Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ЗБОРУ ТА АНАЛІЗУ ВИТРАТ НА ЕНЕРГОРЕСУРСИ З ВИКОРИСТАННЯМ KOTLIN

DEVELOPMENT OF A MOBILE APPLICATION FOR COLLECTING AND ANALYZING ENERGY RESOURCE EXPENSES USING KOTLIN

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-41
Талашук Тарас Миколайович

Керівник:
Гордєєв Олександр Олександрович

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

_____

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«20» 12 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Талашуку Тарасу Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *«Розробка мобільного додатку збору та аналізу витрат на енергоресурси з використанням Kotlin»*
Керівник роботи: *професор Гордєєв О. О.*
затверджені наказом закладу вищої освіти від *«19» грудня 2024 р.* № *474/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*
3. Вихідні дані до роботи: *Kotlin, Android Studio, Android SDK, Gradle, Room, MPAndroidChart, Jetpack ViewModel, LiveData, SharedPreferences методичні вказівки до виконання кваліфікаційної роботи бакалавра.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *аналіз предметної області, постановка завдань, визначення вимог, проєктування архітектури, реалізація інтерфейсу, програмна логіка, розробка бази даних, тестування, підготовка документації.*
5. Перелік графічного матеріалу: *12 рисунків, 2 таблиці, 1 додаток.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	/Гордєєв О. О.		
Специфікація вимог до розробленої системи	/Гордєєв О. О.		
Розробка об'єкта проектування	/Гордєєв О. О.		
Нормоконтроль	Вознюк А. В.		
Гарант ОП	Ліщина Н. М.		
Показник запозичень тексту	1,76 %		
Академічна доброчесність	/Гордєєв О. О.		

7. Дата видачі завдання *«20» грудня 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 04.02.2025 р.	Виконано
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 01.03.2025 р.	Виконано
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 15.03.2025 р.	Виконано
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 29.03.2025 р.	Виконано
5	Практична реалізація об'єкта проектування та розробка бази даних	до 26.04.2025 р.	Виконано
6	Тестування та налагодження об'єкта проектування	до 03.05.2025 р.	Виконано
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 10.06.2025 р.	Виконано

Здобувач вищої освіти _____ Тарас ТАЛАШУК

/Керівник кваліфікаційної роботи _____ Олександр ГОРДЄЄВ

АНОТАЦІЯ

Талашук Т. М. Розробка мобільного додатку збору та аналізу витрат на енергоресурси з використанням Kotlin. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатку.

У роботі проведено аналіз предметної області, сформульовано функціональні та нефункціональні вимоги до програмного продукту, побудовано UML-діаграми. Програмне забезпечення реалізовано з використанням мови програмування Kotlin, середовища Android Studio, бібліотек Room для локального зберігання даних, MPAndroidChart для візуалізації та архітектурних компонентів Jetpack. Проведено ручне функціональне та інтеграційне тестування, виявлені помилки усунуті. У роботі наведено системні вимоги, обґрунтовано безпекові рішення (використання захищеного середовища Android, ProGuard для обфускації коду), розроблено документацію та інструкцію користувача. Результати роботи мають практичну цінність для побутових споживачів, малих підприємств або енергоменеджерів та можуть бути розширені з урахуванням перспектив розвитку технологій мобільного моніторингу ресурсів.

Ключові слова: мобільний додаток, Kotlin, витрати енергоресурсів, Room, Android, графіки, база даних, UI/UX, облік споживання, енергоефективність.

ABSTRACT

Talashuk T. Development of a mobile application for tracking and analyzing energy consumption using Kotlin. Manuscript. Bachelor's qualification thesis of the Educational Program "Software Engineering" in specialty "Software Engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's qualification thesis consists of an introduction, three chapters, conclusions, a list of references, and appendices.

The work includes an analysis of the subject area, formulation of functional and non-functional requirements for the software product, and development of UML diagrams that reflect the architectural structure of the application. The software is implemented using the Kotlin programming language, Android Studio development environment, Room library for local data storage, MPAndroidChart for visualization, and Jetpack architectural components. Manual functional and integration testing were carried out, and identified issues were resolved. The thesis also provides the system requirements, justifies implemented security measures (use of Android's secure environment and code obfuscation via ProGuard), and includes comprehensive user documentation and instructions. The results of this work offer practical value for individual households, small businesses, or energy managers and can be extended considering future advancements in mobile resource monitoring technologies.

Keywords: mobile application, Kotlin, energy consumption, Room, Android, charts, database, UI/UX, consumption tracking, energy efficiency.

ЗМІСТ

ВСТУП 7
РОЗДІЛ 1 АНАЛІЗ СУЧАСНИХ РІШЕНЬ У СФЕРІ МОНІТОРИНГУ ВИТРАТ ЕНЕРГОРЕСУРСІВ ТА ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ 10
1.1 Аналіз сучасного стану проблеми 10
1.2 Постановка завдання на кваліфікаційну роботу бакалавра 13
Висновки до розділу 1 14
РОЗДІЛ 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ 16
2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення 16
2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання 20
Висновки до розділу 2 24
РОЗДІЛ 3 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ЗБОРУ ТА АНАЛІЗУ ВИТРАТ НА ЕНЕРГОРЕСУРСИ 26
3.1 Практична реалізація об’єкта проєктування 26
3.2 Тестування та налагодження інформаційно-комп’ютерної системи 31
3.3 Розробка бази даних 33
3.4 Захист інформаційно-комп’ютерної системи 36
3.5 Розробка документації для програмного продукту 37
Висновки до розділу 3 41
ВИСНОВКИ 43
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 46
ДОДАТКИ 48

ВСТУП

Актуальність теми. У сучасних умовах глобальної цифровізації та зростаючого значення енергозбереження надзвичайно актуальним постає завдання контролю та аналізу витрат на енергоресурси як на побутовому, так і на підприємницькому рівнях. Ефективне управління витратами на електроенергію, воду, газ чи опалення стає необхідною складовою відповідального споживання ресурсів та зменшення фінансових втрат. Разом із розвитком мобільних технологій, особливо під операційну систему Android, відкриваються широкі можливості для створення персональних цифрових інструментів, які дозволяють автоматизувати облік енергоспоживання та здійснювати аналітику на основі зібраних даних.

Наразі існує низка мобільних додатків, що частково вирішують завдання моніторингу витрат енергоресурсів, однак більшість із них орієнтовані на іноземні ринки, мають обмежену підтримку локалізації або не враховують специфіку тарифікації та показників, актуальних для українських користувачів. Крім того, часто бракує інструментів гнучкого аналізу, візуалізації змін, прогнозування витрат або підтримки збереження історії даних для подальшого прийняття рішень.

У світовій практиці спостерігається стійка тенденція до розробки «розумних» мобільних застосунків з елементами машинного навчання, візуалізації даних, інтеграції з розумними лічильниками та хмарними сервісами. Застосування таких підходів дозволяє підвищити рівень енергоефективності, надати користувачеві прозору картину споживання ресурсів та сприяти формуванню енергетично свідомої поведінки.

Тому актуальність цієї роботи полягає в необхідності створення сучасного, зручного та адаптованого до локальних умов мобільного додатку, який дозволяє користувачам вести облік, проводити аналіз витрат на енергоресурси, переглядати статистику та динаміку змін у зручному форматі.

Розробка такого продукту на мові програмування Kotlin, яка є офіційною для Android-розробки, забезпечує продуктивність, безпеку та актуальність рішення.

Метою кваліфікаційної роботи є проєктування та розробка мобільного додатку для збору та аналізу витрат на енергоресурси з використанням Kotlin.

Об’єктом кваліфікаційної роботи бакалавра є процес автоматизованого обліку та аналізу витрат енергоресурсів у побутовому середовищі за допомогою мобільних застосунків.

Предметом кваліфікаційної роботи бакалавра є програмні засоби, алгоритми обробки даних та інтерфейсні рішення, реалізовані у мобільному додатку для обліку споживання електроенергії, води та газу на платформі Android з використанням мови Kotlin.

Для досягнення мети в роботі були поставлені наступні завдання:

1) здійснити аналіз предметної області та обґрунтувати актуальність створення мобільного додатку для обліку витрат енергоресурсів;

2) визначити функціональні та нефункціональні вимоги до програмного забезпечення, побудувати UML-діаграми варіантів використання, класів, активностей і компонентів для моделювання архітектури системи;

3) обґрунтувати вибір інструментів, методів і технологій реалізації мобільного додатку, включаючи мову програмування Kotlin, бібліотеки Room і MPAndroidChart, архітектурні компоненти Jetpack, а також засоби побудови графічного інтерфейсу;

4) реалізувати мобільний додаток EcoTrack з підтримкою введення показників, розрахунку витрат, збереження даних та побудови графіків;

5) провести тестування та налагодження застосунку, встановити мінімальні системні вимоги, оцінити продуктивність і усунути виявлені помилки;

6) розробити структуру локальної бази даних за допомогою бібліотеки Room та забезпечити її інтеграцію в систему;

7) описати реалізовані заходи захисту мобільного додатку, включаючи обмеження доступу до локальних даних і використання ProGuard для захисту коду;

8) підготувати супровідну документацію до мобільного застосунку, зокрема інструкцію користувача, опис інтерфейсу, інструкції з встановлення та мінімальні технічні вимоги.

Результати даної розробки можуть бути використані як індивідуальними користувачами для домашнього обліку витрат, так і підприємствами малого бізнесу або органами енергоменеджменту для ведення спрощеного моніторингу споживання ресурсів. Робота має міждисциплінарний характер і пов’язана з розробками у галузях енергетики, інформаційних технологій, управління ресурсами та сталого розвитку.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНИХ РІШЕНЬ У СФЕРІ МОНІТОРИНГУ ВИТРАТ ЕНЕРГОРЕСУРСІВ ТА ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

У сучасних умовах підвищення тарифів на енергоресурси та глобального фокусу на енергоефективність питання обліку споживання ресурсів набуває особливої актуальності. Для приватних домогосподарств, комунальних об’єктів та малого бізнесу важливо мати доступ до зручних інструментів, які дозволяють оперативно відслідковувати витрати на світло, воду та газ, формувати історію показників і прогнозувати витрати на основі введених даних. Однак, попри попит, більшість наявних рішень не повністю задовольняє потреби локального користувача, зокрема – українського.

