МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Мобільний додаток для ведення особистого бюджету»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студентка групи ІТ-72 Ніколаєнко Вікторія Василівна

Кваліфікаційна робота
бакалавразахищена на засіданні
ЕК
з оцінкою _____«__»_____2021 р.

Науковий керівник _____ к. т. н., Антипенко В.П.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Голова комісії _____ _____
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Суми-2021

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«___»_____2021 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Ніколаєнко Вікторія Василівна

1 Тема роботи *Мобільний додаток для ведення особистого бюджету*

керівник роботи *Антипенко Вікторія Петрівна, к.т.н.*

затверджені наказом по університету від «_»_____2021 р. №_____

2 Строк подання студентом роботи «_»_____2021 р.
3 Вхідні дані до роботи технічне завдання на розробку мобільного додатку для ведення особистого бюджету

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області застосування мобільних додатків для ведення особистого бюджету, моделювання та проектування мобільного додатку, розробка дизайну, реалізація мобільного додатку.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Презентація слайди

6 Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7 Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметні області	04.02.21 – 03.04.21	
2	Моделювання та проектування	05.04.21 – 13.04.21	
3	Створення прототипу та дизайну	13.04.21 - 25.04.21	
4	Розробка додатку	26.04.21 – 26.05.21	
5	Тестування	27.05.21 – 06.06.21	
6	Оформлення пояснювальної записки	26.04.21 06.06.21	

Студент _____ Ніколаєнко В.В.
(підпис)

Керівник роботи _____ к.т.н., Антипенко В.П.
(підпис)

РЕФЕРАТ

Тема роботи «Мобільний додаток для ведення особистого бюджету».

Пояснювальна записка складається зі вступу, трьох основних розділів, висновку, списку використаних джерел із 15 найменувань та чотирьох додатків. Загальний обсяг пояснювальної записки складає 89 сторінок, в тому числі 53 сторінок основного тексту, 2 сторінки списку використаних джерел, 36 сторінок додатків.

У першому розділі розглянуто актуальність питання ведення особистого бюджету та проведено аналіз додатків-аналогів. Також поставлено мету та визначено задачі.

У другому розділі виконано моделювання варіантів використання, побудовано Use Case діаграму, діаграму IDEF0 та діаграму декомпозиції, спроектовано модель бази даних.

У третьому розділі детально описано процес практичної реалізації проекту, розробки дизайну, програмної реалізації. Наведений приклад використання мобільного додатку.

Результатом кваліфікаційної роботи бакалавра є розроблений на основі операційної системи Android мобільний додаток для ведення особистого бюджету.

Ключові слова: МОБІЛЬНИЙ ДОДАТОК, БЮДЖЕТ, МЕТОДИ НАКОПИЧЕННЯ, ДОХОДИ, ВИТРАТИ.

ЗМІСТ

ВСТУП

У наш час практично неможливо уявити життя сучасної людини без використання різноманітних гаджетів. Усе більш популярним та необхідним стає користування мобільними додатками, попит на які з кожним днем тільки зростає. Сьогодні за допомогою смартфону можна купувати, продавати, навчатися, розважатися, отримувати та надавати послуги тощо. Тому, можна зробити висновок, що створення мобільних додатків є досить актуальним та популярним питанням у наш час.

Мобільні додатки значно полегшують виконання багатьох завдань за рахунок автоматизації деяких процесів. У свою чергу це вирішує значну низку проблем користувачів. Наприклад, кожна сучасна людина бажає бути відповідальною щодо її фінансового становища. Це передбачає постійне ведення обліку особистого бюджету – те, до чого прагне велика кількість людей, але виходить лише у деяких. Причинами цього є різні фактори від несистематичності, відсутності конкретної мети, записів у різних місцях до відсутності такої звички взагалі. При використанні мобільних додатків можна зіткнутися з такими проблемами, як незрозрозумілий та/або незручний інтерфейс, недостатня кількість корисної інформації або можливостей тощо.

У період пандемії питання фінансів стає все більш актуальним. Отже, розроблення мобільного додатку для ведення особистого бюджету є досить нагальним у наш час. Він має надавати можливість користувачам вести облік особистих фінансів, накопичувати, ставити цілі, отримати інформацію щодо методів накопичення. Основною задачею при створенні мобільного додатку є привабливий та зручний інтерфейс, який забезпечить користувача необхідним функціоналом.

Отже, метою дипломної роботи є розробка мобільного додатку для ведення особистого бюджету.

Для досягнення поставленої мети необхідно виконати наступні задачі:

- провести аналіз предметної області;
- визначити актуальність;
- провести аналіз продуктів-аналогів;
- виконати моделювання та проектування мобільного додатку;
- створити прототип та дизайн мобільного додатку;
- розробити та протестувати мобільний додаток.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність питання ведення особистого бюджету

Не є секретом, що планування власного бюджету є запорукою фінансового благополуччя. Завдяки правильному веденню домашнього бюджету людина розсудливо розподіляє кошти та має уявлення для чого вони призначені, що в свою чергу не дозволить витратити їх на непотрібні речі [1].

На даний момент існують різні методи накопичення коштів. Найвідомішими серед них є такі:

– «20/80»: спочатку потрібно погасити всі позики, потім п’яту частину доходу треба відкласти, а суму, яка залишилась, важливо раціонально розподілити для ефективного використання;

– «Десятина»: суть методу полягає у відкладанні десятої частини від зароблених коштів;

– «Чотири конверта»: спочатку виділяємо заощадження – віднімаємо 10-20% від суми доходу, потім гроші, які залишилися, без урахування основних витрат, розподіляємо на чотири конверти, призначених для щотижневих повсякденних витрат [2].

Практично щодня стають загальнодоступними нові стратегії розпорядження власним бюджетом та накопичення коштів. Саме тому актуальним є розроблення мобільного додатку, в якому будуть реалізовані сучасні методи, які нещодавно завоювали популярність серед суспільства.

1.2 Аналіз програмних продуктів-аналогів

Огляд існуючих програмних продуктів проводимо на магазині мобільних додатків Google Play Store. Знаходимо безліч додатків-аналогів через пошуковий рядок, які представлено на рисунку 1.1.

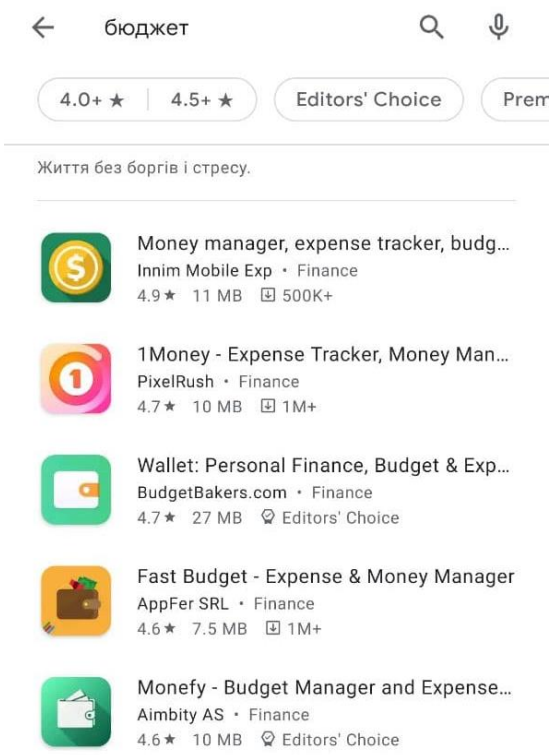

Рисунок 1.1 – Результат пошуку

Першим додатком у рейтингу є «Money manager» (рис. 1.2). Мобільний додаток дає можливість користувачу вести власні доходи та витрати, розділяє їх на категорії (рис. 1.3) та надає можливість створювати свої. Із мінусів слід зазначити незручний інтерфейс, відкриваючи перший раз додаток, інтуїтивно не зрозуміло як ним користуватися.

Рисунок 1.2 – Початковий екран мобільного додатку «Money manager»

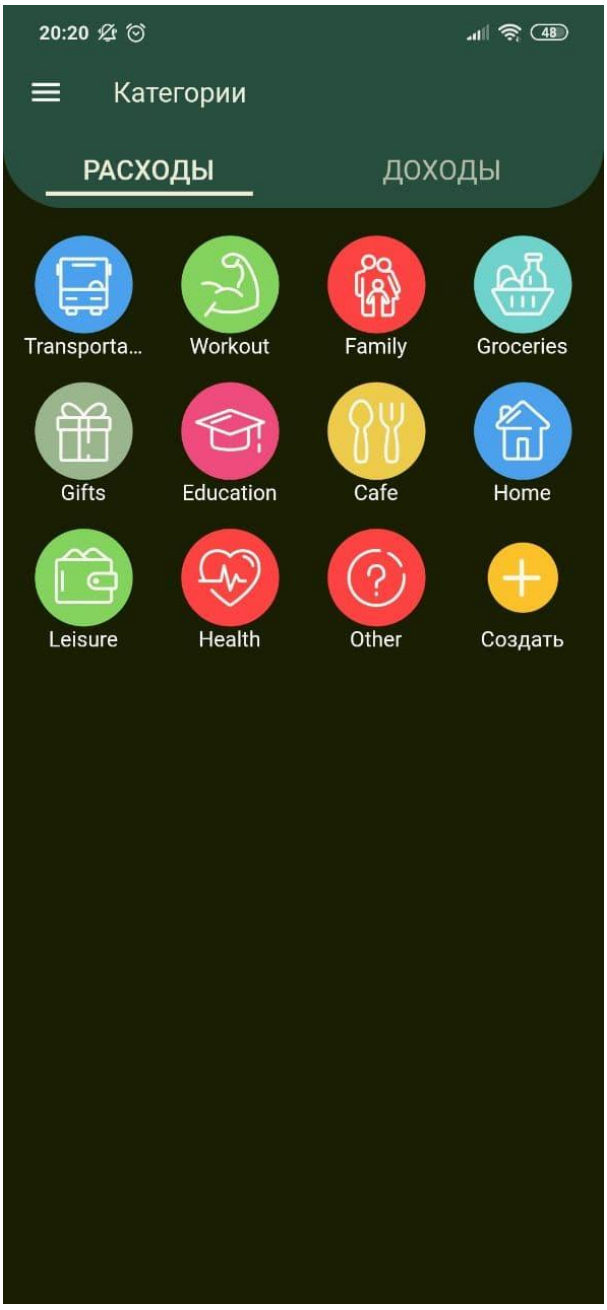

Рисунок 1.3 – Категорії

Наступним у рейтингу є додаток «1Money». Вигляд головного екрану додатку представлено на рисунку1.4. Окрім ведення бюджету (рис. 1.5), він надає можливість записувати борги, ставити цілі на витрати та переглядати статистику.

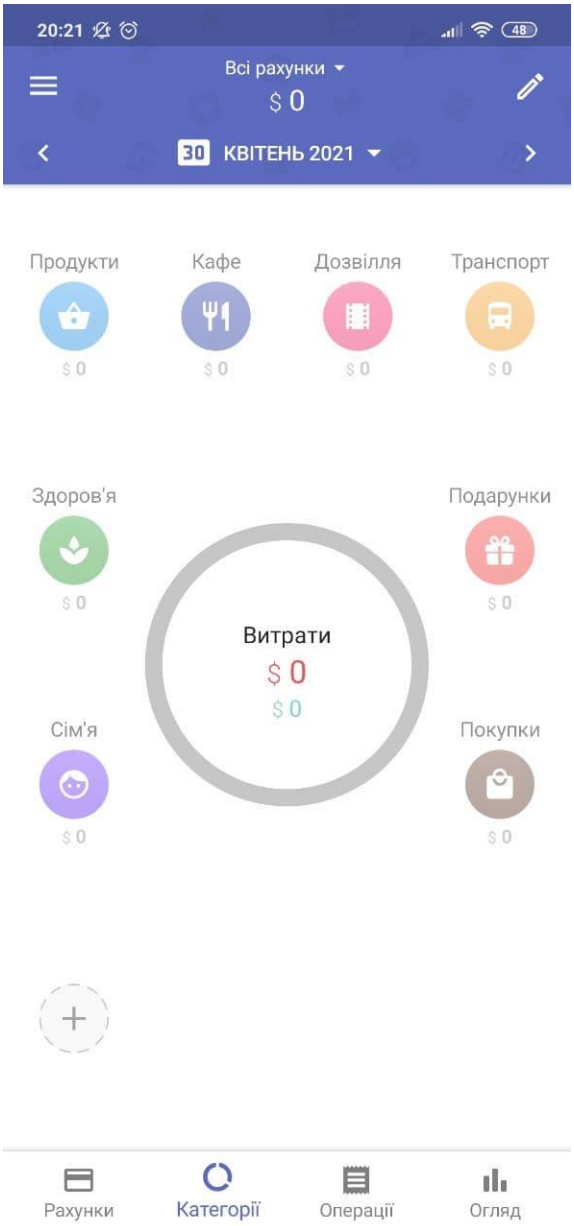

Рисунок 1.4 – Початковий екран додаток «1Money»

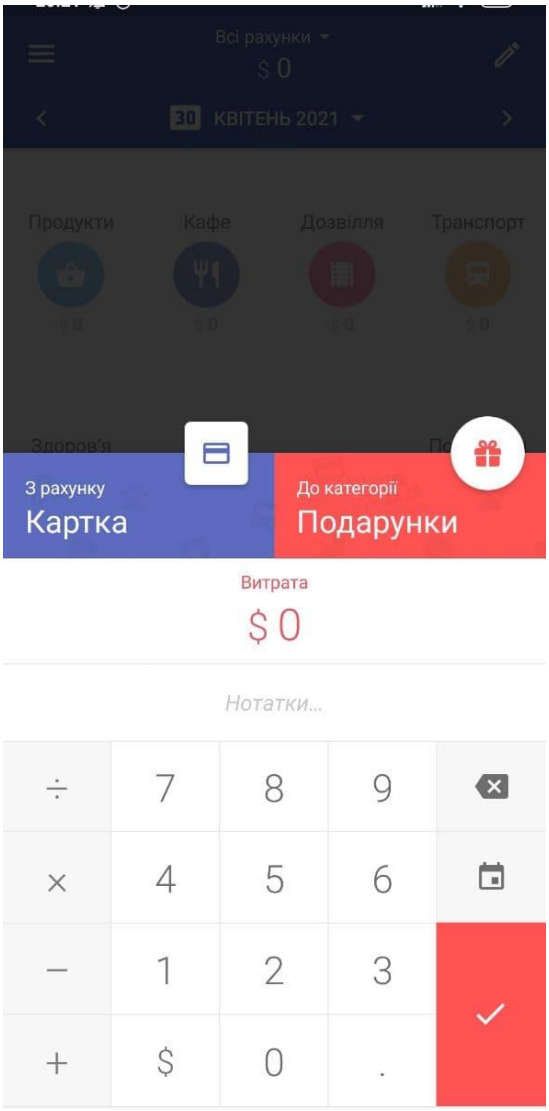

Рисунок 1.5 – Введення витрат

Далі розглянемо додаток «Monefy». Він має зрозумілий інтерфейс, функцію обліку витрат та прибутків із поділом на категорії. Майже весь функціонал додатку представлений на головному екрані (рис. 1.6.).

Рисунок 1.6 – Початковий екран додатку «Monefy»

Отже, на основі проведеного аналізу додатків-аналогів було визначего їх переваги та недоліки. Його результати приведено в таблиці 1.1.

Таблиця 1.1 – Порівняльна таблиця додатків-аналогів розроблюваного продукту

Додаток	Інтерфейс	Ведення доходів та витрат	Цілі	Борги	Інтерактивний метод накопичення	Статистика
Money manager	-	+	-	-	-	+
1Money	+	+	+	+	-	+
Monefy	+	+	-	-	-	+

Проаналізувавши дані з таблиці 1.1 можна зробити висновок, що не один з розглянутих додатків, які мають і звичайний базовий функціонал тематичного програмного продукту, не може поєднати зрозумілий інтерфейс та методи накопичення. Тому, було прийнято рішення розробити власний модільний додаток ведення особистого бюджету, в якому будуть подолані вищезазначені недоліки.

1.3 Постановка задачі

Метою даної роботи є розроблення мобільного додатку для ведення особистого бюджету на базі операційної системи Android.

Для досягнення поставленої мети треба виконати наступні задачі:

- провести аналіз предметної області та визначити актуальність роботи;
- провести аналіз продуктів-аналогів і визначити їх переваги та недоліки;
- виконати моделювання та проектування даного мобільного додатку;
- створити прототип та дизайн мобільного додатку;

– розробити та протестувати мобільний додаток.

Результатом виконання проекту є мобільний додаток, який повинен надавати можливість користувачам записувати особисті витрати та доходи, ставити грошові цілі та накопичувати гроші. Його реалізацію буде виконуватися за допомогою середовища розробки Android Studio з використанням мови програмування Kotlin і мови розмітки XML. Інформація надана у мобільному додатку буде зберігатися у базі даних реалізованій засобами системи управління базами даних SQLite.

2 ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ

2.2 Моделювання IDEF діаграм

IDEF представляє сукупність стандартизованих методів і сім'ю графічних мов, що використовуються для інформаційного моделювання в галузі програмного забезпечення та бізнес процесів. Метою цих методів є підвищення продуктивності виробництва з використанням інформаційних технологій та моделювання [3].

У моделюванні IDEF0 функція завжди відноситься до дії, процесу або перетворення. Роль функції полягає в тому, щоб перетворити деякі вхідні дані до вихідних із використанням деяких ресурсів при певних обмеженнях або правилах [4].

На рисунку 2.1 показано моделювання мобільного додатку для ведення особистого бюджету функції верхнього рівня, що позначається як A-0. У синтаксисі IDEF0 функція, яка представлена блоком, містить стрілки входу, виходу, управління та механізму, які розташовані справа, зліва, зверху та знизу блоку відповідно [5].

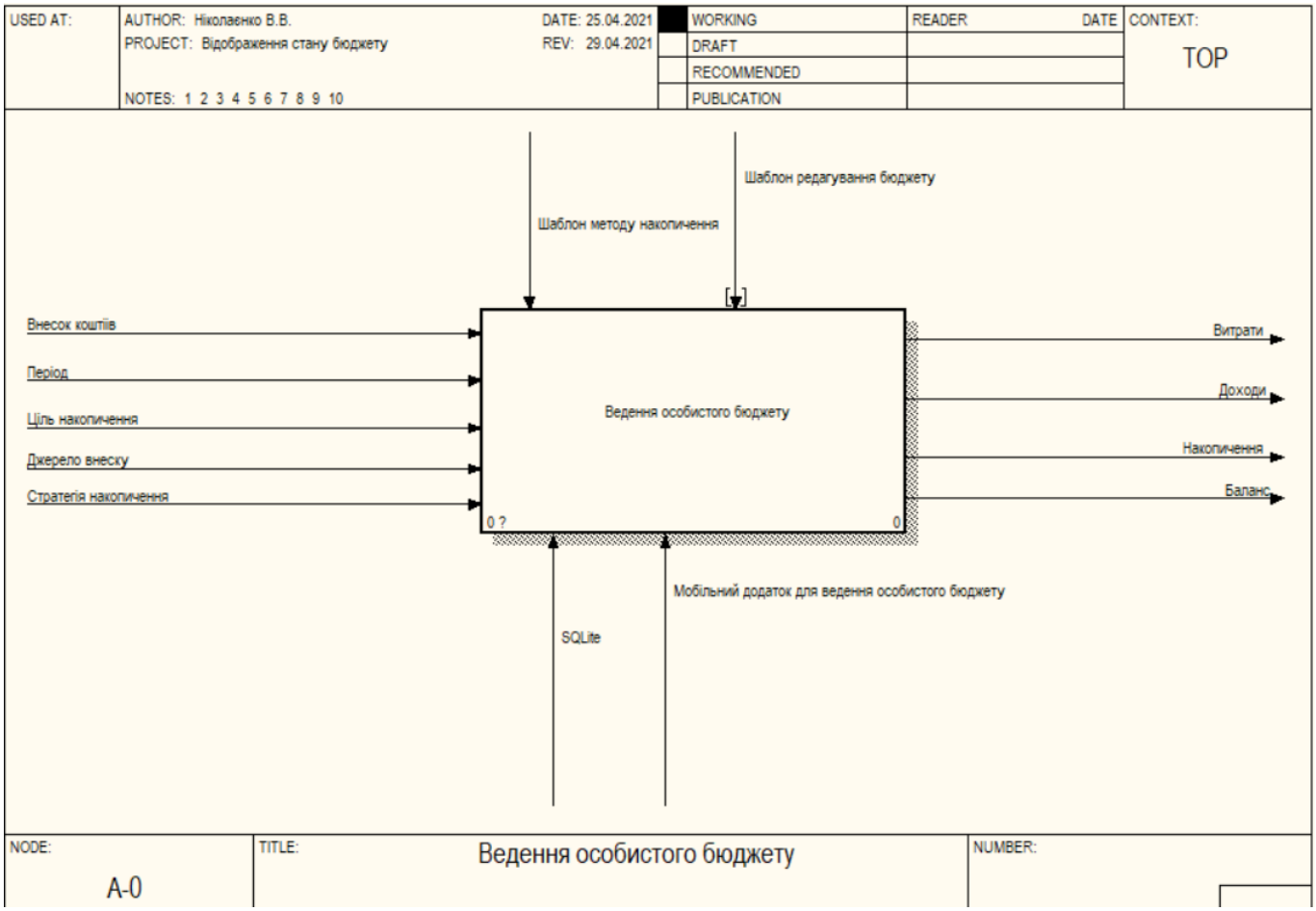

Рисунок 2.1 – Діаграма IDEF0

Далі функцію верхнього рівня розкладаємо на дочірні функції (рис 2.2).

