Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ КОНВЕРТУВАННЯ КУРСУ ВАЛЮТ З ВИКОРИСТАННЯМ KOTLIN

DEVELOPMENT OF A MOBILE APPLICATION FOR CURRENCY CONVERSION USING KOTLIN

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

Виконав: здобувач вищої освіти
групи ІПЗз-41
Гнатюк Владислав Ігорович

Керівник:
Падалко Анатолій Михайлович

Кваліфікаційну роботу
допущено до захисту
«___» _____ 20__ 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
_____
«___» _____ 202__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Гнатюку Владиславу Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «*Розробка мобільного додатку для конвертування курсу валют з використанням Kotlin*».
Керівник роботи: *к.ф.-м..н., доцент Падалко А. М.*
затверджені наказом закладу вищої освіти від «19» *грудня 2024 р.* № *474/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«10» червня 2025 р.*
3. Вихідні дані до роботи: *Kotlin, Android Studio, Retrofit, LiveData, ViewModel, MVVM, OpenExchangeRates API UML, методичні вкзівки до кваліфікаційної роботи.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення, опис функціонального наповнення об'єкта проектування, практична реалізація об'єкта проектування.
5. Перелік графічного матеріалу: *13 рисунків.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Падалко А. М.*		
Специфікація вимог до розробленої системи	*Падалко А. М.*		
Розробка об’єкта проектування	*Падалко А. М.*		
Нормоконтроль	*Вознюк А.В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	___%		
Академічна доброчесність	*Падалко А. М.*		

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.*	
2	Аналіз проблеми розробки та впровадження об’єкту проектування	*до 01.03.2025 р.*	
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	*до 15.03.2025 р.*	
4	Розробка функціонально-структурної схеми роботи об’єкта проектування	*до 29.03.2025 р.*	
5	Практична реалізація об’єкта проектування	*до 26.04.2025 р.*	
6	Тестування та налагодження об’єкта проектування	*до 03.05.2025 р.*	
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	*до 10.06.2025 р.*	

Здобувач вищої освіти _____ Владислав ГНАТЮК_
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Анатолій ПАДАЛКО
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Гнатюк В. І. Розробка мобільного додатку для конвертування курсу валют 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.

Кваліфікаційна робота присвячена розробці мобільного додатку для конвертації валют на платформі Android із використанням мови програмування Kotlin. У першому розділі представлено аналіз предметної області, огляд існуючих аналогів, обґрунтовано актуальність і доцільність створення нового програмного продукту. У другому розділі визначено вимоги до функціональності додатку, здійснено його проєктування з використанням UML-діаграм, описано архітектурне рішення на основі шаблону MVVM. Третій розділ містить опис реалізації основних компонентів додатку, інтерфейсу користувача, взаємодії з API-сервісом OpenExchangeRates, а також проведено функціональне тестування та розроблено супровідну документацію. Результатом є зручний та ефективний мобільний застосунок, що дозволяє здійснювати точну конвертацію валют у режимі реального часу.

Ключові слова: Kotlin, Android, конвертація валют, мобільний додаток, MVVM, Retrofit, ViewModel, API, UML.

ABSTRACT

Hnatiuk V. Development of a Mobile Application for Currency Conversion Using Kotlin. Manuscript. Qualification work of the bachelor's program "Software engineering" in the specialty "Software engineering". Lutsk National Technical University. Lutsk, 2025.

A bachelor's degree thesis consists of an introduction, three chapters, conclusions, and a list of sources used.

The qualification thesis is dedicated to the development of a mobile application for currency conversion on the Android platform using the Kotlin programming language. The first chapter presents an analysis of the subject area, a review of existing analogues, and substantiates the relevance and feasibility of creating a new software product. The second chapter defines the functional requirements of the application, presents its design using UML diagrams, and describes the architectural solution based on the MVVM pattern. The third chapter provides an overview of the implementation of the application's main components, the user interface, interaction with the OpenExchangeRates API service, as well as functional testing and development of supporting documentation. The result is a convenient and efficient mobile application that allows users to perform accurate currency conversions in real time.

Keywords: Kotlin, Android, currency conversion, mobile application, MVVM, Retrofit, ViewModel, API, UML.

ЗМІСТ

ВСТУП

Актуальність теми. В умовах глобалізації світової економіки та постійної динаміки фінансових ринків зростає потреба у швидкому доступі до актуальної інформації про валютні курси. Щоденні операції – як у бізнес-середовищі, так і в побуті – потребують точного й оперативного перерахунку вартості товарів, послуг або фінансових активів у різних валютах. Це особливо актуально для туристів, фрилансерів, підприємців, а також користувачів, які здійснюють міжнародні покупки чи продажі.

Традиційні методи перевірки валютного курсу, як-от перегляд курсів на сайтах банків або в браузері, потребують додаткових дій, займають час та не завжди забезпечують інтерактивну взаємодію чи гнучке налаштування. У зв’язку з широким розповсюдженням мобільних пристроїв та зростанням очікувань користувачів до зручності мобільних застосунків, актуальним завданням є створення мобільного додатку, що дозволяє виконувати конвертацію валют у режимі реального часу з використанням сучасних сервісів обміну даними.

Крім того, Kotlin як офіційна мова розробки Android-додатків, активно підтримується спільнотою розробників та Google. Вона забезпечує лаконічний, безпечний та сучасний синтаксис, який дозволяє створювати надійні та зручні мобільні рішення з високою продуктивністю. Завдяки використанню Kotlin можна реалізувати масштабовану архітектуру додатку, що легко підтримується, оновлюється та відповідає сучасним вимогам до безпеки і якості програмного забезпечення.

Таким чином, розробка мобільного додатку для конвертації валют з використанням Kotlin є не лише актуальним програмним продуктом, що задовольняє потреби користувачів у зручному інструменті фінансових розрахунків, але й демонструє практичне застосування сучасних мов програмування та підходів до розробки ПЗ в умовах реального проєктування.

Мета кваліфікаційної роботи – розробити мобільний додаток для Android-платформи, який забезпечує точну і зручну конвертацію валют з використанням

Kotlin та підключенням до зовнішніх джерел даних (наприклад, API валютних курсів).

Об’єкт кваліфікаційної роботи – процеси розробки мобільного програмного забезпечення для конвертації валют.

Предмет роботи – архітектура, функціональність та реалізація мобільного застосунку з використанням Kotlin і сучасних API для отримання актуального валютного курсу.

Завдання до кваліфікаційної роботи:

- провести аналіз предметної області та існуючих аналогів;
- сформувати вимоги до мобільного додатку;
- вибрати інструменти, мову програмування та архітектурний підхід;
- спроєктувати структуру мобільного застосунку;
- реалізувати функціональність додатку з інтеграцією API валют;
- провести тестування та оцінити зручність використання інтерфейсу;
- розробити користувацьку документацію та провести UX/UI-аналіз.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

У сучасному цифровому середовищі мобільні застосунки відіграють ключову роль у забезпеченні користувачів швидким доступом до інформаційних, фінансових, соціальних та сервісних ресурсів. Зокрема, у сфері фінансів особливу актуальність набули застосунки, що дозволяють оперативно відстежувати валютні курси та здійснювати їх конвертацію. Це зумовлено високою динамікою валютного ринку, значною кількістю онлайн-транзакцій, міжнародною торгівлею, розвитком криптовалют і частими коливаннями національних валют.

На ринку програмного забезпечення вже існує низка мобільних додатків, які реалізують функції валютного калькулятора, серед яких найбільш поширеними є XE Currency, Currency Converter Plus, Easy Currency Converter, Google Finance та вбудовані сервіси банківських додатків. Більшість з них дозволяють працювати з широким спектром валют, переглядати історію курсів та оновлювати інформацію в реальному часі. Однак значна частина таких додатків або перевантажена зайвим функціоналом, або ж вимагає платної підписки для повного доступу до сервісів [1].

Більше того, багато рішень не орієнтовані на інтуїтивну взаємодію з користувачем або не підтримують офлайн-режим з останніми збереженими даними. Деякі з наявних додатків не мають достатньої підтримки української мови, або не відповідають вимогам до сучасного дизайну та швидкодії.

Аналіз сучасних рішень показує, що хоча потреба у таких додатках велика, на ринку все ще існує простір для створення простого, ефективного, легкого у використанні мобільного додатку, який базується на відкритих API джерелах валютних курсів (наприклад, ExchangeRate-API, Open Exchange Rates, Fixer.io) та має чистий, адаптивний інтерфейс.

Додаток XE Currency є одним з найбільш визнаних на міжнародному ринку: він «спостереження за курсом», підтримує конвертацію офлайн за збереженими даними та регулярно оновлюється. При старті додатку показує користувачу сторінку із дуже надлишковою інформацією, графіками, відсотковою різницею і т.д. Тому із перших секунд користувач так і не розуміє, як йому отримати просто конвертацію і шукає новий сервіс (рис. 1.1) [2].

