a detailed description of the work of the developed software product.

In the economic section, the labor intensity of the developed information system is determined, and the cost of the work on the creation of the program is estimated and the time for its creation is calculated.

Practical importance lies in the creation of the application, which will enable the user to conduct an accurate accounting of financial resources of small businesses or own funds.

The relevance of the information system is determined by the growing rate of small business opening by Ukrainian citizens.

Keywords: SMARTPHONE, FINANCE, UI, CORRESTRUCTOR INTERFACE, SMALL BUSINESS, UX, CORRESTRUCTOR DOSAGE, DATA, GAMANETS, ANDROID, IOS, CLINT.

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

МП – мова програмування;

ІС – інформаційна система;

ОС – операційна система;

СР – середовище розробки

ПК – персональний комп’ютер;

UI – user interface;

UX – user experience.

ЗМІСТ

ВСТУП

На поточний час через зниження жорсткості вимог уряду України по відношенню до малого бізнесу, ми можемо передбачити прискорення росту кількості фізичних осіб підприємців, які будуть потребувати зручну середу для контролю за фінансовою складовою їх бізнесу.

2022 рік - вік технологій в якому більшість програмного забезпечення розробляється саме для смартфонів. Переважна більшість останніх флагманів у світі смартфонів може повністю замінити комп’ютер під час виконання певних завдань. Велика кількість різноманітного програмного забезпечення допомогае підприємцям вести свій бізнес без застосовування додаткових людей.

Зараз у світі майже не залишилось людей які не використовували доступні функції мобільного пристрою. Причина через яку смартфони заполонили кишені людей у всьому світу лежить на поверхні, це максимальний функціонал, простота у використанні та доступність яку можна носити із собою будь-де та будь-коли. Без вагань можна зауважити що смартфони навіть у багатьох професіях вже повністю замінили настільний комп'ютер. Мобільні додатки розробляються під різні операційні системи смартфонів (найбільш найпоширеніші: iOS, Android) використовуючи різноманітні середовиа розробки.

У зв’язку з вище перерахованими фактами із сучасних реалій ми можемо зробити висновок що до невід’ємної частини образу сучасної людини у вигляді смартфону. Саме тому я вважаю, що додаток який дозволить контролювати грошову складову ведення бізнесу без застосовування додаткового обладнання та без найму бухгалтера це є дуже зручним рішенням.

Відповідно до проведеного аналізу в роботі поставлені такі основні функціональні задачі та вимоги до розроблюваного застосунку:

- наявність зручного інтуїтивно зрозумілого інтерфейсу;
- можливість проводити фінансові розрахунки самостійно;

– надання користувачеві інформації що до прибутку/затрат за певний період;

– розділення фіансових операцій на різноманітні категорії;

– зручне використання для користувачів всіх вікових категорій;

– створення аккаунту у додатку;

– розділення аккаунту на декілька облікових записів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Загальні відомості з предметної галузі

Люди зіштовхуються із проблемою так званого марнотратства, відсутнього планування затрат розрахованого виходячи з наявної кількості грошей на певний проміжок часу ще з тих пір, як у малоазійській країні Лідя під час правління царя Ардіса (685 роки до н.е.) почали виготовляти перші монети зі сплаву срібла та золота який на сьогодняшній день називається електрумом. Оскільки у сучасному світі без грошей вижити буде достатньо важко, потрібно підходити до питання їх використання с розумом, щоб у якийсь момент не виявити, що гроші закінчилися, їсти щось треба, а заробітна платня тільки через два тижні, адже гроші люблять розрахунок.

Згідно статистики ми можемо спостерігати наступну картину на фронті малого бізнесу: у світі більше половини малого бізнесу закривається не проживши навіть п1ять років, а до десяти років не доживає навіть третина від усього колись заснованого малого бізнесу. Якщо розібратися у причинах закриття бізнесів можна відстежити той факт, що більшість із них потерпає не від низького попиту, а саме через неправильне планування бюджету, адже наймати бухгалтера це дорого, а рахувати цифри на бумазі самому це важко. Саме на це й буде спрямований додаток, щоб допомогти не тільки звичайним людям спланувати витрати, а й допомогти бізнесменам залишатися на плаву.

Згідно статті 55 господарського кодексу України, під малими підприємствами розуміються:

фізичні особи, зареєстровані в установленому законом порядку як фізичні особи — підприємці, у яких середня кількість працівників за звітний період (календарний рік) не перевищує 50 осіб та річний дохід від будь-якої діяльності не перевищує суму, еквівалентну 10 мільйонам євро, визначену за середньорічним курсом Національного банку України та юридичні особи —

суб'єкти господарювання будь-якої організаційно-правової форми та форми власності, у яких середня кількість працівників за звітний період (календарний рік) не перевищує 50 осіб та річний дохід від будь-якої діяльності не перевищує суму, еквівалентну 10 мільйонам євро, визначену за середньорічним курсом Національного банку України

Функції малого бізнесу:

Формування конкурентного середовища.

Надання гнучкості ринковій економіці.

Сприяння швидкому розвитку НТП.

Поглинання надлишкової робочої сили.

Пом'якшення соціальної напруги.

Формування середнього класу.

Сприяє демонополізації.

Виходячи з вище вказаної інформації та сучасних тенденцій на розвиток малого бізнесу можна зрозуміти потребу у фінансовому урегулюванні молодих підприемств.

1.2. Призначення розробки та галузь застосування

Мобільний додаток «Cash keeper» для автоматизації процесу підрахунків та урегулювання фінансової складової малого бізнесу за звичайних людей. Мобільний додаток забезпечує зберігання та обробку інформації про фінансову активність особи.

Застосунок орієнтований на кінцевого користувача, котрий можливо не володіє високою кваліфікацією в галузі економіки та обчислень. Тому

мобільний додаток повинен мати багатий функціонал, просту частину UX та зручну частину UI.

1.3 Підстави до розробки

Підставою до розробки додатку для фінансового урегулювання є наказ ректора Національного технічного університету «Дніпровська політехніка» №268-с від 18.05.2022 на тему «Розробка мобільного додатку гаманця для фінансового обліку витрат».

1.4. Постановка завдання

Метою даної кваліфікаційної роботи є розробка мобільного додатку для смартфонів, що доводить користувачам пристроїв на базі операційної системи Android, автоматизувати процес фінансового обліку. Додаток забезпечує зберігання та обробку інформації про фінансову активність особи або малого підприемства.

1.5. Вимоги до програми або програмного виробу

Відповідно до проведеного аналізу в роботі поставлені такі основні функціональні задачі та вимоги до розроблюваного застосунку:

- розробка дизайн-системи для додатку;
- наявність зручного інтуїтивно зрозумілого адаптивного інтерфейсу;
- створення функціонуючого прототипу додатка для смартфонів.

1.5.1. Вимоги до функціональних характеристик

Додаток повинен надавати такий функціонал, як:

1. Розподілення фінансової активності на категорії.

2. Планування прибутково видаткових операцій.

3. Фіксація поточного балансу на різних засобах збереження коштів.

4. Розподілення спільного поточного балансу.

5. Фіксація прибутково видаткових операцій.

6. Можливість переглядати історію зафіксованих операцій.

7. Створення потрібних категорій витрат в потрібній кількості.

8. Створення категорій фіксації прибутку на різних засобах збереження коштів.

9. Створення обікових саписів.

10. Розділення облікового запису на декілька профілей користувача.

1.5.2. Вимоги до інформаційної безпеки

При розробці додатку «Cash keeper» достатньо встановлених в системі вимог до інформаційної безпеки. Виключень з боку антивірусних програм захисту не потребує.

Прототип може бути використаний лише користувачам, котрі мають файли потрібні для запуску додатку у СР.

1.5.3. Вимоги до складу та параметрів технічних засобів

Прототип, котрий був розроблений в програмі для створення мобільних додатків – Android Studio є вимогливим до оперативної пам’яті та потужностей процесору, так як працюватиме додаток у СР за допомогою емулятора мобільного пристрою, що додатково навантажує комп’ютерну систему

Вимоги до технічних засобів:

Персональний комп’ютер:

1. Оперативна пам’ять – мінімум 4 гігабайти.
2. Внутрішня пам’ять відеокарти – 2 гігабайти.

3. Операційна система:Windows 8 або більш новіші версії с 64-бітним середовищем.

Смартфон:

1. Оперативна пам’ять – мінімум 4 гігабайти.
2. Операційна система: Android 8 (Android Oreo).

1.5.4. Вимоги до інформаційної та програмної сумісності

Для коректного користування прототипом на ПК повинно бути встановлено програмне забезпечення для розробки додатків під ОС Android – Android Studio. В разі відсутності потрібного програмного забезпечення доступу до програми у користувача не буде, так як цей проект не будо опублікований на офіціальних ресурсах розповсюдження мобільних додатків Play Market.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Функціональне призначення додатку полягає в наданні користувачеві можливості вести облік фінансів власних, або контролювати фінансову складову малого бізнесу.

2.2. Опис застосованих математичних методів

Розробка цієї інформаційної системи не вимагае встановлення математичної моделі та застосування математичних методів.

2.3.Опис використаних технологій та мов програмування

Для реалізації данного проекту було обране середовище проектування та розробки додатків під ОС Android – Android studio і мову програмування Java.

2.3.1 Обґрунтування функцій програмного продукту

2.3.1.1 Формування варіантів виконання задачі

Головна задача №0 – розробка програмного продукту за певний проміжок часу та витримка перних вимог до кінцевого продукту у вигляді функціонуючої інформаційної системи для пірдахунку фінансів. Виходячи з основної мети ми можемо розділити її на декілька підзадач:

№1 – Вибір мови програмування.

№2 – Функціонал.

№3 – Середовище розробки.

№4 – Отримання користувачем вихідної інформації.

№5 – Безпечне зберігання інформації користувача.

Кожна з цих підзадач має декілька варіантів подальшого розвитку та використання

№1 – Вибір мови програмування

Варіант 1 – Java.

Варіант 2 – Kotlin.

Варіант 3 – C++.

№2 – Функціонал

Варіант 1 – Максимальний функціонал .

Варіант 2 – Обмежений функціонал.

№3 – Середовище розробки

Варіант 1 – Android Studio.

Варіант 2 – Microsoft Visual Studio.

№4 – Отримання користувачем вихідної інформації

Варіант 1 – На екран мобільного пристрою.

Варіант 2 – Вивід за рахунок внутрішніх файлів пристрою.

№5 – Безпечне зберігання інформації користувача

Варіант 1 – Встановлення додаткової системи захисту інформації.

Варіант 2 – Відмова від використання додаткових захисних систем.

2.3.1.2 Розбір варіантів реалізації кожної задачі

Задача № 1 – Вибір мови програмування

Варіант 1 – МП Java

Переваги: Швидкість написання коду та повне розкриття функціоналу CP Android studio.

Недоліки: Не краща віртуальна машина для тестування додатку.

Варіант №2 – МП Kotlin

Переваги: Швидкість написання Коду.

Недоліки: Нова, не кросплатформенна мова.

Варіант №3 – МП C++

Переваги: Швидкість виконування коду.

Недоліки: Швидкість написання коду.

Задача №2 – Функціонал

Варіант №1 – Максимальній функціонал

Переваги: Більше можливостей для використання додатку.

Недоліки: Важкість у використанні рядовим користувачем та збільшені вимоги до технічних засобів.

Варіант №2 – Обмежений функціонал

Переваги: Легкість у використанні людиною без спеціальних навичок.

Недоліки: Для певних задач не вистачить функціоналу.

Задача №3 – Середовище розробки

Варіант №1 – Android Studio

Переваги: Інтегрований графічний редактор та конструктор додатків.

Недоліки: розробка тильки під ОС Android.

Варіант №2 – Microsoft Visual Studio

Переваги: Швидке виконання коду та зручний відладчик коду.

Недоліки: побудова графічного інтерфейсу виконуется тільки за рахунок коду.

Задача №4 – Отримання користувачем вихідної інформації

Варіант №1 – На екран мобільного пристрою

Переваги: найзручніший способ отримання інформації з телефону.

Недоліки: важка реалізація.

Варіант №2 – Вивід за рахунок внутрішніх файлів пристрою

Переваги: Легкість реалізації.

Недоліки: Дуже не зручно отримувати інформацію.

Задача №5 – Безпечне зберігання інформації користувача

Варіант №1 – встановлення додаткових систем захисту інформації

Переваги: Менше шанс на втрату інформації.

Недоліки: Збільшені вимоги до технічних засобів.

Варіант №2 – Відмова від використання додаткових захисних систем

Переваги: Зменшені вимоги до технічних засобів.

Недоліки: Більший шанс на втрату інформації.

На основі вище наведеного аналізу переваг та недоліків кожного із варіантів розвитку кожноі задачі, ми можемо зробити вискновок, що деякі варіанти розвитку деяких задач потрібно відкинути через те, що вони не відповідають встановленим потребам до кінцевого програмного продукту.

Задача №1 – Оскільки розрахунки у програмному застосунку проводяться з великою кількістю вхідних данних та функціонуючою базою данних, час

написання коду має велике значення, тому було обрано мову програмування Java, так як мови програмування C++ та Kotlin не підходять до потрібних кретерій.

Задача №2 – Оскільки програмний застосунок розрахований більше на використання фізичними особами у власних цілях, було прийняте рішення знизити функціонал застосунку у користь простоти інтерфейсу для більш комфортного використання програми.

Задача №3 – Оскільки швидкість написання коду має велике значення, а другий варіант не надає можливості створювати графічні інтерфейси без втручання у код програми, було обрано середовище програмування під ОС Android – Android studio.

Задача №4 – Оскільки в приорітеті є комфорт користувача при використання додатку, було обрано відображення вихідної інформацфї на екрані мобільного пристою.

Задача №5 – Оскільки використання застосунку не передбачує використання у ньому конфіденційної інформації, був обран варіант без інтеграції у програму додаткових систем захисту інформації у користь зниження навантаження технічного засобу.

Виходячи з вище перерахованих фактів ми маємо зробити программу наступним чином:

Використовуемо мову програмування Java у середовищі програмування Android studio з обмеженим функціоналом, з використанням базових методів захисту інформації (створення користувачем паролю до облікового запису та системи безпеки які інтегровані у ОС Android), та виводом вихідної інформації на екран мобільного пристроб користувача.

2.3.2 Платформа Android

Головна, на мою думку перевага операційної системи Android –

Її відкрисий висхідний код. Операційна система Android за рахунок відкритого висхідного коду, надає можливість розробникам мобільних програмних застосунків зрозуміти архітектуру операційної системи з якою вони працюють та використати всі її переваги на користь програмного застосунку, який вони розроблюють

Android – операційна система яка була розроблена бізнес ал'янсом із 84 компаній по розробці відкритих стандартів до мобільних пристроїв враховуючих Google, HTC, Intel, Motorolla, Asus, Qualcomm, Samsung, LG, T-mobile, Nvidia, Wind River Systems та інші компанії – Open Handset Alliance. Ця операційна система має власні версії для підтримки не тільки смартфонів, а й праншетів, електронних книг, розумних годинників, фітнес браслетів, ноутбуків, ігрових приставок, тощо.

За можливостями, які багаторівнева операційна система Android надае користувачеві та розробникам, вона не поступается навіть операційним системам які обслуговують пресональні компьютери, створена вона була на основі ядра Linux.

У погоні за максимальною єфективністю, найменшим споживанням оперативної пам'яті та заряду батареї, максимальною швидкістю та все це умістити в невеликий корпус, компанія Google вклала величезну кількість ресурсів. Для задоволення цих вимог була спеціально розроблена віртуальна машина Dalvik. Ця віртуальна машина не підтримує JIT компіляцію та була заснована на регістрі, на відміну від стандартного виконавчного середовища Java яке було засноване на стеку. Кожен окремо запущений додатон на мобільному пристрої виконуется у окремо створеній віртуальній машині Dalvik що дозволяе розподіляти пам'ять між ними для зниження витрат. Також віртуальна машина Dalvik має спеціальний компонент що відповідае за скорочення часу запуску віртуальних машин для додатків, що мае назву Zygote, Який заздалегіть створює окремі екземпляри віртуальних машин і якщо на те є потреба, розгалуджує їх та розподіляе пам'ять.

Як і більша частина інших мобільних операційних систем, Android має вбудовані у систему базові додатки, такі як браузер, календар, контактна книга, додаток для користування електронною поштою, та деякі інші.

Різні версії операційної системи Android всеодно залишаються досить однорідними, що дозволяє переважній кількості мобільних додатків працювати на технічних засобак з різними версіями операційної системи, що в свою чергу ствоює сприятливі умови для розробників додатків.

Відкрита файлова система ОС Android надає унікальну можливість взаємодії різних додатків, а саме, будь який додаток, в якому реалізовані ті чи інші можливості, може надати доступ до необхідних файлів іншому додатку, щоб той також мог використовувати ці можливості. Через вище згадану особливість архітектури, ми можемо побачити принцип багаторазового використання методів, додатків та їх складових. Нижче, на рисунку 2.1 наведено схему архітектури операційної системи.

