Київський національний торговельно-економічний університет

Кафедра програмної інженерії та інформаційних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка мобільного додатку обліку особистих фінансів на ОС Android»

Студента 4 курсу, 7групи, спеціалізація 121 «Інженерія програмна інженерія» _____ Юрчак Павло Вадимович

підпис студента

Науковий керівник кандидат технічних наук, професор _____ Пашорін Валерій Іванович

підпис керівника

Гарант освітної програми, кандидат технічних наук, доцент _____ Цензура Микола Олександрович

підпис гаранта

КИЇВ – 2020

ЗМІСТ

ВСТУП

Для багатьох людей телефон це не тільки зв'язок, а маленький комп'ютер у кишені. Сьогодні понад 2 млрд користувачів користуються Android системою. Підтримуються багато пристроїв на даній платформі. Android відкрита система для розробників ПЗ. На приклад платформи Android studio, Eclipse, Visual studio вони є повністю безкоштовні і кожний за бажанням може створити свою програму. Для порівняння візьмемо IPhone (від розробників Apple) всі їх платформи для розробки ПЗ платні і потребують немаленьких фінансових вкладень. Android містить велику кількість безкоштовних бібліотек для роботи зі сторонніми ресурсами наприклад Google Map API, а в той час для Windows Phone Mobile бібліотека не доступна. ОС Android це не тільки смартфони, але дана ОС використовується в смарт телевізорах, смарт годинниках, в системах розумний дім, планшетних комп'ютерах, т.д. То програма написана наприклад для смартфону технічно може підходити і на інші пристрої зазначених вище.

В даній дипломній роботі буде описано створення програми обліку особистих фінансів на платформу ОС Android, для користувачів всіх вікових категорій. Використання новітніх технологій для розробників під ОС Android буде використовуватися платформа Android studio, бо вона є безкоштовною, з відкритим кодом, зручним інтерфейсом та редактором зі своїм компілятором налаштованим для ОС Android.

Актуальність дипломної роботи є розробка програми для ведення особистих фінансів для ОС Android використанні мови java з редактором на платформі Android studio.

Метою дослідження випускної кваліфікаційної роботи є контролювати витрати та заздалегідь визначити частину з них. Проаналізувати створену програму для ведення особистих фінансів для формування певних висновків.

Предмет дослідження – програма обліку особистих фінансів на ОС Android.

Завдання дослідження:

1. Проаналізувати існуючи програми.
2. Розробка архітектури програми ведення особистих фінансів.
3. Створення програми ведення особистих фінансів.

Об'єкт дослідження– управління особистими фінансами.

Методи дослідження– контролювати витрати та заздалегідь визначити частину з них. Користуючись програмою ведення особистих фінансів з`явиться достатній досвід автоматичної економії.

РОЗДІЛ 1.

ІДЕЯ ТА РОЗГЛЯД ПІДХОДІВ ДО РЕАЛІЗАЦІЇ. ОПИС ОСНОВНИХ ЗАДАЧ ПРОГРАМНОГО ПРОДУКТУ

1.1. Створення ідеї та опис плану реалізації.

Після вивчення платформи андроїд було вирішено створити програму обліку особистих фінансів за допомогою мов java та C++.

Кожній ідеї щоб реалізуватися потрібно зробити план. План це є другий етап для створення програми. Плани бувають різні за обсягом, але для кожної програми вони є індивідуальним. То з чого почати:

1. Вибір мови програмування.
2. Створення списку вимог.
3. Розробка архітектури
    a. опис змінних у функціях,
    b. зв’язки між класами та функціями,
    c. створення таблиць для бази даних,
    d. опис функції.
4. Розробка дизайну.
5. Реалізація.
6. Публікація.

1.2. Вибір мови програмування.

Перша версія програми була створена для комп’ютерів на мові C#. В даному випадку програма буде переписана на операційну систему Андроїд де є основними мовами програмування C++ та JAVA. Отже, вибір був очевидним для вибору мов програмування і це C++ та JAVA.

Також можна зазначити мови SQLite 3 для написання запитів на створення та редагування бази даних, та XML для розмітки інтерфейсу.

Мова C++ дозволяє використовувати програмний додаток швидше ніж на інших мовах бо являє собою модифіковану мову С, яка є нискорівневою мовою програмування. Це дозволить оптимізувати програму та пришвидшити її.

Java є високорівневою мовою програмування, яка використовується для написання програм на операційну систему Андроїд.

SQLite 3 це спеціальна мова для створення БД, яка дозволить без використання сервера розгорнути бд для воду особистих фінансів.

XML мова розмітки яка використовується в системі андроїд для створення інтерфейсу для редагування та створення особистих стилів. **Вимоги до програмного продукту та список основних задач.**

Програма повинна виконувати фінансові розрахунки користувача, і робити таблиці та діаграми для візуалізації витрат, доходів.

І розпочнемо з написання бібліотеки для опрацювання з базою даних. В нашому випадку ми будемо використовувати базу даних SQLite 3 яка є стандартною базою даних для андроїд додатків. Бібліотеку будемо писати на мові JAVA.

По друге потрібно створити бібліотеку для роботи з діаграмами, ця бібліотека дозволить використовувати дані з бд(база даних). Даний етап також буде написаний на мові JAVA.

Останнє це зробити інтерфейс, який об'єднує функції програми. Буде написаний на мові JAVA.

1.2.1.Вимоги до створення класу для роботи з базою даних.

В даний етап буде входити написання вимог до класу та для самої бази даних.

Розпочнемо з написання бази даних і розберемо кожний пункт в таблиці на даному випадку це є відповідальний етап. Бо на цьому буде формуватися скелет програми.

Вимоги до бази даних.

Перша вимога. Створити таблицю доходів - ця таблиця буде зберігати всі записані доходи користувача.

Перший стовпець буде зберігати id доходів. Цей id повинен бути унікальним та мати розмір long long для зберігання великих об'ємів даних.

Другий стовпець. Буде зберігати назву Name доходу. Для повного розуміння своїх доходів. Назва може повторюватися то не повинна бути унікальною. Даний стовпець повинен мати тип Varchar() з розміром 255.

Третій стовпець. Буде зберігати суму Sum доходів в цей стовпець буде записуватися значення суми доходу. Даний стовпець буде мати тип float.

Четвертий стовпець. Буде зберігати дату Date дату отримання доходу. Даний стовпець буде мати тип datetime і мати вигляд DD/mm/yyyy.

До створення таблиці була створена діаграма(Рис. 1.1).

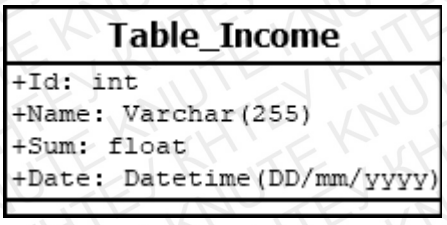

Рис. 1.1. Діаграма таблиці доходів.

Друга таблиця буде зберігати дані про витрати. Дана таблиця не відрізняється від попередньої, але додається ще один стовпець тип Type. В якому буде зберігатися типи витрат, який буде мити id витрати із таблиці типи витрат зі зв'язком первинного ключа.

До створення таблиці була створена діаграма (Рис. 1.2).

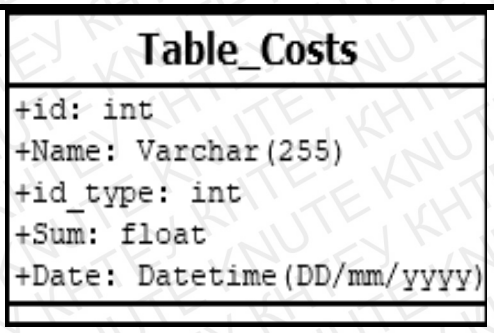

Рис. 1.2. Діаграма таблиці витрат.

Третя таблиця буде зберігати дані про типи витрат. Перший стовпець буде id. Другий стовпець буде назва Name з типом Varchar() і з розміром 50.

До створення таблиці була створена діаграма(Рис. 1.3).

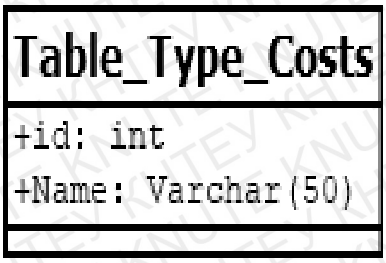

Рис. 1.3. Діаграма таблиці типу витрат.

Зв’язок між таблицями показано в діаграмі(Рис. 1.4).

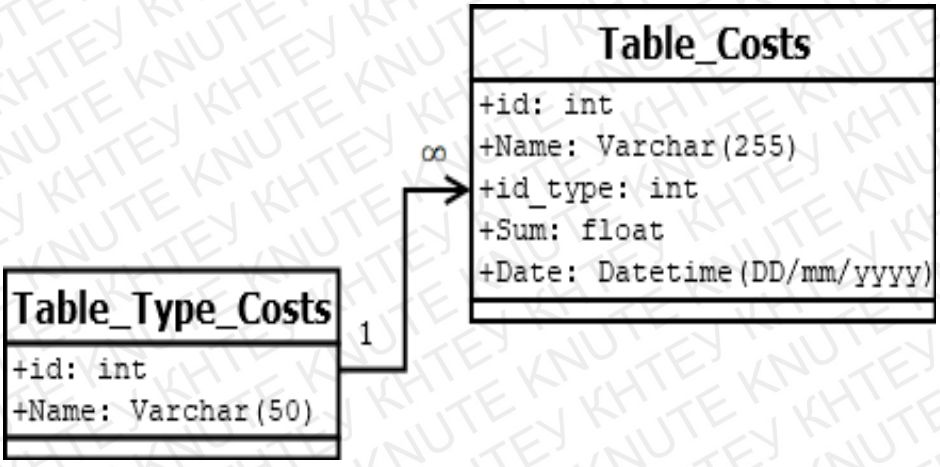

Рис. 1.4. Зв’язок між таблицями витрати та типом витрат.

Четверта таблиця буде нагадуванням в ній буде зберігатися нагадування користувача. Воно така ж як і таблиця витрат(Рис. 1.5).

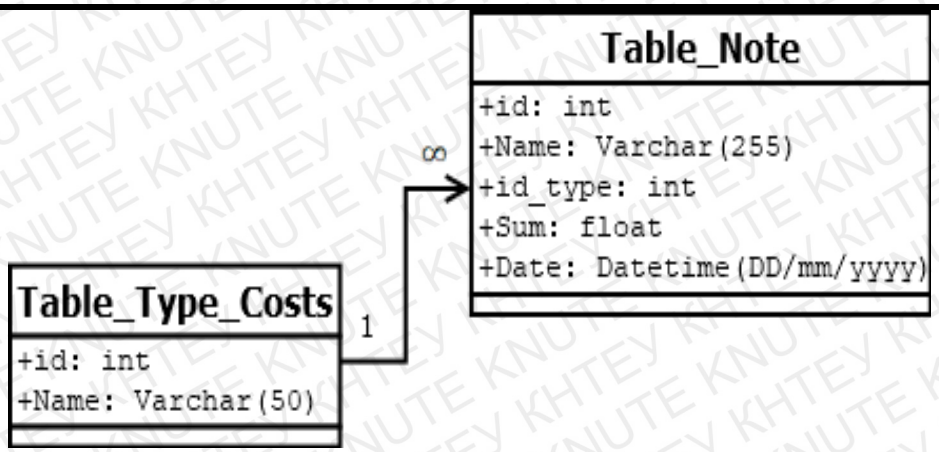

Рис. 1.5. Таблиця нагадувань.

Як ми бачимо на рисунку 1.5 зв'язок з типами витрат такий же як і з нагадуваннями бо нотатки після їх виконання будуть записуватися до таблиці витрат і типи однакові.

П'ята таблиця буде записувати витрату чи нагадування (Рис. 1.6).

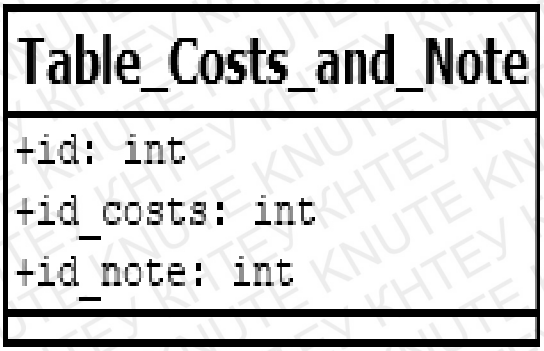

Рис. 1.6. Таблиця витрати чи нагадування.

Шоста таблиця буде зберегти поточний день витрат та доходів (Рис. 1.7).

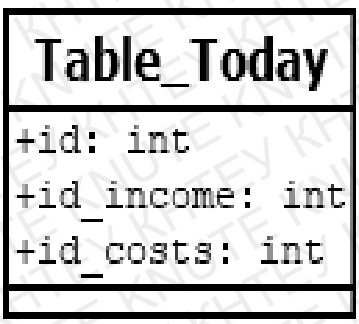

Рис. 1.7. Таблиця записів на сьогодні.

Після створення таблиць була створена діаграма зв'язків(Рис. 1.8).

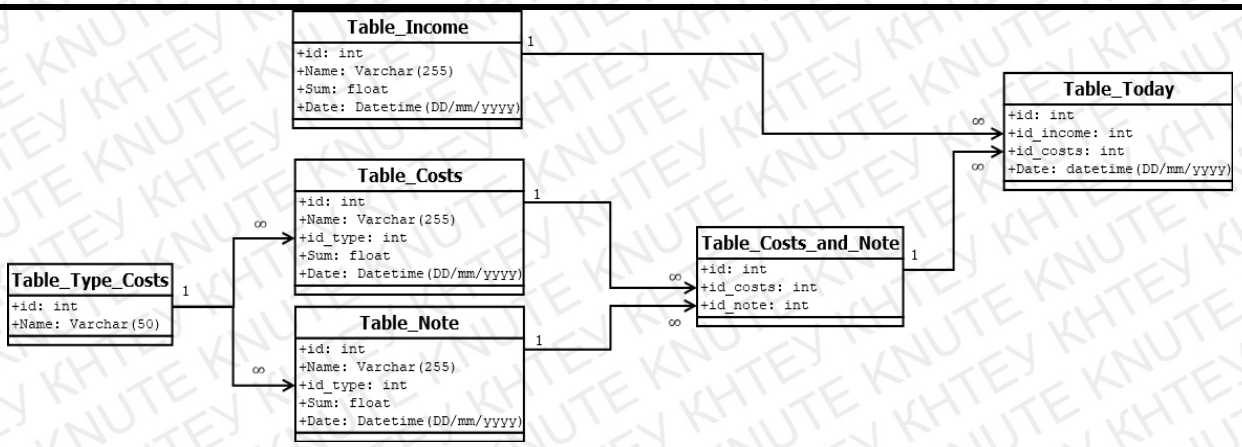

Рис. 1.8. Бд для програми.

Створити бібліотеку для роботи з базою даних SQLite3 та назвати AdapterSQLlibe.

Перша функція, запис до бази даних з використанням перегрузки функції зробити внесення до бази даних таких даних як дохід, витрати та нагадування і мати можливість додавати типи витрат. Вхідні дані в функції назва, сума, дата, тип, назва таблиці.

Перша перегрузка, запис до таблиці дохід матиме вхідні дані: назва, сума, дата та назва таблиці table_income. Дана функція повинна повертати тип bool для підтвердження запису.

Друга перегрузка, запис до таблиці витрати матиме вхідні дані: назва, сума, тип, дата та назва таблиці table_costs. Дана функція повинна повертати тип bool для підтвердження запису.

Третя перегрузка, запис до таблиці нагадування митиме дані: назва, сума, тип, дата та назва таблиці table_note. Дана функція повинна повертати тип bool для підтвердження запису.

Четверта перегрузка, запис до таблиці типу витрат матиме вхідні дані: назва типу витрат та назва таблиці table_type_costs.

Після виконання зробимо діаграму(Рис. 1.9).