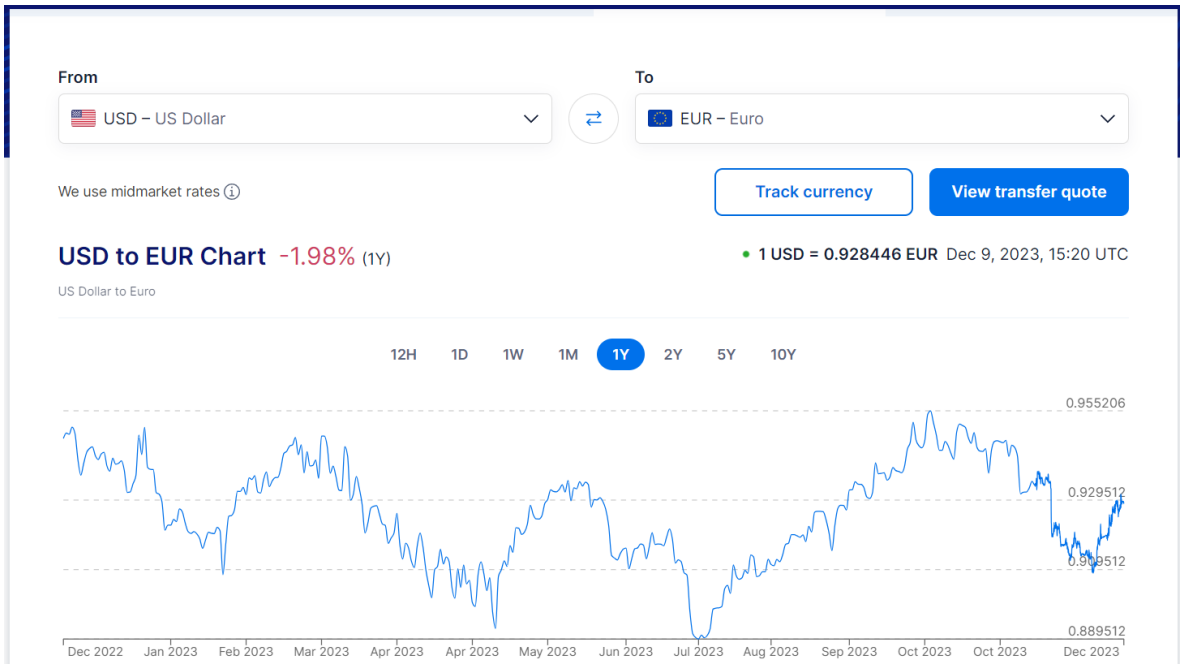

Рисунок 1.1 – Приклад інтерфейсу додатку конвертації [2]

Currency Converter Plus – ще один популярний мобільний застосунок, орієнтований на простоту використання та швидкий доступ до основної функції – конвертації валют. Він підтримує велику кількість валют, зокрема основні світові та менш популярні, а також криптовалюти й дорогоцінні метали. Додаток оснащений вбудованим калькулятором, який дозволяє миттєво перераховувати суми при зміні валют, що є зручним для подорожей, онлайн-покупок та фінансових розрахунків. Інтерфейс програми інтуїтивно зрозумілий, що робить його привабливим для користувачів без технічної підготовки. Водночас одним із недоліків є наявність рекламних банерів, які можуть відволікати та знижувати зручність користування. Для усунення реклами необхідно оформити платну підписку або придбати преміум-версію застосунку (рис. 1.2) [3].

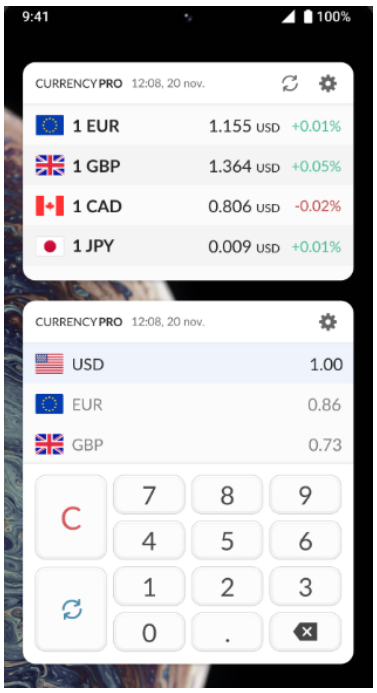

Рисунок 1.2 – Приклад інтерфейсу додатку конвертації [3]

Easy Currency Converter приваблює користувачів своєю швидкістю, офлайн-режимом і легкістю інтерфейсу. Проте функціонал даного застосунку доволі базовий, і він не має можливостей для візуалізації історії курсів або кастомізації конвертацій (рис. 1.3) [4].

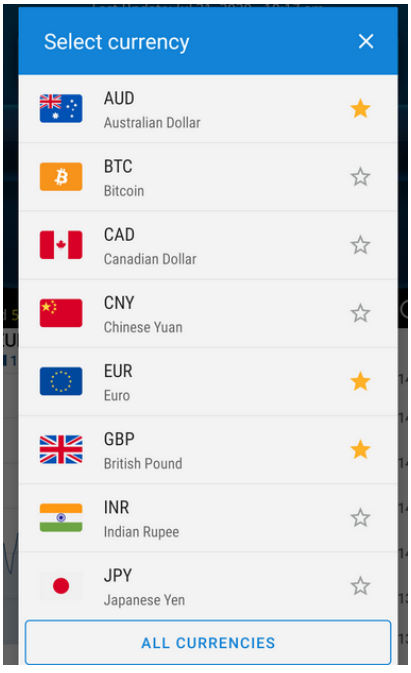

Рисунок 1.3 – Приклад інтерфейсу додатку конвертації [4]

Google Finance не є класичним мобільним додатком, проте його дані інтегруються в численні фінансові сервіси. Він забезпечує достовірні курси, однак потребує додаткових дій з боку користувача (наприклад, відкриття браузера, пошуку відповідного запиту), що знижує зручність у порівнянні з нативними додатками.

У таблиці представлено порівняння функціональних можливостей чотирьох популярних додатків для обміну валют за основними критеріями, такими як мова інтерфейсу, наявність офлайн-режиму, підтримка графіків та доступність платних функцій (табл. 1.1).

Таблиця 1.1 – Порівняльна характеристика популярних мобільних додатків для конвертації валют

Характеристика / додаток	XE Currency	Currency Converter Plus	Easy Currency Converter	Google Finance
Походження	Канада / США	США	Німеччина	США
Мова інтерфейсу (у т.ч. українська)	немає української	(немає української	немає української	немає української
Оновлення курсів у реальному часі	+	+	+	+
Кількість валют	>180	>160	>200	>100
Історія курсів / графіки	в підписці	частково	–	+
Офлайн-режим	+	+	+	–
Підтримка API НБУ	–	–	–	–
Наявність реклами	у підписці	у безкоштовній версії	у безкоштовній версії	–
Платні функції / підписка	+	+	+	–
Додаткові можливості	Сповіщення про курс, мультивалютна конверсія	Вбудований калькулятор	Проста інтеграція з виджетами	Фінансові новини, аналітика

Також багато банківських мобільних застосунків мають вбудовані конвертери валют, проте вони зазвичай обмежуються курсами лише відповідного банку, не дозволяють порівнювати курси різних установ або надають інформацію лише для підтримуваних рахунків.

Загальним недоліком багатьох із розглянутих рішень є:

– надмірна кількість функцій, які перевантажують інтерфейс і ускладнюють використання додатку для простої задачі конвертації;

– рекламні блоки та платні функції, що суттєво знижують зручність використання;

– відсутність адаптивного інтерфейсу, зручного для людей з різними пристроями або особливими потребами;

– застарілий дизайн, що не відповідає сучасним принципам UI/UX.

Це створює нішу для нових рішень, орієнтованих на простоту, легкість, інтеграцію з національними джерелами даних, відсутність реклами та можливість швидкої конвертації навіть при нестабільному інтернет-з’єднанні.

Вибір мови Kotlin для реалізації додатку зумовлений її сучасністю, офіційною підтримкою Google для Android-платформи, безпечним синтаксисом (null-безпека), підтримкою корутин для асинхронних запитів до API, та зручним механізмом обробки JSON-даних за допомогою бібліотек (наприклад, Gson або Moshi) [5].

У відповідь на ці недоліки кваліфікаційною роботою запропоновано концепцію мобільного додатку, який базується на таких принципах:

– мінімалістичний та інтуїтивний інтерфейс;

– максимальна точність курсів за рахунок підключення до зовнішніх API-сервісів, зокрема OpenExchangeRates;

– висока швидкодія завдяки використанню мови Kotlin та бібліотеки Retrofit;

– адаптивність і розширюваність архітектури за рахунок застосування патернів MVVM;

– відсутність зайвого функціоналу й акцент на головну дію – конвертацію валют.

Таким чином, у результаті огляду існуючих програмних рішень, дослідження технологій розробки та виявлення типових проблем в архітектурі й дизайні мобільних валютних конвертерів, можна зробити висновок про

доцільність створення нового програмного продукту. Він має бути орієнтований на простоту, швидкість, актуальні дані з відкритих, перевірених джерел.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Виходячи з визначеної мети – розробити мобільний додаток для Android-платформи, який забезпечує точну і зручну конвертацію валют з використанням Kotlin та підключенням до зовнішніх джерел даних – можна сформулювати такі завдання кваліфікаційної роботи:

- провести аналіз предметної області та існуючих аналогів;
- сформувати вимоги до мобільного додатку;
- вибрати інструменти, мову програмування та архітектурний підхід;
- спроєктувати структуру мобільного застосунку;
- реалізувати функціональність додатку з інтеграцією API валют;
- провести тестування та оцінити зручність використання інтерфейсу;
- розробити користувацьку документацію та провести UX/UI-аналіз.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

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

Задля того, щоб успішно розробити програмне забезпечення необхідно правильно спроектувати план виконання та розробки, визначити основні вимоги та технічне завдання, а також визначити основні проблеми, з якими може зіткнутися додаток при розгортанні для роботи з користувачами.

Аналіз вимог та постановка технічного завдання є одним із найперших і один із вагомих і важливих етапів розробки абсолютно будь-якого типу і цілі програмного забезпечення. Перед початком роботи необхідно завжди точно і ретельно досліджувати ринок, конкуруючі додатки, технології та інструменти, проблеми та майбутні проблеми в тому числі, з якими може зіткнутися додаток, тобто – оцінка ризиків. Це допоможе розробити список основних функції додатку та його можливості, а також майбутні можливості у нових версіях, до яких можна удосконалювати програму. Наприклад, для розробки додатку з конвертації валют, необхідно ще перед самою розробкою проводити даний аналіз і визначати основні функції такого додатку [6-7].

