Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка автоматизованої системи планування і обліку завдань. Частина 2. Розробка мобільного додатку.»

Виконав: студент 2 курсу,
групи 2АКІТ-22м
спеціальності 151 – Автоматизація та
комп'ютерно-інтегровані технології

Назарій БЕКАС .
Ім'я ПРІЗВИЩЕ

Керівник: к.т.н., доцент, доцент кафедри КСУ
степінь, звання, посада
Олег КОВАЛЮК
Ім'я ПРІЗВИЩЕ
«01» 12 2023 р.

Опонент: доцент кафедри АІІТ
степінь, звання, посада
Роман МАСЛІЙ
Ім'я ПРІЗВИЩЕ
«06» 12 2023 р.

Допущено до захисту
Т.в.о.Зав. кафедри КСУ
Марія ЮХИМЧУК
«07» 12 2023

Вінниця ВНТУ – 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління
Рівень вищої освіти другий (магістерський)
Галузь знань – 15 – Автоматизація та приладобудування
Спеціальність – 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітньо - професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ
Т.в.о.Зав. кафедри КСУ

Марія ЮХИМЧУК

"09" жовтня 2023 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Бекасу Назарію Аркадійовичу .
(прізвище, ім'я, по батькові)

1. Тема роботи. Розробка автоматизованої системи планування і обліку завдань. Частина 2. Розробка мобільного додатку
керівник роботи Ковалюк Олег Олександрович
затверджені наказом ВНТУ від "18" вересня 2023 року №247
2. Термін подання студентом роботи "1" грудня 2023 року
3. Вихідні дані до роботи: можливість використання мобільного додатку на різних пристроях андроїд; база даних спроектована спеціально під мобільний додаток; можливість записувати, редагувати, та видаляти задачі до моменту виконання
4. Зміст текстової частини: вступ, аналіз предметної області та огляд аналогів, огляд структури технологій та функціональності мобільного додатку, розробка та тестування мобільного додатку
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): об'єкт, предмет та мета дослідження, задачі дослідження, блок-схема порядку дій, основні системи управління і обліку завдань, середовище розробки та тестування Android Studio, зовнішній вигляд мобільного додатку, тестування програми, економічна частина, висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
4	Бурєннікова Н. В., д.е.н., професор кафедри ЕПВМ.		

7. Дата видачі завдання "09" жовтня 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		При
		початок	закінчення	
1	Аналіз мов програмування для розробки мобільних додатків	10.10. 2023	11.10.2023	
2	Аналіз мов програмування для розробки бази даних для мобільного додатку	12.10. 2023	14.10.2023	
3	Огляд та аналіз аналогів у сфері розподілу та обліку завдань	15.10. 2023	24.10. 2023	
4	Розробка дизайну інтерфейсу додатку, та огляд	25.10. 2023	29.10.2023	
5	Розробка і тестування програмного забезпечення та баз	30.10.2023	04.11.2023	
6	Розрахунок економічної частини	05.11.2023	20.11.2023	
7	Графічні матеріали	21.11.2023	27.11.2023	

Студент (підпис) Назарій БЕ
(Ім'я ПРІЗВИ

Керівник роботи (підпис) Олег КОВАЛ
(Ім'я ПРІЗВИ

АНОТАЦІЯ

УДК 004.9:621

Бекас Н.А. Розробка автоматизованої системи планування і обліку завдань. Частина 2 Розробка мобільного додатку. Магістерська кваліфікаційна робота зі спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології, освітня програма – Інтелектуальні комп'ютерні системи. Вінниця: ВНТУ, 2023. 108с.Українською мовою,4 розділи, 50 джерел.

У магістерській кваліфікаційній роботі розроблено автоматизовану систему планування та обліку завдань. Розроблена система базується на платформі Android і включає в себе функціонал, спрямований на інтелектуальне планування, динамічну підтримку завдань, аналіз продуктивності та високий рівень користувацької зручності. Додаток дозволяє користувачам ефективно організовувати свій робочий час, ставлячи та відстежуючи завдання, а також отримувати інтелектуальні рекомендації щодо пріоритетів та термінів виконання.

Магістерська робота включає в себе детальний аналіз впливу системи на організацію робочого процесу та експериментальні дослідження з метою підтвердження ефективності розробленої системи. Висновки роботи можуть слугувати основою для подальших досліджень у сфері розробки мобільних додатків для планування завдань.

Розроблено додаток за допомогою мови програмування Kotlin, дані вписані в якому зберігаються в базі даних побудованій на основі Room.

Ключові слова: автоматизована система, управління, планування, мобільний додаток, Kotlin.

ANNOTATION

UDC 004.9:621

Bekas N.A. Development of an Automated Task Planning and Tracking System. Part 2: Development of a Mobile Application. Master's Qualification Work in the specialty 151 – Automation and Computer-Integrated Technologies, Educational Program – Intelligent Computer Systems. Vinnytsia: VNTU, 2023. p.

In Ukrainian. Bibliography: 46 titles; figures: 27; tables: 14.

In the master's qualification work, an automated task planning and tracking system has been developed. The system is built on the Android platform and incorporates functionality focused on intelligent planning, dynamic task support, performance analysis, and a high level of user convenience. The application enables users to efficiently organize their work time by setting and tracking tasks, as well as receiving intelligent recommendations regarding priorities and deadlines.

The master's thesis includes a detailed analysis of the system's impact on the organization of the work process and experimental research aimed at confirming the effectiveness of the developed system. The conclusions of the work can serve as a basis for further research in the field of mobile application development for task planning.

The application was developed using the Kotlin programming language, and the data entered into it are stored in a database built on Room.

Keywords: automated system, management, planning, mobile application, Kotlin.

ЗМІСТ

ВСТУП

Актуальність. Сучасний світ, насичений технологічними та вимагає високої продуктивності та ефективності, надає особливого значення системам автоматизованого планування та обліку завдань[1]. В умовах швидкої зміни інформаційного оточення та завдань, стоять перед бізнесом, науково-дослідними установами та іншими організаціями складні завдання з планування та ефективного розподілу ресурсів для досягнення поставлених цілей.

Дана робота спрямована на розробку автоматизованої системи планування і обліку завдань з метою підвищення продуктивності та ефективності управління завданнями людей в організаціях і поза їх межами. Обрана тема є актуальною та стрімко розвивається в контексті сучасного бізнесу[2] та наукової сфери, адже автоматизовані системи можуть значно полегшити процеси планування, аналізу та моніторингу завдань.

Головним рушієм для створення даної роботи було підвищення динаміки інформаціного поля людей, та збільшення навантаження на кожну персону в умовах повсякденного життя. Всі ці фактори збільшують розсіяність уваги конкретної людини і зменшують продуктивність протягом дня, що може привести до факту невиконання людиною поставлених перед нею завдань на роботі, навчанні чи будь-де в іншому середовищі. Для зведення вищевказаних негативних подій в житті до мінімуму було прийняте рішення розробити автоматизовану систему планування і обліку завдань.

Об’єктом дослідження є процеси планування та обліку завдань в організаціях, а також інформаційна система, призначена для автоматизації цих процесів. Цей об'єкт охоплює всі аспекти, пов'язані з плануванням завдань, їхнім розподілом, відстеженням виконання та обліком результатів в рамках роботи організації.

Предметом дослідження є автоматизація планування та обліку завдань в сучасних організаціях, а також методи та засоби їх автоматизації з метою підвищення ефективності та організації робочого процесу.

Метою дослідження є поліпшення ефективності та продуктивності

управління завданнями в організаціях або окремо взятих людей в повсякденному житті. Для цього було вирішено розробити та реалізувати автоматизовану систему планування і обліку завдань. Конкретні цілі дослідження включають:

- Реалізацію автоматизованої системи, що дозволить користувачам створювати, редагувати, відстежувати та аналізувати завдання.
- Проведення експериментальних випробувань та оцінку ефективності системи на реальних об'єктах дослідження.

Новизна роботи може бути визначена таким чином:

Розроблено інноваційний планувальник завдань, який відзначається зручним інтерфейсом для користувачів, гнучкістю та адаптивністю до різних галузей, а також використанням передових технологій, сприяючи вирішенню актуальних завдань управління завданнями в організаціях.

Практичною цінністю є розробка ефективних алгоритмів та програмного забезпечення для організації та обліку власних завдань і планів у мобільному додатку, що сприяє зручному упорядкуванню та надає можливість підбити підсумки успішності виконання завдань.

Апробація. Представлені в роботі результати апробовані в результаті участі в конференції «LIII Всеукраїнська науково-технічна конференція підрозділів Вінницького національного технічного університету НТКП-ВНТУ»:

Публікації: Бекас Н.А., Ковалюк О.О. «Короткий огляд розробки автоматизованої системи планування і обліку завдань», «LIII Всеукраїнська науково-технічна конференція підрозділів Вінницького національного технічного університету НТКП-ВНТУ(2024)».

1 АНАЛІЗ СИСТЕМ ПЛАНУВАННЯ ТА ОБЛІКУ ЗАВДАНЬ. ОГЛЯД АНАЛОГІВ

1.1 Аналіз проблеми автоматизованої системи планування та обліку завдань

Процес еволюції цифрової розробки є своєрідним літописом технічних досягнень і розвитку інформаційної сфери. Починаючи з періоду, коли пошук інформації обмежувався виключно ручним пошуком у друкованих виданнях і бібліотечних каталогах, до сьогоднішнього дня, коли на сцені з’явилися комп’ютери[4], Інтернет[5] і смартфони[6], цей процес пройшов довгий шлях.

З початком ери Інтернету відбулася революція пошуку в Інтернеті, коли такі гіганти, як Google[7], щодня стають незамінними інструментами для мільйонів користувачів. Ці пошукові системи використовують високорозвинені алгоритми для індексування мільярдів веб-сторінок і дозволяють користувачам швидко знаходити потрібну інформацію за допомогою ключових слів.

Також активно розвивається розробка мобільних пристроїв, а саме смартфонів. Так як практично у кожної людини в світі є свій смартфон, програмне забезпечення для телефону стає все більш необхідним. У повсякденному житті обсяг інформації настільки величезний, що часто змінює пріоритети людей і зміщує фокус уваги, через що можна втратити зв'язок з минулими планами і не виконати їх. Саме тому був винайдений тип додатків-планувальників, які допомагають записувати свої плани, цілі та завдання, а також контролювати їх виконання. Перший мобільний додаток-планувальник для запису планів і завдань був створений у 1990-х роках. Одним із перших популярних додатків такого типу був "DateBk" для Palm OS[8], який вийшов у 1997 році. Цей додаток дозволяв користувачам створювати події та завдання в календарі, встановлювати нагадування та відстежувати їх виконання.

Пізніше, з розвитком смартфонів і мобільних операційних систем, таких як

iOS[9] і Android[10], з’явилося багато інших програм для планування завдань і запису, які стали надзвичайно популярними серед користувачів. Отже, початок мобільних додатків для планування та завдань можна побачити приблизно в 1990-х роках, і вони продовжують рости та вдосконалюватися донині.

Детальний аналіз проблеми розробки автоматизованої системи планування та обліку завдань передбачає врахування багатьох факторів і проблем, які можуть виникнути під час цього процесу. Основні виклики в цьому контексті включають наступне:

- Складність системи: Розробка та впровадження автоматизованої системи планування та обліку є складним процесом. Це вимагає багато технічних знань у різних галузях, які повинні вмістити 1-2 людини, які мають розробити та протестувати додаток.

- Сумісність з існуючими операційними системами[11]: Оскільки додаток розроблено на базі системи Android, його неможливо буде встановити на ОС IOS, не кажучи вже про те, щоб його запустити. Це створює дилему, а саме потребу в розробці іншої подібної програми для іншої ОС. Тому в майбутньому це викличе необхідність додаткового навчання кваліфікованого розробника. Необхідно буде розробити та провести низку тестів для одного продукту, щоб розширити потенційних користувачів.

- Безпека та конфіденційність даних: збір і обробка інформації про користувачів, їхні завдання та плани призводить до накопичення цієї інформації в хмарі. Важливо вжити заходів для забезпечення захисту цих даних від несанкціонованого доступу.

- Очікування користувачів: користувачі можуть мати високі очікування щодо якості продукту та різноманітних функцій програми. Невиконання цих очікувань може викликати негативну реакцію.

- Технічні проблеми та збої: такі програми можуть зіткнутися з технічними проблемами, які можуть вплинути на нормальну роботу програми та облік завдань, можуть не зараховувати виконані завдання та втрачати дані про завдання.

1.2 Аналіз предметної області розробки мобільних додатків

В сучасності важко уявити життєдіяльність без сучасних гаджетів. Хоча не у всіх є комп'ютерна техніка, а саме персональний комп'ютер, практично кожна особа має мобільний телефон, зокрема смартфон. Однією з основних переваг смартфона є його компактність та універсальність, оскільки його можна використовувати у будь-якому місці для виконання різноманітних завдань та отримання результатів. Важливою характеристикою також є те, що смартфон, крім базових функцій, таких як дзвінки та повідомлення, здатний виконувати безліч інших операцій.

Ці різноманітні операції включають в себе пошук інформації в Інтернеті, навігацію, виконує функцію нотатника, взаємодію в соціальних мережах та керування різними системами. Використання всіх цих функцій стало можливим завдяки розвитку комп'ютерної техніки, мікропроцесорів та програмування.

Створення додатків для мобільних пристроїв стало популярним зі збільшенням потреб людини, та щороку набувало все більшої актуальності, оскільки кількість таких пристроїв зростає щодня разом з потребою людей в них. Усі мобільні пристрої можна розділити на два основних типи за операційною системою: Android та iOS.

Кожна особа вибирає собі смартфон відповідно власних потреб та матеріального статусу, при такій різноманітності потреб і різних цінових категорій виробники змогли підлаштуватися під кожну з них. Проте варто розуміти, що будь-яка техніка має чіткі характеристики, котрі роблять порявняння видів і аналогів технологій максимально простим навіть для пересічної людини.

Операційні системи смартфонів відрізняються одна від одної більш ніж суттєво, тому варто виділити переваги і недоліки кожної з них.

Переваги IOS:

- Висока якість додатків та магазину додатків;
- Швидке та довгострокове оновлення операційної системи;

- Надійність екосистеми ОС IOS;
- Можливість об’єднати роботу мобільних пристроїв оновлених до останньої версії ОС(Наприклад: відповідати на дзвінки IPhone за допомогою IPad).

Недоліки IOS:

- Замкненість системи;
- Обмеження в кастомізації;
- Висока вартість пристроїв;

Після перегляду переваг і недоліків ОС IOS необхідно звернути увагу на ті ж критерії в ОС Android.

Переваги Android:

- Сумісність практично з усіма марками планшетів;
- Широкий ціновий діапазон і великий вибір продукції;
- Динамічний обсяг пам'яті (Є можливість збільшити постійну пам'ять за допомогою карти пам'яті)
- Кастомізація інтерфейсу;
- Кількість додатків в магазині та зручність браузера.

Недоліки Android:

- Величезна кількість передвстановлених додатків на пристроях;
- Велика залежність від сервісів Google;

На кожну з вищевказаних операційних систем істує безліч додатків, які призначені для різноманітних цілей. Дані додатки розробляються як поодинокими розробниками, групами ентузіастів, так і комерційними компаніями різної величини. Проте цих людей об’єднує одне – знання мов програмування необхідних для розробки додатків. Для кожної ОС є свої мови зі своїми особливостями, синтаксисом і функціоналом.

Лідером по вибору серед розробників багато років є мова програмування Java[12]. Дана мова є настільки широко вживаною через свою об’єктну орієнтованість, є другою найактивнішою мовою на сервісі GitHub[13] і може

працювати на абсолютно всіх платформах, що підтримують Java, без необхідності перекомпіляції. Всі основні переваги та недоліки вказаної мови вказані далі по тексту.

Переваги:

- Низький поріг для вивчення;
- Немає прив’язаності до платформи;
- Має масштабну базову бібліотеку API[14];
- Великий обсяг інструментів для побудови Android додатків;
- Має величезну кількість стандартних бібліотек Java;
- Наявна велика опен-сорс платформа;
- Широка і відкрита спільнота, для допомоги розробникам;
- Додатки побудовані на ній більш легкі і компактні на пристроях;
- Висока швидкість розробки.

Недоліки

- Потребує багато пам’яті під час розробки;
- Необхідно багато коду написати, збільшується ризик виникнення помилок;
- Повільна мова в порівнянні з іншими.

Другою по популярності використання розробників додатків для ОС Android є Kotlin[15]. Мова програмування Kotlin виникла у результаті розробки командою JetBrains, яка спеціалізується на розробці програмного забезпечення. Створення мови Kotlin ставило за мету вирішити конкретні недоліки, що існували у мові програмування Java, такі як відсутність системи нульової безпеки, повільна компіляція та інші обмеження. Зокрема, Kotlin прагнула забезпечити високу продуктивність та сумісність з Java, одночасно надаючи розробникам сучасні можливості програмування, такі як система нульової безпеки, розширення функціональності та конденсовані конструкції коду. Застосування мови Kotlin охоплює широкий спектр варіантів, включаючи розробку мобільних додатків для платформи Android, серверних застосунків та веб-розробку. Переваги і недоліки даної мови розписані нижче.

Переваги:

- Легкість вивчення людям, що працювали з Java;
- Значно більш лаконічна мова ніж Java;
- Широкий список інструментів для роботи з дизайном;
- Можна використовувати в бекенд проєктах, таких як Spring 5;
- Можна взаємодіяти з Java, а також повна сумісність з усіма бібліотеками і середовищами Java та JVM.

Недоліки:

- Більша складність обчислювального процесу ніж в Java;
- Велика складність в пошуках розробника, через молодість мови.

Для розробки ж на ОС IOS існують інші мови програмування, наприклад Objective-C[16] та Swift[17]. Мова програмування Objective-C була розроблена на початку 1980-х років та використовується в основному для розробки програмного забезпечення для платформи Apple[18], зокрема для операційної системи macOS та IOS. Вона стала основною мовою для розробки програм для iPhone та Mac. Objective-C була обрана компанією Apple як основна мова для розробки програмного забезпечення для своїх пристроїв протягом багатьох років. Завдяки своїй довгій історії та популярності, вона визначила стандарти для розробки на платформі Apple до випуску Swift. Переваги і недоліки даної мови – нижче.

Переваги:

- Широке застосування для розробки програм для IOS, macOS, watchOS та tvOS;
- Простота розуміння написання.

Недоліки:

- Синтаксис може вважатися менш зрозумілим і складнішим для читання порівняно з сучаснішими мовами;
- Відсутність деяких сучасних функцій забезпечення безпеки коду;
- Обмеженість функціоналу.

Другою ж по частоті використання серед розробників для IOS є Swift. Це

мова програмування, розроблена компанією Apple та представлена в 2014 році. Ця мова створювалася з метою заміни Objective-C та поліпшення ефективності та безпеки розробки програм для платформи Apple. Swift є потужною мовою, яка широко використовується для розробки мобільних додатків для iOS, macOS, watchOS та tvOS.

Переваги:

- Має вбудовану систему нульової безпеки та інші функції, спрямовані на запобігання помилкам.
- Простий та лаконічний синтаксис полегшує читання та написання коду.
- Пропонує сучасні парадигми програмування, що полегшують та прискорюють розробку.

Недоліки:

- Основна мова для розробки під платформу Apple, що обмежує переносимість коду на інші платформи.
- Перехід від Objective-C до Swift може вимагати часу та зусиль великих проектів, розроблених на старіших мовах.

Objective-C та Swift, обидві спрямовані на розробку програмного забезпечення для платформи Apple, і незважаючи на те, що Swift вже здобуває перевагу в багатьох проектах, Objective-C продовжує підтримуватися та застосовуватися в конкретних сценаріях. Вибір конкретної мови часто обумовлюється історією проекту, потребами щодо швидкості розробки та іншими чинниками, що визначають вибір мови програмування.

Хоч і в сучасності можливо розробити програмне забезпечення, що має працюючий і однаковий код на обох ОС, проте в даній роботі було зроблено вибір на користь саме операційної системи Android, так як функціонал потребує проектування бази даних.

Існує достатньо велика кількість різних баз даних, але не всі з них підходять для розробки саме мобільних додатків. Найчастіше Android-розробники використовують SQLite[19], так як це БД з відкритим вихідним

кодом, яка підтримує базові можливості SQL[20], а саме:

- Синтаксис.
- Транзакція.
- Тригери.
- Параметризовані оператори.
- Агрегатні функції.

Проте під час використання SQLite, доступ до файлової системи може бути обмеженим, що може призвести до значного зниження швидкості операцій. Отже, для вирішення цієї проблеми рекомендується виконувати всі операції з базою даних у фоновому потоці, уникати виконання їх у головному потоці (UI-потоці). Що в свою чергу продукую ряд обмежень в порівнянні з іншими базами даних. Наприклад:

- Підтримувані типи даних. У SQLite їх 4, тоді як у MySQL[21] і PostgreSQL[22] їх понад 20;
- SQLite не є оптимальним вибором, якщо ви плануєте працювати над додатком, до бази даних якого одночасно звертається кілька користувачів. У цьому випадку, наприклад, раціональніше вибрати повнофункціональну Систему Керування Базами Даних (РСУБД) MySQL;
- У SQLite існують недоліки з операціями запису, оскільки ця база даних дозволяє виконувати лише одну операцію запису одночасно;
- Відсутність користувацького керування, тобто відсутність можливості керувати зв'язками в таблицях відповідно до привілеїв;
- Відсутність можливості підвищення продуктивності бази даних за рахунок внутрішніх налаштувань.