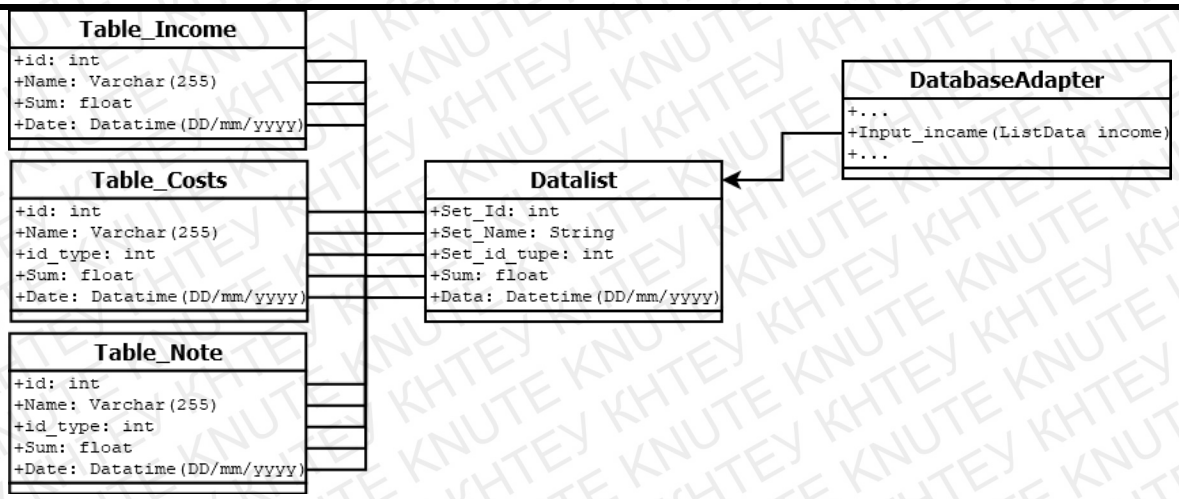

Рис. 1.9. Діаграма взаємодії функції з базою даних.

Друга функція, буде робити видалення із таблиць. Таким чином як описано перша функція але замість функції запису буде використовуватися видалення.

Третя функція, буде робити редагування в таблицях.

Четверта функція, буде робити вивід з таблиці на екран.

1.2.2. Вимоги до створення класу для роботи з формуванням вихідних даних.

Для виводу інформації не в таблиці, а наприклад в діаграми потрібно сформувати дані в певному порядку бо створена діаграма робить статистику вибраного часу. Виводі даних у таблицю можна зробити по останньому записі і до першого для відображення всіх даних, а діаграма може виводити статистику по дням, тижням, місяцям, рокам.

Діаграма це наявний приклад витрат чи доходів на певному проміжку часу. Бо користувачу щоб аналізувати свої витрати чи доходи, потрібно порівнювати з попередніми днями чи місяцями і на цих даних користувач починає розуміти своє положення щодо збільшення чи зменшення свого капіталу.

Почнемо з розробки даної бібліотеки, що повинно там бути:

- Формування даних по заданих періодах.
- Можливість рисування діаграми.

a. Створення діаграми.

b. Вибір кольорів стовпців діаграми.

- Статистичний аналіз.

Перша функція повинна виконувати форматування даних за певний період. Вхідні дані в функцію перший тип початок періоду в форматі Datetime(DD/mm/yyyy) другий аргумент теж в форматі Datetime(DD/mm/yyyy) третій аргумент string в якому буде передаватися вибір по дням, місяцям, рокам. Повертатися буде масив з даними.

З використанням перегрузки функцій також буде варіанти формування даних за період, без вказаних аргументів буде виводитися поточний день і дані за цей місяць і місяці які оточують його.

Третя функція теж перегрузка першої функції яка матиме вхідні дані Datetime(yyyy)тільки рік і вибір по місяцям.

Четверта перегрузка митиме вхідні дані Datetime(mm/yyyy) і вибір по дням.

П’ята перегрузка матиме вхідні дані String дні, місяці, роки та Bool(для того щоб виконати різницю між доходами і витратами). По поточним даним.

Після створення даних функцій приступили до формування самої діаграми і проблема заключається в тому в якій формі буде виводитися дані. Діаграми існують види стовпчиків, кругу, ліній на координатній площині та точками на площині. В даній програмі буде використовуватися кругла діаграма.

Перша функція буде малювати на екрані телефону круг по формулі кругу та по даним які були виведені із бд будуть формуватися вирізи в відсоток від кругу.

Друга функція буде змінювати колір вирізаних деталей.

Статистичний аналіз який буде говорити користувачу що в нього добрий темп чи ні. В даній програмі не буде зроблений повний аналіз по всім місцям чи по рокам бо це є довгий навчальний процес в навчанні нейронної системи,

замість цього буде зроблений індикатор який буде записувати всі доходи та витрати і робити їх різницю, якщо різниця додатна то темп нормальний і його колір відображення буде зелений, якщо різниця покаже в діапазоні від 5% до 15% то це середній темп і його колір жовтий, і якщо менше 5% чи 0 то це є низький темп і його колір червоний. Вхідні дані для функції це дохід double income and double costs на виході повертає різницю.

1.3. Вимоги до класу діаграма.

Вимоги до класу діаграма. Назва даного класу буду CanvasClass. Використання методів рисування на мові java з використанням інструментів android studio графіки будуть вимальовуватися по пункту красним кольором будуть витрати, а зеленим доходи.

Інформація буде братися з бази даних та методом відношення буде вимальовуватися. Діаграма була обрана виді бубликом для компактності. Сортування даних на діаграмі можна по даті. Повинна будуть доступна сортування по даті, місяцю та по окремим дням.

Вимога до створення класу diagramlab створити клас який буде відповідати за:

- Створення діаграми виді Торту;
- витягування даних з бази даних;
- сортування даних по даті;
- перемалювання діаграми.

1.4. Вимоги до створення візуального інтерфейсу

Інтерфейс виконується в простому стилі з візуально примітивною формою для швидкого освоєння користувача.

Головне activity має в собі основні кнопки для вводу інформації для бази даних, всього коштів у користувача та знизу на екрані мобільного телефону меню в якому є кнопки історії, налаштування, головна сторінка.

Activity історія матиме дві кнопки керування вивід у таблиці та діаграма. Табличний вигляд дозволяє клієнту побачити точну інформацію про витрати, доходи та редагування їх. Діаграмний вигляд, візуально відобразить користувачу загальну інформацію.

Activity налаштуванні можна виконувати налаштування мови та грошової одиниці вимірювання.

Activity редагування та вводу дозволяє вносити до бази даних доходи, витрати і можливість їх редагувати чи видаляти з неї.

1.5. Аналіз існуючих програм для ведення особистого та сімейного бюджетів.

За особистими доходами і витратами слідкувати доволі таки просто. Розрахунок бюджету- повноцінна бухгалтерія, в якій треба враховувати всі джерела доходів та особисті інтереси. Існує багато програм, які допоможуть розібратися в сімейних та особистих фінансах. Для ефективного використання програми повинні підходити під різні операційні системи, щоб кожний член сім`ї міг вносити зміни в режимі реального часу.

Програма 1. «Дрібні гроші» - програма будує детальний звіт по загальному бюджету та показує особисті витрати. Кожний член сім`ї може коментувати свої витрати. Програма налаштовує розпізнавання смс від банку та автоматично робить облік в русі коштів. Бухгалтерію можна вести в різних валютах та на різних рахунках. Програма захищена паролем та пін-кодом.

Програма 2. «Дзен-мані»- програма розрахована на багато користувачів, в якій можна одночасно планувати особистий та сімейний бюджет. Вона може імпортувати операції з банківських служб, систем електронних грошей. Вона надає аналіз витрат та враховує рух грошей в різних валютах.

Програма 3. «Коін-кіпер»- керувати фінансами можна за допомогою мобільного чи веб-версії програма дозволяє розібратися з грошовими потоками сім`ї. Можна встановити нагадування про повернення боргів та ліміт на витрати в певний проміжок часу.

Програма 4. «Алзекс-фінанс»- програма дозволяє створити кожному користувачу свою облікову запис. Кожний член сім`ї може вибирати, які рухи коштів зробити доступними у сім, а які ні. Система тегів дозволяє враховувати витрати великим категоріям та дрібним категоріям. В програмі можна слідкувати за боргами та ставити фінансові цілі.

1.6. Висновок до розділу 1.

Отже, в даному розділу було описано технічне завдання у вимогах для створення програмного продукту.

Проводячи аналіз зазначених програм та інших. Можна зробити висновок: самі програми дуже цікаві, але складні у використані. Не зручна панель керування, багато відволікаючий інформації та додаткові налаштування коштують грошей. Програми повинні бути розраховані не тільки для дорослих, а і для дітей, які мають(кишенькові кошти та інше) для того, щоб з дитинства дитина розуміла, як правильно користуватися доходами . Але є багато переваг: поєднання з банківськими рахунками, нагадування та смс повідомлення про рух коштів платіжної карти. Гарна оптимізація програми, яка дає змогу швидко діяти на слабких обчислюваних машин та мобільних телефонів.

Також розписано вимоги до створення бази даних, бібліотеки керування БД, бібліотеки створення діаграми. Розглянута ідея та вирішення проблеми.

Розглянуто вимоги до створення бази даних, сформована фізична та логічна модель, розписані вимоги до типу та розміру даних, зазначені назви таблиць.

Було розглянуто мови програмування такі як java, C++, SQLite 3, XML. Java використовується для написання класі та функцій для роботи з БД та Діаграмою. SQLite 3 переваги якої є розгорнення БД без використання серверу, яка для даного продукт є актуальною. XML допоможе створити більш зручний інтерфейс з унікальним стилем.

Також було розглянуто вимоги до створення класі діаграма та формування даних для вводу та виводу інформації з БД.

В даному розділу була виконана теоретична частина диплому де було описано вимоги до створення програми особистих фінансів та проаналізовано інші програми для підкреслення проблем та уникання їх.

РОЗДІЛ 2.

АРХІТЕКТУРА ТА ОПИС ОСНОВНИХ КЛАСІВ

2.1. Архітектура програмного додатку “Мої гроші”.

Архітектура додатку виконана в UMA діаграмах(Рис. 2.1).

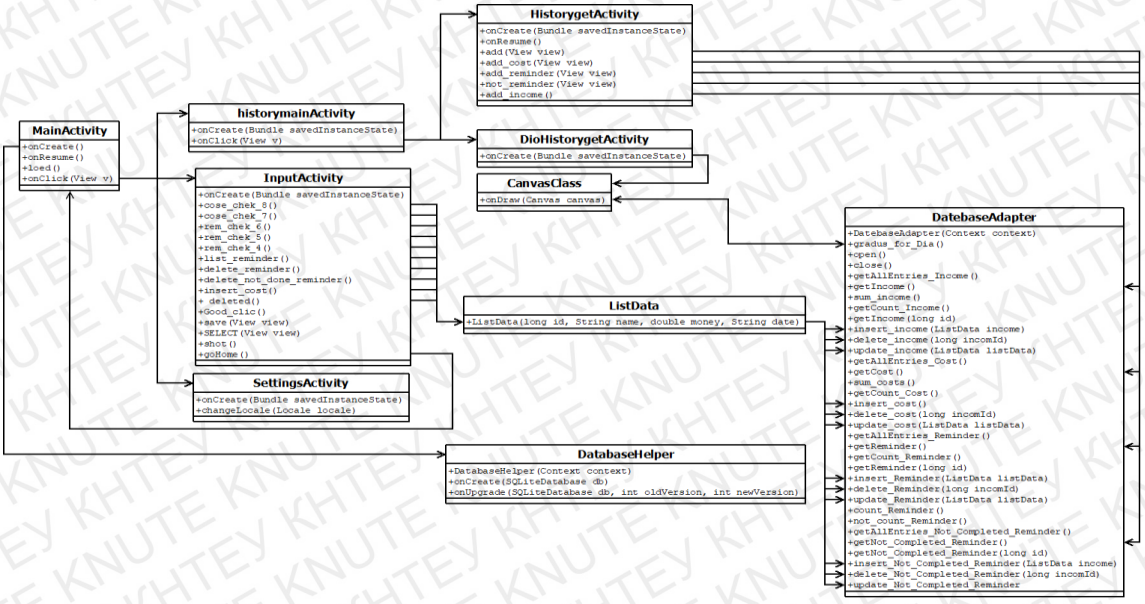

Рис. 2.1. Архітектура програмного додатку “Мої гроші”.

Приклад переходу інформації з activity переноситься до бази даних(Рис. 2.2).

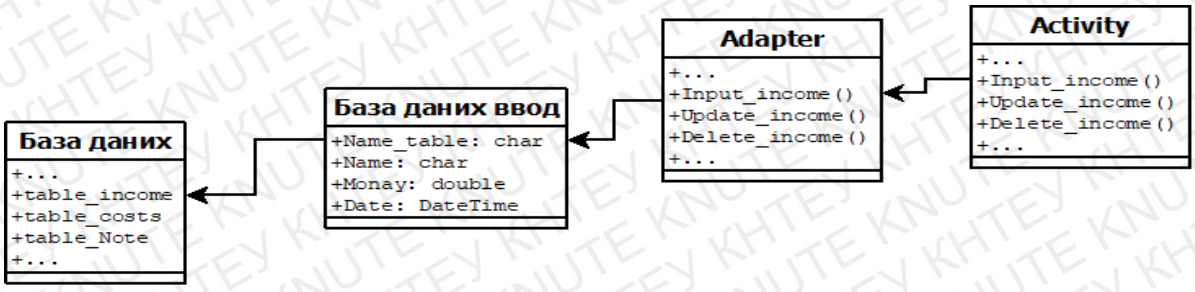

Рис. 2.2. В даній UML діаграмі зображено переміщення даних від користувача до базиданих доходу.

Програмний продукт матиме 6 Activity, 3 класи створені на мові java. Програма була написана з використанням Об'єктно орієнтованим програмуванням.

2.2. Опис класу DatebaseAdapter.

Даний клас створений для управління базою даних. DatebaseAdapter виконує основну роль в додатку(Рис. 2.3). Даний клас має в собі 36 функції з роботою бази даних на вивід та ввід інформації.

Рис. 2.3. DatebaseAdapter в UML діаграмі.

За даною UML діаграмою першим елементом іде конструктор класу DatebaseAdapter(Context context) це спеціальний блок інструкцій, що викликається при створенні об'єкта.

Другим елементом функції gradus_for_Dia() дана функція виводить інформацію у градусах для малювання діаграми. За формулою(2.1).

$$Gradus = \frac{(income1 + income2 + ...) * 360}{(income1 + income2 + ... + costs1 + costs2 + ...)} \quad (2.1).$$

Функція open() відкриває базу даних на запис або на зчитування даних.

Функція close() закриває доступ базам даних.

Функція getAllEnries_Income() отримати всі доходи від публікацій.

Функція getIncome() отримати один із доходів.

Функція Sum_income() потрібна для вирішення формули градуса для діаграми.

Функція getcount_income() отримання кількості записів у таблиці доходів.

Функція get_income(long id) пошук доходу по його id.

Функція insert_income(ListData income) запис до таблиці доходів.

Функція update_income(ListData income) редагування таблиці доходів.

Функція delete_income(ListData income) видалення таблиці доходів.

Функція getAllEnries_costs() отримати всі доходи від публікацій.

Функція getCosts() отримати один із витрат.

Функція Sum_costs() потрібна для вирішення формули градуса для діаграми.

Функція getcount_costs() отримання кількості записів у таблиці витрат.

Функція get_costs(long id) пошук витрати по його id.

Функція insert_costs(ListData listdata) запис до таблиці витрати.

Функція update_costs(ListData listdata) редагування таблиці витрати.

Функція delete_costs(ListData listdata) видалення таблиці витрати.

Функція getAllEnries_Reminder() отримати всі нагадування від публікацій.

Функція get Reminder () отримати один із нагадування.

Функція getcount_ Reminder () отримання кількості записів у таблиці нагадування.

Функція get_ Reminder (long id) пошук нагадування по його id.

Функція insert_ Reminder (ListData listdata) запис до таблиці нагадування.

Функція update_ Reminder (ListData listdata) редагування таблиці нагадування.

Функція delete_ Reminder (ListData listdata) видалення таблиці нагадування.

Функція getAllEnries_Not_completed_Reminder() отримати всі нагадування від публікацій.