У контексті правового регулювання сфери житлово-комунальних послуг важливим джерелом виступає Закон України «Про житлово-комунальні послуги» № 2189-VIII [1], що визначає загальні засади надання, обліку та оплати комунальних ресурсів. У законі підкреслюється необхідність впровадження облікових систем, заснованих на принципах точності, своєчасності та доступності інформації для споживача. Зокрема, визначено, що споживач має право на отримання повної інформації про обсяг спожитих послуг, а постачальники повинні забезпечити облік за допомогою технічних засобів із щомісячним контролем. Відповідно, нормативне середовище підтримує потребу в автоматизованих цифрових рішеннях, що відповідають сучасним вимогам до прозорості та зручності.

Проблематику автоматизації обліку в комунальній сфері розкрито у дослідженні Лучика С. Д. та Бруневича Х. С. [2], де відзначено, що традиційні методи обліку витрат комунальних послуг є застарілими та потребують цифрової трансформації. Автори підкреслюють необхідність розробки зручних

користувацьких систем, які б дозволяли вести персоніфікований облік споживання.

Практичні методичні рекомендації щодо впровадження інформаційної системи обліку розрахунків за комунальні послуги представлено у роботі Терошиної І. М. [3]. Авторка пропонує концепцію впровадження систем обліку в муніципальних підприємствах, що дозволяє мінімізувати кількість механічних помилок та підвищити зручність для населення в отриманні інформації про спожиті послуги.

Актуальний підхід до розробки інформаційної системи на сучасній платформі Salesforce викладено в роботі Казимир М. М. та Басюк Т. М. [4]. Дослідниці описують архітектуру та функціональність проєкту з автоматизації розрахунків за комунальні послуги, орієнтованого на спрощення облікових операцій, що підтверджує високий інтерес до теми створення цифрових рішень для комунального сектору.

Сучасним прикладом прикладної реалізації автоматизації у сфері обліку комунальних послуг є програмний комплекс «ПК ІТА-2» [5], який демонструє функціональні можливості з нарахування оплати, ведення взаєморозрахунків та обліку платежів з абонентами. Згідно з технічною документацією, система підтримує облік у багатоформатному вигляді, інтеграцію з лічильниками, створення звітів і аналіз боргового навантаження. Наявність таких рішень свідчить про широке впровадження цифрових платформ у комунальному секторі та підтверджує затребуваність мобільних і десктопних інструментів для автоматизованого моніторингу споживання ресурсів.

Загалом, проаналізовані джерела засвідчують значний інтерес до автоматизації обліку енергоресурсів, однак у більшості робіт увага приділяється централізованим системам або рішенням для підприємств. Натомість існує потреба у створенні індивідуальних мобільних додатків, що дозволяють кінцевому користувачу зручно здійснювати облік витрат, проводити аналіз споживання та мати доступ до візуалізованої статистики у

режимі реального часу. Це і обумовлює доцільність обраного напряму дослідження.

На ринку присутні такі аналоги, як Energy Consumption Analyzer, Meter Readings, My Usage, Smart Meters, які дозволяють фіксувати показники, обчислювати витрати та будувати графіки. Основними перевагами цих рішень є зрозумілий інтерфейс, мультиплатформність та базова візуалізація. Водночас більшість із них:

1) не адаптовані до українських тарифів або не дозволяють вільно їх редагувати;

2) не підтримують українську мову інтерфейсу;

3) не враховують специфіку національних комунальних облікових одиниць;

4) не зберігають дані локально або не мають зручного способу експорту/імпорту.

Проведений патентний пошук також показав, що більшість зареєстрованих рішень орієнтовані або на складні енергоменеджерські системи, або є частинами великих комунальних сервісів, які вимагають централізованого підключення до «розумних» лічильників. У той же час, звичайні споживачі часто використовують записи вручну, в електронних таблицях або блокнотах, що не дозволяє виконувати ефективний аналіз.

Відповідно, актуальним є створення нового програмного продукту – мобільного додатку, який дозволяє враховувати потреби локального користувача, підтримує ручне введення показників, розрахунок вартості відповідно до введених тарифів та візуалізацію витрат у зручному графічному форматі. Обрана платформа Android є найпоширенішою серед користувачів в Україні, а мова програмування Kotlin – сучасним, рекомендованим інструментом для розробки Android-застосунків.

Запропонований підхід дозволяє реалізувати недороге, автономне, інтуїтивно зрозуміле рішення для ведення обліку витрат на енергоресурси без потреби в зовнішніх пристроях або складній інфраструктурі. Це обґрунтовує

доцільність створення нової системи, адаптованої до реальних побутових потреб споживача.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою даної кваліфікаційної роботи є розробка мобільного застосунку для платформи Android, що дозволяє здійснювати облік та аналіз витрат на основні комунальні енергоресурси (електроенергію, воду, газ). Для досягнення поставленої мети необхідно вирішити такі завдання:

1) здійснити аналіз предметної області та обґрунтувати актуальність створення мобільного додатку для обліку витрат енергоресурсів;

2) визначити функціональні та нефункціональні вимоги до програмного забезпечення, побудувати UML-діаграми варіантів використання, класів, активностей і компонентів для моделювання архітектури системи;

3) обґрунтувати вибір інструментів, методів і технологій реалізації мобільного додатку, включаючи мову програмування Kotlin, бібліотеки Room і MPAndroidChart, архітектурні компоненти Jetpack, а також засоби побудови графічного інтерфейсу;

4) реалізувати мобільний додаток EcoTrack з підтримкою введення показників, розрахунку витрат, збереження даних та побудови графіків;

5) провести тестування та налагодження застосунку, встановити мінімальні системні вимоги, оцінити продуктивність і усунути виявлені помилки;

6) розробити структуру локальної бази даних за допомогою бібліотеки Room та забезпечити її інтеграцію в систему;

7) описати реалізовані заходи захисту мобільного додатку, включаючи обмеження доступу до локальних даних і використання ProGuard для захисту коду;

8) підготувати супровідну документацію до мобільного застосунку, зокрема інструкцію користувача, опис інтерфейсу, інструкції з встановлення та мінімальні технічні вимоги.

Сформульовані завдання окреслюють повний цикл розробки мобільного додатку – від аналізу предметної області та визначення вимог до програмного забезпечення до створення інтерфейсу, реалізації функціональної логіки, тестування та документування. Їхнє поетапне виконання забезпечить досягнення поставленої мети роботи та дозволить створити ефективний програмний продукт для обліку та аналізу витрат на енергоресурси.

Висновки до розділу 1

У першому розділі було всебічно проаналізовано сучасний стан проблеми обліку витрат на енергоресурси в контексті побутового та комунального споживання. На основі аналізу нормативно-правових актів, зокрема Закону України «Про житлово-комунальні послуги», а також досліджень вітчизняних і зарубіжних авторів, встановлено необхідність удосконалення існуючих підходів до ведення обліку, контролю та аналізу споживання електроенергії, води й газу. Особливу увагу приділено огляду актуальних програмних рішень, що представлені на ринку мобільних застосунків. Виявлено, що більшість із них мають ряд функціональних обмежень: обмежена підтримка української мови, неможливість редагування тарифів відповідно до реальних умов, відсутність візуалізації даних та недостатній рівень зручності інтерфейсу для широкої аудиторії користувачів.

На основі порівняльного аналізу аналогічних продуктів і вивчення користувацьких потреб було виявлено наявну прогалину у сегменті простих, локалізованих, автономних рішень для ведення персонального енергетичного обліку. Це дозволило обґрунтувати актуальність створення нового інструменту, орієнтованого саме на українського споживача. У результаті було сформульовано мету кваліфікаційної роботи – розробити мобільний додаток на

базі Android, який дозволяє користувачу вести облік енергоресурсів, аналізувати динаміку витрат та управляти тарифами в зручному форматі.

Окрім того, визначено низку завдань, реалізація яких дозволить повністю досягти поставленої мети, включаючи аналіз вимог, розробку інтерфейсу, логіки обчислень, бази даних, візуалізації статистики та супровідної документації. Висновки першого розділу стали основою для подальшого проєктування архітектури мобільного додатку та реалізації його функціональних компонентів.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Розробка мобільного додатку збору та аналізу витрат на енергоресурси потребує комплексного підходу до аналізу вимог та моделювання системи. На першому етапі було визначено основну мету проєкту: створення Android-додатку, що дозволяє зручно вводити показники лічильників, розраховувати витрати, зберігати історію та візуалізувати дані у вигляді графіків.

Для побудови архітектури програмного забезпечення було обрано об’єктно-орієнтований підхід, що найбільш повно відповідає структурі мобільних додатків на Kotlin. Як основну модель проєктування використано UML-нотацію, яка дає змогу формалізовано описати структуру додатку, його функціональні компоненти та взаємозв’язки.

Для виявлення вимог використано методи:

1) аналізу предметної області (вивчення наявних аналогів, літератури, відгуків користувачів);

2) інтерв’ю з потенційними користувачами (збір функціональних побажань);

3) аналізу use-case сценаріїв (визначення основних випадків використання системи).

Цілі програмного продукту:

1) забезпечення обліку витрат електроенергії, води та газу;

2) проведення автоматизованого розрахунку вартості спожитих ресурсів;

3) надання користувачу статистичної інформації у вигляді графіків;

4) забезпечення простого редагування тарифів та перегляду історії витрат.