Саме тому, що вищевказана БД має такі обмеження, було обрано використовувати для роботи з розробкою додатку бібліотеку Room. Компоненти архітектури Android представляють собою набір бібліотек, що сприяють створенню надійних, тестируваних та обслуговуваних додатків. У цьому контексті, Room входить до складу зазначених компонентів і дозволяє спростити взаємодію з об'єктами SQLiteDatabase у додатку, зменшуючи обсяг

стандартного коду і перевіряючи SQL-запити під час компіляції. Room[23] складається з трьох основних компонентів:

- Entity — це об'єкт таблиці бази даних.
- Dao — надає методи, які додаток може використовувати для взаємодії з базою даних.
- Database — містить базу даних і є основною точкою доступу для базового з'єднання з постійними даними додатка.

Наприклад, деякі методи, які є в DAO:

- Query — запит для отримання, оновлення та видалення даних за певною умовою.
- Insert — додавання об'єкта в базу даних.
- Delete — видаляє об'єкти.
- Update — оновлює об'єкти.

Метод Query працює асинхронно, в той час як інші три є синхронними. Database надає додатку екземпляри DAO, пов'язаних з цією базою даних. З свого боку, додаток може використовувати DAO для отримання даних з бази даних у вигляді екземплярів пов'язаних об'єктів сутностей даних. Додаток також може використовувати конкретні об'єкти даних для оновлення рядків у відповідних таблицях або для створення нових рядків для вставки. На наведеному нижче малюнку показано взаємозв'язок між різними компонентами Room.(Рисунок 1.1)

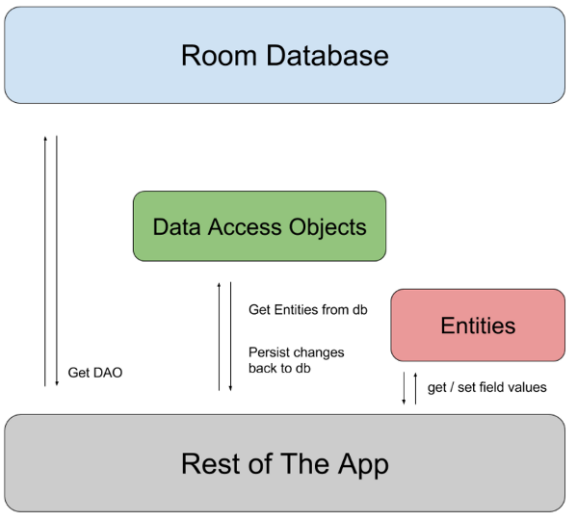


Рисунок 1.1 – Принцип роботи БД Room

Отже зі всього, що було перераховано в підрозділі можна зрозуміти, що робота над мобільними додатками достатньо об’ємна і працезатратна, тому просто необхідно врахувати плюси і мінуси всіх інструментів, що будуть використовуватися і потрібно чітко розуміти які саме необхідно виконувати функції майбутній програмі задля коректного вибору.

1.3 Аналіз існуючих автоматизованих систем планування у сфері мобільних додатків

Сучасні підходи до автоматизації планування і облікузавдань можуть включати комбінацію різних технологій та методів виконання з метою досягнення максимальної ефективності та зручності для користувачів. Вибір оптимального методу залежить від конкретних потреб. Наприклад, сучасні методи можуть включати в себе використання систем штучного інтелекту для оптимізації порядку виконання завдань, використання аналітики даних для планування та прогнозування в майбутньому, технології сучасних комп’ютерів. Вибір конкретного методу повинен ґрунтуватися на специфічних вимогах та призначенні мобільного додатку, який полягає у покращенні ефективності людини протягом дня, або компанії у виконанні певних завдань.

1.3.1 Аналіз історичного розвитку планувального спорядження

Аналіз історії та еволюції способів запису планів та їх відслідковування, починаючи від античних календарів і до сучасних мобільних додатків, допоможе нам краще зрозуміти, як сучасні мобільні програми для планування розвивалися від своїх попередників.

1. Античні календарі: Історія планування справ на календарях має тисячолітню

історію. Вже в античних цивілізаціях, таких як давній Рим та Месопотамія, використовувалися календарі для запису подій та важливих дат. Це були перші спроби систематизувати час та події.

2. Паперові планери: У середньовіччі паперові планери стали популярними серед осіб, які бажали організовувати свій час та плани. Вони включали календарні сторінки, списки завдань і місце для особистих записів.

3. Електронні планери: З появою персональних комп'ютерів у 1980-х роках, електронні планери стали популярними серед бізнесменів та професіоналів. Програми, такі як Microsoft Outlook [24], надали можливість вести електронний календар, завдання та нагадування.

4. Сучасні мобільні додатки: З розвитком смартфонів та мобільних операційних систем, таких як iOS та Android, виникла нова ера планування і відслідковування завдань. Сучасні мобільні додатки, такі як Todoist [25], Wunderlist (тепер Microsoft To-Do)[26], Trello[27], і Google Keep[28], надають користувачам можливість легко створювати, відстежувати та оновлювати завдання з будь-якого місця та в будь-який час. Вони також можуть синхронізуватися з іншими пристроями та надавати сповіщення.

Кореляція між історією та сучасними мобільними додатками полягає в тому, що обидва створені з метою полегшити життя користувачам та допомогти їм краще організовувати свій час та завдання. Спочатку це були фізичні засоби, такі як календарі та паперові планери, а тепер - мобільні додатки, які відкривають безмежні можливості для зручного планування та відстежування завдань. Сучасні мобільні додатки використовують передові технології, такі як синхронізація в хмарі, сповіщення та інтеграція з іншими сервісами для максимальної ефективності та зручності користувачів.

Існують різні додатки для планування і обліку завдань на різні операційні системи та навіть платформи. Нижче вказані різні приклади існуючих додатків на різні операційні системи і різні платформи але, які надають приблизно однакові можливості своїм користувачам та мають схожий функціонал :

Додатки для Android:

Todoist[https://todoist.com]: Особливості: даної програми - є однією з найпопулярніших програм для планування. Вона має простий та інтуїтивний інтерфейс, можливість створювати завдання, надавати їм пріоритети та терміни виконання, а також спільно працювати над завданнями з іншими користувачами.

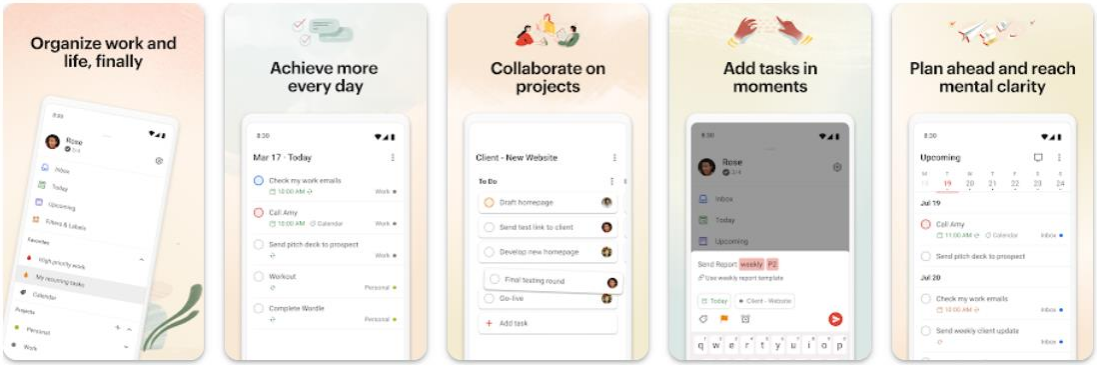


Рисунок 1.2 – Мобільний додаток “ Todoist ”

Google Keep[https://keep.google.com]: Особливості - це легкий та зручний додаток для створення нотаток та списків. Він інтегрований з Google-акаунтом, що дозволяє синхронізувати записи між різними пристроями. Крім текстових нотаток, він також підтримує малювання, голосові записи і фотографії.

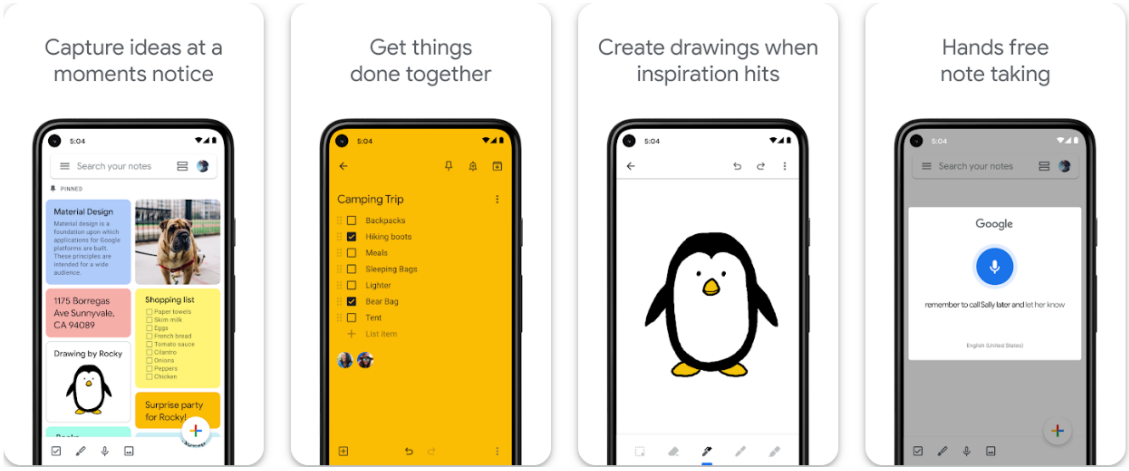


Рисунок 1.3 – Мобільний додаток “ Google Keep ”

Додатки для iOS:

Apple Reminders [https://apps.apple.com/us/app/reminders/id1108187841] [29]: Особливості: - Це вбудований додаток на пристроях Apple. Він дозволяє створювати завдання та списки завдань, призначати їм дедлайни та нагадування. Ще однією важливою особливістю є інтеграція з Siri, що дозволяє додавати завдання голосовими командами. Додаток також синхронізується з iCloud [30], що дозволяє користувачам доступатися до своїх завдань на різних пристроях Apple.

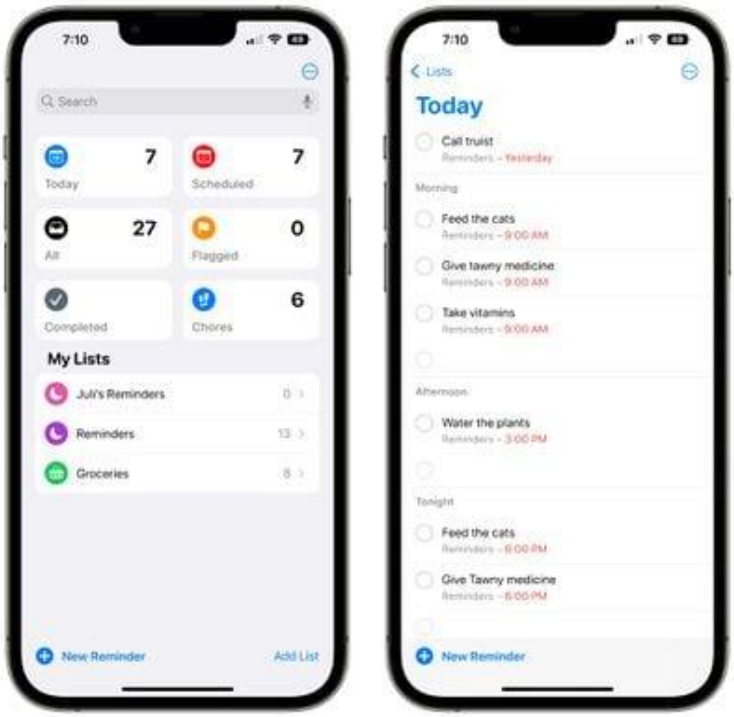


Рисунок 1.4 – мобільний додаток “ Apple Reminders ”

Things [https://apps.apple.com/ru/app/things-3/id904237743] [31]: Особливості - це потужний інструмент для планування та організації завдань. Він надає можливість створювати проекти, групувати завдання за категоріями та встановлювати дедлайни. Додаток також має нагадування та інтеграцію з календарем, що допомагає користувачам краще організовувати свій час.

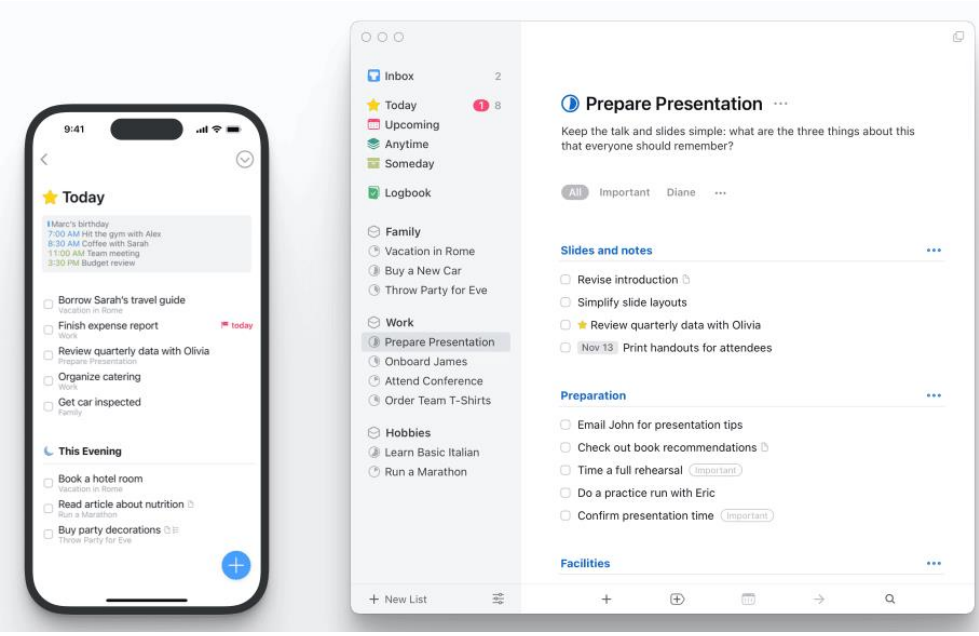


Рисунок 1.5 – Мобільний додаток “ Things ”

Ці приклади додатків на різні ОС та платформи показують різноманіття можливих дизайнів інтерфейсів та інструментів для виконання схожих функцій. Однак успішна розробка та впровадження додатку для використання великою кількістю людей та широким спектром галузей, в яких буде використано вимагає більш широкого розуміння поняття «розробка» та урахування низки ключових аспектів:

1.3.2 – Перелік важливих аспектів врахування під час розробки мобільного додатку планувальника

Почати потрібно з дослідження розуміння потреб і вимог цільової аудиторії. Які завдання вона буде планувати та відстежувати? Які функції будуть для аудиторії корисними? Використати опитування, інтерв'ю та аналіз конкурентів для збору цінної інформації. Дизайн інтерфейсу користувача (UI/UX) [32] є обличчям додатку, тому необхідно розробити інтуїтивно зрозумілий і привабливий інтерфейс користувача. Добре продуманий UX забезпечує легкість використання та задоволення від користування додатком. Важливо враховувати розмір та форму

екрану для мобільних пристроїв. Надалі потрібно визначити, які конкретні функції буде надавати додаток. Це може включати створення списків завдань, налаштування нагадувань, імпорт/експорт даних, синхронізацію з хмаровими службами, можливість встановлення пріоритетів, фільтрацію завдань та інші. Якщо планувальник має функціональність календаря, враховувати можливість створення подій, перегляд подій за днями, тижнями, місяцями, інтеграцію з іншими календарями та нагадування про події. Під час розробки необхідно забезпечити можливість синхронізації даних між різними пристроями і платформами (iOS, Android, веб-версія). Використовувати хмарові служби для збереження і синхронізації даних. Важливим аспектом розробки є забезпечення захисту даних користувачів від несанкціонованого доступу і втрати. Використовувати шифрування та механізми аутентифікації для забезпечення безпеки.

Дбати про продуктивність додатка, щоб він працював швидко та ефективно навіть на пристроях з обмеженими ресурсами.8. Тестування і відлагодження: Провести ретельне тестування додатка, виявляючи та виправляючи помилки і недоліки. Тестування на різних пристроях і платформах є важливим етапом після якого потрібно розробити план запуску, включаючи стратегію маркетингу і просування додатка. Після запуску збирати відгуки від користувачів і вносити поліпшення. На сам кінець кожен розробник повинен постійно вдосконалювати і підтримувати додаток, випускаючи оновлення та реагуючи на відгуки користувачів.

Розробка мобільного додатка планувальника - це важливий і деталізований процес, який вимагає уваги до дрібниць і постійного вдосконалення, щоб створити корисний та популярний інструмент для користувачів. Урахування цих факторів допомагає забезпечити, що система буде працювати справно, не гірше конкурентів утна ринку, а також зможе задовольняти всі потреби користувачів.

1.3.3 – Вибір API для мобільної розробки та їх види

Нещодавно значна частина програм була автономною, здатною самостійно вирішувати завдання на обчислювальних ресурсах конкретної системи. Однак зі значущим поширенням онлайн-технологій відбулося кардинальне змінення: більшість веб-сайтів та мобільних додатків тепер оперують на зовнішніх серверах та хмарних платформах. Це призвело до збільшення гнучкості в розробці програмного забезпечення. Сучасним фахівцям необов'язково створювати функціонал "з нуля". Достатньо інтегрувати відповідний пакет для реалізації вже розробленого функціоналу, починаючи від віджетів з прогнозом погоди і закінчуючи засобами кібербезпеки та алгоритмами машинного навчання.

Потенціал використання API в мобільних додатках величезний і корисний для різних бізнес-сценаріїв. Ряд функцій, які можна реалізувати у додатку через зовнішні інтерфейси, включає:

- Авторизація через обліковий запис у соц-мережах або обліковий запис Google.
- Створення різноманітних клієнтських онлайн-форм (запис приймання, заявка на консультацію тощо).
- Використання конвертерів валют та фінансових калькуляторів.
- Впровадження чат-ботів та засобів зв'язку клієнта із службою підтримки.
- Інтеграція онлайн-карток та використання електронних гаманців.
- Мобільний банкінг.
- Застосування засобів ІІ (розпізнавання голосу, наприклад).

Призначення API для розробки додатків стає не тільки зручним інструментом для розробників, але і окремим бізнес-напрямком. Засоби API дозволяють бізнесу ефективно автоматизувати рутинні процеси, перетворювати застарілі веб-служби та здійснювати швидке впровадження нових технологій з мінімальними затратами часу та ресурсів.

Застосування API визначається конкретними вимогами та умовами конкретного бізнесу. В даний час існують надійні програмні інтерфейси для

різноманітних сфер, таких як доставка даних, аналітика, фінанси тощо.

Для наглядності розглянуто кілька прикладів використання API в мобільних додатках. API соцмережі Twitter[33] надає інформацію про твіти та фоловерів користувача, в той час як API Youtube дозволяє використовувати різноманітні функції, такі як пошук та відтворення контенту. Google та Apple мають API для інструментів машинного навчання, а API Facebook дозволяє авторизацію через соцмережу. До того ж, космічне агентство NASA має свій API, який можна використовувати для оновлення бази даних про космос та отримання свіжих фотографій з Марса.

Застосування API в мобільних додатках є необмеженим, і ці інтерфейси розглядаються як повноцінний продукт для розробників. Процес створення API аналогічний розробці програмного забезпечення, включає в себе етапи тестування, підтримки користувачів та контролю версій.

Використання API у розробці мобільних додатків реалізує чітке відокремлення додатка від джерела інформації та дозволяє легко впроваджувати нові функції сервісної архітектури.

Застосування API відкриває для бізнесу можливість отримання інструментів, розробка яких не вимагає значних ресурсів або експертизи. З іншого боку, це також може використовуватися для перевірки практичних гіпотез чи ідей сервісу, не залучаючи значних зусиль до власної розробки.

Аналіз плюсів та мінусів використання API:

Позитивні аспекти:

- Використання сторонніх API дозволяє розробникам уникнути деталей низького рівня та розглядати прості завдання, прискорюючи роботу над проектом.
- Використання програмних інтерфейсів допомагає знизити ризики уразливостей у софті, надаючи стандартні методи взаємодії.
- API забезпечують структурований формат для взаємодії між програмами, що спрощує впровадження нових сервісів.
- Використання сторонніх API дозволяє уникнути розробки функціоналу,

який вже реалізовано в інших проектах, що призводить до економії ресурсів.

Негативні аспекти:

- Кожен API має свою власну унікальну структуру, що гальмує процес створення єдиної стандартизації програмних інтерфейсів.
- Розробка API не завжди підпорядкована стандартам або сертифікації, що може впливати на загальну якість виконання.
- Використання API вносить залежність від продукту інших розробників, відкриваючи можливість для раптових змін умов використання чи навіть закриття ресурсу.