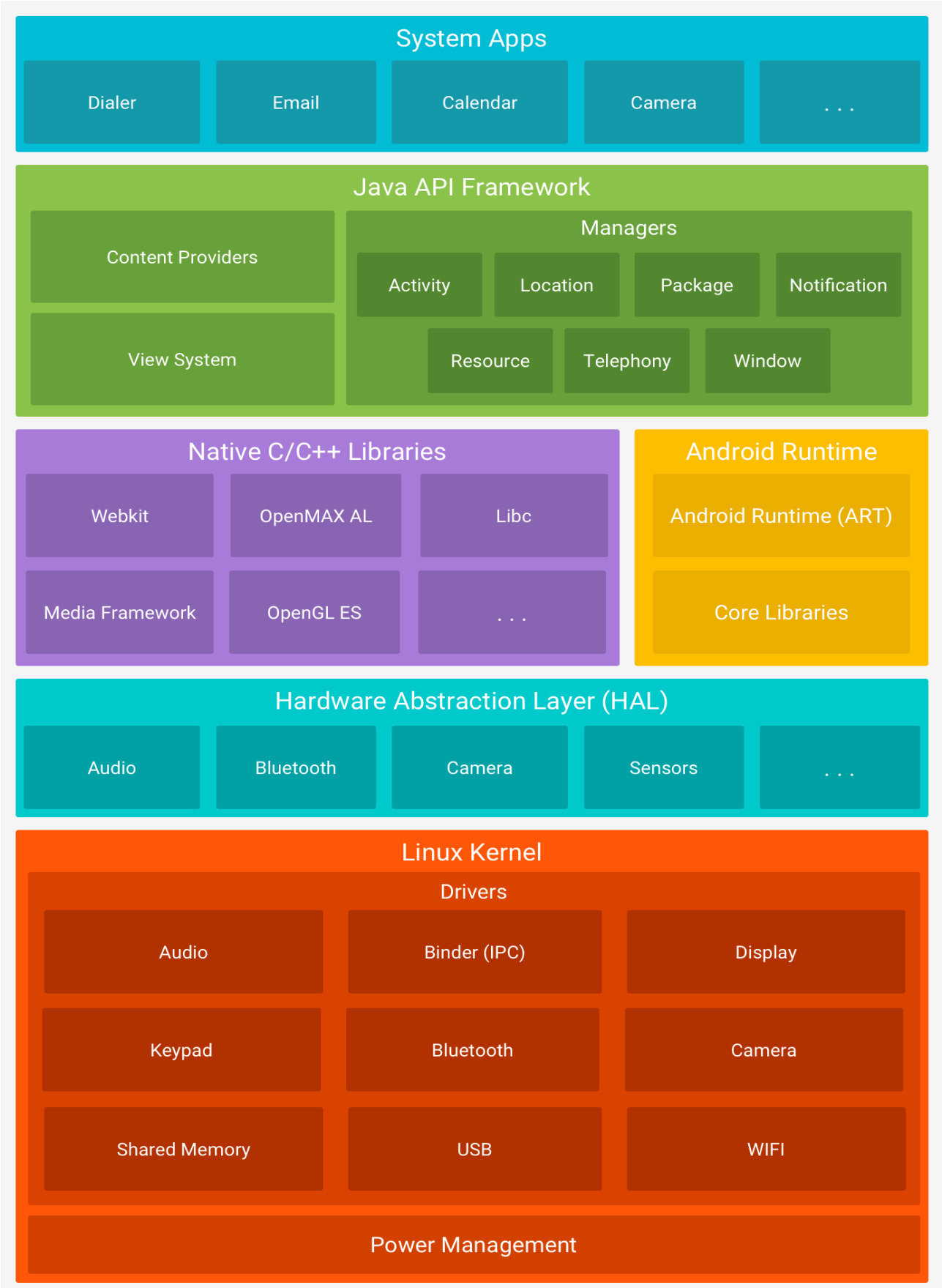


Рис 2.1. Схема архітектури Android

Операційна система Android славнозвісна своїм повним набором добре організованих API інтерфейсів, які потрібні для створення додатків та використання всього спектру функцій які нам надае платформа Android.
API інтерфейси данної операційної системи відносно інтуітивно зрозумілі та не важкі у використанні в процессі розробки додатку через дуже високий рівень абстрактності самих інтерфейсів.

Додатки сторонніх розробників можуть копіювати або взаємодіяти з основними компоненти платформи. Наприклад, Android надає методи Зі списку контактів користувача та розширення інформації контакта з новими полями даних. Також легко створити новий інтерфейс Викликати або реалізовувати користувацьку поведінку для системних подій, наприклад вхідне повідомлення. Зокрема, API дозволяє створювати програми, які Повністю інтегровані з іншою частиною платформи. На рисунку 2.2 наведено приклад застосування API інтерфейсу.

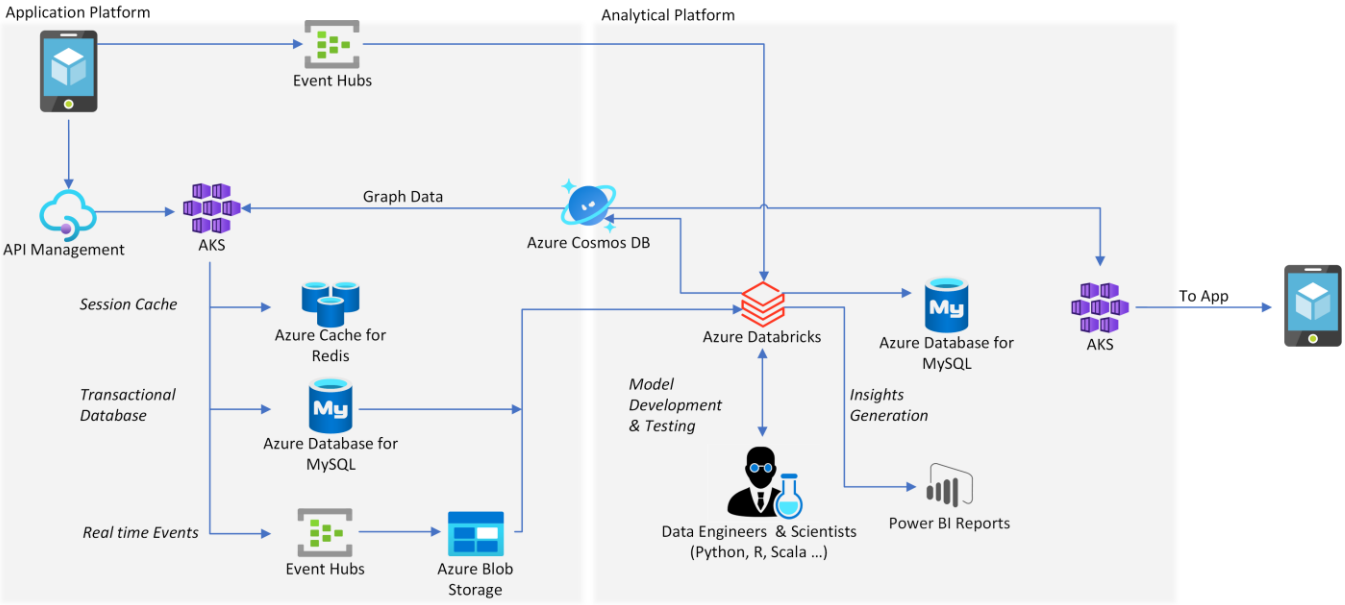


Рис.2.2. Приклад використання API

Набір інструментів для віджетів Android надає велику кількість дуже корисних компонентів. Деякі віджети призначені для зручної взаємодії з

пальцями, з вбудованими функціями, такими як Кінетичне пролистування. Розробники використовують спеціальну мову опису Інтерфейса на основі XML для визначення макета та властивостей віджетів. Отже, опис XML завантажується в програму через систему ресурсів Android.Посилання на окремі віджети,описані в макеті XML,можна отримати за адресою ідентифікатором у програмі.Віджети також можна створювати програмно використовуючи інтерфейс користувача, щоб маніпулювати ними під час виконання програми. Найкращий спосіб створити макет — написати опис XML вручну. Android SDK надає інтегрований інструмент візуального макета, але він підтримує не всі віджети і не завжди працює у координації.

Програми для Android складаються з постачальників, служб, приймачів і діяльності. Усі ці компоненти мають одне завдання в стеку програм Android. Метод взаємодії додатків одне з одним це саме те, що робить ОС Android модульною.

Доволі значна особливість, яка вирізняє ОС Android серед інших операційних систем це те, що ОС підтримує виконання фонових процессів додатків від сторонніх розробників на офіційному рівні. У реалізації цього був використан сервісний компонент Android. Процесси, які виконуются у фоновому режимі без первного обмеження у часі це і є сервіси Android.

Система обміну повідомленнями Android приблизно нагадує подібну систему інформування в Linux. Вбудовані приймачі, створені розробниками, можутьвизначити, коли відбуваються певні системні повідомлення або події, і Виконає якусь дію у відповідь. Багато основних системних подій можна відстежувати за допомогою системи мовлення, так вона використовується для відстеження таких речей, як зміна стану акумулятора, отримання вхідних SMS-повідомлень, натискання клавіш,зміна підключення, налаштування та зміна яркості екрану.

Механізм Android Intens є ключем до розуміння того, як працюють усі ці частини використані разом. Об’єкт Intent, що містить ідентифікатор та URI інформації, що використовується для виклику операцій, послуг і одержувачів,

ідентифікатори дій вказують на бажану поведінку, та необов'язковіURI надає місцезнаходження даних для роботи над ними. Рис. 2.3 Показано огляд схеми компонентів програми Android.

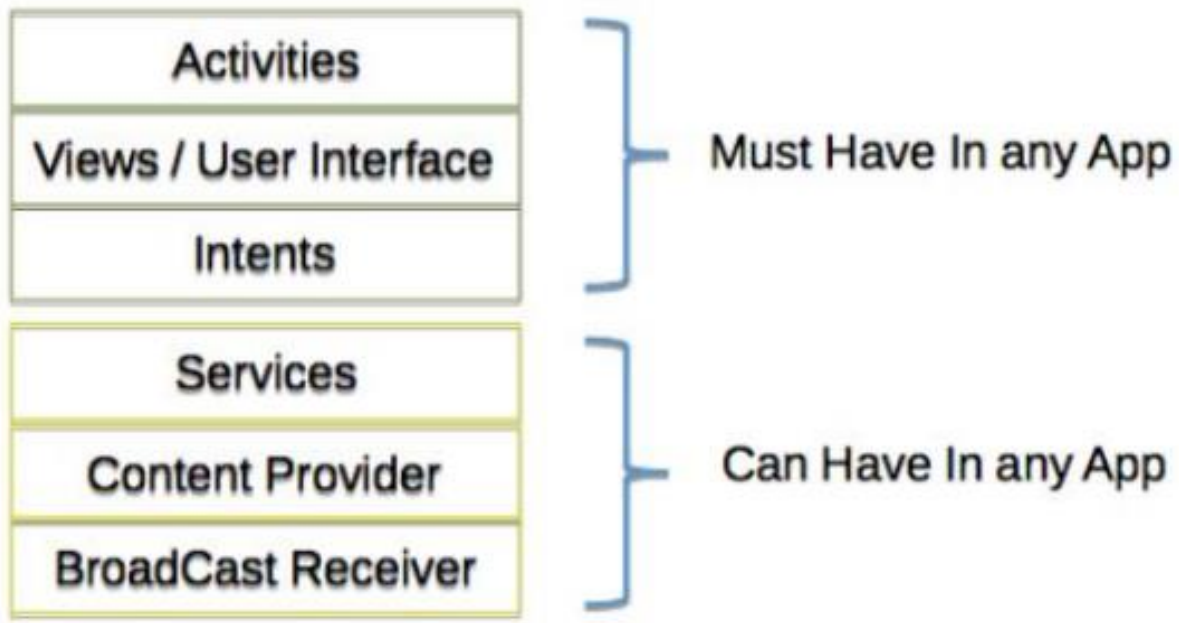


Рис.2.3. схема компонентів Android програм

Як раніше було вказано, операційна система Android має дуже вагомий плюс для розробників – відкритість. Операційна система була розроблена на основі відкритого системного коду, також вона вільно розповсюджується. Саме на ці факти звертає увагу більшість розробників, адже це дозволяє їм повністю розуміти логіку взаємодії розроблюваного додатку з пристроєм, сервісами та іншими додатками.

2.3.3 Середовище розробки Android Studio

Android Studio це офіційна інтегрована середа для розробки програмних додатків для операційної системи Android яка буда збудована на основі IntelliJ IDEA. Використовуючи цю середу розробки ви можете розраховувати на наступні можливості:

1) Багато шаблонних елементів коду, які допоможуть побудувати скелет додатку.
2) Підтримка Drag&Drop системи проектування додатку, що значно скоротшуе витрачений час на розробку додатку та підвищуе рівень комфорту при використанні цієї IDE.
3) Різноманітні методи та інструменти для фіксації сумісності додатку з різними версіями ОС, налагодження продуктивності додатку, тощо.
4) Доволі різноманітна Gradle-система збірки.
5) Використання інструменту командної строки – ProGuard.
6) API Android SDK – API бібліотеки операційної системи для розробки програмних додатків.
7) Проста можливість інтеграції повідомлень Google Cloud завдяки вбудованій підтримці цієї платвоорми.
8) Велика інформаційна документація Android SDK .
9) Вбудований у середу розробки AVD (Android Virtual Device) емулятор мобільного пристрою для наглядного тестування додатку не використовуючи фізичного технічного засобу.
10) Sample Code надае демонстративні базові програми, які показують можливості Операційної системи Android та правильні способи використання у вашому додатку API інтерфейсів.

Це корисно знати, перш ніж починати розробляти програми для Android. Загальний підхід платформи до управління змінами API. Також важливе розуміння рівнів API Android і їх функції. Переконайтеся, що ваша програма сумісна з пристроєм, на якому вона працюватимеЩоб встановити.

Рівень API – це ціле значення, яке однозначно ідентифікує версію API. Платформа надає структуру API, яку можуть використовувати програми для взаємодії з Android. Кожну наступну версію, платформа Android може містити оновлення API.

Оновлення структури API має на меті зберегти новий API

сумісним з попередніми версіями API. Таким чином, більшість змін в API об’єднується та вводить нові функції або виправляйте попередні.

Застарілі API не підтримуються до використання, оскільки деякі API постійно оновлюються, але їх не видалено з міркувань сумісності з існуючими додатками.Визначається рівень API, який використовується додатком Android цілим числом ідентифікатором, який вказується у файлі конфігурації кожного Android-додатку.

Нижче наведена таб. 2.1 сумісності рівнів API з різними версіями операційної системи Android.

Таблиця 2.1.

Сумісність версій Android з рівнями API

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.0 Ice Cream Sandwich	15	
4.1 Jelly Bean	16	99.8%
4.2 Jelly Bean	17	99.2%
4.3 Jelly Bean	18	98.4%
4.4 KitKat	19	98.1%
5.0 Lollipop	21	94.1%
5.1 Lollipop	22	92.3%
6.0 Marshmallow	23	84.9%
7.0 Nougat	24	73.7%
7.1 Nougat	25	66.2%
8.0 Oreo	26	60.8%
8.1 Oreo	27	53.5%
9.0 Pie	28	39.5%
10. Android 10	29	8.2%

Крім емулятора, SDK містить багато інших інструментів.

Інструменти для налагодження та встановлення програм якщо при розробці додатків для Android за допомогою Android Studio IDE багато Інструменти командного рядка, включені в пакет SDK, уже використовуються для складання та тестування проектів. Однак, крім їх SDK містить багато корисних інструментів для розробки та налагодження програм:

• Android є важливим інструментом розробки командного рядкаДозволяє створювати, видаляти та налаштовувати віртуальні рядкипристроїв, створення та оновлення проектів Android (при зовнішній роботісередовище Eclipse) та оновіть Android SDK за допомогою нових платформ,додатків та документацій.

• Служба моніторингу налагодження Dalvik (DDMS) – інтегрована з Dalvik Virtual Machine, стандартна віртуальна машина платформи Android Дозволяє керувати процесами на емуляторі, а також допомагає з налагодженням додатку. За допомогою цього сервісу також можна застосовувати для завершення певної активності та вибору певної Процедури налагодження, генерування слідів даних, перегляд інформації про потоки, отримати скріншоти емулятора тощо.

• Hierarchy Viewer - інструмент візуалізації, який дозволяє налаштувати оптимізований інтерфейс користувача. Він показує візуальне дерево рівнів думки, аналіз швидкості перемальовування екранну графіку і може виконувати багато інших функцій аналізу графічних інтерфейсів програм.

• Layoutopt – інструмент командного рядка, який допомагає оптимізувати схему розмітки та ієрархію тегів у створеній програмі. Потреби вирішення проблеми, створюючи складні графічні інтерфейси впливати на продуктивність програми.

• Draw 9-patch - простий у створенні графічний редактор для розробки графічного інтерфейсу програми, використовується графіка NinePatch.

• sqlite3 – засіб для доступу до даних SQLite, які створені і використовуються додатками Android.

• Traceview - цей засіб забезпечує графічний аналіз слідів журналу, можна створити з програми.

• mksdcard - інструмент для створення образів дисків. Використовується в емуляторі для імітації наявності зовнішнього накопичувача(SD-карта).

• Найважливішим із цих інструментів є мобільний емулятор, але пакет SDK містить також додаткові інструменти для налагодження та завантаження своєї програми на емуляторі.