Правильно, гарно та інтуїтивно розроблений дизайн додатку забезпечує перше враження від користувача і впливає на те, як швидко додаток зможе розповсюджитися та набути популярності серед користувачів. Дотримання вимог UX – user experience, тобто врахування всіх нюансів та проблем, які можуть виникнути з поганим дизайном і плануванням сторінок, щоб користувач міг інтуїтивно користуватися функціоналом і в нього не виникало питань із тим, як користуватись певними частинами програми. Для додатку з конвертування валют – чим простіший дизайн, тим це буде краще для користувача, адже все що йому потрібно в процесі користування – вибрати поточну валюту, валюту в яку конвертувати та введення суми [8-9].

Виконання – розробка, написання коду та тестування додатку відповідає за технічну реалізацію додатку. Саме в цьому етапі визначаються основні архітектурні рішення по додатку – це вибір архітектурного патерну проектування, на якому будувати додаток.

Існує три найбільш відомих патернів проектування – такі як MVC (Model-View-Controller), MVVM (Model-View-ViewModel) та MVP (Model-View-Presenter). Зазвичай MVC патерн використовується для побудови Web додатків, патерн MVVM – зазвичай це для комп'ютерних додатків та для мобільних застосунків Android, а патерн MVP – менш популярний, загалом використовується для побудови мобільних крос-платформенних застосунків. Проте кожен із них повинен застосовуватись в залежності від задачі, а не від вибраної платформи. Дані патерни допомагають розробити застосунок легким для вдосконалення, швидким та надійним, стабільним.

У процесі розробки мобільного додатку для конвертації валют була обрана архітектурна модель MVVM (Model–View–ViewModel) як найбільш доцільна для реалізації структурованого, модульного Android-застосунку. Такий підхід дозволяє чітко розмежувати функціональні обов'язки між компонентами: Model відповідає за доступ до даних (наприклад, API-запити, обробку JSON-відповідей); ViewModel реалізує бізнес-логіку, зберігає стан інтерфейсу і виступає посередником між Model і View; View відображає інформацію для користувача та реагує на події (натискання кнопок, введення даних тощо).

Ця модель забезпечує гнучкість, спрощує тестування компонентів, дозволяє перевикористовувати код та полегшує подальше масштабування проєкту. Як зазначено у курсовій роботі, MVVM значно спрощує реалізацію взаємодії між інтерфейсом та логікою обробки даних [10].

Для кращого розуміння принципу роботи патерну MVVM у контексті проєкту, на рисунку 2.1 подано схематичну взаємодію між його основними компонентами. Зображена схема ілюструє, як дані передаються від моделі до представлення через ViewModel, забезпечуючи ефективну та швидку обробку користувацьких запитів. Такий підхід дозволяє кожному блоку виконувати

виключно свою роль, що знижує ризик помилок та підвищує стабільність програми.

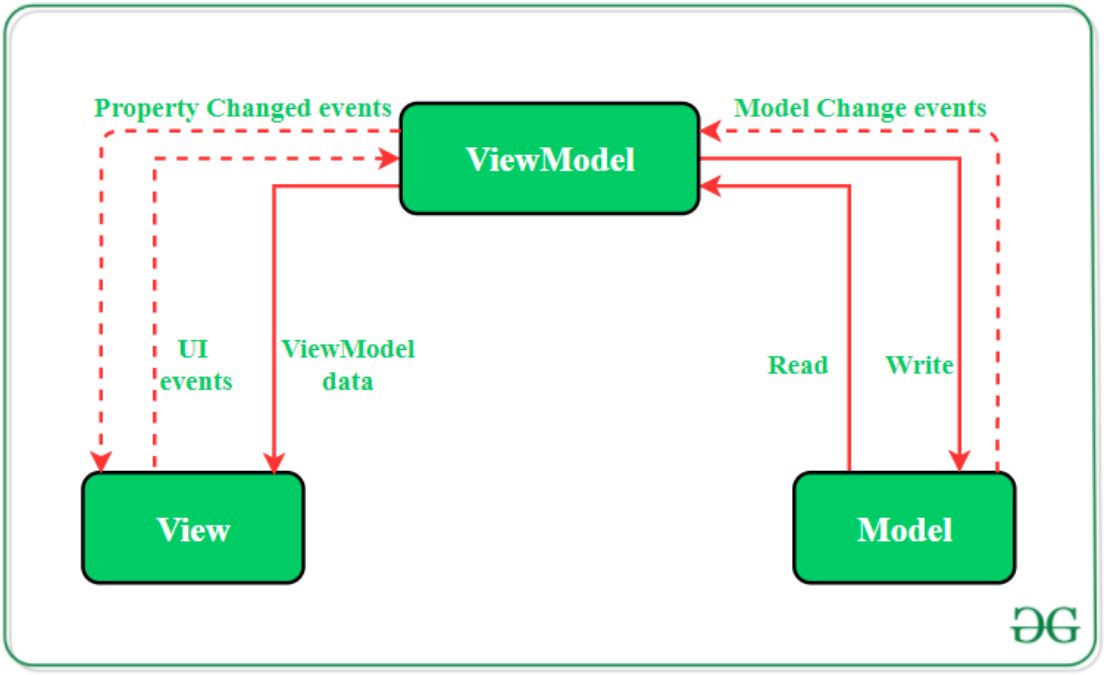

Рисунок 2.1 – Приклад роботи архітектурного патерну MVVM

Ціль: розробити інтуїтивно зрозумілий, швидкий і точний мобільний додаток для конвертації валют з використанням зовнішнього API (OpenExchangeRates), що функціонує на ОС Android і реалізований мовою Kotlin.

Основні задачі:

– забезпечити вибір вхідної та вихідної валюти;

– забезпечити введення суми для конвертації;

– реалізувати підключення до API з актуальними курсами валют;

– здійснити обчислення та відобразити результат користувачу;

– забезпечити зворотний зв’язок (перенаправлення) між екранами;

– реалізувати перевірку введених даних і обробку помилок.

Опишемо специфікацію вимог до ПЗ. Спочатку сформулюємо функціональні вимоги.

Система повинна забезпечувати:

– вибір двох валют зі списку (вихідна, цільова);

– введення числового значення суми;

– отримання актуального курсу з API;

– обчислення конвертованого значення;

– виведення результату на окремому екрані;

– відображення повідомлень про помилки (непідключення до інтернету, порожні поля, некоректний запит тощо).

Нефункціональні вимоги:

– додаток має бути написаний мовою Kotlin з використанням архітектури MVVM;

– комунікація з API повинна здійснюватися через бібліотеку Retrofit;

– інтерфейс має бути адаптивним, побудованим засобами XML;

– дані повинні зберігатися в оперативній пам’яті – кешування не реалізується;

– додаток має коректно працювати на пристроях Android 8+.

Для уточнення функціональних можливостей системи та визначення основних шляхів взаємодії користувача з додатком було сформульовано ключові сценарії використання. Вони представлені в таблиці 2.1.

Таблиця 2.1 – Сценарії використання мобільного додатку для конвертації валют

Сценарій	**Опис**
UC1. Запуск додатку	Користувач відкриває додаток, бачить екран із полями введення та списками валют
UC2. Вибір валют	Користувач вибирає вхідну та цільову валюту зі списку
UC3. Введення суми	Користувач вводить суму для конвертації
UC4. Виконання конвертації	Користувач натискає кнопку «Конвертувати», система звертається до API
UC5. Вивід результату	Результат конвертації виводиться на екран
UC6. Помилки	У разі помилки (порожні поля, немає з’єднання тощо) – система показує повідомлення
UC7. Повторна конвертація	Користувач повертається назад і здійснює нову операцію

У розробленій системі єдиним актором виступає користувач, який взаємодіє із застосунком у межах передбачених сценаріїв. Основними прецедентами є: запуск додатку, вибір валюти-джерела та валюти-призначення,

введення суми для обміну, отримання актуального курсу з API, а також перегляд результату конвертації на окремому екрані [11, 12]. У структурі взаємодії реалізовано зв’язки типу include, які відображають обов’язкову перевірку введених користувачем даних перед виконанням операції, а також зв’язки extend, які охоплюють необов’язкові дії, наприклад, виведення повідомлень про помилки у випадку неправильної взаємодії з інтерфейсом або проблем із мережею. Ця діаграма дозволяє описати функціональну поведінку системи на рівні очікувань користувача та основних варіантів використання (рис. 2.2).

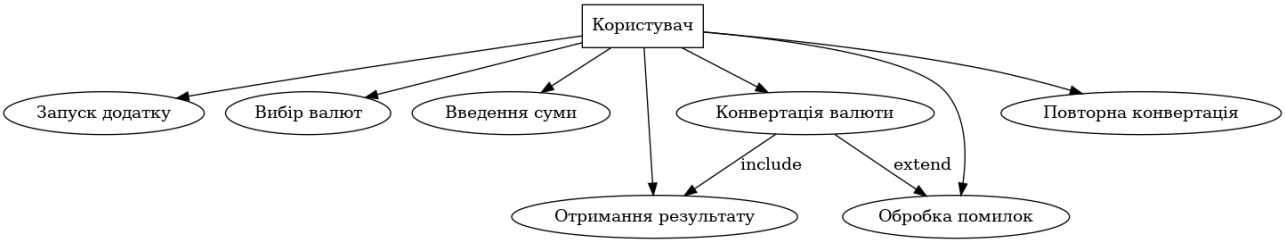

Рисунок 2.2 – Діаграма варіантів використання

У структурі програмного забезпечення, що розробляється, ключовими компонентами є чотири основні класи, які реалізують основну логіку та взаємодію між частинами системи. Центральним елементом виступає клас CurrencyConverterViewModel, який містить основну бізнес-логіку, зокрема методи для обробки запиту на конвертацію валют та управління станом інтерфейсу. Для здійснення зв’язку з віддаленим сервером реалізовано окремий інтерфейс ApiClient, побудований за допомогою бібліотеки Retrofit, який відповідає за відправлення HTTP-запитів та отримання курсів валют у форматі JSON.

