КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»
(шифр і назва спеціальності)

«Прикладне програмування»
(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Мобільний застосунок із планування бізнес-справ»

Виконав ____ __
(Підпис)
Тоноян Данило Сергійович
(прізвище, ім'я, по батькові)

Керівник Зайцев Євген Олександрович
(прізвище, ім'я, по батькові)
_____ _____
(Резолюція «До захисту»)

Попередній захист:

_____

(Висновок: "До захисту в екзаменаційній комісії ")

Завідувач кафедри _____ Плескач В.Л.

(Підпис) (Прізвище, ініціали) (Дата)

Засвідчую, що у цьому курсовому проекті немає запозичень з праць інших авторів без відповідних посилань. Унікальність тексту 99 %

Студент _____ __
(підпис)

Київ – 2022

Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

Назва теми: «Мобільний застосунок із планування бізнес-справ»

Освітня програма: Прикладне програмування
Спеціальність: Комп'ютерні науки

ПІБ	Підпис
Тоноян Данило Сергійович	

Назва роботи українською та англійською мовами

Мобільний застосунок із планування бізнес-справ "Business-project planner" mobile application

Мета бакалаврської роботи, завдання

Мета бакалаврської роботи: Створення простого для використання мобільного застосунку на базі платформи андроїд для відстеження процесу виконання проектів План роботи: 1. Аналітичний огляд сучасних підходів до розробки та впровадження мобільних додатків на системі андроїд 2. Аналіз існуючих варіантів і вибір програмних засобів для самої реалізації андроїд додатків 3. Програмна реалізація додатку для андроїд

ПІБ, ступінь, звання наукового керівника роботи:

Зайцев Євген Олександрович, д.т.н., с.н.с., доцент кафедри прикладних інформаційних систем

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

№з/п	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	09.10.2021	Виконано
2.	Видача завдання кваліфікаційної роботи бакалавра	19.10.2021	Виконано
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	21.10.2021	Виконано
4.	Затвердження плану кваліфікаційної роботи бакалавра	25.10.2022	Виконано
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	01.11.2022	Виконано
6.	Підготовка і подання науковому керівнику першого варіанту I розділу роботи	21.12.2022	Виконано
7.	Підготовка і подання науковому керівнику першого варіанту II розділу роботи	31.01.2022	Виконано
8.	Підготовка і подання науковому керівнику першого варіанту III розділу роботи	30.03.2022	Виконано
9.	Подання роботи у першому варіанті	28.04.2022	Виконано
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	03.05.2022	Виконано
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**23.05.2022**	Виконано
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	27.05.2022	Виконано
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	10.06.2022	Виконано
14.	Захист кваліфікаційної роботи бакалавра	22.06.2022 23.06.2022	

Здобувач вищої освіти _____

(підпис)

Керівник _____ (підпис)

ВІДОМІСТЬ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1 сторінка
Календарний план дипломної роботи	1 сторінка
Відомість дипломної роботи	1 сторінка
Анотація	1 сторінка
Анотація (іноземною мовою-англійською)	1 сторінка
Зміст	1 сторінка
Вступ	1 сторінка
Розділ 1	6 сторінок
Розділ 2	13 сторінок
Розділ 3	26 сторінок
Висновки	1 сторінка
Перелік використаних джерел	2 сторінки

<table>
<tr><td></td><td></td><td></td><td></td><td colspan="3" rowspan="2">ДП XXXX 00.000.00</td></tr>
<tr><td></td><td>ПІБ</td><td>Підп.</td><td>Дата</td></tr>
<tr><td>Розробн.</td><td>Тоноян Д.С.</td><td></td><td></td><td rowspan="4">Відомість дипломної роботи</td><td>Лист</td><td>Листів</td></tr>
<tr><td>Керівн.</td><td>Зайцев Є.О.</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Н/контр.</td><td>Базилюк А.М.</td><td></td><td></td><td colspan="2" rowspan="2"></td></tr>
<tr><td>Зав.каф.</td><td>Плескач В.Л.</td><td></td><td></td></tr>
</table>

Анотація

Дипломна робота складається із 59 с., 32 рис., 2 табл., 16 дж., 1 дод.

Метою дипломної роботи є оптимізації процесу ведення календарних планів бізнес-проектів на основі створеного мобільного застосунку для **Для досягнення поставленої мети потрібно вирішити такі завдання:**

- Створити і описати концептуальну модель мобільного застосунку;
- Визначити роль та перспективи мобільних застосунків для керування проектами на базі андроїд у сучасному інформаційному суспільстві.
- Створити проекцію та реалізувати програмно прототип мобільного застосунку для керування бізнес-планами за допомогою Android studio.

Об'єктом дослідження є процес оптимізації ведення календарних планів бізнес-проектів**.**

Предметом дослідження є програмно-технічні, організаційні засади, принципи, підходи щодо реалізації мобільного застосунку за допомогою засобів Android studio, та призначених для організації ефективного керування календарних планів бізнес-проектів.

Методи дослідження. Моделі, методи та засоби аналізу і синтезу систем, порівняння, описовий метод, UML-моделювання, теорії управління.

Ключові слова: мобільний застосунок, Android studio, керування справами.

Annotation

Thesis consists of 59 pages, 32 pictures, 2 tables, 16 sources, 1 complement.

The purpose of the thesis is to create a mobile application to optimize the process of maintaining business project schedules.

To achieve this goal you need to solve the following tasks:

- Create and describe a conceptual model of a mobile application;
- Identify the role and prospects of mobile applications for managing Android-based projects in the modern information society.
- Create a projection and implement a software prototype of a mobile application for managing business plans using Android studio.

The object of research is the process of optimizing the maintenance of business project schedules.

The subject of the study is software and hardware organizational problems, principles and approaches to the implementation of mobile applications using Android studio, and designed to effectively manage the calendar plans of business projects.

Research methods. Models, methods and means of analysis and synthesis of systems, comparison, descriptive method, UML modeling, control theory.

Keywords: mobile application, Android studio, case management.

ЗМІСТ

ВСТУП

У XXI столітті доволі складно уявити людське життя без якихось складно технологічних пристроїв, у кожного в кишені є смартфон, для робочих процесів використовують планшети чи ноутбуки, також вони допомагають підтримувати зв’язок з різними кутками нашої планети, навіть у скрутні часи. Також приймають участь у розвитку простору потоку інформації.

Кожного дня IT сфера може змінюватись, і кардинально відрізнятись від учорашньої версії. Мобільні застосунки стають все більш різноманітними та гнучкими, на сьогоднішній день навіть у ваш холодильник чи мікрохвильову піч вбудована система Android. А телефон спроможний не лише на дзвінки та фотографії, але також відкриває доступ до інтернет ресурсів, та до безперервного потоку інформації.