2.3.4 Розробка архітектури проекту

Розробка мобільних додатків поділяється на два етапи: статичний макет екрана на XML та побудова логіки на Java. Розробка відбувається в популярній IDE від Google - Android Studio. Затверджені конструкції «розрізаються» на окремі креслення і згодом з них створюється xml-розмітка програми. результатом є код, який можна переглянути на мобільному пристої. А стандартні сторінки пізніше буде використано як шаблон. Потім файл XML динамічно заповнюється необхідною інформацією використовуючи Java. Java - це об'єктно-орієнтована мова програмування. мовна граматика В основному з C і C++. В офіційній реалізації програма Java Скомпільований у байт-код, інтерпретований як віртуальний під час виконання Спеціальні для платформи машини. Oracle надає компілятор Java і Віртуальна машина Java, що відповідає специфікаціям спільноти Java процесу під ліцензією GNU. Java - це дуже гнучка мова з відкритим вихідним кодом. для розвитку. Проект має архітектурний підхід до вирішення проблем стандартної розробки програмного забезпечення. Шаблони-абстрактні схеми, вони не є кодом, але вони допомагають розділити логіку програми на певні модулі. При використанні шаблону проекту цей шаблон адаптується щоб задовольнити їхні конкретні потреби (рис. 2.4)

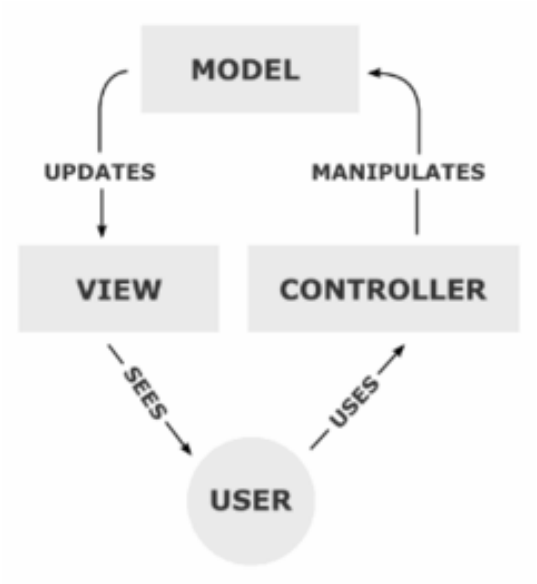


Рис.2.4. Шаблон користувач-контролер-модель-вид

Моделі - це дані, з якими працюють програми. вони можуть як дані бази даних, так і будь-яка інша структура даних, яка описує деякі дані системні об'єкти та їх стани.

Вид є компонентом системи відображення стану Моделі у зрозумілих репрезентаціях осіб. Це можуть бути діалоги, форми та інше (наприклад, синтезатор мовлення) Засоби, за допомогою яких люди взаємодіють із системою. Подання безпосередньо не змінює дані (режим лише для читання), дані змінюються з використанням контролера.

Контролер - це засіб, яким користується користувач для взаємодії з системою. Це може бути клавіатура, маніпулятор миші тощо.Він також контролює обмін даними та інформацією між видом та модел’ю.

Фактично, зв'язок між видом і контролером є інтерфейсом користувача.Крім того, якщо компоненти виду зазвичай можна знайти в інших компонентів системи, контролер зазвичай призначений для цієї конкретної ситуації.

Моделі не залежать ні від типів, ні від дозволу завдяки якому одночасно створюються різні інтерфейси для взаємодії з одним типом та сама моделлю даних. Наприклад, ви можете зробити це як Java SWT або SWING програм

настільних ПК, так і WEB-додатками для взаємодії між однаковими типами даних.

2.3.5 Система автоматизованої збірки Gradle

Для розробників і тестувальників програмного забезпечення процес створення програмних продуктів, без засобів автоматизації може бути дуже повторюваним, нудним і значно підвищує ризик помилки. Введіть кожен крок процесу розробки програмного забезпечення, починаючи з компіляції вихідного коду, остаточна збірка коду в програмне забезпечення для випуску та перевірки на виробництві це доводиться робити вручну. Допомагає автоматизація проектів зменшити кількість ручного втручання в проект і зробити процес розробки більше ефективним та мінімізує можливість збою програмного забезпечення.Автоматизація проекту має кілька явних переваг для процесу розробки програмного продукту. Перший – це штучна профілактика порушення під час складання. Коли ви повинні зробити це вручну, щоб створити доставку програмного забезпечення, це займає багато часу та наштовхуе розробника на ризик зробити помилку. Існує багато розробників і системних адміністраторів кожек з яких мае свої різноманітні завдання, які вони можуть вирішити замість ручної компіляції. Будь-який крок у процесі розробки має бути автоматизованим, якщо це можливо. Наступною перевагою є створення шаблонних збірок. Звичайна структура програмного проекту є відповідною попередньо визначеним та впорядкованим крокам. Наприклад, розробники Спочатку потрібно зібрати вихідний код, а потім запустити тести і згодом зібрати проект з потрібним результатом. Необхідно робити одні і ті ж дії знову і знову кожен день. Виконавчий процес повинен бути так само простим, як натиснути кнопку. Результат цього процесу необхідно повторити для всіх, хто виконує збірки проекту. Переваги також включають портативні шаблонні збірки. Можливість запускати збірки IDE дуже обмежена. По-перше, розробник програмного забезпечення має встановити на своєму пристрої певний продукт.

По-друге, IDE можна використовувати лише для певної операційної системи. Автоматизоване складання не вимагає спеціального середовища виконання незалежно від операційної системи чи CP. У кращому випадку автоматизоване завдання має виконуватися протягом 48 секунд з командної консолі, що дозволяє обирати будь-яке обладнання та в будь-який час для запуску процесу збірки.

Автоматизація проектів, у свою чергу, поділяється на кілька основних видів. Перший з них — збір за вимогою. Типовий випадок використання автоматизації на вимогу, коли користувачі починають побудову певному обладнанні. Як правило, перевіркою керує система контролю версій VSC колекцій та файлів вихідного коду. У більшості випадків виконується скрипт користувачем з командного рядка для виконання завдань на попередньо визначених порядках. Наприклад, щоб скомпілювати вихідний код, скопіюйте файли з першого каталогу до другого або збірка результатів. Зазвичай подібна автоматизація робиться кілька разів на день.

Розглянемо наступий тип автоматизації. Якщо розробник займається agile розробкою програмного забезпечення, то він зацікавлений у швидкому отриманні зворотного зв’язку про статус проекту. Також важливо знати, чи може вихідний код при компіляції не виявити помилки та надати інформацію про існування у проекті потенційних недоліків програмного забезпечення вказаних в блокових або тестових частинах. Цей тип автоматизації зазвичай працює в результаті перевірки системи контролю версій коду. Інший тип - планова автоматична збірка. Ідея полягає в тому, щоб запланувати автоматизацію як розклад завдань на тимчасовій основі. Він виконується через певний інтервал або в певний час. Автоматизація планування зазвичай виконується на виділених серверах. Цей тип автоматизації корисний для створення звітів або документацій для проекту. Практика реалізації планів та запуску Збірки, зазвичай визначаються як безперервна інтеграція CI.

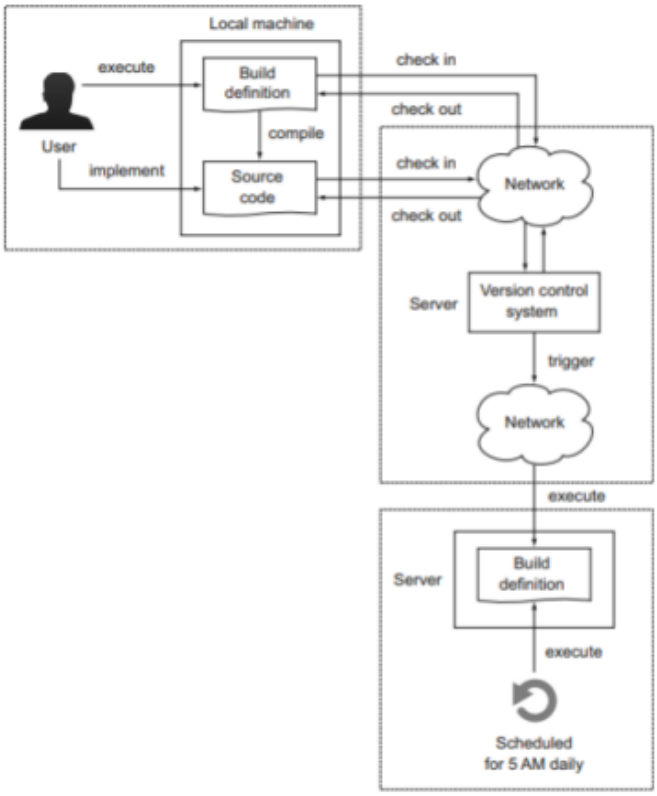


Рис.2.5. Приклад автоматичної збірки

На ряду з іншою повсякденною роботою розробника програмного забезпечення також стоїть і автоматизація проєкту. Сценарій складання Gradle є декларативним, легко читаним та зрозумілим. Використовуйте елементи філософії для написання коду на Groovy замість XMLВи можете використовувати Build-byconvention, щоб значно зменшити розмір сценарію складання.Його буде набагато легше читати, як показано на малюнку №6

Рис.2.6. Порівняння Gradle та Maven

Усі збірки Gradle починаються зі скрипту. Ім'я сценарію Gradle - build.gradle. У команді базової оболонки система збирання шукає файли з однаковими іменами. Якщо його не знайдено, буде виведено повідомлення про помилку. Після створення файлу необхідно визначити 1 або кілька завдань. В цілому, ви можете визначити, чи є Gradle автоматизованим Асемблерна система на основі GROOVYDSL, декларативна та виразна. Поєднуйте гнучкість та простоту розширення з ідеєю конфігурації Підтримка традиційного керування залежностями. Gradle буде найважливішим при створенні безлічі проектів із відкритим вихідним кодом.

2.4. Опис структури системи та алгоритмів її функціонування

2.4.1. Вибір схеми моделі життєвого циклу

Сьогодні існує багато стандартних шаблонів дизайну. Крок за кроком ви можете реалізувати будь-який проект від ідеї до дизайну. Ваш вибір

персоналізації залежить виключно від розробника та його цілей. Прикладами таких моделей є «модель водоспаду», модель «спіраль» і однією з таких моделей, які слід враховувати при розробці програми, є модель Walt Disney складається з 3 фаз проектування.

-концепт

-логіка

-фізика

Фази тривають одна за одною (Малюнок №7), але в окремих випадках можна перейти до наступного етапу, не завершуючи попередній. Наприклад, це може статися, якщо проект має більше одного розробника. Кожен працює в певній частині програми. У всякому разі, після фази фізичного проектування потрібно буде повертатися до початку і дороблювати пропущену частину.

Рис.2.7. Фази проектування

Концептуальний дизайн. Ефективність може бути важко оцінити Для того, щоб прийняти рішення, потрібно знати та розуміти критерії оцінювання. Це добре або погано розвинений ресурс. Є універсальні стандарти. Дуже точно характеризує ефективність програми - це результат розробників реалізація

поставлених перед ними цілей у цьому випадку реалізація перетворюється на високоякісний інструмент, який виконує поставлені завдання. Функція використовується для визначення цілей, завдань концептуального проектування. Опис програми та цільової аудиторії.

На цьому етапі проектування необхідно пояснити наступне:

-головні та другорядні цілі;

-план дій на шляху до виконання поставленної задачі;

-аудиторія програми;

-потреби для досягнення цілі;

-підрозділи програми;

-заінтересованість користувацької групи.

Визначивши цілі та інтереси користувача, можна створити список служб і відділів у Додатку. Логічне оформлення, додаткові розділи були визначені на попередньому кроці. Вони ще не організовані й структуровані, тому мають бути відповідно легкими для розуміння вигляду.

Логічне проектування включає організацію інформації в додатку. Побудова його структури та навігації за розділами. На цьому етапі потрібно задати питання про те, як організувати інформацію, може бути по різному і залежить від типу даних та уподобань автора програми. Розділи, алфавітний порядок, конкретні групи чи інші критерії.

Одночасне використання різних методів залучатиме більше громадськості. На цьому кроці ви зможете швидко знайти потрібну інформацію у своїй програмі. На даному кроці, потрібно пояснити наступне:

- Тип структури програми (лінійна, ієрархічна, контекстна тощо).

- Назва розділу.

- Що входить до кожного розділу.

- Організація роботи розділів та взаємного обміну інформацією.

- Розміщені елементи в певних частинах програми.

Кінцевим результатом логічного проектування є блок-схема або Структурна діаграма, що показує зв’язки між різними частинами програми.

Фізичний дизайн. Цей крок полягає в пошуку проблем, а не їх вирішенні. Процедура повинна пояснювати наступне:

- Технологія, яка використовується в додатку.
- програмне забезпечення, яке буде використовуватися.
- Можливі проблеми та способи їх вирішення.
- Як оновлюється інформація.

Після завершення цього кроку вам потрібно повернутися до концепції. Перевірте, чи потрібно внести зміни в дизайн та пов’язані з ним елементи Перегляньте проект на інших етапах.

Щоб обрати модель конкретного життєвого циклу (LC).

Тематичні сфери, ключові питання, інклюзії створених продуктів або не включення тривіальних робот. Сьогодні основою для формування нових моделей НЧ для конкретних застосувань. Система має стандарт ISO / IEC12207, який визначає повний набір процесів (більше 40). Охоплює всі можливі види робіт і завдань, пов’язаних із будівництвом програмного середовища (ПС), починаючи з аналізу предметної області і завершуючи виготовленням супутніх товарів. Стандарт включає основні та допоміжні процеси (Рис.2.8. та Рис.2.9.)

Рис.2.8. Основні процеси життевого циклу програмного середовища

На малюнку №8 показано процес, безпосередньо пов’язаний з розвитком ПС.Категорія базового процесу також включає "первинний" процес що відповідаее за визначення порядку розробки ПС, підготовка договорів та моніторинг діяльності постачальників ПС для замовника.

Стандарт ISO/IEC12207 надає структуру процесу життевого циклу, але не є обов'язковим використання всіх процесів в моделі життевого циклу програмного забезпечення або спеціальної методики розробки програмного забезпечення.

Рис.2.9. Допоміжні процеси життевого циклу програмного середовища

Під час створення додатку за основу був взят стандарт ISO/IEC12208. На цій основі була розроблена методика додатку зображена на рис.2.10.

Рис.2.10. Методика розробки програми

Методика розробки мобільного додатка включає в себе наступні кроки:

- Аналіз та опис завдання (планування) - визначення мети створення. застосунку, майбутні проблеми та цілі застосування, можливості застосування.

- Розробка технічних інструкцій - визначення рамок. підбір матеріалів та програмного забезпечення для його створення.

- Веб-дизайн - Макет програми, стиль і елементи навігації.

- Розробка програмного коду, модулів, баз даних та інших елементів. Застосунку потрібних в проекті, інтеграція обраного CMS.

- Заповнення застосунку зібраними матеріалами.

- Практичний тест.

- Запуск.

2.4.2. Опис програмної реалізації

Система підрахунку буде складатися з наступних вікон керування:

1) Вікно реєстрації або авторизації у додатку
2) Вікто з підрозділами облікового запису
3) Головне меню з виведеною всією інформацією що до операцій
4) Вікно додання операції
5) Вікно редагування операції
6) Вікно пошуку протрібної операції

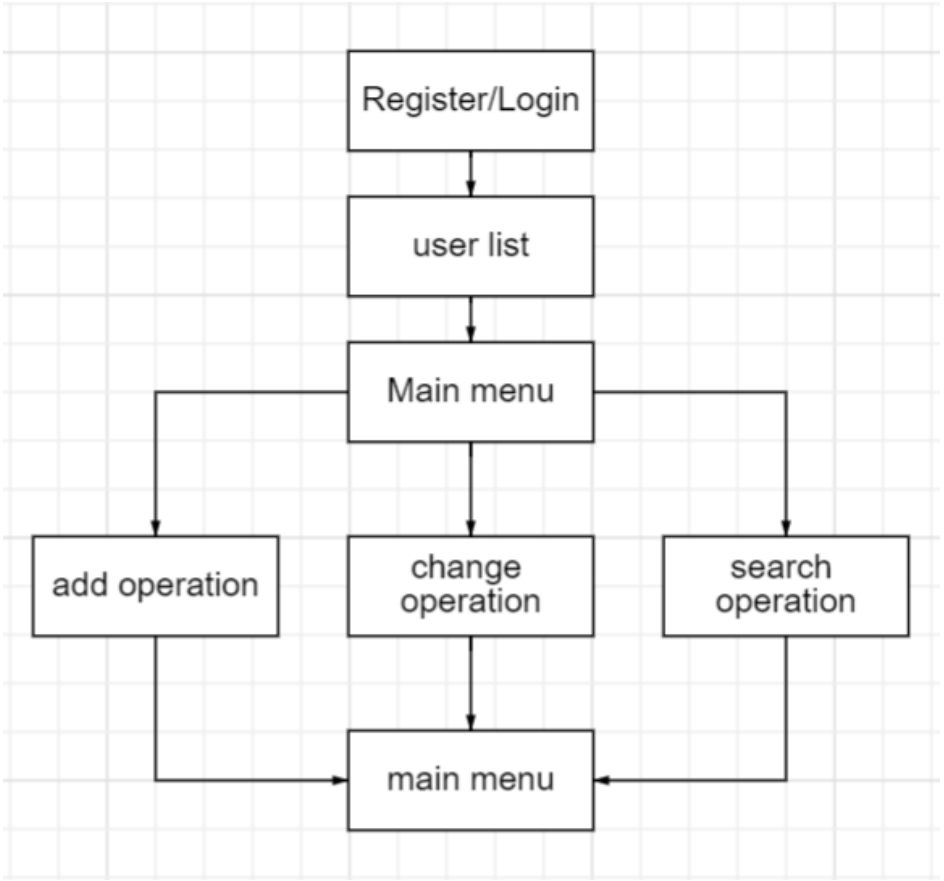


Рис.2.11. Схематично зображена структура системи

2.4.3. Опис функціональності системи

Система локального підрахунку коштів передбачує її використання однією людиною без взаемодії з додатковими кадрами, окрім технічної підтримки для урегулювання недоліків, які було виявлено під час використання

додатку.Використання застосунку можливе з використанням смартфону на базі операційної системи Android

2.4.4. Розробка інтерфейсу користувача

UI розробляемого застосунку має відповідати певним параметрам:

1) Інтуітивно зрозумілі органи управління.
2) Очевидний вивід інформації на екран.
3) Кольорова гамма яна не дає навантаження на зір.
4) Зрозумілість послідовності для виконання потрібних дій.