Класифікація видів API:

- RPC (Remote Procedure Call):[34] Інтерфейси, що базуються на викликах функцій у віддалених системах.
- SOAP (Simple Object Access Protocol):[35] Інтерфейси, побудовані на протоколі обміну інформацією у розподіленому обчислювальному середовищі.
- REST (Representational State Transfer):[36] Інтерфейси, що базуються на архітектурі для розподілених систем.
- GraphQL:[37] Агностик мови опису запитів даних, що дозволяє клієнту взаємодіяти з різними вузлами у межах одного запиту.

Використання стилів REST та GraphQL у розробці мобільних додатків найчастіше виявляється дієвим, оскільки REST простий, безпечний та не скований протоколами, забезпечуючи гнучкість у виборі мов для розробки, таких як PHP[38], Python[39], Ruby[40] та інші.

Усі ці аспекти роблять розробку API важливою задачею в сучасній мобільній розробці, проте існують обмеження, які ускладнюють їхнє подальше розгортання та розвиток.

Вивчення концепції функцій програмування є важливим для усвідомлення принципів роботи API. Функція, як логічний блок, обробляє вхідні дані,

виконує операції і повертає необхідну інформацію. API, у свою чергу, можна розглядати як набір таких функцій для забезпечення зв'язку між платформами та стандартизації передачі даних.

Внутрішній пристрій та функції API можуть варіюватися залежно від вибору розробників. Це може бути одне спільне API зі стандартними функціями або набір окремих інтерфейсів, організованих за функціональністю чи іншими критеріями. Розробники визначають формат передачі, параметри функцій та інші аспекти для додавання підтримки API.

Компанії, такі як Google та Apple, надають внутрішні API для мобільних платформ. Наприклад, для iOS розробники використовують CoreML[41] для машинного навчання, Apple Maps для карт, RealityKit[42] для доповненої реальності, ApplePay для електронного гаманця, та HealthKit[43] для роботи з даними здоров'я. Для Android використовують NNAPI[44] для роботи з нейромережами, MLKit для машинного навчання, та інші.

У розробці мобільних додатків також використовують універсальні інструменти, такі як Firebase для зберігання та синхронізації даних, Flurry[45] та Crashlitics для аналітики та звітів про збої, GoogleADS для реклами.

Створення мобільного додатку часто включає в себе використання одного чи кількох API, і вибір платформи розробки важливий для швидкої та ефективної роботи інтерфейсів. Важливо заздалегідь продумати структуру програми для майбутньої інтеграції API та обговорити розробку з експертами для уникнення можливих проблем у майбутньому.

Використання API у мобільній розробці є невід'ємною частиною створення функціональних та ефективних додатків. Бізнес повинен бути усвідомлений технічних відтінків і співпрацювати з розробниками для досягнення успішного результату.

Для даної роботи було зібрано всю інформацію про актуальні версії Android та версії API, що відповідають їм. Розробку велося на пристрій Pixel 3A версія Android 14.0, актуальна версія API 34.

1.4 Висновки

Висновки до першого розділу дослідження покликані узагальнити отриману інформацію та виділити ключові аспекти, що були розглянуті.

Актуальність роботи підтверджується широким розповсюдженням та популярністю мобільних додатків для планування справ у сучасному світі. Швидкий темп життя, велика кількість завдань та високі вимоги до продуктивності вимагають ефективного інструменту для організації робочого та особистого часу. Розробка такого додатку на мові програмування Kotlin для платформи Android є важливою та відповідальною задачею.

Аналіз існуючих аналогів показав, що на ринку існують різноманітні додатки для планування справ, проте вони мають свої обмеження та недоліки. Багато з них не надають достатньо широкий функціонал, не забезпечують ефективну інтеграцію з іншими сервісами, або мають складний та неінтуїтивний інтерфейс.

Теоретичний огляд розробки мобільних додатків підтверджує важливість вибору правильної архітектури, мови програмування та інструментів для досягнення успішного результату. Акцент на використанні мови Kotlin та архітектурному патерні MVP (Model-View-Presenter) є обґрунтованим вибором, оскільки ці технології надають швидкість розробки, зручність у підтримці коду та високу продуктивність.

Зазначена інформація визначає напрямки подальшого дослідження та розробки з метою створення ефективного мобільного додатку для планування справ.

2 АНАЛІЗ ТЕХНОЛОГІЙ ТА ФУНКЦІОНАЛЬНОСТІ МОБІЛЬНОГО ДОДАТКУ

2.1 Функціональність мобільного додатку

Кожен додаток повинен мати певне призначення і функціональність для досягнення результату та виконання свого призначення, додаток розроблюваний для роботи не виключення. Основна функція даного додатку, це передача введених даних в базу даних та сортування їх по часу виконання. Спочатку буде вводитися завдання, термін виконання, і примітки, після чого додаток розміщає задачу на шкалі дня для більш очного розуміння. Якщо ж необхідно дані редагувати, то необхідно натиснути на конкретний день і обрати серед списку справ необхідну, та редагувати., детальніше про весь функціонал далі у розділі.

Розроблений мобільний додаток для планування справ представляє собою комплексну систему, спрямовану на підвищення ефективності управління часом та задачами користувача. Огляд його функціоналу розгорнуто в наступному науковому тексті.

Однією з ключових функцій додатку є можливість додавання та організації справ. Кожна задача може бути детально описана, а також може включати додаткові дані для більшої контекстної інформації. Такі дані впорядковуються в календарі на основі дати та часу, що сприяє легкому відслідковуванню та плануванню їх виконання.

Для полегшення організації та швидкого доступу до задач реалізовано функції сортування та фільтрації. Користувач може сортувати свій список справ за різними критеріями, такими як дата, пріоритет чи статус завдання. Крім того, фільтрація за різними категоріями дозволяє групувати задачі за певними параметрами.

Для поліпшення візуальної ідентифікації та взаємодії з користувачем

впроваджено функції маркування та категоризації. Завдання може бути відзначено різними кольорами на шкалі для легкого розрізнення, а користувач може створювати та призначати категорії для групування задач.

Система використовує базу даних, і в цьому контексті важливим є використання бібліотеки Room для ефективного зберігання даних про справи та задачі. Також внедрено функціонал для аналізу продуктивності користувача на основі виконаних справ та завдань.

Забезпечено інтерактивність та зручність використання через інтуїтивний і користувацькодружній інтерфейс. Процес постійного оновлення та додавання нових функцій розглядається як необхідний етап для підтримки високого рівня зручності користування та розвитку функціоналу.

Підсумково, розроблений мобільний додаток враховує різні аспекти ефективного управління завданнями та часом, виходячи з потреб користувача. Його функціонал піддається постійному розширенню та адаптації для надання нових можливостей та оптимізації користувацького досвіду.

На рисунку 2.1 зображена UML-діаграма прецедентів, а саме можливостей програми.

У ході програмного проектування передбачено та детально проаналізовано основні виняткові ситуації та можливі проблеми, які можуть виникнути під час експлуатації програмного забезпечення. Цей підхід надає можливість системі автоматично реагувати на непередбачувані обставини та ефективно вирішувати виниклі труднощі. В ході подальшого розвитку та удосконалення додатку планується додавання нових функціональностей, спрямованих на поліпшення зручності користування та попередження непередбачуваних аномалій у роботі системи. Цей постійний процес розширення функціоналу дозволяє динамічно адаптувати програмне забезпечення до змінних потреб користувачів та уникнути негативних наслідків внаслідок несподіваних сценаріїв використання.

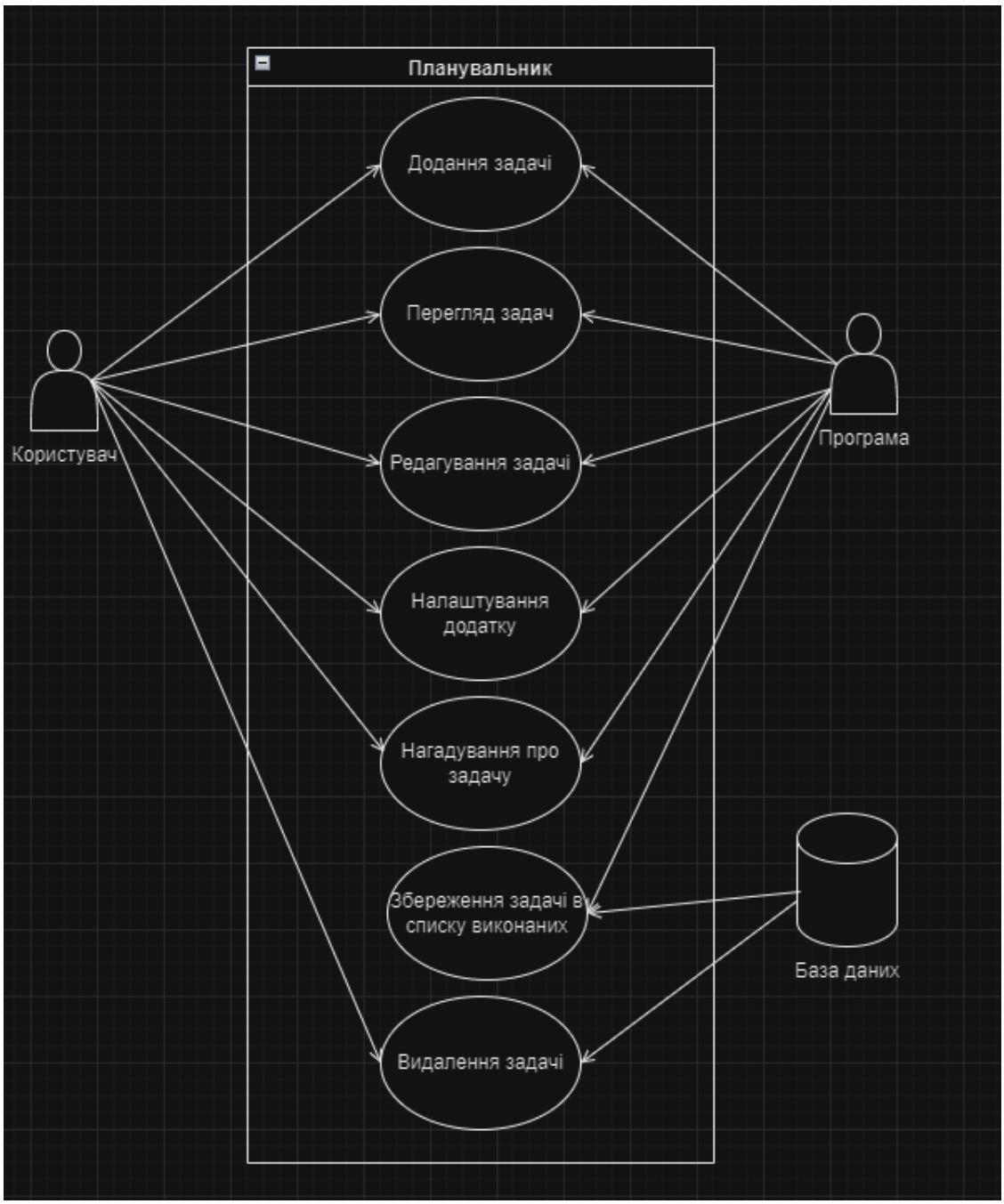


Рисунок 2.1 – UML-діаграма прецедентів

2.2 Принцип роботи архітектурних патернів для розробки Android додатків

Архітектурні шаблони в розробці програмного забезпечення є ключовим елементом для досягнення структурної організації коду, його підтримки, тестування та прискорення процесу розробки. Серед різних архітектурних шаблонів для розробки Android-додатків виокремлюють Model-View-Controller (MVC)[46], Model-View-Presenter (MVP)[47], Model-View-ViewModel (MVVM)

та Clean Architecture[48].

- Model-View-Controller (MVC): Цей шаблон розподіляє обов'язки між компонентами: Model, що відповідає за обробку даних, View – за відображення інтерфейсу, та Controller – за обробку введення та оновлення Model та View.

Призначення: Простота реалізації та зрозумілість.

- Model-View-Presenter (MVP): У шаблоні MVP Model обробляє дані, View – відображає їх, а Presenter – координує взаємодію між Model та View.

Призначення: Відокремлення бізнес-логіки від користувацького інтерфейсу, що полегшує тестування.

- Model-View-ViewModel (MVVM): В MVVM Model відповідає за обробку даних, View – за їх відображення, а ViewModel – за управління та подачу даних до View.

Призначення: Спрощення коду View, декларативна взаємодія з даними.

Clean Architecture [49]: Цей підхід розподіляє код на кілька рівнів (Entities, Use Cases, Interface Adapters, та Frameworks & Drivers), надаючи високу гнучкість та тестовість.

Призначення Забезпечення чистоти коду, високої тестовість та зручності заміни компонентів.

Історія розвитку цих шаблонів свідчить про пошук оптимального підходу до побудови архітектури програмного забезпечення, який враховує вимоги до зручності розробки, тестування та збереження чистоти коду в умовах постійної зміни технологій та вимог користувачів. Для зрозумілості вибору потрібно більш детально розписати шаблони.

MVVM (Model-View-ViewModel) — архітектурний шаблон проектування програмного забезпечення, який використовується для побудови і структуризації коду у мобільних та веб-додатках. MVVM розділяє компоненти програми на три ключові частини: Model, View і ViewModel, щоб сприяти розділенню відповідальностей та полегшити тестування та розширення коду. Схема роботи архітектурного шаблону зображено на Рисунку 2.1.

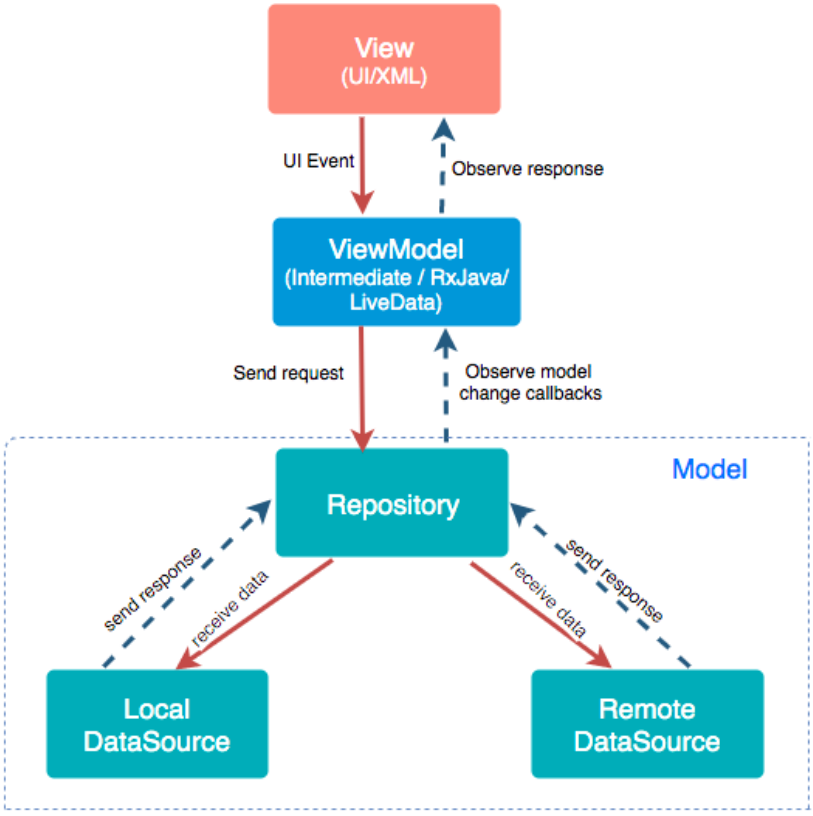


Рисунок 2.1. – Схема роботи патерну MVVM.

Model представляє собою об'єкти даних та логіку, які використовуються для доступу до даних, обробки бізнес-логіки та забезпечення відправлення оновлень. View відповідає за відображення інтерфейсу користувача та реагує на введення від користувача.

Основна ідея MVVM — використовувати проміжний шар, відомий як ViewModel, який виступає в ролі посередника між Model і View. ViewModel містить логіку, пов'язану з відображенням та обробкою подій від користувача. Він не повинен взаємодіяти безпосередньо з елементами інтерфейсу користувача, але надає дані та команди, які використовуються для відображення даних і виконання дій в View.

Цей підхід дозволяє розділити представлення даних від їх обробки та відображення, забезпечуючи більшу гнучкість та тестовуваність. ViewModel служить додатковим шаром абстракції, який дозволяє винести багато логіки взаємодії з даними з View та забезпечити легке управління станом додатка.

Також, MVVM може бути легко поєднаний з паттерном даних "Binding"

для автоматичного оновлення відображення при зміні даних у ViewModel. Це зменшує кількість ручного коду для оновлення інтерфейсу користувача та покращує зрозумілість коду.

Узагальнюючи, MVVM — це ефективний підхід для побудови структури мобільних та веб-додатків, що сприяє легкому розділенню логіки, полегшенню тестування та забезпеченню більшої читабельності коду.

Android Kotlin MVP (Model-View-Presenter) Architecture є однією з архітектурних концепцій розробки програмного забезпечення для Android-платформи. Ця архітектура спрямована на вдосконалення організації коду, забезпечуючи відокремлення бізнес-логіки від користувацького інтерфейсу і полегшуючи процес тестування.

MVP розподіляє програму на три основні компоненти: Model, яка відповідає за роботу з даними; View, що відображає інформацію користувачеві; і Presenter, який виступає посередником між Model та View. Основною ідеєю є те, що View і Model нічого не знають одне про одне, але обидві взаємодіють через Presenter.

Model відповідає за обробку даних, взаємодію з базою даних, мережевими запитаннями і будь-якою бізнес-логікою. Вона повинна бути абстрагована від деталей її використання в Presenter чи View.

View відповідає за відображення інформації користувачеві та обробку введення. Вона передає введені дані Presenter і відображає дані, які їй передає Presenter. View не містить бізнес-логіки і має залишатися максимально простою та гнучкою.

Presenter виступає посередником між Model і View. Він отримує введення від View, обробляє його, ініціює взаємодію з Model та відправляє результат назад у View для відображення. Presenter відповідає за управління станом та взаємодіє як з Model, так і з View.

Більшість проектів створених за допомогою даного шаблону наслідують один структурний приклад, що можна візуалізувати на прикладі Рисунку 2.2.

Особливості:

- Забезпечує високий рівень тестування завдяки відокремленню бізнес-логіки від користувацького інтерфейсу.

- Дозволяє просту заміну або розширення окремих компонентів без впливу на інші частини системи.

- Зменшує зв'язаність між компонентами, що полегшує розвиток та підтримку коду.

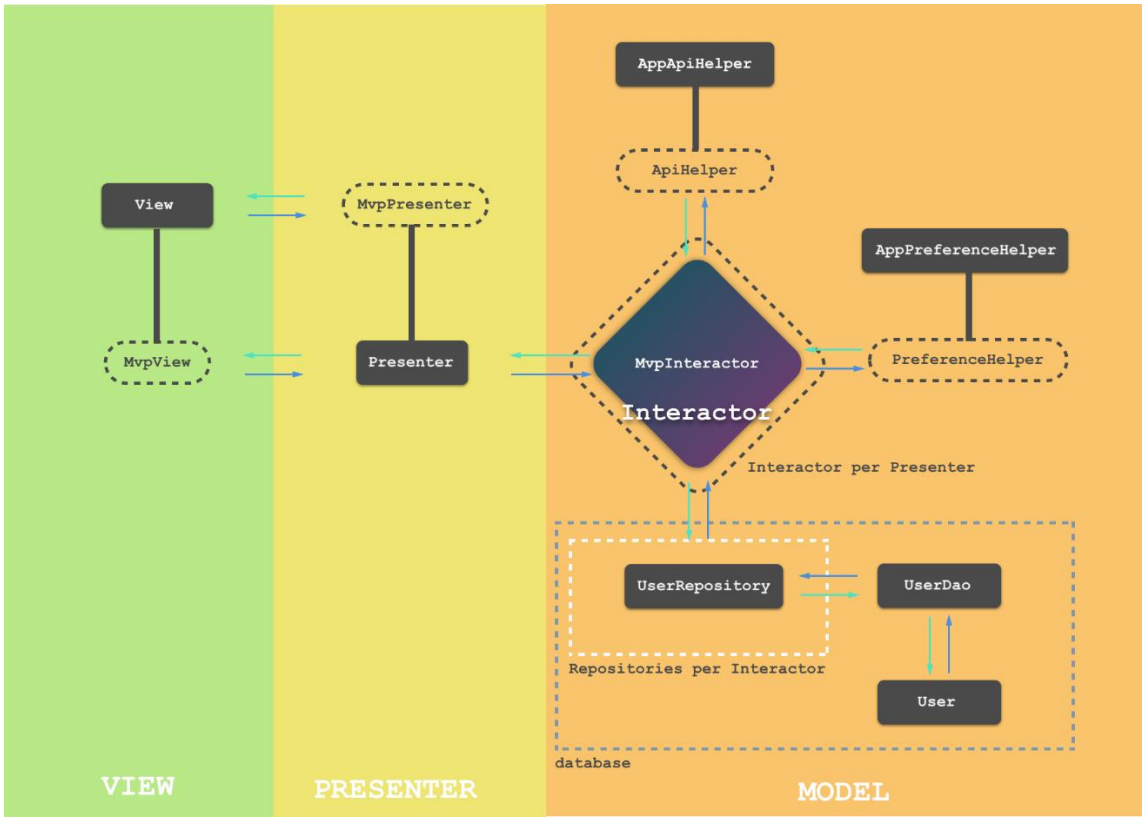


Рисунок 2.2 – Архітектура проекту

Android Kotlin MVP Architecture є потужним інструментом для розробки Android-додатків, спрямованим на полегшення розподілу обов'язків між компонентами, збереження чистоти коду та забезпечення високого рівня тестовості. Його застосування сприяє покращенню скальабельності та обслуговуваності додатків.