Саме завдяки технологіям, які сьогодні у нас існують, ми маємо змогу полегшити власне життя. Інформаційний потік та великі бази даних з кожним день стають все важливішими ресурсами, а їх організація потребує корректного відображення, аби була змога легко скористатись цими ресурсами. Тому створення мобільного застосунку для розподілення та спостереження за процесами власного проекту буде доречним у сьогоденні. Оскільки наявність подібного простого додатку може стати у нагоді кожній людині.

На сьогодні, самою поширеною та популярною ОС для мобільних пристроїв є операційна система Андроїд. Саме Андроїд має широкий спектр підтримки засобів для створення додатків та велику кількість мобільних девайсів. Головною перевагою Андроїд системи, є безкоштовний доступ до засобів для розробки, оскільки система IOS потребує витрат.

Створення цього мобільного застосунку, за допомогою інтегрованої середи Android Studio, виконується з ціллю полегшення відстеження

окремих задач, які виконуються для досягнення мети у единому проекті. Саме Android Studio надає повний інструментарій та бібліотеки для реалізації подібного застосунку.

Актуальність дослідження.

Світу вже давно невідоме старе визначення для мобільного пристрою, адже вони є не лише пристроями для зв'язку на сьогоднішній день, вони також представляють собою багатофункціональні гаджети, без яких вже неможливо уявити повсякденне життя. Існує багато різноманітних моделей з унікальними функціями і запропонованими сервісами, які також залежать від операційної системи, яка встановлена на смартфон, найбільш відомим їх представником є Android. А свою популярність вона отримала за рахунок доступності інструментарію для створення та підтримки застосунків для розробників.

Тому маємо актуальним є розробка та вдосконалення програмних застосунків які дозволяють реалізувати ефективність рішення для зручного відстежування виконання окремих задач та завдань під час роботи над бізнес проектом.

РОЗДІЛ 1
Вивчення та огляд існуючих підходів до проектування мобільних застосунків

1.1 Вплив мобільних пристроїв на сучасне життя

Сьогодні ми живемо у епоху стрімкого розвитку інформаційних технологій та різного виду технологічних пристроїв. У житті кожної людини не буває і дня без використання мобільного пристрою, оскільки на сьогоднішній день він представляє собою багатофункціональний портативний пристрій. Для роботоздатності такого пристрою було створено спеціальні операційні системи, які надавали можливості та спеціальний інструментарій для створення да підтримки різного роду мобільних застосунків. Одним із представників є операційна система Android. Вона представляє собою досить просту портативну ОС, яку використовували для функціонування мобільних пристроїв, а на сьогодні її також встановлюють на телевізори, смарт годинники та деякі побутові прилади.

Однією із переваг ОС Android перед іншими, є те, що власне вихідний код публічно розміщений, і будь хто має можливість відредагувати цей код, змінити функціонал, і також опублікувати його для загального користування, чого інші ОС не можуть запропонувати.

Першочерговою метою для використання ОС Android було встановлення її на камери, для зручного конекту з персональним комп’ютером. Створена компанією “Android, Inc”, пізніше придбана компанією Google. Продовжуючи підтримку придбаного продукту, компанія Google постійно поновлювала поточний продукти і розробляла нові версії для покращення роботи. Завдяки доступності операційної системи, є можливість зміни вихідного коду для кращої сумісності з мобільним пристроєм. Цей крок позитивно вплинув на поширення та зростання популярності операційної системи. На сьогоднішній день вже

існує багато версій Android з різним запропонованим функціоналом, інтерфейсом, та багатим вибором пристроїв, які підтримують ОС Android.

1.2 Популярність мобільних застосунків на Android

У минулому, факт того, що у когось був пристрій для бездротового зв’язку здавалось як щось незвичайне і дуже дороге, але сьогодні таке враження виникає, коли людина не має сучасний смартфон у кишені.

З кожним днем зростає різноманіття мобільних пристроїв на ОС Android, також зростає і популярність, а за цим слідує і зростання актуальності питання розробки мобільних застосунків для цієї операційної системи.

Рисунок 1.1 – Частка ринку операційних систем у світі

Дивлячись на статистику (рисунок 1.1), можна дійти до висновку що ринок ОС Android займає значний процент, у порівнянні з іншими ОС.

Рисунок 1.2 – Частка ринку операційних систем в Україні

Якщо подивитись на статистику в Україні (рисунок 1.2), то числа будуть в перевагу ОС Windows, але з кожним роком Android наздоганяє і через 2 роки вони зрівняються, чи навіть лідерство захопить ОС Android.

Завдяки виконаному аналізу, маємо остатучну відповідь на питання актуальності розробки мобільного застосунку на ОС Android.

1.3 Аналіз наявних версій операційної системи

Від самого початку і до сьогодні продовжують працювати над покращенням та розширенням функціоналу ОС Android.

Існуючі версії які підтримують:

- Ice Cream Sandwich (ver. 4.0)
- Jelly Bean (ver. 4.3)
- KitKat (ver. 4.4)
- Lollipop (ver.5.0)
- Marshmallow (ver 6.0)
- Nougat (ver. 7.0)
- Oreo (ver. 8.0)
- Pie (ver. 9.0)
- Q (ver, 10.0)
- R (ver, 11.0)
- Q (ver, 12.0)

Рисунок 1.3 – Популярність версій Android у світі

Кожна нова версія заміняє попередню по популярності. На сьогоднішній день найбільш популярними, як не дивно, є останні версії операційної системи (рисунок 1.3).

1.4 Альтернативні операційні системи

Окрім Android, існує ще декілька операційних систем, які мають свого користувача, але встигла закріпитись як доволі сильний конкурент змогла лише операційна система від Apple - IOS.

Список використовуваних операційних систем:

- Android
- IOS
- Samsung
- KaiOS
- Windows
- Nokia
- BlackBerry
- Linux

Спираючись на статистику ринок мобільних пристроїв на базі ОС Android більш розповсюджений, як в Україні так і у світі, ніж Apple IOS.

Android - представляє собою портативну операційну систему для мобільних пристроїв, смарт годинників, телевізорів та іншого. Відкритість вихідного коду велика перевага для цієї ОС, що дозволяє розробникам повністю його редагувати та змінювати.