Функція getNot_completed_Reminder () отримати один із нагадування.

Функція getcount_Not_completed_Reminder () отримання кількості записів у таблиці невиконані нагадування.

Функція get_Not_completed_Reminder (long id) пошук невиконані нагадування по його id.

Функція insert_Not_completed_Reminder (ListData listdata) запис до таблиці невиконані нагадування.

Функція update_Not_completed_Reminder (ListData listdata) редагування таблиці невиконані нагадування.

Функція delete_Not_completed_Reminder (ListData listdata) видалення таблиці невиконані нагадування.

2.3. Опис класу CanvasClass.

Даний клас створений для побудові діаграми. CanvasClass виконує основну роль в побудові діаграми(Рис. 2.4). Даний клас має одну функцію для побудови діаграми.

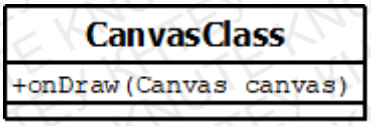

Рис. 2.4. CanvasClass в UML діаграмі.

Створення діаграми за допомогою бібліотеки Canvas яка дозволяє малювати буть які зображення на Activity. Першим кроком для побудови діаграми потрібно вибрати тип. Діаграма виду торта буде економічно для розташовані на маленьких екранах телефону.

2.4. Опис класу DatabaseHelper.

Даний клас створений для побудови бази даних та для перевірки наявності. Даний клас визивається при першому запуску програми для побудови бази даних. Потім іде перевірка на наявність бази даних, якщо перевірка видає хибний результат то даний клас створює нову базу даних(Рис. 2.5).

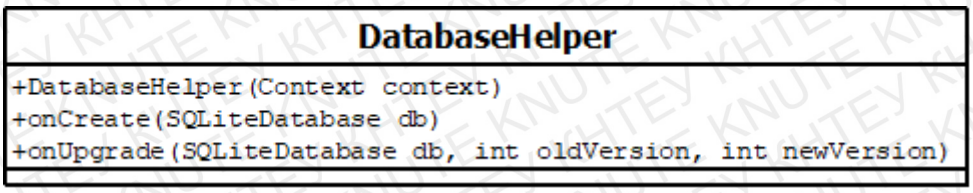

Рис. 2.5. DatabaseHelper в UML діаграмі.

Даний клас складається з конструктора та двох функцій.

Функція OnCreate(SQLiteDatabase db) створює базу даних та таблиці.

Функція onUpgrade(SQLiteDatebase db, int oldVersion, int newVersion) перевіряє наявність бази даних та версію.

2.5. Ознайомлення з Android Studio.

Android Studio - це інтегроване середовище розробки (IDE) для роботи з платформою Android, оголошене 16 травня 2013 року на конференції Google I / O.

IDE був у вільному доступі з версії 0.1, опублікованої в травні 2013 року, а потім перейшов до бета-тестування, починаючи з версії 0.8, яка була випущена в червні 2014 року. Перша стабільна версія 1.0 була випущена в грудні 2014 року, потім підтримка Плагін Android Development Tools (ADT) для Eclipse припинено.

Android Studio, заснований на програмному забезпеченні IntelliJ IDEA від JetBrains, є офіційним інструментом розробки програм Android. Це середовище розробки доступне для Windows, OS X та Linux. 17 травня 2017 року на щорічній конференції Google I / O компанія Google оголосила про підтримку мови Kotlin, що використовується Android Studio як офіційної мови програмування для платформи Android, крім Java та C ++.

Особливості Android Studio:

- Розширений редактор макета: WYSIWYG, можливість роботи з компонентами інтерфейсу за допомогою Drag-and-Drop, функція попереднього перегляду макета в декількох конфігураціях екрана.
- Створення додатків на основі Gradle.
- Різні типи складання та створення декількох файлів .apk
- Код рефакторингу
- Статичний аналізатор коду (Lint), що дозволяє знаходити проблеми продуктивності, несумісність версій тощо.
- Вбудований ProGuard та утиліта для підписання програм.
- Шаблони основних макетів та компонентів Android.
- Підтримка розробки додатків для Android Wear та Android TV.

- Рідна підтримка платформи Google Cloud, яка включає інтеграцію з Google Cloud Messaging та App Engine.
- Android Studio 2.1 підтримує Android N Preview SDK, що означає, що розробники можуть розпочати роботу над створенням програми для нової програмної платформи.
- Нова версія Android Studio 2.1 здатна працювати з оновленим компілятором Jack, а також має покращену підтримку Java 8 та покращену функцію Instant Run.
- Починаючи з платформних інструментів 23.1.0 для Linux, це виключно 64-розрядні.
- В Android Studio 3.0 інструменти Kotlin на основі ID JetBrains IDE будуть включені як стандарт.

2.6. Висновок до розділу 2.

Можна зробити висновки, у другому розділу описано три класи в UML діаграмах та розглянута архітектура проекту.

Було розглянуто три класи які взаємодіють з базою даних для формування даних для вводу та виводу інформації до лістингу чи діаграми виді торта. Можна зазначити у класі DatabaseAdapter є 33 функції яких 28 роботи з БД, 5 внутрішніх функції та конструктор класу. В DatabeseHelper конструктор та 2-ві функції для створення та перевірки наявності БД. CanvasClass створений лише для одної цілі для малювання діаграми з використанням даних БД.

Проаналізовано платформу Android Studio було описано її особливості та підкреслення мов програмування.

Сформовано архітектуру програми для ведення особистих фінансів за допомогою UML діаграм. Було показано всі класи та функції які взаємодіють між собою та передавання у функцію даних та їх тип.

РОЗДІЛ 3.

РОЗРОБКА МОБІЛЬНОЇ ПРОГРАМИ ОБЛІКУ ОСОБИСТИХ ФІНАНСІВ НА ОС ANDROID

3.1. Створення проекту в Android Studio.

Запускаємо Android Studio та вибираємо створити новий android проект (Рис. 3.1).

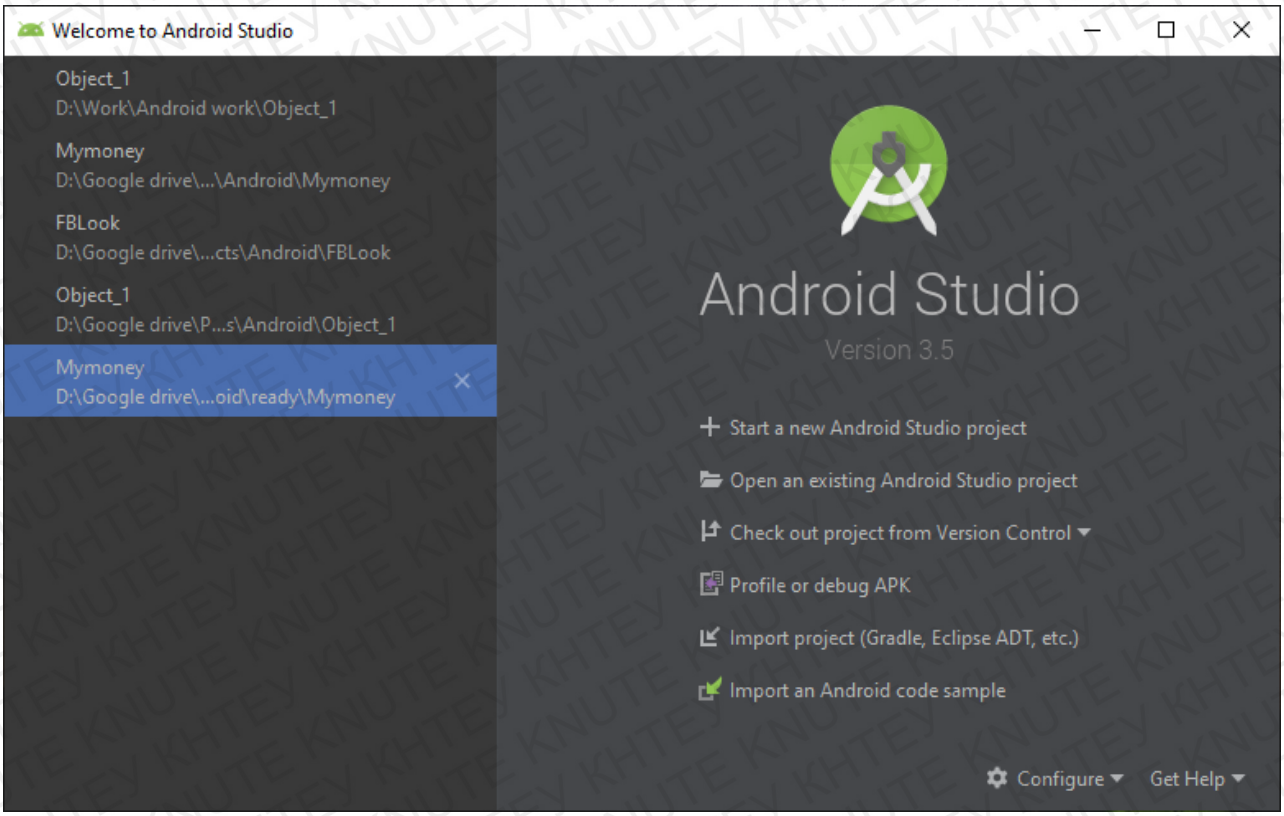

Рис. 3.1. Стартове вікно Android Studio.

Переходимо до наступної форми налаштування проекту тут потрібно обрати пустий проекту (Рис. 3.2).

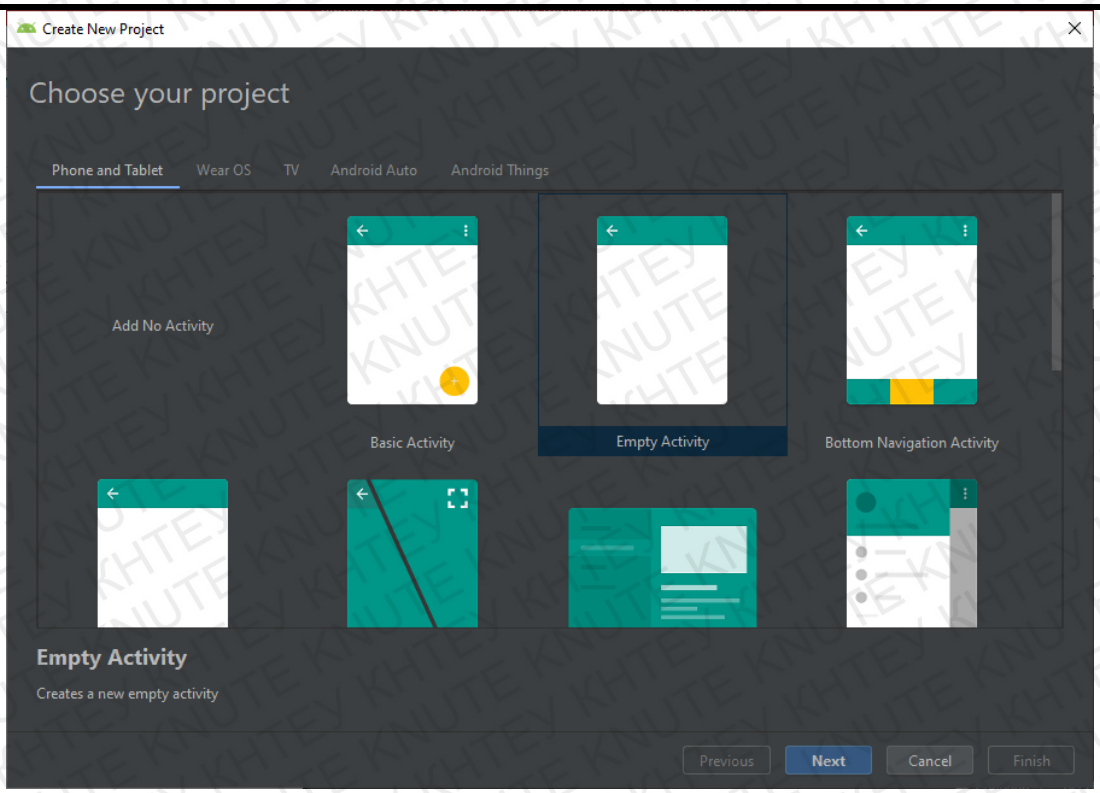

Рис. 3.2. Вікно вибору проекту Android Studio.

Далі вікно налаштування назви та каталогу розташування (Рис. 3.3).

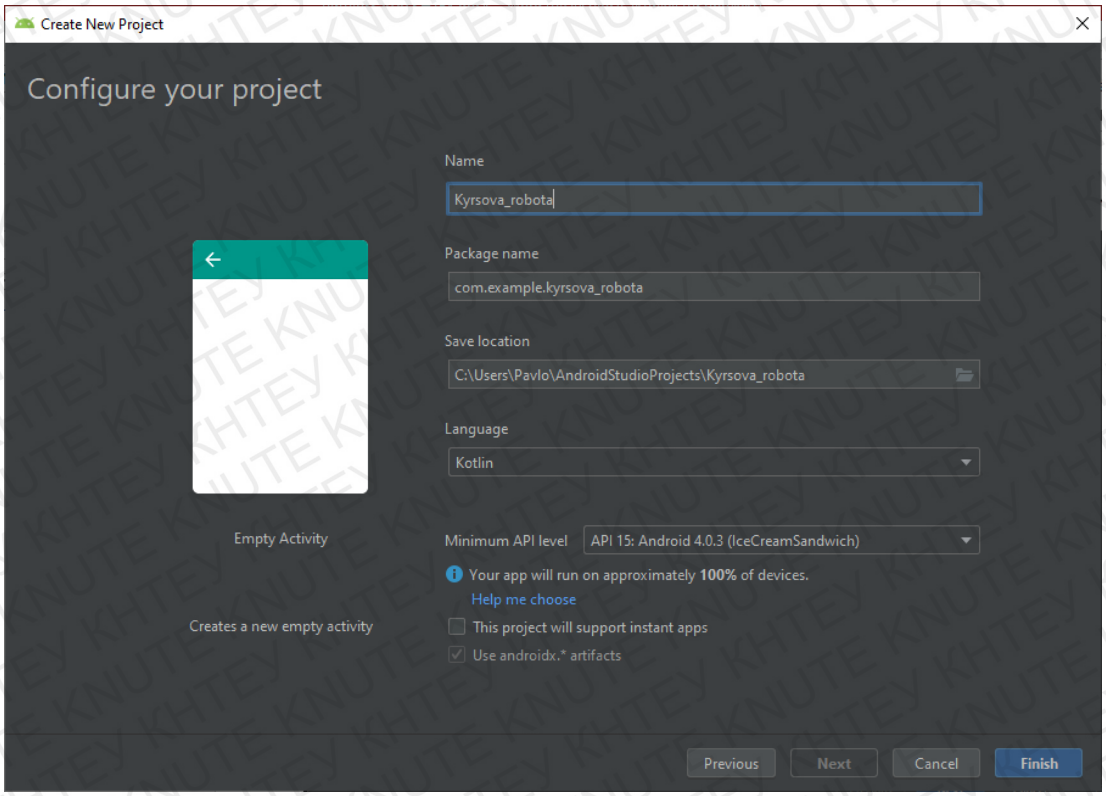

Рис. 3.3. Вікно налаштування назви та каталогу Android Studio.

Проект створений можна розпочинати роботу по створенню програмного додатку для обліку особистих фінансів (Рис. 3.4).