MVI, що означає "Model-View-Intent," є архітектурним підходом у розробці програмного забезпечення, спрямованим на побудову чистих та

прогнозованих застосунків, особливо підходить для реактивного програмування. Цей підхід визнається за свою спрощену обробку стану, зменшення змішаності та використання реактивних потоків.

Основні принципи MVI

Model (Модель) представляє собою поточний стан додатку. Вона є незмінною і визначає, як виглядає стан додатку в даний момент. Модель не може бути змінена безпосередньо; вона тільки читається.

View (Вид): поточний стан моделі. Він відображає дані та реагує на події від користувача. Вид не зберігає жодних даних стану, але взаємодіє з Presenter чи відправляє "наміри" (intents).

Intent (Намір): представляє собою події чи команди, які виникають від користувача або системи. Вони передаються усередину MVI для обробки. Наприклад, клік на кнопку чи отримання даних з сервера може бути наміром.

Reducer (Зменшувач) є функцією, яка приймає поточний стан і намір, і повертає новий стан. Ця функція визначає, які зміни слід внести в стан додатку відповідно до отриманих намірів.

Цикл оновлення стану (State Update Loop): Одиниця роботи MVI, де наміри відправляються до зменшувача, який оновлює модель. Після цього відбувається оновлення відображення з урахуванням нового стану.

Переваги MVI

Простота та прогнозованість: - MVI спрощує розробку, оскільки визначає чіткі правила для зміни стану.

Тестовість - Завдяки відсутності побічних ефектів та чистій функціональності, MVI добре піддається тестуванню.

Однозначність - Визначені ролі та чітка модель сприяють однозначності розробки та розумінню коду.

MVI - це архітектурний підхід, який полегшує розробку та тестування застосунків шляхом визначення чітких правил для зміни стану. Його функціональний підхід та реактивний характер роблять його привабливим

вибором для сучасних мобільних додатків.

У розробці програмного продукту вибір архітектурного шаблону є ключовим етапом, який визначає основні принципи та підходи до структуризації системи. В даному випадку, розглядається вибір Android Kotlin MVP Architecture, що базується на принципах розділення відповідальностей та створення легко змінюваного коду.

Однією з ключових переваг обраної архітектури є її здатність чітко розділяти компоненти системи на три основні частини: Model, View та Presenter. Це визначає зони відповідальності, забезпечуючи структурованість та модульність коду. Такий підхід дозволяє ефективно керувати кодом та здійснювати швидкі зміни в системі.

Далі важливим аспектом є її здатність чітко розділяти компоненти системи на три основні частини: Model, View та Presenter. Це визначає зони відповідальності, забезпечуючи структурованість та модульність коду. Такий підхід дозволяє ефективно керувати кодом та здійснювати швидкі зміни в системі.

Додатково, обрана архітектура сприяє простоті тестування. Presenter, який відповідає за логіку додатку, може бути легко та ізольовано протестований, оскільки він не залежить від Android-специфічних компонентів. Це робить тестування ефективним та забезпечує високий рівень якості коду.

Окрім того, MVP Architecture забезпечує зручну роботу зі станом додатку. Його структура дозволяє чітко керувати та уникати проблем, пов'язаних із станом системи. Це робить систему стабільною та легко підтримуваною в різних умовах.

Враховуючи всі переваги обраної архітектури, можна зробити висновок, що Android Kotlin MVP Architecture є стратегічним рішенням, спрямованим на досягнення ефективності та гнучкості у розробці програмного продукту. Вибір даного шаблону покладає основи для стійкості та високої якості коду, сприяючи успішній реалізації проекту.

2.3 Аналіз та підбір методів розробки інтерфейсів

У сучасному інформаційному суспільстві розробка мобільних додатків набула великого значення, враховуючи поширеність мобільних пристроїв серед користувачів. Одним із ключових аспектів розробки є створення ефективного та зручного інтерфейсу для взаємодії з додатком. У цьому контексті розробка інтерфейсів мобільних додатків є важливим напрямком, що вимагає комплексного підходу та вивчення різноманітних аспектів, таких як дизайн, взаємодія, адаптація та оптимізація.

Ефективність та привабливість додатку значною мірою залежать від дизайну та зовнішнього вигляду інтерфейсу. В той час , як розробка способів взаємодії користувача з додатком, включаючи елементи управління, жести та системи навігації, є ключовим етапом розробки інтерфейсів мобільних додатків.

Врахування різних розмірів екранів та орієнтацій мобільних пристроїв є необхідною умовою для створення ефективного інтерфейсу. Один з методів спрощення розробки та забезпечення консистентності інтерфейсу - використання готових компонентів.

Анімація відіграє важливу роль у візуальному досвіді користувача. Останнім етапом в розробці інтерфейсу є проведення тестів та оптимізація для забезпечення продуктивності та ефективності додатка.

Основними відомими методами розробки інтерфейсів є Jetpack Compose[50], та XML верста в контексті мобільних додатків, длі буде проаналізовано всі можливі аспекти і можливості обох методів, для вибору одного конкретного.

XML (Extensible Markup Language)[51] верстка в контексті Android використовується для створення інтерфейсів користувача за допомогою файлів розмітки XML. Ось інформація про іксмл верстку:

XML використовується в Android розробці практично з самого початку, коли було введено парадигму відокремлення розмітки від логіки додатку.

Формат XML дозволяє зручно представляти структуру інтерфейсу та описувати його вигляд, розміщення та властивості елементів.

Основні Переваги:

- XML дозволяє декларативно визначати структуру інтерфейсу, що спрощує читання та розуміння коду.
- XML дозволяє відокремити розмітку інтерфейсу від логіки додатку, що полегшує роботу дизайнерів та розробників.
- Багатий набір властивостей і можливостей XML дозволяє розробникам створювати різноманітні та креативні дизайни.
- XML верстка може використовуватися на різних платформах, оскільки це стандартний формат даних.

Недоліки:

- На великих проектах XML може привести до збільшення кількості коду через велику кількість тегів і атрибутів.
- У порівнянні з декларативними підходами, такими як Jetpack Compose, XML є менш гнучким і зручним для динамічних змін інтерфейсу.

Перспективи:

XML залишається популярним і широко використовується в Android-розробці. З оновленням Android та розвитком нових технологій, можливо, з'являться нові підходи до верстки інтерфейсу, але XML залишиться важливим інструментом для створення Android-додатків.

Jetpack Compose - це сучасний фреймворк для розробки інтерфейсів користувача в Android-додатках, який використовує декларативний підхід. Ось докладна інформація про Jetpack Compose, включаючи історію, переваги та недоліки:

Jetpack Compose був офіційно представлений на конференції Google I/O у травні 2019 року як експериментальний проект. Це було значущим кроком для Android-розробки, оскільки Compose змінює традиційний підхід до створення інтерфейсів, використовуючи код Kotlin замість XML. Протягом наступних

версій Jetpack Compose проходив багато етапів вдосконалення та розширення, і відверто підтримується Google як рекомендований спосіб створення інтерфейсів для Android-додатків.

Основні Переваги:

- Замість опису інтерфейсу у вигляді послідовності операцій, Compose дозволяє декларативно описувати, як має виглядати інтерфейс на конкретний момент часу.
- Оскільки Compose базується на Kotlin, розробники можуть використовувати всі переваги цієї мови, включаючи коротший та зрозумілий код.
- Compose легко розширюється і може використовуватися разом із існуючими View-based компонентами.
- Compose інтегрується із іншими Jetpack-бібліотеками та інструментами для розробки Android-додатків.

Недоліки:

- На етапі введення Jetpack Compose був у експериментальному стані, і це може викликати певні труднощі у розробці.
- Для тих, хто звик до традиційного підходу, використання Compose може вимагати часу для адаптації.
- У найновіших версіях бібліотеки може виникати проблеми з сумісністю, оскільки вона постійно розвивається.

Перспективи:

Jetpack Compose продовжує активно розвиватися, і з виходом стабільних версій очікується, що його використання стане більш розповсюдженим серед розробників Android-додатків. Зростаюча підтримка та розширення функціоналу сприятимуть популярності Jetpack Compose в майбутньому.

У даному випадку, обрано використання XML (eXtensible Markup Language) для розробки інтерфейсу в зв'язку з його простотою, широкою підтримкою та відомістю серед розробників.Обрання XML для розробки інтерфейсу є обдуманим стратегічним вибором для забезпечення простоти,

читабельності та ефективності розробки, а також легкої інтеграції та сумісності з іншими елементами проекту.

2.4 Висновки

В результаті обгрунтування основних аспектів функціоналу мобільного додатку, що розробляється було розглянуто основні можливі способи, засоби розробки та технології, можна використовувати в подібних проектах. Також було розглянуто основні переваги та спеціалізації технологій, можливі недоліки, та способи запобігання їх.

Побудовано блок-схему послідовності дій користувача в різних сценаріях використання мобільного додатку, та реакція самого додатку на вказані дії.

3 РОЗРОБКА ТА ТЕСТУВАННЯ МОБІЛЬНОГО ДОДАТКУ ДЛЯ ПЛАНУВАННЯ ТА ОБЛІКУ ЗАВДАНЬ

3.1 Вибір та аналіз інструментів розробки

Для будь-якої розробки потрібно обрати спочатку інструменти і засоби розробки за допомогою яких буде відбуватися розробка продукту, в даному випадку розробка мобільного додатку. Найбільш необхідними і базовими інструментами в процесі є мова програмування, середовище розробки, середовище тестування, середовище створення дизайну та бази дааних, а також всі аспекти мають мати усі необхідні функції для коректної взаємодії між собою.

Спершу потрібно зупинити пошуки серед мов програмування. Основними критеріями є легкість написання і розуміння коду, наявність необхідних в програмі функцій, швидкий вивід результату, та легкість в реалізації попереднього дизайну в середовищі розробки. Не менш важливим є наявність всіх функціонально-необхідних бібліотек для реалізації задуманого, зокрема бібліотеки, що пришвидшують, або забезпечують в цілому роботу з базами даних. Раніше по тексту було порівняно основні мови програмування, що підходять для розробки, а саме Kotlin та Java, так як це буквально найпопулярніші мови мобільної розробки.

На етапі зосередження уваги на мовах програмування, було обрано використовувати мову Kotlin, оскільки вона має обширний функіонал попри простоту і лаконічність написання коду. Має вбудовану бібліотеку, що забезпечує роботу з базами даних.

Найбільш базовим компонентом, що використовується в розробці є бібліотека RecyclerView. Вона визначає сучасний підхід до ефективного відображення значущих обсягів інформації. Цей інструмент забезпечує зручний механізм передачі та відображення великої кількості даних. Розробник визначає

структуру та вигляд кожного елемента, після чого бібліотека RecyclerView динамічно створює та управляє цими елементами відповідно до потреб користувача. Такий підхід дозволяє оптимально використовувати ресурси та покращує продуктивність відображення даних.

В умовах вимог до цієї роботи було обрано середовище розробки та тестування Android Studio Code, оскільки воно є безкоштовним, має повний список необхідних інструментів розробки.

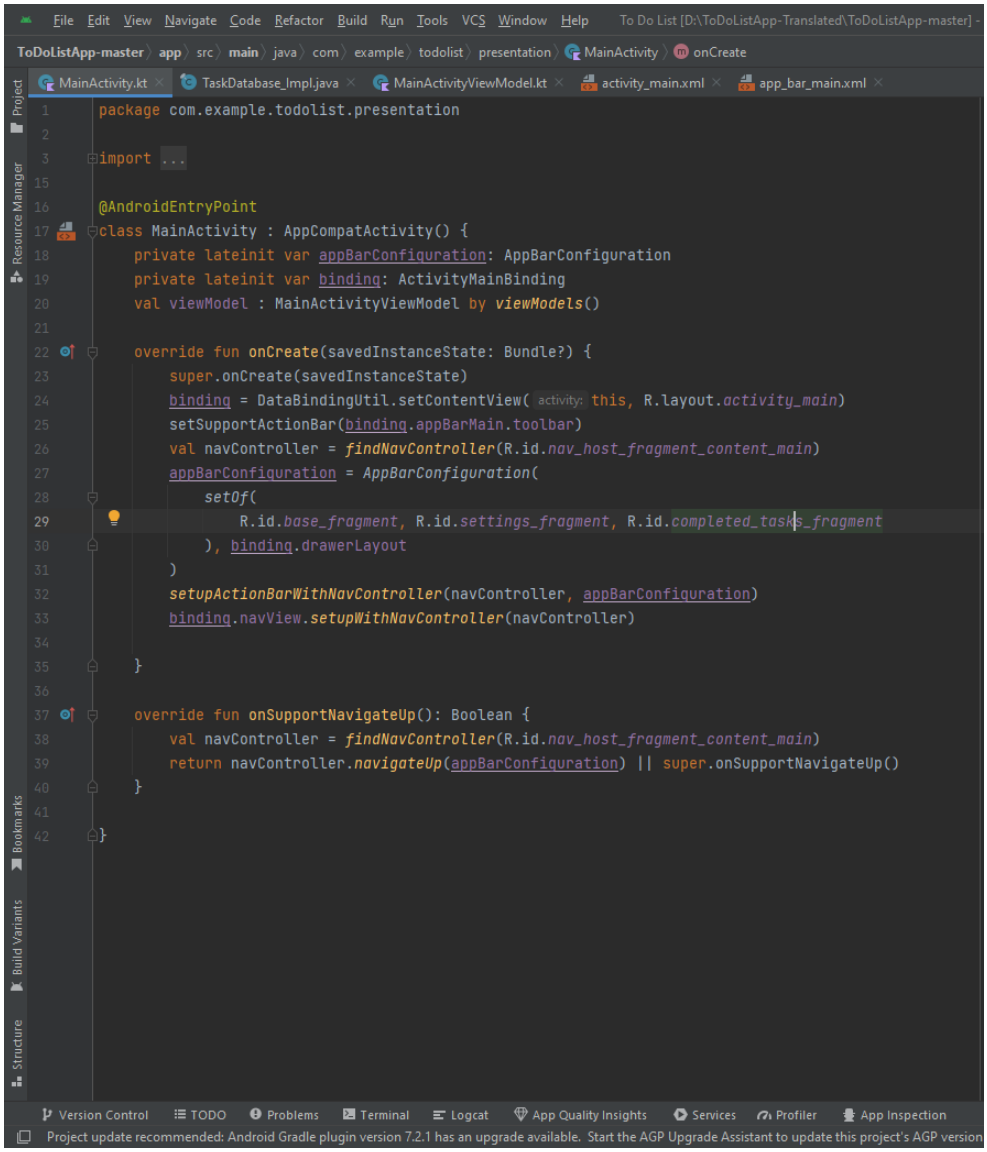


Рисунок 3.1 – Середовище розробки Android Studio Code

Після успішного встановлення робочого середовища для розробки додатка на мобільні платформи, необхідно докладно розглянути методи тестування та

перегляду розроблюваного продукту. Для мобільної розробки широко використовуються програмні емулятори, які в реальному часі відображають функціонал додатка на віртуальному телефоні. Після кожного збереження коду функціонал та інтерфейс змінюються відповідно до внесених змін у коді. Важливо відзначити, що цей метод тестування є належним та обгрунтованим, оскільки компіляція проекту в додаток для мобільного телефону забирає значний час, і використання емуляторів дозволяє зберегти час та зробити процес розробки більш ефективним.

Оскільки додаток, який буде розроблятися є одноплатформенним, то більше середовищ розробки не буде потрібно. Головним чином вся робота буде проводитися в Android Studio, і лиш розробка дизайну в Figma.

Спершу буде необхідно вибрати віртуальний мобільний пристрій на якому буде відбуватися тестування додатку, та перегляд поточного прогресу. В цьому випадку було обрано найновішу модель пристрою Pixel, а саме Pixel 3A, з версією API – 34.

Рисунок 3.2 – Емульований пристрій Pixel 3A

У даній роботі встановлення необхідних бібліотек планується проводити по мірі розробки з обґрунтованої причини оптимізації процесу. Такий підхід дозволяє уникнути зайвого навантаження та ресурсозатрат, оскільки встановлення бібліотек може бути виконано лише тоді, коли це необхідно для конкретного етапу розробки. Це сприяє ефективній роботі розробників та дозволяє уникнути надмірного збільшення обсягу використовуваних ресурсів на початковому етапі проекту.

Щодо обрання середовища для розробки, в даній роботі використовується лише Android Studio з обґрунтованою метою. Android Studio є офіційним інтегрованим середовищем розробки для Android, розробленим компанією Google. Воно надає широкий функціонал для розробки мобільних додатків, включаючи емуляцію пристроїв, візуальні редактори інтерфейсу та інструменти для налагодження. Використання одного офіційного середовища сприяє стабільності та взаємодії з різними компонентами Android-екосистеми, зменшуючи ймовірність конфліктів та покращуючи продуктивність розробки.

3.2 Розробка та тестування програмного забезпечення

Перед стартом розробки необхідно визначити основні функції програми та побічні(покращуючі роботу з додатком та підвищуючі зручність використання додатку). Додаток повинен включати такі функція як:

- Запис і збереження завдань;
- Редагування існуючих завдань;
- Передчасна відмітка виконаних заздалегідь;
- Наявну класифікацію справ;
- Надсилання сповіщень про дедлайн;
- Темна тема додатку;
- Англійська версія додатку при наявності такої ж базової мови на пристрої;

- Бокове меню(Навігація).

Локалізація додатку розроблялася окремо українська та англійська. За замовчуванням стоїть англійська відповідно до мови пристрою, якщо ж мова пристрою українська, то і мова додатку буде українська, приклади будуть представлені далі в розділі.

Основна навігація представлена у вигляді бокового меню. В ньому реалізована можливість зайти в налаштування, розділ з закінченими завданнями та повернутися на головний екран. Бокове меню створене за допомогою Drawer Layout, що реалізує можливість мови Kotlin відображення складних елементів з мінімумом коду. Вигляд бокового меню в додатку зображено на рисунку 3.3.

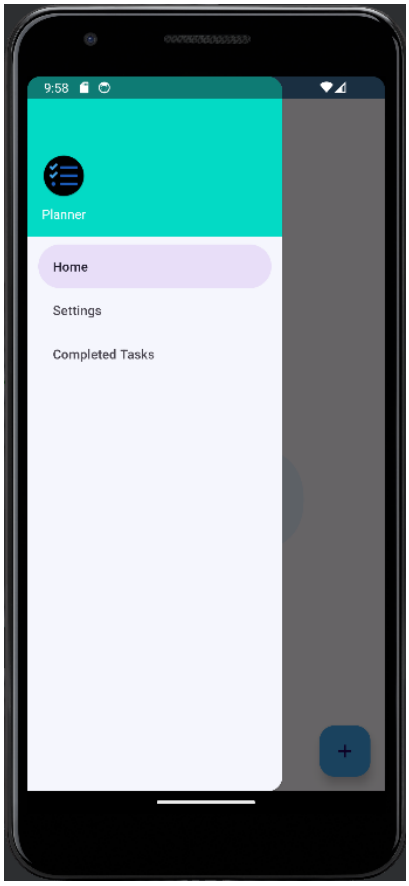


Рисунок 3.3 – Вигляд бокового меню

Так як програма залишається на стадії розробки, то бокове меню можливо буде мати більш розширений функціонал до моменту релізу в Google Play Market. На даний момент розробленого функціоналу вистачає аби додаток мав всі необхідні переваги над іншими додатками-аналогами.

При додаванні завдання необхідно натиснути на сегмент з кнопкою «+». Після чого ввести текст завдання, дедлайн,якщо такий є, обрати пріоритетність завдання, та категорію(за необхідності створити, так як це допомагає сортувати завдання). При створенні категорії є можливість обрати будь-який колір, задля більшої виразності серед списку, це допоможе набагато ефективніше орієнтуватися.

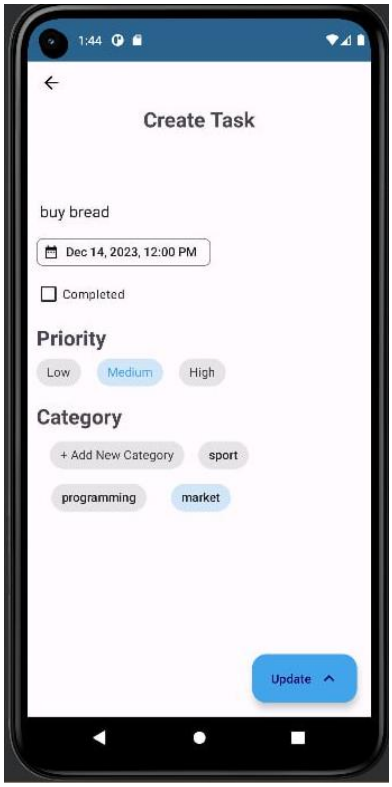


Рисунок 3.4 – Зображення меню створення завдання

Як можна побачити вже є створена тестова категорія «mono». Що показує справність підключених баз даних Room. Room надає зручний інтерфейс для роботи з SQLite-базою даних в Android, дозволяючи зберігати, оновлювати та отримувати дані в додатку. Ці функції дозволяють легко та ефективно взаємодіяти з базою даних в мобільному додатку, забезпечуючи зручний та високопродуктивний інтерфейс для роботи з персистентними даними. Більше доданих завдань можна побачити на рисунку 3.5.