Інтерфейс користувача представлений двома активностями – MainActivity та ResultActivity, які відповідають відповідно за введення даних і відображення результатів обчислень. Ці компоненти реалізують логіку представлення (View) та викликають відповідні методи ViewModel. Для зберігання та передачі валютних даних у системі передбачено клас CurrencyModel, який є

структурованим об’єктом (data class), що включає назви валют, базову валюту та поточний курс.

Взаємодія між цими класами є прямолінійною та чітко визначеною згідно з принципами архітектури MVVM: View звертається до ViewModel, яка в свою чергу ініціює запити до API через клієнтський інтерфейс, отримує та обробляє дані, після чого повертає результат для відображення у View. Така структура забезпечує гнучкість, модульність та полегшує супровід системи. Архітектура застосунку реалізована за шаблоном MVVM, як показано на рисунку 2.3.

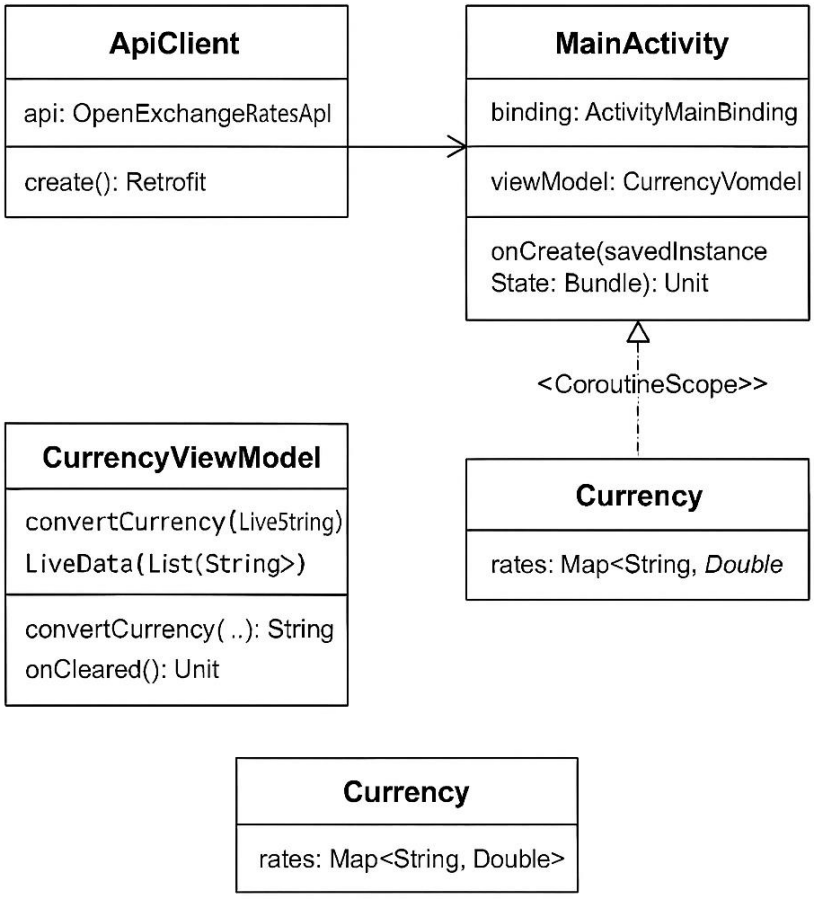

Рисунок 2.3 – Структура взаємодії компонентів у моделі MVVM для валютного конвертера

У межах реалізації основного сценарію використання системи – конвертації валюти – побудовано діаграму послідовності, яка відображає взаємодію між основними компонентами застосунку. Взаємодія розпочинається

з дій користувача, який через графічний інтерфейс (View) ініціює запит на конвертацію шляхом введення суми, вибору валют і натискання відповідної кнопки. Після цього інтерфейс передає керування до компонента ViewModel, який відповідає за обробку логіки запиту.

ViewModel звертається до інтерфейсу ApiClient, реалізованого з використанням бібліотеки Retrofit, для надсилання HTTP-запиту до віддаленого сервера – API валютного сервісу (наприклад, OpenExchangeRates). У відповідь API надсилає структуру даних у форматі JSON, яка повертається через ApiClient назад у ViewModel.

Після отримання та обробки відповіді ViewModel виконує розрахунок результату конвертації на основі поточного валютного курсу і передає підготовлену інформацію назад до інтерфейсу користувача. View відображає результат на екрані, після чого користувач бачить завершення процесу конвертації. Така послідовність забезпечує чіткий і контрольований потік даних, сприяє зменшенню кількості залежностей між компонентами, а також відповідає принципам архітектури MVVM, реалізованої у додатку. Для візуалізації логіки взаємодії між основними компонентами системи під час виконання операції конвертації валют було побудовано діаграму послідовності (рис. 2.4), яка ілюструє послідовність викликів та передачу даних між об’єктами.

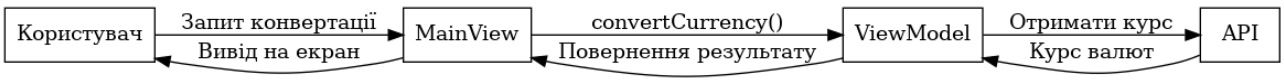

Рисунок 2.4 – Діаграма послідовності обробки запиту на конвертацію валют

Інформаційні потоки включають:

– отримання JSON-відповіді з API;

– парсинг у моделі Kotlin;

– відображення результату на UI.

Результати моделювання показали відповідність функціоналу заявленим вимогам. Обрана архітектура MVVM забезпечує розділення обов’язків між View та логікою, спрощує тестування й дозволяє легко оновлювати або масштабувати

систему. Використання Retrofit забезпечує стабільну роботу з API, а Kotlin – безпечну реалізацію з мінімальним ризиком помилок. Макети Figma підтверджують відповідність інтерфейсу принципам UX/UI, а реалізація через Android Studio – підтримку нативних можливостей ОС Android [13].

Для повнішого розуміння структури розроблюваної системи побудовано діаграму розгортання, яка відображає фізичні компоненти, що взаємодіють під час виконання операції конвертації валют. Зокрема, у моделі вказано вузли «Мобільний пристрій», «API-сервер» та «Інтернет-з’єднання», що демонструють фізичні залежності між елементами інфраструктури. Такий підхід дозволяє візуалізувати архітектуру системи та виявити потенційні вузькі місця при обміні даними (рис. 2.5).

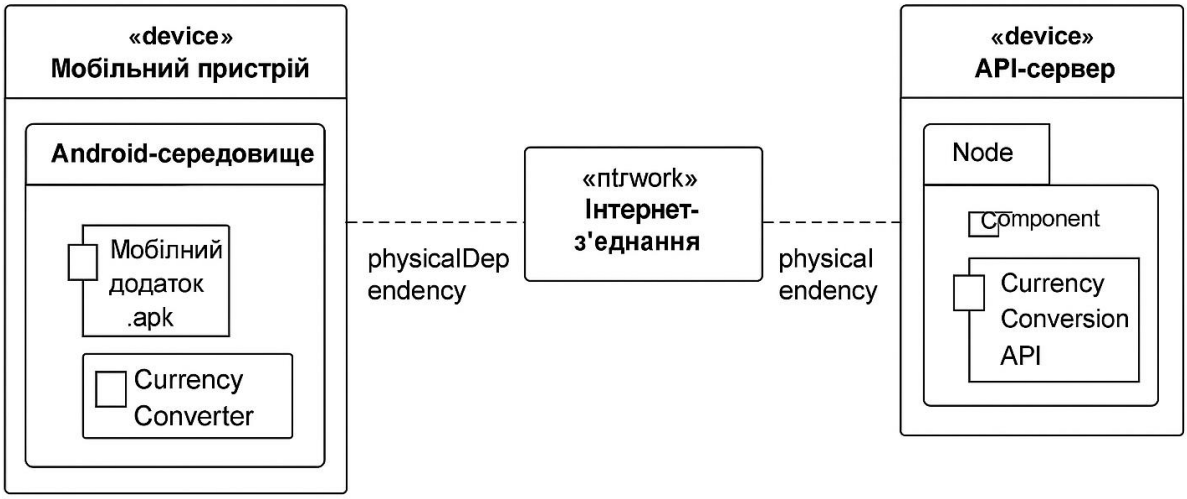

Рисунок 2.5 – Діаграма розгортання системи

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

У процесі реалізації мобільного додатку для конвертації валют необхідно обрати ефективні засоби, методи та інструменти, які дозволяють якісно вирішити поставлені завдання розробки програмного забезпечення. Враховуючи вимоги до додатку, такі як стабільність роботи, зручність інтерфейсу, точність

розрахунків і оперативний обмін з віддаленим сервером, доцільно провести аналіз наявних інструментів, доступних для розробки під платформу Android.

Серед мов програмування основними кандидатами були Java, Kotlin і Dart (у зв'язку з поширенням Flutter). Проте, зважаючи на офіційну підтримку Kotlin з боку Google, її лаконічний синтаксис, кращу безпеку при роботі з null-значеннями, а також зручну інтеграцію з архітектурою MVVM, саме Kotlin було обрано основною мовою реалізації додатку. Розробка здійснюється в середовищі Android Studio, яке забезпечує всі необхідні засоби для повного циклу розробки – від побудови графічного інтерфейсу до емуляції пристроїв і налагодження коду [14, 15].