Функціональні вимоги:

1) введення даних про показники (дата, значення, тип ресурсу);

2) розрахунок вартості споживання відповідно до тарифу;

3) збереження історії у локальній базі даних;

4) побудова графіків витрат за типами ресурсів;

5) редагування тарифів користувачем;

6) UX-дизайн інтерфейсу з простим переходом між вкладками;

7) валідація введених даних (запобігання нереалістичним значенням);

8) темна тема та адаптивність для різних розмірів екранів.

Нефункціональні вимоги:

1) продуктивність: швидкий час відповіді (<1 сек);

2) надійність: відсутність збоїв під час розрахунків;

3) портативність: підтримка Android 7.0+;

4) безпека: зберігання даних лише локально, без доступу до мережі.

Інтерфейс користувача складається з кількох основних екранів:

1) головна форма для вибору типу ресурсу (світло, газ, вода);

2) форма введення показників з інтерактивними полями для дати та значення;

3) графіки витрат, де кожен ресурс відображається у вигляді окремої діаграми;

4) редагування тарифів, що дозволяє встановлювати ціни за одиницю.

Інформаційні потоки в системі побудовані за принципом: користувач → введення даних → збереження в БД → обробка → виведення результату.

Розробимо також відповідні UML-моделі. Діаграма варіантів використання зображена на рисунку 2.1

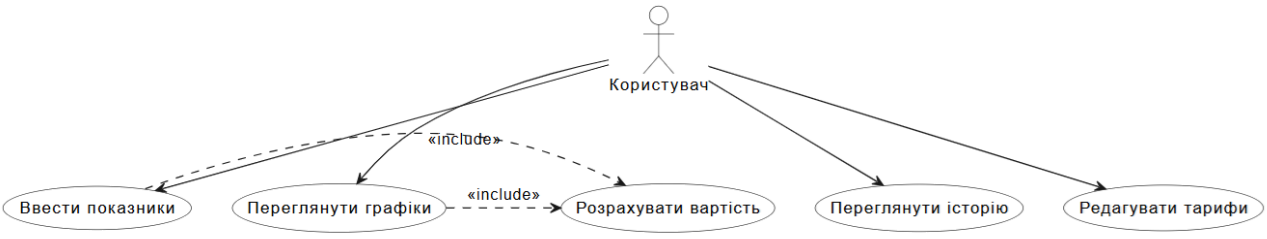

Рисунок 2.1 – Діаграма варіантів використання

Ця діаграма показує взаємодію користувача з основними функціями системи: введення показників, перегляд графіків, історії, редагування тарифів. Вона також демонструє, що розрахунок вартості є невід'ємною частиною інших дій.

Діаграма класів (рис. 2.2) відображає структуру основних класів мобільного додатку, їхні атрибути та взаємозв'язки. Вона ілюструє, як об'єкти «Показник», «Тариф», «Користувач» і «База даних» взаємодіють між собою для реалізації логіки програми.

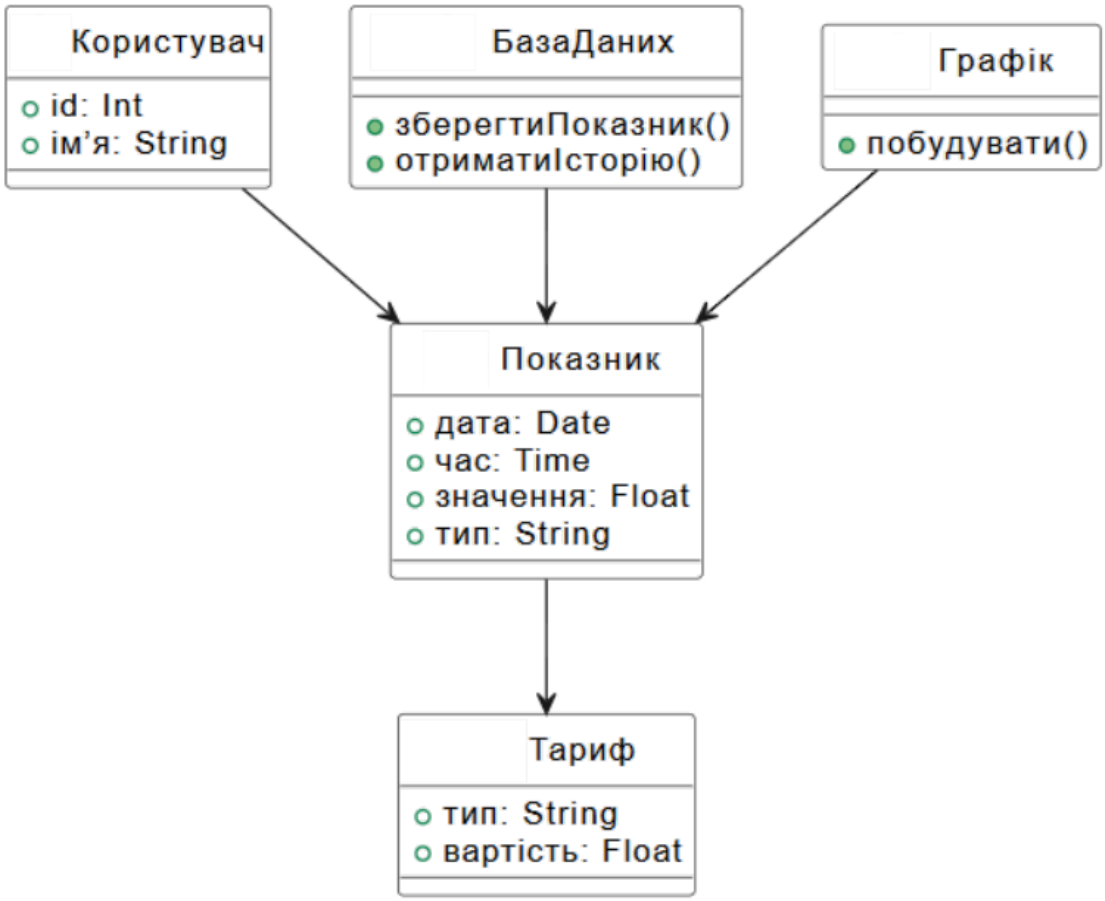

Рисунок 2.2 – Діаграма класів

Діаграма активностей (рис. 2.3) описує послідовність дій при введенні нового показника користувачем – від вибору типу ресурсу до збереження даних у базі. Вона відображає логіку обробки введення даних користувачем.

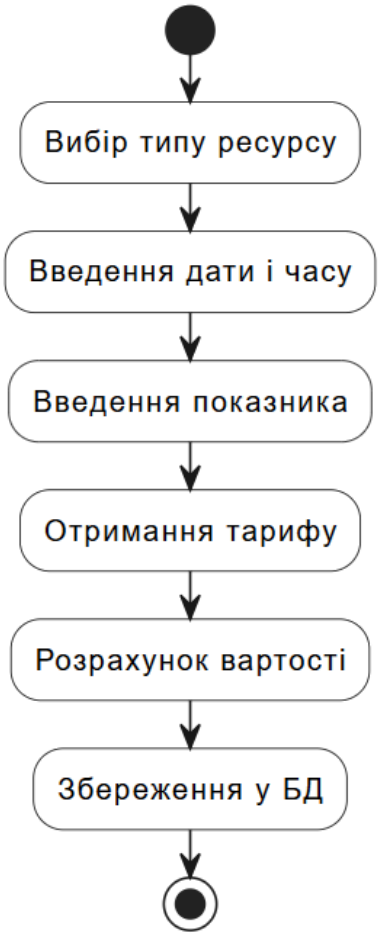

Рисунок 2.3 – Діаграма активностей

Діаграма послідовності зображена на рисунку 2.4.

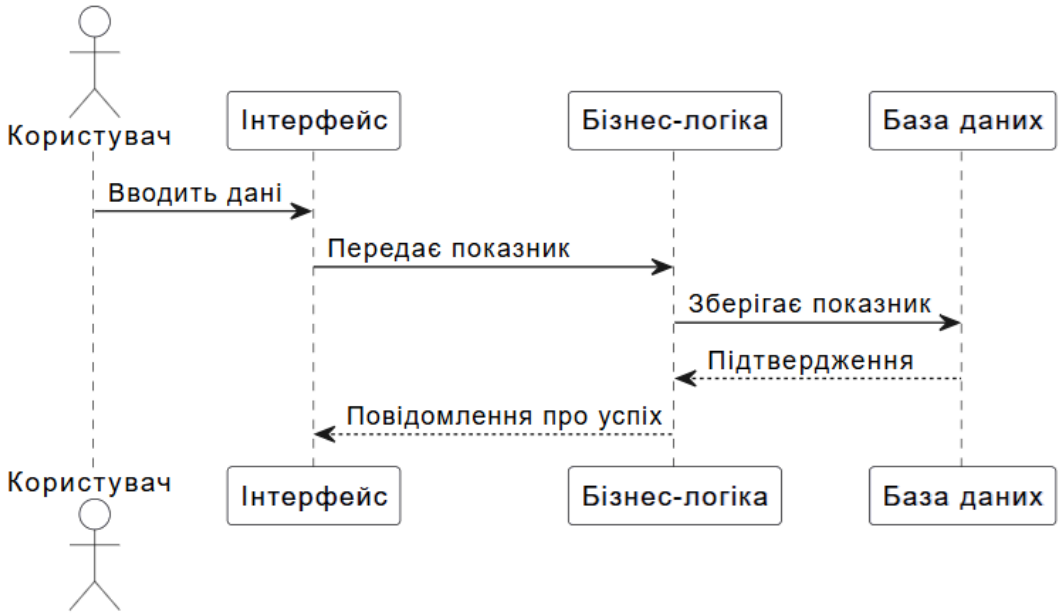

Рисунок 2.4 – Діаграма послідовності

Ця діаграма демонструє взаємодію між актором (користувачем), інтерфейсом, бізнес-логікою та базою даних під час збереження показника. Вона описує порядок викликів і обміну повідомленнями між компонентами системи.