Для виконання цих параметрів, був розроблений UI програмного застосунку.
Користувацький інтерфейс включає в себе декілька вікон, між якими користувач може пересуватися в залежності від потреби.
Як тільки користувач запускае додаток, його зустрічае перше вікно реєстрациі у додатку та запитуе певну інформацію(електронна пошта, ім'я, пароль тощо) Якщо користувач вже мае обліковий запис у додатку, він може натиснути кнопку авторизації. Після регістрації/авторизації у додатку, користувач потрапляє на наступне вікно, а саме на вікно вибору чи створення підрозділу користувача, наступним вікном вже буде головне меню, де користувач може нажати на значок «плюс» та перейде у вікно для створення нового запису що до операції також на головному меню можна переглянути раніше створені фінансові операції разом з поточним балансом. Знаходячись на головному меню користувач може також відредагувати будь-яку створену раніше операцію у спеціальному вікні.

2.4.4.1. Вікно реєстрації

Це вікно яке зустрічае користувача при першому запуску застосунку,
Воно містить в собі всі необхідні поля запиту інформації для ппроходження реєстрації, а саме:

1) Поле для вводу електронної пошти.
2) Поле для вводу нікнейму.
3) Поле для вводу імені користувача.
4) Поле для вводу пароля.
5) Поле для підтвердження паролю.
6) Кнопку реєстрації.
7) Кнопку входу, якщо аккаун був створений раніше.

2.4.4.2. Вікно авторизації

Це вікно для входу в вже існуючий запис користувача, яке зустрічає його при натисканні на кнопку входу у вікні реєстрації. Вікно містить в собі запити на інформацію для входу в обліковий запис, а саме:

1) Поле для вводу електронної пошти.
2) Поле для вводу паролю.
3) Кнопку входу.
4) Кнопку для переходу на сторінку реєстрації, якщо потрібно створити новий обліковий запис.

2.4.4.3. Вікно вибору підрозділу облікового запису

Це вікно надае можливість користувачеві обрати або вже створений підрозділ облікового запису, або створити новий. Для створення підрозділу користувача, ніякої інформації не запитується. Вікно складається з:

1) Перелік всіх існуючих підрозділів користувача.
2) Кнопку для створення нового підрозділу.

2.4.4.3.1. Вікно створення підрозділу

Це вікно запитує тільки найменування нового підрозділу, складається з:

1) Поле вводу найменування підрозділу.
2) Кнопку створення підрозділу.

2.4.4.4. Головне меню

Це вікно, яке користувач бачить першим, після вибору аккаунту, Воно містить в собі всю зібрану інформацію що до фінансових операцій, які користувач раніше вказував у додатку

Головне меню складається з:

1) Текстовий напис с поточним балансом.
2) Перелік всіх створених у додатку операцій.
3) Кнопки для додання нової операції.

2.4.4.5. Створення нової категорії

Це вікно викликається натисканням кнопки «плюс» на екрані головного меню та містить в собі поля для сбору інформації яка необхідна для створення нової операції, а саме:

1) Чекбокс з вибором типу операції.
2) Поле для вводу суми операції.
3) Спливаючий список з вибором категорії операції.
4) Поле+календар для вибору дати проведення операції.
5) Поле для вводу коментаря до операції.
6) Кнопка зберегти.
7) Кнопка назад, що повертае користувача у меню.

2.4.4.6. Редагування існуючої операції

Це вікно викликається натисканням на одну з операцій представлених у головному меню, які користувач раніше створював самостійно. Вікно дозволяє змінити будь-яку інформацію, яку ви вказували при створенні обраної операції, окрім її типу. Для цього вікно містить в собі:

1) Поле для зміни суми.
2) Спливаючий список для зміни категорії.
3) Поле+календар для зміни дати проведення.
4) Поле для зміни коментаря.
5) Кнопка збереження інформації.
6) Кнопка видалення операції з підтвердженням на видалення.

2.4.5. Розробка додатку

Першим кроком до створення програми для Android є налаштування СР. На щастя, цей процес можна легко виконати за допомогою інструментів AndroidSDK. Достатньо завантажити Android Studio (Рис.2.12.). Встановіть його у вашій системі. Після завершення встановлення та завантаження, усі необхідні інструменти доступні для запуску вашого проекту Android.

Наступним кроком ми маємо створити новий проект у AndroidStudio. Увімкніть підтримку Java у діалоговому вікні, що з’явиться, тому що це мова, яку ми використовуємо.

Потім завантажуємо операційну систему, яку ми будемо використовувати при розробці нашого застосунку. Я обрав операційну систему Android Oreo з підтримкою API 26 рівня, оскільки створювана програма не вимагає високих рівнів API інтерфейсів та сама використовуе 21 рівень, тому була зроблена ставка на підтримку більшої кількості пристроїв.

Останньою частиною цього пункта буде створення проекту у раніше встановленій СР Android studio. На цьому етапі студія нам пропонуе вибір з

багатьох пресетів на створення програми (Рис.2.13.). Був обраний пресет «basic activity» який найкраще підходить до поставленої задачі.

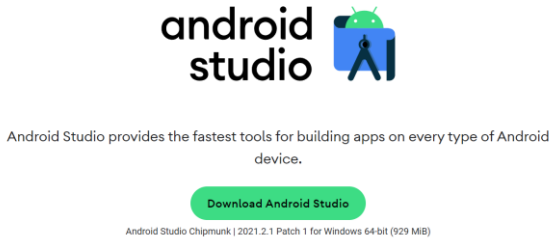


Рис.2.12. Завантаження безкоштовної CP Android Studio

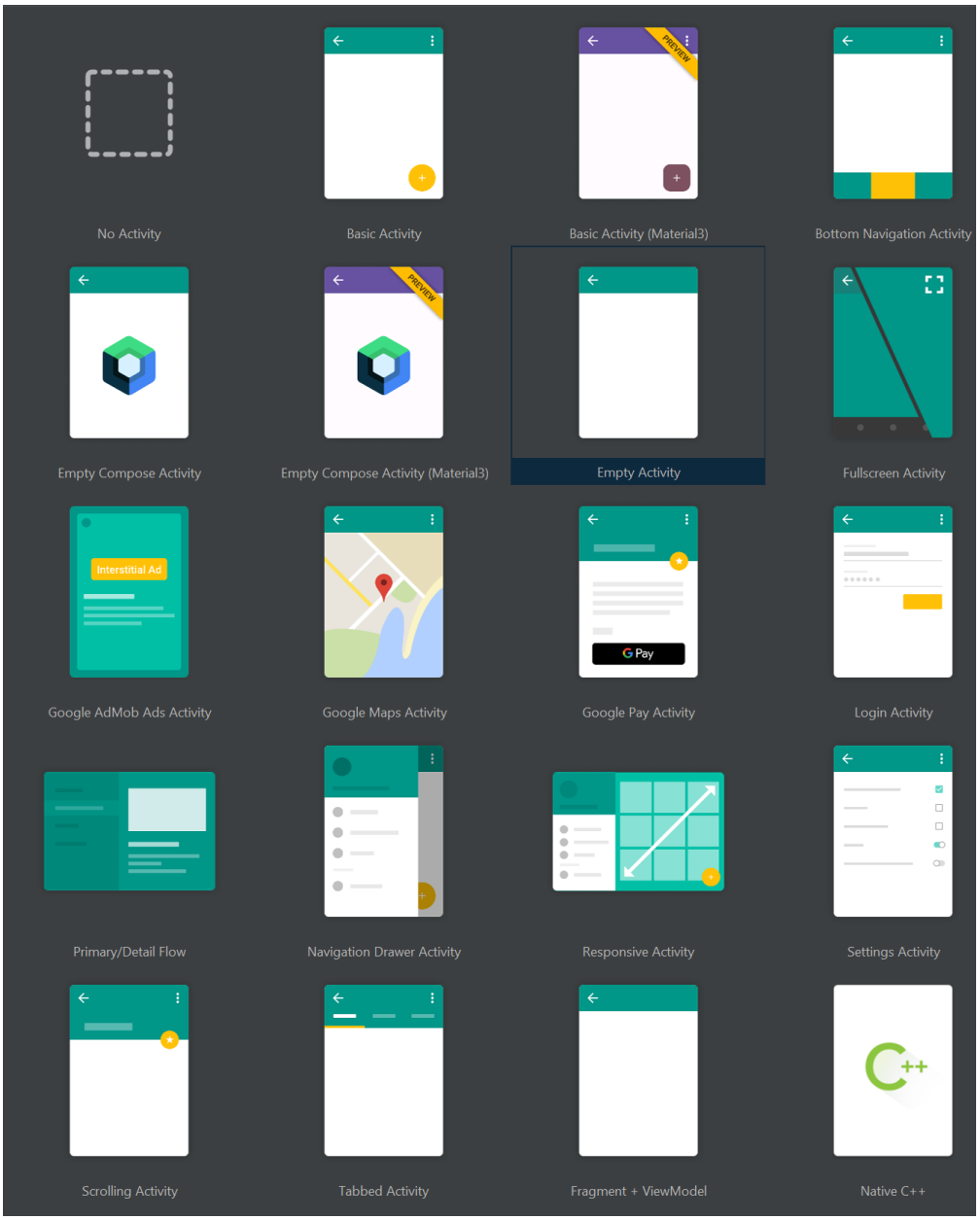


Рис.2.13. Широкий вибір пресетів для проекту

Перший етап після створення проекту це створення необхідних файлів, каталогів та їх ієрархічна структуризація.

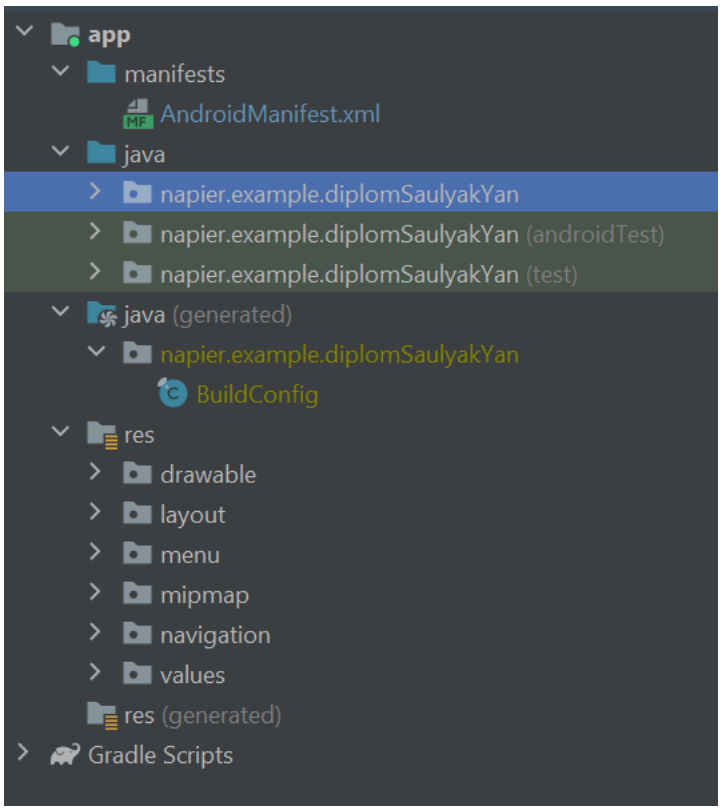


Рис.2.14. Структуризація проектних каталогів

Ніяка розробка ніякого програмного забезпечення не обійдеться без постійних запусків та тестів прототипів. Так як ми розробляемо мобільний додаток, це повинен бути відповідно телефон, цей телефон повинен буди на базі операційної системи Android. Саме для того, щоб зробити процесс тестування додатку максимально зручнішим та швидшим, обрана нами СР підтримує використання віртуальних машин завдяки AVD(Android Virtual Device) емулятору. Створений віртуальній телефон (Рис.2.15.) ні краплі не обмежує нас у тестуванні додатку, він має абсолютно всі ті-ж функції, що і звичайний мобільний телефон. Використання такого методу тестування допоможе робити це швидше та зручніше, адже дані програми не передаються по проводу у фізичний девайс, а одразу запускаются використовуючи потужності компьютеру.

Рис.2.15. Емулятор мобільного телефону AVD

Коли вже розібралися і з створенням проекту і налаштували віртуальний пристрій, перчим чином потрібно побудувати перший екран, це екран реєстрації у додатку, який перший зустрічае користувача при запуску програми. Для цього створюємо файл «fragment_register.xml» (Рис.2.16.)

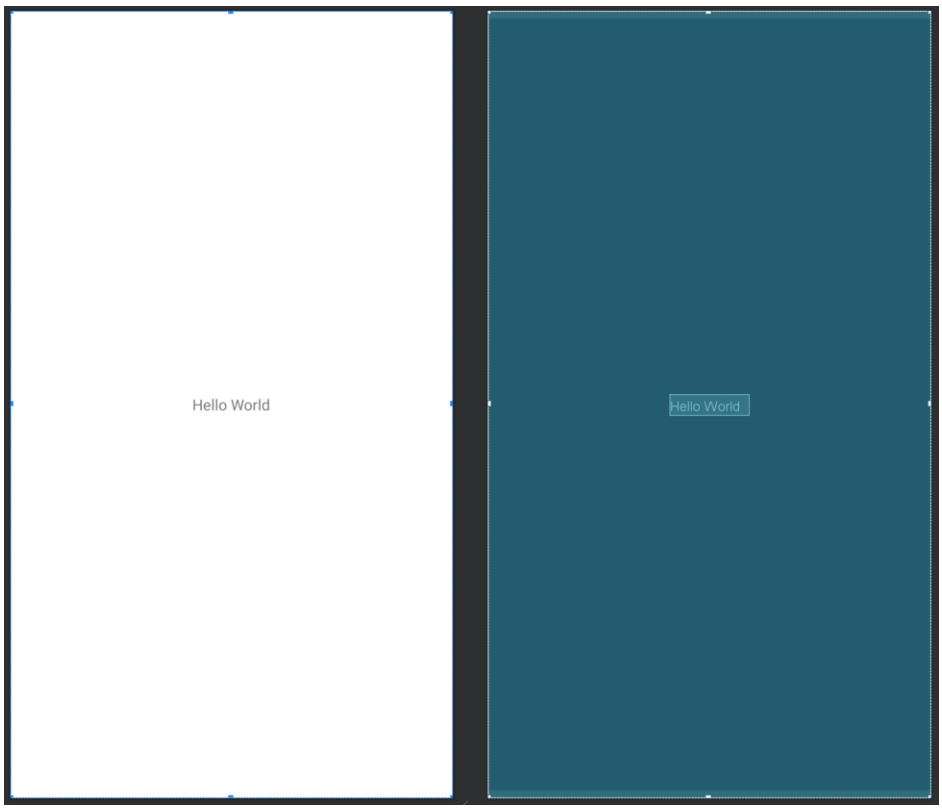


Рис.2.16. Пустий макет регістраційного вікна

Використовуючи інтегрований редактор інтерфейсу, створюємо всі необхідні поля для реєстрації та кнопки. (Рис.2.17.)