У контексті побудови архітектури застосунку використано патерн MVVM (Model–View–ViewModel), який дозволяє чітко відокремити інтерфейс користувача від логіки обробки даних. Такий підхід сприяє легкості підтримки коду, масштабуванню і підвищенню тестованості компонентів. Комунікація з віддаленим сервером реалізується через HTTP-клієнт Retrofit, який спрощує виконання REST-запитів та автоматизує обробку JSON-даних. Для серіалізації та десеріалізації JSON-структур використано бібліотеку Gson, що дозволяє легко перетворювати дані у Kotlin-моделі.

Для отримання курсів валют обрано сервіс OpenExchangeRates, який надає надійні й актуальні дані у форматі JSON, що легко інтегруються у мобільний застосунок. Обмін з сервером виконується через HTTPS-протокол, що гарантує безпеку передачі інформації.

Алгоритм конвертації валют реалізується на основі простої математичної моделі, яка передбачає множення суми, введеної користувачем, на коефіцієнт валютного курсу, отриманий з API. За необхідності використовується нормалізація курсу щодо базової валюти (наприклад, долара США), а також здійснюється округлення результату до необхідної кількості знаків. Цей алгоритм є лінійним за складністю, що забезпечує високу швидкодію застосунку.

Візуальна частина додатку створена з урахуванням принципів UX/UI-дизайну. Інтерфейс користувача реалізовано у вигляді XML-макетів, де

передбачено окремі екрани для введення вхідних даних та відображення результату. Взаємодія користувача з додатком реалізується через вибір валюти зі списку, введення числової суми та отримання результату натисканням кнопки. Для зручності користування передбачено обробку помилок, відображення повідомлень про відсутність підключення до мережі або некоректний ввід.

У процесі проєктування програмного забезпечення були побудовані UML-діаграми: діаграма класів, що ілюструє структуру взаємодії між View, ViewModel, ApiClient та моделлю валют; діаграма прецедентів, яка описує сценарії взаємодії користувача з додатком; та діаграма послідовності, яка демонструє послідовність обміну повідомленнями між компонентами системи при виконанні основного сценарію – конвертації валюти. Також сформовано логічну блок-схему алгоритму обробки запиту конвертації з урахуванням обробки помилок.

Таким чином, вибір засобів і методів для розробки мобільного додатку базується на аналізі їх функціональності, продуктивності, зручності в реалізації та відповідності сучасним вимогам до розробки програмного забезпечення. Обрані рішення дозволяють ефективно реалізувати поставлену задачу з розробки зручного, функціонального і сучасного мобільного додатку для конвертації валют.

РОЗДІЛ 3
РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ КОНВЕРТУВАННЯ КУРСУ ВАЛЮТ

3.1 Практична реалізація об’єкта проектування

Проаналізувавши відповідні проблеми наявних API сервісів, для розробки додатку конвертування валют найкращим варіантом підходить OpenExchangeRatesAPI.

Даний API використовує метод авторизації через створення відповідного посилання на розроблюваний сервіс і присвоєння йому ідентифікатора – AppId. Даний ідентифікатор дозволить через систему запитів отримати потрібну інформацію із серверу Open Exchange Rates. Тому для початку потрібно створити AppId на сервісі (рис. 3.1).

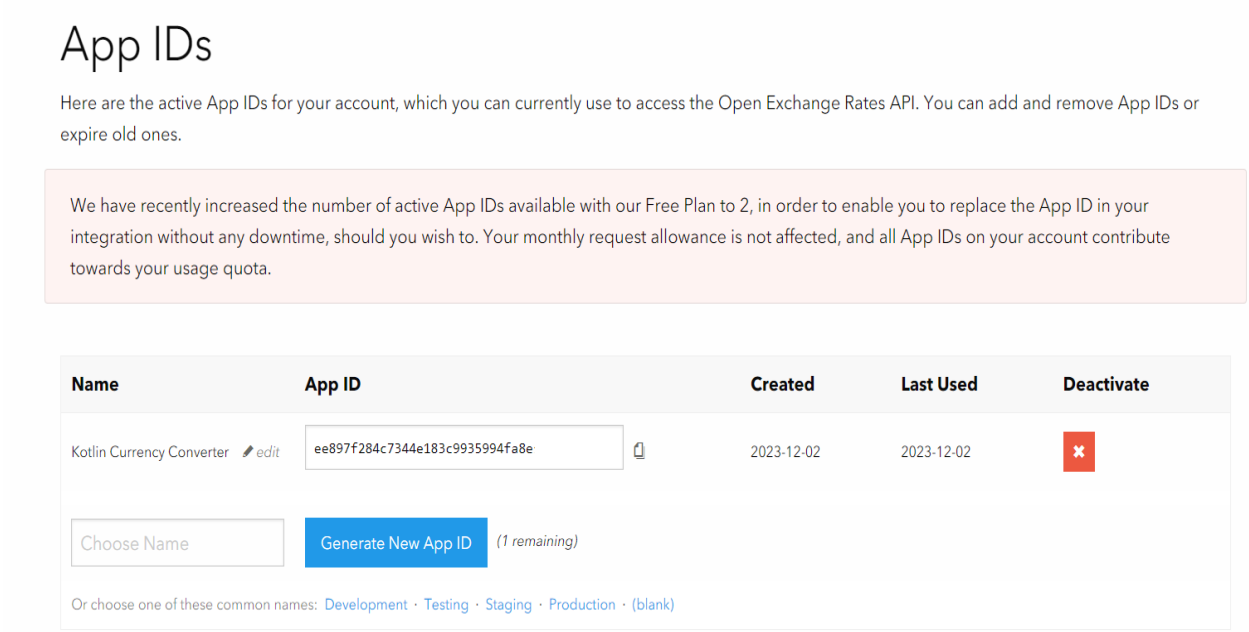

Рисунок 3.1 – Створення AppId на сервісі OpenExchangeRates

Для проведення успішного запиту, потрібно у запит додавати інформацію, яка буде підтверджуватись на сервері API. Створений AppId і є тією інформацією, згідно документації API сервісу, посилання запиту потрібно, щоб містило в собі параметр query і відповідний AppId (рис. 3.2).

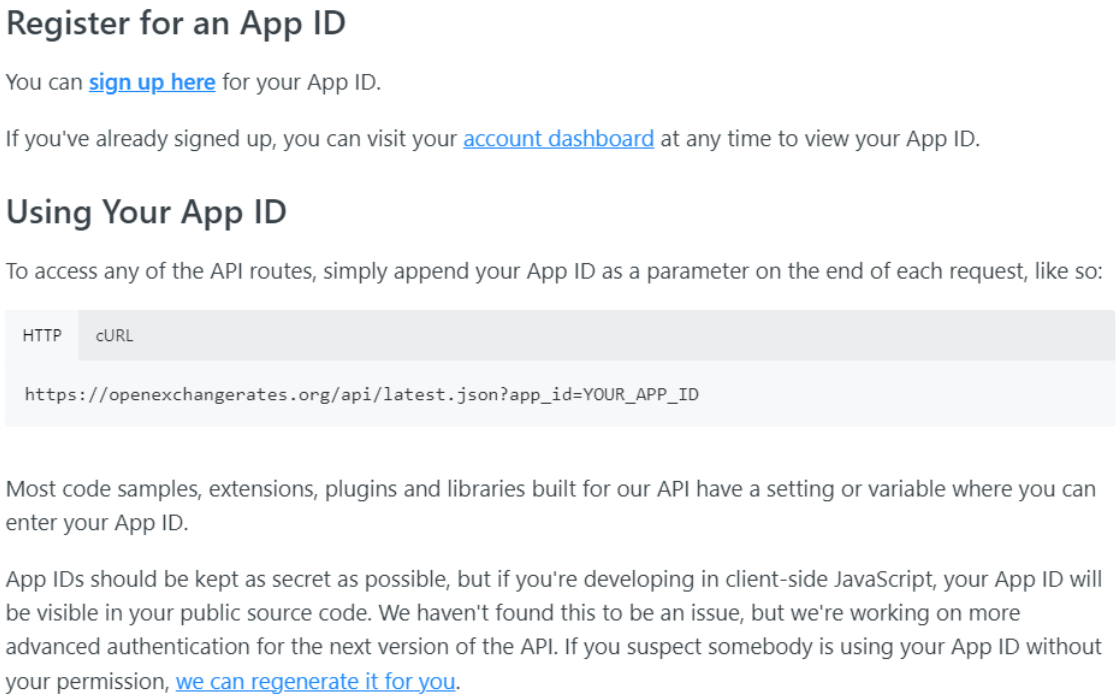

Рисунок 3.2 – Використання авторизації у запиті на OpenExchangeRates

Як уже зазначалося раніше і проаналізувавши всі вище сказані пункти, проблеми та недоліки додатків із проблемним інтерфейсом, інтерфейс даного додатку розроблюватиметься з врахуванням усіх згаданих проблем. Інтерфейс додатку повинен бути максимально спрощеним та інтуїтивним, що дасть змогу користувачам розібратися у додатку з першого погляду і успішно використати його в своїх цілях. Для того, щоб час на розробку інтерфейсу використовувався з максимальною економією ресурсів та коштів, прийнято макет інтерфейсу виготовляти у сторонніх дизайн ресурсах і сервісах, адже при розробці інтерфейсу на цих сервісах, етапи дають зрозуміти основні недоліки додатку і швидко їх виправляти, натомість якби інтерфейс розроблявся прямо при розробці у коді – це витрачало би значно більше ресурсів, тому попередньо узгоджений інтерфейс додатку – є однією із головних цілей, що дозволить усунути появу багатьох проблем, як при розробці, так і при розгортанні додатку.

Попередній макет інтерфейсу програмного забезпечення розроблятиметься у системі Figma. Дана система дозволяє користуватися потужними інструментами, які з легкістю і швидко дозволяють створити попередній вигляд сторінки із усіма потрібними параметрами. Для початку

потрібно визначити основні елементи взаємодії із користувачем – це кнопки, поля вводу та меню вибору валюти (рис. 3.3).