Обрана структура додатку дозволяє забезпечити масштабованість, зручність використання та адаптацію до потреб користувачів. Завдяки локальному збереженню даних забезпечується незалежність від підключення до інтернету та захист персональної інформації. Інтерфейс створений з урахуванням принципів адаптивного дизайну, що робить систему зручною як для молоді, так і для старшої аудиторії.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Для розробки мобільного додатку збору та аналізу витрат на енергоресурси було використано офіційний стек інструментів Android, адаптований під мову Kotlin. Структура проєкту відповідає типовій для Android Studio зі встановленим Android SDK.

Дослідження показують, що Kotlin забезпечує більш високу ефективність розробки мобільних застосунків порівняно з Java, особливо в аспектах скорочення шаблонного коду та зменшення помилок [6].

Проєкт реалізований із застосуванням SDK Android [7] і Gradle [8] як системи збирання. У структурі проєкту чітко виокремлено:

1) /src/main/java/ – основна бізнес-логіка додатку;

2) /src/main/res/ – ресурси інтерфейсу: макети, іконки, стилі;

3) AndroidManifest.xml – конфігурація додатку, реєстрація активностей;

4) build.gradle, gradle.properties, settings.gradle – визначення залежностей, параметрів компіляції, схеми підпису.

Ця структура дозволяє забезпечити розділення логіки, інтерфейсу, ресурсів і конфігурацій, що сприяє підтримуваності та масштабованості проєкту.

Основна мова програмування – Kotlin, рекомендована Google для Android-розробки [9]. Вона дозволяє зменшити обсяг коду завдяки лаконічному синтаксису, а також забезпечує надійність через null-safety.

До проєкту підключено:

1) Room (androidx.room) – ORM-бібліотека для роботи з локальною базою даних SQLite [10];

2) Material Components – для реалізації сучасного інтерфейсу [11];

3) MPAndroidChart – для побудови бар-діаграм споживання [12];

4) ViewModel + LiveData – для забезпечення реактивної архітектури і життєвого циклу [13].

Для конфігурації системи збірки було використано Gradle, який дозволяє автоматизувати процеси компіляції та керування залежностями [14].

Як показують дослідження, вибір оптимальної білд-системи позитивно впливає на надійність і стабільність мобільного додатку [15].

Усі залежності визначено у build.gradle файлах, що дає змогу централізовано керувати версіями компонентів.

Основний алгоритм полягає у визначенні вартості ресурсу: «Вартість» = («Поточний показник» – «Попередній показник») × «Тариф». Алгоритм реалізовано у вигляді функції, що перевіряє валідність введених даних (відсутність від’ємних значень, повторів дати) та здійснює розрахунок, після чого результат зберігається у локальну базу Room.

Система поділена на наступні функціональні компоненти:

1) UI-компонент: реалізує інтерфейс (в XML або Jetpack Compose);

2) Data-компонент: Room-DAO, Entity, Database;

3) Logic-компонент: ViewModel, Repository;

4) Chart-компонент: генерація діаграм;

5) Settings-компонент: управління тарифами користувача.

Реалізовано підтримку життєвого циклу активності через ViewModel. Це дає змогу зберігати стан навіть при зміні орієнтації або закритті програми.

Діаграма компонентів (рис. 2.5) описує розподіл обов'язків між основними модулями.

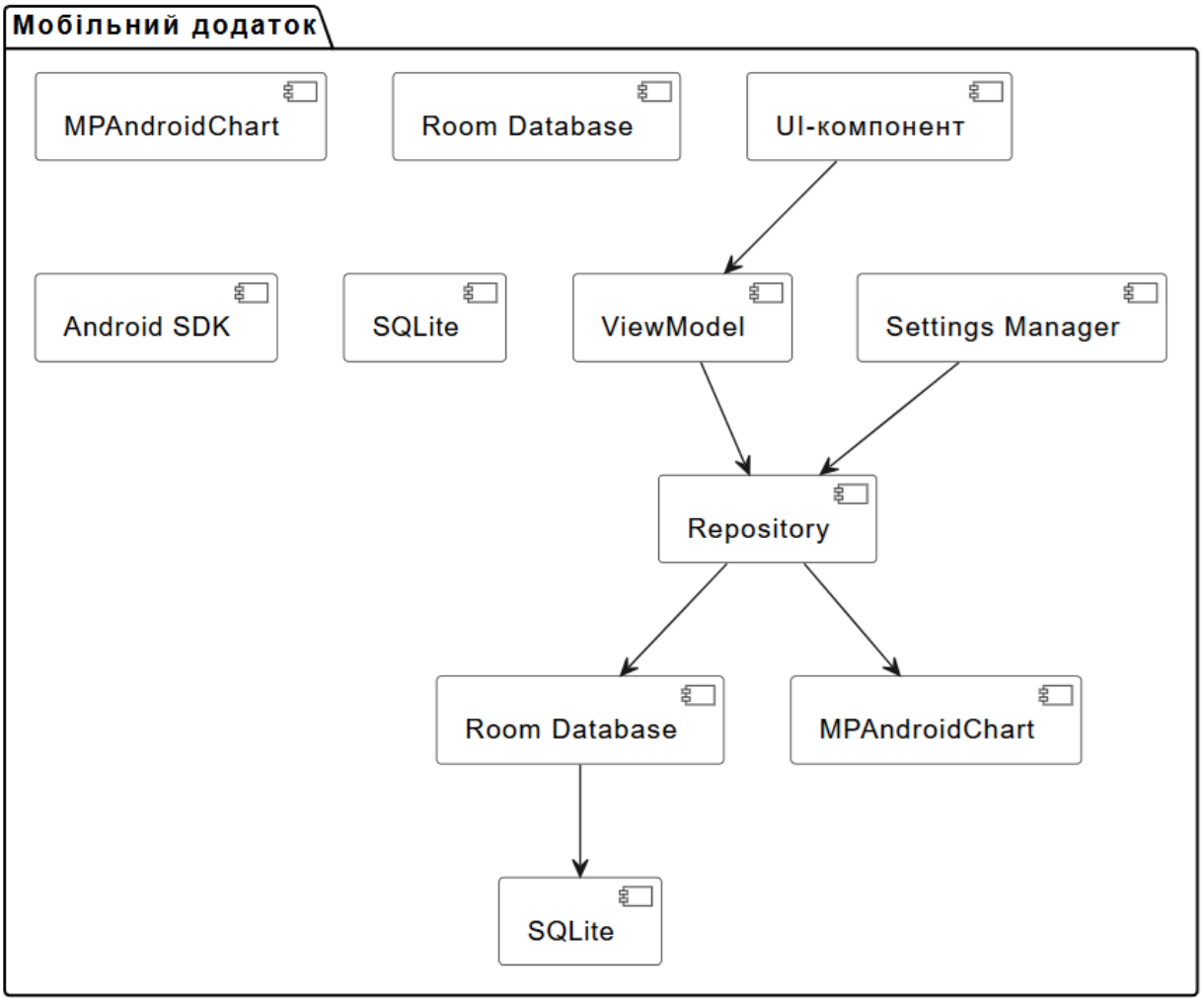

Рисунок 2.5 – Діаграма компонентів

Блок-схема розрахунку витрат (рис. 2.6) демонструє послідовність дій користувача від вибору типу ресурсу та введення показників до обчислення вартості й збереження результату. У разі помилки введення (поточне значення менше попереднього) система виводить повідомлення про помилку і повертає користувача до введення даних.

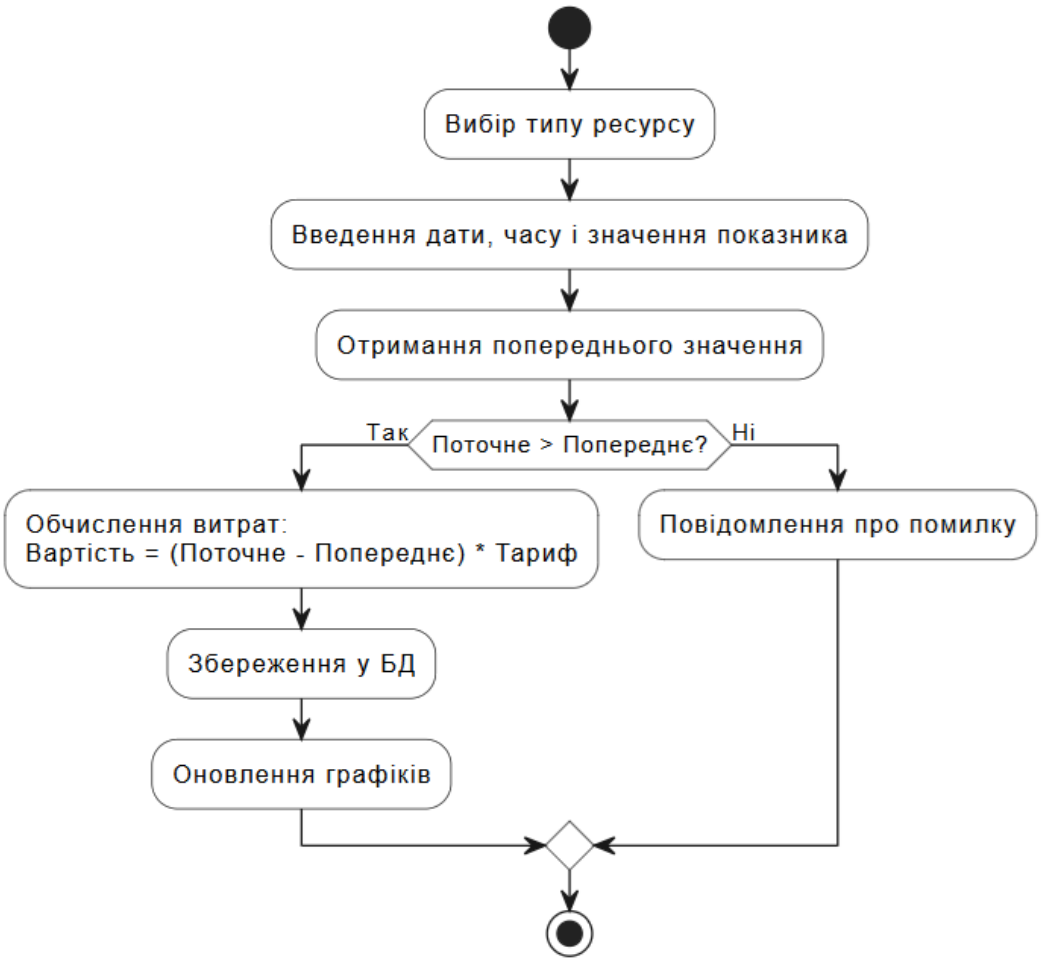

Рисунок 2.6 – Блок-схема розрахунку витрат

Діаграма розгортання (рис. 2.7) – взаємодія додатку з Android OS, SQLite, користувачем.