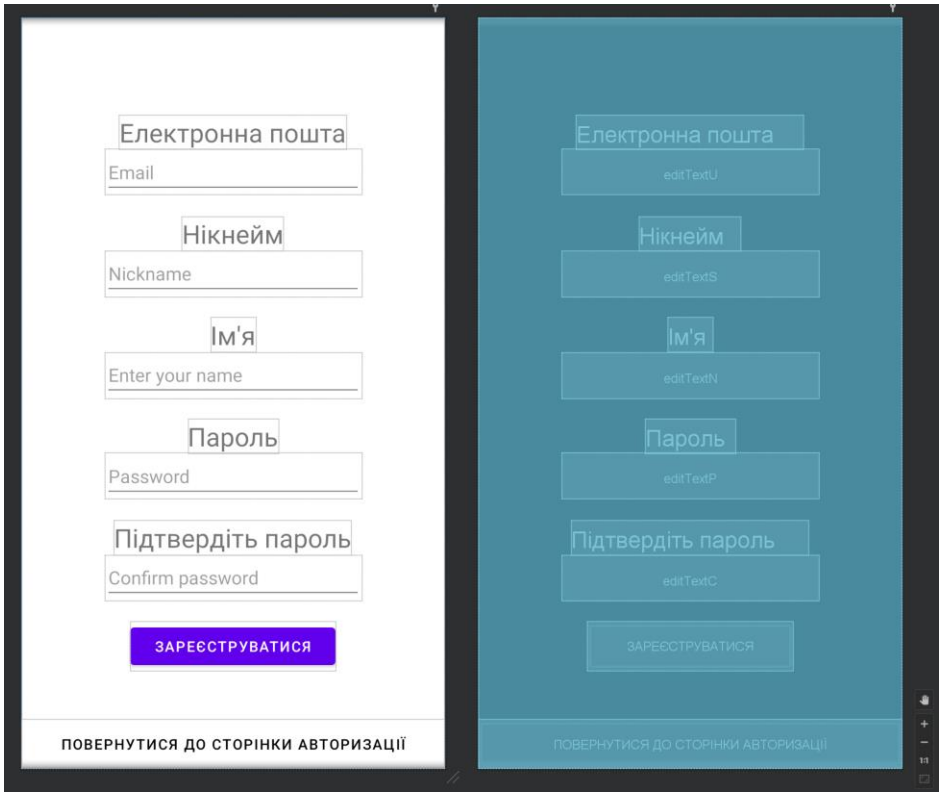


Рис.2.17. Готова сторінка реєстрації

Після створення оболонки реєстраційного вікна, потрібно зробити йому функціонал, для цього був створений Java клас RegisterFragment, у який був записан наступний код (Рис.2.18.)

```
@Override
public void onClick(View v) {
    switch(v.getId()){
        case R.id.button3:
            Bundle bundle = new Bundle();
            bundle.putString("client_id",editText1.getText().toString());
            Navigation.findNavController(getView()).navigate(R.id.action_registerFragment_to_loginFragment,bundle);
            break;
        case R.id.buttonRegister:
            String edit1 = editText1.getText().toString();
            String edit2 = editText2.getText().toString();
            String edit3 = editText3.getText().toString();
            String edit4 = editText4.getText().toString();
            String edit5 = editText5.getText().toString();
            if(edit1.equals("") || edit2.equals("") || edit3.equals("") || edit4.equals("") || edit5.equals("")){
                showSnackBar("Please fill all the inputs");
            }
            else if(!edit4.equals(edit5)){
                showSnackBar("Please enter same password");
            }
            else if(edit4.length()<5){
                showSnackBar("Please enter a password of at least 5 characters");
            }
            else{
                boolean condi = false;
                for(String name[] : names){
                    if(name[0].equals(edit1)){
                        condi = true;
                    }
                }
                if(condi==true){
                    showSnackBar("Enter another username");
                    editText1.setText("");
                }
                else{
                    dm.insert(edit1,edit2,edit3,edit4);
                    Navigation.findNavController(getView()).navigate(R.id.action_registerFragment_to_accountFragment);
                }
            }
            break;
    }
}
```

Рис.2.18. Частина коду класу реєстрації

Після налаштування екрану реєстрації нам необхідно наступним пунктом зробити екран авторизації, для цього ми виконуемо наступні кроки: як ми вже робили, створюемо файл «fragment_login.xml» та відкриваемо його у редакторі дизайну для встановлення необхідних елементів керування і будуемо дизайн вікна (Рис.2.19.)

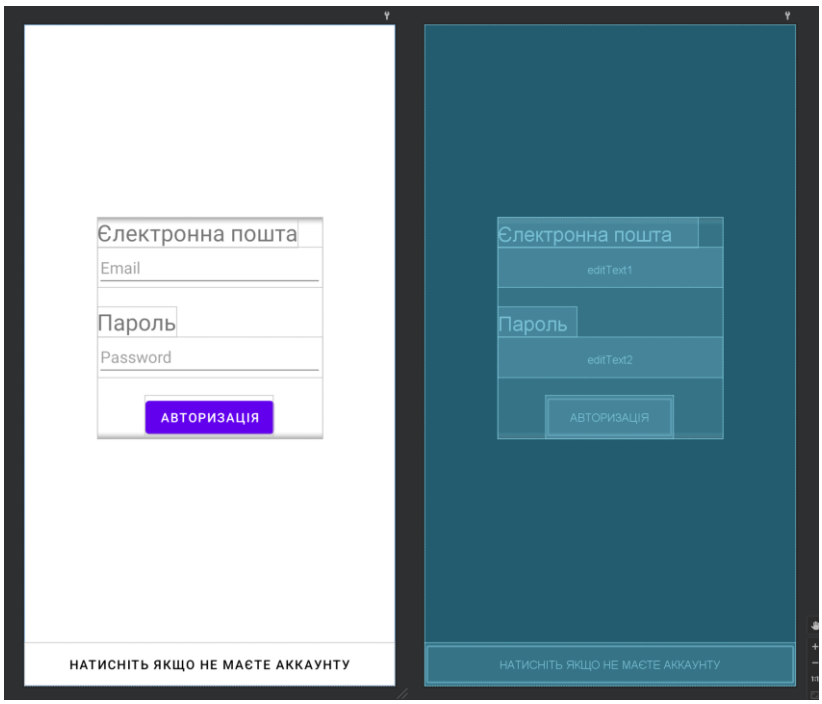


Рис.2.19. Вигляд вікна авторизації

Після створення сторінки, створюємо також відповідний клас для задання функціоналу єлементам керування.(Рис.2.20.)

```
@Override
public void onClick(View v) {
    switch (v.getId()){
        case R.id.button1:
            String login = editText1.getText().toString();
            String password = editText2.getText().toString();
            boolean condi = false;
            for(int i=0;i<clientIds.size();i++){
                if(login.equals(clientIds.get(i).getUsername()) && password.equals(clientIds.get(i).getPassword())){
                    condi = true;
                }
            }
            if(condi==true){
                Bundle bundle = new Bundle();
                bundle.putString("client_id",login);
                Navigation.findNavController(getView()).navigate(R.id.action_loginFragment_to_accountFragment, bundle);
            }
            else{
                if(login.equals(""))
                    showSnackBar("Введіть ім'я користувача");
                else if(password.equals(""))
                    showSnackBar("Введіть пароль");
                else{
                    showSnackBar("Помилковій пароль, або ім'я");
                    editText1.setText("");
                    editText2.setText("");
                }
            }
            break;
        case R.id.button2:
            Navigation.findNavController(getView()).navigate(R.id.action_loginFragment_to_registerFragment);
            break;
    }
}
```

Рис.2.20. Частина коду класу авторизації

Після того, як ми закінчили з реєстрацією та авторизацією, нам потрібно зробити домашню сторінку(меню) з списком операцій та можливістю їх додавати/редагувати(Рис.2.21.). Для цього також створюємо файл «fragment_home.xml» та редагуємо його за допомогою інтегрованої Drag&Drop системи побудови UI (Рис.2.22.)

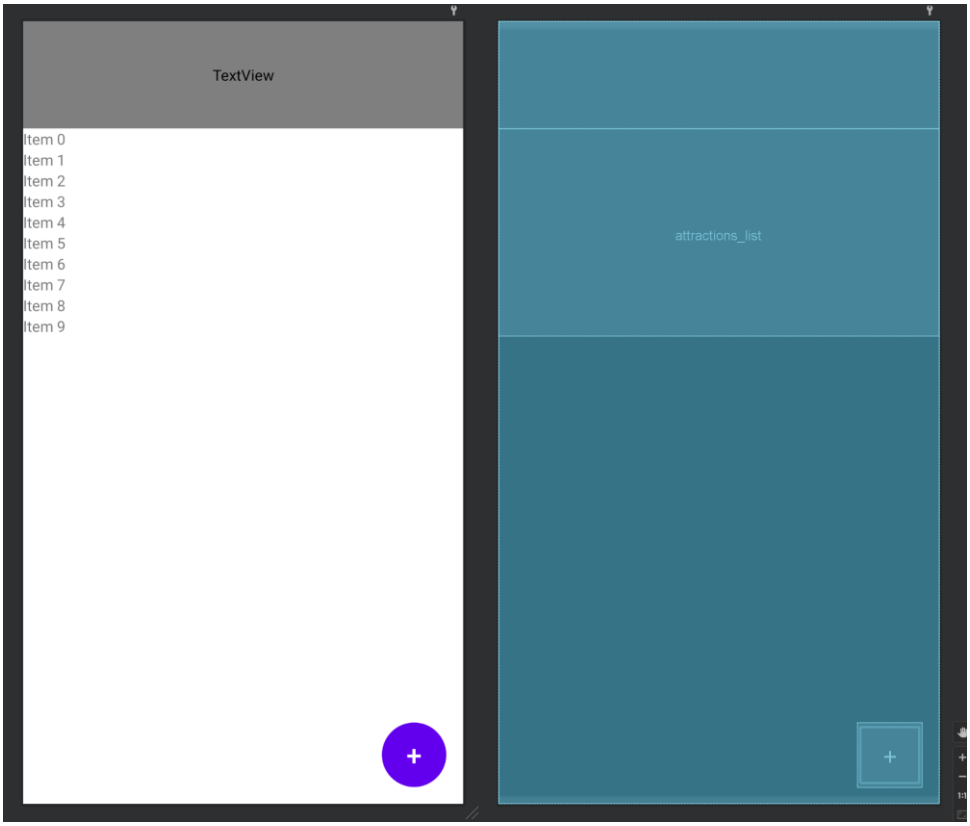


Рис.2.21. Вигляд домашньої сторінки

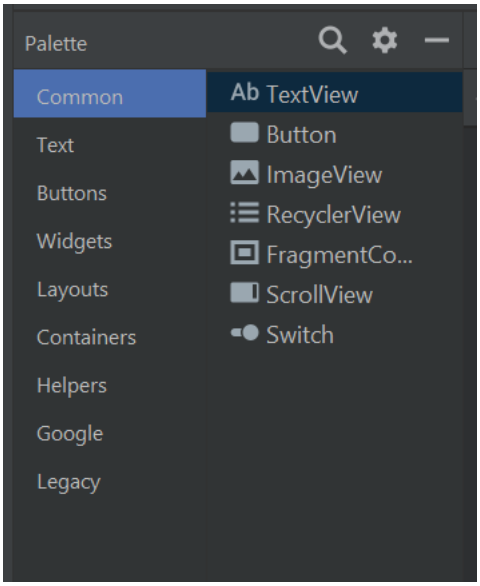


Рис.2.22. Drag&Drop система побудови UI

Після створення дизайну, створюємо джава клас та прописуємо функціонал для домашньої сторінки (Рис.2.23.)

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_account, container, false);

    LayoutInflater vi = (LayoutInflater)getActivity().getApplicationContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    LinearLayout insertPoint = view.findViewById(R.id.insert_point);
    dataBaseAccount = new DataBaseAccount(getActivity());
    names = dataBaseAccount.selectAll();
    for(String[] name: names){
        Log.i( tag: "message", msg: "Here : "+name[0]+" "+name[1]+" "+name[2]);
    }
    int i=0;
    for(String[] name: names){
        if(name[2].equals(getArguments().getString( key: "client_id"))){
            View v = vi.inflate(R.layout.blocks_account,null);
            //Log.i("message","hello from "+name[0]);
            Button button = v.findViewById(R.id.testText);
            button.setText(name[1]);
            final String stp = name[0];
            insertPoint.addView(v,i,new ViewGroup.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT, ViewGroup.LayoutParams.WRAP_CONTENT));
            i++;
            button.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    Bundle bundle = new Bundle();
                    bundle.putString("account_id",stp);
                    Navigation.findNavController(getView()).navigate(R.id.action_accountFragment_to_menuFragment,bundle);
                }
            });
        }
    }
    if(names.size()<9){
        View v = vi.inflate(R.layout.blocks_account,null);
        Button button = v.findViewById(R.id.testText);
        button.setText("Додати обліковий запис");
        button.setId(R.id.view1);
        button.setOnClickListener(this);
        insertPoint.addView(v,i,new ViewGroup.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT, ViewGroup.LayoutParams.WRAP_CONTENT));
    }
    return view;
}
```

Рис.2.23. Частина коду клас головного меню

Далі у нас йде вікно створення нової операції, яке буде відкриватися по натисканню плюса на домашній сторінці. Для цього створюємо файл «activity_add_transaction.xml» та створюємо у ньому UI сторінки (Рис.2.24.)

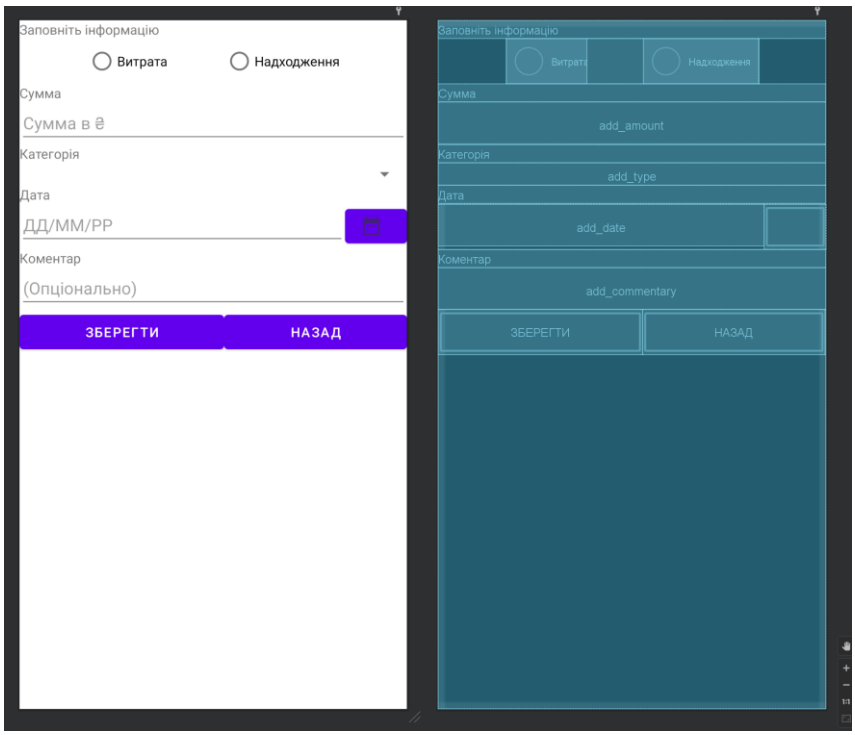


Рис.2.24. Дизайн сторінки створення нової операції

Як і з попередніми сторінками, після розробки користувацького інтерфейсу сторінки, зараз необхідно їй надати певний функціонал, для цього створюемо окремий клас та прописуемо потрібні функції (Рис.2.25.)