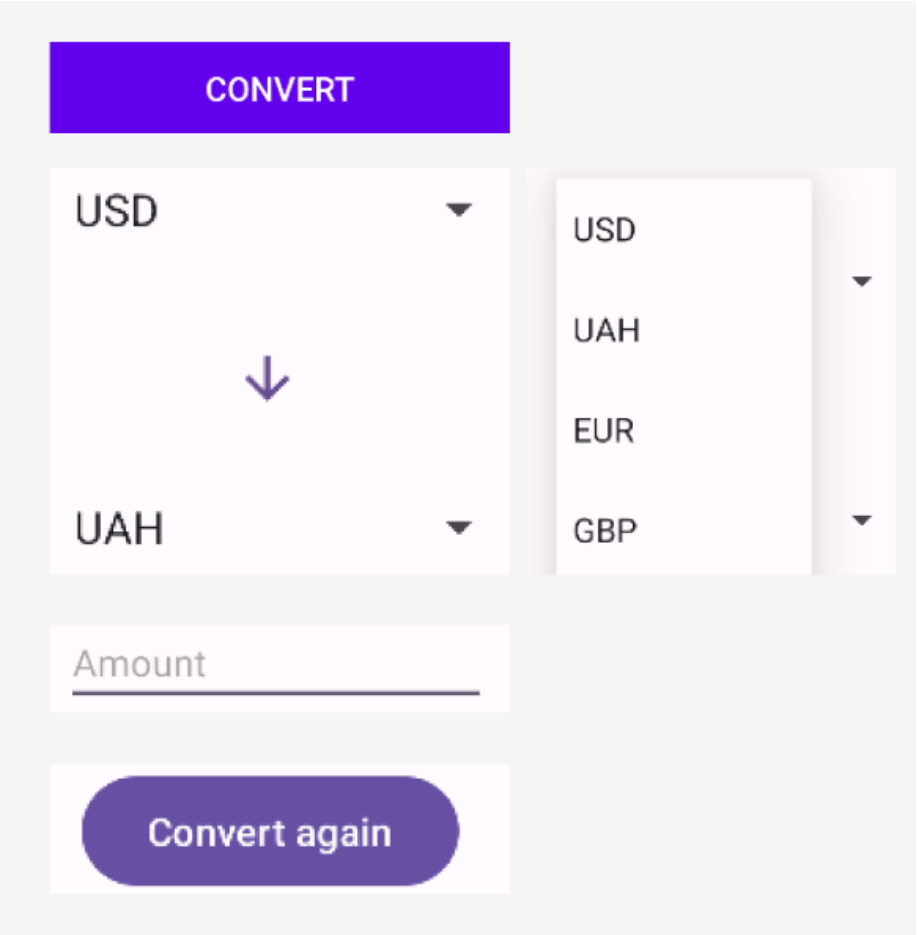

Рисунок 3.3 – Елементи інтерфейсу для взаємодії з користувачем

Розроблюваний додаток повинен мати лише 2 основних сторінки – перша сторінка відповідатиме за вибір валюти із якої конвертувати та вибір валюти у яку конвертувати, поле вводу суми, яку потрібно конвертувати та кнопку, яка буде робити запит, конвертувати, робити навігацію на другу сторінку, яка і буде показувати результат. Поєднавши усе вище сказане і уже розроблені елементи керування додатком у сторінки – можна побудувати попередній макет обох сторінок, з якими взаємодіятиме користувач (рис. 3.4), що дозволяє візуалізувати майбутню логіку взаємодії та краще спроєктувати UX/UI.

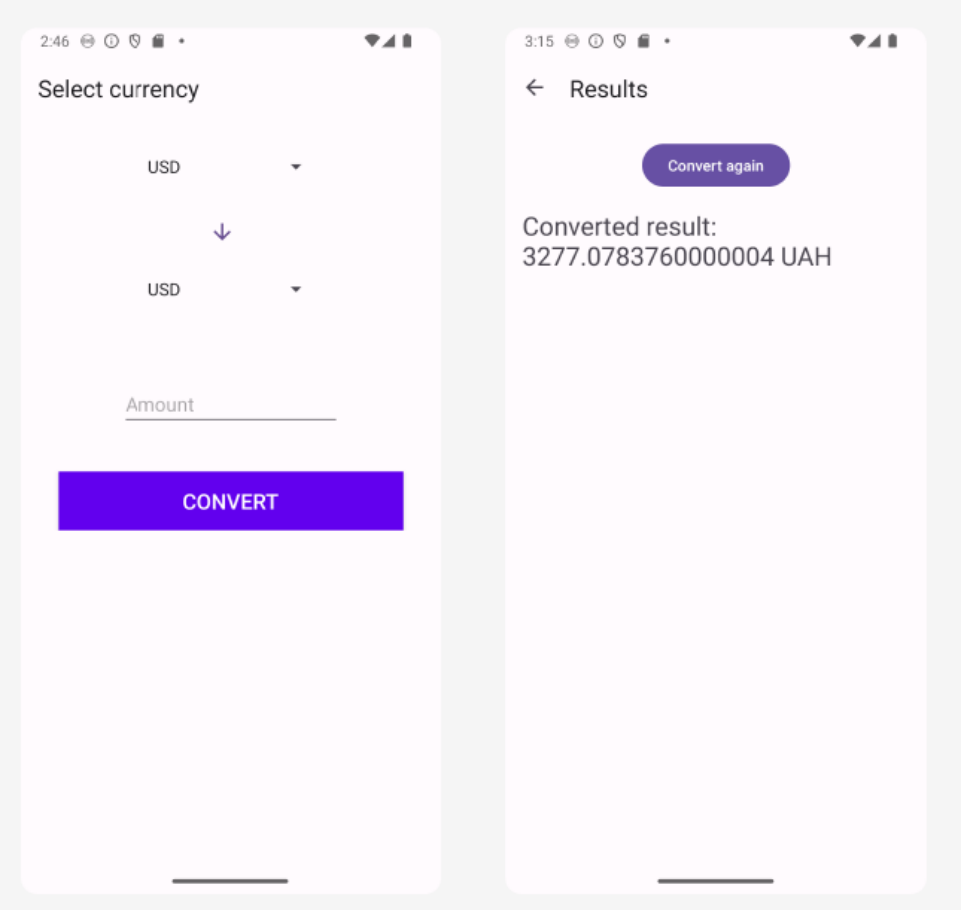

Рисунок 3.4 – Макет сторінок додатку

Дана реалізація інтерфейсу програмного додатку забезпечить легке користування програмою, інтуїтивно і швидко виконувати поставлені завдання перед користувачем і задовільнити весь процес користування відсутністю надлишковим та набитим інтерфейсом.

Наступним етапом розробки даного інтерфейсу є його імплементація та розробка саме у кодовій частині. Найкраще всього розпочинати із базових елементів, а саме з таких, як налаштування навігації по додатку. Даний програмний продукт має лише дві сторінки, тому реалізація навігації є доволі простою. Для початку потрібно налаштувати основну сторінку із її елементом AppBar та включаючим елементом content та створити об’єкт сторінки (content), який буде містити в собі об’єкт навігації, тобто файли activity_main.xml і content_main.xml (рис. 3.5).

activity_main.xml

Code Split Design

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
    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:fitsSystemWindows="true"
    tools:context=".MainActivity">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:fitsSystemWindows="true">

        <com.google.android.material.appbar.MaterialToolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize" />

    </com.google.android.material.appbar.AppBarLayout>

    <include layout="@layout/content_main" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

content_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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior">

    <fragment
        android:id="@+id/nav_host_fragment_content_main"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:defaultNavHost="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:navGraph="@navigation/nav_graph" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Рисунок 3.5 – Базові файли інтерфейсу додатку

Наступним етапом є створення фрагментів та імплементації навігації між ними. Як уже було вище зазначено – перша сторінка складатиметься із елементів вибору валюти, поля вводу суми та кнопки для підтвердження. Дані елементи використані із бібліотеки AppCompat – Spinner, EditText і Button (ліст. 3.1).

Лістинг 3.1 – Реалізація основної першої сторінки додатку

```
<androidx.appcompat.widget.AppCompatSpinner
    android:id="@+id/destination_spinner"
    android:layout_width="170dp"
    android:layout_height="50dp"
    android:layout_marginStart="16dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="16dp"
    android:entries="@array/currencies"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/arrow_image" />
<androidx.appcompat.widget.AppCompatEditText
    android:id="@+id/amount_edit_text"
    android:layout_width="205dp"
    android:layout_height="45dp"
    android:layout_marginStart="16dp"
    android:layout_marginTop="60dp"
    android:layout_marginEnd="16dp"
    android:hint="@string/hint_amount"
```

```
    android:inputType="number"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/destination_spinner"
/>
<androidx.appcompat.widget.AppCompatButton
    android:id="@+id/convert_button"
    android:layout_width="322dp"
    android:layout_height="55dp"
    android:layout_marginStart="16dp"
    android:layout_marginTop="40dp"
    android:layout_marginEnd="16dp"
    android:background="@color/design_default_color_primary"
    android:text="@string/convert"
    android:textColor="@color/white"
    android:textSize="20sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/amount_edit_text" />
```

кінець лістингу 3.1

Розробка другої сторінки не представляє собою великої складності, адже вона складається лише із кнопки та текстового представлення, яке повинно прив’язуватися до відповідного результату конвертування та показувати користувач результат конвертації його суми. Також кнопка повинна відразу користувачу показувати, що він може повернутись назад на першу основну сторінку, щоб зробити повторну конвертацію із новою валютою або сумою. Після побудови другої сторінки з результатами отримаємо готову реалізацію інтерфейсу та схему навігації, попередньо реалізувавши її у файлі nav_graph.xml.