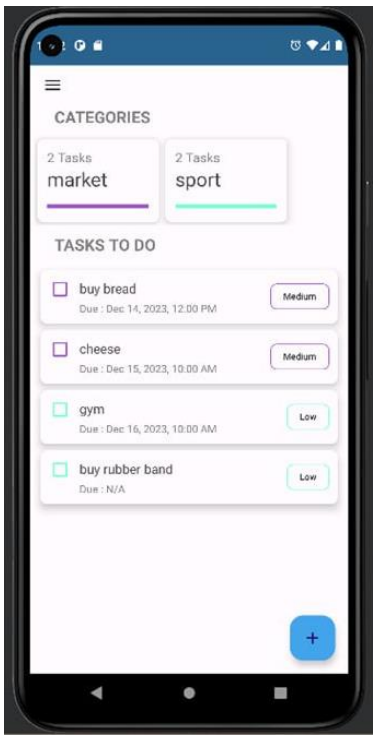


Рисунок 3.5 – Демонстрація збережених завдань

Зображений приклад демонструє, що завдання з різних категорій зображені списком скороченим у верхній каруселі, а більш розгорнуто видно нижче. Вибравжи з існуючих завдань можна перейти в меню редагування і відмітити як виконане.(Рисунок 3.6) В ході розробки було вирішено додати такі функції як поділ на категорії та пріоритетність задля збільшення ряду можливих галузей використання додатку. Наприклад як офісний список справ, так як категорії можуть відображати аспекти або спеціальності в яких треба виконати завдання, а пріоритетність же, впливає на сповіщення. Якщо завдання не підлягає терміновому виконанню, то і пріоритетність буде нижча і в списку буде показано нижче. При збіганні терміну виконання всіх завдань, де такий був зазначений буде показано Push-сповіщення. Проте для зручності було розроблено меню налаштувань, в якому можна відключити сповіщення про збігання часу певних пріоритетів. Також в налаштуваннях можна змінити тему на нічну/темну для зручності користувача. Приклад зображено на рисунку 3.7.

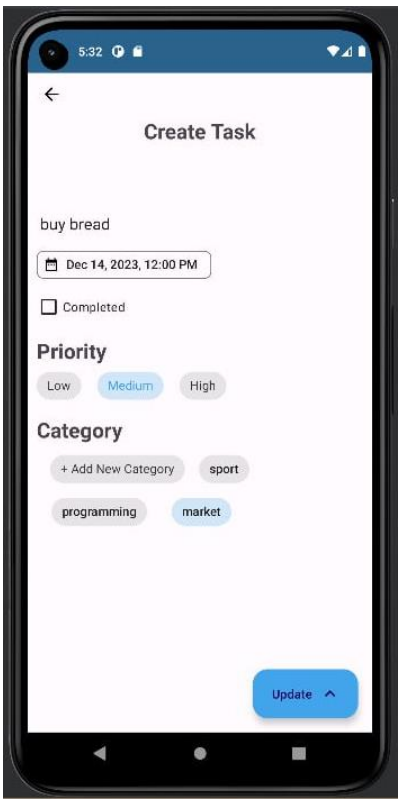


Рисунок 3.6 – Меню редагування існуючого завдання

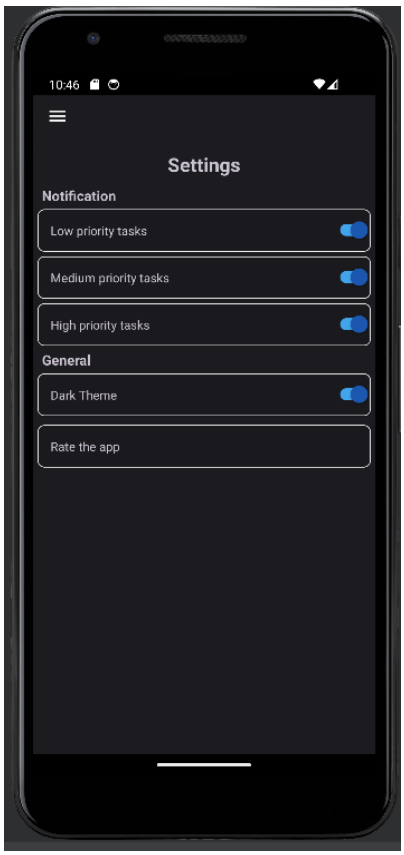


Рисунок 3.7 – Зображення меню налаштувань з увімкненим темним режимом інтерфейсу

Однією з важливих функцій є пункт меню, що виводить список виконаних завдань, за допомогою якого можна провести аналіз по успішності виконання завдань різних категорій і пріоритетів. Меню виконаних завдань зображено на рисунку 3.8.

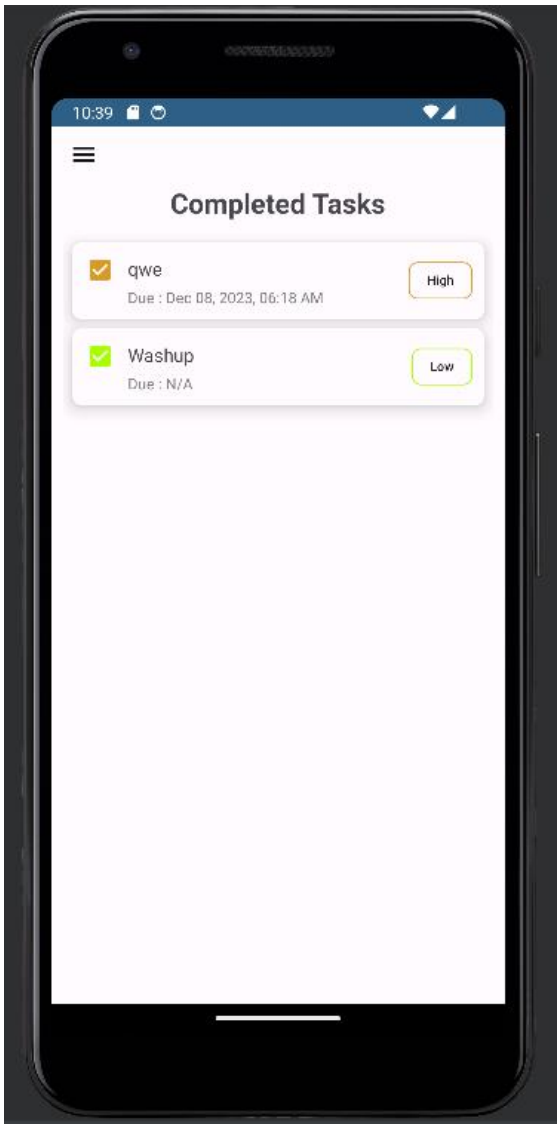


Рисунок 3.8 – Меню виконаних завдань

Абсолютно всі сторінки, на які ведуть кнопки, або пункти меню були реалізовані за допомогою Fragment. Фрагмент в Kotlin — це один з ключових компонентів розробки інтерфейсу користувача в Android-додатках. Фрагмент представляє собою частину користувацького інтерфейсу або поведінки, яку

можна використовувати та пере використовувати в різних частинах додатка. Використання фрагментів дозволяє створювати більш гнучкі та адаптивні інтерфейси для різних розмірів екрану та орієнтацій пристрою. Використання фрагментів дозволяє створювати більш ефективні та повторно використовувані компоненти для побудови складних інтерфейсів у додатках для платформи Android.

Activity в Kotlin - це основний компонент для створення користувацького інтерфейсу та взаємодії з користувачем в мобільних додатках на платформі Android. Activity представляє собою один екран додатка, і вона відповідає за відображення графічних елементів, обробку подій та взаємодію з іншими компонентами додатка. Цикл життя Activity - це набір методів, які викликаються системою Android під час різних фаз життєвого циклу Activity. Знання цього циклу дуже важливе для правильного управління ресурсами та взаємодії з користувачем. Основні методи циклу життя Activity включають наступні фази:

- onCreate(): Цей метод викликається, коли Activity створюється. В ньому зазвичай проводять ініціалізацію змінних та ресурсів.
- **onStart(): Activity стає видимою для користувача. У цьому методі можна виконувати операції, які повинні відбутися при початку взаємодії з користувачем.
- onResume(): Викликається після завершення методу onStart(). Activity стає активною та готовою до взаємодії з користувачем.
- onPause(): Цей метод викликається, коли інша Activity стає видимою, а поточна Activity переходить в фоновий режим. Тут зазвичай зберігають стан Activity.
- onStop(): Activity стає невидимою для користувача. В цьому методі можна виконувати операції звільнення ресурсів.
- onDestroy(): Цей метод викликається перед закриттям Activity. В ньому слід виконати завершення роботи з ресурсами та очищення пам'яті.

Це основні методи циклу життя Activity. Проте, Activity може також переходити в інші стани, наприклад, коли користувач повертається до Activity зі стеку навігації (onRestart()), або коли Activity знищується системою через нестачу

ресурсів. Знання і правильне використання цих методів допомагає забезпечити коректну роботу додатка і оптимальне використання ресурсів.

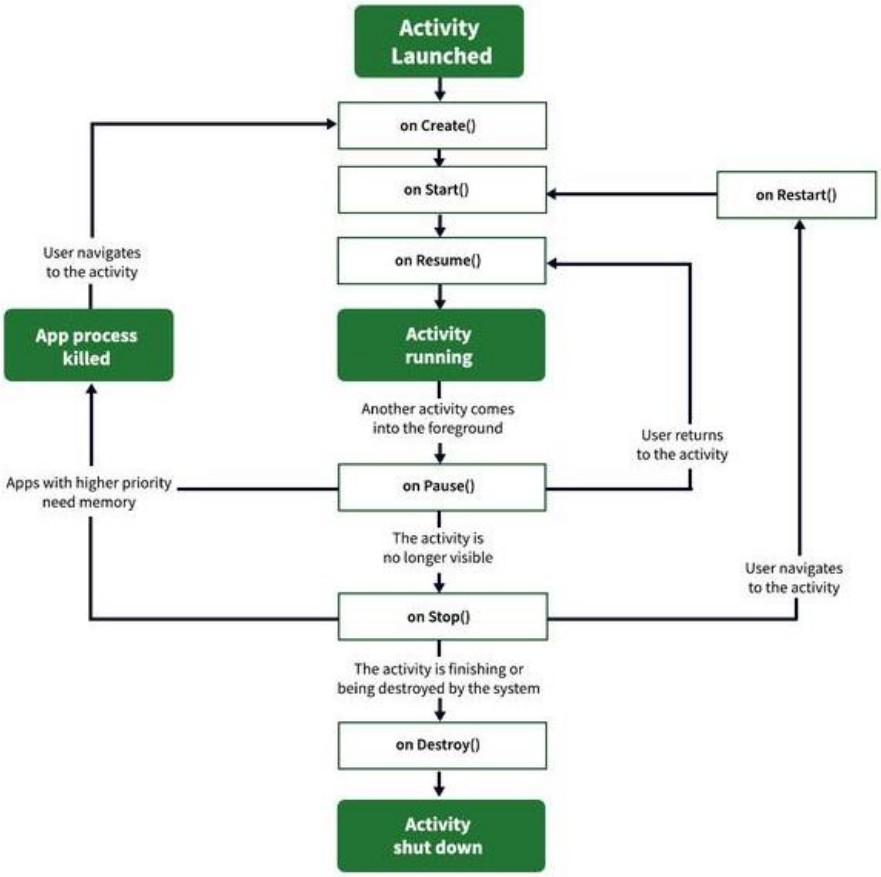


Рисунок 3.9 – Діаграма діяльності Activity

Важливим аспектом роботи є розробка додаткової локалізації інтерфейсу. Використання англійської мови для базового інтерфейсу програми в даній роботі обгрунтоване кількома фундаментальними причинами.

По-перше, англійська мова є глобально визнаною мовою в галузі інформаційних технологій та програмування. Значна частина документації, технічних ресурсів та бібліотек доступна англійською, що спрощує взаємодію та обмін знаннями з глобальною спільнотою розробників.

По-друге, використання англійської мови у найменуваннях полегшує інтеграцію з екосистемою. Технічні терміни та ключові слова вже стали стандартом у англійському використанні, що сприяє зручній інтеграції з

існуючими фреймворками та рішеннями.

По-третє, англійська мова є універсальною та зрозумілою для програмістів з різних країн та культур, що гарантує універсальність програми та сприяє зручній взаємодії з користувачами із різних частин світу.

По-четверте, використання англійської мови спрощує розробку та документацію. Багато інструментів розробки мають найкращу підтримку для англійської мови, а саме вона є стандартом для написання коментарів та документації, що полегшує співпрацю та обмін інформацією між розробниками.

З іншого боку, використання української мови для вспоміжного інтерфейсу визначено як стратегічний вибір для покращення зручності користування програмою місцевою аудиторією. Це рішення забезпечує комфорт та доступність для українськомовних користувачів, роблячи продукт більш дружнім та придатним для максимального комфорту української аудиторії.

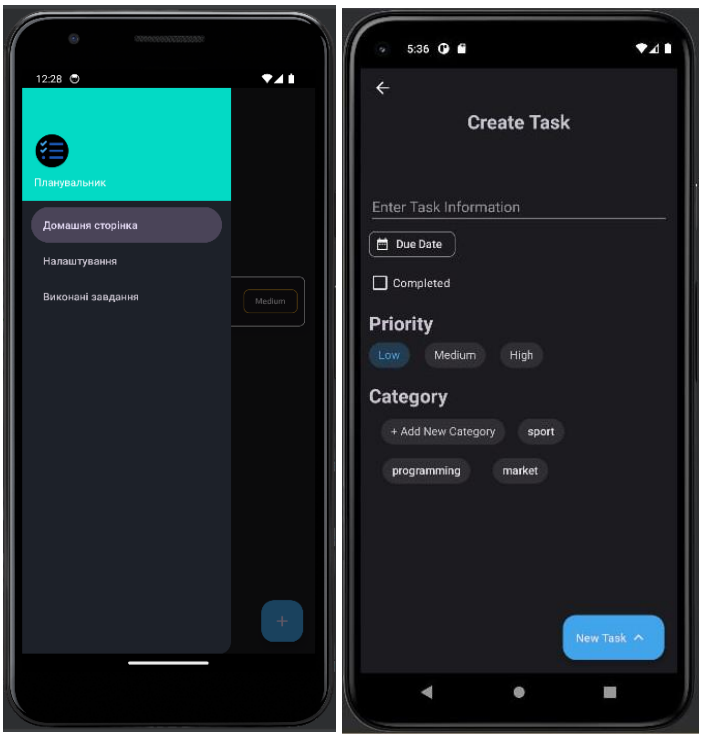


Рисунок 3.10 – Зображенні версії програми з українською локалізацією

Реалізація локалізації а додатку була виконана завдяки особливості

побудови файлової комплексності в мові Kotlin, що була успадкована від java. Додавання іншої локалізації для слів в Android Studio з використанням мови Kotlin передбачає використання ресурсів для текстових рядків у відповідних файлів ресурсів та роботу із класом `Resources`. Ось кроки, які можна виконати:

Створення Файлу Ресурсів:

Було необхідно створити папку `res` у проекті, якщо вона вже не існує. У цій папці створити підпапку `values`, яка буде містити ресурси для рядків. У цій папці створити файл `strings.xml` або скористатисяя вже існуючим файлом, якщо він є. Файл `strings.xml` виглядатиме приблизно так:

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">MyApp</string>
    <!-- Додайте інші рядки за потребою -->
</resources>
```

Додавання Локалізацій:

У папці `res` створити нові папки `values-xx`, де `xx` - це код мови локалізації (наприклад, `values-uk` для української). У цих папках створити аналогічні файли `strings.xml` та переклсти рядки на відповідну мову. Приклад для української локалізації (`values-uk/strings.xml`):

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Моя програма</string>
    <!-- Додайте інші рядки за потребою -->
</resources>
```

```

Використання Ресурсів у Коді Kotlin:

Код на Kotlin може отримати доступ до рядків з ресурсів, використовуючи клас `Resources`. Наприклад:

```kotlin
val appName = resources.getString(R.string.app_name)
```

У цьому прикладі, якщо обрана локалізація `uk`, то `appName` отримає значення "Моя програма".

Зміна Локалізації Програмно (Опціонально):

Якщо потрібно змінювати локалізацію програмно, можна використовувати клас `Configuration` разом із `Context`. Наприклад, для установки локалізації на українську:

```kotlin
val configuration = Configuration(resources.configuration)
configuration.setLocale(Locale("uk"))
val context = createConfigurationContext(configuration)
```

Після цього можна використовувати `context` для отримання ресурсів із вибраною локалізацією.

Варто зазначити, що це лише загальний підхід, і в інших ситуаціях може знадобитися додаткова настройка в залежності від особливостей проекту. Файл локалізації в даній роботі ж виглядає наступним чином:

```
<resources>
    <string name="app_name">Планувальник</string>
```
```

```
    <!-- TODO: Remove or change this placeholder text -->
    <string name="hello_blank_fragment">Hello blank
fragment</string>
    <string name="title_activity_main2">MainActivity2</string>
    <string name="navigation_drawer_open">Open navigation
drawer</string>
    <string name="navigation_drawer_close">Close navigation
drawer</string>
    <string name="nav_header_title">Android Studio</string>
    <string
name="nav_header_subtitle">android.studio@android.com</string>
    <string name="nav_header_desc">Navigation header</string>
    <string name="action_settings">Settings</string>

    <string name="menu_home">Домашня сторінка</string>
    <string name="menu_gallery">Галерея</string>
    <string name="menu_slideshow">Слайдшоу</string>
    <string name="add_category">Додати категорії</string>
    <string name="enter_category">Виберіть категорії</string>
    <string name="category_color">Колір категорії</string>
    <string name="add_color">Додати колір</string>
    <string name="categories">Категорії</string>
    <string name="tasks_to_do">Завдання</string>
    <string name="completed_tasks">Виконані завдання</string>
    <string name="create_task">Створити завдання</string>
    <string name="enter_task_information">Введіть завдання</string>
    <string name="due_date">Термін</string>
    <string name="completed">Виконано</string>
    <string name="priority">Пріоритетність</string>
    <string name="low">Низький</string>
    <string name="medium">Середній</string>
    <string name="high">Високий</string>
    <string name="category">Категорії</string>
    <string name="add_new_category">Додати категорію</string>
    <string name="new_task">Нове завдання</string>
    <string name="settings">Налаштування</string>
    <string name="notification">Сповіщення</string>
    <string name="low_priority_tasks">Завдання низької
пріоритетності</string>
    <string name="medium_priority_tasks">Завдання середньої
пріоритетності</string>
    <string name="high_priority_tasks">Завдання високої
пріоритетності</string>
    <string name="general">Загальні</string>
    <string name="rate_the_app">Оцінити програму</string>
</resources>
```

3.3 Висновки

В даному розділі надано докладні вказівки щодо користування програмою для планування справ, розробленою на основі архітектурного шаблону Android Kotlin MVP (Model-View-Presenter) Architecture. Детально описано функціонал програми та наведено приклади використання кожної з основних можливостей.

Користувач може зручно вводити опис та додаткові дані для кожної справи, організовувати їх в календарі на основі дати і часу, а також аналізувати продуктивність своєї роботи на основі виконаних завдань. Реалізовано сортування списку справ за різними характеристиками, що забезпечує зручність користування програмою.

Важливим аспектом розробки є використання мови програмування Kotlin та патерну MVP, що сприяє створенню чистого та організованого коду. Введено поняття моделі, представлення та презентера для ефективного управління логікою програми та відокремлення компонентів для підтримки майбутнього розвитку.

Також, наведено приклади використання бібліотеки Room для роботи з базою даних, що дозволяє зберігати та синхронізувати дані між різними пристроями. Застосування цих технологій покращує надійність та швидкість роботи програми.

Загальною метою даного проєкту є підвищення ефективності ведення та організації списків справ, забезпечення зручного взаємодії користувача з програмою та можливості подальшого розвитку та розширення функціоналу.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Комерційний та технологічний аудит науково-технічної розробки

В сучасному світі, де час – це один із найцінніших ресурсів, задачі планування і обліку завдань стають надзвичайно важливими. Незалежно від того, чи ви студент, професіонал у сфері бізнесу чи просто людина, яка шукає спосіб ефективніше управляти своїм часом, автоматизована система планування стане надійним помічником. Вона допомагає організовувати завдання, визначати пріоритети та раціонально розподіляти час для досягнення максимальної продуктивності.

Однією з основних проблем, які стикаються користувачі, є недостатня ефективність традиційних методів планування, таких як записи вручну або використання паперових щоденників. Ці методи вимагають багато часу і зусиль, і часто не забезпечують гнучкість і зручність в управлінні завданнями. Тому розробка автоматизованої системи планування стає актуальним завданням, яке може революціонізувати спосіб, яким ми ведемо своє повсякденне життя і роботу.

Основною метою цієї роботи є розробка програмного забезпечення, яке допоможе користувачам ефективніше планувати, відстежувати і виконувати завдання. Вона буде спрямована на полегшення процесу планування і оптимізацію використання часу, щоб досягти кращих результатів. Завдяки цій роботі, ми маємо намір внести вагому спільний внесок у покращення управління часом та ефективності роботи кожного користувача.