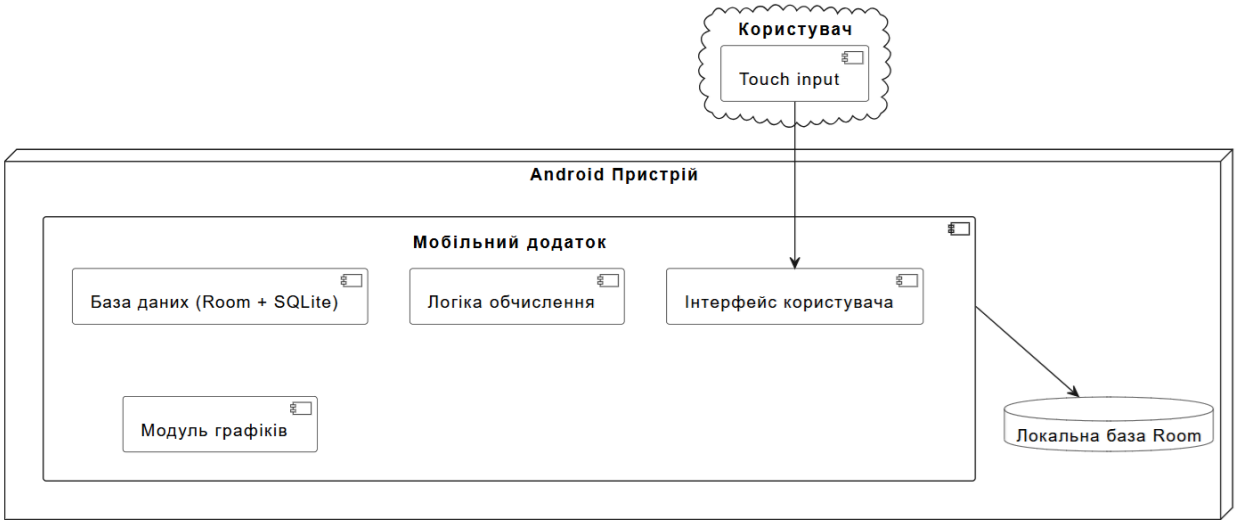

Рисунок 2.7 – Діаграма розгортання

Отже, було обґрунтовано вибір інструментів і архітектури, що дозволили ефективно реалізувати мобільний додаток з чітким поділом компонентів, підтримкою обчислень і збереженням даних у локальній базі.

Висновки до розділу 2

У другому розділі було детально проаналізовано вимоги до функціональності мобільного додатку, визначено його цілі, задачі, а також обґрунтовано вибір об’єктно-орієнтованого підходу до моделювання. Було розроблено UML-діаграми, зокрема діаграми варіантів використання, класів, активностей, послідовності та компонентів, що дозволило формалізувати логіку роботи системи та структуру її основних частин. Це надало змогу наочно представити сценарії взаємодії користувача з додатком, механізми обробки введених даних і способи виводу інформації у вигляді графіків і таблиць.

Особливу увагу приділено виявленню функціональних і нефункціональних вимог, таких як продуктивність, зручність інтерфейсу, стабільність, безпека та адаптивність до різних пристроїв. Здійснений UX/UI-аналіз забезпечив розробку інтерфейсу, який є інтуїтивно зрозумілим, візуально впорядкованим і зручним для використання широкою аудиторією.

Також було проведено обґрунтований вибір інструментальних засобів розробки. Основною мовою програмування обрано Kotlin, яка рекомендована Google для Android-розробки та забезпечує лаконічний синтаксис, високий рівень безпеки та інтеграцію з сучасними бібліотеками. Як середовище розробки використано Android Studio, що дозволяє ефективно керувати проєктом, налагоджувати та тестувати код. Для зберігання даних використано Room – сучасну ORM-бібліотеку для SQLite, що забезпечує надійність і зручність доступу до локальної бази. Для побудови графіків застосовано бібліотеку MPAndroidChart, яка дозволяє ефективно візуалізувати статистику витрат.

Архітектура додатку реалізована у вигляді модульної структури з чітким поділом на компоненти: інтерфейс користувача, логіка обробки даних (ViewModel), зберігання (Room) та аналітика (графічна візуалізація). Такий підхід забезпечує гнучкість, масштабованість та можливість подальшого розширення функціональності додатку без істотної модифікації основної архітектури.

Загалом, результати, отримані в другому розділі, заклали надійну методологічну основу для практичної реалізації мобільного додатку та забезпечили його відповідність сучасним вимогам до ефективного програмного забезпечення.

РОЗДІЛ 3
РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ЗБОРУ ТА АНАЛІЗУ ВИТРАТ НА ЕНЕРГОРЕСУРСИ

3.1 Практична реалізація об’єкта проєктування

Розробка мобільного додатку EcoTrack здійснювалася на мові програмування Kotlin у середовищі Android Studio з використанням Android SDK. Kotlin було обрано як офіційно рекомендовану Google мову для створення Android-застосунків, яка забезпечує високу безпечність завдяки механізмам null safety, лаконічність синтаксису та сучасну підтримку Android API.

Структура застосунку реалізована за принципом чіткого поділу на функціональні модулі, кожен з яких представлений окремою Activity:

1) AddActivity – введення показників і розрахунок вартості ресурсів;

2) AnalysisActivity – графічна візуалізація витрат;

3) SettingsActivity – редагування тарифів;

4) InfoActivity – перегляд порад щодо економії енергії.

Форма додавання витрат реалізована у AddActivity з використанням ViewBinding для безпечного доступу до елементів інтерфейсу. Обчислення проводиться на основі введеного значення та актуального тарифу, який зберігається в SharedPreferences. Нижче наведено приклад фрагмента коду, який виконує обчислення (ліст. 3.1).

Лістинг 3.1 – Обчислення витрат

```
private fun setResult() {
    val amountString = binding.textInputEditTextAmount.text.toString()
    amount = if (amountString.isEmpty()) 0f else amountString.toFloat()

    result = amount * price
    binding.tvResult.text = result.toString() + " " +
getString(R.string.currency_uah)
}
```

кінець лістингу 3.1

На наступному рисунку представлено інтерфейс екрана додавання витрати (AddActivity), який дозволяє користувачеві ввести дані про спожитий ресурс, вибрати дату та зберегти інформацію в локальну базу даних (рис. 3.1).

Рисунок 3.1 – Екран додавання витрат (AddActivity)

Після натискання кнопки «Зберегти» введені дані зберігаються у локальну базу даних SQLite за допомогою бібліотеки Room, що реалізує ORM-модель. Дані зберігаються у таблиці bill, структура якої визначається відповідним Entity-класом (ліст. 3.2).

Лістинг 3.2 – Структура таблиці bill

```
@Entity(tableName = "bill")
```

```
data class Bill(
    @PrimaryKey(autoGenerate = true)
    val id: Int = 0,
    val utility: Utility,
    val date: String,
    val result: Float
)
```

кінець лістингу 3.2

Для доступу до бази даних створено клас AppDatabase та DAO-інтерфейс BillDao, у якому описано метод додавання та агрегації даних (ліст. 3.3).

Лістинг 3.3 – Метод для додавання запису та підрахунку витрат за місяць

```
@Insert(onConflict = OnConflictStrategy.REPLACE)
suspend fun add(bill: Bill)
@Query("""
    SELECT SUM(result)
    FROM bill
    WHERE SUBSTR(date, 7, 4) = :year AND SUBSTR(date, 4, 2) = :month AND
utility = :utility
""")
suspend fun getSumOfResult(year: String, month: String, utility: Utility):
Float?
```

кінець лістингу 3.3

Форма аналізу (AnalysisActivity) реалізована з використанням бібліотеки MPAndroidChart, яка дозволяє створювати стовпчикові діаграми. Користувач може обрати рік із випадаючого списку, після чого дані автоматично агрегуються й виводяться по кожному ресурсу – світло, вода, газ. Нижче наведено приклад формування BarEntry (ліст. 3.4).

Лістинг 3.4 – Формування даних для графіка

```
val resultValue = ArrayList<BarEntry>()
for (i in 0 until listOfData.size) {
    resultValue.add(BarEntry(i.toFloat(), listOfData[i]))
}
```

кінець лістингу 3.4

На наступному рисунку можна побачити інтерфейс екрана аналізу витрат (рис. 3.2).