IOS - операційна система що була розроблена компанією Apple та є прямим конкурентом ОС Android. Головною відмінністю є те, що розробники зробили ОС с закритим вихідним кодом, що робить процес розробки більш складним. Створений з однотипним GUI для усіх застосунків. Відмічається своїми перевагами, такими як більш швидкий відгук при взаємодії з пристроєм, плавні анімації, та регулярне покращення ос, її оновлення та активна підтримка.

Mobile LINUX - це використання операційної системи на базі Linux для портативних мобільних пристроїв, де як основний елемент інтерфейсу для людского управління виступає сенсорний екран. Основними їх представниками виступають мобільни пристрої та планшети, також сюди входять деякі персональні цифрові помічники, медіаплеєри в яких також використовується сенсорна панель.

Ця операційна система досить недавнім доповненням до списку місць, де використовують Linux, а першою концепцією мобільної операційної системи стала Android що розробляється Google. Хоча інші розробники тоді спробували наслідувати приклад Ubuntu Touch, але широкий, по-справжньому сприятливий, розвиток повністю безкоштовних мобільних операційних систем Linux відбувся тільки наприкінці 2010 року, коли багато різних невеликих компаній розпочали власні проекти, де велась розробка смартфонів з відкритим кодом.

Windows Phone - на сьогоднішній день не є активною операційною системою та отримала своє останнє оновлення ще у 2015 році. Однм із недоліків є те, що пристрої на цій операційній системі потребували наявність трьох фізичних кнопок. Данна операційна система в цілому досить незручна для роботи з нею, не кажучи вже про використання. Якість роботи сильно залежить від внутрішнього заліза, досить складний та трудомісткий процес внесення змін у ПО.

KaiOS - операційна система що створена спеціально для клавішних телефонів. Має підтримку 4G, WiFI, LTE та GPS. Надає змогу використовувати мобільні додатки, що були побудовані на базі HTML5. Але головним недоліком є неможливість гідно конкурувати з лідерами таблиці.

Висновки до розділу

В першому розділі було розглянуто основні поняття мобільних застосунків, системи андроїд та її конкурентів. Тому є можливість зробити висновки що:

- Найоптимальнішою системою для розробки мобільного застосунку є ОС Android;
- ОС Android є найпопулярнішою операційною системою для використання;
- Для збільшення кількості користувачів застосунку, рекомендується розробляти його на більш ранній версії операційної системи ніж найбільш використовувана.

РОЗДІЛ 2
Аналіз можливих архітектурних рішень та вибір програмних засобів для реалізації прототипу мобільного застосунку

2.1 Що собою представляє мобільний застосунок

Як ми раніше зазначили, смартфон – це сучасний пристрій, що надає доступ до великої кількості мобільних додатків, та великому обсягу інформації. Але що саме означає мобільний застосунок?

Мобільний застосунок представляє собою певну програму, яка була створена та налаштована для використання на мобільному пристрої чи смартфоні.

На сьогоднішній день існує незліченна кількість мобільних додатків. Головним пунктом в історії появи мобільних застосунків стала поява сенсорних екранів. Першим екземпляром мобільних застосунків стали заздалегідь вбудовані розробниками, де кожний застосунок виконував власну функцію.

Першою ОС, яка надавала доступ до інструментарію для розробників, була Symbian. Але ця система не отримала популярності, оскільки розробка для цієї операційної системи була дуже складною, а також були певні обмеження, які були у тогочасних пристроїв. Єдиною мовою для створення застосунків, була мова C++, яка в свою чергу сама по собі була складною для вивчення.

Паралельно з цим, щодня зростала популярність ринку застосунків написаних мовою Java. Освоєння інструментарію для створення цих додатків займала набагато менше часу, і водночас мала зручну підтримку з Android та WindowsMobile.

2.2 Java

Java - це мова програмування і водночас платформа для вичислень. На сьогоднішній день велика кількість комп’ютерних та мобільних застосунків створені з використанням мови Java.

Завдяки сильній перевазі того, що Java є безкоштовною та досить багатозадачною мовою, тому в неї є декілька шляхів використання:

Створення штучного інтелекту - за рахунок стабільності даної мови є найкращим варіантом для цього напряму.

Робота з великою кількістю даних - також мова Java часто використовується для створення механізмів обробки даних. Зручний інструментарій для роботи з великим об’ємом даних у реальному часі.

Game development - велика кількість знайомих комп’ютерних та мобільних ігор були створені з використанням технологій Java. Навіть сьогодні цю мову використовують для реалізації віртуальної реальності.

Мова Java має у своєму арсеналі великий багаж переваг для зручного та успішного використання навіть у сучасних продуктах. Незалежність платформи, що дозволяє працювати з базовими платформами, такими як Windows, Android та інші.

Містить у собі велику кількість вбудованих бібліотек для зручної розробки застосунків та подальша підтримка. Якісний інструмент розробки що пропонує великий арсенал для підтримки редагування, тестування, та керування відредагованими блоками.

Також для того, щоб розпочати зручно використовувати мову Java, потрібно додатково встановити SDK, який в свою чергу являється великим набором бібліотек, також має підтримку вищого рівня абстракції для більш простого процесу розробки.

2.3 JDK

Інструментарій для розробки ПО, представляється як набір певних інструментів, які використовуються для розробки програмного забезпечення, що надають доступ для створення програми для платформ, комп. систем, консолей, та інших ОС.

Аби створити застосунок розширеним функціоналом, майже усі розробники використовують додаткові бібліотеки для розробки ПО. Певні devkit`и є дуже важливими для створення спеціальних застосунків. Як ось наприклад для створення застосунку на платформі Android необхідно встановити JDK, а для Windows необхідно отримати .NET.

2.4 Середовище для розробки мобільних застосунків AndroidStudio

AndroidStudio представляє собою інтегроване середовище для розробки мобільних застосунків на базі ОС Android. Перед появою AndroidStudio, розробники використовували для своїх проектів плагін ADT, який був створений для Eclipse. Це середовище було спроектоване на базі коду intelliJ, яка підтримується і покращується JetBrains.

Інструментарій надає змогу для розроблення застосунків не лише для мобільних пристроїв та планшетів, також і для інших пристроїв, які на сьогоднішній день можуть володіти операційною системою Android, це телевізори, автомобілі, холодильники.

Середовище для розробки було налаштоване для роботи з типовими задачами, які були обрані для виконання під час проектування мобільного застосунку для Android. У середовище були введені певні інструменти для більш гнучкого тестування та зручний спосіб перевірки на можливість роботи з старішими версіями. Також можливість гнучко спроектувати

відображення для пристроїв з різними екранами. В AndroidStudio було створено декілька унікальних функцій, для прикладу можна навести нову підсистему для зручного тестування і розгортання мобільних застосунків, вона заснована за допомогою використання інструментарії Gradle.