Для реалізації логічної частини додатку необхідно розбити всю розробку додатку на деякі пункти та створити план, що дозволить швидше розробити частини та поєднати їх між собою, а саме:

– реалізація під’єднання до API та взаємодія із зовнішнім API;

– отримання даних з API, парсинг відповідних даних;

– реалізація логіки конвертації;

– відповідне підтягування отриманих даних до елементів View.

Для того, щоб успішно взаємодіяти із зовнішніми ресурсами, у Android і Kotlin існує безліч інструментів, які дозволяють це зробити. Одним із найбільш

розповсюджених інструментів, який є доволі зручним та широко розповсюдженим – є Retrofit. Retrofit представляє собою один із найбезпечніших клієнтів HTTP протоколу для Android та Java. Даний інструмент дозволяє доволі помітно спростити використання API у додатку. Використання даного Retrofit пакету надає велику кількість переваг, а саме:

– надсилання запитів різних методів (Get, Post і тд);

– отримання відповідей із запитів;

– парсинг відповідей із запитів різних форматів у сутності та моделі Kotlin;

– зручний метод відловлювання помилок запитів та відповідей від зовнішніх ресурсів.

Для того, щоб додати Retrofit до проєкту необхідно додати визначені залежності у файлі build.gradle (ліст. 3.2).

Лістинг 3.2 – Додавання залежностей Retrofit до проєкту

```
dependencies {
    implementation("androidx.core:core-ktx:1.9.0")
    implementation("androidx.appcompat:appcompat:1.6.1")
    implementation("com.google.android.material:material:1.10.0")

implementation("androidx.constraintlayout:constraintlayout:2.1.4")
    implementation("androidx.navigation:navigation-fragment-
ktx:2.7.5")
    implementation("androidx.navigation:navigation-ui-ktx:2.7.5")
    implementation("com.squareup.retrofit2:retrofit:2.9.0")
    implementation("com.squareup.retrofit2:converter-gson:2.9.0")

    testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")
    androidTestImplementation("androidx.test.espresso:espresso-
core:3.5.1")
}
```

кінець лістингу 3.2

Наступним кроком у розробці фундаменту для роботи API сервісу необхідно створити об’єкт Retrofit із його базовими параметрами (такі як основне посилання на API, додати фабрику конвертування, у даному випадку це JSON

конвертер, так як відповіді із API сервісу надсилаються у JSON форматі, так як це дозволяє зекономити ресурси та тримати в оперативній пам’яті пристрою менші дані, що дозволяє збільшити швидкість обробки результату та зменшити навантаження на пам’ять пристрою та відповідно під’єднати до нього інтерфейс, який буде відповідати за посилання запитів на сервер API (ліст. 3.3).

Лістинг 3.3 – Створення об’єкту Retrofit

```
private val api = Retrofit.Builder()
    .baseUrl("https://openexchangerates.org/")
    .addConverterFactory(GsonConverterFactory.create())
    .build()
    .create(OpenExchangeRatesApi::class.java)
```

кінець лістингу 3.3

Для забезпечення потрібного функціоналу конвертації валют необхідно звертатись до API за двома запитами – для отримання усього списку валют, який підтримується, для того, щоб подані елементи містились у меню для користувача і забезпечували точність даних, адже зберігаючи список валют, які підтримуються, у коді, може викликати помилки у роботі програми, якщо сервіс API не буде більше підтримувати певну валюту і тоді конвертація вибраної користувачем валюти може бути неможливою і приведе до помилок програми, тому усі валюти потрібно брати саме із зовнішнього ресурсу. Для забезпечення точної конвертації, потрібно звертатися до сервісу задля того, щоб отримати курси певних валют на момент запиту і провести обчислення на основі вибраних у додатку даних користувачем і отриманих даних із API (ліст. 3.4).

Лістинг 3.4 – Імплементація інтерфейсу для роботи із API

```
package com.example.currencyconverter.api

import retrofit2.Call
import retrofit2.http.GET
import retrofit2.http.Query

interface OpenExchangeRatesApi {

    @GET("currencies.json")
```

```
    fun getCurrencies(@Query("app_id") appId: String):
Call<Map<String, String>>

    @GET("latest.json")
    fun getLatestRates(@Query("app_id") appId: String):
Call<Currency>
}
```

кінець лістингу 3.4

Успішне надсилання запитів на зовнішній ресурс забезпечений багатьма чинниками і залежить від них, такі як підключення до інтернет мережі, існування безперебійного підключення зовнішнього сервісу та інші. При надсиланні запитів може виникнути будь-яка помилка і для того, щоб забезпечити користування додатком в різних умовах, потрібно забезпечити надсилання запитів у додатку через спеціальний механізм, котрий буде відловлювати помилки та повідомляти користувача про неуспішність певної дії. В тому числі відловлювання помилок необхідно забезпечити і в самому процесі користування інтерфейсом, наприклад, для того, щоб користувач не міг конвертувати валюту у таку ж саму вибрану валюту, або якщо поле вводу суми залишилось пустим, або дорівнює нулю – показувати помилку із її описом. Для цього будуть використовуватися спеціальні методи перевірки полів, а показування повідомлення буде відбуватись за допомогою пакету Toast.

Заключною частиною розробки даного програмного забезпечення є імплементація та реалізація логіки конвертування валюти. Дана логіка повинна зчитувати дані із елементів інтерфейсу Spinner, вихідний код валюти і код валюти у який конвертувати, зчитувати суму, яка введена у поле і проводити конвертування. Курси валют будуть стягуватись із зовнішнього ресурсу саме тут запитом і відповідно поміщати його у раніше створені сутності та моделі. Результат конвертації повинен проводитись відповідно до базової валюти, яка установлена у зовнішнього сервісу API – у даному випадку це USD. Далі на основі вибраних даних і отриманих із API – проводити конвертацію та робити навігацію до сторінки результатів (ліст. 3.5).

Лістинг 3.5 – Реалізація конвертування валюти

```
private val baseRateName = "USD"

private fun convertCurrency(
    rates: Map<String, Double>?,
    source: String,
    destination: String,
    amount: Double
): String {
    val sourceToUsdRate =
        rates?.get(source) ?: throw RuntimeException("Undefined
value: $source")
    val destinationToUsdRate =
        rates[destination] ?: throw RuntimeException("Undefined
value: $destination")

    if (source == baseRateName) {
        return formatResult(amount * destinationToUsdRate,
destination)
    }

    if (destination == baseRateName) {
        return formatResult(amount / sourceToUsdRate, destination)
    }

    return formatResult(amount * (destinationToUsdRate /
sourceToUsdRate), destination)
}
```

кінець лістингу 3.5

Дані розроблені частини коду узагальнені у додатку та успішно функціонують один між одним та забезпечують успішно функціонал вибору, отримання курсу валют та інших даних із API і конвертацію валют.

3.2 Тестування та налагодження системи

Після завершення етапу реалізації мобільного додатку наступним обов’язковим етапом є проведення тестування та налагодження. Це дозволяє виявити можливі помилки, перевірити відповідність функціональності встановленим вимогам та забезпечити стабільну роботу програмного продукту в різних умовах.

У процесі тестування основна увага приділяється перевірці працездатності всіх функціональних компонентів системи: взаємодії з API, коректності обчислень, обробки винятків і помилок введення, а також стабільності навігації між екранами. Окремо перевіряється працездатність додатку в умовах відсутності інтернет-з’єднання, при введенні некоректних даних (наприклад, нульове значення або порожні поля), а також при спробі виконання повторної конвертації.

Функціональне тестування проводиться на реальних та емуляторних пристроях із різними версіями Android (від 8.0 і вище). Також перевіряється, чи правильно працює введення значень у EditText, чи зберігається стан полів після повороту екрана, і чи не виникає конфліктів у логіці ViewModel при надходженні асинхронної відповіді від API.

В таблиці 3.1 наведено приклади тест-кейсів, що демонструють типові сценарії використання додатку.

Таблиця 3.1 – Результати функціонального тестування мобільного додатку

№	Назва тесту	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Запуск додатку	–	Завантажується головний екран	Завантажується	Пройдено
2	Вибір валют	USD → EUR	Валюти вибрано, елементи оновлено	Вибрано	Пройдено
3	Введення некоректної суми	abc	Виводиться повідомлення про помилку	Помилка відображена	Пройдено
4	Відсутність інтернету	Вимкнене підключення	Повідомлення: «Немає з’єднання з мережею»	Повідомлення є	Пройдено
5	Коректна конвертація	Вибрано: USD → PLN, сума: 100	Відображається результат (наприклад, 431.21 PLN)	Результат вірний	Пройдено
6	Повторна конвертація	Нові дані: EUR → GBP, сума: 50	Новий результат виведено	Результат виведено	Пройдено

Особлива увага приділяється обробці виключних ситуацій. Було протестовано наступні сценарії:

– запуск додатку без доступу до Інтернету;
– надсилання запиту з неправильним або недійсним AppId;
– введення некоректних числових значень або символів;
– ситуації, коли API тимчасово недоступне.

Після кожного запуску і кожної модифікації коду проводиться регресійне тестування для перевірки, чи не було порушено роботу існуючих функцій. Для цього використовуються як вбудовані інструменти Android Studio (логування, logcat, debug), так і модулі тестування, включені через JUnit та Espresso для UI-тестів.

Налагодження здійснюється за допомогою логування в Logcat, а також за допомогою повідомлень Toast, які інформують користувача про стан програми, особливо у випадках помилок. Перевірка коректності формул конвертації здійснюється шляхом порівняння результатів із офіційними валютними калькуляторами (наприклад, Google, XE.com).

На основі результатів тестування всі виявлені недоліки усуваються. Після проходження всіх сценаріїв із позитивним результатом додаток вважається готовим до використання.