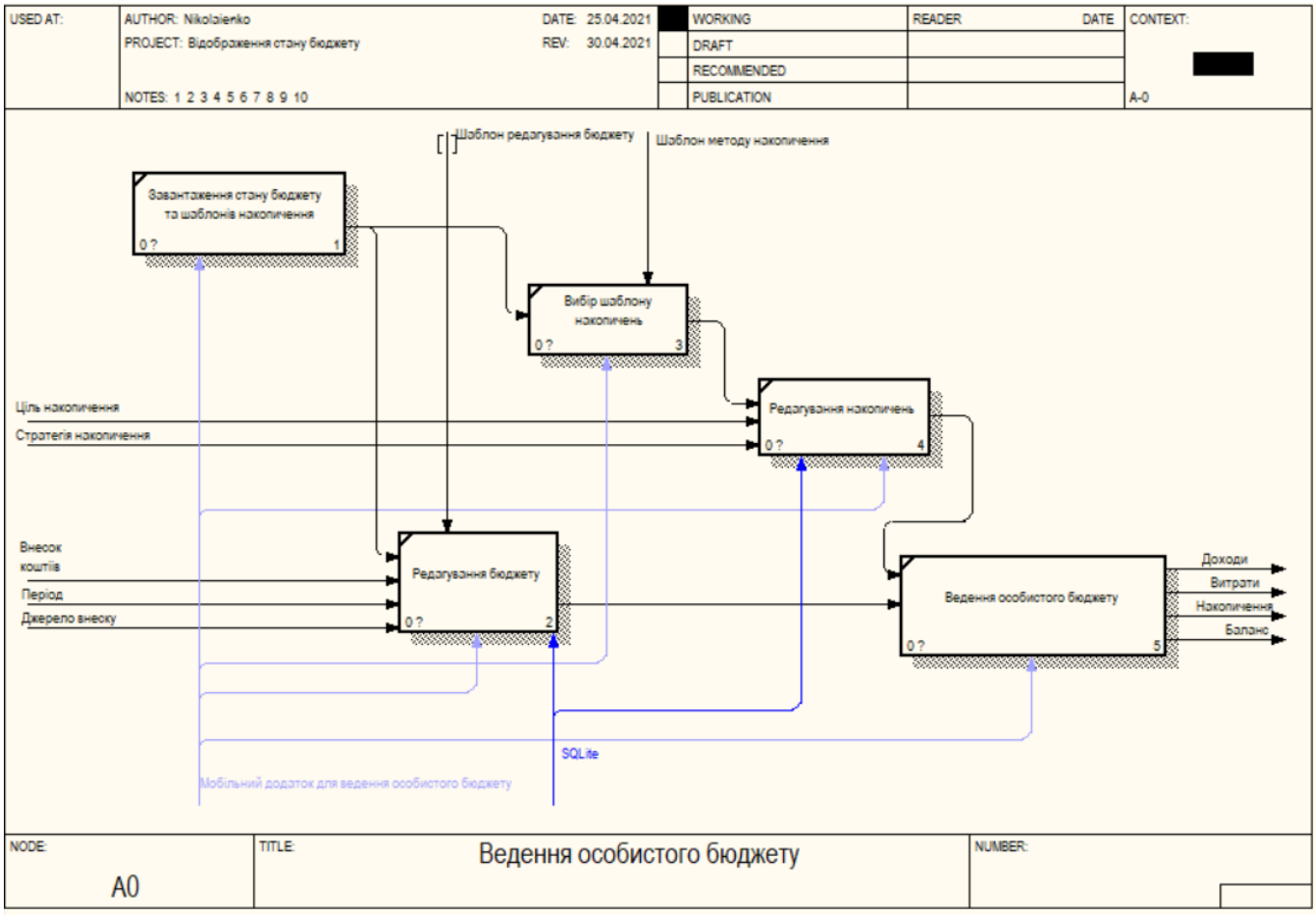

Рисунок 2.2 – Діаграма декомпозиції

2.2 Моделювання варіантів використання

Здійснимо моделювання варіантів використання мобільного додатку для ведення особистого бюджету. Будемо використовувати стандартну мову для опису та візуалізації UML [6].

Діаграма варіантів використання відображає програмний продукт, сценарії його застосування та актора (рис. 2.1). Актор – користувач, який використовує функції даного мобільного додатка [7].

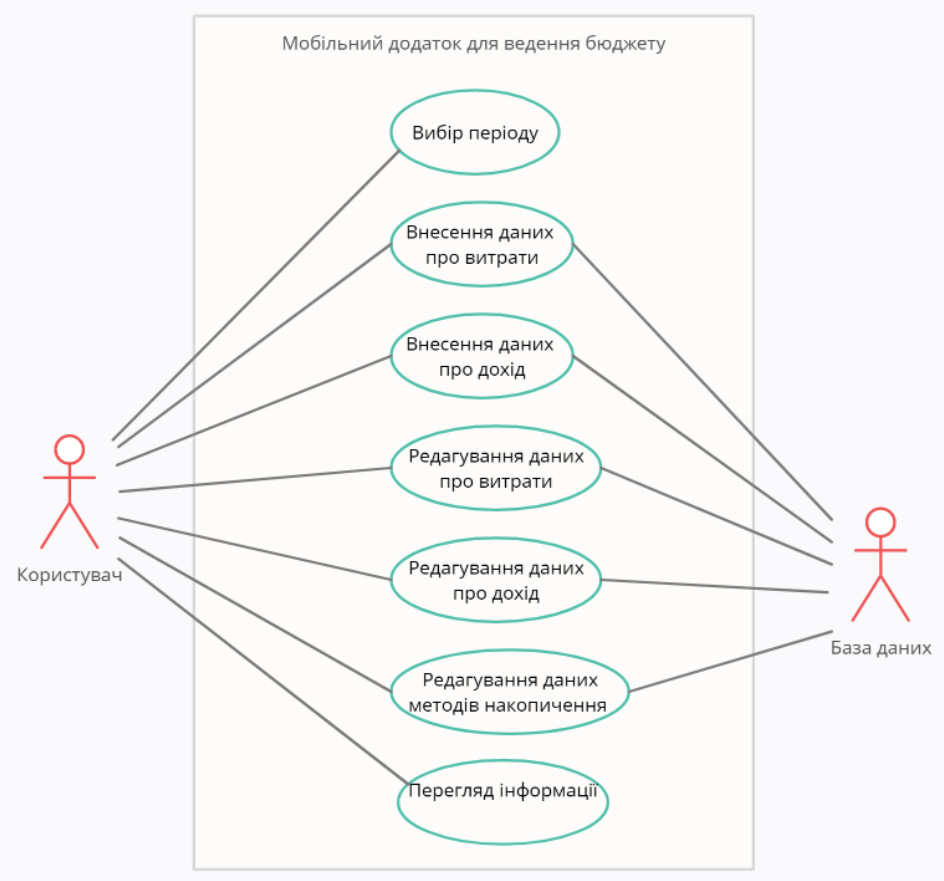

Рисунок 2.3 – Use Case діаграма

Конкретною ціллю діаграми варіантів використання є зібрати системні вимоги та виконавців [8]. Отже, можна виділити наступні варіанти використання мобільного додатку для ведення особистого бюджету для користувача:

– Внесення даних про дохід;
– Внесення даних про витрати;
– Редагування даних про дохід;
– Редагування даних про витрати;
– Редагування даних методів накопичення;
– Перегляд інформації;
– Вибір періоду.

2.3 Проектування моделі бази даних

Моделі даних – це система представлення для структурного моделювання фізичних систем. Застосування моделей даних приводить до створення баз даних [9-10].

Отже, розглянемо таблиці бази даних мобільного додатку для ведення особистого бюджету (рис. 2.4-2.6).

На рисунку 2.4 зображена таблиця транзакцій, яка включає такі атрибути, як розмір внеску коштів, дату внеску та джерело внеску.

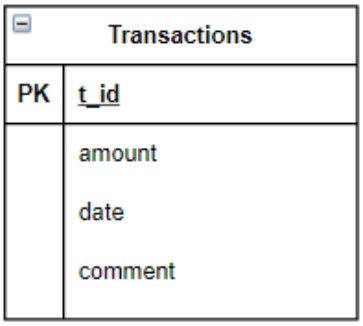

Рисунок 2.4 – Таблиця «Транзакції»

На рисунку 2.5 зображена таблиця цілей та включає такі атрибути, як назву самої цілі, кінцеву суму та накопичені кошти.

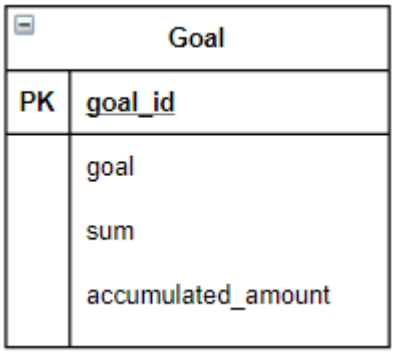

Рисунок 2.5 – Таблиця «Ціль»

На рисунку 2.6 зображена таблиця числа Гаусса, яка включає такі атрибути, як значення та статус числа.

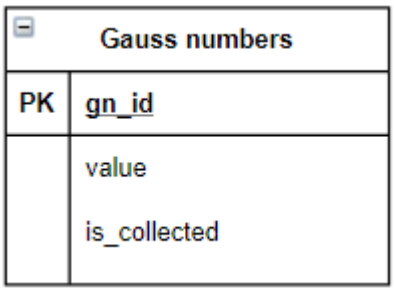

Рисунок 2.6 – Таблиця «Число Гаусса»

3 РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ

3.1 Розробка дизайну мобільного додатку

Реалізацію мобільного додатку починаємо з розробки дизайну на онлайн-сервісі розробки інтерфейсів та прототипування Figma (рис. 3.1). Обираючи гамму, увагу було звернуто на кольори 2021 року по версії інституту кольору Pantone – «Ultimate Grey» та «Illuminating» [11]. Отже, головним кольоровим мотивом обрано жовті та сірі відтінки, а також синій, який підкреслює контрастність.

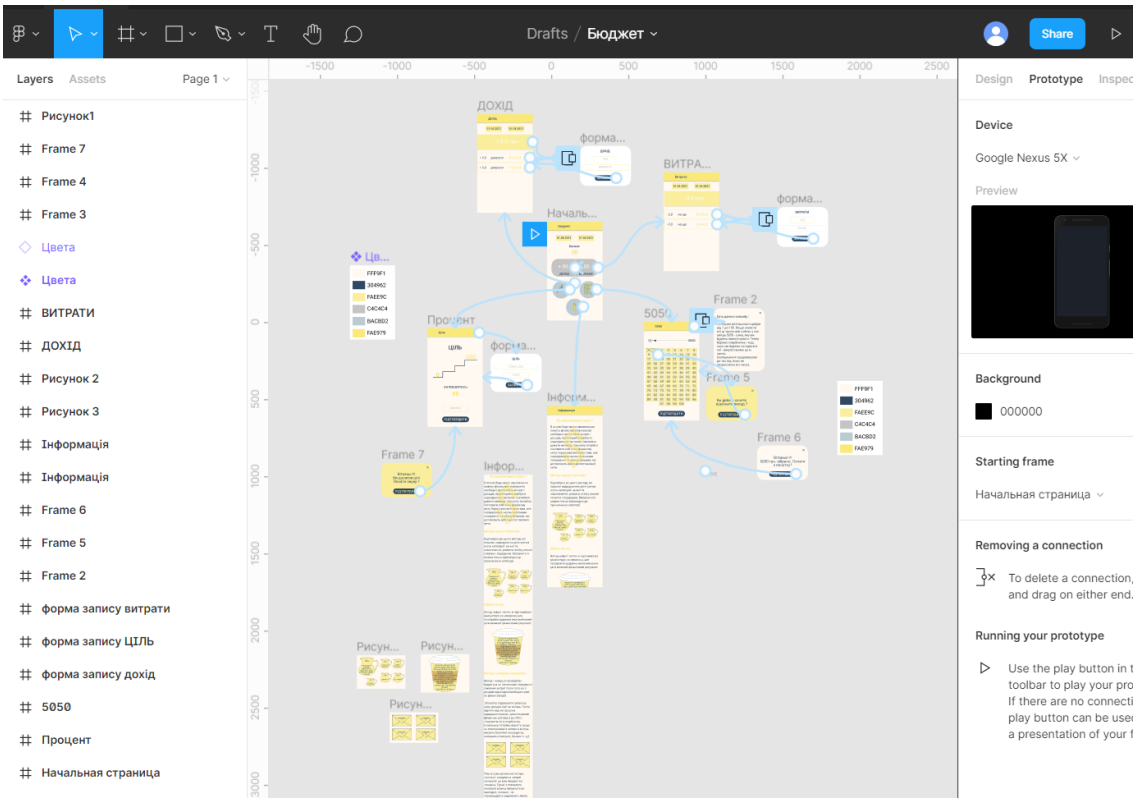

Рисунок 3.1 – Демонстрація сторінки Figma у режимі прототипу

На початковому екрані мобільного додатку розташовуємо кнопки для вибору періоду, значення балансу, доходів та витрат, за обраний період, кнопки переходу на екрани «Дохід», «Витрати», «Ціль», «Інформація», «5050» (рис. 3.2).

Рисунок 3.2 – Дизайн головного екрану

Екран «Дохід» також містить вибір періоду та стан доходу, а також можливість додавання та редагування доходів, із використанням діалогового вікна (рис. 3.3).

Екран «Витрати» створюємо аналогічно (рис. 3.4).

Екран «Ціль» – це реалізація методу накопичення коштів, який включає в себе формулювання конкретної цілі, кінцевого результату та проміжні цифри (рис. 3.5).

Екран «Інформація» містить текст та рисунки, які його підкріплюють (рис. 3.6).

Екран «5050» – це реалізація методу накопичення коштів, який містить кнопки з цифрами від 1 до 100, повзунок, що показує прогрес та кнопку підтвердження дії й інструкцію та діалогові вікна з попередженнями (рис. 3.7).

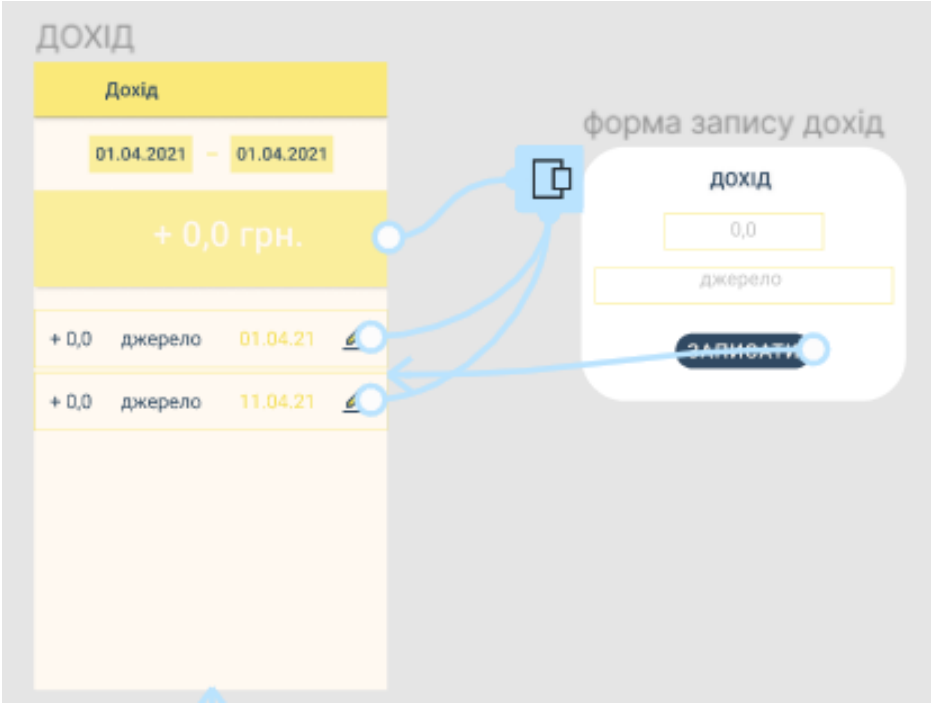

Рисунок 3.3 – Дизайн екрану «Дохід»

Рисунок 3.4 – Дизайн екрану «Витрати»

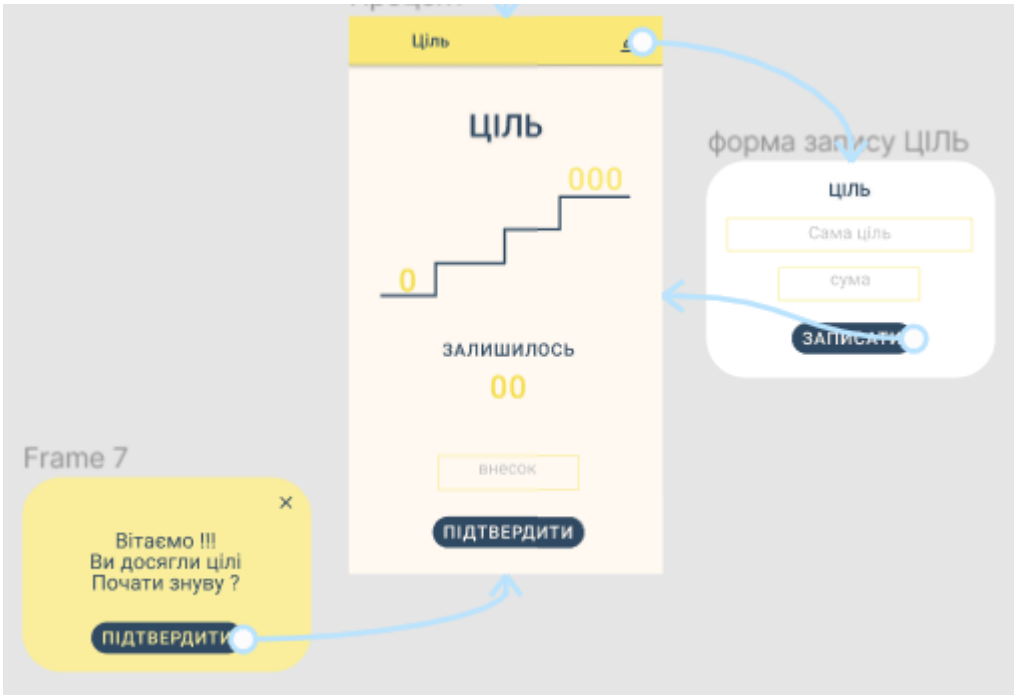

Рисунок 3.5 – Дизайн екрану «Цілі»

Рисунок 3.6 – Дизайн екрану «Інформація»

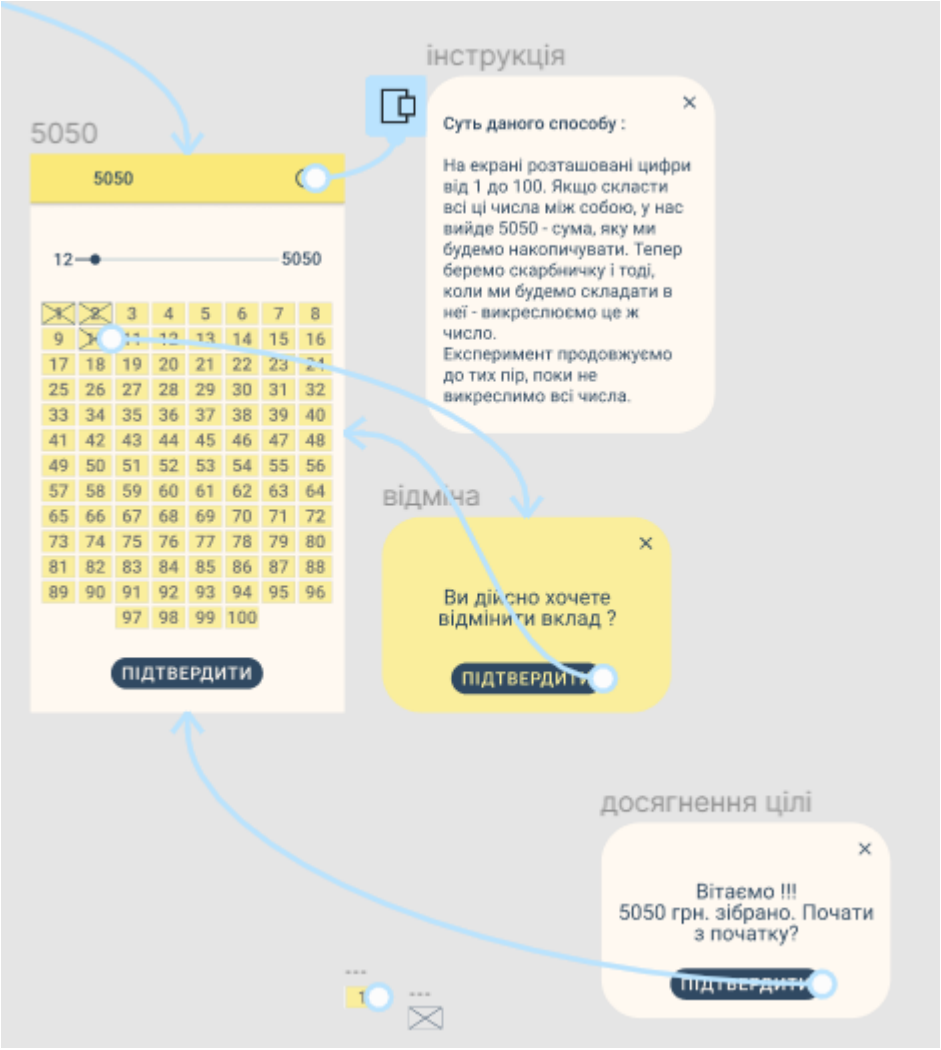

Рисунок 3.7 – Дизайн екрану «5050»

3.2 Програмна реалізація

Програмну реалізацію мобільного додатку виконуємо мовою програмування Kotlin в середовищі Android Studio. Для навігації між фрагментами додатку використовуємо Navigation component (рис. 3. 8) [12].