2.5 Методи проектування та створення GUI

Задля зручності та збереження часу для розробки мобільних застосунків, створено набір інструментів для проектування та створення Graphic User Interface. Цей набір надає змогу попередньо оглянути інтерфейс у різних його положеннях чи як він буде відображатись на смартфонах з різним розширення екрану. Також у себе включає інструмент, який може створювати власні графічні елементи.

У середовищі передбачені такий основний функціонал:

- Layout макети
- Зручний редактор макетів, який дозволяє переглянути список наявних графічних елементів і перетягнути їх на макет GUI
- Середовище базується на системі Gradle
- Консоль для розробника, допомога та підказки для коректної оптимізації

Вбудований у середовище інструмент для рефакторингу, дозволяють перевірити та порівняти сумісність, моніторинг споживання ресурсів, та перевірку на зручність у використанні. Наявна функція для швидкого зберігання змін. Також для зручного відстеження присутня система для знаходження та помічення помилок.

2.6 Мова Kotlin

Kotlin представляє собою типізовану мову для програмування, яка контролюється розробкою компанії JetBrains.

Ціллю авторів було створення мови більш зрозумілої та безпечної ніж Java. Вона розпочала своє створення ще з 2010, а вже у 2011 була представлена для людей. А вже у 2017 році являється повністю офіційною мову, яка представляє можливості для розробки мобільних застосунків на базі Android.

Головна особливість цієї мови, яку розробники ставлять собі за мету під час проектування та розробки, це зручність та простота у використанні, а головне це сумісність з мовою Java. JetBrains займається створення дуже багатьох продуктів які взаємодіють з мовою Java і також має великий досвід у сьогоденних програмах для розробки. Потреба у новій мові була головною проблемою до появи Kotlin, адже створити мову, яка мала можливість взаємодіяти з вже написаними рядками коду Java, Що допоможе плавно перейти і продовжувати написання. Саме JetBrains відчули що саме потребують споживачі та розробники, і дала їм саме такий покращенний інструмент.

Kotlin являється чудовою програмою, яка є зручною для розробки застосунків на базі Android, з усіма перевагами мови для Android.

Перевагами є:

- Чудова продуктивність, за рахунок схожості структури байткоду, котлин працює на рівні з Java
- Присутність поетапної компіляції
- А головне, це підтримка JDK6, саме завдяки цьому застосунки створені на Котлин мають можливість працювати без перебоїв навіть на старих смартфонах.

Завдяки простоті, нею може скористатися будь який розробник.

2.7 Manifest

У вихідній папці кожного мобільного застосунка знаходитися файл під назвою AndroidManifest.xml. Цей файл маніфесту завжди містить у собі

головну інформацію про сам застосунок, ця інформація там зберігається для зчитування системою Android. Адже після отримання цієї інформації система має змогу виконати заданий код.

Основними задачами маніфесту є:

1) Встановлює ім’я програми. Встановлене ім’я представляє собою унікальний ідентифікатор для проекта.

2) У цьому файлі зберігається інформація про певні компоненти, які відносяться до проекту, тобто служби, приймання повідомлень і інформації про контент, що містить у собі інформацію про проект. Також встановлює назви класів і поширює їх можливості.

3) Надає визначення процесів, в котрих буде відображатись компоненти застосунку.

4) Оголошення дозволів для отримання можливості взаємодіяти з захищеними частинами.

5) Встановлює дозволи, які надають доступ до потрібних компонентів.

6) Він оголошує мінімальний рівень інтерфейсу Android, параметри якого потрібні для синхронізації з додатком.

7) Надає повний список з бібліотеками, з якими пов’язаний застосунок.

2.8 Room Data Base

«Локальна база даних» може мати кілька значень, але в сенсі архітектури комп’ютера вона зазвичай означає наступне:

- База даних — і, можливо, екземпляр менеджера бази даних — який працює на тій же фізичній машині, що й програма.
- Жодні інші програми не мають доступу до збережених даних або екземпляра менеджера бази даних.

Багато програм, зокрема програми, що працюють на смартфонах, використовують щось на зразок SQLite для керування локальними даними. У цій ситуації не знадобиться екземпляр керування базою даних, оскільки «менеджер баз даних» — це бібліотека, яка живе в адресному просторі вашої програми.

«Локальна база даних» архітектурно протиставлена «віддаленій або серверній базі даних». Внутрішня база даних — це та, до якої програма буде підключатися за допомогою мережі або буде звертатись до неї, підключаючись до зв'язувального програмного забезпечення, яке у результаті звернеться до серверної бази даних.

Є можливість використовувати локальну базу даних як сховище для тимчасових даних, оскільки взаємодія з серверною базою даних може бути повільною та дорогою, ви можете використовувати її для зберігання локальної інформації про конфігурацію, а часто і те й інше.

RoomDB являє собою частину нової бібліотеки від Google, що дозволяє зберігати дані застосунків. Тобто ця функція дозволяє створити локальну базу даних, і використовувати її для зберігання інформації, створеної під час роботи з застосунком.

На відміну від інших інструментів для роботи з базами даних, Room використовує обробку анотацій для виконання власного алгоритму зберігання. Це означає, що ніякі класи не повинні нічого розширювати.

Також є можливість переглядати зміну даних інтегруючи їх з API LiveData. Це означає якщо у вас складна схема, коли зміни будуть з'являтись одночасно у декількох місцях, то Room повідомляє про це.

2.9 Життєвий цикл застосунку

Коли користувач відкриває застосунок, виходить із нього, та потім повертається до нього, екземпляри Activity у застосунку переходить у різні

стани свого життєвого циклу. Клас Activity надає ряд зворотних команд, які дозволяють дізнатись що стан змінився. Чи система запускає, зупиняє або повертається до застосунку або завершує процес, у якому знаходиться ця діяльність.

У методах звернення до життєвого циклу після зміни його стану, є можливість задати команди для того, як буде поводиться активність, коли користувач закриває, призупиняє чи знову повертається до застосунку. Наприклад, якщо створити відео програвач, є можливість призупинити відео, коли користувач перемикається на іншу програму. Якщо користувач повернеться, є можливість знову дозволити користувачеві відновити відео з того самого місця на якому воно було призупинено. Якщо описати іншими словами, кожна відповідь на дію дозволяє виконувати конкретну функцію, що буде відповідати заданій зміні стану для конкретної відповіді. Виконання правильної відповіді в потрібний момент і коректна обробка зміни станів роблять застосунок більш надійним та продуктивним.