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності. Магістерська кваліфікаційна робота за темою тему «Розробка автоматизованої системи планування і обліку завдань. Частина 1 Розробка мобільного додатку»: передбачає розробку програмного забезпечення, яке може задовільнити окремих осіб або компаній розраховуючи свій час та плани на поточний

день/тиждень/місяць.

Проведемо оцінювання комерційного потенціалу даної розробки. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу буде проведено на основі відгуку трьох експертів: Ковалюк Олег Олександрович – науковий керівник, доцент кафедри комп`ютерних систем управління ВНТУ, кандидат технічних наук; Кавецький Вячеслав Валерійович доцент кафедри менеджменту, маркетингу та економіки ВНТУ; Галущак Анастасія Володимирівна – асистент кафедри комп'ютерних наук ВНТУ; із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із таблицею 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	Критерій	Критерій	Критерій	Критерій	Критерій
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові	Потрібні незначні фінансові	Потрібні значні фінансові	Потрібні незначні фінансові	Не потребує додаткового фінансуванн

	ресурси, які відсутні. Джерела фінансування ідеї відсутні	ресурси. Джерела фінансування відсутні	ресурси. Джерела фінансування є	ресурси. Джерела фінансування є	я

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	Критерій	Критерій	Критерій	Критерій	Критерій
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки занесемо у таблицю 4.2.

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Ковалюк О. О.	Кавецький В. В.	Галущак А. В.
	Бали, виставлені експертами:		
1	3	2	3
2	3	2	3
3	3	4	3
4	3	3	4
5	3	3	2
6	3	3	2
7	4	3	2
8	3	3	3
9	3	3	2
10	4	4	4
11	4	3	3
12	3	3	2
Сума балів	$СБ_1$=39	$СБ_2$=36	$СБ_3$=33
Середньоарифметична сума балів	*(39+36+33) / 3 = 36*		

За даними таблиці 4.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 4.3.

Таблиця 4.3 − Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Рівень комерційного потенціалу розробки, становить 36 балів. Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є вище середнього. Дана розробка буде корисним продуктом для людей в яких власний час дорожче за гроші, а гроші можна отримати за час. Відносно

конкурентів вона буде кращою завдяки свому удобному інтерфейсу. Основний елемент програми, який дозволяє бачити користувачеві свої задачі у так званому таймлайні, дозволяє програмному забезпеченню виділятися на фоні конкурентів, та надавати додаткові можливості при користуванні програмою, а значить і при плануванні свого часу.

4.2 Прогнозування витрат на виконання науково-дослідної(дослідно-конструкторської) роботи

Проведемо прогнозування витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи для розробки програмного забезпечення. Процес прогнозування можна розділити на три етапи:

- розрахунок витрат на виконавців даного розділу роботи;
- розрахунок загальних витрат на виконання роботи;
- прогнозування загальних витрат на виконання та впровадження результатів даної роботи.

Виконаємо розрахунок витрат, основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t \qquad (4.1)$$

де M – місячний посадовий оклад конкретного розробника (дослідника), грн;

$T_р$ – число робочих днів в місяці, 22 днів;

t – число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 4.4.

Таблиця 4.4 – Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту, дизайнер	17600	800	56	44800
Експерт у системній логістиці	16500	750	45	33750
Програміст-Спеціаліст, спеціаліст по базам даних	26400	1200	53	63600
Всього				142150

Додаткова заробітна плата прийнято розраховувати як 15 % від основної заробітної плати розробників та робітників:

$$З_д = З_о * \frac{15\%}{100\%}; \quad (4.2)$$

$$З_д = 142150 * \frac{15\%}{100\%} = 21323 \text{ (грн.)}.$$

Нарахування на заробітну плату $Н_{ЗП}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується за формулою:

$$Н_з = (З_о + З_д) * \frac{\beta}{100\%}; \quad (4.3)$$

де $З_0$ – основна заробітна плата розробника, грн;

$З_Д$ – додаткова заробітна плата розробника, грн;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування.

Згідно діючого законодавства нарахування на заробітну плату складають

22% від суми основної та додаткової заробітної плати.

$$Н_з = (142150 + 21323) * \frac{22\%}{100\%} = 35964{,}06 \text{ (грн.)}.$$

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби й предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за прямим призначенням згідно з нормами їх витрачання, а також витрачені придбані напівфабрикати, що підлягають монтажу або виготовленню й додатковій обробці в цій організації, чи дослідні зразки, що виготовляються виробниками за документацією наукової організації.

Таблиця 4.5 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір для принтера А4 Zoom 500шт., щільність 80 г/м²	260	1	-	-	268
Набір олійних ручок Buromax Orion 0.5 мм Тригранний корпус Сині 12 шт	64	1	-	-	64
Маркер перманентний Centropen 1 мм	15,7	10	-	-	157
Всього:					489

До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10…12% від вартості програмного забезпечення. Балансову вартість програмного забезпечення розраховують за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \tag{4.4}$$

де *Ціпрг* – ціна придбання одиниці програмного засобу цього виду, грн;

Спрг.і – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

Кі – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (*Кі* = 1,10…1,12);

k – кількість найменувань програмних засобів.

Отримані результати необхідно звести до таблиці.

Таблиця 4.6 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ОС Windows 11 Pro	1	10000	11200
Прикладний пакет Microsoft Office	1	6500	7800
Програмне середовище розробки Android Studio	1	250	280
Програмне середовище для створення макетів Figma	1	300	336
Всього:			19616

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді розраховується за формулою:

$$A = \frac{Ц}{Т_в} * \frac{t_{вик}}{12} \ [грн.], \tag{4.5}$$

де *Ц* – балансова вартість обладнання, грн;

Т – термін корисного використання обладнання згідно податкового законодавства, років;

$t_{вик}$ – термін використання під час розробки, місяців.

Розрахуємо, для прикладу, амортизаційні витрати на смартфон балансова вартість якого становить 14000 грн, термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 2 міс.

$$А_{обл} = \frac{15000}{2} * \frac{3}{12} = 1875 \text{ (грн)}.$$

Визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 4.7. Для розрахунку амортизації нематеріальних ресурсів використовується формула:

$$А_{н.р.} = Ц_{н.р.} * Н_а * \frac{t_{вик}}{12}. \quad (4.6)$$

Проте, вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн, то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю.

Таблиця 4.7 – Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн.
Комп’ютер та комп’ютерна периферія	28000	2	3	3500
Смартфон для тестування	15000	2	3	1875
Приміщення	412000	20	3	5150
Всього				10525

Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільчих компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільчих компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В * П * Ф * К_п, \quad (4.7)$$

де *В* – вартість 1 кВт-години електроенергії для 1 класу підприємства, *В* = 7,5 грн/кВт;

П – встановлена потужність обладнання, кВт. *П* = 0,5 кВт;

Ф – фактична кількість годин роботи обладнання, годин;

$К_п$ – коефіцієнт використання потужності, $К_п$ = 0,9,

Використаємо дану формулу для розрахунку споживання електроенергії комп'ютером:

$$В_е = 7,5 * 0,5 * 270 * 0,9 = 911,25(грн.).$$

Таблиця 4.8 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Комп'ютер та комп'ютерна периферія	0,5	270	911,25
Смартфон	0,005	130	4,39
Всього:			915,64

Статті «Витрати на службові відрядження» немає, то ж витрат на них не

передбачено. До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$\text{І}_\text{в} = \left(\text{З}_\text{о} * \text{З}_\text{р}\right) * \frac{\text{Н}_\text{ів}}{100\%}, \quad (4.8)$$

де $H_{ів}$ – норма нарахування за статтею «Інші витрати»,

$$\text{І}_\text{в} = 142150 * \frac{110\%}{100\%} = \ 156365\ (\text{грн.}).$$

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$\text{Н}_\text{нзв} = \left(\text{З}_\text{о} * \text{З}_\text{р}\right) * \frac{\text{Н}_\text{ів}}{100\%}, \quad (4.9)$$

де $H_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$\text{Н}_\text{нзв} = 142150\ * \frac{120\%}{100\%} = \ 170580\ (\text{грн.}).$$

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$$\text{В}_\text{заг} = \ 142150 + 21323 + 35964{,}06 + 489 + 19616 + 10525 + 843{,}75 + 156365 + 170580 \ = \ 557\,855{,}81(\text{грн.}).$$

Фінальним етапом проведемо обрахунок загальних витрат на завершення

науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються *ЗВ*, визначається за формулою:

$$ЗВ = \frac{В_{заг}}{\eta}[грн.], \quad (4.10)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії:

– науково-дослідних робіт, то η=0,1;

– технічного проектування, то η=0,2;

– розробки конструкторської документації, то η=0,3;

– розробки технологій, то η=0,4;

– розробки дослідного зразка, то η=0,5;

– розробки промислового зразка, то η=0,7;

– впровадження, то η=0,9.

Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

$$ЗВ = \frac{557\,855{,}81}{0{,}9} = 619\,839{,}79(грн).$$

4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

Витрати на провадження дослідження складаються з витрат на провадження розробок за напрямками «Розробка автоматизованої системи планування і обліку завдань. Частина 1 Розробка дизайну мобільного додатку та бази даних» та «Розробка автоматизованої системи планування і обліку завдань. Частина 2 Розробка мобільного додатку». Сумарні витрати складуть:

$$ЗВ = 619\,839{,}79 + 486\,139{,}809 = 1\,105\,979{,}6$$

Збільшення чистого прибутку є позитивним результатом, який може отримати інвестор при вкладені певного об'єму ресурсів. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації розроблювального інформаційного продукту.

В даній ситуації розглядається розробка чи суттєве вдосконалення програмного засобу. Тоді майбутній економічний ефект буде формуватися на основі таких даних: ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик; N –кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки; $Цб$ – вартість програмного продукту у році до впровадження результатів розробки; $\pm\Delta Цо$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу.

Майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm\Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (4.11)$$

де $\pm\Delta Ц_о$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_о$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_о = Ц_б \pm \Delta Ц_о$;

$Цб$ – вартість програмного продукту у році до впровадження результатів розробки;

ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$;

p – коефіцієнт, який враховує рентабельність продукту;

ϑ – ставка податку на прибуток, у 2023 році ϑ =18%.

Припустимо, що при прогнозованій ціні 300 грн за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на 100 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 30000 шт., протягом другого року – на 20000 шт., протягом третього року на 18000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$$\Delta П1 = (500 * 100000 + (0 + 300) * 30000) * 0{,}8333 * 0{,}49 * (1 - 0{,}18) = = 19\,754\,376{,}46\ (\text{грн.}),$$

$$\Delta П2 = (500 * 100000 + (300 + 100) * (30000 + 20000)) * 0{,}8333 * 0{,}49 * (1 - 0{,}18) = 23\,437\,395{,}8(\text{грн.}),$$

$$\Delta П3 = (500 * 100000 + (400 + 100) * (30000 + 20000 + 18000)) * 0{,}8333 * 0{,}49 * (1 - 0{,}18) = 28\,124\,874{,}96(\text{грн.}).$$

Отже, відповідно до проведених розрахунків, прогнозований комерційний ефект від впровадження розробки виражається у значному

збільшенні чистого прибутку підприємства.

Проведемо розрахунок ефективності вкладених інвестицій та періоду їх окупності. Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.12)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

T – період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0{,}05 \ldots 0{,}15$;

t – період часу (в роках).

Збільшення прибутку ми отримаємо починаючи з першого року:

$$\text{ПП} = \left(\frac{19\,754\,376{,}46}{(1+0{,}15)^1}\right) + \left(\frac{23\,437\,395{,}8}{(1+0{,}15)^2}\right) + \left(\frac{28\,124\,874{,}96}{(1+0{,}15)^3}\right) =$$
$$= 17\,177\,718{,}66 + 17\,722\,038{,}41 + 18\,492\,561{,}82 = 53\,392\,318{,}89 \text{ (грн.)}.$$

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{\text{інв}} * \text{ЗВ}, \quad (4.13)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку

приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 * 1\,105\,979{,}6 = 2\,211\,959{,}2 (\text{грн.}).$$

Тоді абсолютний економічний ефект *Еабс* або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$\text{Е}_{\text{абс}} = \text{ПП} - PV, \qquad (4.14)$$

$$\text{Е}_{\text{абс}} = 53\,392\,318{,}89 - 2\,211\,959{,}2 = 51\,180\,359{,}7 \text{ (грн.)}.$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-технічної роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої норми дохідності вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$\text{Е}_{\text{в}} = \sqrt[\text{Т}_{\text{ж}}]{1 + \frac{\text{Е}_{\text{абс}}}{PV}} - 1, \qquad (4.15)$$

де $T_{ж}$ – життєвий цикл наукової розробки, роки.

$$\mathrm{E_в} = \sqrt[3]{1 + \frac{51\,180\,359{,}7}{2\,211\,959{,}2}} - 1 = 1{,}9.$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \qquad (4.16)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d = (0,09…0,14);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{min} = 0{,}14 + 0{,}4 = 0{,}54.$$

Оскільки $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$\mathrm{T_{ок}} = \frac{1}{\mathrm{E_в}}, \qquad (4.17)$$

$$\mathrm{T_{ок}} = \frac{1}{1{,}9} = 0{,}53(\text{р.}).$$

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,23 роки, то фінансування даної наукової розробки є доцільним.

4.4 Висновок

Економічна частина роботи містить розрахунок витрат на розробку нового програмного продукту. Загальна сума витрат складає 1 105 979,6 гривень.

Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні цієї розробки.

Термін окупності розробки програмного забезпечення складає 0,53 роки. Тому інвестування в розробку автоматизованої системи планування і обліку завдань є доцільним для інвесторів.

ВИСНОВКИ

В ході виконання наукової роботи було проведено глибоке вивчення сучасних технологій мобільної розробки, зосереджуючись на платформі Android та використанні мови програмування Kotlin. Основний акцент був зроблений на архітектурному шаблоні MVP (Model-View-Presenter), який був обраний як основа для розробки мобільного додатка для планування справ.

Розглядаючи MVP Architecture, були детально вивчені та реалізовані його компоненти, такі як Model, View і Presenter. Це дозволило покращити організацію коду, полегшити тестування та розширення функціоналу додатку. Крім того, вибір Kotlin як мови програмування позитивно позначився на читабельності коду та продуктивності розробки.

Дослідження в сфері розробки інтерфейсів на платформі Android включало в себе вивчення якісних підходів до створення UI, використання бібліотек та інструментів, таких як Jetpack Compose та XML-верстка. Проведене порівняння та аналіз обраних інструментів дало змогу вибрати найбільш оптимальний та ефективний підхід для створення інтерфейсу додатку для користувача.

Окремо слід відзначити вибір мови для інтерфейсу, де англійська використовується для базового інтерфейсу, що сприяє глобальній взаємодії та інтеграції зі світовою спільнотою розробників, та українська - для вспоміжного інтерфейсу, спрямованого на зручність використання програми українською аудиторією.

Важливим аспектом дослідження було використання бази даних з використанням бібліотеки Room, що забезпечило зручне та ефективне зберігання даних додатку, а також підвищило швидкість доступу до них.

Висновки нашої наукової роботи відображають ретельне дослідження вибраних технологій, професійне їх використання та аналітику результатів. Розроблений мобільний додаток враховує сучасні підходи до розробки, високо зручний для користувачів і готовий до подальшого розвитку

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Дмитро Худенко, Автоматичні інструменти для планування, 20 березня 2023 [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwi0jpXLmYCDAxXDQ_EDHSzNAIYQFnoECBIQAQ&url=https%3A%2F%2Fworksection.com%2Fua%2Fblog%2Fbest-project-planning-tools.html&usg=AOvVaw2q_R7UTf8hFBipehcxvb8D&opi=89978449 (Дата звернення 10.09.2023)

2. Ляліна Н.С., Сучасний бізнес в Україні [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwj1x4fdmoCDAxWxQvEDHW5jCJ0QFnoECBAQAQ&url=http%3A%2F%2Fwww.market-infr.od.ua%2Fjournals%2F2021%2F53_2021%2F11.pdf&usg=AOvVaw28Qsh9yr0ZwIp9AiPjgMky&opi=89978449 (Дата звернення 10.10.2023)

3. Ільченко К.В., Сучасні інформаційні інтерфейси [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiyxPDjm4CDAxX8avEDHRheChAQFnoECBIQAQ&url=https%3A%2F%2Fopenarchive.nure.ua%2Fbitstreams%2F7f5f9c9f-7baa-45dd-9646-43878c706b3f%2Fdownload&usg=AOvVaw0Qmc9WL-HBuQmiZLklgTM6&opi=89978449 (Дата звернення 15.10.2023).

4. J. G., Коли виник перший комп'ютер [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiChPr2noCDAxUJQfEDHTH7BZkQFnoECBEQAQ&url=https%3A%2F%2Fdovidka.biz.ua%2Fkoli-vinik-pershiy-komp-yuter%2F&usg=AOvVaw3FirZ82xRymXYovhq5QftB&opi=89978449 (Дата звернення 10.10.2023)

5. Історія інтернету, Wikipedia [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiX3uD6noCDAxXmavEDHQN8C5sQFnoECA4QAQ&url

=https%3A%2F%2Fuk.wikipedia.org%2Fwiki%2F%25D0%2586%25D1%2581%25D1%2582%25D0%25BE%25D1%2580%25D1%2596%25D1%258F_%25D0%2586%25D0%25BD%25D1%2582%25D0%25B5%25D1%2580%25D0%25BD%25D0%25B5%25D1%2582%25D1%2583&usg=AOvVaw1LNeTYIMZ9CXGjxPzP3Mx3&opi=89978449 (Дата звернення 10.10.2023)

6. Історія смартфонів, Wikipedia [[Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiKw4GGn4CDAxUhVfEDHRS0BB8QFnoECBAQAQ&url=https%3A%2F%2Fuk.wikipedia.org%2Fwiki%2F%25D0%25A1%25D0%25BC%25D0%25B0%25D1%2580%25D1%2582%25D1%2584%25D0%25BE%25D0%25BD&usg=AOvVaw0AQfY-GRXOR3cXSNmw-5VK&opi=89978449 (Дата звернення 12.10.2023)

7. Google, Wikipedia [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjJu9DVn4CDAxVQRPEDHVOjBBEQFnoECBIQAQ&url=https%3A%2F%2Fuk.wikipedia.org%2Fwiki%2FGoogle&usg=AOvVaw00S7F8deLSWKKmftNUQhfY&opi=89978449 (Дата звернення 13.10.2023)

8. Palm OS, Wikipedia [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjgzZ-DoICDAxWHRvEDHfGjAWcQFnoECBUQAQ&url=https%3A%2F%2Fuk.wikipedia.org%2Fwiki%2FPalm_OS&usg=AOvVaw3FiiLzlo9wZyDdYoc7SfS1&opi=89978449 (Дата звернення 13.10.2023)

9. IOS, Wikipedia [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjx29G8oICDAxXSVvEDHQ6AApYQFnoECBcQAQ&url=https%3A%2F%2Fuk.wikipedia.org%2Fwiki%2FIOS&usg=AOvVaw3bk4flqf4WAkZiKjQMUlgA&opi=89978449 (Дата звернення 15.10.2023)

10. Android, Wikipedia [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&

uact=8&ved=2ahUKEwipguu_oICDAxUSVvEDHRWFApcQFnoECC4QAQ&url=https%3A%2F%2Fuk.wikipedia.org%2Fwiki%2FAndroid&usg=AOvVaw1sp2F7aBmxdyRIvVXuXb9w&opi=89978449 (Дата звернення 16.10.2023)

11. Список операційних систем , Wikipedia [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiuyOeOoYCDAxV3SvEDHa87A1EQFnoECA4QAQ&url=https%3A%2F%2Fuk.wikipedia.org%2Fwiki%2F%25D0%25A1%25D0%25BF%25D0%25B8%25D1%2581%25D0%25BE%25D0%25BA_%25D0%25BE%25D0%25BF%25D0%25B5%25D1%2580%25D0%25B0%25D1%2586%25D1%2596%25D0%25B9%25D0%25BD%25D0%25B8%25D1%2585_%25D1%2581%25D0%25B8%25D1%2581%25D1%2582%25D0%25B5%25D0%25BC&usg=AOvVaw2HB5IenyaQQnZ7loXTmQve&opi=89978449 (Дата звернення 18.10.2023)

12. Java, Oracle [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwi8x-vkoYCDAxWxQvEDHW5jCJ0QFnoECBgQAQ&url=https%3A%2F%2Fwww.oracle.com%2Fcis%2Fjava%2F&usg=AOvVaw03oHq0pyXcWDGih-rZILwL&opi=89978449 (Дата звернення 20.10.2023)

13. Github, Wikipedia [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwi61sbooYCDAxWjQ_EDHQBQAggQFnoECBoQAQ&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FGitHub&usg=AOvVaw36pWCH3bqY2GTz-KUkccuo&opi=89978449 (Дата звернення 20.10.2023)

14. API, Wikipedia [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjFr7fwoYCDAxWMVPEDHWXCCVkQFnoECAYQAQ&url=https%3A%2F%2Fuk.wikipedia.org%2Fwiki%2F%25D0%259F%25D1%2580%25D0%25B8%25D0%25BA%25D0%25BB%25D0%25B0%25D0%25B4%25D0%25BD%25D0%25B8%25D0%25B9_%25D0%25BF%25D1%2580%25D0%25BE%25D0%25B3%25D1%2580%25D0%25B0%25D0%25BC%25D0%25BD%25D