```
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.add_btnBack:
            this.finish();
            break;
        case R.id.add_btnSave:
            String amount = ((EditText) findViewById(R.id.add_amount)).getText().toString();
            String type = ((Spinner) findViewById(R.id.add_type)).getSelectedItem().toString();
            String date = ((EditText) findViewById(R.id.add_date)).getText().toString();
            String commentary = ((EditText) findViewById(R.id.add_commentary)).getText().toString();
            if(amount.equals("") || amount.charAt(0)=='-'){
                Toast.makeText( context: this,  text: "Введіть кількість", Toast.LENGTH_SHORT).show();
            }
            else if(type.equals("") || type.equals("--Choose Type--")){
                Toast.makeText( context: this,  text: "Оберіть категорію транзакції", Toast.LENGTH_SHORT).show();
            }
            else if(date.equals("")){
                Toast.makeText( context: this,  text: "Зазначте дату", Toast.LENGTH_SHORT).show();
            }
            else{
                if(commentary==null)
                    commentary = "";
                this.dm = new DataBaseTransac( context: this);
                long identification;
                String stringToReturn;
                if (rb0.isChecked()){
                    identification = this.dm.insert(amount, type, date,  idt: "0", id_account, commentary);
                    stringToReturn = amount+" - "+type+" - "+date+" - "+identification+" - 0 - "+id_account+" - "+commentary;
                }
                else{
                    identification = this.dm.insert(amount, type, date,  idt: "1", id_account, commentary);
                    stringToReturn = amount+" - "+type+" - "+date+" - "+identification+" - 1 - "+id_account+" - "+commentary;
                }
                Intent returnIntent = new Intent();
                returnIntent.putExtra( name: "String", stringToReturn);
                setResult( resultCode: 1, returnIntent);
```

Рис.2.25. Частина коду класу додання операції

Після того, як ми закінчили с додаванням операцій, потрібно реалізувати також вікно, де ми зможемо ці операції редагувати, яке відкриватиметься по натисканню на операцію розташовану на домашній сторінці. Для цього знову створюємо файл «activity_transaction_details.xml» та редагуємо його під наші потреби (Рис.2.26.)

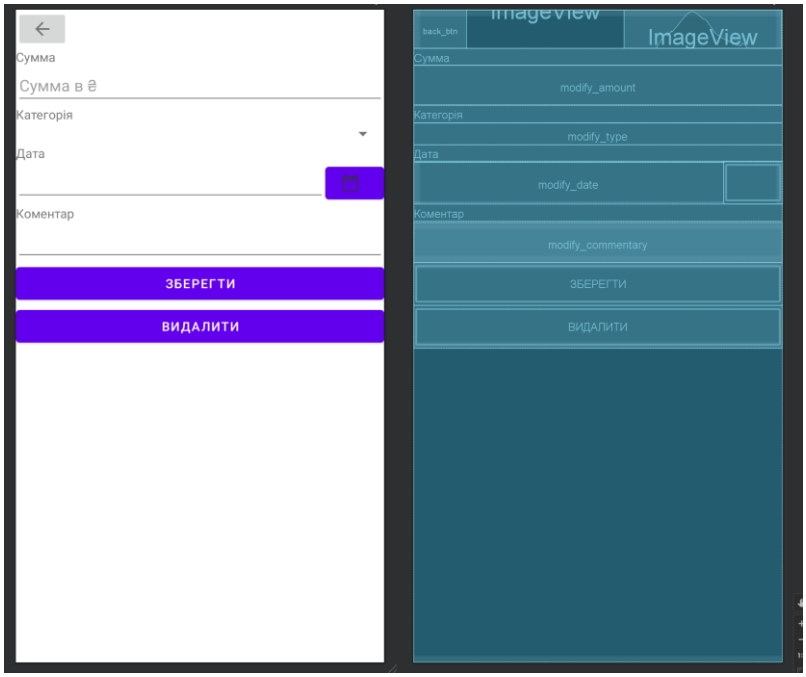


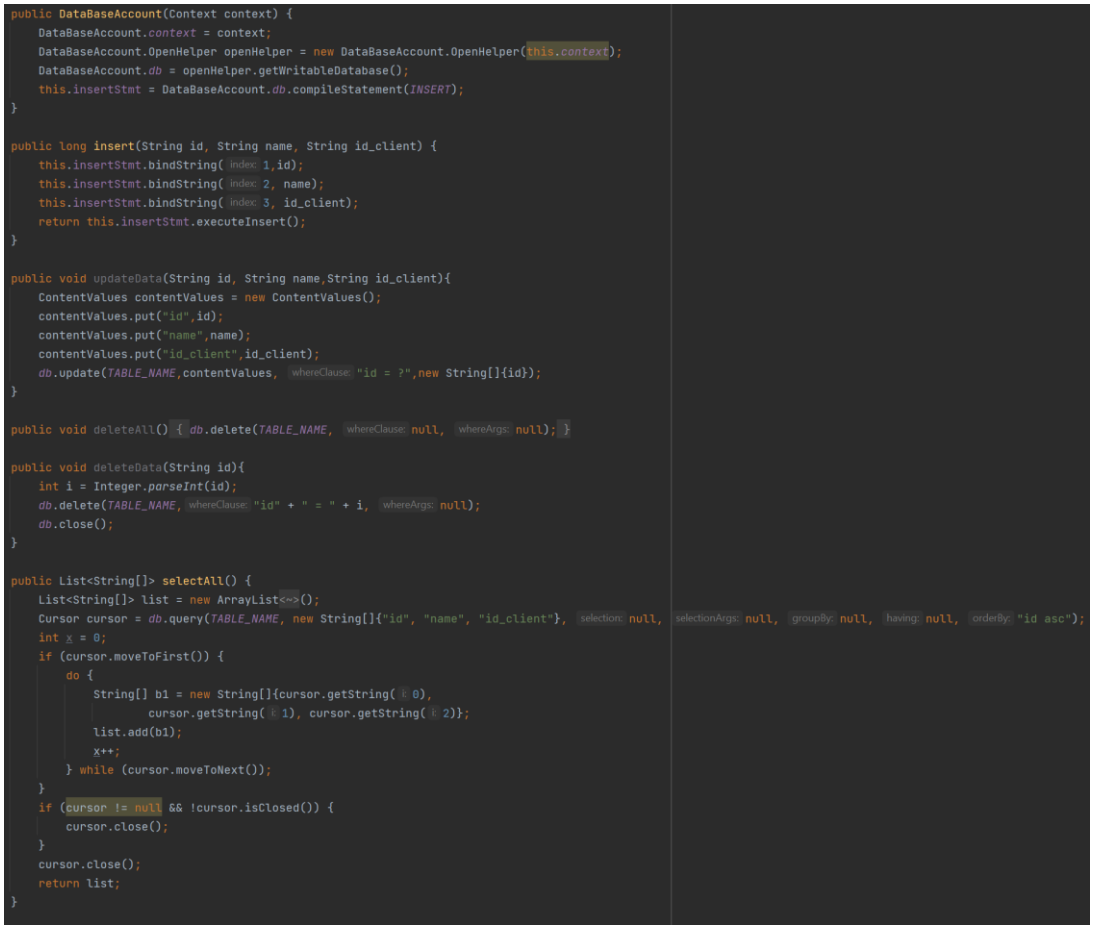
Рис.2.26. Вигляд сторінки редагування операції

Та одразу ж надаємо новій сторінці потрібного функціоналу з використанням нового створенного джава класу (Рис.2.27)

```
if (transaction != null) {
    View saveButton = findViewById(R.id.save_btn);
    saveButton.setOnClickListener(this);
    View back = findViewById(R.id.back_btn);
    back.setOnClickListener(this);
    View delete = findViewById(R.id.delete_btn);
    delete.setOnClickListener(this);

    ImageView imageView = (ImageView)findViewById(R.id.type_image);
    imageView.setImageDrawable(getResources().getDrawable(transaction.getIconId()));

    Button date_pick = (Button) findViewById(R.id.modify_picker);
    date_pick.setOnClickListener(this);

    Spinner spinner = findViewById(R.id.modify_type);
    ArrayAdapter<CharSequence> adapter;
    if(transaction.getIdt().equals("0"))
        adapter = ArrayAdapter.createFromResource( context: this, R.array.type_pick, android.R.layout.simple_spinner_item);
    else
        adapter = ArrayAdapter.createFromResource( context: this, R.array.type_pick2, android.R.layout.simple_spinner_item);
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    spinner.setAdapter(adapter);
    spinner.setOnItemSelectedListener(this);

    editText1 = (EditText) findViewById(R.id.modify_amount);
    editText2 = (Spinner) findViewById(R.id.modify_type);
    editText3 = (EditText) findViewById(R.id.modify_date);
    editText4 = (EditText) findViewById(R.id.modify_commentary);

    editText1.setText(transaction.getAmount());
    editText3.setText(transaction.getDate());
    editText4.setText(transaction.getCommentary());
    if(transaction.getIdt().equals("0"))
        switch(transaction.getType()){...}
    else
        switch(transaction.getType()){...}
```

Рис.2.27. частина коду редагування операції

Також для зберігання авторізаційних даних користувачів та даних що до їх фінансової активності, була створена локальна база даних(Рис.2.28.)



```
public DataBaseAccount(Context context) {
    DataBaseAccount.context = context;
    DataBaseAccount.OpenHelper openHelper = new DataBaseAccount.OpenHelper(this.context);
    DataBaseAccount.db = openHelper.getWritableDatabase();
    this.insertStmt = DataBaseAccount.db.compileStatement(INSERT);
}

public long insert(String id, String name, String id_client) {
    this.insertStmt.bindString( index: 1,id);
    this.insertStmt.bindString( index: 2, name);
    this.insertStmt.bindString( index: 3, id_client);
    return this.insertStmt.executeInsert();
}

public void updateData(String id, String name,String id_client){
    ContentValues contentValues = new ContentValues();
    contentValues.put("id",id);
    contentValues.put("name",name);
    contentValues.put("id_client",id_client);
    db.update(TABLE_NAME,contentValues, whereClause: "id = ?",new String[]{id});
}

public void deleteAll() { db.delete(TABLE_NAME, whereClause: null, whereArgs: null); }

public void deleteData(String id){
    int i = Integer.parseInt(id);
    db.delete(TABLE_NAME, whereClause: "id" + " = " + i, whereArgs: null);
    db.close();
}

public List<String[]> selectAll() {
    List<String[]> list = new ArrayList<>();
    Cursor cursor = db.query(TABLE_NAME, new String[]{"id", "name", "id_client"}, selection: null, selectionArgs: null, groupBy: null, having: null, orderBy: "id asc");
    int x = 0;
    if (cursor.moveToFirst()) {
        do {
            String[] b1 = new String[]{cursor.getString( i: 0),
                    cursor.getString( i: 1), cursor.getString( i: 2)};
            list.add(b1);
            x++;
        } while (cursor.moveToNext());
    }
    if (cursor != null && !cursor.isClosed()) {
        cursor.close();
    }
    cursor.close();
    return list;
}
```

Рис.2.28. база даних для авторизації

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Розроблений програмний засіб використовує лише вхідну інформацію, яку користувач самостійно вписує у певні поля запити на певних сторінках програмного застосунку. Зі сторонніх ресурсів та віддалених серверів застосунок ніякої інформації не отримує.

Вихідною інформацією є відображення підведеного виходячи з усіх вказаних користувачем операцій поточного балансу та перелік всіх операцій які користувач вводить в застосунок самостійно.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Під час проектування, розробки та тестування програмного застосунку «kash keeper» було застосовано персональний комп’ютер з характеристиками:

CPU – AMD ryzen 5 2600

RAM – 16GB ddr4

GPU – NVIDIA RTX2060

HDD – 1TB

SSD – 250GB

Пристрої вводу – клавіатура + миша

Пристрої виводу – монітор 31”

Операційна система – Windows 10 pro

2.6.2 Використані програмні засоби

Для розробки програмного застосунку “kash keeper” було використано:

1) Мова програмування Java.
2) Середа розробки Android Studio.
3) Технологія автоматичної збірки Gradle.
4) Графічний редактор Figma.

2.6.3 Виклик та завантаження програми

Виклик та тестування програмного застосунку відбувається в середовищі розробки під ОС Android – Android Studio.У вільний доступ, на платформи публічного росповсюджування застосунків по типу Play Market та інші, дана програма не була викладена.Для запуску програми на сторонніх мобільних пристроях потрібно вивантажити з середи розробки APK файл та надати його користувачеві.

2.6.4 Опис інтерфейсу користувача

Програма підрахунку фінансових операцій розроблена для використання на мобільних пристроях на базі операційної системи Android. Підключення до облачних сервісів та інтернету не потребує.

2.6.4.1 Інсталяція та системні вимоги

Для встановлення додатку на пристрій користувача потрібно завантажити APK файл даної програми. Оскільки додаток розроблявся під ОС Android, Пристрій користувача повинен керуватися саме цією операційною системою, яка повинна бути версією не нижче Android 5.0.0.(Lolipop).

2.6.4.2 Інструкція з використання програмного продукту

При запуску додатка, користувач бачить екран авторизації, якщо користувач вже має обліковий запис, то він вводить пошту та пароль у підписані строки запиту даної інформації. Якщо користувач не має облікового запису, то він натискае на кнопку знизу «натисніть, якщо не маєте облікового запису», після чого користувач потрапляє на сторінку реєстрації, де вводить пошту, нікнейм, ім’я, пароль та підтвердження паролю у спеціальні підписані строки запиту інформації. Після авторизації користувача у своему обліковому записі, він потрапляє на вікто створення або вибору підрозділу, якщо користувач вже має створений підрозділ, то він на нього натискае і потрапляє то головного меню, якщо користувач не має підрозділів, або хоче створити новий, то він натискає на кнопку «створити підрозділ», перед ним відкривається вікно із однією строкою запита на найменування підрозділу, яку користувач заповняє, створюює підрозділ, обирає його та потрапляе у головне меню.

Коли користувач потрапив в головне меню, він побачить зверху свій поточний баланс, а під балансом буде розташований список всіх транзакцій, які користувач раніше додавав, і там-же будуть відображені всі подальші записані

операції. Знизу праворуч у головному меню розташована кругла кнопка «плюс», натиснувши на яку, користувач потрапляє на сторінку створення операції, де бачить підписані поля запиту суми операції, категорії, характеру, дати проведення та коментаря до операції, користувач заповнює всі обов’язкові поля (коментар не обов’язковий) і натискає або кнопку «підтвердити» щоб зафіксувати вказану операцію, або на кнопку «назад», щоб відхилити і стерти всю вище вказану інформацію і не фіксувати операцію.

Знаходячись на головному меню, користувач може натиснути на будь-яку з операцій, які відображені у списку та потрапити на вікно редагування операції, де він побачить заповнені поля запиту в яких міститься інформація, що користувач вказував при створенні обраної операції, яку він може відредагувати просто натиснувши на потрібне йому поле на змінивши вказані дані, окрім характеру операції.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1.Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

– передбачуване число операторів – 1010;

– коефіцієнт складності програми – 2.1;

– коефіцієнт корекції програми в ході її розробки – 0,07;

– годинна заробітна плата програміста, грн / год – 70;

– коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,2;

– вартість машино-годин ЕОМ – 7 грн/год.

- Коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі (1,2...1,5) - 1,4.

Трудомісткість розробки ПЗ розраховується за формулою:

$t = to + tu + ta + tn + t$отл$ + t$д , людино-годин, (3.1)

де to - витрати праці на підготовку й опис поставленої задачі (приймається 50),

tu - витрати праці на дослідження алгоритму рішення задачі,

ta - витрати праці на розробку блок-схеми алгоритму,

tn - витрати праці на програмування по готовій блок-схемі,

tотл - витрати праці на налагодження програми на ЕОМ,

tд - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів:

$Q = qC(1+p)$, людино-годин, (3.2)

де q – передбачуване число операторів,

С – коефіцієнт складності програми,

р – коефіцієнт корекції програми в ході її розробки.

Q=1010*2.1*(1+0,07) = 2 269.47 , людино-годин

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t = \frac{Q*B}{(75..85)*k}, \text{ людино-години,} \quad (3.3)$$

де B, яке дорівнює 1,4 - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі,

k, яке дорівнює 1,2, - коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності.

$$t = \frac{2269.47 * 1,4}{80 * 1,2} = 33.09 \text{ , людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t = \frac{Q}{(20..25)*k}; \quad (3.4)$$

$t = \frac{2269.47}{20*1,2} = 94,56$, людино-годин.

Витрати на складання програми по готовій блок-схемі:

$$t = \frac{Q}{(20..25)*k}; \quad (3.5)$$

$$t = \frac{2269.47}{21*1,2} = 90,05, \text{ людино-годин;}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t = \frac{Q}{(4..5)*k}; \quad (3.6)$$

$$t = \frac{2269.47}{5*1,2} = 378.24, \text{ людино-годин,}$$

– за умови комплексного налагодження завдання;

$$t^k_{отл} = 1,4*t_{отл}; \quad (3.7)$$

$$t^k_{отл} = 1,4*378.24 = 529.54, \text{ людино годин.}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \quad (3.8)$$

де $t_д$ – трудомісткість підготовки матеріалів і рукопису:

$$t = \frac{Q}{(15..20)*k}; \quad (3.9)$$

$$t = \frac{2269.47}{20*1.2} = 94.56, \text{ людино-годин}$$

де $t_{до}$ – трудомісткість редагування, печатки й оформлення документів:

$$t_{до} = 0,75* t_{др}; \quad (3.10)$$

$$t_{до} = 0,75 * 94.56 = 70.92, \text{ людино-годин.}$$

$$t_д = 94.56 + 70.92 = 165.48, \text{ людино-годин.}$$

Розрахуемо трудомісткість розробки додатку:

$t_д = 60 + 33.09 + 94.56 + 90,05 + 378.24 + 165.48 = 821.42$, людино-годин.

У результаті ми розрахували, що необхідно 821.42 людино-годин для розробки даного програмного засобу.

3.2 Розрахунок витрат на створення програми

Витрати на розробку додатку $К_{ПО}$ включають витрати на заробітну платню програміста $З_{ЗП}$ і витрат машинного часу, необхідного для налагодження засобу на ЕОМ.

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.} \quad (3.11)$$

де $З_{ЗП}$ – заробітна платня програмістів, яка рахується за формулою:

$$З_{ЗП} = t*С_{ПР}, \text{ грн.} \quad (3.12)$$

де t – загальна трудомісткість людино-годин,

$С_{ПР}$ – середня годинна заробітня плата програмісту, грн/година. Середня заробітня плата розробника Java становить 70 грн/год.

$$З_{ЗП} = 70 * 821.42 = 57516, \text{ грн.}$$

$З_{МВ}$ - вартість машинного часу, необхідно для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} * С_{МЧ}, \text{ грн.} \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$С_{МЧ}$ - вартість машино-години ЕОМ, грн/год., дорівнює 4,2.

$$З_{МВ} = 378.24 * 7 = 2647, \text{ грн.}$$

$$К_{ПО} = 57516 + 2647 = 60163 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_к * F_р}, \text{ мес,} \quad (3.14)$$

де $B_к$ – число виконавців,

$F_р$ – місячний фонд робочого часу (при 40 годинному робочому тижні $F_р$ = 176 годин).

$$T = \frac{821.42}{1*176} = 4{,}66 \text{ міс.}$$

Висновок: час розробки даного програмного забезпечення складає 821.42 людино-годин. Таким чином, очікувана тривалість розробки складе 4.66 місяця при 40 годинному робочому тижні (місячний фонд робочого часу 176 годин), а витрати на створення програмного забезпечення складатимуть 60163 грн.

ВИСНОВКИ

Метою цієї кваліфікаційної роботи є розробка мобільного застосунку гаманця для обліку фінансів користувача. Представлена розробка надасть користувачеві можливість вести облік персональних фінансів у додатку, планувати витрати на майбутне, розподіляти бюджет.

Для реалізації поставлених цілей та досягнення потрібного функціоналу додатку, було використано сучасне програмне забезпечення та мову програмування.

Проаналізувавши ринок подібних мобільних застосунків я прийшов до висновку, що вони частіше за все занадто важкі для розуміння і розібратися в інтерфейсі з кількома десятками різних кнопок та не зрозумілих функцій, може бути доволі важко. Тому було прийняте рішення створити концепт програмного додатку який дозволить вести облік фінансів на базовому рівні без зайвих функцій.

Процесс розробки програмної частини додатку виконувався у середовищі розробки під мобільну операційну систему Android під назвою - Android studio. Перший концепт вікон програми на рівні схеми був створений у додатку для графічному редакторі інтерфейсів під назвою Figma.

У ході роботи було протестовано всі необхідні від додатку функції та не було виявлено проблем у їх функціонуванні.

Враховуючи все вище зазначене можна зробити висновок, що у ході розробки було досягнено усіх поставлених цілей та виконано всі вимоги до програмного застосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gayele Laakmann MCdowell – Cracking the coding interview 189 programming questions & solutions 2016 – 441 с.
2. Малий бізнес - wikipedia.org
3. ISO стандарти – iso.org
4. Android Sturio допоміжник – developer.android.com
5. Java інформація – wikipedia.org
6. Android інформація– wikipedia.org
7. Java бази даних допоміжник – metanit.com
8. Java класи допоміжник – examclouds.com
9. Кен Арнольд, Джеймс Гослинг. «Мова програмування» Java 1997 – 192с.
10. Роберт Матрін «Ідеальний програміст. Як стати професіоналом у розробці ПЗ» 2011 – 312с.
11. Борис Бейзер «тестування чорної скрині» 2004 – 98с.
12. Робін Вільямс «Книга для недизайнерів» 2016 – 44с.
13. Рідна Мова андроіду [Електронний ресурс]: toster.ru – Режим доступу: https://qna.habr.com/q/8860
14. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” / Укладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 21с.
15. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 122 «Комп’ютерні науки» галузі знань 12 Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2019. – 52 с.
16. Розподіл ринку смартфонів – ixbt.com
17. Panigrahy N. Xamarin «розробка мобільних додатків під андроід» 2015. – 367с.
18. Налаштування Android SDK – metanit.com

19. AVD manager – android.cn
20. Етапи створення додатку – sibdev.pro
21. Версії ОС Android – wikipedia.org
22. Ян Дарвін «задачі та рішення для розробників додатків» 2018 – 401с.

ДОДАТОК А

КОД ПРОГРАМИ