Наприклад, хороша реалізація звернення до зміни життєвого циклу може допомогти уникнути:

- Збій застосунку коли користувач отримує телефонний дзвінок або перемикається на інший застосунок під час використання поточного застосунку.
- Оптимізувати витрати цінних системних ресурсів, коли користувач не користується застосунком у активному вікні.
- Збій та видалення прогресу, якщо користувач залишає застосунок та пізніше повертається до неї.
- Можливий збій виконаного прогрессу користувача, якщо екран змінює орієнтацію з альбомної на книжкову.

Для переходу між станами життєвого циклу, клас Activity надає можливість використати набір із шести відповідей:

- onCreate()
- onStart()
- onResume()
- onPause()
- onStop()
- onDestroy()

Якщо користувач покидає застосунок, система викликає метод для зупинення активності. У деяких випадках це призупинення являється лише частковим. Активність досі зберігається в пам’яті, якщо користувач вирішив перемкнутись на іншу програму, тому він все ще має можливість повернутися до застосунку. Якщо користувач повертається до цієї дії, дія відновлюється з того місця, де він зупинився. За деякими винятками, програмам заборонено запускати діяльність у фоновому режимі.

Можливість, що система знищить процес застосунку разом із діяльністю в ньому, залежить від поточного стану активності. Стан активності та видалення з пам’яті предоставляє більше інформації про стан зв’язку між самим станом та його можливою вразливістю до термінового завершення.

В залежності від того, наскільки складним є застосунок, то не є обов’язковим впроваджувати цілком усі методи життєвого циклу у застосунок. Проте головне потрібно зрозуміти, реалізація яких саме станів необхідна для застосунку, вони будуть забезпечувати роботу застосунку.

onCreate()

Метод який необхідний для реалізації, що буде приводитись в дію коли застосунок буде вперше запущений на мобільному пристрої. Після того як застосунок був запущений, він переходить у стан “створений”. Цей метод використовує базову логіку для запуску мобільного застосунку, що має виконатись тільки один раз за весь час роботи застосунку. Для

прикладу, створена реалізація методу має змогу зв'язати дані зі списками, зв’язати дію з вью модель та створити екземпляри можливих змінних для діапазону класу. Реалізований метод буде отримувати параметр, який є об’єктом бандлу, який містить заздалегідь збережений стан. Якщо раніше такого не існувало, то значення об’єкта буде нульовим.

onStart()

Якщо активність отримала команду для переходу у стан “розпочато”, тоді система викликає цей метод. Він робить активність яка є видимою для користувача, тому що програма робить підготовку до переходу у стан активності, для виходу на перший план і стала можливою для взаємодії. Тобто завдяки цьому методу застосунок відтворює код, що підтримує активним інтерфейс для користувача.

Метод може завершуватись досить швидко, також як і у стані “створено”, активність не буде залишатись постійною. Коли метод завершується, активність переходить у стан “продовжено”, і система викликає відповідний метод.

onResume()

Якщо застосунок отримує стан “продовжено”, то він повертається до роботи на передньому плані, вже після цього, система викликає відповідний метод. У цьому стані застосунок взаємодіє з користувачем. Застосунок буде залишатись у цьому стані до тих пір, поки не відбудеться те, що переведе застосунок у інший стан. Для прикладу можливість отримати телефонний дзвінок, чи можливо зміна першопланової активності користувача, чи звичайне блокування екрану мобільного пристрою.

Якщо стан переходить у “відновлено”, будь-який компонент що відповідає життєвому циклу, а також має прив’язку до життєвого циклу дії, буде отримувати подію для продовження. Тому саме тут кожен компонент

життєвого циклу застосунку може увімкнути будь-який можливий функціонал, що повинен працювати тоді коли компонент є активним на першому плані.

Якщо відбувається якась подія, що може перервати процес, тоді активність переходить у стан “призупинено”. Якщо активність повертається у стан “відновлено” зі стану “пауза”, система знову звертається до методу. Саме тому потрібно використовувати метод для ініціалізації компонентів, що звільняються під час стану паузи, і виконувати будь-які інші ініціалізації, які мають відбуватися щоразу, коли діяльність переходить у стан відновленого.

onPause()

Виклик цього методу, зе причина того, що користувач перестає взаємодіяти з застосунком, тобто він не займає перший план екрана пристрою. Використання методу потрібно для того, щоб призупинити або налаштувати операції, які не повинні продовжуватися, поки тим часом мобільний застосунок знаходиться у стані “призупинено”, який через певний час буде відновлений. Для переходу у ций стан існує кілька причин:

1.Якісь події можуть перервати виконання;

2.Нові версії андроїд надають змогу працювати з кількома програмами одночасно, тобто функція багатовіконного режиму. Через те, що увага зосереджується тільки на одному вікні, всі інші вікна отримують стан “призупинено”.

Коли застосунок отримує стан “призупинено”, тоді будь-який компонент що відповідає життєвому циклу, також отримує стан “призупинено”. Тому кожен компонент життєвого циклу може зупинити свою функціональність, виконання якої на даний момент не потрібне, допоки він не вийде на передній план.

onStop()

Якщо користувач виходить із мобільного застосунку, він отримує команду перейти у стан “зупинено”, саме тоді застосунок звертається до нашого методу. Таке може відбуватись тоді, коли новий, щойно запущений застосунок охоплює увесь екран. Також звернення до методу відбувається після того, як застосунок завершив роботу, або її робота була перервана.

Якщо мобільний застосунок перейшов у стан “зупинено”, тоді будь-який компонент що має відношення до стану життєвого циклу, буде також отримувати подію “зупинено”. Завдяки цьому кожен компонент життєвого циклу може припинити свою функціональність, до тих пір, поки застосунок знову не буде відображатися на екрані.

onDestroy()

Даний метод викликається перед повним знищенням активності мобільного застосунку. Застосунок переходе у цей стан по причині того, що:

1.Завершення діяльності, тобто повний вихід із застосунку користувачем;

2.Тимчасове знищення роботи через зміну конфігурацій.

Якщо застосунок перейшов у стан “знищено”, тоді будь-який компонент, що пов’язаний, також отримує статус “знищено”.

Коло робота застосунку завершується, цей метод є останнім у життєвому циклі, що може отримувати активність. Якщо метод викликається в результаті зміни конфігурації, тоді негайно створюється новий екземпляр активності, а потім викликається метод “створити” для новоствореного екземпляра в цій новій конфігурації.

Висновки до розділу

В другому розділі було розглянуто поняття мобільного застосунку, сучасні методи та підходи до його розробки на базі ОС Android.