0%25B8%25D0%25B9_%25D1%2596%25D0%25BD%25D1%2582%25D0%25B5%25D1%2580%25D1%2584%25D0%25B5%25D0%25B9%25D1%2581&usg=AOvVaw0YrqzY6xzF-WXx_gM5Offe&opi=89978449 (Дата звернення 21.10.2023)

15. Kotlin, Kotlin Foundation [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiIvPX4oYCDAxUrcvEDHVgpD6oQFnoECAYQAQ&url=https%3A%2F%2Fkotlinlang.org%2F&usg=AOvVaw1Hr3700IZI3BCYi-egoGAT&opi=89978449 (Дата звернення 22.10.2023)

16. Objective-C, Wikipedia [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjOkL2NooCDAxWDavEDHQRTDIQQFnoECA8QAQ&url=https%3A%2F%2Fuk.wikipedia.org%2Fwiki%2FObjective-C&usg=AOvVaw1hPVSs1LOJGAKMxZeI7L33&opi=89978449 (Дата звернення 22.10.2023)

17. Swift, Wikipedia [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjY1beQooCDAxWnQPEDHXARDhcQFnoECAYQAQ&url=https%3A%2F%2Fuk.wikipedia.org%2Fwiki%2FSWIFT&usg=AOvVaw0dJ1x4fmf16ly54AZpON2z&opi=89978449 (Дата звернення 23.10.2023)

18. Apple, Apple.Inc [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiKm66UooCDAxXhSfEDHZ2CDFkQFnoECAcQAQ&url=https%3A%2F%2Fwww.apple.com%2Fua%2F&usg=AOvVaw2s81xmTSir-au1Jp1Jnr20&opi=89978449 (Дата звернення 24.10.2023)

19. SQLite [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjkp8q9ooCDAxXZRvEDHeO1CqMQFnoECAUQAQ&url=https%3A%2F%2Fwww.sqlite.org%2F&usg=AOvVaw2FGx1kWp6WBAJWy5IhYh3r&opi=89978449 (Дата звернення 25.10.2023)

20. SQL, Wikipedia [Електронний ресурс] – URL:

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjU2PK_ooCDAxULbvEDHZwAApYQFnoECAYQAQ&url=https%3A%2F%2Fuk.wikipedia.org%2Fwiki%2FSQL&usg=AOvVaw0xQbDJ2jmgZjozo3g2g0_Z&opi=89978449 (Дата звернення 25.10.2023)

21. MySQL, Oracle [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjShenIooCDAxVdcvEDHRdHB6YQFnoECAUQAQ&url=https%3A%2F%2Fwww.mysql.com%2F&usg=AOvVaw20c6IrMAtNC1A9NZPsDpWW&opi=89978449 (Дата звернення 26.10.2023)

22. PostgreSQL, The PostgreSQL Global Development Group [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwie0dPNooCDAxUsRPEDHQhWDhQQFnoECAUQAQ&url=https%3A%2F%2Fwww.postgresql.org%2F&usg=AOvVaw0He1mmeTUi_lhXjiRGJtzr&opi=89978449 (Дата звернення 27.10.2023)

23. RoomDB, Google for Developers [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjMwtDjooCDAxV4Q_EDHc-dAsIQFnoECAYQAQ&url=https%3A%2F%2Fdeveloper.android.com%2Ftraining%2Fdata-storage%2Froom&usg=AOvVaw3zNIzWq32_4C5U2awzrPZ9&opi=89978449 (Дата звернення 29.10.2023)

24. Microsoft Outlook, Microsoft [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjj97X_ooCDAxWwSPEDHcwjCSYQFnoECBMQAQ&url=https%3A%2F%2Fuk.wikipedia.org%2Fwiki%2FMicrosoft_Outlook&usg=AOvVaw2OGtKN8M1e5b7JbM62r9Er&opi=89978449 (Дата звернення 29.10.2023)

25. Todoist, Doist Inc. [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwj26JWUo4CDAxUfVvEDHWDRCwEQFnoECAUQAQ&u

rl=https%3A%2F%2Ftodoist.com%2Fru&usg=AOvVaw1ffAGgFHKoNzdmUjlaiuHN&opi=89978449 (Дата звернення 01.11.2023)

26. Microsoft To Do, Microsoft [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjP44qfo4CDAxWGQvEDHbksCeYQFnoECBEQAQ&url=https%3A%2F%2Fto-do.office.com%2Ftasks%2F&usg=AOvVaw0-b325nU9E1rcllc_XQLZJ&opi=89978449 (Дата звернення 02.11.2023)

27. Trello, Atlassian [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiZzvGio4CDAxUzcvEDHbbTANwQFnoECAYQAQ&url=https%3A%2F%2Ftrello.com%2Fuk&usg=AOvVaw2w9eW7Nxl6u7hHpFtEQkiE&opi=89978449 (Дата звернення 03.11.2023)

28. Google Keep, Google [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwji0dino4CDAxVpcfEDHXkgAlsQFnoECAUQAQ&url=https%3A%2F%2Fkeep.google.com%2F&usg=AOvVaw1NKNFGiEfwHk17jfPT0BM0&opi=89978449 (Дата звернення 04.11.2023)

29. Apple Reminders, Apple.Inc [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiJuZGgpYCDAxVTRfEDHVONCxYQFnoECAUQAQ&url=https%3A%2F%2Fsupport.apple.com%2Fen-us%2FHT205890&usg=AOvVaw1sNzevuJkC3G_EKBWSUQBF&opi=89978449 (Дата звернення 04.11.2023)

30. iCloud, Apple.Inc [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjqg6OqpYCDAxX1R_EDHdiSD58QFnoECAUQAQ&url=https%3A%2F%2Fwww.icloud.com%2F&usg=AOvVaw1azKdK3PJz2fmQEPwy4CX_&opi=89978449 (Дата звернення 04.11.2023)

31. Things 3 Appstore, Apple.Inc [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&

uact=8&ved=2ahUKEwiShaC2pYCDAxX1RPEDHYmIBXsQFnoECBMQAQ&url=https%3A%2F%2Fapps.apple.com%2Fru%2Fapp%2Fthings-3%2Fid904237743&usg=AOvVaw0OMwh7HAGr2jp1bHBY_ozs&opi=89978449 (Дата звернення 05.11.2023)

32. UI/UX, © All Right Reserved.Projector Mag [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjhvv3ApYCDAxXDSfEDHTAaBo8QFnoECBcQAQ&url=https%3A%2F%2Fprjctrmag.com%2Fuxui-questions&usg=AOvVaw2zx6XfVZx2xytxcosHJfTj&opi=89978449 (Дата звернення 06.11.2023)

33. Twitter [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjO4p7YpYCDAxU9QvEDHWrrAP4QFnoECAYQAQ&url=https%3A%2F%2Fuk.wikipedia.org%2Fwiki%2F%25D0%25A2%25D0%25B2%25D1%2596%25D1%2582%25D1%2582%25D0%25B5%25D1%2580&usg=AOvVaw0NxWZgYbeL7_wFLLYDOM_h&opi=89978449 (Дата звернення 06.11.2023)

34. RPC (Remote Procedure Call), Wikipedia [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjd3avopYCDAxXwbvEDHdivCesQFnoECBYQAQ&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FRemote_procedure_call&usg=AOvVaw1douwZmTrKtetRb_DgSQXa&opi=89978449 (Дата звернення 08.11.2023)

35. SOAP (Simple Object Access Protocol), Wikipedia [Електронний ресурс] – URL:https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiEjOvspYCDAxU2QvEDHVN4BKQQFnoECBcQAQ&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FSOAP&usg=AOvVaw0buZlEQ_xQSD9gc-wLuBkf&opi=89978449 (Дата звернення 10.11.2023)

36. REST, Wikipedia [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjMjaT5pYCDAxVUZ_EDHRgDAaEQFnoECBEQAw&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FREST&usg=AOvVaw2hflHIIDJhA-T9uPulrL80&opi=89978449 (Дата звернення 15.11.2023)

37. GraphQLThe GraphQL Foundation [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiG1vmBpoCDAxUxQvEDHZYlBVEQFnoECBYQAQ&url=https%3A%2F%2Fgraphql.org%2F&usg=AOvVaw0mfmkFd-vcSKFxzGwioZ4J&opi=89978449 (Дата звернення 07.11.2023)

38. PHP, The PHP Group [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiZu7uJpoCDAxU2a_EDHYaJBqEQFnoECAYQAQ&url=https%3A%2F%2Fwww.php.net%2F&usg=AOvVaw1JqmoYucRr7TaiUciQETQq&opi=89978449 (Дата звернення 17.11.2023)

39. Phyton Python Software Foundation [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjlxMCRpoCDAxVxXvEDHUl2BqAQFnoECAoQAQ&url=https%3A%2F%2Fwww.python.org%2F&usg=AOvVaw0QREvGsjwHKp2GtoYvs1JH&opi=89978449 (Дата звернення 08.11.2023)

40. Ruby [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiw6MWYpoCDAxUTX_EDHe8IDcUQFnoECBAQAQ&url=https%3A%2F%2Fwww.ruby-lang.org%2Fru%2F&usg=AOvVaw2MBjdrM6Mo90Z5I1S2Vr25&opi=89978449 (Дата звернення 08.11.2023)

41. CoreML, Apple Inc [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiWv-2mpoCDAxXGRPEDHSwxDqEQFnoECAYQAQ&url=https%3A%2F%2Fdeveloper.apple.com%2Fdocumentation%2Fcoreml&usg=AOvVaw0-4qelIRaiWp8rOBWX8RAx&opi=89978449 (Дата звернення 20.11.2023)

42. Reality Kit Overview, Apple Inc [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjYoeOzpoCDAxXTX_EDHeDmD-

UQFnoECBIQAQ&url=https%3A%2F%2Fdeveloper.apple.com%2Faugmented-reality%2Frealitykit%2F&usg=AOvVaw2fVxTso5fyPYQQPDNPexGz&opi=89978449 (Дата звернення 25.11.2023)

43. HealthKit, Apple Inc. [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjtuP67poCDAxWrR_EDHVHGAekQFnoECAYQAQ&url=https%3A%2F%2Fdeveloper.apple.com%2Fdocumentation%2Fhealthkit&usg=AOvVaw2YQ5r9QxigwcfVKwQijoV-&opi=89978449 (Дата звернення 28.11.2023)

44. Neural Network API, Apple Inc. [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiawZTGpoCDAxWzQPEDHfsBBY4QFnoECBMQAQ&url=https%3A%2F%2Fdeveloper.android.com%2Fndk%2Fguides%2Fneuralnetworks&usg=AOvVaw3aUnQAisIYa-5JfTsseWPH&opi=89978449 (Дата звернення 28.12.2023)

45. Flurry, Apple Inc. [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwj635DbpoCDAxUjX_EDHZHdA2oQFnoECBMQAQ&url=https%3A%2F%2Fwww.flurry.com%2F&usg=AOvVaw1K5ahWR0x6sGw2T3VvOFCP&opi=89978449 (Дата звернення 01.12.2023)

46. Model-View-Controller (MVC), Wikipedia [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjoz77ypoCDAxVVc_EDHZvUCiMQFnoECB4QAQ&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FModel%25E2%2580%2593view%25E2%2580%2593controller&usg=AOvVaw1wpuCUJRz1WxG51eRibnYX&opi=89978449 (Дата звернення 01.12.2023)

47. Model-View-Presenter, Wikipedia [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjsr52Ap4CDAxVMQvEDHedgChQQFnoECBUQAQ&url=https%3A%2F%2Fuk.wikipedia.org%2Fwiki%2FModel-View-Presenter&usg=AOvVaw0vKIxK-qokVoBr87PLeZSP&opi=89978449 (Дата

звернення 04.12.2023)

48. Микита Гончарук, MVVM [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjdgouGp4CDAxXZX_EDHQQuBL4QFnoECDcQAQ&url=https%3A%2F%2Fmedium.com%2Fclean-code-channel%2Fmvvm-d01c432217af&usg=AOvVaw3VKDPAY9XfLSoFpi3FhG7m&opi=89978449 (Дата звернення 05.12.2023)

49. Robert C. Martin, Clean Architecture (Uncle Bob) [Електронний ресурс] – URL:https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwi8seyPp4CDAxW4X_EDHVCMD_sQFnoECBcQAQ&url=https%3A%2F%2Fblog.cleancoder.com%2Funcle-bob%2F2012%2F08%2F13%2Fthe-clean-architecture.html&usg=AOvVaw3xu8nWOw5bn6goxZ2zxg-i&opi=89978449 (Дата звернення 06.12.2023)

50. Jetpack Compose [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjr466qp4CDAxUTefEDHThpAFAQFnoECAYQAQ&url=https%3A%2F%2Fdeveloper.android.com%2Fjetpack%2Fcompose&usg=AOvVaw2BpAQ4DMwUiKSOtVRyUczM&opi=89978449 (Дата звернення 07.12.2023)

51. XML, Wikipedia [Електронний ресурс] – URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiCpdKfp4CDAxWjYPEDHRrZBrgQFnoECAYQAQ&url=https%3A%2F%2Fuk.wikipedia.org%2Fwiki%2FXML&usg=AOvVaw3DbahYOSwrhzLoxYKTE5dy&opi=89978449 (Дата звернення 07.12.2023)

ДОДАТКИ

Додаток А – Протокол перевірки кваліфікаційної роботи

(обов'язковий)

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: **« Розробка автоматизованої системи планування і обліку завдань. Частина 2. Розробка мобільного додатку»**

Тип роботи: Магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ КСУ, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 97,7% Схожість 2,3%

Аналіз звіту подібності (відмітити потрібне)

ˇ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Володимир ДУБОВОЙ
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unichek щодо роботи.

Автор роботи _____ Назарій БЕКАС
(підпис) (прізвище, ініціали)

Керівник роботи _____ Олег КОВАЛЮК
(підпис) (прізвище, ініціали)

Додаток Б
(обов'язковий)

ВНТУ

ЗАТВЕРДЖЕНО
Т.в.о.Зав. кафедри КСУ ВНТУ,
д.т.н., доцент
_____Марія ЮХИМЧУК
"06" 10 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

Розробка автоматизованої системи планування і обліку завдань. Частина 2. Розробка мобільного додатку.»

08-33.МКР.08.00.000 ТЗ

Студент групи 2АКІТ-22М
_____Назарій БЕКАС
Підпис *Ім'я ПРІЗВИЩЕ*

Керівник к.т.н., доцент
_____Олег КОВАЛЮК
Підпис *Ім'я ПРІЗВИЩЕ*

Вінниця 2023

1. Назва та галузь застосування

1.1. Назва – Розробка автоматизованої системи планування і обліку завдань.

Частина 2 Розробка мобільного додатку**.**

1.2. Галузь застосування – Планування і облік завданнь.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ №247 від “18” вересня 2023 року.

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є підвищення ефективності планування та виконання завдань.

4. Джерела розробки.

Магістерська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1. "Kotlin for Android Developers" - автор Antonio Leiva
2. "Programming Kotlin for Android Development" - автор Peter Sommerhoff
3. "Kotlin Programming By Example" - автор Iyanu Aboyeji5.

5. Вимоги до розробки.

5.1. Перелік головних функцій:

- Запис завдання;
- Редагування завдання;
- Відмітки виконання завдання;
- Налаштування для зручності інтерфейсу.

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- OS Android;
- Середовище для розробки і тесування Android Studio;

5.2.2. Умови експлуатації системи:

- робота на мобільних та веб-додатках;

- можливість цілодобового функціонування системи;
- дані оновлюються і є актуальними.

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

1. Аналіз методів планування та обліку завдань 03.10.2023 р.
2. Розробка структури мобільного додатку 19.10.2023р
3. Розробка програмного забезпечення 14.11. 2023 р.
4. Тестування програмного забезпечення програми 24.11.2023 р.

6.2 Графічні матеріали:

1. Розробка моделі системи 21.10.2023
2. Розробка моделі бази даних системи 24.11.2023 р.
3. Тестування програмного забезпечення 26.11.2023 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до 28.11. 2023 р.

7.2. Атестація МКР здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до 11.12. 2023 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістерської кваліфікаційної роботи провести до 20.12. 2023 р.

Додаток В – Лістинг головної сторінки

(довідковий)

```
package com.example.todolist.presentation

import android.os.Bundle
import androidx.activity.viewModels
import androidx.navigation.findNavController
import androidx.navigation.ui.AppBarConfiguration
import androidx.navigation.ui.navigateUp
import androidx.navigation.ui.setupActionBarWithNavController
import androidx.navigation.ui.setupWithNavController
import androidx.appcompat.app.AppCompatActivity
import androidx.databinding.DataBindingUtil
import com.example.todolist.R
import com.example.todolist.databinding.ActivityMainBinding
import dagger.hilt.android.AndroidEntryPoint

@AndroidEntryPoint
class MainActivity : AppCompatActivity() {
    private lateinit var appBarConfiguration: AppBarConfiguration
    private lateinit var binding: ActivityMainBinding
    val viewModel : MainActivityViewModel by viewModels()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = DataBindingUtil.setContentView(this,
R.layout.activity_main)
        setSupportActionBar(binding.appBarMain.toolbar)
        val navController =
findNavController(R.id.nav_host_fragment_content_main)
        appBarConfiguration = AppBarConfiguration(
            setOf(
                R.id.base_fragment, R.id.settings_fragment,
R.id.completed_tasks_fragment
            ), binding.drawerLayout
        )
        setupActionBarWithNavController(navController,
appBarConfiguration)
        binding.navView.setupWithNavController(navController)

    }

    override fun onSupportNavigateUp(): Boolean {
        val navController =
findNavController(R.id.nav_host_fragment_content_main)
        return navController.navigateUp(appBarConfiguration) ||
super.onSupportNavigateUp()
    }

}
```

Додаток Г – Лістинг програми Main Activity View Model

(довідковий)

```
import android.util.Log
import androidx.lifecycle.LiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.todolist.data.model.CategoryInfo
import com.example.todolist.data.model.NoOfTaskForEachCategory
import com.example.todolist.data.model.TaskCategoryInfo
import com.example.todolist.data.model.TaskInfo
import com.example.todolist.domain.TaskCategoryRepository
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.*
import kotlinx.coroutines.Dispatchers.IO
import java.util.*
import javax.inject.Inject

@HiltViewModel
class MainActivityViewModel @Inject constructor (private val
repository: TaskCategoryRepository) : ViewModel() {

    fun updateTaskStatus(task: TaskInfo) {
        viewModelScope.launch(IO) {
            repository.updateTaskStatus(task)
        }
    }

    fun deleteTask(task: TaskInfo) {
        viewModelScope.launch(IO) {
            repository.deleteTask(task)
        }
    }

    fun insertTaskAndCategory(taskInfo: TaskInfo, categoryInfo:
CategoryInfo) {
        viewModelScope.launch(IO) {
            repository.insertTaskAndCategory(taskInfo,
categoryInfo)
        }
    }

    fun updateTaskAndAddCategory(taskInfo: TaskInfo, categoryInfo:
CategoryInfo){
        viewModelScope.launch(IO) {
            repository.updateTaskAndAddCategory(taskInfo,
categoryInfo)
        }
    }

    fun updateTaskAndAddDeleteCategory(
        taskInfo: TaskInfo,
        categoryInfoAdd: CategoryInfo,
```

```
        categoryInfoDelete: CategoryInfo
    ) {
        viewModelScope.launch(IO) {
            repository.updateTaskAndAddDeleteCategory(taskInfo,
categoryInfoAdd, categoryInfoDelete)
        }
    }

    fun deleteTaskAndCategory(taskInfo: TaskInfo, categoryInfo:
CategoryInfo) {
        viewModelScope.launch(IO) {
            repository.deleteTaskAndCategory(taskInfo,
categoryInfo)
        }
    }

    fun getUncompletedTask(): LiveData<List<TaskCategoryInfo>> {
        return repository.getUncompletedTask()
    }

    fun getCompletedTask(): LiveData<List<TaskCategoryInfo>> {
        return repository.getCompletedTask()
    }

    fun getUncompletedTaskOfCategory(category: String):
LiveData<List<TaskCategoryInfo>> {
        return repository.getUncompletedTaskOfCategory(category)
    }

    fun getCompletedTaskOfCategory(category: String):
LiveData<List<TaskCategoryInfo>> {
        return repository.getCompletedTaskOfCategory(category)
    }

    fun getNoOfTaskForEachCategory():
LiveData<List<NoOfTaskForEachCategory>>{
        return repository.getNoOfTaskForEachCategory()
    }

    fun getCategories(): LiveData<List<CategoryInfo>>  {
        return repository.getCategories()
    }

    suspend fun getCountOfCategory(category: String): Int{
        var count: Int
        coroutineScope() {
            count = withContext(IO) {
repository.getCountOfCategory(category) }
        }
        return count
    }

    fun getAlarms(currentTime : Date){
        CoroutineScope(Dispatchers.Main).launch {
```

```
            val list = repository.getActiveAlarms(currentTime)
            Log.d("DATA", list.toString())
        }
    }
}
```