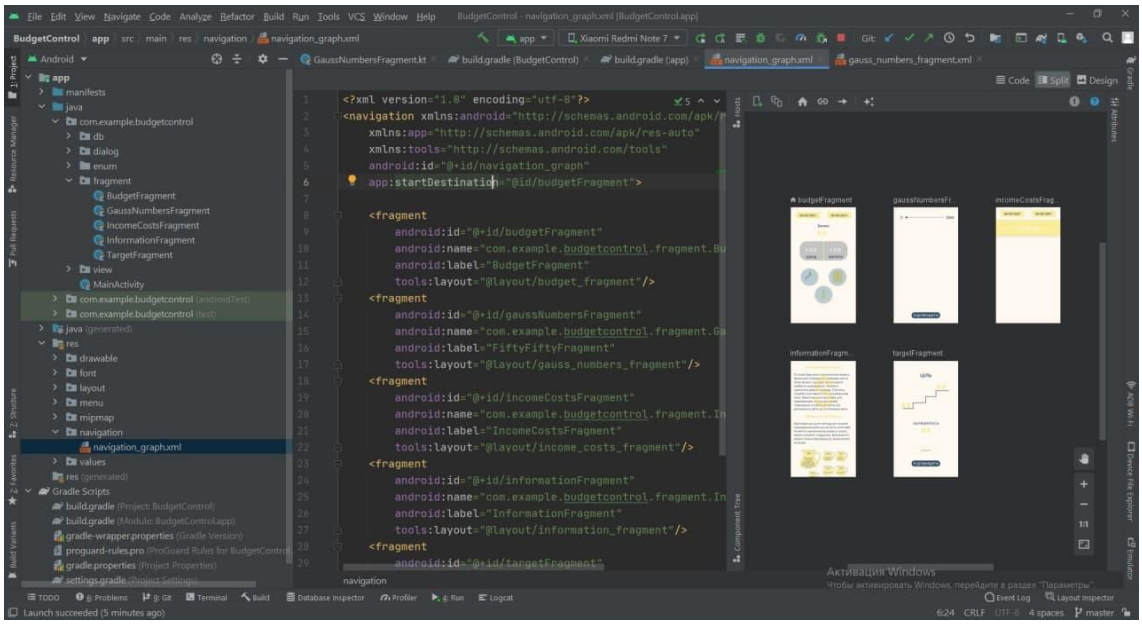

Рисунок 3.8 – Файл опису графу навігації

Для роботи з базою даних використовуємо бібліотеку Room. Для виконання запитів у базу даних асинхронно використовуємо kotlin Coroutines [13]. Таблиці у базі даних генеруються за описом Entity класів [14]. Для зберігання даних розроблюємо три таблиці (рис. 3.9-3.11).

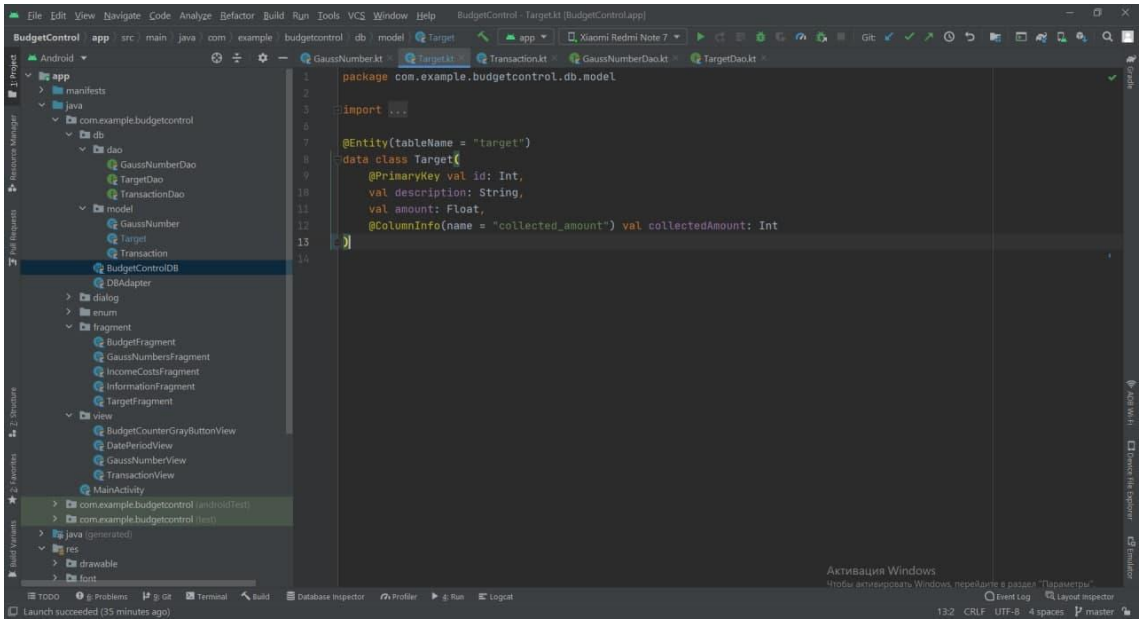

Рисунок 3.9 – Entity клас Target

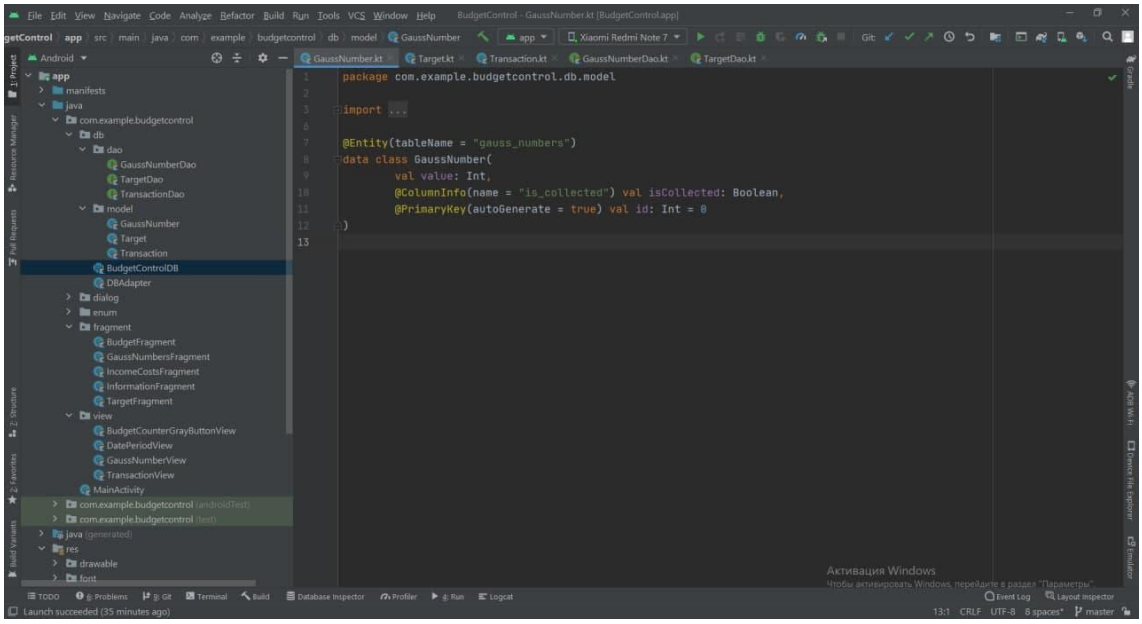

Рисунок 3.10 – Entity клас GaussNamber

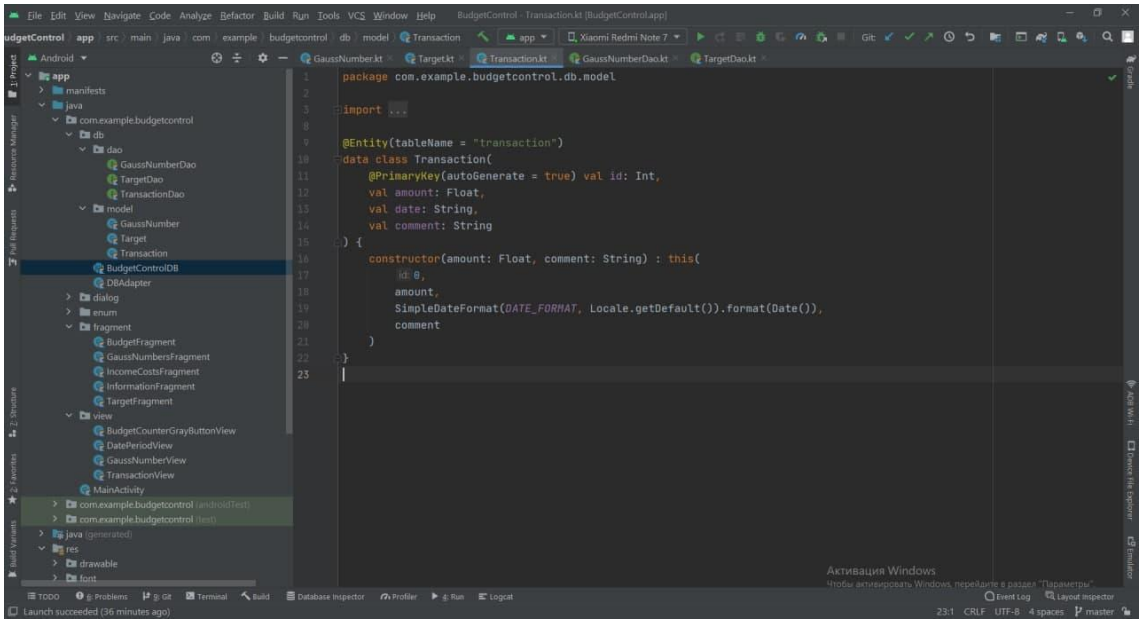

Рисунок 3.11 – Entity клас Transaction

Запис та зчитування даних з бази даних виконується за допомогою Dao інтерфейсів, в яких описані методи запитів до бази даних (рис. 3.12-3.14) [15].

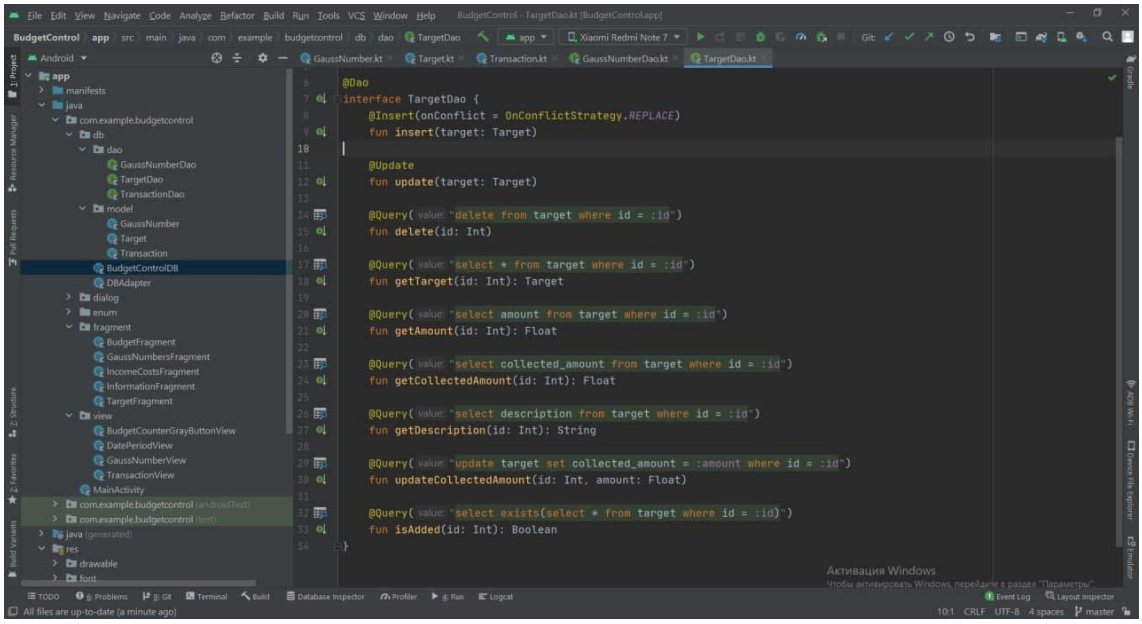

Рисунок 3.12 – Dao інтерфейс Target

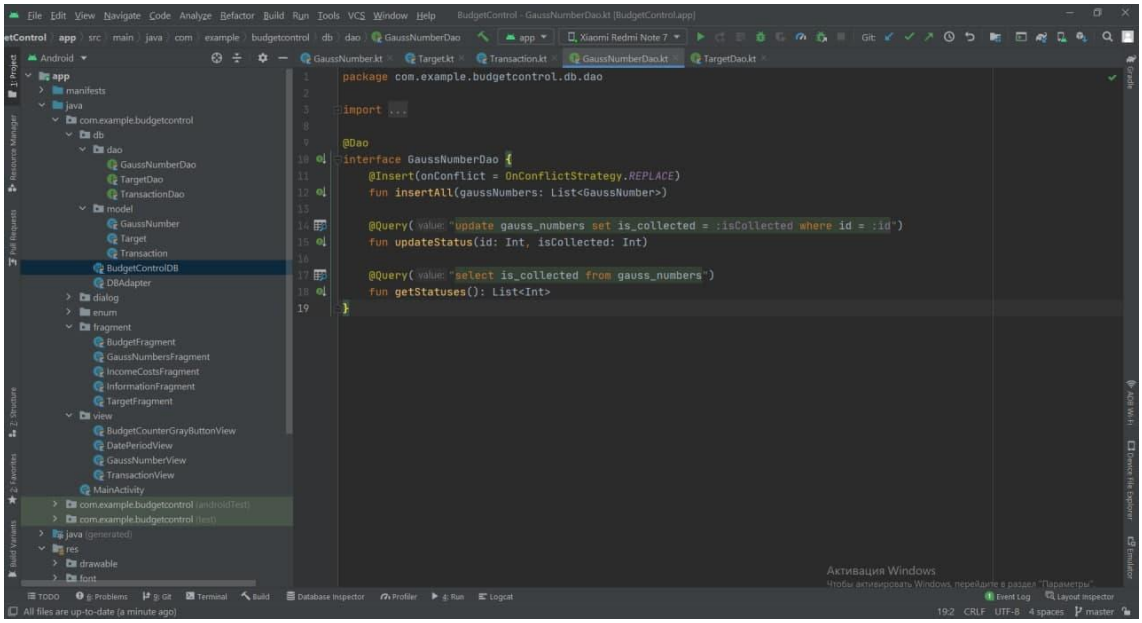

Рисунок 3.13 – Dao інтерфейс GaussNamber

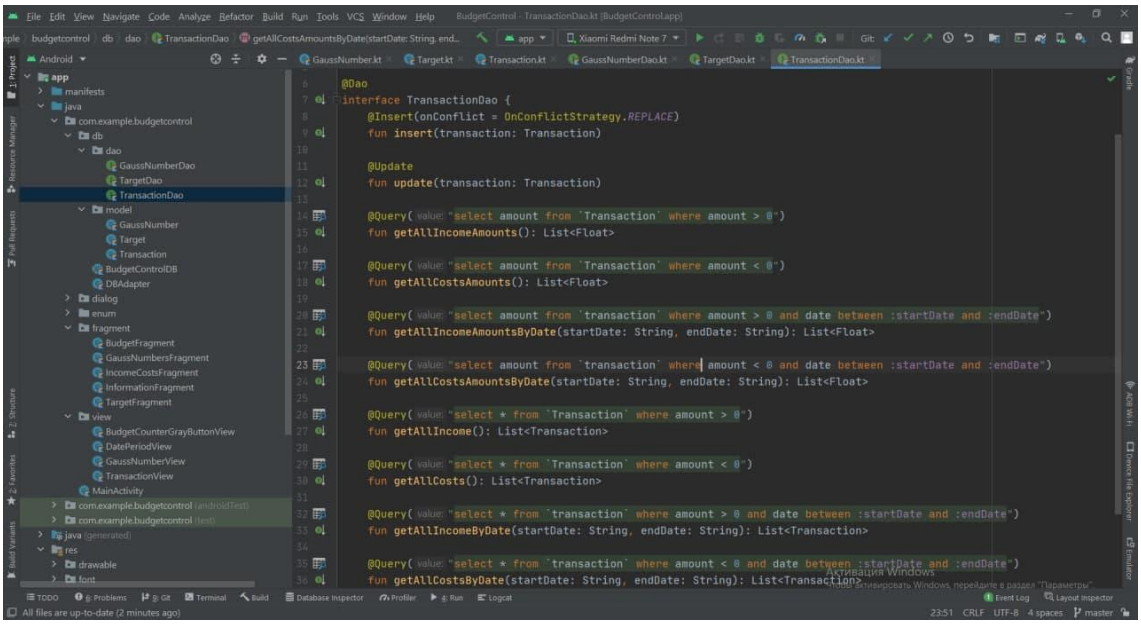

Рисунок 3.14 – Dao інтерфейс Target

Інтерфейс додатку розроблюємо за допомогою мови розмітки XML (рис. 3.15).

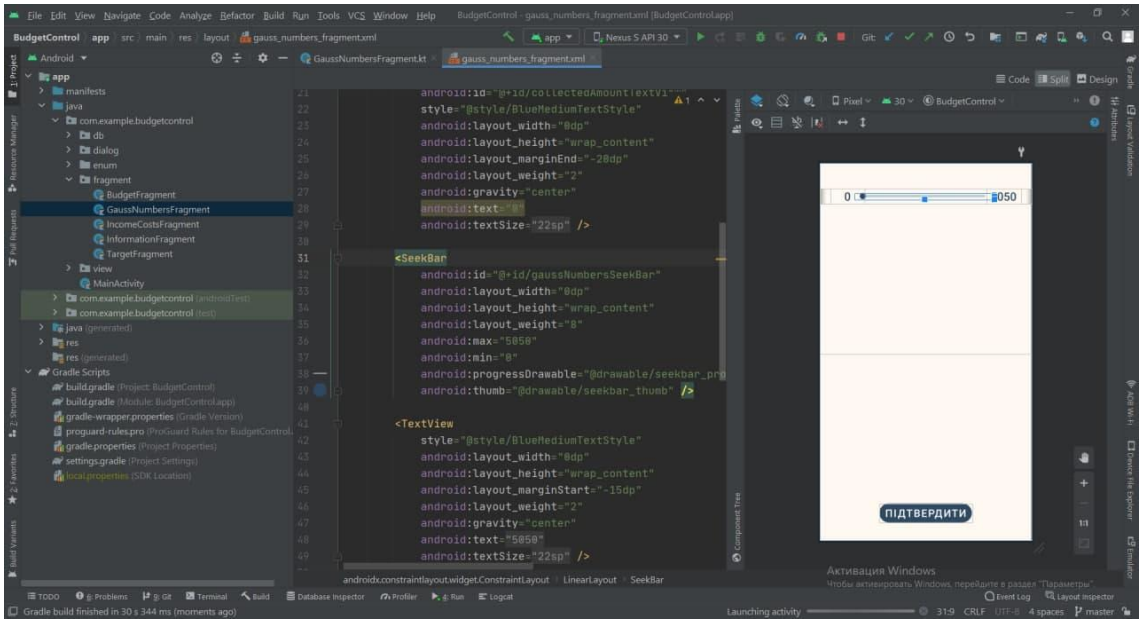

Рисунок 3.15 – Розмітка екрану «5050»

3.3 Використання додатку

Завантажуючи мобільний додаток ведення особистого бюджету, користувач потрапляє на початкову сторінку з головним меню (рис. 3.16). Тут він має можливість обрати період (при натисканні на дату, за замовчуванням перша – за місяць до поточної дати, друга – поточна дата) (рис. 3.17), за який буде показано баланс, доходи та витрати; перейти на екрани «Дохід», «Витрати», «Ціль», «Інформація» або «5050».