Проведений аналіз наявного інструментарію, вивчення методів, та ознайомлення з принципами його використання. Також проведено ознайомлення з літературою, лекціями та іншими ресурсами на відповідні теми.

РОЗДІЛ 3
Проектування та програмна реалізація мобільного застосунку для керування справами на базі андроїд

3.1 Створення архітектури

Архітектура створеного програмного застосунку на базі ОС Android містить у собі комплект фрагментів, кожен з яких прив’язаний до власного вікна керування.

Фрагмент - виступає як клас мови Java, який зберігає свої дані у файлі з такою ж назвою.

XML - файл який містить у собі опис дизайну та розташування об’єктів з використанням xml-коду, для кожної активності відводиться власний файл.

Під час роботи застосунку при виконанні дій, відбувається автоматичне розпізнавання розширення екрану мобільного пристрою, і автоматично змінює розмір контенту спираючись на опис у xml-файлі вікна для зручного відображення. Завдяки цьому, усі наші активності будуть вміщенні в вікно мобільного пристрою, на якому запущений мобільний застосунок.

Після встановлення застосунку, користувачу потрібно лише його запустити. Після чого відразу потрапляє на сторінку з списком створених проектів. Для створення нового проекту потрібно натиснути відповідну кнопку і заповнити поля з інформацією. Після чого усі дані записуються у локальну базу даних RoomDB. RoomDB являє собою частину нової бібліотеки від Google, що дозволяє зберігати дані застосунків. Тобто ця функція дозволяє створити локальну базу даних, і використовувати її для зберігання інформації, створеної під час роботи з застосунком.

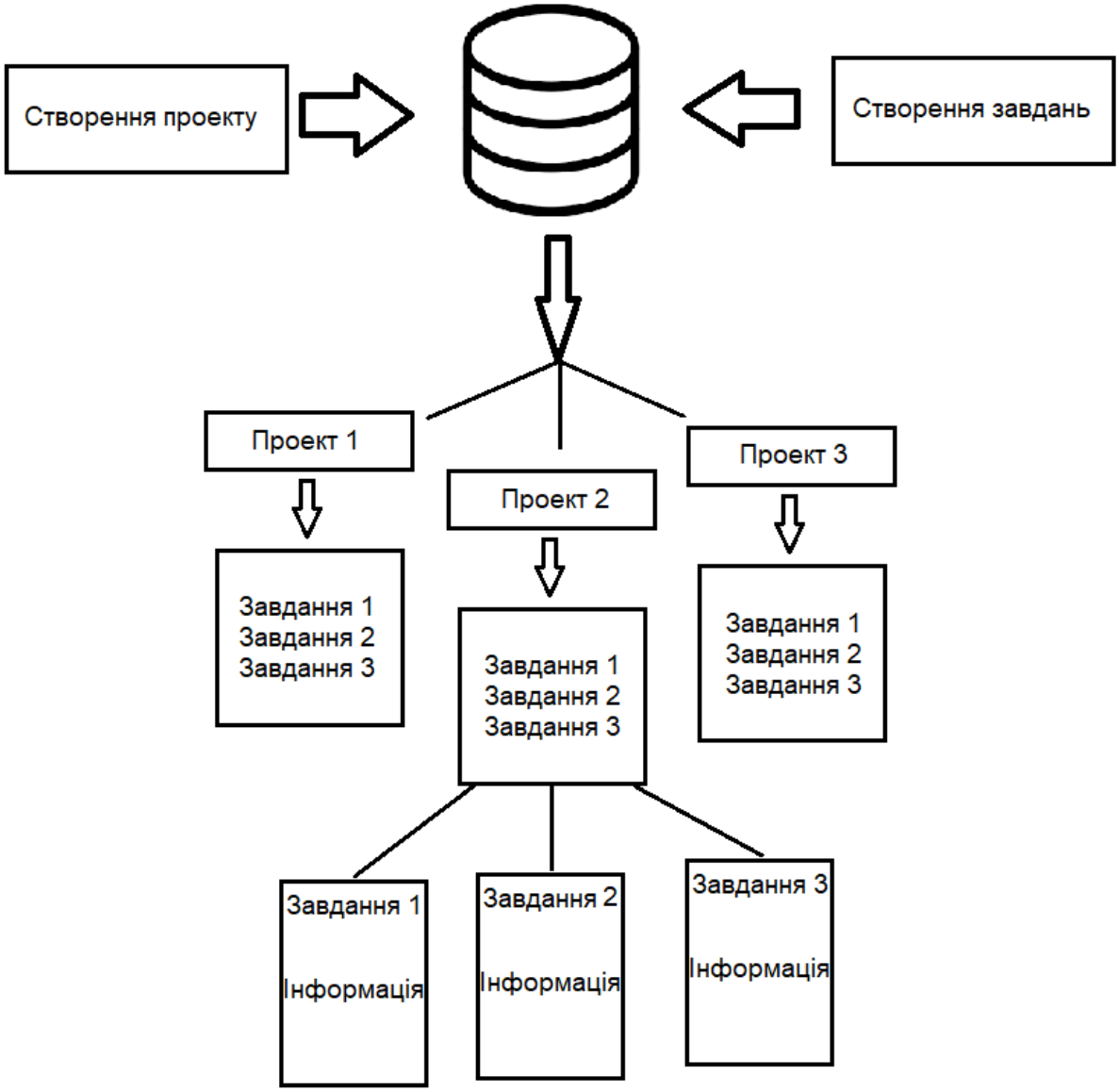


Рисунок 3.1.1 – Схема локальної бази даних

Дані будуть зберігатись у базі шляхом створення загального проекту, у якому міститься інформація про те, які саме завдання відносяться до цього проекта, а власе завдання будуть у собі містити детальну інформацію, опис, та інше.

Для доступу до створеної локальної бази даних було використано методи DAO. Використовуючи Data Access Objects, для доступу до інформації у базі даних мобільного застосунку, полегшується імітація доступу.

3.2 Макет інтерфейсу

Для початку було попередньо створено макет GUI нашого мобільного застосунку:

- Головний екран
- Вікно створення проекту
- Сторінка створеного проекту
- Вікно створення задачі для проекту
- Вікно “детальніше” для кожної задачі

Першим представлено головний екран (рисунок 3.2.1) на якому відображається список з поточними проектами. Основною метою було створення дуже простого та гнучкого графічного інтерфейсу для користувачів. Зроблено це було з метою залишити тільки необхідне, звісно це лише початок, і після отримання першого фідбеку від користувачів, буде відбуватись робота над покращенням графічного інтерфейсу.