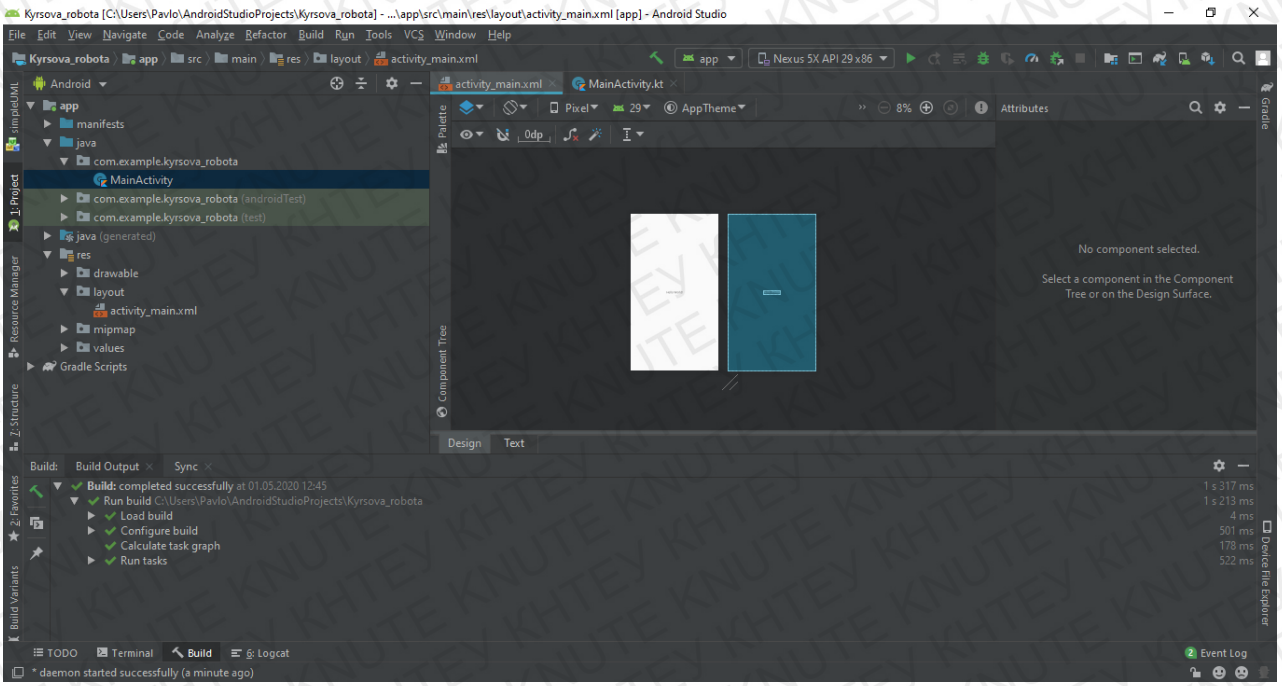

Рис. 3.4. Основне вікно роботи в Android Studio.

3.2. Створення класу DatabaseHelper.

Для створення класу DatabaseHelper потрібно натиснути ПКМ(правою кнопкою миші) на папку з проектом та вибрати пункт новий та java Class (Рис. 3.5).

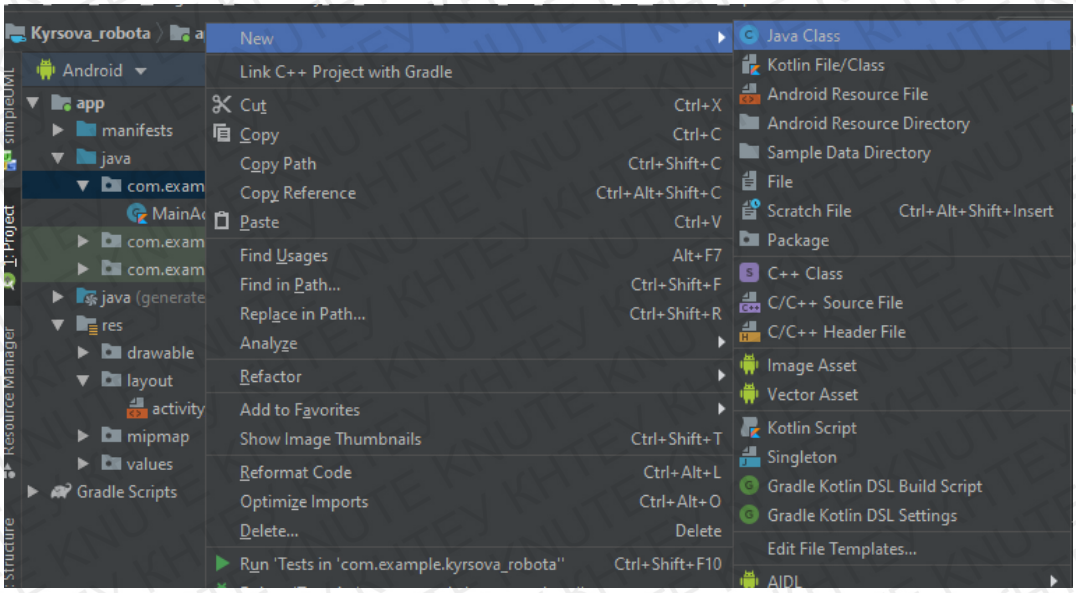

Рис. 3.5. Створення Class в Android Studio.

З’являється вікно з налаштуванням класу, змінюємо лише назву(Рис. 3.6).

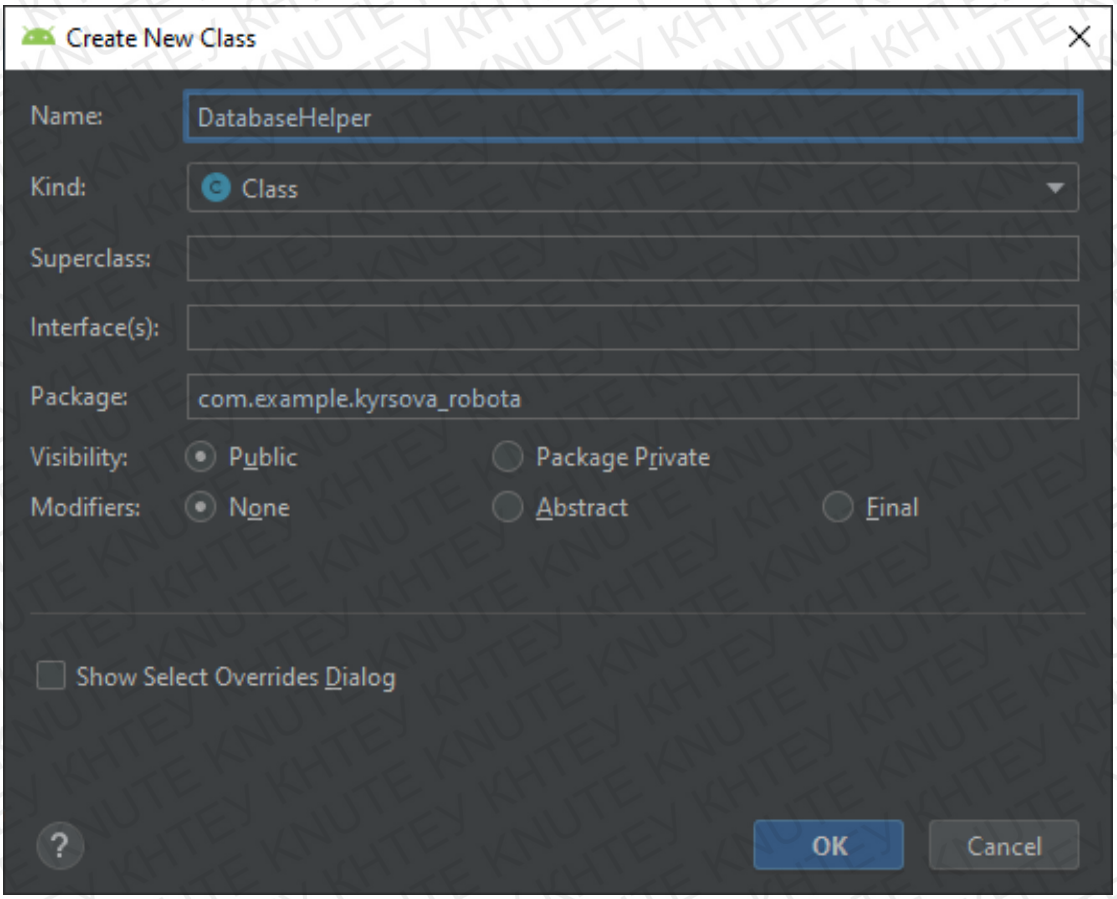

Рис. 3.6. Вікно налаштування Class в Android Studio.

Після натиску кнопки «ОК» з’являється вікно редагування тексту та починаємо писати програмний код для класу DatabaseHelper (Рис. 3.7).

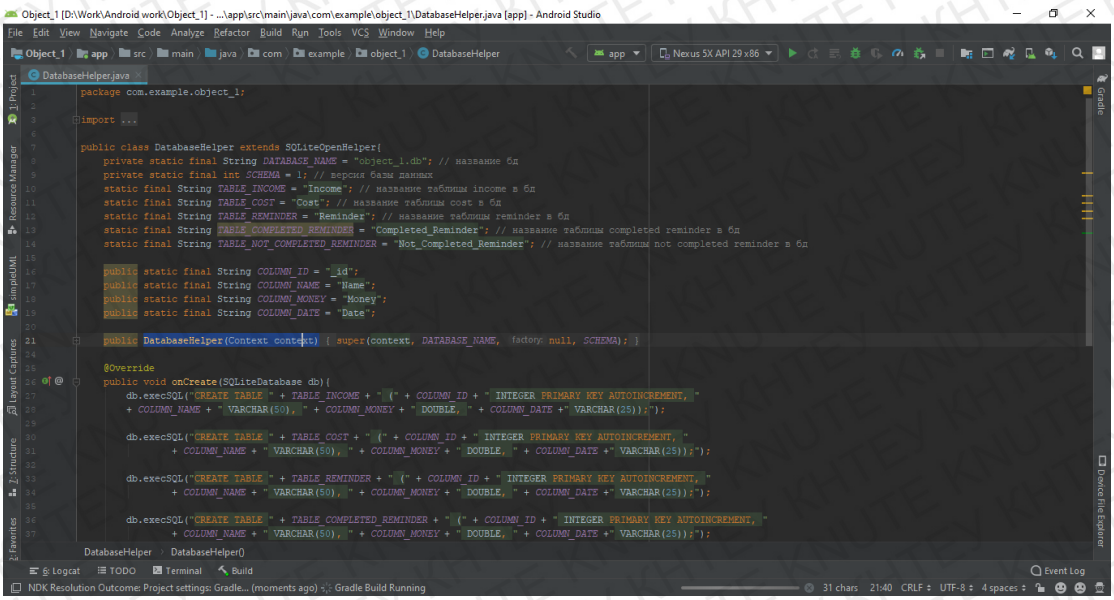

Рис. 3.7. Частина коду класу DatabaseHelper.

3.3. Створення класу DatabaseAdapter.

Створення класу DatabaseAdapter починається з кроків натиснути ПКМ(правою кнопкою миші) на папку з проектом та вибрати пункт новий та java Class (Рис. 3.8).

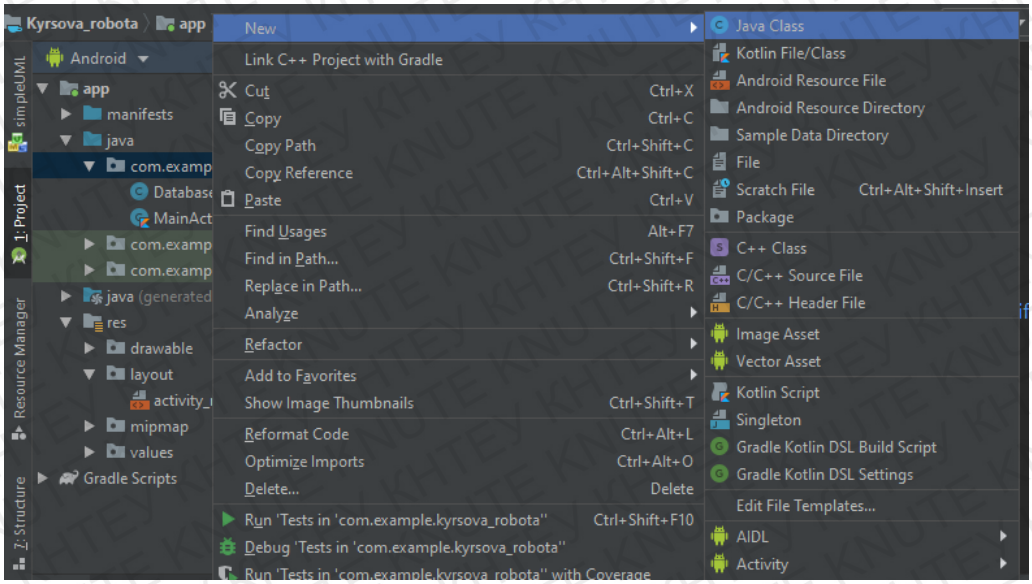

Рис. 3.8. Створення Class в Android Studio.

З’являється вікно з налаштуванням класу, змінюємо лише назву (Рис. 3.9).

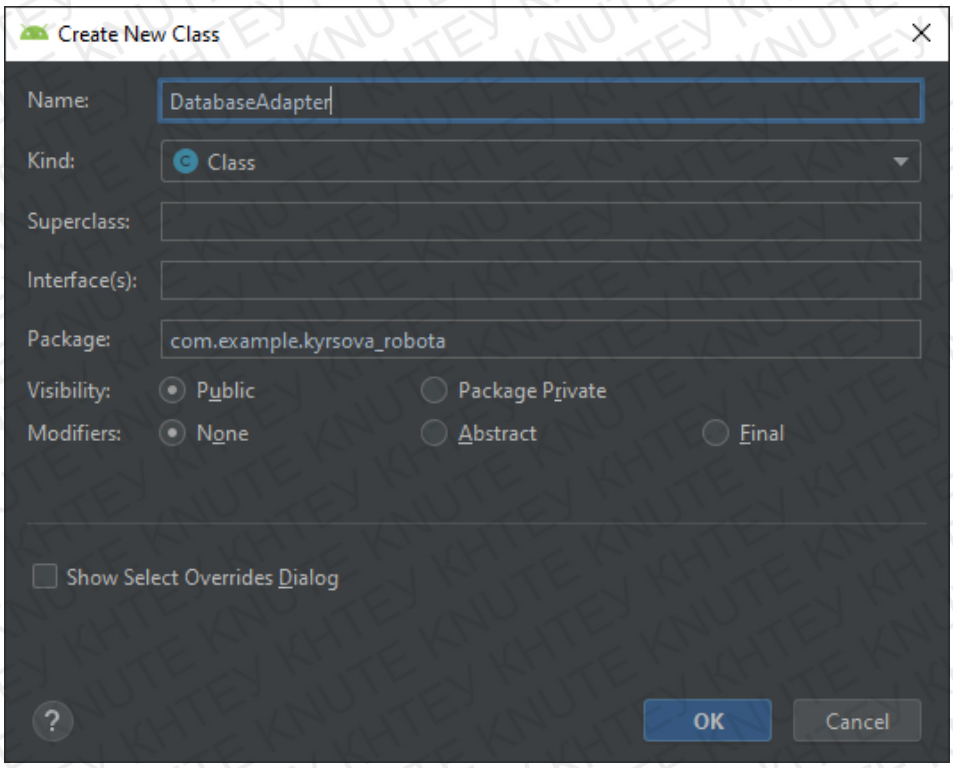

Рис. 3.9. Вікно налаштування Class в Android Studio.

Після натиску кнопки «ОК» з’являється вікно редагування тексту та починаємо писати програмний код для класу DatabaseAdapter (Рис. 3.10).