Рисунок 3.16 – Початковий екран

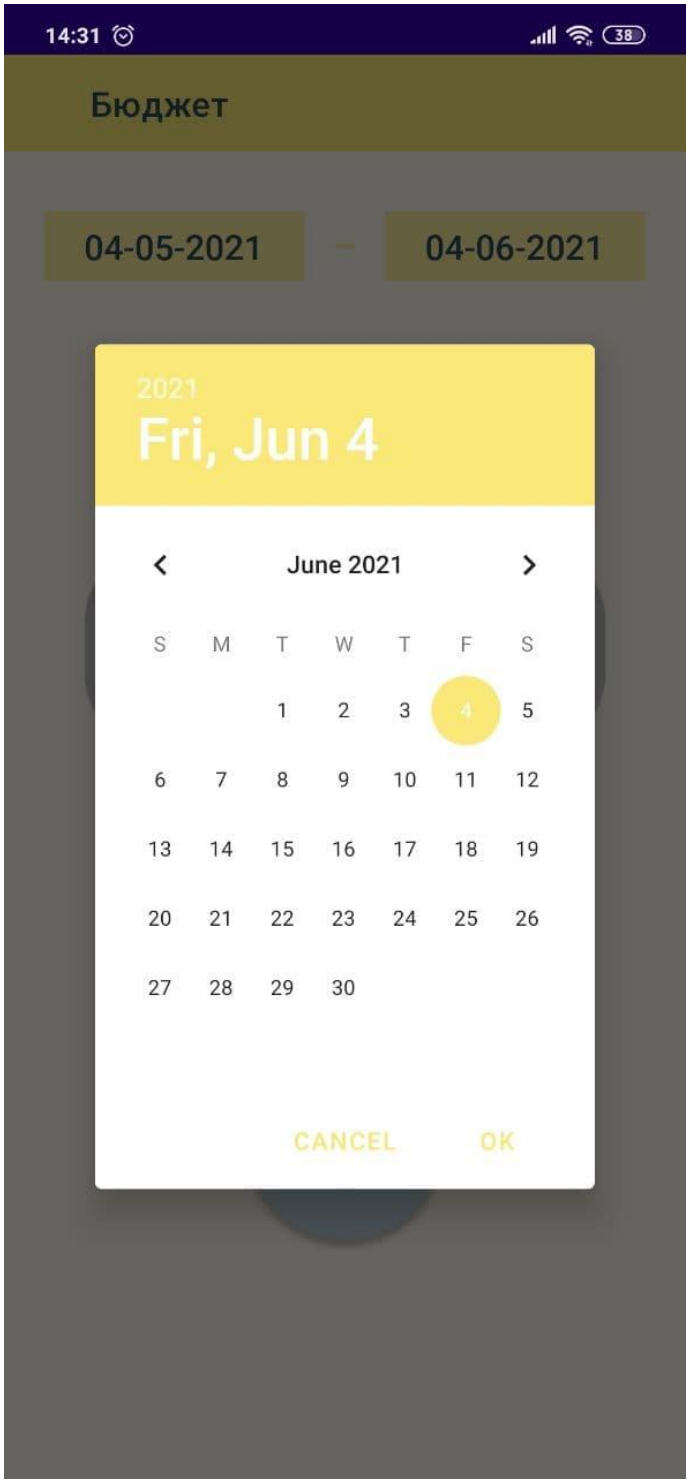

Рисунок 3.17 – Вибір періоду

Натиснувши на кнопку «Дохід», переходимо на відповідний екран (рис. 3.18).

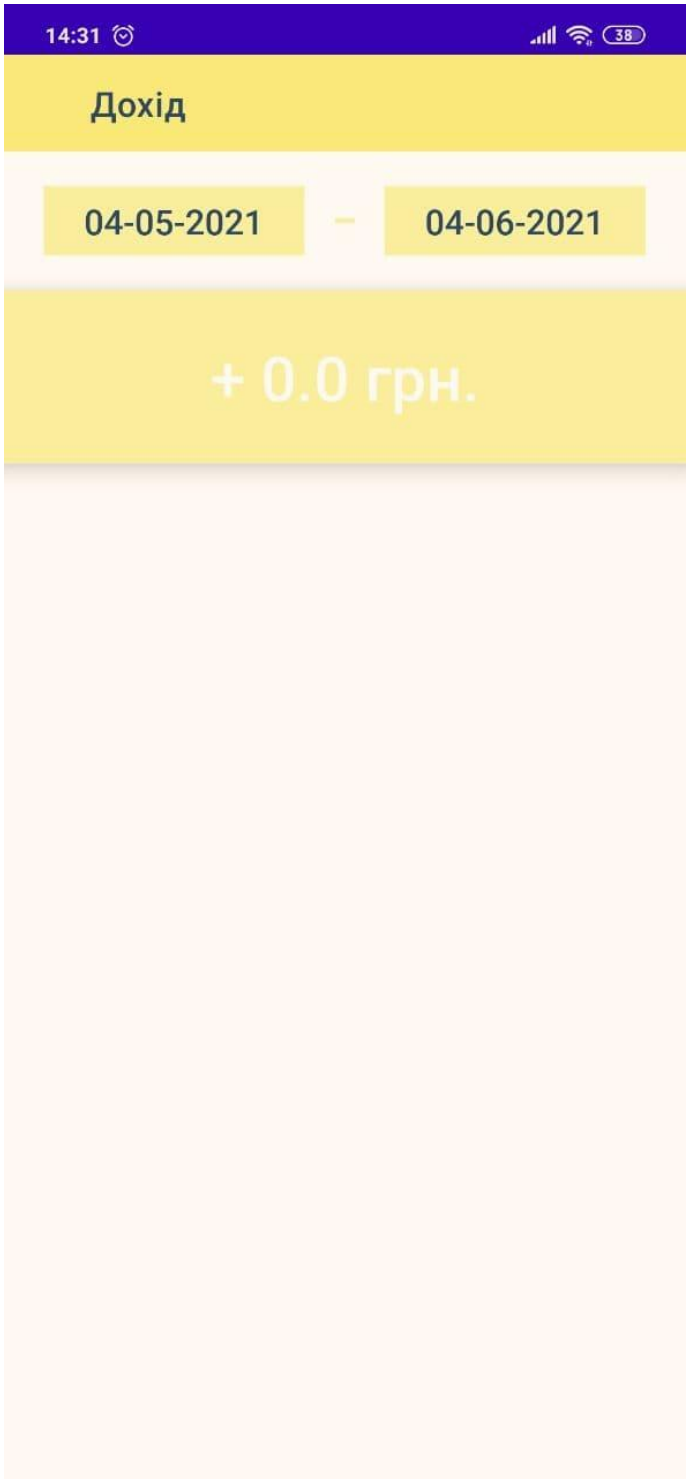

Рисунок 3.18 – Екран «Дохід»

Натискаємо на кнопку з плюсом та записуємо доходи (рис. 3.19). Біля кожного запису про дохід є кнопка редагування та дата (рис. 3.20).

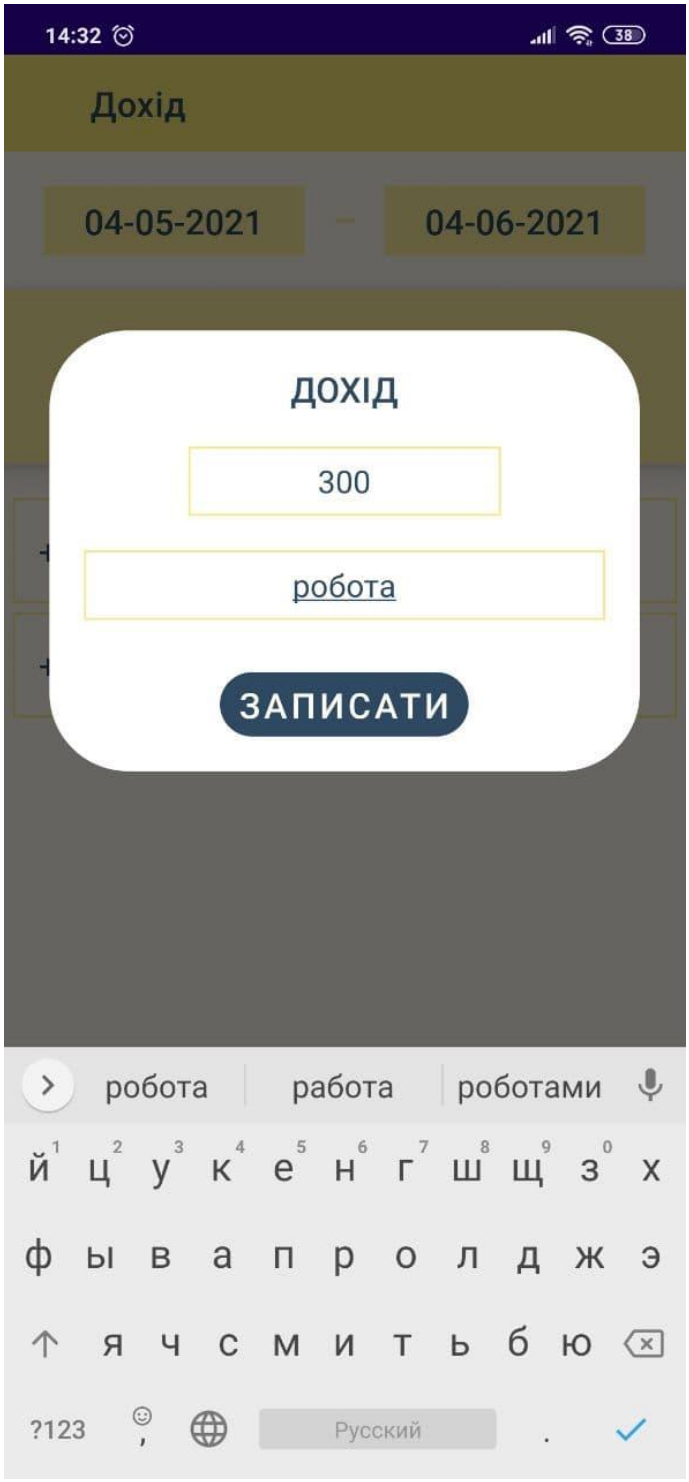

Рисунок 3.19 – Заповнення полів доходу (частина 1)

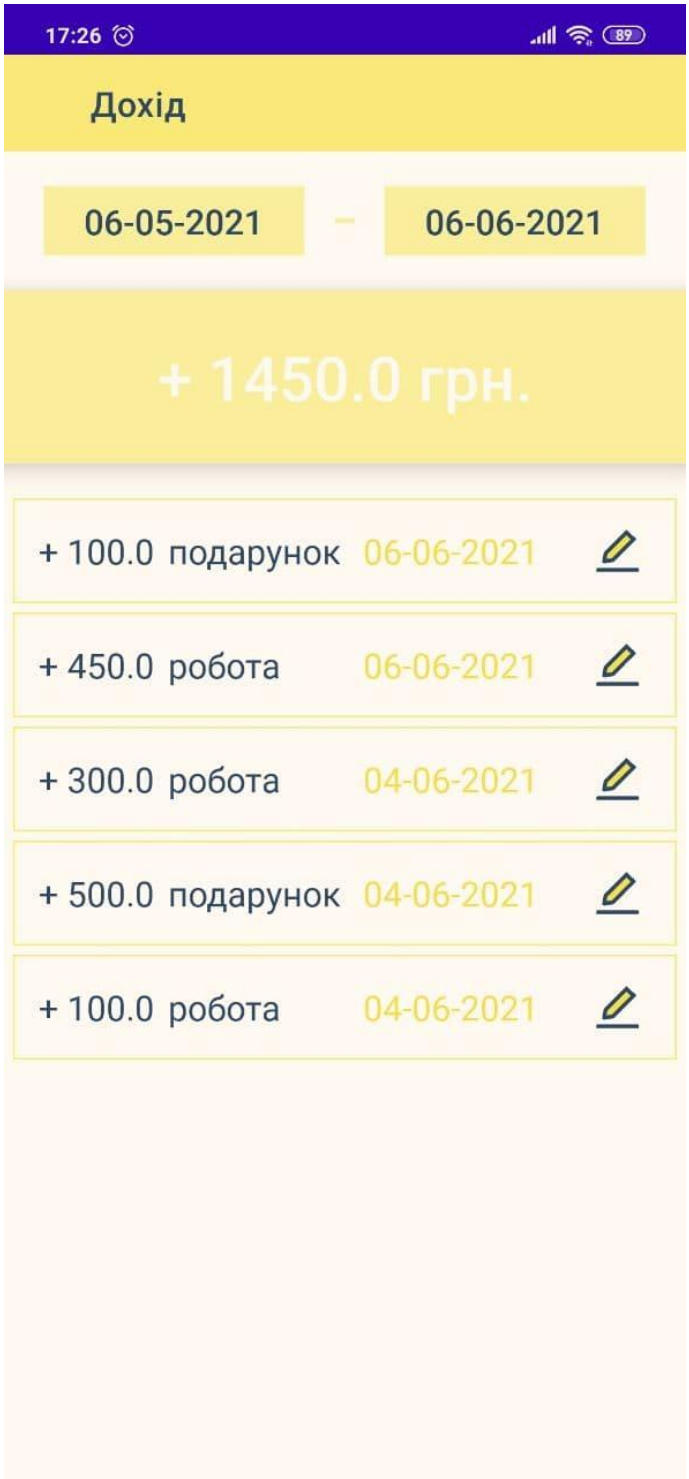

Рисунок 3.20 – Заповнений екран доходу (частина 2)

Отже, зараз можемо обрати період, за який буде відображено інформацію про доходи (рис. 3.21).

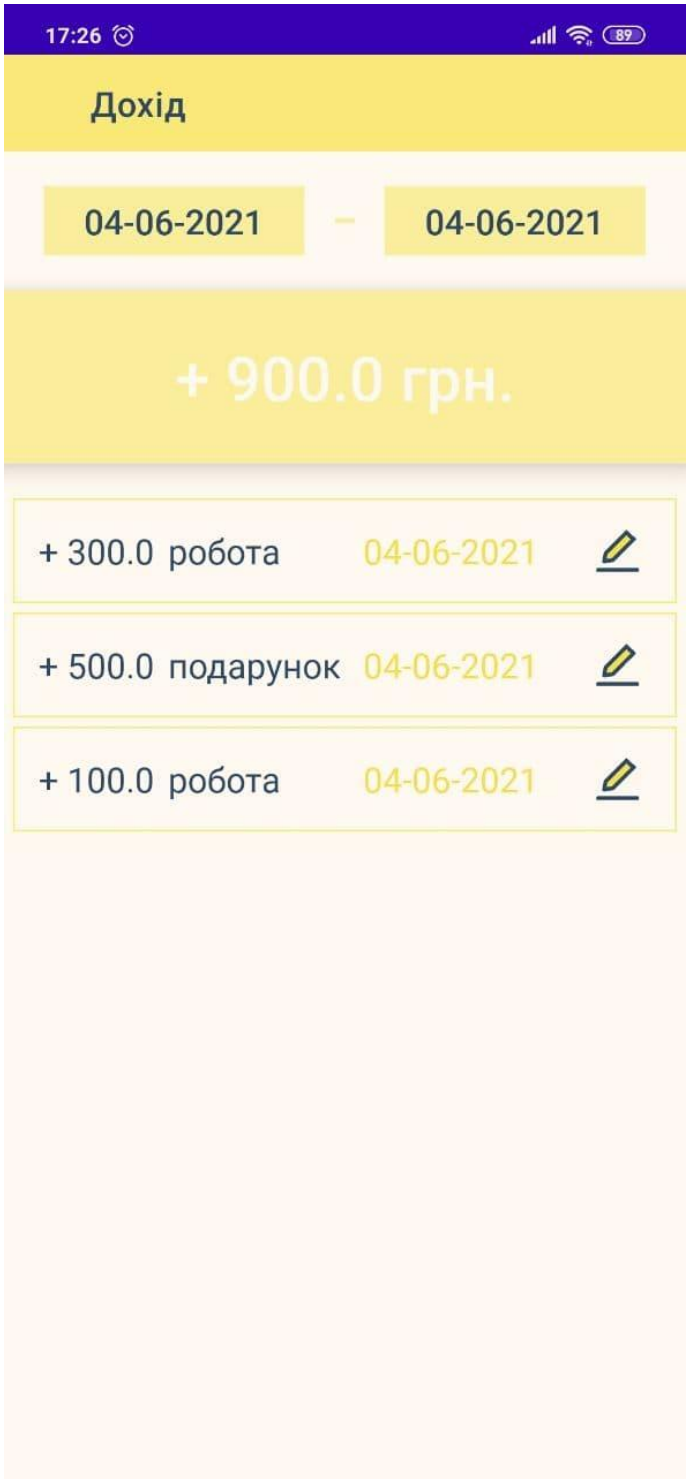

Рисунок 3.21 – Результат сортування даних про доходи за 04.06.21

Аналогічно функціонує екран «Витрати» (рис. 3.22).

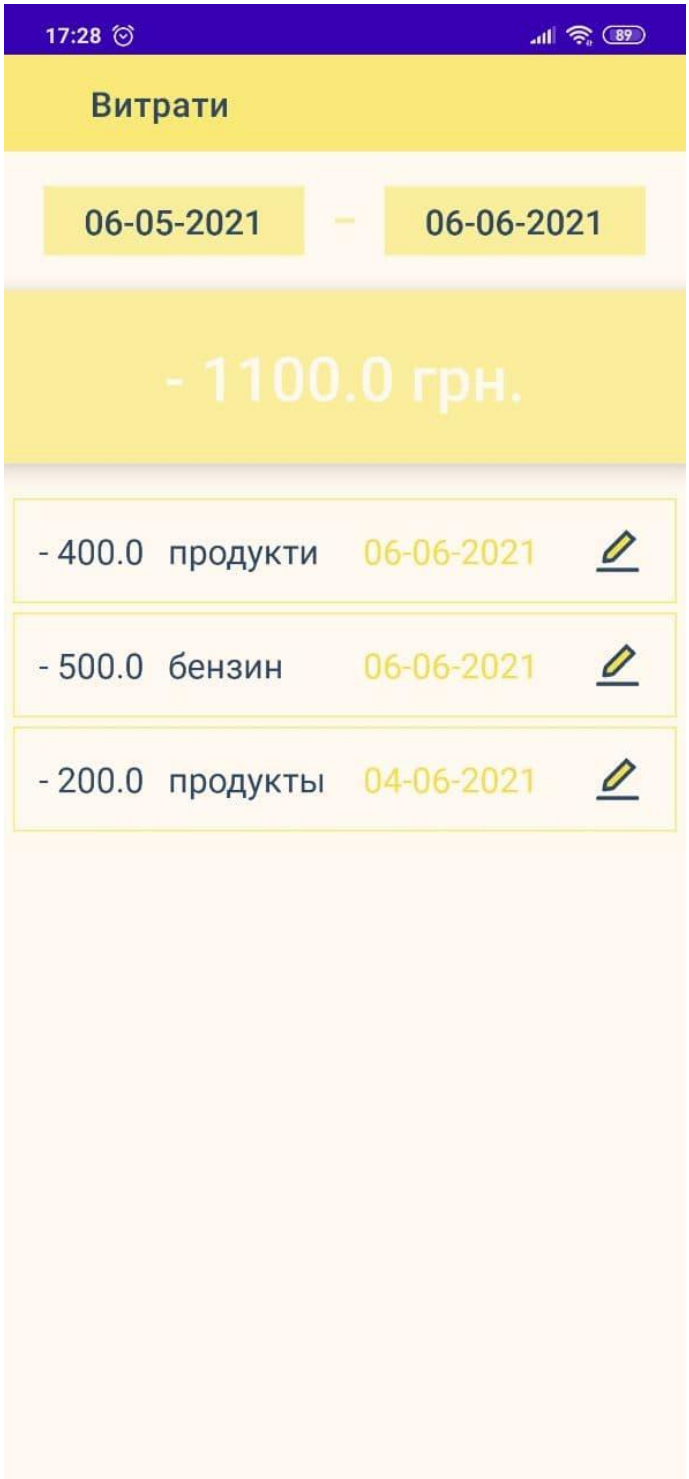

Рисунок 3.22 – Заповнений екран витрат

Натискаючи на кнопку зі сходинками, переходимо на екран «Ціль», в якому реалізовано один із методів накопичення (рис. 3.23).

Рисунок 3.23 – Діалогове вікно «Ціль»

Отже, потрібно визначити та заповнити конкретну ціль та суму на її реалізацію. Після цього на екрані буде зазначено скільки залишилось, скільки накопичено, саму ціль та всю суму (рис. 3.24).

Рисунок 3.24 – Результат заповнення початкових даних

Поле для вводу «внесок» заповнюємо сумою поточного внеску (рис. 3.25) та отримуємо результат (рис.3.26). У разі досягнення цілі, можна почати спочатку (рис. 3.27).