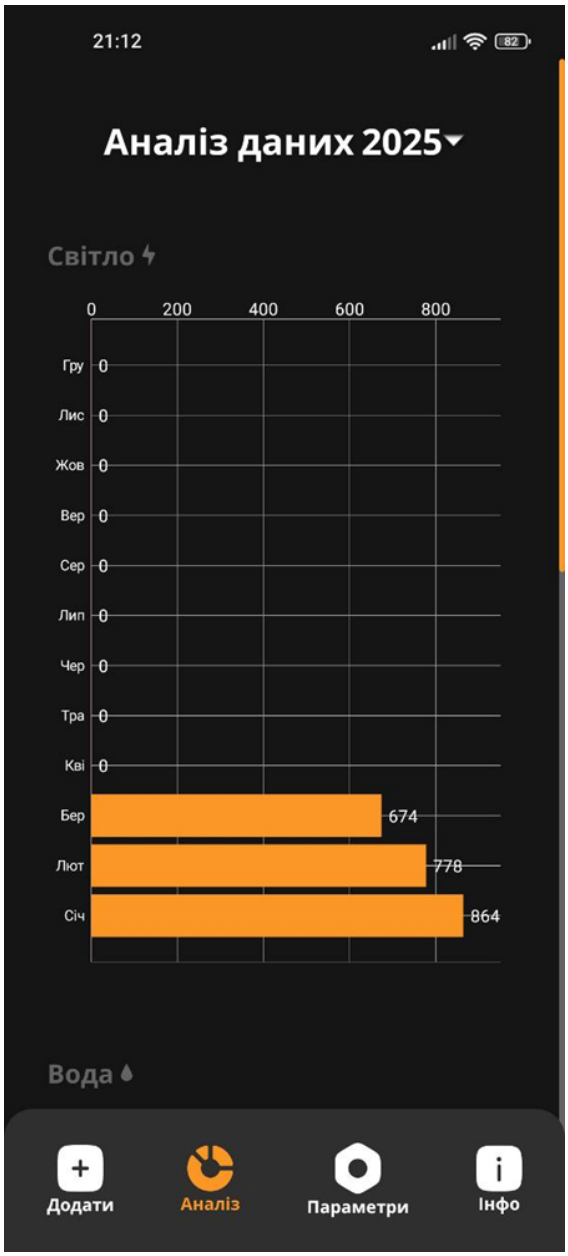

Рисунок 3.2 – Екран аналізу витрат (AnalysisActivity)

Кожен тип ресурсу має окрему діаграму, стилізовану згідно з обраною темною темою інтерфейсу, що покращує візуальне сприйняття та зручність користування додатком.

Редагування тарифів реалізовано через SettingsActivity, у якій користувач може оновити ціни за одиницю споживання (грн/кВт·год, грн/м³). Ці значення

зберігаються у SharedPreferences, що забезпечує їх доступність між сеансами (ліст. 3.5).

Лістинг 3.5 – Збереження тарифів у налаштуваннях

```
SharedPreferencesPrice.electricityPrice = electricity
SharedPreferencesPrice.gasPrice = gas
SharedPreferencesPrice.waterPrice = water
```

кінець лістингу 3.5

Екран редагування тарифів (SettingsActivity) забезпечує можливість змінювати значення тарифів на електроенергію, газ та воду. На наступному рисунку подано вигляд даного екрану (рис. 3.3)

Рисунок 3.3 – Екран редагування тарифів (SettingsActivity)

Особливу увагу приділено валідації даних. Наприклад, у AddActivity не дозволяється зберігати пусті або нульові значення показників (ліст. 3.6).

Лістинг 3.6 – Валідація даних перед збереженням

```
if (amount != 0f && result != 0f && date.isNotEmpty() &&
time.isNotEmpty()) {
    val bill = Bill(0, currentUtility, date, result)
    lifecycleScope.launch {
        matchRepository.add(bill)
        showToast("Bill saved successfully!")
    }
} else {
    showToast("Please enter valid values")
}
```

кінець лістингу 3.6

Загалом, реалізація мобільного додатку демонструє володіння засобами мови Kotlin, застосування сучасних бібліотек Android (Room, MPAndroidChart, ViewModel, LiveData) та дотримання архітектурних принципів при створенні функціонального, масштабованого і зручного застосунку.

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

У процесі розробки мобільного додатку EcoTrack було проведено поетапне тестування та налагодження всіх функціональних компонентів системи. Основна мета тестування полягала у виявленні помилок, перевірці коректності логіки обчислень, стабільності інтерфейсу та відповідності системи вимогам користувача.

У процесі тестування мобільного додатку застосовувалися різні підходи, зокрема ручне функціональне тестування, під час якого перевірялися основні сценарії використання, включаючи введення показників, зміну тарифів, перегляд графіків і збереження даних. Також проводилось юніт-тестування, метою якого була перевірка базових обчислювальних функцій, зокрема правильності розрахунку витрат у гривнях на основі введених значень і тарифів.

Із метою забезпечення коректної взаємодії між складовими системи здійснювалося інтеграційне тестування, що охоплювало взаємодію між модулями ViewModel, базою Room, SharedPreferences та графічним інтерфейсом. Окрім цього, проводилось UX-тестування, у рамках якого оцінювалася зручність навігації додатком, зокрема робота з полями введення, кнопками, графіками та загальне враження користувача від інтерфейсу. Приклад юніт-тесту наведено в лістингу 3.7.

Лістинг 3.7 – Юніт-тест

```
@Test
fun addition_isCorrect() {
    assertEquals(4, 2 + 2)
}
```

кінець лістингу 3.7

Цей базовий тест використовується як перевірка конфігурації середовища. У подальшому додатково реалізовано перевірку логіки розрахунків та доступу до даних.

У процесі тестування мобільного додатку було виявлено кілька недоліків, які впливали на стабільність його роботи. Однією з проблем стала помилка розрахунку у випадку, коли поле для введення показника залишалося порожнім, то це призводило до аварійного завершення програми. Для усунення цієї помилки було реалізовано перевірку amountString.isEmpty() перед конвертацією значення у формат Float. Інша проблема стосувалася некоректного збереження дати та часу: через несинхронізоване оновлення інтерфейсу іноді фіксувалася не вибрана користувачем дата, а поточна системна. Вирішенням стало використання прямої прив’язки до елементів TextView, значення яких оновлюються одразу після вибору в діалогових вікнах DatePickerDialog і TimePickerDialog.

Також було виявлено випадки збереження нульових або від’ємних значень, що є некоректним з точки зору обліку витрат. Для цього було додано валідацію введених даних перед їх записом у базу даних. Останньою

проблемою стала некоректна робота темного інтерфейсу на деяких пристроях із версіями Android нижче 8.0. Для її усунення були адаптовані стилі інтерфейсу з урахуванням перевірки версії SDK пристрою, що дозволило забезпечити стабільну роботу застосунку на ширшому спектрі пристроїв.

Для стабільної роботи додатку визначено відповідні мінімальні системні вимоги (табл. 3.1).

Таблиця 3.1 – Мінімальні системні вимоги

Параметр	Мінімальне значення	Рекомендоване значення
ОС Android	Android 7.0 (API 24)	Android 10.0 (API 29) і вище
Оперативна пам’ять	2 ГБ	4 ГБ і більше
Вільне місце на пристрої	100 МБ	200 МБ
Дозвіл екрана	720×1280 px	1080×1920 px і більше
Права доступу	Немає спеціальних дозволів	-

Проведене тестування дозволило виявити та усунути ряд незначних помилок, що покращило стабільність роботи мобільного додатку. У результаті реалізовано надійний інструмент з високим рівнем відповідності функціональним вимогам. Після налагодження система повністю готова до використання користувачами на більшості сучасних Android-пристроїв.

3.3 Розробка бази даних

У межах реалізації мобільного додатку EcoTrack було створено локальну базу даних для зберігання інформації про витрати енергоресурсів. Для цього використано інструментарій Room – об’єктно-реляційну бібліотеку Android Jetpack, яка спрощує доступ до бази даних SQLite та забезпечує безпечну роботу з даними у вигляді об’єктів Kotlin.

У системі використовується одна таблиця – bill, яка містить інформацію про споживання конкретного ресурсу у визначений день. Її структура задається через анотації @Entity у класі Bill (ліст. 3.8).

Лістинг 3.8 – Структура таблиці

```
@Entity(tableName = "bill")
data class Bill(
    @PrimaryKey(autoGenerate = true)
    val id: Int = 0,
    val utility: Utility,
    val date: String,
    val result: Float
)
```

кінець лістингу 3.8

Поле utility – це тип енергоресурсу (електроенергія, газ або вода), що реалізовано як enum-клас Utility. Поле date зберігає дату в форматі dd/MM/yyyy, а result – обчислену вартість споживаного ресурсу у гривнях. Схема бази даних зображена на рисунку 3.4.

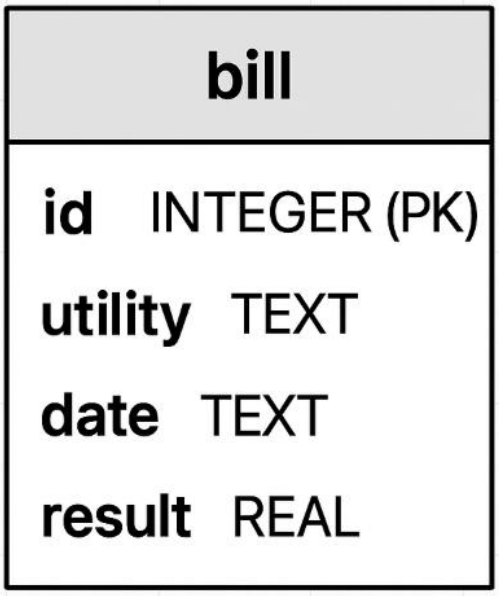

Рисунок 3.4 – Схема бази даних

Доступ до таблиці реалізовано через інтерфейс DAO – BillDao, який містить методи для читання та запису (ліст. 3.9).