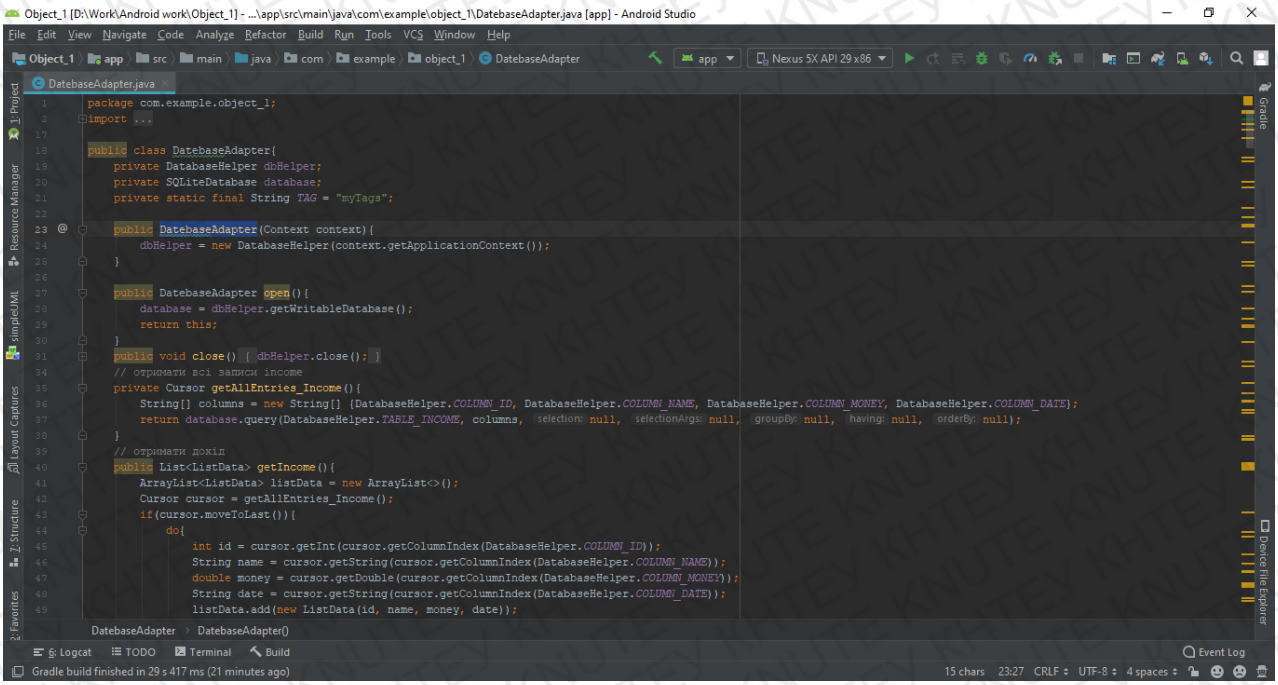

Рис. 3.10. Частина коду класу DatabaseAdapter.

3.4. Створення класу CanvasClass.

Створення класу для побудови діаграми з використанням даних з бази даних. Бібліотека Canvas дозволить малювати на шаблоні. Тип діаграми був обраний формі торта для економії простору на мобільному екрані (Рис. 3.11).

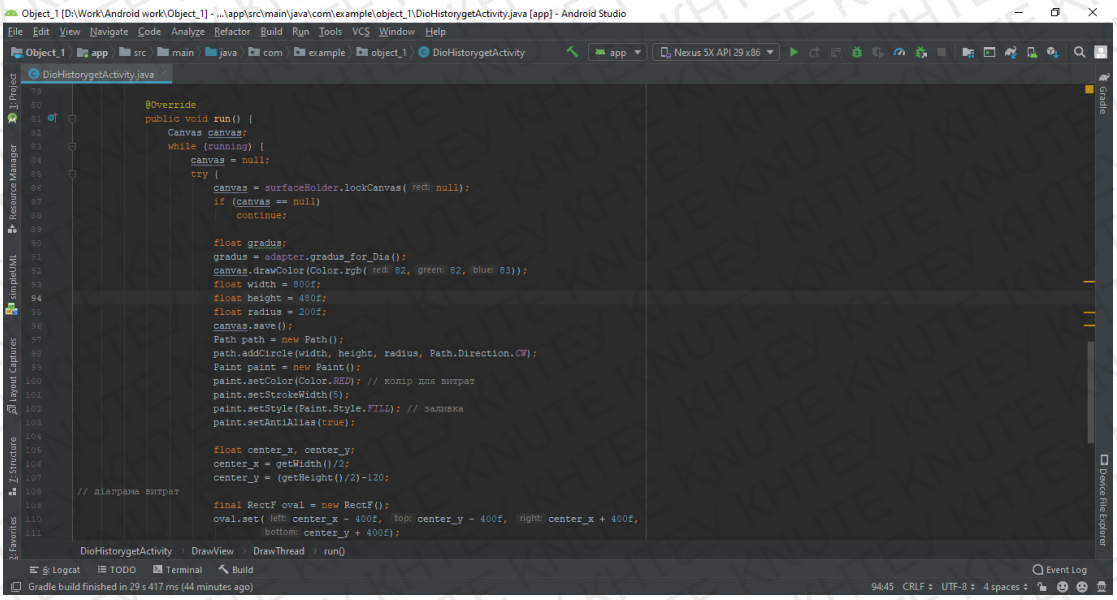

Рис. 3.11. Частина коду класу CanvasClass.

3.5. Створення Activity.

Основні класи були створенні тепер потрібно створити Activity для зручного керування користувачем (Рис. 3.12).

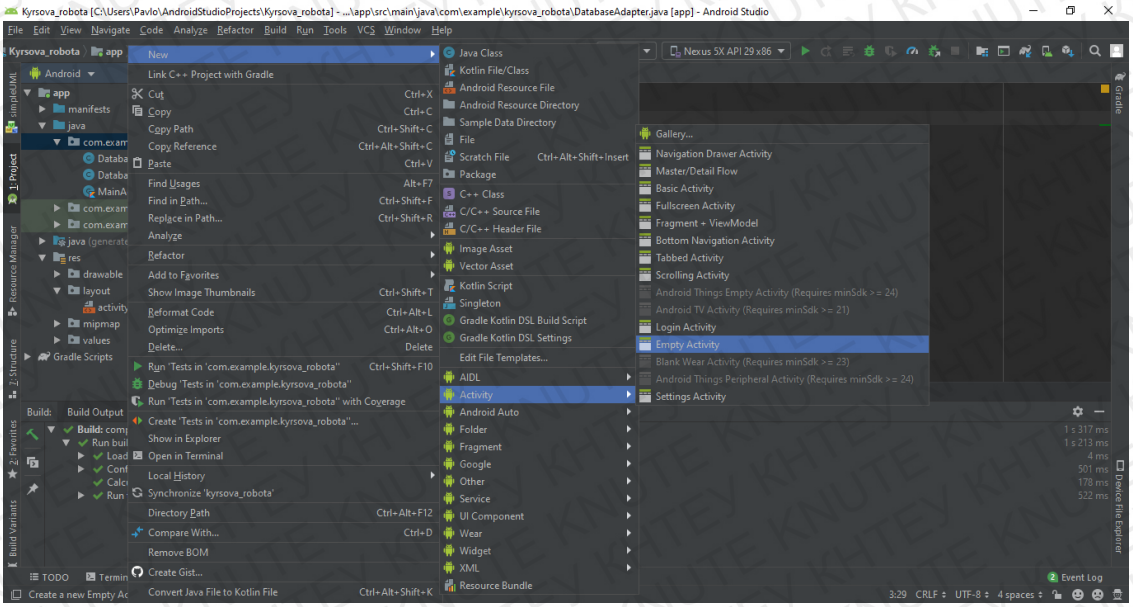

Рис. 3.12. Створення Activity.

Вікно відображення налаштування Activity (Рис. 3.13).

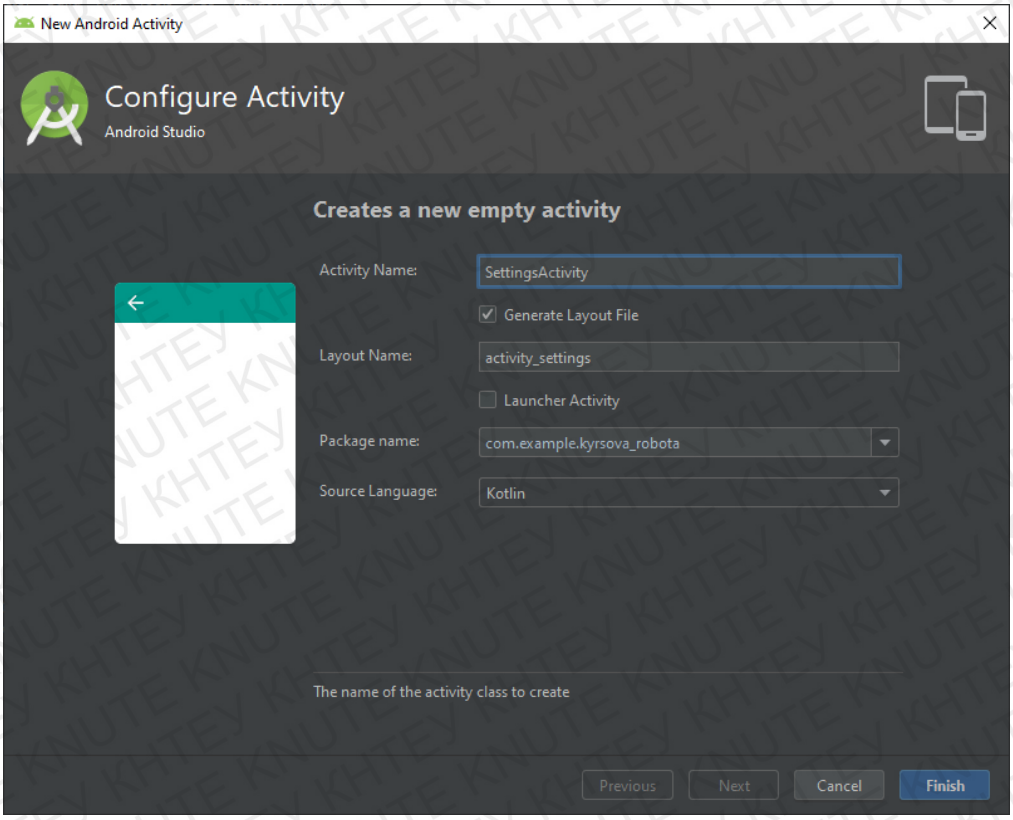

Рис. 3.13.Вікно налаштування Activity.

В панелі елементів проекту можна побачити створений клас activity та xml файл для розмітки дизайну та функціональності шаблону (Рис. 3.14).

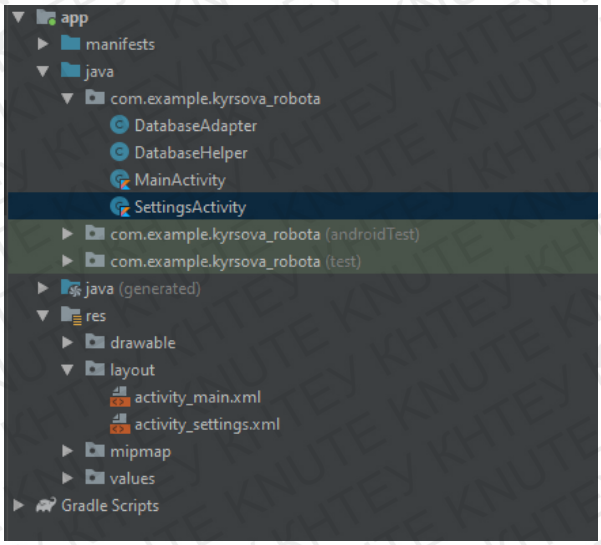

Рис. 3.14. Вікно елементів проекту.

Залишилося створити всі заплановані activity, які були зазначені в архітектурі програмного забезпечення (Рис. 3.15).

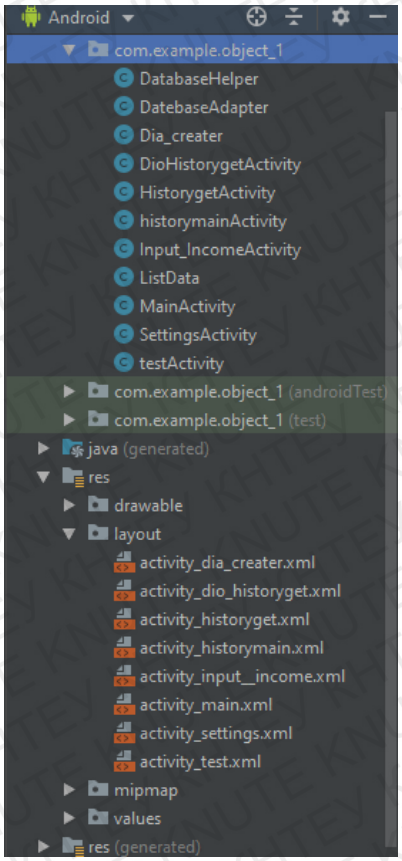

Рис. 3.15. Готові всі Activity.

Після створення всіх Activity можемо зробити перехід між ними (Рис. 3.16).

```
@Override
public void onClick(View v) {
    Intent intent = new Intent( packageContext: this, Input_IncomeActivity.class);
    switch (v.getId()){
        case R.id.btn_income:
            intent = new Intent( packageContext: this, Input_IncomeActivity.class);
            intent.putExtra( name: "table", income);
            intent.putExtra( name: "click",  value: 25);
            break;
        case R.id.btn_cost:
            intent = new Intent( packageContext: this, Input_IncomeActivity.class);
            intent.putExtra( name: "table", cost);
            intent.putExtra( name: "click",  value: 25);
            break;
        case R.id.btn_reminder:
            intent = new Intent( packageContext: this, Input_IncomeActivity.class);
            intent.putExtra( name: "table", reminder);
            intent.putExtra( name: "click",  value: 25);
            break;
        case R.id.btn_reminder_count:
            intent = new Intent( packageContext: this, Input_IncomeActivity.class);
            intent.putExtra( name: "table", reminder_count);
            intent.putExtra( name: "click",  value: 25);
            break;
        case R.id.btn_history_activity:
            intent = new Intent( packageContext: this, historymainActivity.class);
            break;
        case R.id.btn_settings:
            intent = new Intent( packageContext: this, SettingsActivity.class);
            break;
            default:
                intent = new Intent( packageContext: this, MainActivity.class);
```

Рис. 3.16. Приклад переходу між Activity при використані onClick та switch.

3.6. Створення дизайну Activity.

Налаштування activity виконується у файлі xml. Для дизайну було обрано простий інтерфейс для більш швидкого освоєння користувачем.

Головне вікно редагується файлом Activity_main.xml (Рис. 3.17).

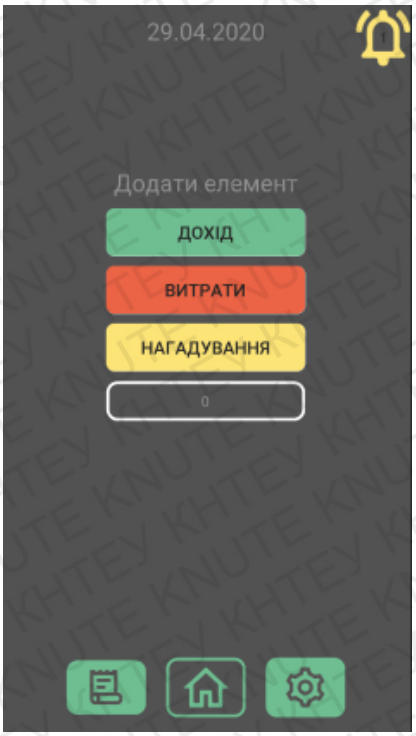

Рис. 3.17. Дизайн Activity_main.

Вікно Історія редагується файлом Activity_historymain.xml (Рис. 3.18).

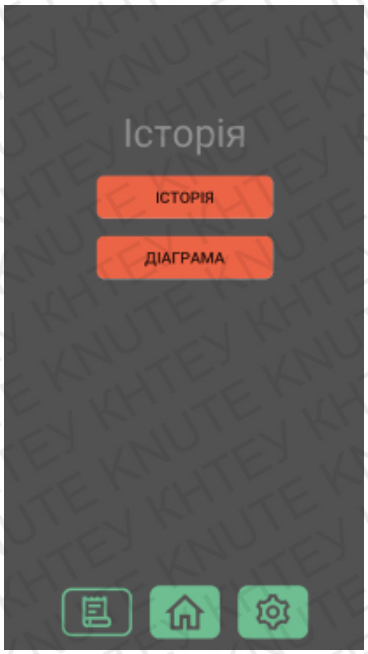

Рис. 3.18. Дизайн Activity_historimain.

Вікно налаштування редагується файлом Activity_settings.xml (Рис. 3.19).

Рис. 3.19. Дизайн Activity_ settings.

Вікно ввід інформації до бази даних редагується файлом Activity_inputIncome.xml (Рис. 3.20).