Рисунок 3.25 – Введення даних

Рисунок 3.26 - Результат

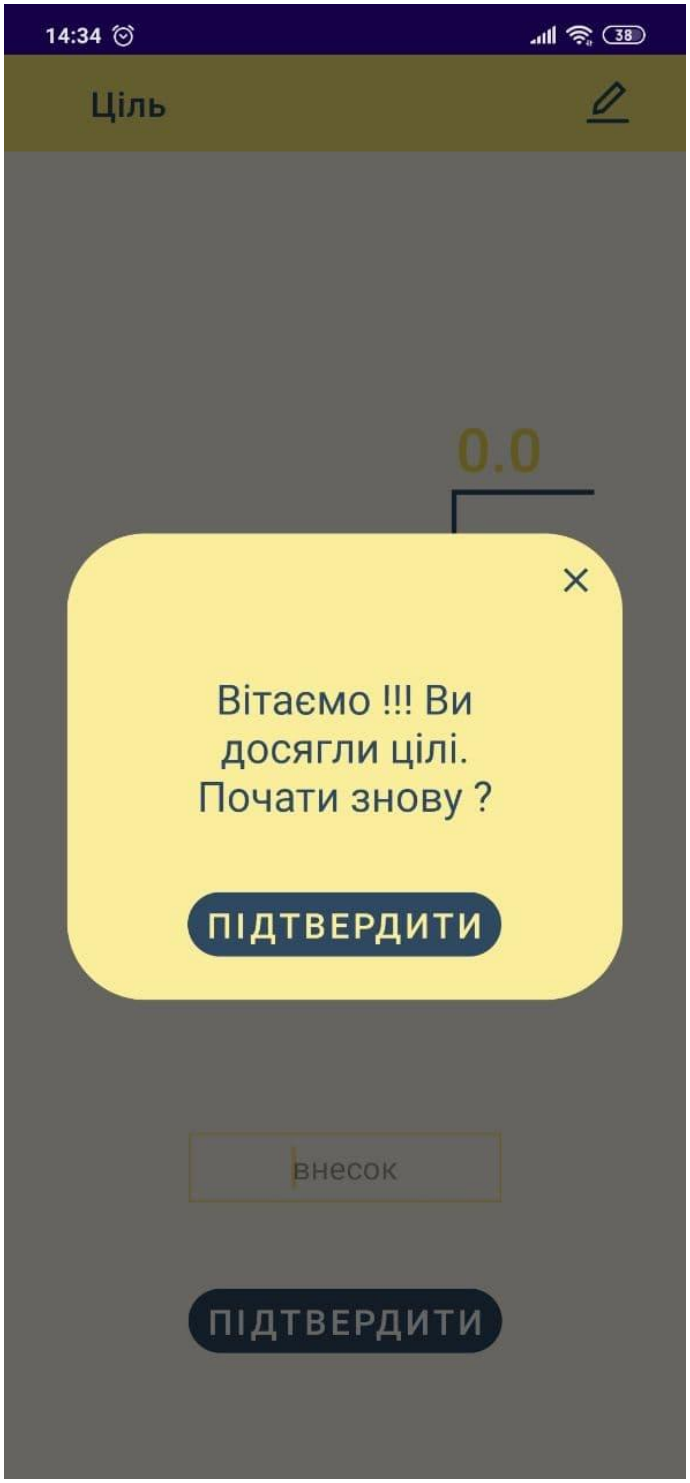

Рисунок 3.27 – Діалогове вікно досягнення цілі

У результаті натискання на кнопку з літерою «І», переходимо на екран з детальною інформацією про методи накопичення коштів з візуальним представленням (рис. 3.28-3.30).

Рисунок 3.28 – Екран інформації

Рисунок 3.28 – Екран інформації (продовження)

Рисунок 3.28 – Екран інформації (продовження)

Натискаючи на кнопку «5050», переходимо на екран трендового методу накопичення коштів.

Суть даного способу: на екрані розташовані цифри від 1 до 100 (рис 3.29). Якщо скласти всі ці числа між собою, то вийде 5050 – сума, яку будемо накопичувати. Тепер беремо скарбничку й тоді, коли будемо складати в неї – викреслюємо це ж число.

Експеримент продовжуємо до тих пір, поки не викреслимо всі числа.

Суть методу можна прочитати, натиснувши на кнопку «і» екрану «5050».

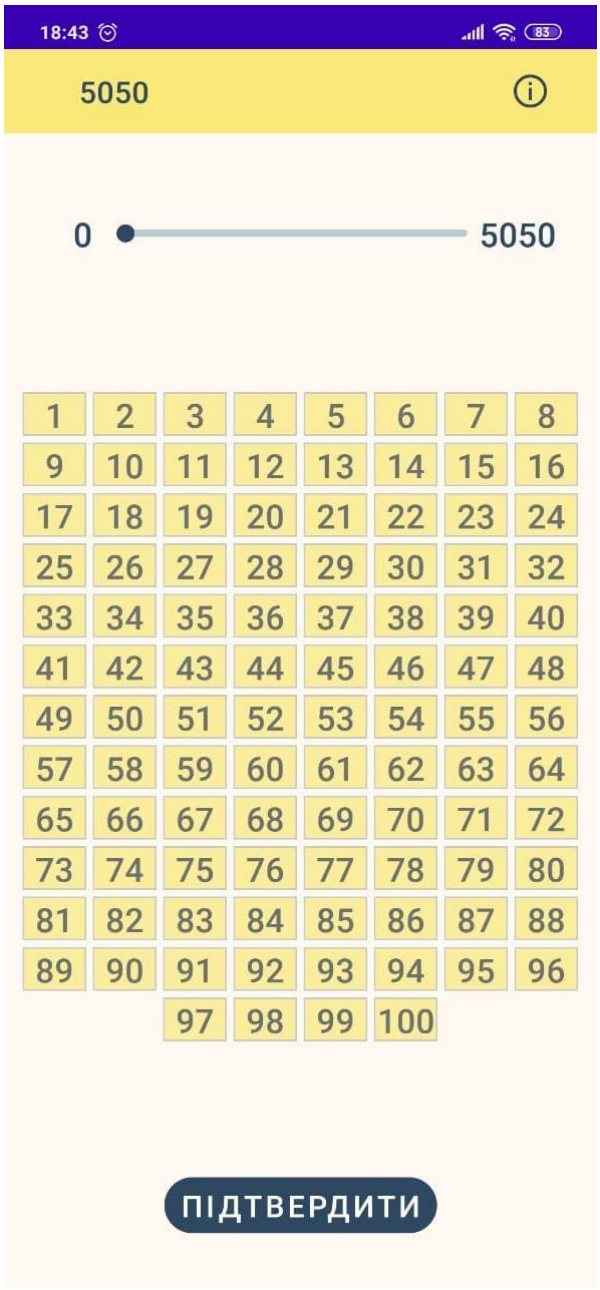

Рисунок 3.29 – Початковий екран «5050»

Числа викреслюються шляхом натискання на них (рис. 3.30), для того, щоб зафіксувати суму внеску потрібно натиснути на кнопку «ПІДТВЕРДИТИ».

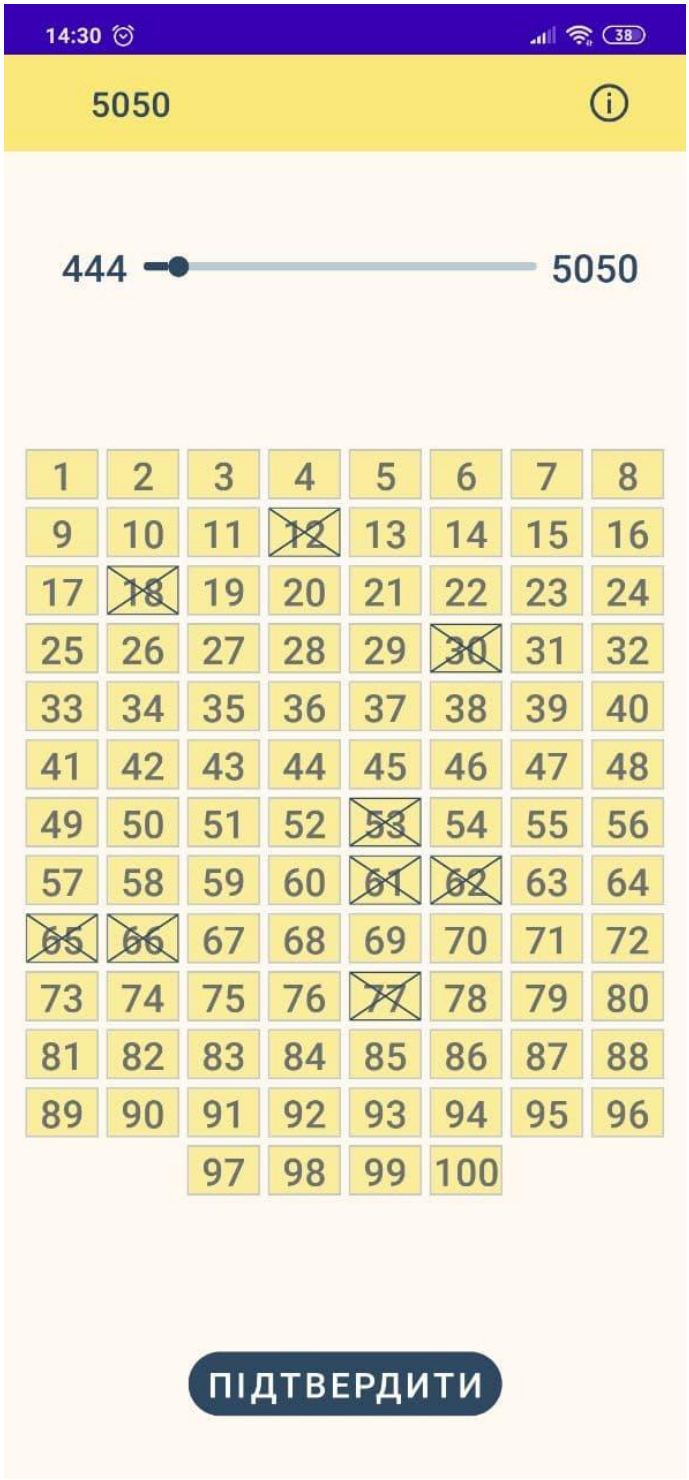

Рисунок 3.30 – Закреслені комірки

Якщо після підтвердження, потрібно відмінити вклад, то натискаємо на число й стає доступним діалогове вікно, де потрібно підтвердити відміну (рис. 3.31).

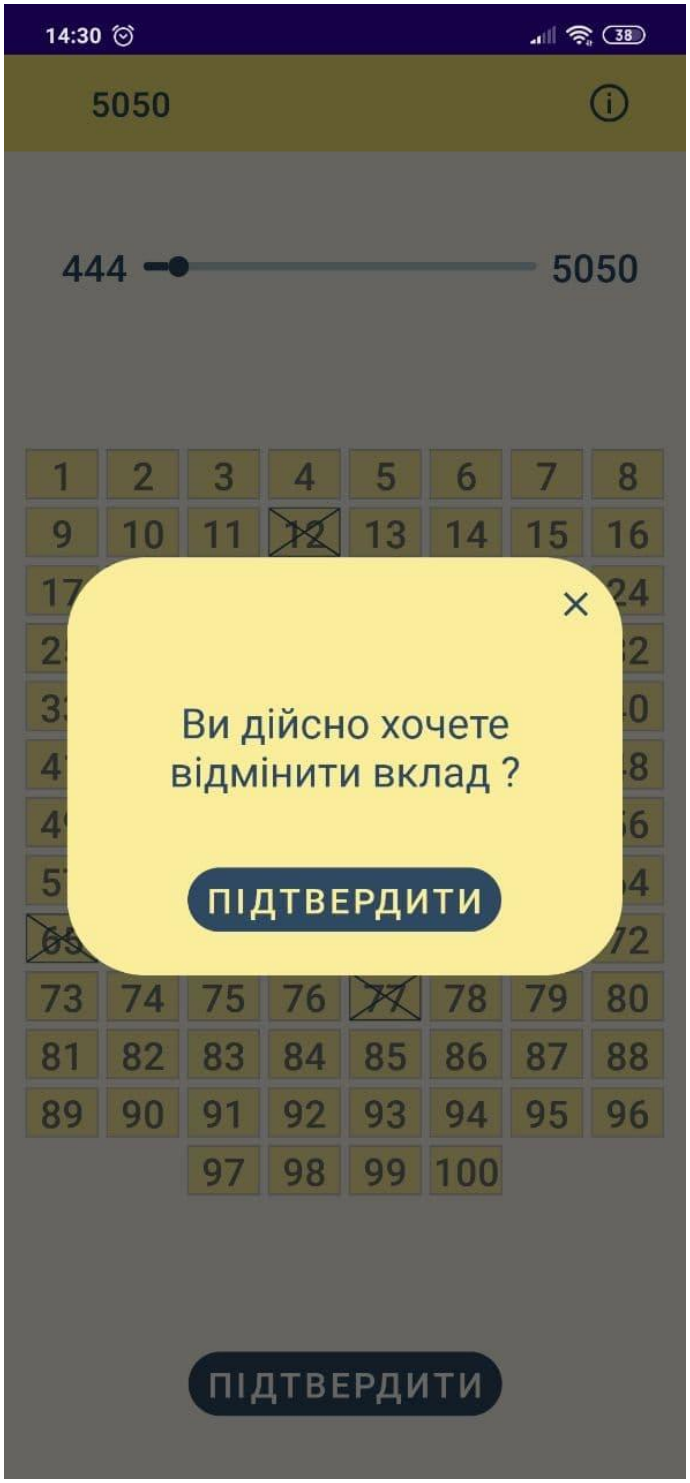

Рисунок 3.31 – Підтвердження відміни

Після досягнення мети, результат обнуляється (рис. 3. 32).

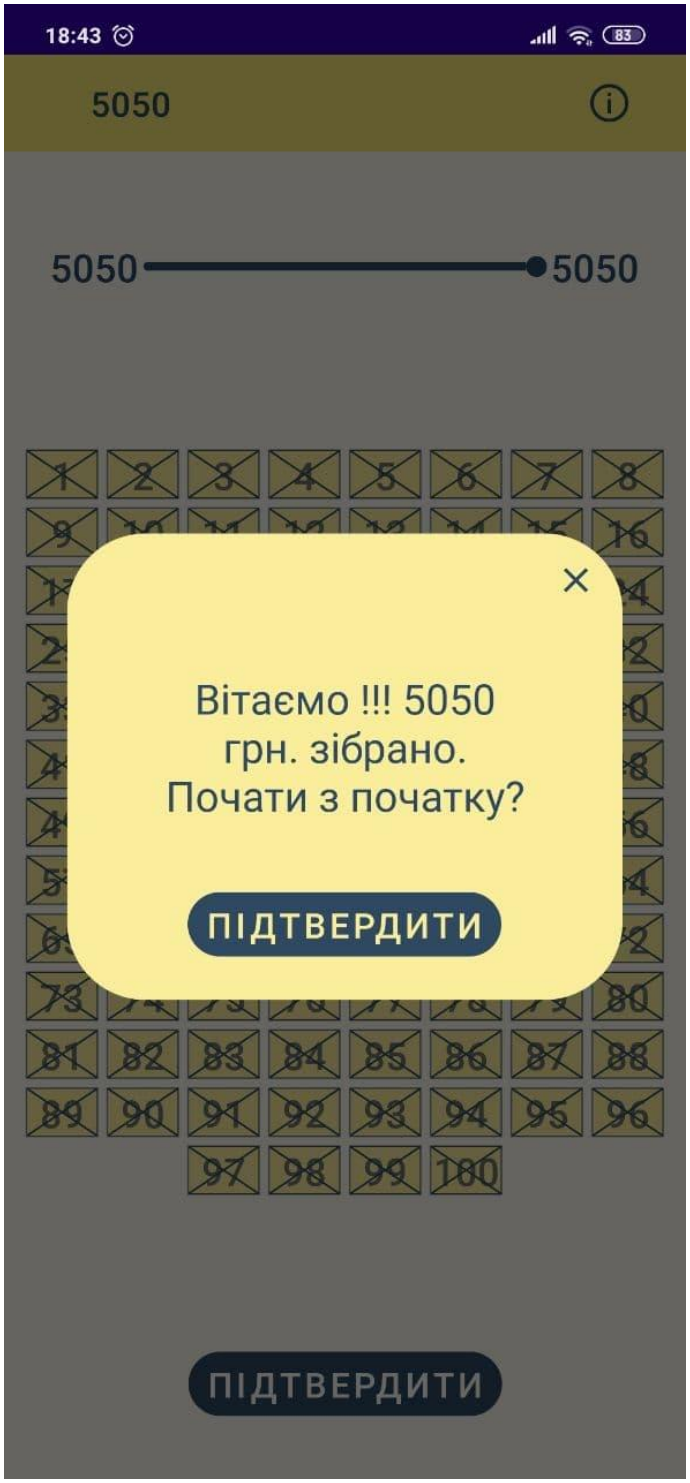

Рисунок 3.32 – Досягнення мети

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра було розроблено мобільний додаток для ведення особистого бюджету на базі операційної системи Android.

У ході роботи було визначено основні задачі проекту, з яких було виконано наступні:

– проведено аналіз предметної області та визначено актуальність роботи;
– проведено аналіз продуктів-аналогів і визначено їх переваги та недоліки;
– виконано моделювання та проектування даного мобільного додатку;
– створено прототип та дизайн мобільного додатку;
– розроблено та протестовано мобільний додаток.

Виконано моделювання варіантів використання, побудовано Use Case діаграму, побудовано діаграму IDEF0 та діаграму декомпозиції, спроектовано модель бази даних.

Складено технічне завдання, яке представлено у додатку А.

Виконано планування робіт (додаток Б).

Апробацію результатів дослідження наведено у додатку В.

Код розробленого мобільного додатку для ведення особистого бюджету наведено у додатку Г.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. V. Nikolaienko, V. Antypenko. Mobile application for maintaining a personal budget. Інформатика, математика, автоматика: матеріали та програма міжнародної науково-технічної конференції, м. Суми, 19-23 квітня 2021 р. – Суми: Сумський державний університет, 2021.

2. 6 способів комфортно накопичити гроші [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://minfin.com.ua/ua/2019/04/17/37445354/.

3. Veis Serifi. Functional and information modeling of production using IDEF methods / Veis Serifi, Predrag Dašić, R. Ječmenica, D.Labović. // Strojniski Vestnik . – 2013. – №55(2). – С. 131-140.

4. Gary R.Waissi. Automation of strategy using IDEF0 — A proof of concept / Gary R.Waissi, Mustafa Demir, Jane E. Humble, Benjamin Lev. // Operations Research Perspectives. – 2015. – №2. – С. 106-113.

5. Шатілов О.В. МОДЕЛЮВАННЯ ПРОЦЕСУ УПРАВЛІННЯ СТРАТЕГІЧНОЮ ГНУЧКІСТЮ ПІДПРИЄМСТВА. // Ефективна економіка. – 2013. – №1.

6. UML для бізнес-моделювання: для чого потрібні діаграми процесів [Електронний ресурс] – Режим доступу до ресурсу: https://evergreens.com.ua/ua/articles/uml-diagrams.html.

7. Методичні вказівки до виконання комп’ютерних практикумів з дисципліни «Управління ІТ-проектами». Управління ІТ-проектами. / Уклад.: А. В. Яковенко, О. О. Коновал. – К.: НТУУ «КПІ ім. І. Сікорського», 2017. – 47 с.

8. Radoslaw Klimekек. Formal Analysis Of Use Case Diagrams / Radoslaw Klimekек, Piotr Szwed. // Computer Science. – 2010. – №11. – С. 115–131.

9. Radoslaw Klimekек. Formal Analysis Of Use Case Diagrams / Radoslaw Klimekек, Piotr Szwed. // Computer Science. – 2010. – №11. – С. 115–131.

10. Зарицька О.Л. Бази даних та інформаційні системи: Методичний посібник. – Житомир: Вид-во ЖДУ ім. І. Франка, 2009. – 132 с.

11. Pantone Reveals Color Of The Year 2021: PANTONE® 17-5104 Ultimate Gray And PANTONE 13-0647 Illuminating [Електронний ресурс] – Режим доступу до ресурсу: https://www.pantone.com/articles/press-releases/pantone-reveals-color-of-the-year-2021-pantone-175104-ultimate-gray-and-pantone-130647-illuminating.

12. Get started with the Navigation component [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/navigation/navigation-getting-started.

13. Coroutines guide [Електронний ресурс] – Режим доступу до ресурсу: https://kotlinlang.org/docs/coroutines-guide.html.

14. Room. Entity [Електронний ресурс] – Режим доступу до ресурсу: https://startandroid.ru/ru/courses/architecture-components/27-course/architecture-components/530-urok-6-room-entity.html.

15. Dao [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/reference/android/arch/persistence/room/Dao#inherited-methods.

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку проекту
«Мобільний додаток для ведення особистого бюджету»

1 Призначення й мета створення мобільного додатку

1.1 Призначення мобільного додатку

Мобільний додаток має надавати можливість користувачам записувати особисті витрати та доходи, ставити грошові цілі та накопичувати власні зберігання.

1.2 Мета створення мобільного додатку

Метою роботи є розроблення мобільного додатку для ведення особистого бюджету на базі операційної системи Android, застосування якого надасть можливість підвищення ефективності планування щодо використання власних фінансових ресурсів.

1.3 Цільова аудиторія

До цільової аудиторії додатку можна віднести практично всіх людей, які мають намір грамотно слідкувати за особистими коштами й витратами й володіють смартфоном на базі операційної системи Android.

2 Вимоги до мобільного додатку

2.1 Вимоги до мобільного додатку в цілому

2.1.1 Вимоги до структури й функціонування мобільного додатку

Мобільний додаток повинен складатися із головної сторінки, на якій можна обрати потрібну функцію.

2.1.2 Вимоги до збереження інформації

Уся інформація надана у мобільному додатку буде зберігатися у базі даних реалізованій засобами системи управління базами даних SQLite.

2.1.3 Вимоги до розмежування доступу

Розроблюваний мобільний додаток має бути загальнодоступним. У мобільному додатку відсутнє розмежування за рівнем доступу до інформації.