Лістинг 3.9 – Інтерфейс BillDao для доступу до таблиці bill

```
@Dao
interface BillDao {
    @Query("SELECT * FROM bill")
    fun getAllBill(): LiveData<List<Bill>>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun add(bill: Bill)

    @Query("""
        SELECT SUM(result)
        FROM bill
        WHERE SUBSTR(date, 7, 4) = :year
          AND SUBSTR(date, 4, 2) = :month
          AND utility = :utility
    """)
    suspend fun getSumOfResult(year: String, month: String, utility:
Utility): Float?
}
```

кінець лістингу 3.9

Тут використано SQL-запити стандарту SQLite, зокрема:

1) SELECT * FROM bill – для отримання всіх записів;

2) INSERT – для додавання нового запису;

3) SELECT SUM(result) ... WHERE ... – для агрегації витрат за обраний місяць і тип ресурсу.

У проєкті реалізовано повну підтримку додавання, збереження, читання та агрегованого аналізу даних. Уся взаємодія з DAO відбувається через репозиторій BillRepository, який виступає проміжною ланкою між базою та ViewModel.

База даних створюється при першому запуску застосунку за допомогою Room.databaseBuilder (ліст. 3.10).

Лістинг 3.10 – Створення бази даних при першому запуску додатка

```
AppDatabase.getDatabase(context)
```

кінець лістингу 3.10

Вона зберігається у внутрішньому сховищі додатку та не потребує доступу до мережі. Такий підхід забезпечує автономність, безпеку та швидкий доступ до даних.

Отже, база даних у додатку реалізована засобами Android Room, що дає змогу ефективно оперувати інформацією про енергоспоживання. Завдяки об’єктно-реляційній моделі даних, SQL-запитам та репозиторному підходу було досягнуто високого рівня надійності й масштабованості збереження даних у межах мобільного середовища.

3.4 Захист інформаційно-комп’ютерної системи

У процесі розробки мобільного застосунку EcoTrack особливу увагу було приділено забезпеченню базового рівня безпеки користувацьких даних, що зберігаються локально. Оскільки додаток функціонує в автономному режимі без використання мережі або зовнішніх API, основні аспекти захисту стосуються зберігання та обробки даних у межах пристрою користувача.

Для збереження інформації про витрати енергоресурсів використовується база даних SQLite, реалізована за допомогою бібліотеки Room. Усі дані зберігаються в локальній базі у внутрішньому сховищі Android-пристрою за шляхом /data/data/com.student.ecotrack/databases/, доступ до якого мають лише застосунки з таким самим ідентифікатором пакета. Завдяки цьому забезпечується ізоляція даних на рівні операційної системи.

Ціни на електроенергію, воду та газ зберігаються за допомогою стандартного механізму SharedPreferences. Доступ до цих налаштувань також обмежено внутрішнім середовищем Android. Оскільки ці значення не є чутливими і не передаються мережею, шифрування не застосовувалося, однак структура додатку дозволяє легко інтегрувати механізми на кшталт EncryptedSharedPreferences у разі потреби.

У проєкті застосовано механізм обфускації та мінімізації коду з використанням ProGuard, який налаштований у конфігураційному файлі Gradle (ліст. 3.11).

Лістинг 3.11 – Підключення ProGuard у конфігурації Gradle

```
proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
'proguard-rules.pro'
```

кінець лістингу 3.11

Цей підхід дозволяє зменшити й ускладнити зворотну інженерію APK-файлу, що є важливим аспектом захисту логіки роботи програми, особливо в разі подальшої публікації у відкритому доступі.

Застосунок не обробляє мережевих запитів, не підключається до зовнішніх серверів і не використовує механізми онлайн-автентифікації. Завдяки цьому повністю виключаються ризики, пов'язані з перехопленням даних, атакою типу «людина посередині» (MITM) або DDoS-атаками. Така архітектура забезпечує підвищену приватність та безпеку використання.

У застосунку EcoTrack реалізовано базові, але ефективні механізми захисту, що відповідають його архітектурі та призначенню. Зберігання даних обмежене внутрішнім середовищем Android, конфіденційність забезпечується ізоляцією файлової системи, а захист коду – завдяки ProGuard. Такий підхід дозволяє користувачам безпечно працювати із застосунком без ризику втрати або витоку даних, що особливо важливо при обліку витрат на енергоресурси в особистому користуванні.

3.5 Розробка документації для програмного продукту

Документація до програмного продукту EcoTrack охоплює основні аспекти його використання, встановлення та технічні вимоги. Оскільки застосунок є мобільним додатком, який працює автономно на платформі Android, процес його розгортання не передбачає підключення до серверів або

налаштування зовнішніх служб. Основна увага приділяється простоті встановлення, доступності інтерфейсу та зручності використання.

При формуванні нефункціональних вимог особлива увага приділялась зручності використання, швидкодії та енергоефективності застосунку [16].

Для стабільної роботи мобільного застосунку EcoTrack на Android-пристроях сформульовано такі мінімальні та рекомендовані параметри (табл. 3.2).

Таблиця 3.2 – Мінімальні системні вимоги

Параметр	Мінімальні вимоги	Рекомендовані параметри
Операційна система Android	Android 7.0 (API 24)	Android 10.0 (API 29) або новіше
Оперативна пам’ять	2 ГБ	4 ГБ або більше
Вільне місце на пристрої	100 МБ	200 МБ
Дозвіл екрана	720×1280 px	1080×1920 px або більше
Доступ до інтернету	Не потрібен	-
Необхідні дозволи	Відсутні	-

Інтерфейс застосунку побудовано інтуїтивно – без потреби в додатковому навчанні. Проте для зручності користувача можна виділити такі функціональні можливості:

1) екран додавання витрат (AddActivity) дозволяє вводити кількість спожитого ресурсу, обирати дату та час, автоматично розраховує вартість відповідно до встановленого тарифу, а також забезпечує збереження результату у локальну базу даних;

2) екран налаштувань (SettingsActivity) надає змогу редагувати тарифи на електроенергію, воду та газ. Усі внесені значення зберігаються у пам’яті пристрою, що гарантує збереження налаштувань після завершення роботи застосунку;

3) екран аналізу (AnalysisActivity) забезпечує графічне відображення витрат по місяцях. Користувач може обрати потрібний рік за допомогою випадаючого списку та переглянути три окремі діаграми за типами ресурсів;

4) екран інформації (InfoActivity) містить текстову довідку з порадами щодо економного споживання енергоресурсів та інструкцію з користування додатком, оформлену у зручному для читання форматі.

Оскільки застосунок є Android-додатком, він може бути встановлений одним із двох способів:

1) ручне встановлення через APK-файл передбачає копіювання інсталяційного файлу на мобільний пристрій, увімкнення дозволу на встановлення з «невідомих джерел» у налаштуваннях системи, а також запуск файлу за допомогою менеджера файлів і підтвердження встановлення;

2) запуск через Android Studio (для тестування) виконується шляхом підключення фізичного пристрою або емулятора, відкриття проєкту в Android Studio та запуску програми за допомогою функції «Run» (гаряча клавіша Shift+F10), після чого додаток відкриється на вибраному пристрої.

У застосунку також реалізовано вбудовану довідкову сторінку (InfoActivity), яка містить поради щодо ефективного використання електроенергії, газу та води.

Інтеграція виконана через окремий екран з можливістю переходу з будь-якої іншої частини інтерфейсу. Цей підхід дозволяє уникнути складного структурування довідкової документації, зберігаючи зручність для користувача.

У додатку передбачено окремий екран для перегляду довідкової інформації, реалізований у вигляді активності InfoActivity, що є складовою частиною загальної навігаційної структури інтерфейсу. Цей екран містить стислу, проте змістовну та інформативну текстову інформацію, що охоплює основні правила економії ресурсів, зокрема електроенергії, води та газу. Візуально сторінка оформлена у мінімалістичному стилі, який підтримує загальний дизайн застосунку і не перевантажує користувача зайвими елементами.

Основний текст структуровано за допомогою логічно розділених абзаців, що значно полегшує читання з екрана мобільного пристрою навіть для користувачів без технічної підготовки. У майбутньому цей розділ може бути розширений за рахунок інтеграції з онлайн-документацією, додаванням ілюстрацій або навіть відеоінструкцій. Такий підхід дозволяє не лише надати

користувачам базові рекомендації, а й поступово впроваджувати нові можливості без необхідності оновлення всієї програми (рис. 3.5).

Рисунок 3.5 – Довідкова сторінка з порадами (InfoActivity)

Документація до застосунку EcoTrack охоплює всі необхідні аспекти його використання. Простий процес встановлення, відсутність потреби у підключенні до мережі та інтуїтивний інтерфейс роблять програму зручною для

широкого кола користувачів. У майбутньому можливе розширення довідкової системи або розробка інсталяційного пакета з автоматичним оновленням через Google Play.

Висновки до розділу 3

У третьому розділі було здійснено практичну реалізацію мобільного додатку EcoTrack, що призначений для обліку та аналізу витрат на енергоресурси. Реалізація виконана з використанням мови програмування Kotlin у середовищі Android Studio з використанням бібліотек Android Jetpack, зокрема Room, LiveData, ViewModel та MPAndroidChart.

Було реалізовано повний набір функціональних можливостей: введення даних про спожиті ресурси, автоматичне обчислення вартості на основі тарифів, збереження результатів у локальну базу даних, побудова графіків витрат за місяцями, редагування тарифів та перегляд аналітичної інформації. Інтерфейс додатку створено відповідно до принципів адаптивного дизайну та забезпечено зручну навігацію між екранами.

Проведене тестування охоплювало функціональні та UX-аспекти, дозволило виявити та усунути низку недоліків, що підвищило стабільність та надійність додатку. Сформульовані мінімальні системні вимоги підтверджують, що додаток може ефективно працювати на більшості сучасних Android-пристроїв, починаючи з Android 7.0.

Особливу увагу приділено безпеці: реалізовано захист локальних даних на рівні файлової системи Android, обфускацію коду за допомогою ProGuard, а також обґрунтовано подальші напрямки вдосконалення, такі як шифрування даних і використання EncryptedSharedPreferences. Крім того, розроблено супровідну документацію, яка описує встановлення, використання додатку та його функціональні можливості, що робить систему зручною та доступною для кінцевого користувача.