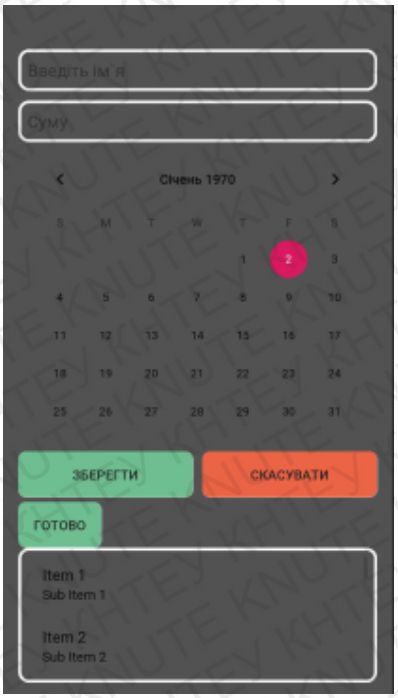

Рис. 3.20. Дизайн Activity_inputIncome.

Вікно вивід інформації з бази даних редагується файлом Activity_historyget.xml (Рис. 3.21).

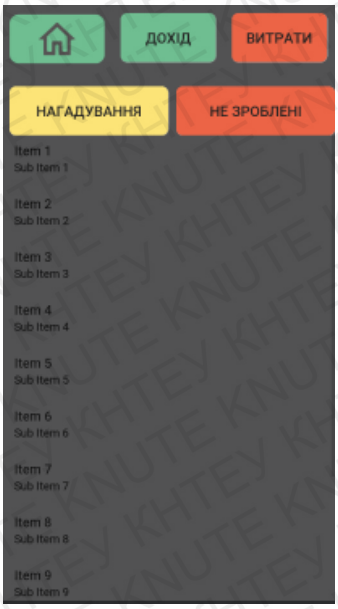

Рис. 3.21. Дизайн Activity_historyget.

Вікно налаштування редагується файлом Activity_historymain.xml (Рис. 3.18).

3.7. Інструкція застосування програмного програми ведення особистих фінансів.

Після скачування програми потрібно натиснути на програмний продукт з назвою «Mymoney» та іконку з монетою. Відкривається перше вікно програми (Рис. 3.22).

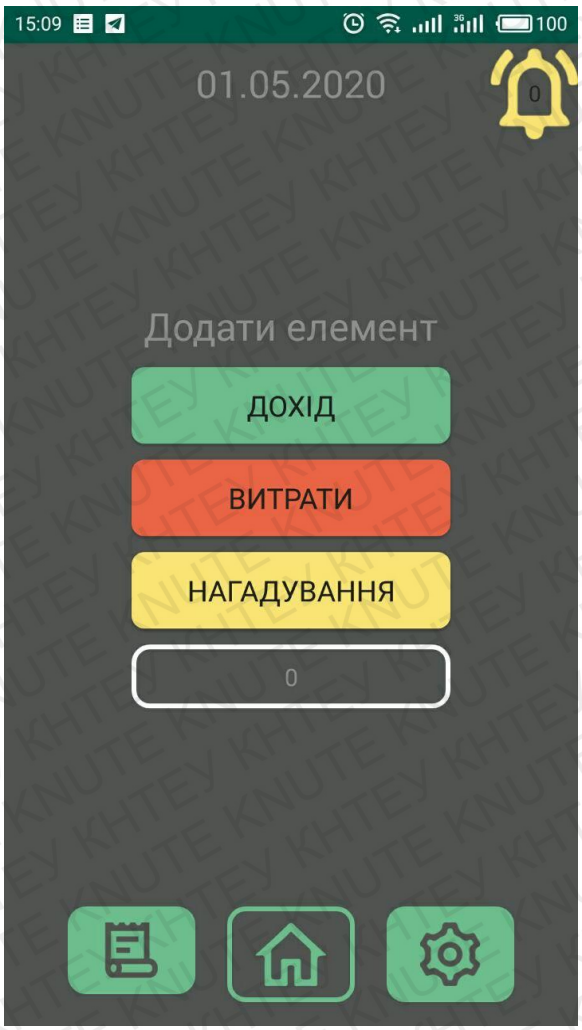

Рис. 3.22. Головне вікно.

Для початку потрібно ввести до бази даних свій доход, натисніть на кнопку дохід в пункті додати елемент. Вікно змінилося на записування доходу до бази даних. Перше поле призначено для вводу назви доходу, а друга для суми яку ви отримали (Рис. 3.23).

Рис. 3.23. Вікно вводу доходу.

Після заповнення натискаємо зберегти або якщо не треба вводити натискаємо скасувати. Користувача перекине до головного вікна.

Тепер потрібно вести витрати, якщо вони були зроблені. Натискаємо на витрати. Користувача перекидає на вікно заповнення витрат (Рис. 3.24). Тут також потрібно заповнити поле назва, сума. Якщо користувач зробить помилку в сумі яка перевищує його дохід то програма йому важить на це та не дасть записати даний пункт. Після збереження користувача перенесуть до головного вікна.

Рис. 3.24. Вікно вводу витрати.

Тепер пункт нагадування, після натискання на цей пункт знову перенесе до вікна заповнення (Рис. 3.25). Після заповнення та збереження користувача перекине на головну сторінку.

Рис. 3.25. Вікно вводу нагадування.

Після всіх заповнень на головному вікні знизу будуть показані повідомлення про правильність вводу інформації чи помилки (Рис. 3.26).

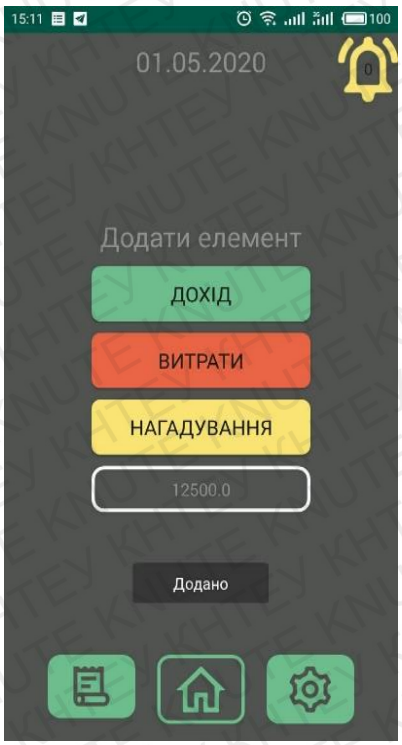

Рис. 3.26. Головне вікно з додано.

На головному вікні знизу можна побачити меню, в якому зображено лист, будинок та шестерня. По натиску на лист користувача перекине на вікно з історією де можна побачити дві кнопки (Рис. 3.27).

Рис. 3.27. Вікно історія.

Якщо користувач натисне на кнопку історія то його перекине на вікно з відображенням історії у вигляді таблиці або списком (Рис. 3.28). Перший вивід який побачить користувач це лист доходів, ще можна побачити кнопки витрати, нагадування, не зроблені.

Якщо натиснути на кнопку витрати то можна побачити лист витрат (Рис. 3.29). Якщо користувач побачить помилку вводі інформації то він з цього меню

може по натиску на лист витрат змінити його. Вікно зміни повністю аналогічне вікну вводу але тільки з однією відмінністю доданий календар.

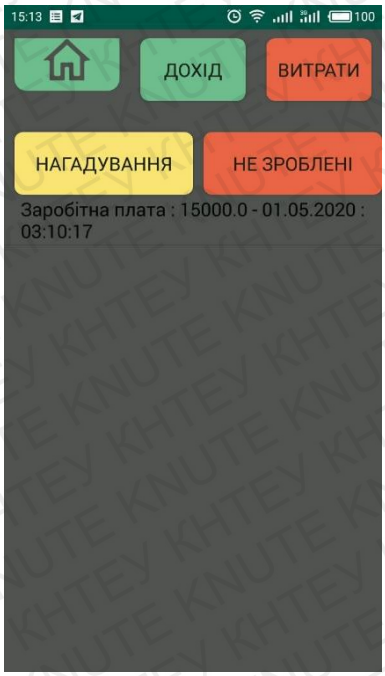

Рис. 3.28. Вікно історія в списком доходів.

Рис. 3.29. Вікно історія списком витрат.

Вікні історія при натисканні кнопки діаграма, користувача перекине на вікно діаграма де можна зробити редагування діаграми по неділям, місяцям, півріччя та за рік (Рис. 3.30). Після вибору редагування користувач натискає на клавішу створити діаграму та його перекине до діаграми (Рис. 3.31). Де зеленим кольором вказано доходи, а червони витрати.

Рис. 3.30. Вікно історія діаграма.

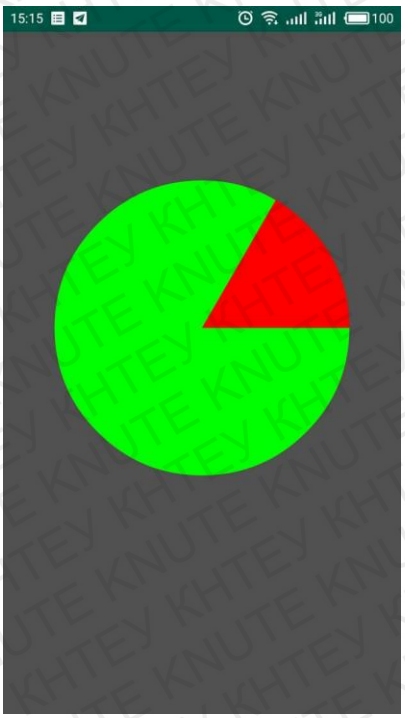

Рис. 3.31. Вікно діаграма.

Вікно налаштування в ньому можна змінити мову додатку (Рис. 3.32).

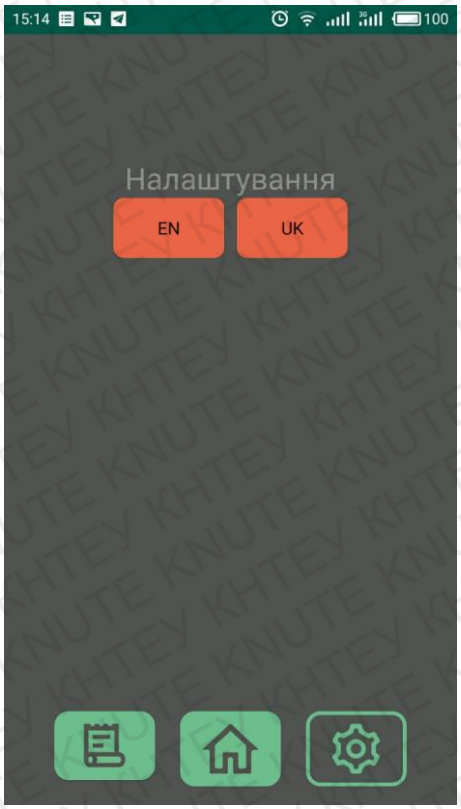

Рис. 3.32. Вікно налаштування.

Користувач якщо володіє англійською мовою то може обрати пункт англійська мова (Рис. 3.33).

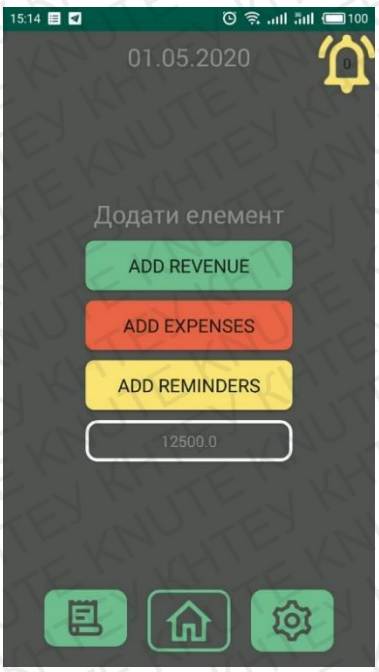

Рис. 3.33. Головне вікно з Англійською мовою.

Користувач бачить в верхньому правому куті дзвіночок з цифрою 1 (Рис. 3.34).

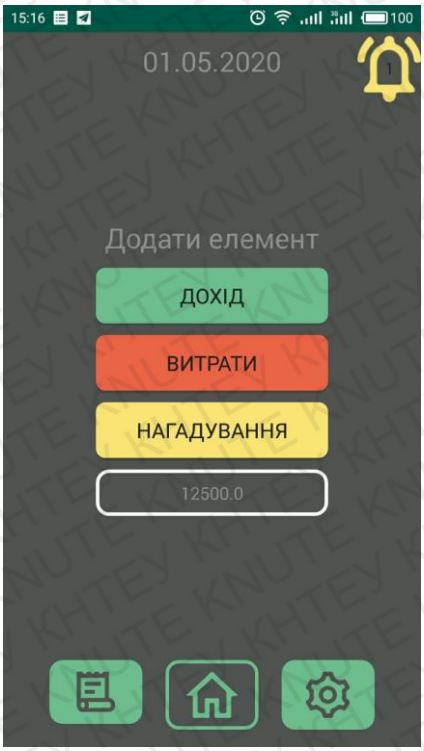

Рис. 3.34. Головне вікно з дзвіночком.

Це значить що на сьогодні є нагадування, якщо натиснути на нього то користувача перекине в вікно нагадування (Рис. 3.35).

Рис. 3.35. Вікно нагадування на сьогодні.

Натиском на лист з нагадування користувач побачить панель вводу інформації до бази даних назву, суму та дату на календарі (Рис. 3.36). Якщо користувачу потрібно змінити суму чи назву він може в цьому же вікні змінити і воно збережеться. Після натискання на готово, дана інформація переміститься до таблиці витрат.

Рис. 3.36. Вікно додавання нагадування на сьогодні.

В головному вікні ми можемо побачити в рамці число (Рис. 3.37). Це число загальної суми на даний момент.

Рис. 3.37. Головне вікно з числом.

Користувач може подивитися на правильність вводу інформації у вікні історія (Рис. 3.38).

Рис. 3.38. Вікно історія таблиця витрат.

3.8. Висновок до розділу 3.

Отже, в даному розділі було описано створення програмного додатку з використанням Android studio. Описано створення основних класів та activity. Поетапно описано інструкцію використання програми обліку особистих фінансів. Було зроблено:

1. Клас DatabaseHelper.
2. Клас DatabaseAdapter.
3. Клас CanvasClass.
4. Activity Main.
5. Activity History.
6. Activity Settings.
7. Activity GetInput.
8. Activity Diagram.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Одже, було виконано роботу по створенню програми для ведення особистих фінансів на операційну систему Android. Даний програмний продукт це мобільний посібник витрат який показує людині на його витрати та дохід. Це дозволить проаналізувати свої активи та пасиви і зробити висновок. З початку було розроблена архітектура програми обліку особистих фінансів з якої потім було розроблено базу даних, класи для воду та виводу інформації та інтерфейс програми.

Програма виконує:

- Запис до бази даних витрат, доходів, нагадувань.
- Вивід інформації витрат, доходів, нагадувань до таблиць.
- Малювання діаграми.
- Налаштування мови.
- Нагадування за поточний день.
- Редагування даних в базі даних.
- Видалення з бази даних.
- Вивід поточної суми

В першому розділу було описано технічне завдання у вимогах для створення програмного продукту. Існує багато програм для підрахунків та розподілу коштів, які визначають особистий та сімейний бюджет. Не всі програми доступні для певних верст населення. Не кожна людина, тем більш дитина зможе користуватися програмою з більш складними функціями, але взагалі, у всіх програмах є одна мета навчити людей правильно користуватись своїми коштами. Навчити людей заощаджувати, досягати своєї мети та бути фінансово незалежними.

Також розписано вимоги до створення бази даних, бібліотеки керування БД, бібліотеки створення діаграми. Розглянута ідея та вирішення проблеми.

Другому розділу описано два класи в UML діаграмах та розглянута архітектура проекту. Третьому розділу було описано створення програмного додатку з використанням Android studio. Описано створення основних класів та activity. Поетапно описано інструкцію використання програми «Mymonay».

Було зроблено:

1. Клас DatabaseHelper.
2. Клас DatabaseAdapter.
3. Клас CanvasClass.
4. Activity Main.
5. Activity History.
6. Activity Settings.
7. Activity GetInput.
8. Activity Diagram.
9. Activity Appitems.