2.2 Структура мобільного додатку

2.2.1 Загальна інформація про структуру мобільного додатку

Структура додатку представлена набором екранів, які також є пунктами головного меню.

Такими екранами є наступні:

– Головний – початкова сторінка з головним меню;
– Доходи – інформацію про дохід;
– Витрати – інформацію про витрати;
– Ціль – накопичення коштів шляхом встановлення цілі;
– 5050 – «трендовий» метод накопичення коштів;
– Інформація – рекомендації щодо накопичування коштів.

2.2.2 Навігація

Навігація у мобільному додатку відбувається переходом із головного екрану при натисненні на відповідні кнопки меню.

2.2.3 Дизайн та структура додатку

Стиль мобільного додатку має бути сучасним, приємним для сприйняття, інтерфейс – зручним і зрозумілим, у якості основних кольорів було обрано жовті, сірі та сині відтінки.

Розташування елементів на головній сторінці додатку показано на рисунку А.1.

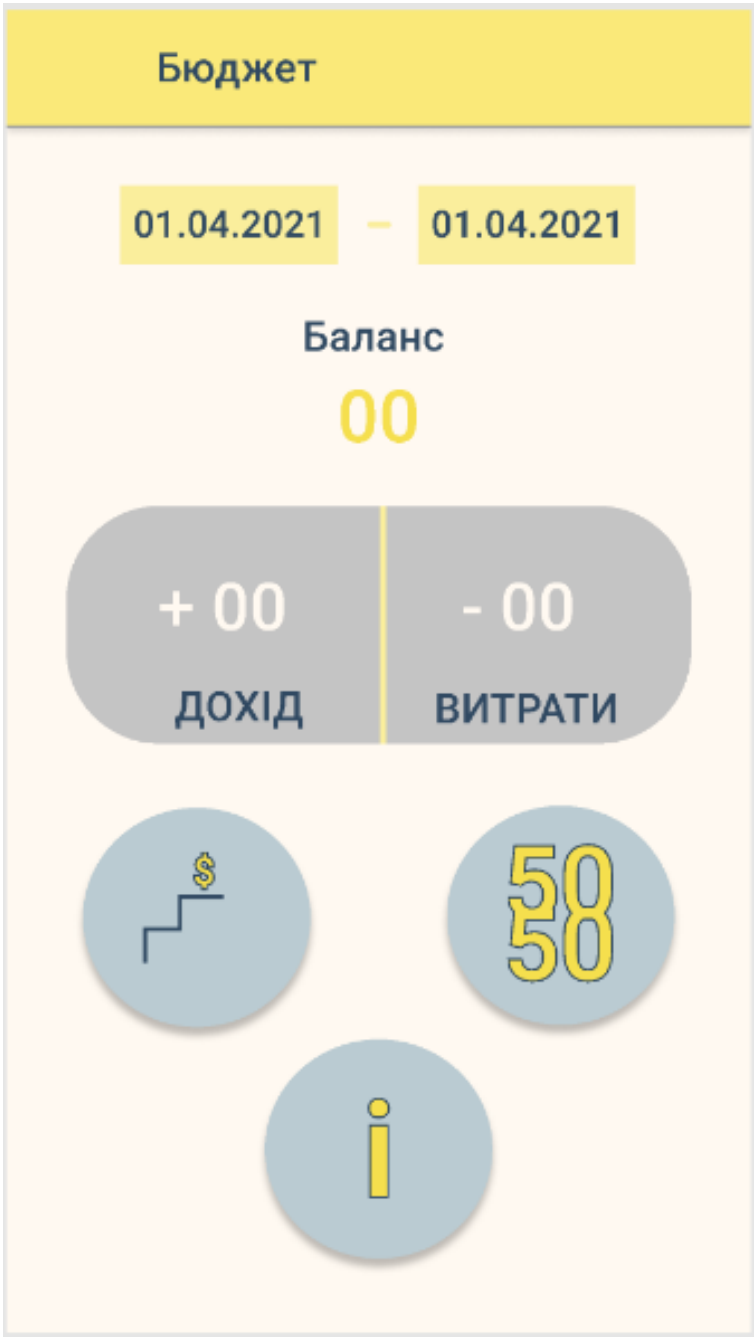

Рисунок А.1 – Схема головної сторінки мобільного додатку

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Потреби користувача представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	Потреби користувача	Джерело
UN-01	Перегляд даних про дохід	Користувач
UN-02	Перегляд даних про витрати	Користувач
UN-03	Редагування даних про дохід	Користувач
UN-04	Редагування даних про витрати	Користувач
UN-05	Редагування даних методів накопичення	Користувач
UN-06	Перегляд розділу інформації	Користувач

2.3.2 Функціональні вимоги

На основі потреб користувача були визначені такі функціональні вимоги:
-запис доходів та витрат, з можливістю визначення виду та коротким описом;
- наявність інформації щодо накопичення коштів;
-реалізація методів накопичення коштів.

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація додатку відбувається з використанням наступних технологій:

– Android Studio;
– Kotlin;
– SQLite;
– XML.

2.4.2 Вимоги до лінгвістичного забезпечення

Мобільний додаток має бути виконаний українською мовою.

2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступній вимозі – версія операційної системи Android Pie або вище.

3 Склад і зміст робіт зі створення мобільного додатку

Докладний опис етапів роботи зі створення мобільного додатку наведено в таблиці А.2.

Таблиця А.2 – Етапи створення web-додатку

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Постановка цілей необхідних для досягнення певного результату	1 день
2	Складання технічного завдання	2 дні
3	Створення прототипу	3 дні
4	Створення макету дизайну додатку	2 дні
5	Розробка	30 днів
6	Тестування	1 день
7	Підготовка до завантаження	1 день
8	Завантаження додатку на “Google Play”	1 день
9	Завершення роботи	1 день
	Загальна тривалість робіт	42 дні

4 Вимоги до складу й змісту робіт із введення мобільного додатку в експлуатацію

Для того, щоб додатком могли користуватися, необхідно розмістити його у магазині додатків від Google «Google Play».

ДОДАТОК Б.

ПЛАНУВАННЯ РОБІТ

Деталізація мети проекту методом SMART. Продуктом дипломного проекту є додаток для ведення особистого бюджету.

Результати деталізації методом SMART розміщені у таблиці Б.1.

Таблиця Б.1 − Деталізація мети методом SMART

Specific (конкретна)	Створити додаток, що має надавати можливість користувачам записувати особисті витрати та доходи, ставити грошові цілі та накопичувати власні зберігання.
Measurable (вимірювана)	Результатом роботи проекту є оцінка замовника.
Achievable (досяжна)	Реалізації системи здійснюється за допомогою середовища розробки Android Studio, з використанням мови програмування Kotlin, мови розмітки XML, інформація надана у мобільному додатку буде зберігатися у базі даних реалізованій засобами системи управління базами даних SQLite.
Relevant (реалістична)	У наявності є всі необхідні технічні та програмні засоби. Розробники достатньо кваліфіковані для виконання поставлених задач.
Time-framed (обмежена у часі)	Ціль має часове обмеження. Робота повинна бути виконана у терміни, що були оговорені замовником проекту. Проект повинен бути виконаний згідно з календарним планом.

Планування змісту структури робіт. Основним інструментом для планування змісту структури робіт служить WBS діаграма − графічне подання згрупованих елементів проекту у вигляді пакета робіт, які ієрархічно пов’язані з продуктом проекту. Побудуємо структуру WBS, у якій детально опишемо роботи, які потрібно виконати на кожному етапі створення проекту. Виконаємо декомпозицію робіт для даного проекту. Діаграма WBS зображена на рисунку Б.1.

Планування структури організації, для впровадження готового проекту (OBS). Після побудови WBS розробимо організаційну структуру виконавців OBS. Організаційна структура проекту стосується тільки внутрішньої організаційної структури проекту і не стосується відносин проектних груп чи учасників з батьківськими організаціями. Діаграма OBS зображена на рисунку Б.2. Список виконавців, що функціонують в проекті знаходиться в таблиці Б.2.

Таблиця Б.2 − Виконавці проекту

Роль	Ім’я	Проектна роль
Розробник	Ніколаєнко В.В.	Виконує розробку мобільного додатку.
Дизайнер	Ніколаєнко В.В.	Створює дизайн програми.
Тестувальник	Ніколаєнко В.В.	Відповідає за тестування функціоналу та дизайну додатку, перевірку додатку на адекватність.
Косультант проекту	Антипенко В.П.	Формує завдання на розробку проекту.
Менеджер проекту	Ніколаєнко В.В.	Відповідає за виконання термінів, розподіл ресурсів та завдань. Виконує збір та аналіз даних.

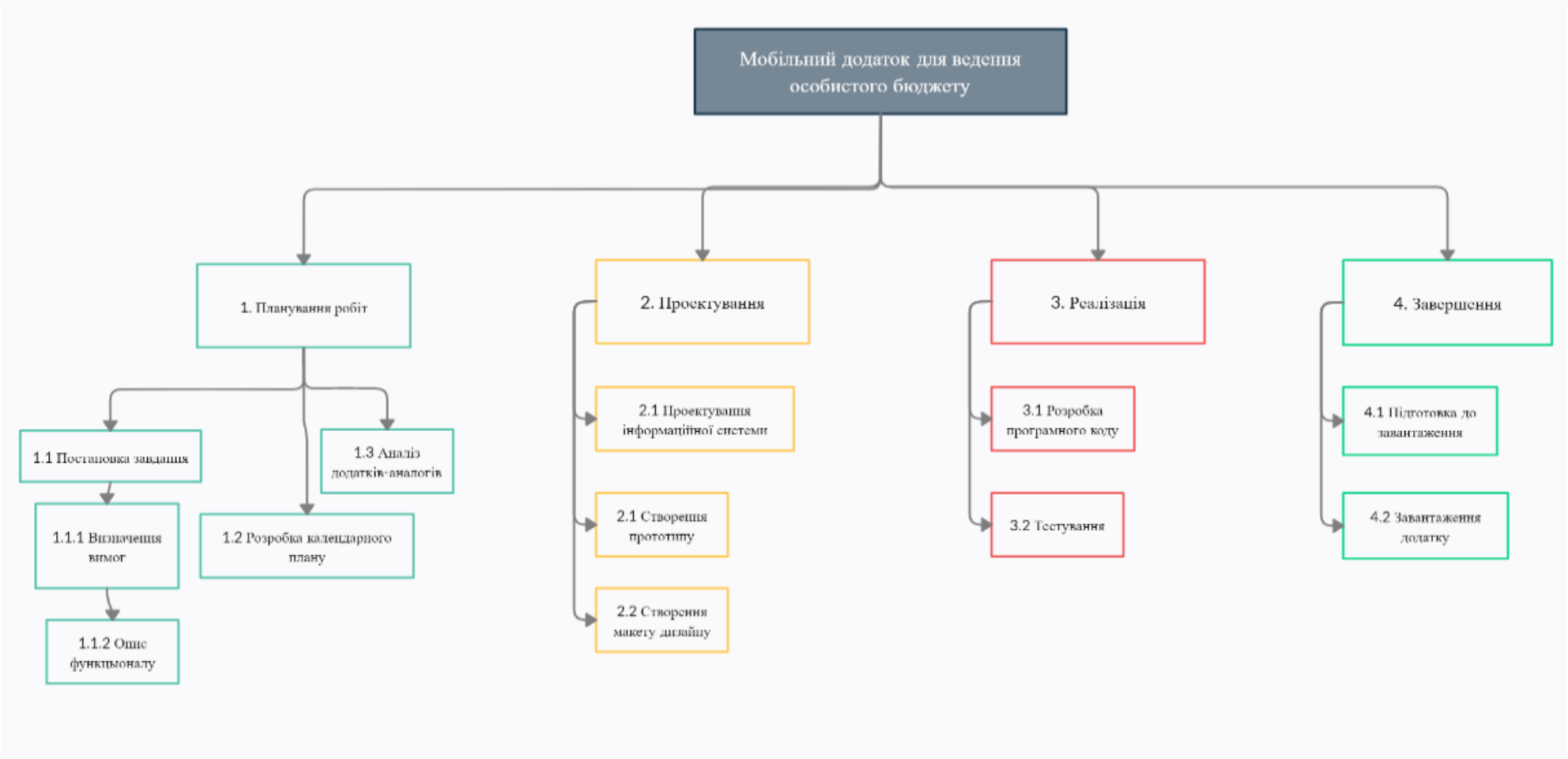

Рисунок Б.1 – WBS структура робіт проекту

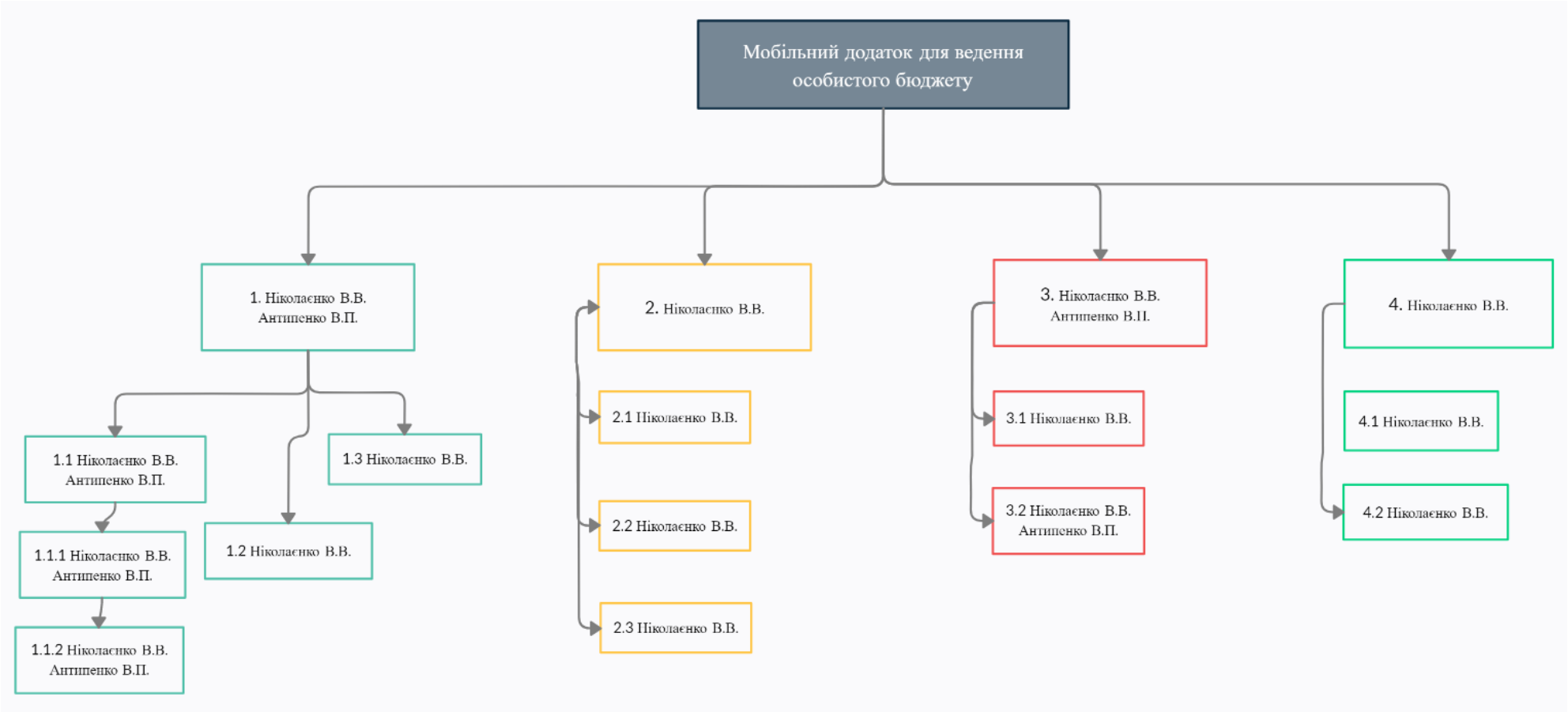

Рисунок Б.2 – Організаційна структура проекту (OBS)

Діаграма Ганта. Далі побудуємо календарний план виконання дипломного проекту. Найпоширеніший формат графіка в будь-якій галузі – діаграма Ганта. Цей графік дозволяє менеджерам проекту та всій команді розробників візуалізувати графіки часу і взаємозв'язок між окремими завданнями та етапами роботи над проектом. Тривалість виконання робіт зазначена в днях, але фактична тривалість виконання робіт приблизно дорівнює 2-3 години на день. Для того щоб мати реальне уявлення про тривалість виконання робіт з урахуванням обмеженості у використанні ресурсів, із урахуванням вихідних та святкових днів, побудовано календарний графік. Діаграма Ганта та список робіт діаграми Ганта зображені на рисунках Б.3-Б.4.

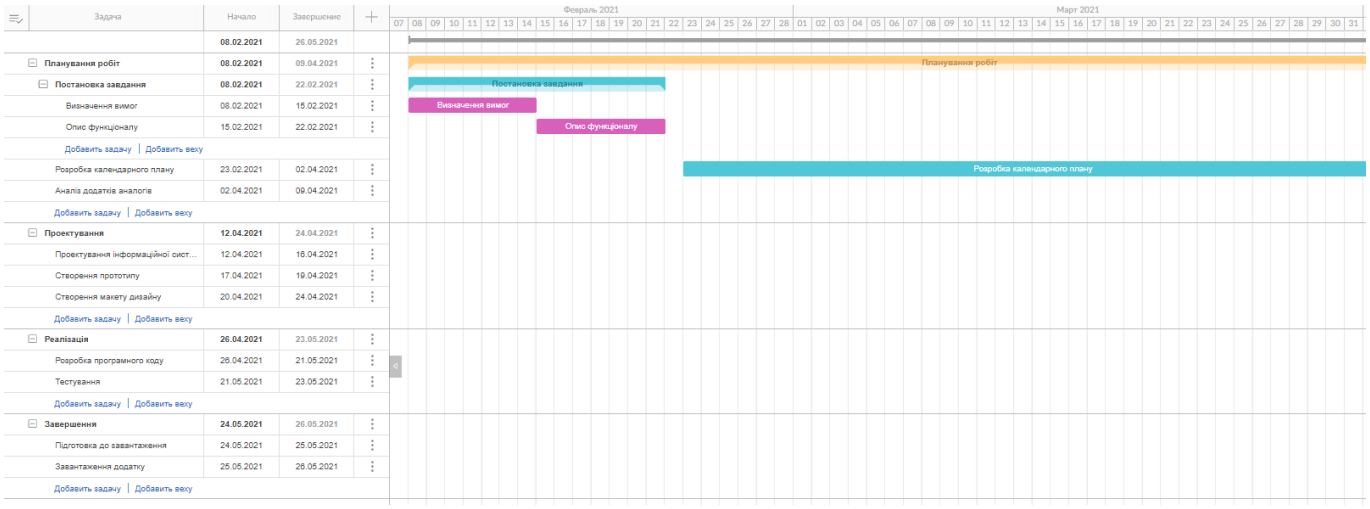

Рисунок Б.3 − Діаграма Ганта

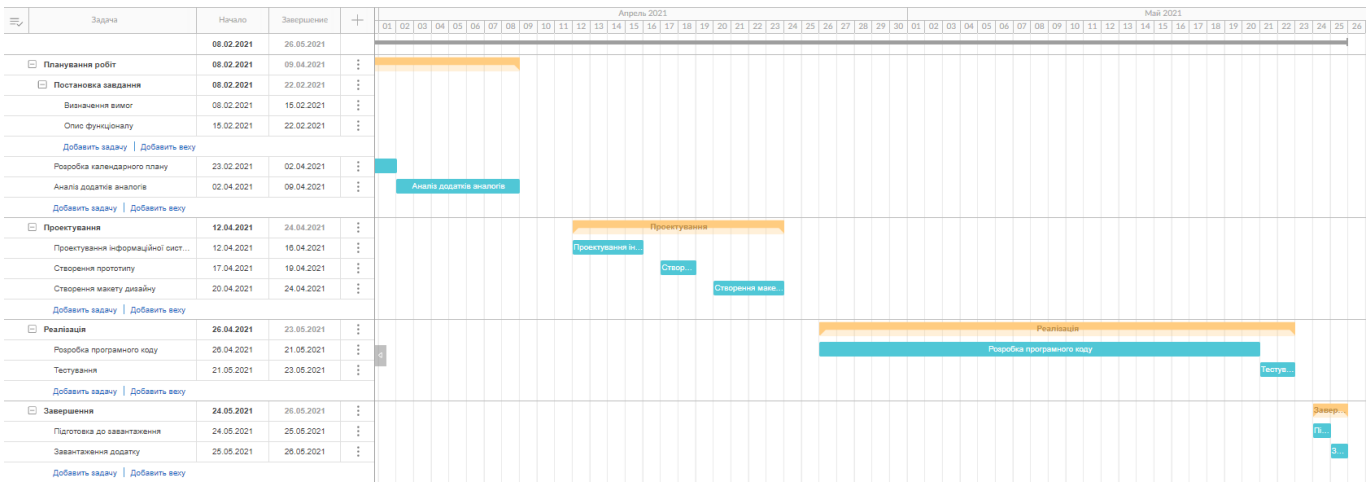

Рисунок Б.4 − Продовження діаграми Ганта

Аналіз ризиків. Управління ризиками в процесі управління проектами визначається як комплекс заходів, що включають ідентифікацію, аналіз ризиків та прийняття рішень, направлених на зниження імовірності та ступеня їхнього впливу на хід, результати та продукти цих проектів.