```
package napier.example.diplomSaulyakYan;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.DialogFragment;

import android.app.DatePickerDialog;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import java.text.DateFormat;
import java.util.Calendar;

public class AddTransaction extends AppCompatActivity implements View.OnClickListener,
DatePickerDialog.OnDateSetListener {

    private DataBaseTransac dm;
    private EditText editText1, editText3, editText4;
    private Spinner editText2;
    private RadioButton rb0,rb1;
    private Spinner spinner;
    private String id_account;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_transaction);

        View saveButton = findViewById(R.id.add_btnSave);
        saveButton.setOnClickListener(this);
        View back = findViewById(R.id.add_btnBack);
        back.setOnClickListener(this);

        Button date_pick = (Button) findViewById(R.id.date_picker);
        date_pick.setOnClickListener(this);

        spinner = findViewById(R.id.add_type);
        ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(AddTransaction.this,
R.array.type_pick, android.R.layout.simple_spinner_item);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spinner.setAdapter(adapter);

        editText1 = (EditText) findViewById(R.id.add_amount);
        editText2 = (Spinner) findViewById(R.id.add_type);
        editText3 = (EditText) findViewById(R.id.add_date);
        editText4 = (EditText) findViewById(R.id.add_commentary);

        rb0 = (RadioButton) findViewById(R.id.radioSpent);
        rb0.setChecked(true);
```

```
        rb1 = (RadioButton) findViewById(R.id.radioIncome);

        rb0.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {

            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
                // TODO Auto-generated method stub
                if (isChecked) {
                    ArrayAdapter<CharSequence> adapter1 = ArrayAdapter.createFromResource(AddTransaction.this,
R.array.type_pick, android.R.layout.simple_spinner_item);
                    adapter1.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
                    spinner.setAdapter(adapter1);
                }
            }
        });

        rb1.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
                // TODO Auto-generated method stub
                if (isChecked) {
                    ArrayAdapter<CharSequence> adapter2 = ArrayAdapter.createFromResource(AddTransaction.this,
R.array.type_pick2, android.R.layout.simple_spinner_item);
                    adapter2.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
                    spinner.setAdapter(adapter2);
                }
            }
        });

        id_account = getIntent().getStringExtra("id_account");
    }

    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.add_btnBack:
                this.finish();
                break;
            case R.id.add_btnSave:
                String amount = ((EditText) findViewById(R.id.add_amount)).getText().toString();
                String type = ((Spinner) findViewById(R.id.add_type)).getSelectedItem().toString();
                String date = ((EditText) findViewById(R.id.add_date)).getText().toString();
                String commentary = ((EditText) findViewById(R.id.add_commentary)).getText().toString();
                if(amount.equals("") || amount.charAt(0)=='-'){
                    Toast.makeText(this, "Введіть кількість", Toast.LENGTH_SHORT).show();
                }
                else if(type.equals("") || type.equals("--Choose Type--")){
                    Toast.makeText(this, "Оберіть категорію транзакції", Toast.LENGTH_SHORT).show();
                }
                else if(date.equals("")){
                    Toast.makeText(this, "Зазначте дату", Toast.LENGTH_SHORT).show();
                }
                else{
                    if(commentary==null)
                        commentary = "";
                    this.dm = new DataBaseTransac(this);
                    long identification;
                    String stringToReturn;
                    if (rb0.isChecked()){
                        identification = this.dm.insert(amount, type, date, "0", id_account, commentary);
                        stringToReturn = amount+" - "+type+" - "+date+" - "+identification+" - 0 - "+id_account+" -
"+commentary;
                    }
                    else{
                        identification = this.dm.insert(amount, type, date, "1", id_account, commentary);
```

```
                stringToReturn = amount+" - "+type+" - "+date+" - "+identification+" - 1 - "+id_account+" -
"+commentary;
            }
            Intent returnIntent = new Intent();
            returnIntent.putExtra("String", stringToReturn);
            setResult(1, returnIntent);

            editText1.getText().clear();
            editText3.getText().clear();
            editText4.getText().clear();
            editText2.setSelection(0);
            AddTransaction.this.finish();
          }
          break;
        case R.id.date_picker:
          DialogFragment datePicker = new DataPickerFragment();
          datePicker.show(getSupportFragmentManager(), "date picker");
          break;
      }
    }


    @Override
    public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {
      Calendar calendar = Calendar.getInstance();
      calendar.set(Calendar.YEAR, year);
      calendar.set(Calendar.MONTH, month);
      calendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);
      String currentDateString = DateFormat.getDateInstance(DateFormat.SHORT).format(calendar.getTime());
      TextView textView = (TextView)findViewById(R.id.add_date);
      textView.setText(currentDateString);
    }
}


package napier.example.diplomSaulyakYan;

import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;

import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.os.Parcelable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.List;

public class HomeFragment extends Fragment implements View.OnClickListener, OnListenerT{
    private String issue;
    private RecyclerView attractionsListView;
    private ArrayList<Transaction> transactions;
    private TransactionAdapter adapter;
    private TextView txtV;
```

```
    private DataBaseTransac dm;
    private List<String[]> names2 = null;
    private ArrayList<String> stg1 = new ArrayList<>();
    private Double total;
    private Button button;

    public HomeFragment(String issue) {
        this.issue = issue;
        // Required empty public constructor
    }

    @Override
    public View onCreateView(LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_home, container, false);
        attractionsListView = view.findViewById(R.id.attractions_list);
        txtV = view.findViewById(R.id.total);
        button = (Button)view.findViewById(R.id.button);
        button.setOnClickListener(this);
        return view;
    }

    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        total = 0.0;
        dm = new DataBaseTransac(getActivity());
        names2 = dm.selectAll();
        String stg;

        transactions = new ArrayList<>();
        attractionsListView.setHasFixedSize(false);

        for (String[] name : names2) {
            if(name[5].equals(issue)){
                stg = name[0] + " - "
                        + name[1] + " - "
                        + name[2] + " - "
                        + name[3] + " - "
                        + name[4] + " - "
                        + name[5] + " - "
                        + name[6];
                stg1.add(stg);
                Transaction transac;
                switch(name[2]){
                    case "Transport":
                        transac = new Transaction(name[1],name[2],name[3],name[6],R.drawable.ic_action_transport,name[0],
name[4], name[5]);
                        break;
                    case "Family":
                        transac = new Transaction(name[1],name[2],name[3],name[6],R.drawable.ic_action_family,name[0],
name[4], name[5]);
                        break;
                    case "Sport":
                        transac = new Transaction(name[1],name[2],name[3],name[6],R.drawable.ic_action_sport,name[0],
name[4], name[5]);
                        break;
                    case "Gift":
                        transac = new Transaction(name[1],name[2],name[3],name[6],R.drawable.ic_action_gift,name[0],
name[4], name[5]);
                        break;
                    case "Restaurant":
                        transac = new Transaction(name[1],name[2],name[3],name[6],R.drawable.ic_action_restaurant,name[0],
```

```
name[4], name[5]);
                    break;
                case "Shopping":
                    transac = new Transaction(name[1],name[2],name[3],name[6],R.drawable.ic_action_shopping,name[0],
name[4], name[5]);
                    break;
                case "Education":
                    transac = new Transaction(name[1],name[2],name[3],name[6],R.drawable.ic_action_education,name[0],
name[4], name[5]);
                    break;
                case "Housing":
                    transac = new Transaction(name[1],name[2],name[3],name[6],R.drawable.ic_action_housing,name[0],
name[4], name[5]);
                    break;
                case "Health":
                    transac = new Transaction(name[1],name[2],name[3],name[6],R.drawable.ic_action_health,name[0],
name[4], name[5]);
                    break;
                case "Other":
                    transac = new Transaction(name[1],name[2],name[3],name[6],R.drawable.ic_action_other,name[0],
name[4], name[5]);

                case "Business":
                    transac = new Transaction(name[1],name[2],name[3],name[6],R.drawable.ic_action_business,name[0],
name[4], name[5]);
                    break;
                case "Pension":
                    transac = new Transaction(name[1],name[2],name[3],name[6],R.drawable.ic_action_pension,name[0],
name[4], name[5]);
                    break;
                case "Dividend":
                    transac = new Transaction(name[1],name[2],name[3],name[6],R.drawable.ic_action_dividend,name[0],
name[4], name[5]);
                    break;
                case "Repayment":
                    transac = new
Transaction(name[1],name[2],name[3],name[6],R.drawable.ic_action_repayment,name[0], name[4], name[5]);
                    break;
                case "Salary":
                    transac = new Transaction(name[1],name[2],name[3],name[6],R.drawable.ic_action_salary,name[0],
name[4], name[5]);
                    break;
                case "Saving money transfer":
                    transac = new Transaction(name[1],name[2],name[3],name[6],R.drawable.ic_action_saving,name[0],
name[4], name[5]);
                    break;
                default:
                    transac = new Transaction(name[1],name[2],name[3],name[6],R.drawable.ic_action_other,name[0],
name[4], name[5]);
                    break;
            }
            transactions.add(transac);
            if(name[4].equals("0"))
                total-=Double.parseDouble(name[1]);
            else
                total+=Double.parseDouble(name[1]);
        }
    }
    if(total<0)
        txtV.setTextColor(Color.RED);
    else
        txtV.setTextColor(Color.GREEN);
    adapter = new TransactionAdapter(getActivity(), R.layout.transaction_entry, transactions, this);
```

```
        RecyclerView.LayoutManager layoutManager = new LinearLayoutManager(getActivity());
        attractionsListView.setLayoutManager(layoutManager);
        attractionsListView.setAdapter(adapter);
        txtV.setText("Баланс : "+total+" ₴");
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.button:
                Intent i = new Intent(getActivity(), AddTransaction.class);
                i.putExtra("id_account", issue);
                startActivityForResult(i, 1);
                break;
        }

    }

    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        //Check which request we're responding to - only 1!
        if (requestCode == 1) {
            //Make sure request was successful
            if (resultCode == 1) {
                String returnString = data.getStringExtra("String");
                Toast.makeText(getActivity(), returnString, Toast.LENGTH_LONG).show();
                String[] temp = new String[7];
                int x=0,y=0;
                for(int i=0;i<returnString.length();i++){
                    if(returnString.charAt(i)==' '){
                        temp[x] = returnString.substring(y,i);
                        i = i+2;
                        y = i+1;
                        x++;
                    }
                }
                Transaction transac;
                switch(temp[1]){
                    case "Transport":
                        transac = new Transaction(temp[0],temp[1],temp[2],temp[6],R.drawable.ic_action_transport,temp[3],
temp[4], temp[5]);
                        break;
                    case "Family":
                        transac = new Transaction(temp[0],temp[1],temp[2],temp[6],R.drawable.ic_action_family,temp[3],
temp[4], temp[5]);
                        break;
                    case "Sport":
                        transac = new Transaction(temp[0],temp[1],temp[2],temp[6],R.drawable.ic_action_sport,temp[3],
temp[4], temp[5]);
                        break;
                    case "Gift":
                        transac = new Transaction(temp[0],temp[1],temp[2],temp[6],R.drawable.ic_action_gift,temp[3],
temp[4], temp[5]);
                        break;
                    case "Restaurant":
                        transac = new Transaction(temp[0],temp[1],temp[2],temp[6],R.drawable.ic_action_restaurant,temp[3],
temp[4], temp[5]);
                        break;
                    case "Shopping":
                        transac = new Transaction(temp[0],temp[1],temp[2],temp[6],R.drawable.ic_action_shopping,temp[3],
temp[4], temp[5]);
                        break;
                    case "Education":
                        transac = new Transaction(temp[0],temp[1],temp[2],temp[6],R.drawable.ic_action_education,temp[3],
```

```
temp[4], temp[5]);
                    break;
                case "Housing":
                    transac = new Transaction(temp[0],temp[1],temp[2],temp[6],R.drawable.ic_action_housing,temp[3],
temp[4], temp[5]);
                    break;
                case "Health":
                    transac = new Transaction(temp[0],temp[1],temp[2],temp[6],R.drawable.ic_action_health,temp[3],
temp[4], temp[5]);
                    break;
                case "Other":
                    transac = new Transaction(temp[0],temp[1],temp[2],temp[6],R.drawable.ic_action_other,temp[3],
temp[4], temp[5]);
                    break;
                case "Business":
                    transac = new Transaction(temp[0],temp[1],temp[2],temp[6],R.drawable.ic_action_business,temp[3],
temp[4], temp[5]);
                    break;
                case "Pension":
                    transac = new Transaction(temp[0],temp[1],temp[2],temp[6],R.drawable.ic_action_pension,temp[3],
temp[4], temp[5]);
                    break;
                case "Dividend":
                    transac = new Transaction(temp[0],temp[1],temp[2],temp[6],R.drawable.ic_action_dividend,temp[3],
temp[4], temp[5]);
                    break;
                case "Repayment":
                    transac = new Transaction(temp[0],temp[1],temp[2],temp[6],R.drawable.ic_action_repayment,temp[3],
temp[4], temp[5]);
                    break;
                case "Salary":
                    transac = new Transaction(temp[0],temp[1],temp[2],temp[6],R.drawable.ic_action_salary,temp[3],
temp[4], temp[5]);
                    break;
                case "Saving money transfer":
                    transac = new Transaction(temp[0],temp[1],temp[2],temp[6],R.drawable.ic_action_saving,temp[3],
temp[4], temp[5]);
                    break;
                default:
                    transac = new Transaction(temp[0],temp[1],temp[2],temp[6],R.drawable.ic_action_other,temp[3],
temp[4], temp[5]);
                    break;
            }
            if(temp[4].equals("0"))
                total-=Double.parseDouble(temp[0]);
            else
                total+=Double.parseDouble(temp[0]);
            txtV.setText("Amount : "+total+" ₹");
            if(total<0)
                txtV.setTextColor(Color.RED);
            else
                txtV.setTextColor(Color.GREEN);
            transactions.add(transac);
            adapter.notifyItemInserted(transactions.size()-1);
            adapter.notifyDataSetChanged();
        }
        else if(resultCode==3){
            Transaction returnTransaction = (Transaction) data.getExtras().getSerializable("Transaction");
            for(int i=0;i<transactions.size();i++){
                if(transactions.get(i).getId().equals(returnTransaction.getId())){
                    if(returnTransaction.getIdt().equals("0")){
                        total+=Double.parseDouble(transactions.get(i).getAmount());
                        total-=Double.parseDouble(returnTransaction.getAmount());
```