В програмі обліку особистих фінансів була задача зробити більш доступний інтерфейс та щоб вона була доступною для всіх вікових категорій людей. Mymonay, перший крок для досягнення фінансових цілей та фінансової незалежності. За її допомогою склавши особистий та сімейний бюджет, люди зможуть грамотно витрачати кошти, що підніме економічний стан людей. Мета програми, навчитися слідкувати та берегти кошти та щоб витрати не перевищували доходи. Інвестували сольдо в майбутнє. Програма розрахована як для дорослого так і для дитини, щоб з дитинства вони привчались до накопичували заощадженню.

При дослідженні всіх існуючих програм, була представлена більш зручна в користуванні інтерфейсу, функцій програми для широкого використані.

Пропозиції для програми, це додати більше можливості вводу даних до таблиці наприклад з використанням бібліотеки tensorflow, можна зробити зчитування даних з чеків, для зручності вводу даних. Ще можна зробити підв'язку програми до карточки банку, щоб автоматично записувати дані з електронних маніпуляцій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Berglund, Tim; McCullough, Matthew (July 2011). Building and Testing with Gradle. Foreword by Hans Dockter (First ed.). O'Reilly Media. p. 116. IP ПРОГРАМНИХ ЗАСОБІВ ТА АРХІТЕКТУРИSBN 978-1-4493-0463-8.

2. IP ПРОГРАМНИХ ЗАСОБІВ ТА АРХІТЕКТУРИkkink, Hubert (November 2012). Gradle Effective IP ПРОГРАМНИХ ЗАСОБІВ ТА АРХІТЕКТУРИmplementation Guide (First ed.). Packt Publishing. p. 382. IP ПРОГРАМНИХ ЗАСОБІВ ТА АРХІТЕКТУРИSBN 978-1849518109.

3. Berglund, Tim; McCullough, Matthew (May 2013). Gradle DSLs (First ed.). O'Reilly Media. pp. 50 est. IP ПРОГРАМНИХ ЗАСОБІВ ТА АРХІТЕКТУРИSBN 978-1-4493-0467-6.

4. Muschko, Benjamin (Fall 2013). Gradle IP ПРОГРАМНИХ ЗАСОБІВ ТА АРХІТЕКТУРИn Action (First ed.). Manning Publications. p. 390. IP ПРОГРАМНИХ ЗАСОБІВ ТА АРХІТЕКТУРИSBN 9781617291302.

5. Галина Острякова, Хто поклав грощі в тумбу, Острякова Г. Ф., Київ. 11. Effective java Thrid Edition, Joshua Bloch/ М.: Лори, 2002. — 224 с

6. Java Puzzlers: Traps, Pitfalls, and Corner Cases, ISBN 0-321-33678-X, 200513. Date C. J. Introduction to Database Systems = Introduction to Database Systems. - 8th ed. - M.: Williams, 2005 .-- 1328 p.

7. Kuznetsov S. D. Fundamentals of databases. - 2nd ed. - M .: Internet University of Information Technologies; Laboratory of Knowledge, 2007. - 484 p.

8. Kogalovsky M.R. Encyclopedia of database technologies. - M.: Finance and Statistics, 2002. - 800 p.

9. Garcia-Molina G., Ulman J., Widom J. Database systems. Full course. - M.: Williams, 2003 .-- 1088 p.

10. Kogalovsky M. R. Encyclopedia of database technologies. - M.: Finance and Statistics, 2002. - 800 p.

11. Tsikritzis D., Lochowski F. Data models - D. Tsichritzis, F. Lochovsky. Data Models. Prentice Hall, 1982. - 344 p.

12. Ben Fort. Learn your own SQL query language / Per. from English - 3rd ed. - M .: Dialectics, 2005 .-- 288 p.

13. Paul Wilton, John Colby. SQL query language for beginners / Per. from English - M .: Dialectics, 2005 .-- 496 p.

14. K.J. Date. Introduction to database systems / Per. from English - 8th ed. - M.: Williams, 2005 .-- 1328 p.

15. Kevin Kline. SQL Directory. - M .: Kudits-Obraz, 2006 .-- 832 p.

16. Yuriy Bekarevich. Microsoft Access 2014. Self-proprietor of “The Doorway of the Tribute” / U. Bekarevich, N. Pushkina. 2014.–89 p.

ДОДАТКИ

Дотаток А

Лістинг класу DatabaseHelper

```
package com.example.object_1;

import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteDatabase;
import android.content.Context;

public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "object_1.db"; // название бд
    private static final int SCHEMA = 1; // версия базы данных
    static final String TABLE_INCOME = "Income"; // название таблицы income в бд
    static final String TABLE_COST = "Cost"; // название таблицы cost в бд
    static final String TABLE_REMINDER = "Reminder"; // название таблицы reminder в бд
    static final String TABLE_COMPLETED_REMINDER = "Completed_Reminder"; // название таблицы completed reminder в бд
    static final String TABLE_NOT_COMPLETED_REMINDER = "Not_Completed_Reminder"; // название таблицы not completed reminder в бд
    static final String TABLE_TYPE_COSTS = "Type_COSTS";
    public static final String COLUMN_ID = "_id";
    public static final String COLUMN_NAME = "Name";
    public static final String COLUMN_MONEY = "Money";
    public static final String COLUMN_DATE = "Date";
    public static final String COLUMN_TYPE = "Type";
    public static final String COLUMN_COLOR = "Color";
```

```
public DatabaseHelper(Context context){
    super(context, DATABASE_NAME, null, SCHEMA);
}

@Override
public void onCreate(SQLiteDatabase db){
    db.execSQL("CREATE TABLE " + TABLE_INCOME + " (" + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
    + COLUMN_NAME + " VARCHAR(50), " + COLUMN_MONEY + " DOUBLE, " + COLUMN_DATE +" VARCHAR(25));");

    db.execSQL("CREATE TABLE " + TABLE_COST + " (" + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
    + COLUMN_NAME + " VARCHAR(50), " + COLUMN_MONEY + " DOUBLE, "+ COLUMN_TYPE + " VARCHAR(50)," + COLUMN_DATE +" VARCHAR(25));");

    db.execSQL("CREATE TABLE " + TABLE_REMINDER + " (" + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
    + COLUMN_NAME + " VARCHAR(50), " + COLUMN_MONEY + " DOUBLE, "+ COLUMN_TYPE + " VARCHAR(50)," + COLUMN_DATE +" VARCHAR(25));");

    db.execSQL("CREATE TABLE " + TABLE_COMPLETED_REMINDER + " (" + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
    + COLUMN_NAME + " VARCHAR(50), " + COLUMN_MONEY + " DOUBLE, "+ COLUMN_TYPE + " VARCHAR(50)," + COLUMN_DATE +" VARCHAR(25));");

    db.execSQL("CREATE TABLE " + TABLE_NOT_COMPLETED_REMINDER + " (" + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
```

```
                + COLUMN_NAME + " VARCHAR(50), " + COLUMN_MONEY +
" DOUBLE, "+ COLUMN_TYPE + " VARCHAR(50)," + COLUMN_DATE +"
VARCHAR(25));");


            db.execSQL("CREATE TABLE " + TABLE_TYPE_COSTS + " (" +
COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + COLUMN_NAME + " VARCHAR(50), " + COLUMN_COLOR +"
VARCHAR(25));");
        }


        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
            db.execSQL("DROP TABLE IF EXISTS " + TABLE_INCOME);
            onCreate(db);
            db.execSQL("DROP TABLE IF EXISTS " + TABLE_COST);
            onCreate(db);
            db.execSQL("DROP TABLE IF EXISTS " + TABLE_REMINDER);
            onCreate(db);
            db.execSQL("DROP TABLE IF EXISTS " +
TABLE_COMPLETED_REMINDER);
            onCreate(db);
            db.execSQL("DROP TABLE IF EXISTS " +
TABLE_NOT_COMPLETED_REMINDER);
            onCreate(db);
            db.execSQL("DROP TABLE IF EXISTS " + TABLE_TYPE_COSTS);
            onCreate(db);
        }


    }
```