Класифікація ризиків проекту зазвичай проводиться на етапі їх аналізу за такими критеріями як імовірність виникнення та ступінь впливу на проект (табл. Б.4).

Ідентифікація ризиків проекту – визначення ризиків, здатних впливати на проект, і документування їхніх характеристик: опис і природа ризику, категорія, умови виникнення, методи реагування.

Оцінюємо ризики за показниками, що знаходяться в таблиці Б.3. На основі оцінки будуємо матрицю ймовірності виникнення ризиків та впливу ризику, що зображена в таблиці Б.5.

Таблиця Б.3 − Шкала оцінювання

Оцінка	Ймовірність виникнення	Вплив ризику
1	Низька	Низький
2	Середня	Середній
3	Висока	Високий

Таблиця Б.4 – Класифікація ризиків

№	Ризик	Ймовірність	Ступінь впливу
1	Нечітке ТЗ	2	2
2	Неоптимальний розподіл часу	3	3
3	Низька продуктивність	2	2
4	Проблеми роботи програмного забезпечення	1	2
5	Проблеми роботи апаратного забезпечення	1	2
6	Виникнення проблем реалізації	2	3

Таблиця Б.5 – Матриця ризиків

Ймовірність					
	3			2	
	2	4, 5	1, 3	6	
	1				
		1	2	3	
	Ступінь впливу				

Планування реагування на виявлені значимі ризики – визначення процедур і методів щодо ослаблення негативного впливу ризикових подій. План при виникненні ризику наведений у таблиці Б.6.

Таблиця Б.6 – План при виникненні ризику

Ризик	План при виникненні ризику
Нечітке ТЗ	Обговорення та затвердження ТЗ.
Неоптимальний розподіл часу	Звернути особливу увагу на правильність розподілу часу. Правильно визначити пріоритети виконання робіт. Чітко дотримуватися календарного плану.
Низька продуктивність	Правилний розподіл часу роботи та відпочинку.
Проблеми роботи програмного забезпечення	Перезавантажити програмне забезпечення.
Проблеми роботи апаратного забезпечення	Виконати ремонт або знайти альтернативу.
Виникнення проблем реалізації	Шукати додаткову інформацію, інші методи реалізації

ДОДАТОК В

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Mobile application for maintaining a personal budget

V. Nikolaienko, *Student*; V. Antypenko, *Associate Professor*
Sumy State University, Sumy, Ukraine

Currently, technologies and gadgets have become a vital part of our trendy life. With the increasing number of digital users, the mobile app industry has seen an immense growth, which only continuously rises. Given how often we use many apps on our smartphones and tablets to shop, sell, study, play, listen to music, communicate, order food, book travel tickets, find answers to questions etc., there is no end to mobile app usage in the upcoming years. Therefore, mobile apps development is a very relevant and popular issue today as it has promising future extension prospects.

Mobile applications make it much easier to perform daily tasks by automating some processes. For instance, every modern person wants to be financially responsible and in control of his or her own finances. This implies constant accounting of the personal budget – something that many people aspire to, but only a few of them succeed. On one hand, the major reasons behind this can be various factors such as inconsistency, lack of a specific purpose, leaving notes in different places or even the absence of such habit at all. On the other hand, during using mobile applications, it is possible to encounter problems such as unclear and/or inconvenient interface, lack of useful information or features etc., which simply can prevent further user's work with this application.

During the pandemic, the issue of finance planning is becoming increasingly important. Therefore, the purpose of this project is to develop a mobile application for personal budgeting based on the Android operating system.

The aim of this application is to enable the user to keep track of personal finances, accumulate, set goals, obtain information on methods for saving money and choose the one that suits him or her the most. The main aspect when creating a mobile application is an attractive and user-friendly interface that will provide the user with the necessary functionality.

The result of this work is a developed mobile app for personal budget maintaining based on the Android operating system, which allows efficient managing and monitoring person's own budget.

ДОДАТОК Г.

ПРОГРАМНИЙ КОД МОБІЛЬНОГО ДОДАТКУ

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/budgetControlToolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="@color/primary_yellow"
        android:paddingEnd="20dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:titleMarginStart="50dp"
        app:titleTextColor="@color/text_color_blue" />

    <fragment
        android:id="@+id/fragment_container"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/budgetControlToolbar"
        app:layout_constraintBottom_toBottomOf="parent"
        app:defaultNavHost="true"
        app:navGraph="@navigation/navigation_graph" />

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

budget_fragment.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.example.budgetcontrol.view.DatePeriodView
        android:id="@+id/budgetFragmentDatePeriodView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toTopOf="@+id/balanceLinearLayout"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <LinearLayout
        android:id="@+id/balanceLinearLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintBottom_toTopOf="@+id/grayIncomeCostsCounter"
```

```
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/budgetFragmentDatePeriodView">

    <TextView
        style="@style/BlueMediumTextStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/balance"
        android:textSize="22sp"
        android:layout_gravity="center"/>

    <TextView
        android:id="@+id/balanceValue"
        style="@style/YellowMoneyNumberTextViewStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp"
        android:text="0.0" />

</LinearLayout>

<LinearLayout
    android:id="@+id/grayIncomeCostsCounter"
    android:layout_width="300dp"
    android:layout_height="120dp"
    android:orientation="horizontal"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.35">

    <!-- add clicklisteners-->
    <com.example.budgetcontrol.view.BudgetCounterGrayButtonView
        android:id="@+id/incomeCounter"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@drawable/income_counter_button_view_background"
        app:budget_component="income" />

    <View
        android:layout_width="3dp"
        android:layout_height="match_parent"
        android:background="@color/light_yellow" />

    <com.example.budgetcontrol.view.BudgetCounterGrayButtonView
        android:id="@+id/costsCounter"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@drawable/costs_counter_button_view_background"
        app:budget_component="costs" />

</LinearLayout>

<ImageView
    android:id="@+id/targetImageView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/target"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.2"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.62" />

<ImageView
    android:id="@+id/gaussNumbersImageView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/fifty_fifty"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.8"
```

```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@id/targetImageView"
        app:layout_constraintVertical_bias="0.0" />

    <ImageView
        android:id="@+id/infoImageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/info"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.8" />

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

BudgetComponent.kt

```
package com.example.budgetcontrol.enum

enum class BudgetComponent {
    INCOME,
    COSTS
}
```

BudgetCounterGrayButtonView.kt

```
package com.example.budgetcontrol.view

import android.content.Context
import android.util.AttributeSet
import android.view.View
import android.widget.TextView
import androidx.constraintlayout.widget.ConstraintLayout
import com.example.budgetcontrol.R
import com.example.budgetcontrol.enum.BudgetComponent
import kotlinx.android.synthetic.main.income_costs_counter.view.*

class BudgetCounterGrayButtonView(context: Context, attributeSet: AttributeSet?) : ConstraintLayout(context, attributeSet) {

    private lateinit var budgetComponent: BudgetComponent
    private var value: TextView

    init {
        val view = View.inflate(context, R.layout.income_costs_counter, this)
        value = view.incomeCostsValue

        getAttributes(context, attributeSet)
    }

    fun setValue(value: Float) {
        this.value.text = resources.getString(
                R.string.income_costs_value_placeholder,
                when (budgetComponent) {
                    BudgetComponent.INCOME -> "+"
                    BudgetComponent.COSTS -> "-"
                },
                value
        )
    }

    private fun setTitle(budgetComponent: BudgetComponent) {
        incomeCostsTextView.text = resources.getString(
                when (budgetComponent) {
                    BudgetComponent.INCOME -> R.string.income
                    BudgetComponent.COSTS -> R.string.costs
                })
    }

    private fun getAttributes(context: Context, attributeSet: AttributeSet?) {
        val attributes = context.obtainStyledAttributes(
                attributeSet,
                R.styleable.BudgetCounterGrayButtonView
```

```
        )
        try {
            budgetComponent = when (attributes.getInteger(R.styleable.BudgetCounterGrayButtonView_budget_component, 0)) {
                0 -> BudgetComponent.INCOME
                else -> BudgetComponent.COSTS
            }
            setTitle(budgetComponent)
        } finally {
            attributes.recycle()
        }
    }
}
```

BudgetFragment.kt

```
package com.example.budgetcontrol.fragment

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import com.example.budgetcontrol.*
import com.example.budgetcontrol.db.BudgetControlDB
import com.example.budgetcontrol.enum.BudgetComponent
import com.example.budgetcontrol.enum.FragmentType
import kotlinx.android.synthetic.main.budget_fragment.*
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch
import kotlin.math.abs

class BudgetFragment : Fragment() {

    override fun onCreateView(
            inflater: LayoutInflater,
            container: ViewGroup?,
            savedInstanceState: Bundle?
    ): View? {
        return inflater.inflate(R.layout.budget_fragment, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        (activity as MainActivity).setToolbarTitle(getString(R.string.budget))
        fetchData()

        setupNavigationButtonsOnClickListeners()
        budgetFragmentDatePeriodView.setDatePickerOnDateSetListener(this::handleDateSet)
    }

    private fun handleDateSet(){
        budgetFragmentDatePeriodView.apply {
            fetchData(getStartDate(), getEndDate())
        }
    }

    private fun setupNavigationButtonsOnClickListeners() {
        setNavigationOnClickListener(targetImageView, FragmentType.TARGET)
        setNavigationOnClickListener(gaussNumbersImageView, FragmentType.GAUSS_NUMBERS)
        setNavigationOnClickListener(infoImageView, FragmentType.INFORMATION)
        setNavigationOnClickListener(incomeCounter, FragmentType.INCOME)
        setNavigationOnClickListener(costsCounter, FragmentType.COSTS)
    }

    private fun setNavigationOnClickListener(view: View, fragmentType: FragmentType) {
        view.setOnClickListener {
            (activity as MainActivity).navigateToFragment(fragmentType)
        }
    }

    private fun fetchData(startDate: String? = null, endDate: String? = null) {
        GlobalScope.launch {
            val income = getTotalIncomeCostsAmount(BudgetComponent.INCOME, startDate, endDate)
            val costs = abs(getTotalIncomeCostsAmount(BudgetComponent.COSTS, startDate, endDate))
            val balance = income - costs
```

```
            activity?.runOnUiThread {
                incomeCounter.setValue(income)
                costsCounter.setValue(costs)
                balanceValue.text = getString(
                        R.string.income_costs_value_placeholder,
                        if (balance < 0) "-" else "",
                        abs(balance)
                )
            }
        }
    }

    private fun getTotalIncomeCostsAmount(budgetComponent: BudgetComponent, startDate: String?, endDate: String?): Float {
        var totalAmount = 0f
        val transactionDao = BudgetControlDB.getInstance(requireContext()).transactionDao()
        val allAmounts = when (budgetComponent) {
            BudgetComponent.INCOME ->{
                if(startDate.isNullOrBlank() || endDate.isNullOrBlank())
                    transactionDao.getAllIncomeAmounts()
                else
                    transactionDao.getAllIncomeAmountsByDate(startDate, endDate)
            }
            BudgetComponent.COSTS -> {
                if(startDate.isNullOrBlank() || endDate.isNullOrBlank())
                    transactionDao.getAllCostsAmounts()
                else
                    transactionDao.getAllCostsAmountsByDate(startDate, endDate)
            }
        }
        allAmounts.forEach { value ->
            totalAmount += value
        }
        return totalAmount
    }
}
```

confirmation_dialog.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/confirmation_dialog_background"
    android:gravity="center"
    android:orientation="vertical"
    android:paddingHorizontal="15dp"
    android:paddingTop="15dp"
    android:paddingBottom="25dp">

    <ImageView
        android:id="@+id/confirmationDialogCloseButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="end"
        android:src="@drawable/ic_close" />

    <TextView
        android:id="@+id/confirmationDialogTextView"
        style="@style/BlueRegularTextStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="40dp"
        android:gravity="center"
        android:paddingHorizontal="40dp"
        android:text="@string/you_want_to_cancel_contribution"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/dialogConfirmButton"
        style="@style/TextViewButtonStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_marginTop="40dp"
        android:text="@string/to_confirm"
        android:textColor="@color/confirmation_dialog_background_color" />

</LinearLayout>
```

date_period_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:gravity="center">

    <TextView
        android:id="@+id/startDate"
        style="@style/DateTextViewStyle"
        android:text="@string/from" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/date_dash"
        android:layout_gravity="center_vertical"
        android:paddingHorizontal="17dp"/>

    <TextView
        android:id="@+id/endDate"
        style="@style/DateTextViewStyle"
        android:text="@string/till" />

</LinearLayout>
```

DatePeriodView.kt

```
package com.example.budgetcontrol.view

import android.app.DatePickerDialog
import android.content.Context
import android.util.AttributeSet
import android.view.View
import android.widget.LinearLayout
import android.widget.TextView
import com.example.budgetcontrol.R
import kotlinx.android.synthetic.main.date_period_view.view.*
import java.text.SimpleDateFormat
import java.util.*

class DatePeriodView(
    context: Context,
    attributeSet: AttributeSet?
) : LinearLayout(context, attributeSet) {

    companion object{
        const val DATE_FORMAT = "dd-MM-yyyy"
    }

    //todo save dates sorting untill app closes
    //todo disable setting startdate newer than enddate

    private lateinit var startDate: TextView
    private lateinit var endDate: TextView

    private var onDateSetListener: (() -> Unit?)? = null

    init {
        init()
        setDateTextViewOnClickListener(startDate)
        setDateTextViewOnClickListener(endDate)
    }
```

```
    fun setDatePickerOnDateSetListener(listener: () -> Unit){
        onDateSetListener = listener
    }

    fun getStartDate(): String{
        return startDate.text.toString()
    }

    fun getEndDate(): String{
        return endDate.text.toString()
    }

    private fun setDate(view: TextView, date: Calendar) {
        val dateFormat = SimpleDateFormat(DATE_FORMAT, Locale.getDefault())
        view.text = dateFormat.format(date.time)

        if (isStartLaterThanEnd()) {
            handleStartLaterThanEnd()
        }
    }

    private fun isStartLaterThanEnd(): Boolean {
        return startDate.text.toString() > endDate.text.toString()
    }

    private fun handleStartLaterThanEnd() {
        startDate.text = endDate.text
    }

    private fun setDateTextViewOnClickListener(textView: TextView) {
        val calendar = Calendar.getInstance()
        val dateSetListener = DatePickerDialog.OnDateSetListener { _, year, month, dayOfMonth ->
            calendar.set(year, month, dayOfMonth)
            setDate(textView, calendar)
            onDateSetListener?.invoke()
        }

        textView.setOnClickListener {
            DatePickerDialog(
                context,
                dateSetListener,
                calendar.get(Calendar.YEAR),
                calendar.get(Calendar.MONTH),
                calendar.get(Calendar.DAY_OF_MONTH),
            ).show()
        }
    }

    private fun init() {
        val view = View.inflate(context, R.layout.date_period_view, this)
        startDate = view.startDate
        endDate = view.endDate

        setDefaultDates()
    }

    private fun setDefaultDates(){
        val oneMonthAgoDate = Calendar.getInstance()
        val currentDate = Calendar.getInstance()
        val previousMonth = currentDate.get(Calendar.MONTH) - 1

        oneMonthAgoDate.set(
                currentDate.get(Calendar.YEAR),
                previousMonth,
                currentDate.get(Calendar.DAY_OF_MONTH)
        )

        setDate(startDate, oneMonthAgoDate)
        setDate(endDate, currentDate)
    }
}
```

FragmentType.kt

```
package com.example.budgetcontrol.enum
```

```
enum class FragmentType {
    BUDGET,
    INCOME,
    COSTS,
    TARGET,
    GAUSS_NUMBERS,
    INFORMATION
}
```

gauss_number_info_dialog.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/gauss_number_info_dialog_background"
    android:gravity="center_vertical"
    android:orientation="vertical"
    android:paddingHorizontal="15dp"
    android:paddingTop="15dp"
    android:paddingBottom="25dp">

    <ImageView
        android:id="@+id/gaussNumberInfoDialogCloseButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="end"
        android:src="@drawable/ic_close" />

    <TextView
        style="@style/BlueMediumTextStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dp"
        android:text="@string/method_essence_header"
        android:textSize="20sp" />

    <TextView
        style="@style/BlueRegularTextStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:lineHeight="24dp"
        android:text="@string/method_essence_info_text"
        android:textSize="20sp"
        tools:targetApi="p" />


</LinearLayout>
```

gauss_number_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="2dp">

    <TextView
        android:id="@+id/gaussNumberTextView"
        android:layout_width="@dimen/gauss_number_width"
        android:layout_height="@dimen/gauss_number_height"
        android:background="@drawable/gauss_number_background"
        android:fontFamily="@font/roboto_medium"
        android:gravity="center"
        android:text="1"
        android:textColor="@color/gauss_number_text_color"
        android:textSize="@dimen/gauss_number_text_size" />

    <ImageView
        android:id="@+id/collectedGaussNumberImageView"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:src="@drawable/collected_gauss_number"
        android:visibility="invisible" />

</FrameLayout>
```

gauss_numbers_fragment.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:id="@+id/seekBarLinearLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:orientation="horizontal"
        android:paddingHorizontal="20dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.07">

        <TextView
            android:id="@+id/collectedAmountTextView"
            style="@style/BlueMediumTextStyle"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginEnd="-20dp"
            android:layout_weight="2"
            android:gravity="center"
            android:text="0"
            android:textSize="@dimen/gauss_number_text_size" />

        <SeekBar
            android:id="@+id/gaussNumbersSeekBar"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="8"
            android:max="5050"
            android:min="0"
            android:progressDrawable="@drawable/seekbar_progress_line"
            android:thumb="@drawable/seekbar_thumb" />

        <TextView
            style="@style/BlueMediumTextStyle"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="-15dp"
            android:layout_weight="2"
            android:gravity="center"
            android:text="@string/fifty_fifty"
            android:textSize="@dimen/gauss_number_text_size" />

    </LinearLayout>

    <com.google.android.flexbox.FlexboxLayout
        android:id="@+id/gaussNumbersFlexBoxLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingHorizontal="10dp"
        app:flexDirection="row"
        app:flexWrap="wrap"
        app:justifyContent="center"
        app:layout_constraintBottom_toTopOf="@+id/gaussNumbersFragmentConfirmButton"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/seekBarLinearLayout" />

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

```
    style="@style/TextViewButtonStyle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/to_confirm"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.95" />

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

GaussNumbersFragment.kt

```
package com.example.budgetcontrol.fragment

import android.app.Dialog
import android.graphics.Color
import android.graphics.drawable.ColorDrawable
import android.os.Bundle
import android.view.*
import androidx.core.view.forEach
import androidx.fragment.app.Fragment
import com.example.budgetcontrol.MainActivity
import com.example.budgetcontrol.R
import com.example.budgetcontrol.db.BudgetControlDB
import com.example.budgetcontrol.db.model.Target
import com.example.budgetcontrol.dialog.ConfirmationDialog
import com.example.budgetcontrol.enum.FragmentType
import com.example.budgetcontrol.view.GaussNumberView
import kotlinx.android.synthetic.main.gauss_number_info_dialog.*
import kotlinx.android.synthetic.main.gauss_numbers_fragment.*
import kotlinx.android.synthetic.main.record_dialog.*
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch

class GaussNumbersFragment : Fragment() {

    companion object {
        private const val IS_NOT_COLLECTED = 0
        private const val IS_COLLECTED = 1
    }

    private var changedGaussNumbersMap = HashMap<Int, Boolean>()
    private var changedCollectedAmount = 0F

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {
        return inflater.inflate(R.layout.gauss_numbers_fragment, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        setupViews()
        setData()
        (activity as MainActivity).setToolbarTitle(getString(R.string.fifty_fifty))

        setHasOptionsMenu(true)
    }

    override fun onCreateOptionsMenu(menu: Menu, inflater: MenuInflater) {
        super.onCreateOptionsMenu(menu, inflater)
        inflater.inflate(R.menu.toolbar_menu, menu)
    }

    override fun onPrepareOptionsMenu(menu: Menu) {
        super.onPrepareOptionsMenu(menu)
        menu.findItem(R.id.toolbarMenuItemInfo).isVisible = true
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        return if (item.itemId == R.id.toolbarMenuItemInfo) {
            handleInfoButtonClick()
            true
        } else {
            super.onOptionsItemSelected(item)
```

```
    }
}

private fun handleInfoButtonClick() {
    val dialog = Dialog(requireContext())
    dialog.apply {
        val dialogView = layoutInflater.inflate(R.layout.gauss_number_info_dialog, null)
        setContentView(dialogView)
        window?.setBackgroundDrawable(ColorDrawable(Color.TRANSPARENT))
        gaussNumberInfoDialogCloseButton.setOnClickListener {
            dismiss()
        }
        show()
    }
}

private fun setupViews() {
    gaussNumbersSeekBar.isEnabled = false
    gaussNumbersFragmentConfirmButton.setOnClickListener(this::handleConfirmButtonClick)
}

private fun setData() {
    fillGaussNumbersLayout()
    setCollectedAmount()
}

private fun setCollectedAmount() {
    GlobalScope.launch {
        val targetCollectedAmount = BudgetControlDB.getInstance(requireContext())
                .targetDao()
                .getCollectedAmount(Target.GAUSS_NUMBER_TARGET_ID)
        changedCollectedAmount = targetCollectedAmount

        activity?.runOnUiThread {
            collectedAmountTextView.text = targetCollectedAmount.toInt().toString()
            gaussNumbersSeekBar.progress = targetCollectedAmount.toInt()

            if (isTargetAchieved(targetCollectedAmount)) {
                handleAmountCollected()
            }
        }
    }
}

private fun fillGaussNumbersLayout() {
    gaussNumbersFlexBoxLayout.removeAllViews()