3.3 Розробка документації для програмного продукту

Документація є важливою складовою процесу життєвого циклу програмного забезпечення. Вона забезпечує структурований опис архітектури системи, її функціональних можливостей, інтерфейсів, способів встановлення та експлуатації. Повнота та якість документації значно впливають на ефективність супроводу та масштабування системи, а також на зручність її використання кінцевими користувачами.

У межах проєкту з розробки мобільного додатку для конвертації валют на платформі Android створено повний комплект документації, який охоплює як технічні, так і користувацькі аспекти.

Керівництво користувача містить покрокові інструкції щодо встановлення додатку, першого запуску, взаємодії з головним екраном, вибору вихідної та цільової валюти, введення суми та отримання результату конвертації. Особливу увагу приділено інструкціям на випадок виникнення помилок (відсутність інтернету, некоректні дані, порожні поля тощо), а також поясненням до повідомлень, які з'являються у таких випадках. Інтерфейс описано з використанням скріншотів та пояснень до кожного елемента керування (список валют, поле введення, кнопка конвертації тощо).

Технічна документація для розробників включає опис архітектури системи, побудованої на основі шаблону MVVM (Model-View-ViewModel), з поясненням ролі кожного компонента. Зокрема, описано призначення класів:

– MainActivity – відповідає за відображення інтерфейсу та взаємодію з користувачем;

– CurrencyConverterViewModel – виконує обробку даних, зберігає стан інтерфейсу та ініціює запити до API;

– ApiClient – реалізує клієнтську частину взаємодії з OpenExchangeRates API за допомогою Retrofit;

– CurrencyModel – містить структуру для збереження отриманих курсів валют.

UML-діаграми надають графічну інтерпретацію архітектурного рішення. У документацію включено:

– діаграму класів – ілюструє зв'язки між основними компонентами системи;

– діаграму варіантів використання – описує основні сценарії взаємодії користувача із додатком;

– діаграму послідовності – демонструє порядок викликів між об'єктами під час операції конвертації;

– діаграму розгортання – відображає фізичне розміщення компонентів та середовищ їх виконання.

Фрагменти коду та конфігураційні файли включають найважливіші приклади реалізації, такі як метод convertCurrency, виклики до API через Retrofit, приклади обробки LiveData, а також вміст файлу build.gradle, що містить інформацію про залежності проєкту.

Коментарі до коду додаються безпосередньо в кожному програмному модулі для пояснення логіки роботи функцій, параметрів, особливостей обробки помилок, що полегшує розуміння та подальше обслуговування проєкту іншими розробниками.

Уся створена документація структурована згідно з рекомендованими підходами до розробки технічної та експлуатаційної документації в галузі програмної інженерії. Такий підхід дозволяє забезпечити не лише прозорість і повторюваність процесу розробки, а й створити ґрунтовну базу для подальшого розвитку або адаптації додатку під інші задачі.

3.4 UX/UI-аналіз розробленої системи

Впровадження програмного продукту є завершальним етапом життєвого циклу програмного забезпечення, який передбачає перенесення розробленої системи до середовища кінцевого користувача, її налаштування, первинне тестування на реальних пристроях, а також підготовку до подальшої експлуатації.

Мобільний додаток для конвертації валют було впроваджено на платформі Android із використанням сучасного стеку технологій: мова програмування Kotlin, архітектурний шаблон MVVM, бібліотека Retrofit для взаємодії з API, Android Jetpack компоненти для управління станом та навігацією. Збірка проєкту виконувалася в середовищі Android Studio Arctic Fox і вище.

Інсталяція додатку здійснюється стандартним способом – шляхом встановлення APK-файлу через Android Debug Bridge (ADB) або безпосередньо з Android Studio на фізичний пристрій. Після встановлення додаток може бути запущений без додаткових налаштувань, за умови наявності інтернет-з’єднання.

У процесі впровадження було протестовано:

– стабільність запуску додатку на пристроях із різними розмірами екранів;

– коректність відображення інтерфейсу на смартфонах із темною та світлою темами;

– роботу додатку на Android-версіях 8.0, 10.0 та 13.0;

– взаємодію з OpenExchangeRates API з використанням реального API-ключа.

Для подальшої експлуатації користувачеві достатньо мати мобільний пристрій з операційною системою Android 8.0 або новішою, підключення до мережі Інтернет та дозволи на використання мережевого трафіку. Додаток не зберігає персональні дані, не потребує реєстрації та не використовує зовнішні сервіси автентифікації, що спрощує його використання та мінімізує ризики, пов’язані з конфіденційністю.

У разі оновлення API або зміни специфікації сервера (наприклад, перехід на інший формат відповіді або зміну базової URL-адреси), додаток може бути адаптовано шляхом модифікації відповідних інтерфейсів Retrofit та оновлення ViewModel. Для спрощення подальшого обслуговування проєкту передбачено документацію, опис архітектури та структурований вихідний код з коментарями.

Таким чином, мобільний додаток повністю готовий до використання кінцевими користувачами, демонструє стабільну роботу в цільовому середовищі та не потребує спеціальної підготовки для встановлення чи налаштування.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було реалізовано повний цикл розробки мобільного додатку для конвертації валют з використанням мови програмування Kotlin та середовища Android Studio.

У ході роботи було проаналізовано предметну область, виявлено потребу у створенні нового валютного конвертера з мінімалістичним та зручним інтерфейсом. Проаналізовано низку популярних мобільних додатків, визначено їх переваги й недоліки, що дозволило обґрунтувати доцільність створення власного рішення.

Сформульовано як функціональні, так і нефункціональні вимоги до системи. Визначено ключові сценарії використання додатку, побудовано специфікацію вимог, що слугувала основою для подальшого проєктування.

Обґрунтовано вибір мови Kotlin, середовища Android Studio, архітектурної моделі MVVM, бібліотеки Retrofit для взаємодії з API, а також використано компоненти Android Jetpack. Обрані технології забезпечили ефективність, підтримуваність та масштабованість рішення.

На основі сформульованих вимог розроблено структуру додатку. Побудовано UML-діаграми: варіантів використання, класів, послідовності, розгортання, що дозволило формалізувати логіку взаємодії між компонентами системи.

Реалізовано основну функціональність мобільного додатку: введення даних, вибір валют, виконання запиту до OpenExchangeRates API, обробка результату та виведення його на екран. Забезпечено чітке розділення логіки в рамках MVVM.

Виконано функціональне тестування додатку з перевіркою різних сценаріїв: коректного та некоректного введення, відсутності інтернет-з’єднання, перевантаженого API тощо. Забезпечено стабільність роботи та перевірено зручність інтерфейсу відповідно до принципів UX/UI.

Створено технічну та користувацьку документацію, що включає опис архітектури, основних компонентів, інструкцію з використання додатку. Проведено UX/UI-аналіз, який підтвердив зручність взаємодії з інтерфейсом і зрозумілу навігацію між екранами.

У результаті виконання роботи досягнуто поставлену мету – створено простий, ефективний та функціональний мобільний додаток для конвертації валют, який може бути використаний широким колом користувачів. Застосовані підходи до модульної структури та документування дозволяють у подальшому легко адаптувати або розширювати систему, наприклад, шляхом додавання історії конвертацій або графіків змін курсів.

Таким чином, усі поставлені завдання виконані, функціональні можливості реалізовані, а мобільний додаток відповідає вимогам сучасної інженерії програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Listing Down 10 Currency Converter Apps for 2025. URL: https://www.tekrevol.com/blogs/currency-converter-apps/?utm_source=chatgpt.com (дата звернення 23.02.2025).

2. US Dollar to Euro Exchange Rate Chart. URL: https://www.xe.com/currencycharts/?from=USD&to=EUR (дата звернення 22.02.2025).

3. Currency Converter Plus. URL: https://play.google. com/store/apps/details?id=com.digitalchemy.currencyconverter&hl=en_US (дата звернення 23.02.2025).

4. Easy Currency Converter. https://www.amazon.com/ExtraAndroary-Easy-Currency-Converter/dp/B00CUGD4WS (дата звернення 23.02.2025).

5. Why Kotlin? Eight features that could convince Java developers to switch. URL: https://www.infoworld.com/article/3396141/why-kotlin-eight-features-that-could-convince-java-developers-to-switch.html (дата звернення 28.03.2025).

6. Activity Lifecycle in Android with Demo App. URL: https://www.geeksforgeeks.org/activity-lifecycle-in-android-with-demo-app/ (дата звернення 02.04.2025).

7. Android Developers Introduction to activities. URL: https://developer.android. com/guide/components/activities/intro-activities (дата звернення 01.03.25).

8. Android – Fragments. URL: https://www.tutorialspoint. com/android/android_fragments.htm (дата звернення 02.04.25).

9. Android View and ViewGroup. URL: https://medium.com/ @huseyinozkoc/android-view-and-viewgroup-6667a20724c5 (дата звернення 03.04.25).

10. Difference Between MVC, MVP and MVVM Architecture Pattern in Android. URL: https://www.geeksforgeeks.org/difference-between-mvc-mvp-and-mvvm-architecture-pattern-in-android/ (дата звернення 21.04.2025).

11. What is REST API? – A Comprehensive Guide To RESTful APIs. URL: https://revaalolabs.com/post/what-is-rest-api-a-comprehensive-guide-to-restful-apis (дата звернення 21.04.25).

12. Open Exchange Rates API. URL: https://docs. openexchangerates.org/reference/authentication (дата звернення 21.04.25).

13. What is Gradle? Why Do We Use Gradle? URL: https://www.simplilearn.com/tutorials/gradle-tutorial/what-is-gradle (дата звернення 28.04.2025).

14. Programming Languages for Mobile App Development. URL: https://buildfire.com/programming-languages-for-mobile-app-development/ (дата звернення 25.04.2025).

15. Top 5 Programming Languages & Frameworks for Mobile App Development. URL: https://www.linkedin.com/pulse/top-5-programming-languages-frameworks-mobile-app-development (дата звернення 26.04.2025).