Додаток Б

Лістинг класу DatabaseAdapter

```
package com.example.object_1;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.DatabaseUtils;
import android.database.sqlite.SQLiteDatabase;
import android.nfc.Tag;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;

public class DatebaseAdapter extends AppCompatActivity {
   private DatabaseHelper dbHelper;
   private SQLiteDatabase database;
   private static final String TAG = "myTags";

   public DatebaseAdapter(Context context){
      dbHelper = new DatabaseHelper(context.getApplicationContext());
   }
```

```
public DatebaseAdapter open(){
    database = dbHelper.getWritableDatabase();
    return this;
}
public void close(){
    dbHelper.close();
}
// отримати всі записи income
private Cursor getAllEntries_Income(){
    String[] columns = new String[] {DatabaseHelper.COLUMN_ID, DatabaseHelper.COLUMN_NAME, DatabaseHelper.COLUMN_MONEY, DatabaseHelper.COLUMN_DATE};
    return database.query(DatabaseHelper.TABLE_INCOME, columns, null, null, null, null, null);
}
// отримати дохід
public List<ListData> getIncome(){
    ArrayList<ListData> listData = new ArrayList<>();
    Cursor cursor = getAllEntries_Income();
    if(cursor.moveToLast()){
        do{
            int id = cursor.getInt(cursor.getColumnIndex(DatabaseHelper.COLUMN_ID));
            String name = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_NAME));
            double money = cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
            String date = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_DATE));
            listData.add(new ListData(id, name, money, date));
```

```
        } while(cursor.moveToPrevious());
      }
      cursor.close();
      return listData;
    }
    //sum_inclome
    public double sum_income(){
      open();
      double sum = 0;
      ArrayList<ListData> listData = new ArrayList<>();
      Cursor cursor = getAllEntries_Income();
      if(cursor.moveToLast()){
        do{
          int id = cursor.getInt(cursor.getColumnIndex(DatabaseHelper.COLUMN_ID));
          String name = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_NAME));
          double money = cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
          sum = sum + cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
          String date = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_DATE));
          listData.add(new ListData(id, name, money, date));
        } while(cursor.moveToPrevious());
      }
      cursor.close();
      close();
      return sum;
    }
```

```
    // отримати граф income
    private  long getCount_Income(){
        return
DatabaseUtils.queryNumEntries(database,DatabaseHelper.TABLE_INCOME);
    }
    // пошук по id income
    public ListData getIncome(long id){

        open();
        ListData listData = null;
        String query = String.format("SELECT * FROM %s WHERE
%s=?",DatabaseHelper.TABLE_INCOME, DatabaseHelper.COLUMN_ID);
        Cursor cursor = database.rawQuery(query, new String[]{ String.valueOf(id)});
        if(cursor.moveToFirst()){
            String name =
cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_NAME));
            double money =
cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
            String date =
cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_DATE));
            listData = new ListData(id,name,money,date);
        }
        cursor.close();
        close();
        return listData;
    }
    // внесення даних до таблиці income
    public long insert_income(ListData income){
        ContentValues cv = new ContentValues();
        cv.put(DatabaseHelper.COLUMN_NAME, income.getName());
```

```
        cv.put(DatabaseHelper.COLUMN_MONEY, income.getMoney());
        cv.put(DatabaseHelper.COLUMN_DATE, income.getDate());
        return database.insert(DatabaseHelper.TABLE_INCOME, null, cv);
    }
    //видалення з таблиці income
    public  long delete_income(long incomId){
        String whereClause = "_id = ?";
        String[] whereArgs = new String[]{String.valueOf(incomId)};
        return database.delete(DatabaseHelper.TABLE_INCOME, whereClause,
whereArgs);
    }
    //редагування елементів в таблиці income
    public long update_income(ListData listData){
        String whereClause = DatabaseHelper.COLUMN_ID + "=" +
String.valueOf(listData.getId());
        ContentValues cv = new ContentValues();
        cv.put(DatabaseHelper.COLUMN_NAME, listData.getName());
        cv.put(DatabaseHelper.COLUMN_MONEY, listData.getMoney());
        cv.put(DatabaseHelper.COLUMN_DATE, listData.getDate());
        return database.update(DatabaseHelper.TABLE_INCOME, cv, whereClause,
null);
    }
    //Cost
    private Cursor getAllEntries_Cost(){
        String[] columns = new String[] {DatabaseHelper.COLUMN_ID,
DatabaseHelper.COLUMN_NAME,
DatabaseHelper.COLUMN_MONEY,DatabaseHelper.COLUMN_TYPE,DatabaseHe
lper.COLUMN_DATE};
        return database.query(DatabaseHelper.TABLE_COST, columns, null, null,
null, null);
    }
```

```
    public List<ListData_type_costs> getCost(){
        ArrayList<ListData_type_costs> listData = new ArrayList<>();
        Cursor cursor = getAllEntries_Cost();
        if(cursor.moveToLast()){
            do{
                int id = cursor.getInt(cursor.getColumnIndex(DatabaseHelper.COLUMN_ID));
                String name = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_NAME));
                double money = cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
                String date = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_DATE));
                String type = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_TYPE));
                listData.add(new ListData_type_costs(id, name, money, date, type));
            } while(cursor.moveToPrevious());
        }
        cursor.close();
        return listData;
    }
// сум витрат
    public double sum_costs(){
        open();
        double sum = 0;
        ArrayList<ListData_type_costs> listData = new ArrayList<>();
        Cursor cursor = getAllEntries_Cost();
        if(cursor.moveToLast()){
            do{
```

```
            int id = cursor.getInt(cursor.getColumnIndex(DatabaseHelper.COLUMN_ID));
            String name = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_NAME));
            double money = cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
            sum = sum + cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
            String date = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_DATE));
            String type = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_TYPE));
            listData.add(new ListData_type_costs(id, name, money, date, type));
        } while(cursor.moveToPrevious());
    }
    cursor.close();
    close();
    return sum;
  }
  // отримати граф cost
  private  long getCount_Cost(){
    return DatabaseUtils.queryNumEntries(database,DatabaseHelper.TABLE_COST);
  }
  // пошук по id cost
  public ListData_type_costs getCost(long id){
    ListData_type_costs listData = null;
    String query = String.format("SELECT * FROM %s WHERE %s=?",DatabaseHelper.TABLE_COST, DatabaseHelper.COLUMN_ID);
    Cursor cursor = database.rawQuery(query, new String[]{ String.valueOf(id)});
    if(cursor.moveToFirst()){
```

```
        String name = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_NAME));
        double money = cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
        String date = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_DATE));
        String type = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_TYPE));
        listData = new ListData_type_costs(id,name,money,date,type);
      }
      cursor.close();
      return listData;
    }
    // пошук по data income
    public List<ListData> getIncome_date(String date_){
      ArrayList<ListData>  listData = new ArrayList<>();
      Cursor cursor = database.rawQuery("SELECT * FROM "+DatabaseHelper.TABLE_INCOME+" WHERE "+ DatabaseHelper.COLUMN_DATE +" like '%"+ date_+"%'", null);
      if(cursor.moveToLast()){
        do{
          int id = cursor.getInt(cursor.getColumnIndex(DatabaseHelper.COLUMN_ID));
          String name = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_NAME));
          double money = cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
          String date = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_DATE));
          listData.add(new ListData(id, name, money, date));
        } while(cursor.moveToPrevious());
      }
```

```
        cursor.close();
        return listData;
    }
    public String Add_xpenses(){
        String s = getString(R.string.Add_expenses);
        return s;
    }
    // пошук по data cost
    public List<ListData_type_costs> getCost_date(String date_){
        ArrayList<ListData_type_costs>  listData = new ArrayList<>();
        Cursor cursor = database.rawQuery("SELECT * FROM
"+DatabaseHelper.TABLE_COST+" WHERE "+ DatabaseHelper.COLUMN_DATE
+" like '%"+ date_+"%'", null);
        if(cursor.moveToLast()){
            do{
                int id =
cursor.getInt(cursor.getColumnIndex(DatabaseHelper.COLUMN_ID));
                String name =
cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_NAME));
                double money =
cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
                String date =
cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_DATE));
                String type =
cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_TYPE));
                listData.add(new ListData_type_costs(id, name, money, date,type));
            } while(cursor.moveToPrevious());
        }
        cursor.close();
        return listData;
    }
```

```
    // пошук по data sum cost
    public String getCost_date_sum(String date_){
        double sum = 0;
        open();
        Cursor cursor = database.rawQuery("SELECT * FROM
"+DatabaseHelper.TABLE_COST+" WHERE "+ DatabaseHelper.COLUMN_DATE
+" like '%"+ date_+"%'", null);
        if(cursor.moveToLast()){
            do{
                double money =
cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
                sum = sum + money;
            } while(cursor.moveToPrevious());
        }
        cursor.close();
        close();
        return String.valueOf(sum);
    }
    // пошук по data sum income
    public String getIncome_date_sum(String date_){
        double sum = 0;
        open();
        Cursor cursor = database.rawQuery("SELECT * FROM
"+DatabaseHelper.TABLE_INCOME+" WHERE "+
DatabaseHelper.COLUMN_DATE +" like '%"+ date_+"%'", null);
        if(cursor.moveToLast()){
            do{
                double money =
cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
                sum = sum + money;
```

```
        } while(cursor.moveToPrevious());
      }
      cursor.close();
      close();
      return String.valueOf(sum);
   }
   // внесення даних до таблиці cost
   public long insert_cost(ListData_type_costs listData){
      ContentValues cv = new ContentValues();
      cv.put(DatabaseHelper.COLUMN_NAME, listData.getName());
      cv.put(DatabaseHelper.COLUMN_MONEY, listData.getMoney());
      cv.put(DatabaseHelper.COLUMN_DATE, listData.getDate());
      cv.put(DatabaseHelper.COLUMN_TYPE, listData.getType());
      return database.insert(DatabaseHelper.TABLE_COST, null, cv);
   }
   //видалення з таблиці cost
   public  long delete_cost(long incomId){
      String whereClause = "_id = ?";
      String[] whereArgs = new String[]{String.valueOf(incomId)};
      return database.delete(DatabaseHelper.TABLE_COST, whereClause,
whereArgs);
   }
   //редагування елементів в таблиці cost
   public long update_cost(ListData_type_costs listData){
      String whereClause = DatabaseHelper.COLUMN_ID + "=" +
String.valueOf(listData.getId());
      ContentValues cv = new ContentValues();
      cv.put(DatabaseHelper.COLUMN_NAME, listData.getName());
      cv.put(DatabaseHelper.COLUMN_MONEY, listData.getMoney());
```

```
        cv.put(DatabaseHelper.COLUMN_DATE, listData.getDate());
        cv.put(DatabaseHelper.COLUMN_TYPE, listData.getType());
        return database.update(DatabaseHelper.TABLE_COST, cv, whereClause, null);
    }
    //Reminder
    private Cursor getAllEntries_Reminder(){
        String[] columns = new String[] {DatabaseHelper.COLUMN_ID,
DatabaseHelper.COLUMN_NAME,
DatabaseHelper.COLUMN_MONEY,DatabaseHelper.COLUMN_TYPE,
DatabaseHelper.COLUMN_DATE};
        return database.query(DatabaseHelper.TABLE_REMINDER, columns, null,
null, null, null, null);
    }
    public List<ListData_type_costs> getReminder(){
        ArrayList<ListData_type_costs> listData = new ArrayList<>();
        Cursor cursor = getAllEntries_Reminder();
        if(cursor.moveToLast()){
            do{
                int id =
cursor.getInt(cursor.getColumnIndex(DatabaseHelper.COLUMN_ID));
                String name =
cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_NAME));
                double money =
cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
                String date =
cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_DATE));
                String type =
cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_TYPE));
                listData.add(new ListData_type_costs(id, name, money, date,type));
            } while(cursor.moveToPrevious());
        }
```

```
        cursor.close();
        return listData;
    }
    // отримати граф Reminder
    private  long getCount_Reminder(){
        return
DatabaseUtils.queryNumEntries(database,DatabaseHelper.TABLE_REMINDER);
    }
    // пошук по id Reminder
    public ListData_type_costs getReminder(long id){
        ListData_type_costs listData = null;
        String query = String.format("SELECT * FROM %s WHERE
%s=?",DatabaseHelper.TABLE_REMINDER, DatabaseHelper.COLUMN_ID);
        Cursor cursor = database.rawQuery(query, new String[]{ String.valueOf(id)});
        if(cursor.moveToFirst()){
            String                                  name                                =
cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_NAME));
            double                                  money                               =
cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
            String                                  date                                =
cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_DATE));
            String                                  type                                =
cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_TYPE));
            listData = new ListData_type_costs(id,name,money,date,type);
        }
        cursor.close();
        return listData;
    }
    // внесення даних до таблиці Reminder
    public long insert_Reminder(ListData_type_costs listData){
```

```
        ContentValues cv = new ContentValues();
        cv.put(DatabaseHelper.COLUMN_NAME, listData.getName());
        cv.put(DatabaseHelper.COLUMN_MONEY, listData.getMoney());
        cv.put(DatabaseHelper.COLUMN_DATE, listData.getDate());
        cv.put(DatabaseHelper.COLUMN_TYPE, listData.getType());
        return database.insert(DatabaseHelper.TABLE_REMINDER, null, cv);
    }
    //видалення з таблиці Reminder
    public  long delete_Reminder(long incomId){
        String whereClause = "_id = ?";
        String[] whereArgs = new String[]{String.valueOf(incomId)};
        return database.delete(DatabaseHelper.TABLE_REMINDER, whereClause,
whereArgs);
    }
    //редагування елементів в таблиці Reminder
    public long update_Reminder(ListData_type_costs listData){
        String whereClause = DatabaseHelper.COLUMN_ID + "=" +
String.valueOf(listData.getId());
        ContentValues cv = new ContentValues();
        cv.put(DatabaseHelper.COLUMN_NAME, listData.getName());
        cv.put(DatabaseHelper.COLUMN_MONEY, listData.getMoney());
        cv.put(DatabaseHelper.COLUMN_DATE, listData.getDate());
        cv.put(DatabaseHelper.COLUMN_TYPE, listData.getType());
        return database.update(DatabaseHelper.TABLE_REMINDER, cv, whereClause,
null);
    }
    public List<ListData_type_costs> count_Reminder(){
        ArrayList<ListData_type_costs> listData = new ArrayList<>();
          SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("dd.MM.yyyy");
```

```
        Date today = Calendar.getInstance().getTime();
        String date_ = simpleDateFormat.format(today);
        String query = String.format("SELECT * FROM %s WHERE %s=?",DatabaseHelper.TABLE_REMINDER, DatabaseHelper.COLUMN_DATE);
        Cursor cursor = database.rawQuery(query, new String[]{ String.valueOf(date_)});
        if(cursor.moveToFirst()){
            do {
                long id = cursor.getLong(cursor.getColumnIndex(DatabaseHelper.COLUMN_ID));
                String name = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_NAME));
                double money = cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
                String date = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_DATE));
                String type = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_TYPE));
                listData.add(new ListData_type_costs(id, name, money, date, type));
            } while (cursor.moveToNext());
        }
    cursor.close();
    Log.d(TAG, String.valueOf(simpleDateFormat.format(today)));
    return listData;
  }
  //кількість нагадувань
  public int count_Reminder_now(){
    int i=0;
    ArrayList<ListData_type_costs> listData = new ArrayList<>();
    SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd.MM.yyyy");
```

```
        Date today = Calendar.getInstance().getTime();
        String date_ = simpleDateFormat.format(today);
        Cursor cursor = database.rawQuery("SELECT * FROM "+DatabaseHelper.TABLE_REMINDER+" WHERE "+ DatabaseHelper.COLUMN_DATE +" like '%"+ date_+"%'", null);
        if(cursor.moveToFirst()){
            do {
                long id = cursor.getLong(cursor.getColumnIndex(DatabaseHelper.COLUMN_ID));
                String name = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_NAME));
                double money = cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
                String date = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_DATE));
                String type = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_TYPE));
                listData.add(new ListData_type_costs(id, name, money, date,type));
                i++;
            } while (cursor.moveToNext());
        }
        cursor.close();
        Log.d(TAG, String.valueOf(simpleDateFormat.format(today)));
        return i;
    }
    public void not_count_Reminder() throws ParseException {
        Date date2 = Calendar.getInstance().getTime();
        SimpleDateFormat formatter1=new SimpleDateFormat("dd.MM.yyyy");
        ArrayList<ListData_type_costs> listData = new ArrayList<>();
        ListData_type_costs listData11 = null;
```

```
SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd.MM.yyyy");
Date today = Calendar.getInstance().getTime();
String date_ = simpleDateFormat.format(today);
Date date1 = formatter1.parse(date_);
String query = String.format("SELECT * FROM %s",DatabaseHelper.TABLE_REMINDER);
Cursor cursor = database.rawQuery(query, null);
if(cursor.moveToFirst()){
  do {
    date2 = formatter1.parse(cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_DATE)));

    if (date1.after(date2) && !(date1.equals(date2))) {
      Log.d(TAG, "true" + today + "||||" + date2);
      long id = cursor.getLong(cursor.getColumnIndex(DatabaseHelper.COLUMN_ID));
      String name = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_NAME));
      double money = cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
      String date = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_DATE));
      String type = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_TYPE));
      listData.add(new ListData_type_costs(id, name, money, date,type));
      listData11 = new ListData_type_costs(id,name,money,date,type);
      insert_Not_Completed_Reminder(listData11);
      delete_Reminder(id);
    }
```

```
        } while (cursor.moveToNext());
      }
      cursor.close();
    }
    //                                                                    Всего
денег_____

    //-------------------------------------------------------------------
//Not_Completed_Reminder
    // отримати всі записи Not_Completed_Reminder
    private Cursor getAllEntries_Not_Completed_Reminder(){
      String[] columns = new String[] {DatabaseHelper.COLUMN_ID,
DatabaseHelper.COLUMN_NAME, DatabaseHelper.COLUMN_MONEY,
DatabaseHelper.COLUMN_TYPE, DatabaseHelper.COLUMN_DATE};
      return
database.query(DatabaseHelper.TABLE_NOT_COMPLETED_REMINDER,
columns, null, null, null, null, null);
    }
  // отримати Not_Completed_Reminder
  public List<ListData_type_costs> getNot_Completed_Reminder(){
    ArrayList<ListData_type_costs> listData = new ArrayList<>();
    Cursor cursor = getAllEntries_Not_Completed_Reminder();
    if(cursor.moveToLast()){
      do{
        int                                id                                =
cursor.getInt(cursor.getColumnIndex(DatabaseHelper.COLUMN_ID));
        String                             name                              =
cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_NAME));
        double                             money                             =
cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
```

```
            String date = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_DATE));
            String type = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_TYPE));
            listData.add(new ListData_type_costs(id, name, money, date,type));
        } while(cursor.moveToPrevious());
    }
    cursor.close();
    return listData;
  }
  // отримати граф Not_Completed_Reminder
  private  long getCount_Not_Completed_Reminder(){
    return DatabaseUtils.queryNumEntries(database,DatabaseHelper.TABLE_NOT_COMPLETED_REMINDER);
  }
  // пошук по id Not_Completed_Reminder
  public ListData_type_costs getNot_Completed_Reminder(long id){
    ListData_type_costs listData = null;
    String query = String.format("SELECT * FROM %s WHERE %s=?",DatabaseHelper.TABLE_NOT_COMPLETED_REMINDER, DatabaseHelper.COLUMN_ID);
    Cursor cursor = database.rawQuery(query, new String[]{ String.valueOf(id)});
    if(cursor.moveToFirst()){
      String name = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_NAME));
      double money = cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
      String date = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_DATE));
      String type = cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_TYPE));
```

```
            listData = new ListData_type_costs(id,name,money,date,type);
        }
        cursor.close();
        return listData;
    }
    // внесення даних до таблиці Not_Completed_Reminder
    public long insert_Not_Completed_Reminder(ListData_type_costs income){
        ContentValues cv = new ContentValues();
        cv.put(DatabaseHelper.COLUMN_NAME, income.getName());
        cv.put(DatabaseHelper.COLUMN_MONEY, income.getMoney());
        cv.put(DatabaseHelper.COLUMN_DATE, income.getDate());
        cv.put(DatabaseHelper.COLUMN_TYPE, income.getType());
        return
database.insert(DatabaseHelper.TABLE_NOT_COMPLETED_REMINDER,      null,
cv);
    }
    //видалення з таблиці Not_Completed_Reminder
    public  long delete_Not_Completed_Reminder(long incomId){
        String whereClause = "_id = ?";
        String[] whereArgs = new String[]{String.valueOf(incomId)};
        return
database.delete(DatabaseHelper.TABLE_NOT_COMPLETED_REMINDER,
whereClause, whereArgs);
    }
    //редагування елементів в таблиці Not_Completed_Reminder
    public long update_Not_Completed_Reminder(ListData_type_costs listData){
        String whereClause = DatabaseHelper.COLUMN_ID + "=" +
String.valueOf(listData.getId());
        ContentValues cv = new ContentValues();
        cv.put(DatabaseHelper.COLUMN_NAME, listData.getName());
```

```
        cv.put(DatabaseHelper.COLUMN_MONEY, listData.getMoney());
        cv.put(DatabaseHelper.COLUMN_DATE, listData.getDate());
        cv.put(DatabaseHelper.COLUMN_TYPE, listData.getType());
        return
database.update(DatabaseHelper.TABLE_NOT_COMPLETED_REMINDER,      cv,
whereClause, null);
    }
//type table get
private Cursor getAllEntries_Type(){
    String[]      columns      =      new      String[]      {DatabaseHelper.COLUMN_ID,
DatabaseHelper.COLUMN_NAME, DatabaseHelper.COLUMN_TYPE};
    return database.query(DatabaseHelper.TABLE_TYPE_COSTS, columns, null, null,
null, null, null);
}
    // отримати type
    public List<ListData_type> getType(){
        ArrayList<ListData_type> listData = new ArrayList<>();
        Cursor cursor = getAllEntries_Income();
        if(cursor.moveToLast()){
            do{
                int                                    id                                    =
cursor.getInt(cursor.getColumnIndex(DatabaseHelper.COLUMN_ID));
                String                                 name                                  =
cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_NAME));
                String                                 color                                 =
cursor.getString(cursor.getColumnIndex(DatabaseHelper.COLUMN_DATE));
                listData.add(new ListData_type(id, name, color));
            } while(cursor.moveToPrevious());
        }
        cursor.close();
        return listData;
```

```
    }

    // внесення даних до таблиці type
    public long insert_type(ListData_type income){
        ContentValues cv = new ContentValues();
        cv.put(DatabaseHelper.COLUMN_NAME, income.getName());
        cv.put(DatabaseHelper.COLUMN_COLOR, income.getColor());
        return database.insert(DatabaseHelper.TABLE_TYPE_COSTS, null, cv);
    }
    //видалення з таблиці type
    public  long delete_type(long incomId){
        String whereClause = "_id = ?";
        String[] whereArgs = new String[]{String.valueOf(incomId)};
        return database.delete(DatabaseHelper.TABLE_TYPE_COSTS, whereClause,
whereArgs);
    }
    //редагування елементів в таблиці type
    public long update_type(ListData_type listData){
        String whereClause = DatabaseHelper.COLUMN_ID + "=" +
String.valueOf(listData.getId());
        ContentValues cv = new ContentValues();
        cv.put(DatabaseHelper.COLUMN_NAME, listData.getName());
        cv.put(DatabaseHelper.COLUMN_COLOR, listData.getColor());
        return database.update(DatabaseHelper.TABLE_TYPE_COSTS, cv,
whereClause, null);
    }
    // пошук по type sum type
    public double getType_type_sum(String type){
        double sum = 0;
        open();
```

```
    Cursor cursor = database.rawQuery("SELECT * FROM "+DatabaseHelper.TABLE_COST+" WHERE "+ DatabaseHelper.COLUMN_TYPE +" like '%"+ type +"%'", null);
    if(cursor.moveToLast()){
      do{
        double money = cursor.getDouble(cursor.getColumnIndex(DatabaseHelper.COLUMN_MONEY));
        sum = sum + money;
      } while(cursor.moveToPrevious());
    }
    cursor.close();
    close();
    return sum;
  }}
```