    GlobalScope.launch {
        val gaussNumbersStatusesList = BudgetControlDB.getInstance(requireContext())
                .gaussNumberDao()
                .getStatuses()

        activity?.runOnUiThread {
            for (number in 1..100) {
                val gaussNumber = GaussNumberView(context, number)
                val index = number - 1
                val isCollected = gaussNumbersStatusesList[index]
                gaussNumber.setStatus(isCollected == IS_COLLECTED)

                gaussNumber.apply {
                    setOnClickListener(this@GaussNumbersFragment::handleGaussNumberViewClick)
                }
                gaussNumbersFlexBoxLayout.addView(gaussNumber)
            }
        }
    }
}

private fun handleGaussNumberViewClick(view: View) {
    val gaussNumber = view as GaussNumberView

    gaussNumber.apply {
        if (isCollected()) {
            val dialog = ConfirmationDialog(context, getString(R.string.you_want_to_cancel_contribution))
            dialog.apply {
                confirmButton.setOnClickListener {
                    changeGaussNumberStatus(gaussNumber)
```

```
                dismiss()
            }
            show()
        }
    } else {
        changeGaussNumberStatus(gaussNumber)
    }
  }
}

private fun changeGaussNumberStatus(gaussNumber: GaussNumberView) {
  gaussNumber.apply {
    val isCollected = !isCollected()
    setStatus(isCollected)
    changedGaussNumbersMap[getValue()] = isCollected

    when (isCollected) {
      true -> changedCollectedAmount += getValue()
      false -> changedCollectedAmount -= getValue()
    }
    gaussNumbersSeekBar.progress = changedCollectedAmount.toInt()
    collectedAmountTextView.text = changedCollectedAmount.toInt().toString()
  }
}

private fun handleConfirmButtonClick(view: View) {
  GlobalScope.launch {
    changedGaussNumbersMap.forEach {
      val value = it.key
      val isCollected = it.value
      BudgetControlDB.getInstance(requireContext())
          .gaussNumberDao()
          .updateStatus(value, if (isCollected) IS_COLLECTED else IS_NOT_COLLECTED)
    }
    BudgetControlDB.getInstance(requireContext())
        .targetDao()
        .updateCollectedAmount(Target.GAUSS_NUMBER_TARGET_ID, changedCollectedAmount)
  }
  if (isTargetAchieved(changedCollectedAmount)) {
    handleAmountCollected()
  } else {
    (activity as MainActivity).navigateToFragment(FragmentType.BUDGET)
  }

}

private fun handleAmountCollected() {
  val dialog = ConfirmationDialog(
      requireContext(),
      getString(R.string.gauss_numbers_are_collected)
  )
  dialog.apply {
    confirmButton.setOnClickListener {
      startCollectingAgain()
      dismiss()
    }
    show()
  }
}

private fun startCollectingAgain() {
  val db = BudgetControlDB.getInstance(requireContext())
  GlobalScope.launch {
    for (number in 1..100) {
      db.gaussNumberDao().updateStatus(number, IS_NOT_COLLECTED)
    }
    val target = Target(
        Target.GAUSS_NUMBER_TARGET_ID,
        Target.GAUSS_NUMBER_TARGET_DESCRIPTION,
        Target.GAUSS_NUMBER_TARGET_AMOUNT,
        0
    )
    db.targetDao().insert(target)
    activity?.runOnUiThread {
      setData()
    }
  }
```

```
    }

    private fun isTargetAchieved(collectedAmount: Float): Boolean {
        return collectedAmount >= Target.GAUSS_NUMBER_TARGET_AMOUNT
    }
}
```

GaussNumberView.kt

```
package com.example.budgetcontrol.view

import android.content.Context
import android.view.View
import android.widget.LinearLayout
import android.widget.TextView
import com.example.budgetcontrol.R
import kotlinx.android.synthetic.main.gauss_number_view.view.*

class GaussNumberView(context: Context?, value: Int) : LinearLayout(context) {

    private var gaussNumber: TextView
    private var isCollected: Boolean = false

    init {
        val view = View.inflate(context, R.layout.gauss_number_view, this)
        gaussNumber = view.gaussNumberTextView
        gaussNumber.text = value.toString()
    }

    fun getValue(): Int {
        return gaussNumber.text.toString().toInt()
    }

    fun setStatus(isCollected: Boolean) {
        collectedGaussNumberImageView.visibility = if (isCollected) VISIBLE else INVISIBLE
        this.isCollected = isCollected
    }

    fun isCollected(): Boolean {
        return isCollected
    }
}
```

IncomeCostsFragment.kt

```
package com.example.budgetcontrol.fragment

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import com.example.budgetcontrol.MainActivity
import com.example.budgetcontrol.MainActivity.Companion.BUDGET_COMPONENT
import com.example.budgetcontrol.R
import com.example.budgetcontrol.db.BudgetControlDB
import com.example.budgetcontrol.db.model.Transaction
import com.example.budgetcontrol.dialog.RecordDialog
import com.example.budgetcontrol.enum.BudgetComponent
import com.example.budgetcontrol.view.TransactionView
import kotlinx.android.synthetic.main.income_costs_fragment.*
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch
import kotlin.math.abs

class IncomeCostsFragment : Fragment() {

    private lateinit var budgetComponent: BudgetComponent
    private lateinit var updatingTransaction: Transaction

    override fun onCreateView(
            inflater: LayoutInflater,
            container: ViewGroup?,
```

```
        savedInstanceState: Bundle?
): View? {
    return inflater.inflate(R.layout.income_costs_fragment, container, false)
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    budgetComponent = arguments?.getSerializable(BUDGET_COMPONENT) as BudgetComponent
    (activity as MainActivity).setToolbarTitle(getString(when (budgetComponent) {
        BudgetComponent.INCOME -> R.string.income
        BudgetComponent.COSTS -> R.string.costs
    }))
    fetchData()
    setupCounter()
    incomeCostsDatePeriodView.setDatePickerOnDateSetListener(this::handleDateSet)
}

private fun handleDateSet(){
    incomeCostsDatePeriodView.apply {
        fetchData(getStartDate(), getEndDate())
    }
}

private fun fetchData(startDate: String? = null, endDate: String? = null) {
    setCounterValue(startDate, endDate)
    fillTransactionContainer(startDate, endDate)
}

private fun setCounterValue(startDate: String?, endDate: String?) {
    GlobalScope.launch {
        val amount = abs(getTotalIncomeCostsAmount(budgetComponent, startDate, endDate))
        activity?.runOnUiThread {
            incomeCostsYellowCounter.text = getString(
                    R.string.income_costs_value_with_grivnas_placeholder,
                    when (budgetComponent) {
                        BudgetComponent.INCOME -> "+"
                        BudgetComponent.COSTS -> "-"
                    },
                    amount
            )
        }
    }
}

private fun fillTransactionContainer(startDate: String?, endDate: String?) {
    transactionsListLinearLayout.removeAllViews()

    val transactionDao =  BudgetControlDB.getInstance(requireContext()).transactionDao()
    GlobalScope.launch {
        val transactionList = when(budgetComponent){
            BudgetComponent.INCOME ->{
                if(startDate.isNullOrBlank() || endDate.isNullOrBlank())
                    transactionDao.getAllIncome()
                else
                    transactionDao.getAllIncomeByDate(startDate, endDate)
            }
            BudgetComponent.COSTS -> {
                if(startDate.isNullOrBlank() || endDate.isNullOrBlank())
                    transactionDao.getAllCosts()
                else
                    transactionDao.getAllCostsByDate(startDate, endDate)
            }
        }
        activity?.runOnUiThread {
            transactionList.forEach { transaction ->
                val transactionView = TransactionView(
                        requireContext(),
                        transaction,
                        this@IncomeCostsFragment::handleEditButtonClick
                )
                transactionsListLinearLayout.addView(transactionView, 0)
            }
        }
    }
}

private fun setupCounter() {
```

```
    incomeCostsYellowCounter.setOnClickListener {
        val dialog = RecordDialog(
                requireContext(),
                when (budgetComponent) {
                    BudgetComponent.INCOME -> context?.getString(R.string.income)
                    BudgetComponent.COSTS -> context?.getString(R.string.costs)
                },
                when (budgetComponent) {
                    BudgetComponent.INCOME -> context?.getString(R.string.source)
                    BudgetComponent.COSTS -> context?.getString(R.string.costs_edit_text_hint)
                },
                this::recordTransaction
        )
        dialog.setOnDismissListener {
            //todo add the lest transaction
        }
        dialog.show()
    }
}

private fun recordTransaction(amount: Float, description: String) {
    val transaction = Transaction(
            when (budgetComponent) {
                BudgetComponent.INCOME -> amount
                BudgetComponent.COSTS -> (-amount)
            },
            description
    )

    GlobalScope.launch {
        BudgetControlDB.getInstance(requireContext())
                .transactionDao()
                .insert(transaction)
        activity?.runOnUiThread {
            fetchData()
        }
    }
}

//TODO in what way to prevent code duplication??????
private fun getTotalIncomeCostsAmount(budgetComponent: BudgetComponent, startDate: String?, endDate: String?): Float {
    var totalAmount = 0f
    val transactionDao = BudgetControlDB.getInstance(requireContext()).transactionDao()
    val allAmounts = when (budgetComponent) {
        BudgetComponent.INCOME -> {
            if (startDate.isNullOrBlank() || endDate.isNullOrBlank())
                transactionDao.getAllIncomeAmounts()
            else
                transactionDao.getAllIncomeAmountsByDate(startDate, endDate)
        }
        BudgetComponent.COSTS -> {
            if (startDate.isNullOrBlank() || endDate.isNullOrBlank())
                transactionDao.getAllCostsAmounts()
            else
                transactionDao.getAllCostsAmountsByDate(startDate, endDate)
        }
    }
    allAmounts.forEach { value ->
        totalAmount += value
    }
    return totalAmount
}

private fun handleEditButtonClick(transaction: Transaction) {
    updatingTransaction = transaction

    val dialog = RecordDialog(
            requireContext(),
            when (budgetComponent) {
                BudgetComponent.INCOME -> context?.getString(R.string.income)
                BudgetComponent.COSTS -> context?.getString(R.string.costs)
            },
            when (budgetComponent) {
                BudgetComponent.INCOME -> context?.getString(R.string.source)
                BudgetComponent.COSTS -> context?.getString(R.string.costs_edit_text_hint)
            },
            this::updateTransaction
```

```
        )
        dialog.apply {
            setOnDismissListener {
                fetchData()
            }
            show()
        }
    }

    private fun updateTransaction(amount: Float, description: String){
        val newTransaction = updatingTransaction.copy(
                amount = when (budgetComponent) {
                    BudgetComponent.INCOME -> amount
                    BudgetComponent.COSTS -> (-amount)
                },
                comment = description
        )

        GlobalScope.launch {
            BudgetControlDB.getInstance(requireContext())
                    .transactionDao()
                    .update(newTransaction)
        }
    }
}
```

InformationFragment.kt

```
package com.example.budgetcontrol.fragment

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import com.example.budgetcontrol.MainActivity
import com.example.budgetcontrol.R

class InformationFragment : Fragment() {

    override fun onCreateView(
            inflater: LayoutInflater,
            container: ViewGroup?,
            savedInstanceState: Bundle?
    ): View? {
        return inflater.inflate(R.layout.information_fragment, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        (activity as MainActivity).setToolbarTitle(getString(R.string.information))
    }
}
```

MainActivity.kt

```
package com.example.budgetcontrol

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.appcompat.widget.Toolbar
import androidx.navigation.NavController
import androidx.navigation.Navigation
import com.example.budgetcontrol.enum.BudgetComponent
import com.example.budgetcontrol.enum.FragmentType
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {
    //todo add progress view while fetching data

    companion object{
        const val BUDGET_COMPONENT: String = "BUDGET_COMPONENT"
    }
```

```
    private lateinit var navController: NavController
    private lateinit var toolbar: Toolbar

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        setupToolbar()
        navController = Navigation.findNavController(this, R.id.fragment_container)
    }

    fun navigateToFragment(fragment: FragmentType) {
        when (fragment) {
            FragmentType.BUDGET -> navController.navigate(R.id.budgetFragment)
            FragmentType.INCOME -> navigateToIncomeCostsFragment(BudgetComponent.INCOME)
            FragmentType.COSTS -> navigateToIncomeCostsFragment(BudgetComponent.COSTS)
            FragmentType.TARGET -> navController.navigate(R.id.targetFragment)
            FragmentType.GAUSS_NUMBERS -> navController.navigate(R.id.gaussNumbersFragment)
            FragmentType.INFORMATION -> navController.navigate(R.id.informationFragment)
        }
    }

    fun setToolbarTitle(title: String) {
        toolbar.title = title
    }

    private fun setupToolbar() {
        this.toolbar = budgetControlToolbar
        setToolbarTitle(getString(R.string.budget))
        setSupportActionBar(toolbar)
    }

    private fun navigateToIncomeCostsFragment(budgetComponent: BudgetComponent) {
        val bundle = Bundle()
        bundle.putSerializable(BUDGET_COMPONENT, budgetComponent)
        navController.navigate(R.id.incomeCostsFragment, bundle)
    }
}
```

Record.kt

```
package com.example.budgetcontrol.enum

enum class Record {
    INCOME_TRANSACTION,
    COSTS_TRANSACTION,
    NEW_TARGET
}
```

TargetFragment.kt

```
package com.example.budgetcontrol.fragment

import android.os.Bundle
import android.view.*
import androidx.core.view.forEach
import androidx.fragment.app.Fragment
import com.example.budgetcontrol.MainActivity
import com.example.budgetcontrol.R
import com.example.budgetcontrol.db.BudgetControlDB
import com.example.budgetcontrol.db.model.Target
import com.example.budgetcontrol.dialog.ConfirmationDialog
import com.example.budgetcontrol.dialog.RecordDialog
import com.example.budgetcontrol.enum.FragmentType
import kotlinx.android.synthetic.main.target_fragment.*
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch
import java.util.*

class TargetFragment: Fragment() {

    //todo add target ids consts (two)
```

```
override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
): View? {
    return inflater.inflate(R.layout.target_fragment, container, false)
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    setHasOptionsMenu(true)
    (activity as MainActivity).setToolbarTitle(getString(R.string.target))

    fetchData()
    setConfirmButtonOnClickListener()
}

override fun onCreateOptionsMenu(menu: Menu, inflater: MenuInflater) {
    super.onCreateOptionsMenu(menu, inflater)
    inflater.inflate(R.menu.toolbar_menu, menu)
}

override fun onPrepareOptionsMenu(menu: Menu) {
    super.onPrepareOptionsMenu(menu)
    menu.findItem(R.id.toolbarMenuItemEdit).isVisible = true
}

override fun onOptionsItemSelected(item: MenuItem): Boolean {
    return if (item.itemId == R.id.toolbarMenuItemEdit) {
        handleEditButtonClick()
        true
    } else {
        super.onOptionsItemSelected(item)
    }
}

private fun handleEditButtonClick() {
    RecordDialog(
            requireContext(),
            context?.getString(R.string.target),
            context?.getString(R.string.target)?.toLowerCase(Locale.getDefault()),
            this::updateTarget
    ).show()
}

private fun updateTarget(newAmount: Float, description: String) {
    val targetDao = BudgetControlDB.getInstance(requireContext()).targetDao()

    GlobalScope.launch {
        val currentTarget = targetDao.getTarget(Target.MAIN_TARGET_ID)
        if (newAmount < currentTarget.collectedAmount) {
            handleTargetAchievement()
        } else {
            val newTarget = currentTarget.copy(
                    amount = newAmount,
                    description = description
            )

            BudgetControlDB.getInstance(requireContext())
                    .targetDao()
                    .update(newTarget)

            activity?.runOnUiThread {
                refreshTargetInfo()
            }
        }
    }
}


private fun fetchData() {
    GlobalScope.launch {
        val isTargetAdded = BudgetControlDB.getInstance(requireContext())
                .targetDao()
                .isAdded(Target.MAIN_TARGET_ID)
```

```
        activity?.runOnUiThread {
            if (isTargetAdded) {
                refreshTargetInfo()
            } else {
                createTarget()
            }
        }
    }
}

private fun createTarget() {
    val dialog = RecordDialog(
        requireContext(),
        context?.getString(R.string.target),
        context?.getString(R.string.target)?.toLowerCase(Locale.getDefault()),
        this::recordTarget
    )
    dialog.setOnDismissListener {
        GlobalScope.launch {
            val isTargetAdded = BudgetControlDB.getInstance(requireContext())
                .targetDao().isAdded(Target.MAIN_TARGET_ID)
            if (!isTargetAdded) {
                activity?.runOnUiThread {
                    (activity as MainActivity).navigateToFragment(FragmentType.BUDGET)
                }
            }
        }
    }
    dialog.show()
}

private fun refreshTargetInfo() {
    val targetDao = BudgetControlDB.getInstance(requireContext()).targetDao()
    var description: String
    var amount: Float
    var collectedAmount: Float
    var leftAmount: Float

    GlobalScope.launch {
        amount = targetDao.getAmount(Target.MAIN_TARGET_ID)
        collectedAmount = targetDao.getCollectedAmount(Target.MAIN_TARGET_ID)
        description = targetDao.getDescription(Target.MAIN_TARGET_ID)
        leftAmount = amount - collectedAmount

        activity?.runOnUiThread {
            targetCollectedAmountTextView.text = collectedAmount.toString()
            targetAmountTextView.text = amount.toString()
            leftAmountTextView.text = leftAmount.toString()
            targetTextView.text = description
        }
    }
}

private fun setConfirmButtonOnClickListener() {
    confirmTargetContributionButton.setOnClickListener {
        if (!targetContributionEditText.text.isNullOrBlank()) {
            val targetDao = BudgetControlDB.getInstance(requireContext()).targetDao()

            GlobalScope.launch {
                val currentCollectedAmount = targetDao.getCollectedAmount(Target.MAIN_TARGET_ID)
                val contribution = targetContributionEditText.text.toString().toFloat()
                targetContributionEditText.text.clear()
                val newCollectedAmount = currentCollectedAmount + contribution
                val targetAmount = targetDao.getAmount(Target.MAIN_TARGET_ID)

                if (newCollectedAmount >= targetAmount) {
                    handleTargetAchievement()
                } else {
                    targetDao.updateCollectedAmount(Target.MAIN_TARGET_ID, newCollectedAmount)

                    activity?.runOnUiThread {
                        refreshTargetInfo()
                    }
                }
            }
        }
    }
```

```
    }

    private fun handleTargetAchievement() {
        val targetDao = BudgetControlDB.getInstance(requireContext()).targetDao()
        targetDao.delete(Target.MAIN_TARGET_ID)

        activity?.runOnUiThread {
            val dialog = ConfirmationDialog(
                requireContext(),
                getString(R.string.you_achieve_goal)
            )
            dialog.apply {
                confirmButton.setOnClickListener {
                    createTarget()
                    dismiss()
                }
                closeButton.setOnClickListener {
                    (activity as MainActivity).navigateToFragment(FragmentType.BUDGET)
                    dismiss()
                }
                show()
            }
            refreshTargetInfo()
        }
    }

    private fun recordTarget(amount: Float, description: String) {
        val target = Target(
            Target.MAIN_TARGET_ID,
            description,
            amount,
            0
        )

        GlobalScope.launch {
            BudgetControlDB.getInstance(requireContext())
                .targetDao()
                .insert(target)
            activity?.runOnUiThread {
                refreshTargetInfo()
            }
        }
    }
}
```

TransactionView.kt

```
package com.example.budgetcontrol.view

import android.annotation.SuppressLint
import android.content.Context
import android.view.View
import android.widget.LinearLayout
import com.example.budgetcontrol.R
import com.example.budgetcontrol.db.model.Transaction
import com.example.budgetcontrol.enum.BudgetComponent
import kotlinx.android.synthetic.main.transaction_view.view.*
import kotlin.math.abs

@SuppressLint("ViewConstructor")
class TransactionView(context: Context, transaction: Transaction, private val editButtonClickHandler: (transaction: Transaction) -> Unit) :
LinearLayout(context) {

    private var budgetComponent: BudgetComponent

    init {
        View.inflate(context, R.layout.transaction_view, this)

        budgetComponent = when (transaction.amount > 0) {
            true -> BudgetComponent.INCOME
            false -> BudgetComponent.COSTS
        }

        transactionValue.text = context.getString(
            R.string.income_costs_value_placeholder,
            when (budgetComponent) {
```

```
                BudgetComponent.INCOME -> "+"
                BudgetComponent.COSTS -> "-"
            },
            abs(transaction.amount)
        )
        transactionComment.text = transaction.comment
        transactionDate.text = transaction.date

        transactionEditButton.setOnClickListener {
            editButtonClickHandler(transaction)
        }
    }


}
```