Загалом, результати, досягнуті в цьому розділі, підтверджують можливість ефективного застосування мобільного додатку EcoTrack для побутового моніторингу витрат на енергоресурси та формують основу для його подальшого розвитку.

ВИСНОВКИ

У межах виконання кваліфікаційної роботи бакалавра було успішно реалізовано мобільний додаток EcoTrack, призначений для ведення обліку та аналізу витрат на основні енергоресурси – електроенергію, воду та газ. Застосунок створено з використанням сучасних технологій Android-розробки, зокрема мови програмування Kotlin, архітектурних компонентів Jetpack (Room, LiveData, ViewModel), а також бібліотек для побудови діаграм та зручного інтерфейсу. Завдання, визначені на початковому етапі дослідження, були виконані в повному обсязі.

Аналіз предметної області дозволив виявити актуальну проблему: більшість існуючих додатків або не підтримують локалізацію, або не враховують специфіку українських тарифів. Це обґрунтувало необхідність створення адаптованого інструменту для українського користувача.

Визначення функціональних і нефункціональних вимог дало змогу чітко окреслити межі системи, її основні функції, сценарії використання, вимоги до продуктивності, надійності, безпеки й адаптивності. Також створено відповідні UML-діаграми, які формалізують архітектуру додатку.

Обґрунтування вибору інструментів і технологій дозволило підібрати ефективний стек: Kotlin, Android Studio, Room, Jetpack-компоненти, MPAndroidChart. Це забезпечило сучасність і підтримуваність розробки.

Реалізація мобільного додатку EcoTrack охопила створення всіх ключових екранів: додавання витрат, перегляд графіків, редагування тарифів, інформування користувача. Функціональність повністю відповідає вимогам, сформульованим на початку роботи.

Тестування та налагодження системи дало змогу виявити і виправити критичні помилки, пов’язані з валідацією, збереженням дати та сумісністю тем оформлення. Застосунок пройшов ручне функціональне, UX- та інтеграційне тестування.

Розробка структури локальної бази даних реалізована з використанням бібліотеки Room. Створено сутність Bill, DAO-інтерфейс і репозиторій, що забезпечують додавання, читання й агрегацію витрат. Дані зберігаються безпечно у внутрішньому сховищі.

Захист мобільного додатку реалізовано шляхом використання внутрішнього сховища Android, обмеження доступу до файлів та застосування ProGuard для обфускації коду. Архітектура без використання інтернету підвищує безпеку.

Підготовка супровідної документації охопила опис інтерфейсу, інструкції з встановлення, вимоги до системи, а також вбудовану довідку в самому додатку. Це робить застосунок зрозумілим і доступним для користувача.

Результати роботи відповідають актуальним світовим тенденціям у сфері цифровізації побутових процесів, підвищення енергоефективності та відповідального ресурсоспоживання. У світі активно впроваджуються мобільні рішення для моніторингу витрат, однак більшість із них орієнтовані на глобальний ринок, тоді як EcoTrack адаптований під українські реалії: він дозволяє редагувати тарифи, використовує українську мову інтерфейсу та не вимагає доступу до Інтернету. Загалом, розроблений застосунок є актуальним та конкурентоспроможним у локальному інформаційному середовищі.

У процесі реалізації досягнуто високих якісних показників: стабільність роботи, простота використання, адаптивний інтерфейс, візуалізація даних у графічному вигляді. Кількісні показники включають реалізацію повного циклу функціональності (збереження, обчислення, перегляд, аналітика) без використання зовнішніх баз чи серверів, що забезпечує автономність роботи додатку. Система є масштабованою й готовою до майбутнього розширення (наприклад, синхронізація з хмарними сервісами або інтеграція з «розумними» лічильниками).

Розробка має потенціал для практичного використання в побуті, а також може бути застосована в малих підприємствах або ОСББ для обліку споживання ресурсів. Соціальна значущість проєкту полягає у формуванні

енергетично свідомої поведінки користувачів та сприянні відповідальному споживанню ресурсів.

Тематика роботи узгоджується з напрямами науково-дослідних розробок, пов’язаних із впровадженням інформаційних систем в енергоменеджмент, та є перспективною для інтеграції в освітні дисципліни, що викладаються в університеті на кафедрі інформаційних технологій. Програмне забезпечення може бути використане як демонстраційний приклад у курсах мобільного програмування або енергетичного моделювання.

У перспективі можливе розширення функціональності: додавання оновлюваної бази тарифів, авторизації користувача, хмарного резервного копіювання даних, розсилки статистики, реалізація інтелектуального прогнозування витрат. Також можливо інтегрувати біометричну автентифікацію або синхронізацію між пристроями.

Отже, поставлені завдання кваліфікаційної роботи виконано повністю. Розроблений мобільний додаток є практичним, ефективним інструментом для обліку витрат на енергоресурси, що має значний потенціал для подальшого розвитку та впровадження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про житлово-комунальні послуги . Закон України від 09.11.2017 № 2189-VIII : станом на 9 лип. 2023 р. URL: https://zakon.rada.gov.ua/laws/show/2189-19#Text (дата звернення: 27.04.2025).

2. Лучик С. Д., Бруневич, Х. С. Автоматизація обліку в сфері комунальних послуг. Облік, аналіз і аудит: виклики інституціональної економіки. 2018. С. 152-155.

3. Teroshyna I. M. Особливості обліку розрахунків за житлово-комунальні послуги з населенням. Ukrainian Journal of Educational Studies and Information Technology. 2016. Т. 3, № 1. С. 55-58. URL: https://doi.org/10.32919/10.32919/uesit.2016.01. (дата звернення: 27.04.2025).

4. Казимир М. М., Басюк Т. М. Проєкт інформаційної системи із автоматизації розрахунків з надання комунальних послуг на платформі salesforce. In The 11th International scientific and practical conference «Modern generation: current problems, experience, development prospects»(November 12-15, 2024) Seville, Spain. International Science Group. 2024. 415 p.

5. ПК ІТА-2 – Автоматизація нарахувань за комунальні послуги, обліку плати, ведення стану взаєморозрахунків з абонентами. URL: https://ssb.com.ua/support/index.php?option=com_content&task=view&id=169&Itemid=45 (date of access: 27.04.2025).

6. Effectiveness of Kotlin vs. Java in android app development tasks / L. Ardito et al. Information and Software Technology. 2020. Vol. 127. P. 106374. URL: https://doi.org/10.1016/j.infsof.2020.106374 (date of access: 27.04.2025).

7. SDK Platform Tools release notes. Android Studio. Android Developers. Android Developers. URL: https://developer.android.com/tools/releases/platform-tools (date of access: 28.04.2025).

8. Gradle Build Tool. Gradle. URL: https://gradle.org/ (date of access: 28.04.2025).

9. Kotlin Programming Language. Kotlin. URL: https://kotlinlang.org/ (date of access: 28.04.2025).

10. Mawlood-Yunis A.-R. Android SQLite, Firebase, and Room Databases. Android for Java Programmers. Cham, 2022. P. 475-530. URL: https://doi.org/10.1007/978-3-030-87459-9_11 (date of access: 30.04.2025).

11. Forrester A., Boudjnah E., Dumbravan A., Tigcal J. How to Build Android Apps with Kotlin: A Practical Guide to Developing, Testing, and Publishing Your First Android Apps. Packt Publishing, Limited, 2023. 704 p.

12. Joshi I. A. Unblind the charts: Towards making interactive charts accessible in android applications. arXiv preprint arXiv:2109.12442, 2021. URL: https://doi.org/10.48550/arXiv.2109.12442. (date of access: 01.05.2025).

13. Vijaywargi A., Boddapati U. K. Architectural Patterns in Android Development: Comparing MVP, MVVM, and MVI. International Journal for Research in Applied Science and Engineering Technology. 2024. Vol. 12, no. 4. P. 4611-4616. URL: https://doi.org/10.22214/ijraset.2024.60762 (date of access: 01.05.2025).

14. Gradle Documentation. Gradle User Manual. URL: https://docs.gradle.org/current/samples/sample_building_kotlin_applications.html (date of access: 01.05.2025).

15. Understanding the quality and evolution of Android app build systems / P. Liu et al. Journal of Software: Evolution and Process. 2023. URL: https://doi.org/10.1002/smr.2602 (date of access: 02.05.2025).

16. How do Android developers improve non-functional properties of software? / J. Callan et al. Empirical Software Engineering. 2022. Vol. 27, no. 5. URL: https://doi.org/10.1007/s10664-022-10137-2 (date of access: 02.05.2025).

ДОДАТКИ

Додаток А

Фрагмент коду AddActivity.kt – додавання витрат в мобільному застосунку EcoTrack

```
class AddActivity : AppCompatActivity() {

    private lateinit var binding: ActivityAddBinding

    private lateinit var matchDatabase: AppDatabase
    private lateinit var matchRepository: BillRepository
    private var currentUtility: Utility = Utility.ELECTRICITY
    private val calendar = Calendar.getInstance()

    private var amount: Float = 0f
    private var price: Float = 0f
    private var result: Float = 0f

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()

        binding = ActivityAddBinding.inflate(layoutInflater)
        setContentView(binding.root)

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main))
{ v, insets ->
            val                        systemBars                        =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left,                systemBars.top,
systemBars.right, systemBars.bottom)
            insets
        }

        setupDataBase()
        setupUI()
```

```
        setListener()
    }

    companion object {
        fun launch(activity: AppCompatActivity) {
            val intent = Intent(activity, AddActivity::class.java)
            activity.startActivity(intent)
        }
    }

    private fun setupDataBase() {
        SharedPreferencesPrice.initialize(this@AddActivity)
        matchDatabase = AppDatabase.getDatabase(this@AddActivity)
        matchRepository = BillRepository(matchDatabase.matchDao())
    }
```