```
                }
                else{
                    total-=Double.parseDouble(transactions.get(i).getAmount());
                    total+=Double.parseDouble(returnTransaction.getAmount());
                }
                transactions.set(i,returnTransaction);
                adapter.notifyItemChanged(i);
                adapter.notifyDataSetChanged();
                txtV.setText("Кількість : "+total+" ₴");
                if(total<0)
                    txtV.setTextColor(Color.RED);
                else
                    txtV.setTextColor(Color.GREEN);
            }
        }
    }
    else if(resultCode==2){
        String returnString = data.getStringExtra("String");
        Toast.makeText(getActivity(), returnString, Toast.LENGTH_LONG).show();
        for(int i=0;i<transactions.size();i++){
            if(transactions.get(i).getId().equals(returnString)){
                if(transactions.get(i).getIdt().equals("0")){
                    total+=Double.parseDouble(transactions.get(i).getAmount());
                }
                else
                    total-=Double.parseDouble(transactions.get(i).getAmount());
                transactions.remove(i);
                attractionsListView.removeViewAt(i);
                adapter.notifyItemRemoved(i);
                adapter.notifyItemRangeChanged(i,transactions.size());
                adapter.notifyDataSetChanged();
                txtV.setText("Кількість : "+total+" ₴");
                if(total<0)
                    txtV.setTextColor(Color.RED);
                else
                    txtV.setTextColor(Color.GREEN);
            }
        }
    }
  }

  @Override
  public void onClickT(int position) {
    Intent intent = new Intent(getActivity(), TransactionDetails.class);
    intent.putExtra("transaction", (Parcelable) transactions.get(position));
    startActivityForResult(intent,1);
  }
}


package napier.example.diplomSaulyakYan;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteStatement;

import java.util.ArrayList;
import java.util.List;
```

```
public class DataBaseAccount {
    private static final String DATABASE_NAME = "myDatabaseAccountTest.db";
    private static int DATABASE_VERSION = 1;
    static final String TABLE_NAME = "newtable";
    private static Context context;
    static SQLiteDatabase db;
    private SQLiteStatement insertStmt;

    private static final String INSERT = "insert into " + TABLE_NAME
            + " (id,name,id_client) values (?,?,?)";

    public DataBaseAccount(Context context) {
        DataBaseAccount.context = context;
        DataBaseAccount.OpenHelper openHelper = new DataBaseAccount.OpenHelper(this.context);
        DataBaseAccount.db = openHelper.getWritableDatabase();
        this.insertStmt = DataBaseAccount.db.compileStatement(INSERT);
    }

    public long insert(String id, String name, String id_client) {
        this.insertStmt.bindString(1,id);
        this.insertStmt.bindString(2, name);
        this.insertStmt.bindString(3, id_client);
        return this.insertStmt.executeInsert();
    }

    public void updateData(String id, String name,String id_client){
        ContentValues contentValues = new ContentValues();
        contentValues.put("id",id);
        contentValues.put("name",name);
        contentValues.put("id_client",id_client);
        db.update(TABLE_NAME,contentValues, "id = ?",new String[]{id});
    }

    public void deleteAll() {
        db.delete(TABLE_NAME, null, null);
    }

    public void deleteData(String id){
        int i = Integer.parseInt(id);
        db.delete(TABLE_NAME,"id" + " = " + i, null);
        db.close();
    }

    public List<String[]> selectAll() {
        List<String[]> list = new ArrayList<String[]>();
        Cursor cursor = db.query(TABLE_NAME, new String[]{"id", "name", "id_client"}, null, null, null, null, "id asc");
        int x = 0;
        if (cursor.moveToFirst()) {
            do {
                String[] b1 = new String[]{cursor.getString(0),
                        cursor.getString(1), cursor.getString(2)};
                list.add(b1);
                x++;
            } while (cursor.moveToNext());
        }
        if (cursor != null && !cursor.isClosed()) {
            cursor.close();
        }
        cursor.close();
        return list;
    }

    private static class OpenHelper extends SQLiteOpenHelper {
```

```
        OpenHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }

        public void onCreate(SQLiteDatabase db) {
            db.execSQL("CREATE TABLE "
                    + TABLE_NAME
                    + " (id TEXT PRIMARY KEY, name TEXT, id_client TEXT)");

            ContentValues values = new ContentValues();
            values.put("id","0");
            values.put("name","Account 1");
            values.put("id_client","clem");
            db.insert(TABLE_NAME,null,values);

            values = new ContentValues();
            values.put("id","1");
            values.put("name","Account 2");
            values.put("id_client","clem");
            db.insert(TABLE_NAME,null,values);
        }

        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
            DATABASE_VERSION = newVersion;
            db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
            onCreate(db);
        }
    }
}


package napier.example.diplomSaulyakYan;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteStatement;

import java.util.ArrayList;
import java.util.List;

public class DataBaseTransac {
    private static final String DATABASE_NAME = "myDatabaseProjectTransaction.db";
    private static int DATABASE_VERSION = 1;
    static final String TABLE_NAME = "newtable";
    private static Context context;
    static SQLiteDatabase db;
    private SQLiteStatement insertStmt;

    private static final String INSERT = "insert into " + TABLE_NAME
            + " (amount,type,date,idt,id_account,commentary) values (?,?,?,?,?,?)";

    public DataBaseTransac(Context context) {
        DataBaseTransac.context = context;
        OpenHelper openHelper = new OpenHelper(this.context);
        DataBaseTransac.db = openHelper.getWritableDatabase();
        this.insertStmt = DataBaseTransac.db.compileStatement(INSERT);
    }

    public long insert(String amount, String type, String date, String idt ,String id_account, String commentary) {
        this.insertStmt.bindString(1, amount);
```

```
        this.insertStmt.bindString(2, type);
        this.insertStmt.bindString(3, date);
        this.insertStmt.bindString(4, idt);
        this.insertStmt.bindString(5, id_account);
        this.insertStmt.bindString(6, commentary);
        return this.insertStmt.executeInsert();
    }

    public void updateData(String id,String amount, String type, String date, String idt, String id_account, String
commentary){
        ContentValues contentValues = new ContentValues();
        contentValues.put("amount",amount);
        contentValues.put("type",type);
        contentValues.put("date",date);
        contentValues.put("idt",idt);
        contentValues.put("id_account",id_account);
        contentValues.put("commentary",commentary);
        db.update(TABLE_NAME,contentValues, "id = ?",new String[]{id});
    }

    public void deleteAll() {
        db.delete(TABLE_NAME, null, null);
    }

    public void deleteData(String id){
        int i = Integer.parseInt(id);
        db.delete(TABLE_NAME,"id" + " = " + i, null);
        db.close();
    }

    public List<String[]> selectAll() {
        List<String[]> list = new ArrayList<String[]>();
        Cursor cursor = db.query(TABLE_NAME, new String[]{"id", "amount", "type", "date", "idt", "id_account",
"commentary"}, null, null, null, null, "date asc");
        int x = 0;
        if (cursor.moveToFirst()) {
            do {
                String[] b1 = new String[]{cursor.getString(0),
                        cursor.getString(1), cursor.getString(2),
                        cursor.getString(3), cursor.getString(4), cursor.getString(5),
                        cursor.getString(6)};
                list.add(b1);
                x++;
            } while (cursor.moveToNext());
        }
        if (cursor != null && !cursor.isClosed()) {
            cursor.close();
        }
        cursor.close();
        return list;
    }

    private static class OpenHelper extends SQLiteOpenHelper {
        OpenHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }

        public void onCreate(SQLiteDatabase db) {
            db.execSQL("CREATE TABLE "
                    + TABLE_NAME
                    + " (id INTEGER PRIMARY KEY, amount TEXT, type TEXT, date TEXT, idt TEXT, id_account TEXT,
commentary TEXT)");
        }
```

```
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
            DATABASE_VERSION = newVersion;
            db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
            onCreate(db);
        }
    }
}


package napier.example.diplomSaulyakYan;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.DialogFragment;

import android.app.DatePickerDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.os.Parcelable;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import java.text.DateFormat;
import java.util.Calendar;

public class TransactionDetails extends AppCompatActivity implements View.OnClickListener,
DatePickerDialog.OnDateSetListener, AdapterView.OnItemSelectedListener {

    private DataBaseTransac dataBaseManipulator;
    private EditText editText1, editText3, editText4;
    private Spinner editText2;
    Transaction transaction;

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

        transaction = (Transaction) getIntent().getParcelableExtra("transaction");

        if (transaction != null) {
            View saveButton = findViewById(R.id.save_btn);
            saveButton.setOnClickListener(this);
            View back = findViewById(R.id.back_btn);
            back.setOnClickListener(this);
            View delete = findViewById(R.id.delete_btn);
            delete.setOnClickListener(this);

            ImageView imageView = (ImageView)findViewById(R.id.type_image);
            imageView.setImageDrawable(getResources().getDrawable(transaction.getIconId()));

            Button date_pick = (Button) findViewById(R.id.modify_picker);
```

```
        date_pick.setOnClickListener(this);

        Spinner spinner = findViewById(R.id.modify_type);
        ArrayAdapter<CharSequence> adapter;
        if(transaction.getIdt().equals("0"))
            adapter = ArrayAdapter.createFromResource(this, R.array.type_pick, android.R.layout.simple_spinner_item);
        else
            adapter = ArrayAdapter.createFromResource(this, R.array.type_pick2,
android.R.layout.simple_spinner_item);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(this);

        editText1 = (EditText) findViewById(R.id.modify_amount);
        editText2 = (Spinner) findViewById(R.id.modify_type);
        editText3 = (EditText) findViewById(R.id.modify_date);
        editText4 = (EditText) findViewById(R.id.modify_commentary);

        editText1.setText(transaction.getAmount());
        editText3.setText(transaction.getDate());
        editText4.setText(transaction.getCommentary());
        if(transaction.getIdt().equals("0"))
            switch(transaction.getType()){
                case "Transport":
                    editText2.setSelection(1);
                    break;
                case "Family":
                    editText2.setSelection(2);
                    break;
                case "Sport":
                    editText2.setSelection(3);
                    break;
                case "Gift":
                    editText2.setSelection(4);
                    break;
                case "Restaurant":
                    editText2.setSelection(5);
                    break;
                case "Shopping":
                    editText2.setSelection(6);
                    break;
                case "Education":
                    editText2.setSelection(7);
                    break;
                case "Housing":
                    editText2.setSelection(8);
                    break;
                case "Health":
                    editText2.setSelection(9);
                    break;
                case "Other":
                    editText2.setSelection(10);
                    break;
                default:
                    editText2.setSelection(10);
                    break;
            }
        else
            switch(transaction.getType()){
                case "Business":
                    editText2.setSelection(1);
                    break;
                case "Pension":
```

```
                editText2.setSelection(2);
                break;
            case "Dividend":
                editText2.setSelection(3);
                break;
            case "Repayment":
                editText2.setSelection(4);
                break;
            case "Salary":
                editText2.setSelection(5);
                break;
            case "Saving money transfer":
                editText2.setSelection(6);
                break;
            case "Other":
                editText2.setSelection(7);
                break;
            default:
                editText2.setSelection(7);
                break;
        }
    }
}

@Override
public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {
    Calendar calendar = Calendar.getInstance();
    calendar.set(Calendar.YEAR, year);
    calendar.set(Calendar.MONTH, month);
    calendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);
    String currentDateString = DateFormat.getDateInstance(DateFormat.SHORT).format(calendar.getTime());
    TextView textView = (TextView)findViewById(R.id.modify_date);
    textView.setText(currentDateString);
}

@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.back_btn:
            TransactionDetails.this.finish();
            break;
        case R.id.modify_picker:
            DialogFragment datePicker = new DataPickerFragment();
            datePicker.show(getSupportFragmentManager(), "date picker");
            break;
        case R.id.save_btn:
            String amount = ((EditText) findViewById(R.id.modify_amount)).getText().toString();
            String type = ((Spinner) findViewById(R.id.modify_type)).getSelectedItem().toString();
            String date = ((EditText) findViewById(R.id.modify_date)).getText().toString();
            String commentary = ((EditText) findViewById(R.id.modify_commentary)).getText().toString();
            if(amount.equals("") || amount.charAt(0)=='-'){
                Toast.makeText(this, "Введіть кількість", Toast.LENGTH_SHORT).show();
            }
            else if(type.equals("") || type.equals("--Choose Type--")){
                Toast.makeText(this, "Please select the type of the transaction", Toast.LENGTH_SHORT).show();
            }
            else if(date.equals("")){
                Toast.makeText(this, "Please pick a date", Toast.LENGTH_SHORT).show();
            }
            else{
                this.dataBaseManipulator = new DataBaseTransac(this);

this.dataBaseManipulator.updateData(transaction.getId(),amount,type,date,transaction.getIdt(),transaction.getId_accoun
```

```
t(),commentary);
                Transaction transac;
                Intent returnIntent = new Intent();
                switch(type){
                    case "Transport":
                        transac = new
Transaction(amount,type,date,commentary,R.drawable.ic_action_transport,transaction.getId(),transaction.getIdt(),
transaction.getId_account());
                        break;
                    case "Family":
                        transac = new
Transaction(amount,type,date,commentary,R.drawable.ic_action_family,transaction.getId(),transaction.getIdt(),
transaction.getId_account());
                        break;
                    case "Sport":
                        transac = new
Transaction(amount,type,date,commentary,R.drawable.ic_action_sport,transaction.getId(),transaction.getIdt(),
transaction.getId_account());
                        break;
                    case "Gift":
                        transac = new
Transaction(amount,type,date,commentary,R.drawable.ic_action_gift,transaction.getId(),transaction.getIdt(),
transaction.getId_account());
                        break;
                    case "Restaurant":
                        transac = new
Transaction(amount,type,date,commentary,R.drawable.ic_action_restaurant,transaction.getId(),transaction.getIdt(),
transaction.getId_account());
                        break;
                    case "Shopping":
                        transac = new
Transaction(amount,type,date,commentary,R.drawable.ic_action_shopping,transaction.getId(),transaction.getIdt(),
transaction.getId_account());
                        break;
                    case "Education":
                        transac = new
Transaction(amount,type,date,commentary,R.drawable.ic_action_education,transaction.getId(),transaction.getIdt(),
transaction.getId_account());
                        break;
                    case "Housing":
                        transac = new
Transaction(amount,type,date,commentary,R.drawable.ic_action_housing,transaction.getId(),transaction.getIdt(),
transaction.getId_account());
                        break;
                    case "Health":
                        transac = new
Transaction(amount,type,date,commentary,R.drawable.ic_action_health,transaction.getId(),transaction.getIdt(),
transaction.getId_account());
                        break;
                    case "Other":
                        transac = new
Transaction(amount,type,date,commentary,R.drawable.ic_action_other,transaction.getId(),transaction.getIdt(),
transaction.getId_account());
                        break;
                    case "Business":
                        transac = new
Transaction(amount,type,date,commentary,R.drawable.ic_action_business,transaction.getId(),transaction.getIdt(),
transaction.getId_account());
                        break;
                    case "Pension":
                        transac = new
Transaction(amount,type,date,commentary,R.drawable.ic_action_pension,transaction.getId(),transaction.getIdt(),
transaction.getId_account());
```

```
                    break;
                case "Dividend":
                    transac = new
Transaction(amount,type,date,commentary,R.drawable.ic_action_dividend,transaction.getId(),transaction.getIdt(),
transaction.getId_account());
                    break;
                case "Repayment":
                    transac = new
Transaction(amount,type,date,commentary,R.drawable.ic_action_repayment,transaction.getId(),transaction.getIdt(),
transaction.getId_account());
                    break;
                case "Salary":
                    transac = new
Transaction(amount,type,date,commentary,R.drawable.ic_action_salary,transaction.getId(),transaction.getIdt(),
transaction.getId_account());
                    break;
                case "Saving money transfer":
                    transac = new
Transaction(amount,type,date,commentary,R.drawable.ic_action_saving,transaction.getId(),transaction.getIdt(),
transaction.getId_account());
                    break;
                default:
                    transac = new
Transaction(amount,type,date,commentary,R.drawable.ic_action_other,transaction.getId(),transaction.getIdt(),
transaction.getId_account());
                    break;
            }
            Toast.makeText(this, transac.getId(), Toast.LENGTH_LONG).show();
            returnIntent.putExtra("Transaction", (Parcelable) transac);
            setResult(3, returnIntent);
            TransactionDetails.this.finish();
        }
        break;
    case R.id.delete_btn:
        showInformationSavedDialog();
        break;
  }
}

@Override
public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {

}

@Override
public void onNothingSelected(AdapterView<?> parent) {

}

protected void showInformationSavedDialog() {
  AlertDialog.Builder builder;
  if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
    builder = new AlertDialog.Builder(TransactionDetails.this, android.R.style.Theme_Material_Dialog_Alert);
  } else {
    builder = new AlertDialog.Builder(TransactionDetails.this);
  }
  builder.setMessage("Ви впевнені, що хочете це видалити ?");
  builder.setCancelable(false);
  builder.setNegativeButton("Так",
      new DialogInterface.OnClickListener() {
        private DataBaseTransac dm;
        public void onClick(DialogInterface dialog, int which){
          this.dm = new DataBaseTransac(getApplicationContext());
```

```
                this.dm.deleteData(transaction.getId());
                Intent returnIntent = new Intent();
                returnIntent.putExtra("String", transaction.getId());
                setResult(2, returnIntent);
                TransactionDetails.this.finish();
            }
        });
    builder.setPositiveButton("Hi",
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which){
                dialog.cancel();
            }
        });
    AlertDialog alert = builder.create();
    alert.show();
  }
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім'я файлу	**Опис**
Пояснювальні документи	
Кваліфікаційна робота Сауляка Яна Максимовича 122-18-2.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Сауляка Яна Максимовича 122-18-2.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Проєкт	
Wallet-app.rar	WinRar архів з усіма внутрішніми файлами додатку
Презентація	
Презентація до кваліфікаційної роботи Сауляка Яна Максимовича 122-18-2.pdf	Презентація кваліфікаційної роботи