Рисунок 3.2.1 – Головний екран

Наступним кроком було створено спливаюче вікно для створення нового поточного проекту (рисунок 3.2.2), у якому вказуємо назву нашого проекта і його статус, для більшої гнучкості та зручності більшість полів юзер заповнює самостійно, наприклад поле “Статус”, на кому відповідно вказується на якому етапі поточний проект.

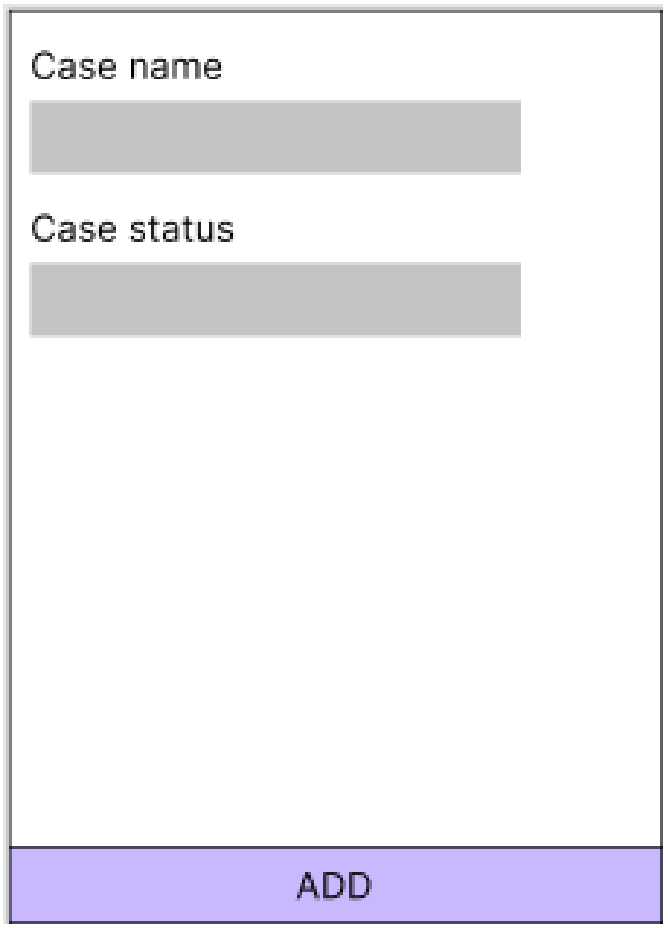


Рисунок 3.2.2 – Вікно створення проекту

Після створення проекту, нас повертає на голову, де у вигляді списку відображаються створені проекти. Усі дані були записані у локальну базу даних. Після натискання на один із створених проектів, нас переносить на сторінку з поточними завданнями власне проекту (рисунок 3.2.3).

Рисунок 3.2.3 – Сторінка з списком завдань для певного проекту

На цій сторінці ми також маємо можливість натиснути кнопку “Додати”. Після чого нам показує вікно для створення нового завдання (рисунок 3.2.4), у якому маємо поля:

- Назва нового завдання
- Поле для розгорнутого написання усіх пунктів завдання
- Ім’я людини, якій доручено роботу над новою задачею
- Статус виконання завдання
- Кінцева дата для завершення

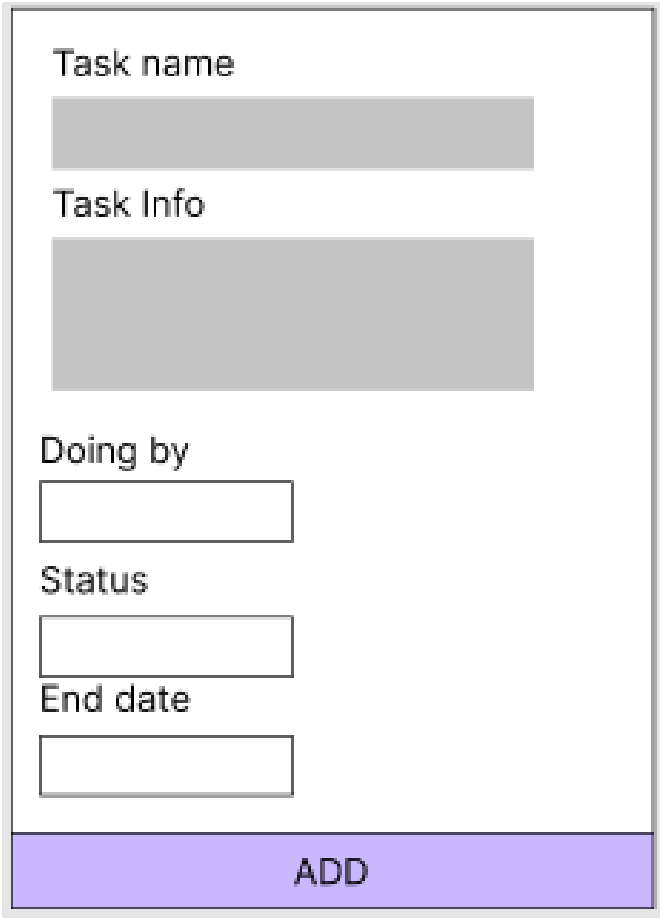


Рисунок 3.2.4 – Форма створення завдання

Після натискання кнопки “Додати”, нас повертає на сторінку з завданнями, усі створені задачі відображаються у вигляді списка, на якому вказано назву, статус та кінцеву дату.

Якщо натиснути на нове створене завдання, нас перенесе у вікно з усією інформацією яку ми вказали раніше у “формі для створення завдання” (рисунок 3.2.5)