Додаток Ґ – Лістинг програмної частини отримання сповіщення

(довідковий)

```
@AndroidEntryPoint
class AlarmReceiver : BroadcastReceiver() {
    private var notificationManager: NotificationManagerCompat? = null
    @Inject
    lateinit var sharedPreferences: SharedPreferences

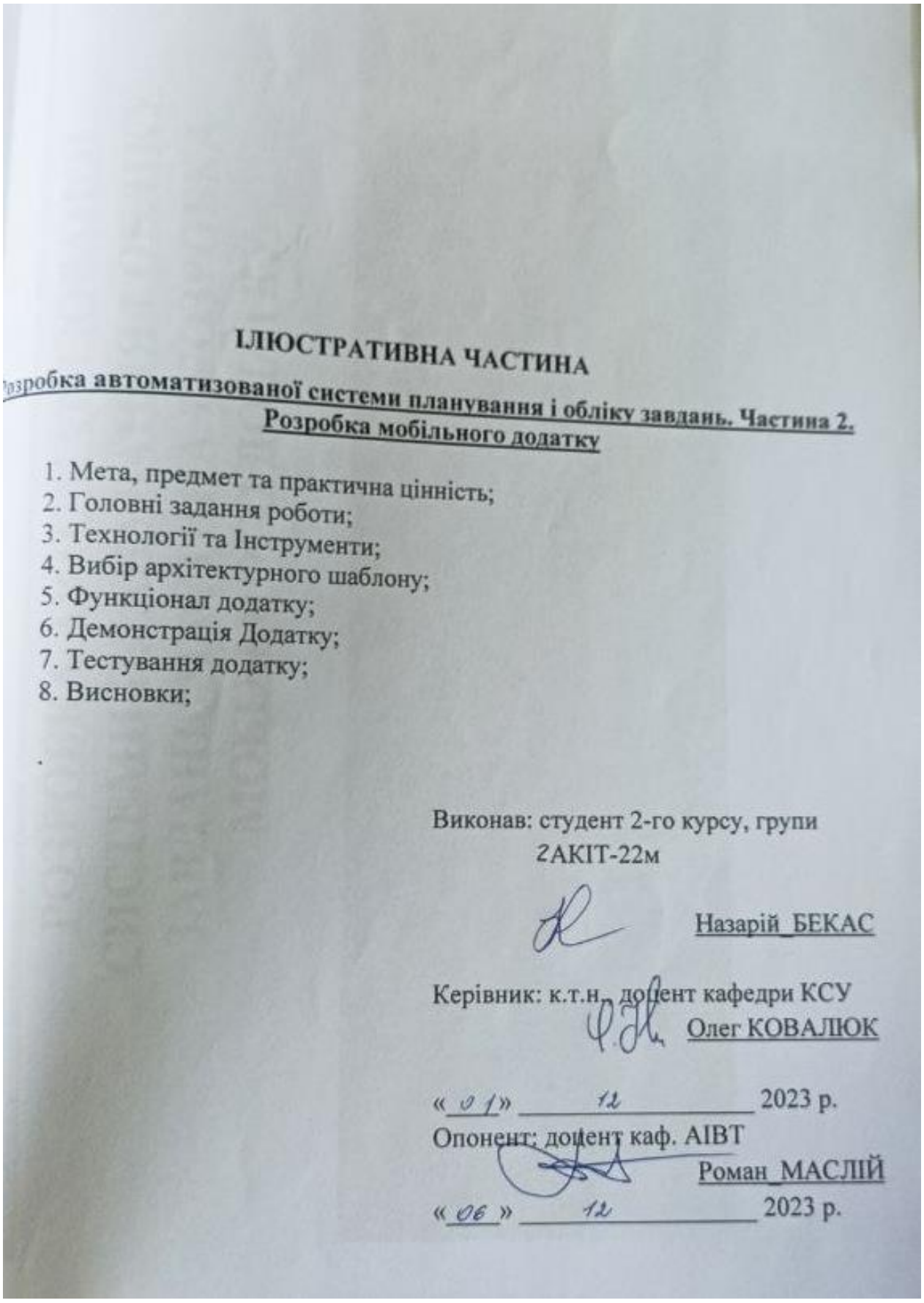
    override fun onReceive(p0: Context?, p1: Intent?) {
        val taskInfo = p1?.getSerializableExtra("task_info") as? TaskInfo
        if(sharedPreferences.getBoolean(taskInfo?.priority.toString(),
true)){
            val tapResultIntent = Intent(p0, MainActivity::class.java)
            tapResultIntent.flags = Intent.FLAG_ACTIVITY_SINGLE_TOP
            val pendingIntent: PendingIntent = getActivity(
p0,0,tapResultIntent,FLAG_UPDATE_CURRENT or FLAG_IMMUTABLE)

            val intent1 = Intent(p0,
OnCompletedBroadcastReceiver::class.java).apply {
                putExtra("task_info", taskInfo)
            }
            val pendingIntent1: PendingIntent? =
                taskInfo?.let { getBroadcast(p0,
it.id,intent1,FLAG_UPDATE_CURRENT or FLAG_IMMUTABLE) }
            val action1 : NotificationCompat.Action =
NotificationCompat.Action.Builder(0,"Completed",pendingIntent1).build()

            val notification = p0?.let {
                NotificationCompat.Builder(it, "to_do_list")
                    .setContentTitle("Task Reminder")
                    .setContentText(taskInfo?.description)
                    .setSmallIcon(R.mipmap.ic_launcher)
                    .setAutoCancel(true)
                    .setPriority(NotificationCompat.PRIORITY_HIGH)
                    .setContentIntent(pendingIntent)
                    .addAction(action1)
                    .build()
            }
            notificationManager = p0?.let {
NotificationManagerCompat.from(it) }
            notification?.let { taskInfo?.let { it1 ->
notificationManager?.notify(it1.id, it) } }
        }
    }
}
```

Додаток Д – Ілюстративна частина

(обов'язковий)



ІЛЮСТРАТИВНА ЧАСТИНА

Розробка автоматизованої системи планування і обліку завдань. Частина 2. Розробка мобільного додатку

1. Мета, предмет та практична цінність;
2. Головні задання роботи;
3. Технології та Інструменти;
4. Вибір архітектурного шаблону;
5. Функціонал додатку;
6. Демонстрація Додатку;
7. Тестування додатку;
8. Висновки;

.

Виконав: студент 2-го курсу, групи
ZАКІТ-22м

Назарій_БЕКАС

Керівник: к.т.н., доцент кафедри КСУ
Олег КОВАЛЮК

«01» 12 2023 р.

Опонент: доцент каф. АІВТ
Роман_МАСЛІЙ

«06» 12 2023 р.

РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ ПЛАНУВАННЯ І ОБЛІКУ ЗАВДАНЬ. ЧАСТИНА 2.РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

СТУДЕНТ ГРУПИ 2АКІТ-22М

НАЗАРІЙ БЕКАС

КЕРІВНИК РОБОТИ

К.Т.Н., ДОЦЕНТ КАФ. КСУ

КОВАЛЮК О.О.

МЕТА, ПРЕДМЕТ ТА ПРАКТИЧНА ЦІННІСТЬ

- **Метою дослідження** є поліпшення ефективності та продуктивності управління завданнями в організаціях або окремо взятих людей в повсякденному житті.
- **Предметом дослідження** є автоматизація планування та обліку завдань в сучасних організаціях, а також методи та засоби їх автоматизації з метою підвищення ефективності та організації робочого процесу.
- **Практичною цінністю** є розробка ефективних алгоритмів та програмного забезпечення для організації та обліку власних завдань і планів у мобільному додатку, що сприяє зручному упорядкуванню та надає можливість підбити підсумки успішності виконання завдань.

ГОЛОВНІ ЗАВДАННЯ РОБОТИ

- Завдання роботи:
  1. Аналіз існуючих додатків на ринку;
  2. Розробка структури додатку;
  3. Розробка програмного забезпечення;
  4. Тестування програмного забезпечення.

ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ

android studio

Kotlin

ВИБІР АРХІТЕКТУРНОГО ШАБЛОНУ

В розробці програмного забезпечення для мобільних додатків на Kotlin використовують архітектурні шаблони для організації коду та взаємодії компонентів. Приклади:

1. MVP (Model-View-Presenter): Розділяє додаток на Модель, Вид та Презентер, що полегшує тестування та зберігає чистоту коду.

2. MVVM (Model-View-ViewModel): Організовує код за допомогою Моделі, Виду та ViewModel, що спрощує управління станом та відділення бізнес-логіки від інтерфейсу користувача.

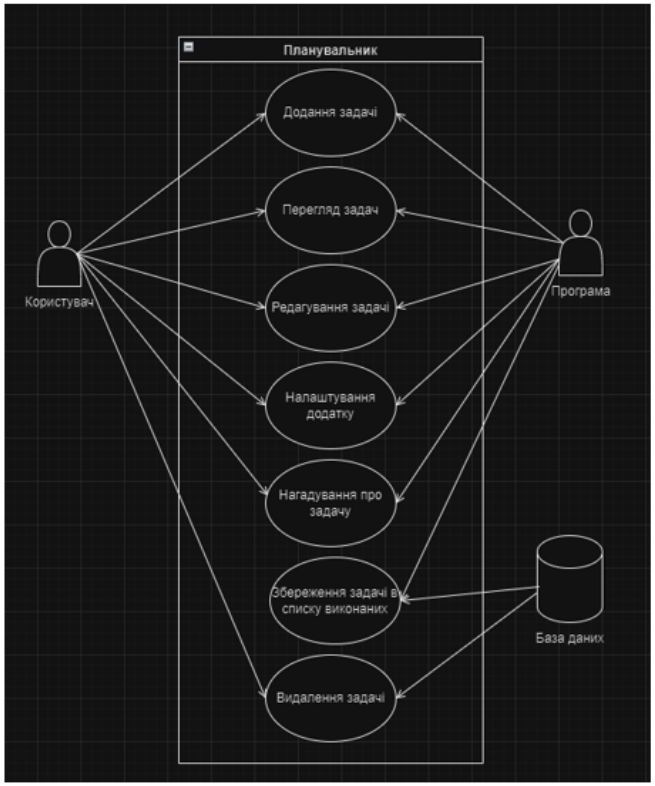
3. Clean Architecture:Розділяє додаток на рівні внутрішньої чистоти, забезпечуючи використання зовнішніх та внутрішніх шарів.

4. Redux Architecture:Використовує принципи одностороннього потоку даних для управління станом додатку.

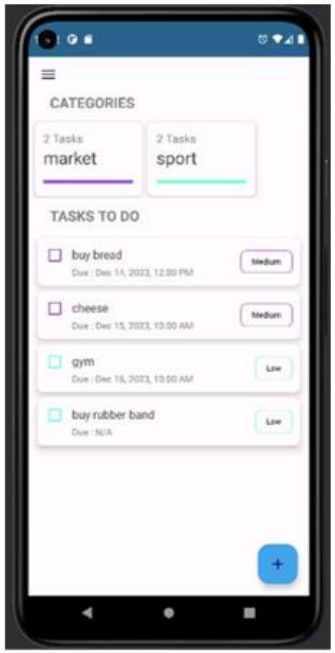
Алгоритм роботи MVP

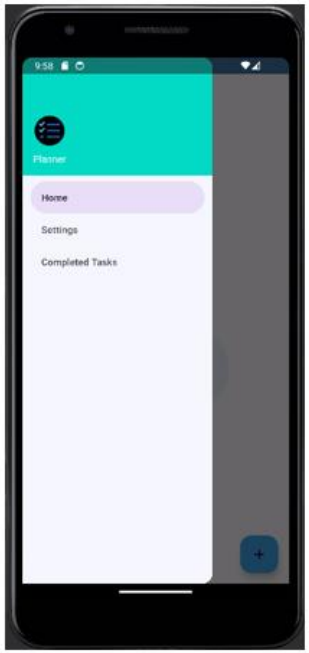
ФУНКЦІОНАЛ ДОДАТКУ

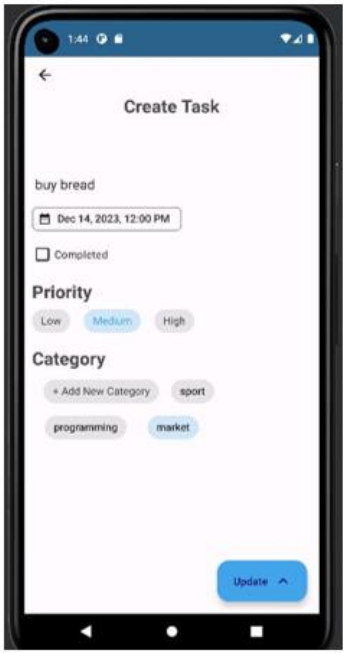
- Додання задачі включає: створення категорії, вказання дедлайну (опціонально), вказання пріоритетності задачі;
- Налаштування додатку включає: Налаштування сповіщень відповідно пріоритетності, зміна теми додатку, та посилання на оцінку в Play Market(дана функція буде додана на релізі)

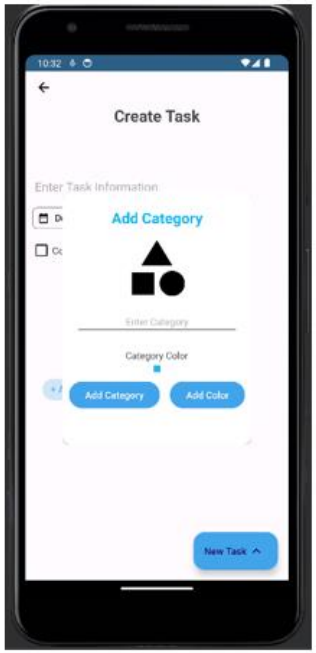


ДЕМОНСТРАЦІЯ ДОДАТКУ

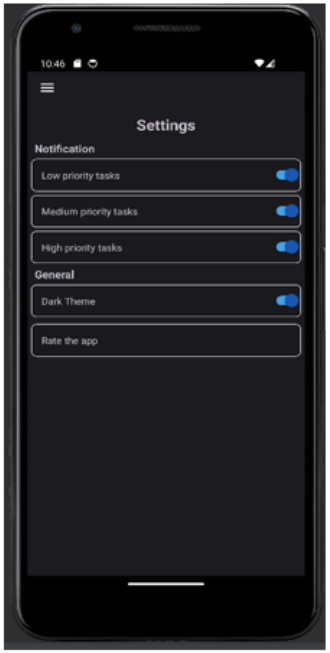


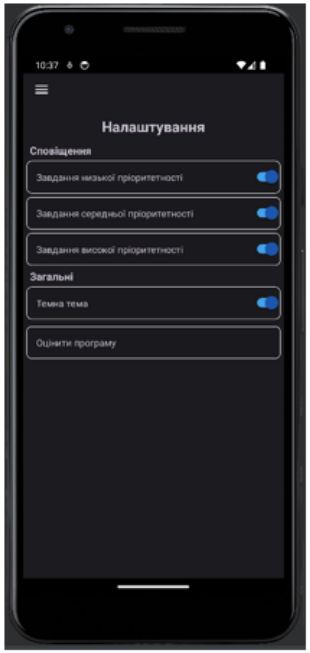


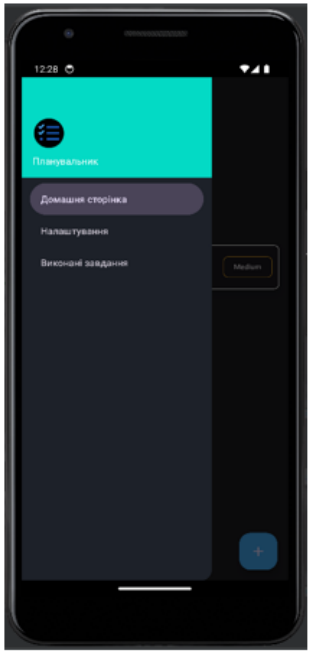


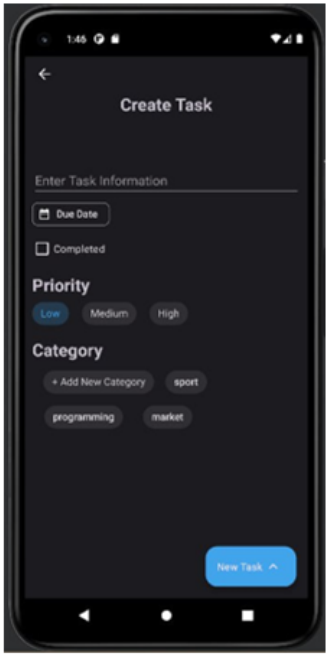


ДЕМОНСТРАЦІЯ ДОДАТКУ

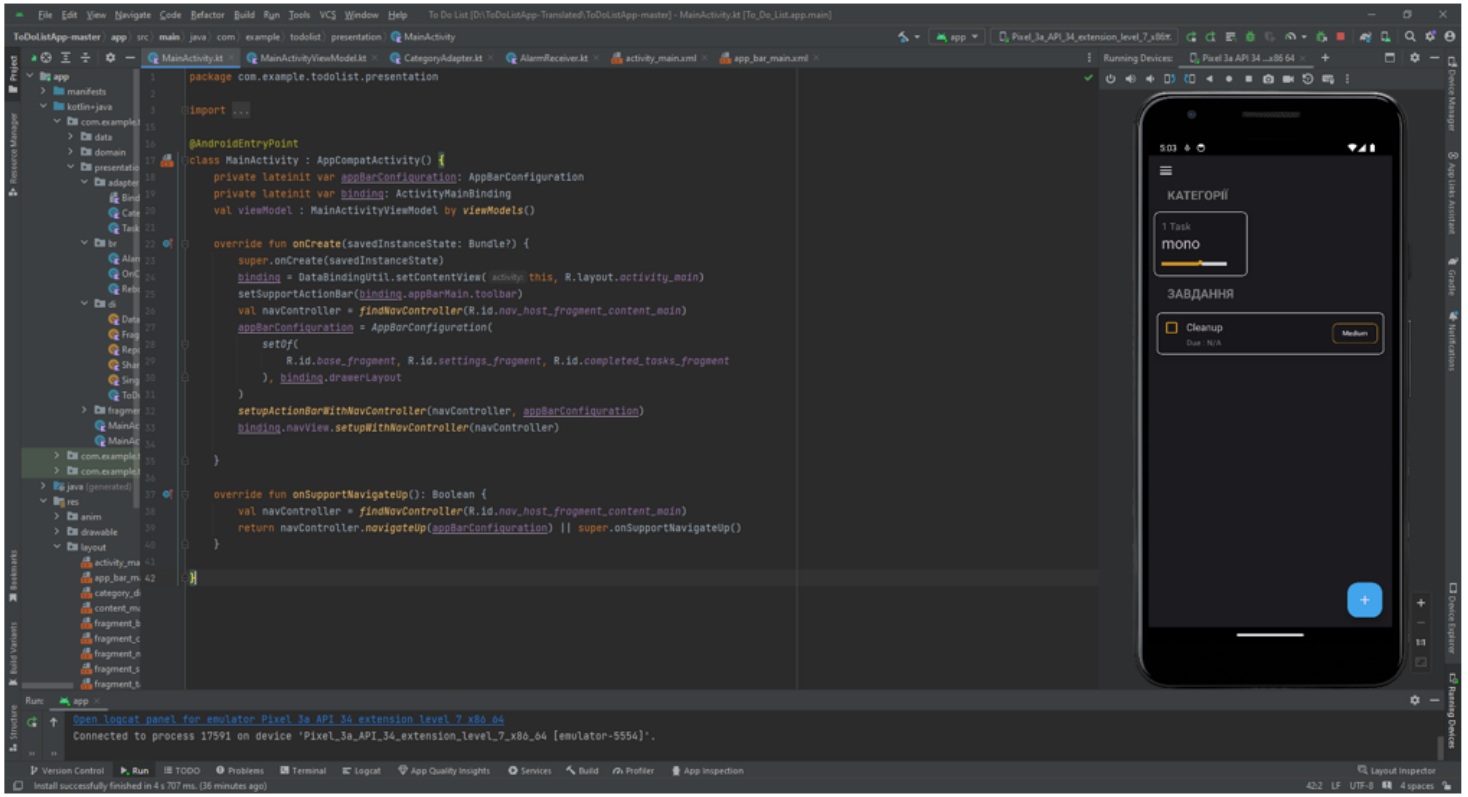








ТЕСТУВАННЯ ДОДАТКУ



ВИСНОВКИ

- У роботі було проведено обширне дослідження та розробка мобільного додатку для планування та організації справ. Основні висновки та досягнення включають:
- Ретельно вивчено та обрано архітектурний шаблон MVP (Model-View-Presenter) для реалізації додатку. Цей вибір зумовлений його ефективністю та зручністю для розширення функціоналу.
- Застосовано сучасні технології, такі як Kotlin та XML для програмування логіки додатку та розмітки інтерфейсу відповідно.
- Використовувалася бібліотека Room для роботи з локальною базою даних, що забезпечило надійне та швидке зберігання інформації.
- Забезпечено можливість використання додатку як для українськомовних, так і для англомовних користувачів, що робить його більш доступним та зручним.
- Розроблено та реалізовано різноманітний функціонал, включаючи сортування списків, введення та відображення деталей справ, аналіз продуктивності користувача та інше.
- Загальний внесок у розробку мобільного застосунку "Планувальник" визначається обґрунтованим вибором технологій, реалізацією важливого функціоналу та забезпеченням зручного інтерфейсу для користувачів.

ДЯКУЮ ЗА УВАГУ!