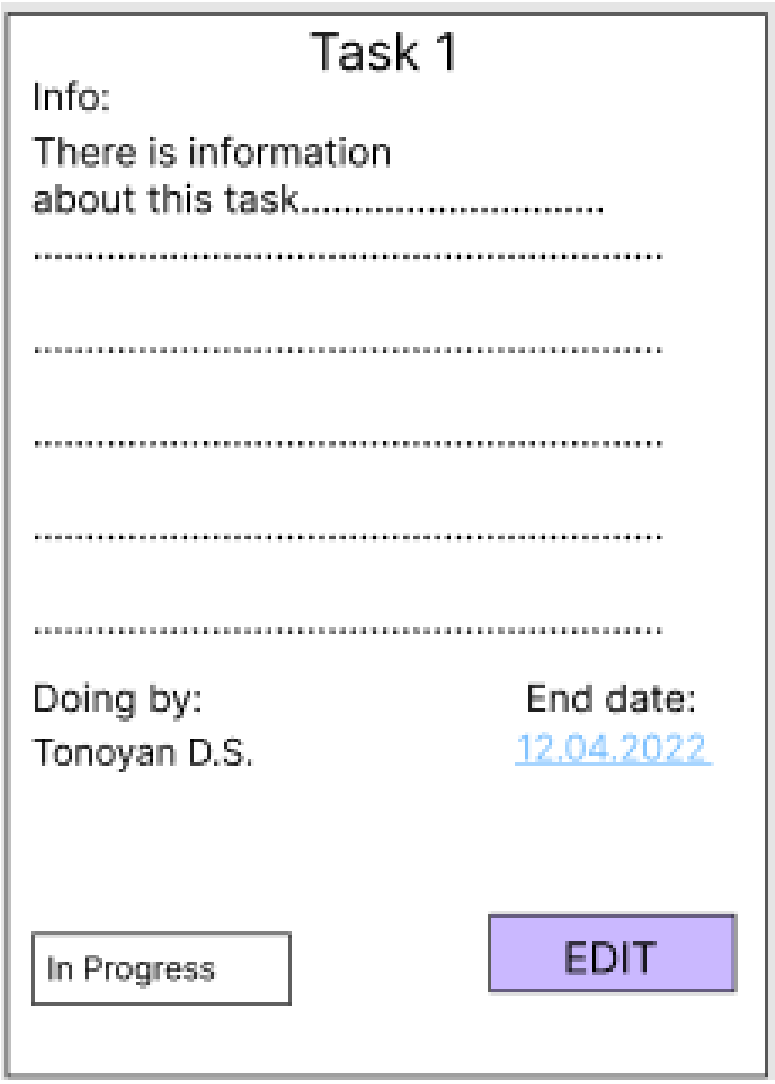


Рисунок 3.2.5 – Інформація про завдання

У цьому вікні відображаються вся заповнена нами заздалегідь інформація, яку можна відредагувати при необхідності, натиснувши кнопку “Змінити”.

Звісно це лише перший варіант графічного інтерфейсу, завдяки відгукам користувачів, і отриманої інформації з власного досвіду, буде створено багатофункціональний, максимально простий, гнучкий, і зручний мобільний застосунок для використання та відстеження власних бізнес проектів.

3.3 Робота з кодом

Архітектура застосунку складається з створеної активності, та декількох фрагментів які зв’язані між собою. Орієнтацію застосунку було зроблено строго портретною.

Для кожного екрану представлений свій клас fragment, сам застосунок в свою чергу побудований із декількох екранів, і під час взаємодії з мобільним застосунком відбувається перехід між існуючими ними.

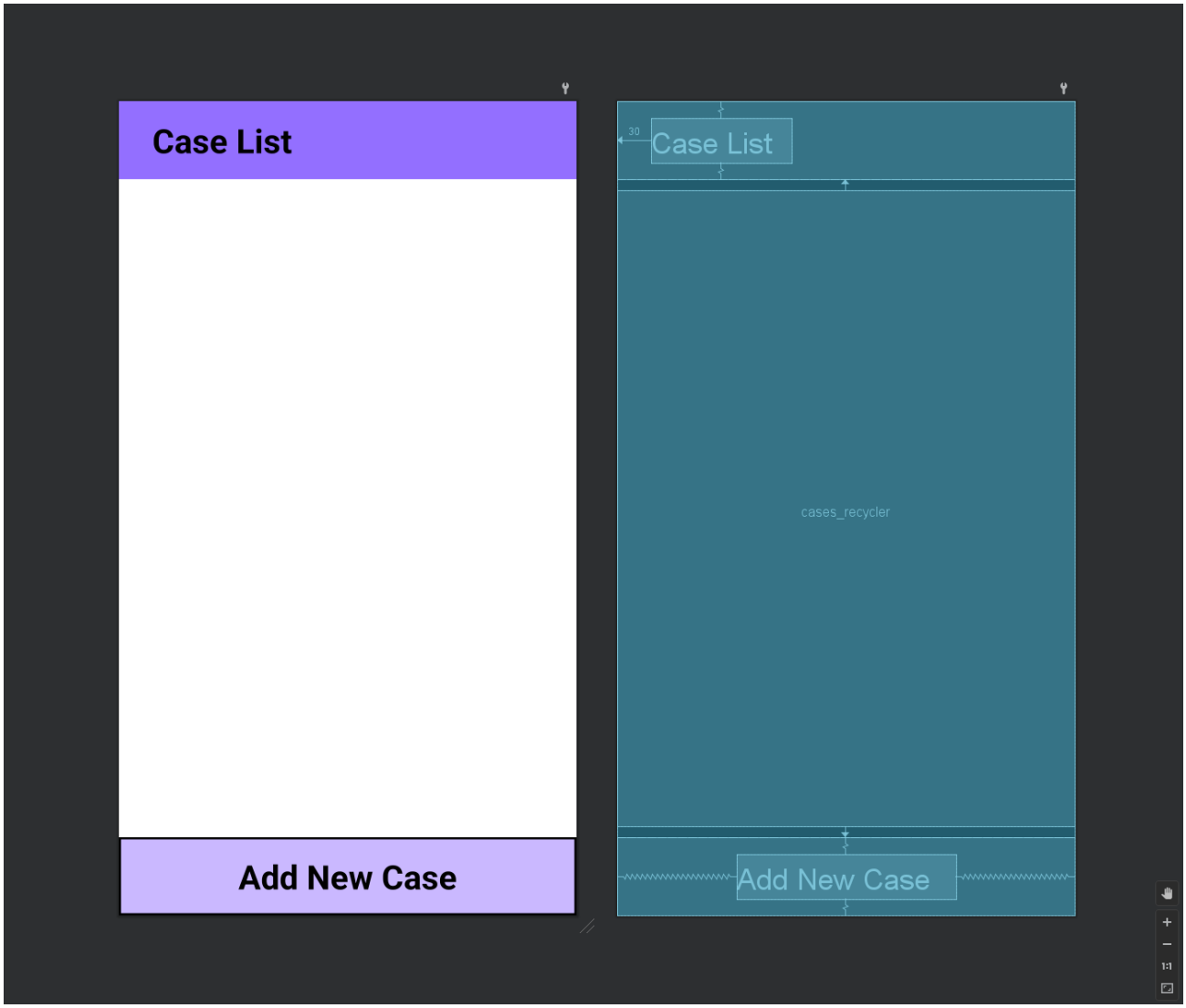


Рисунок 3.3.1 – Фрагмент з списком проектів

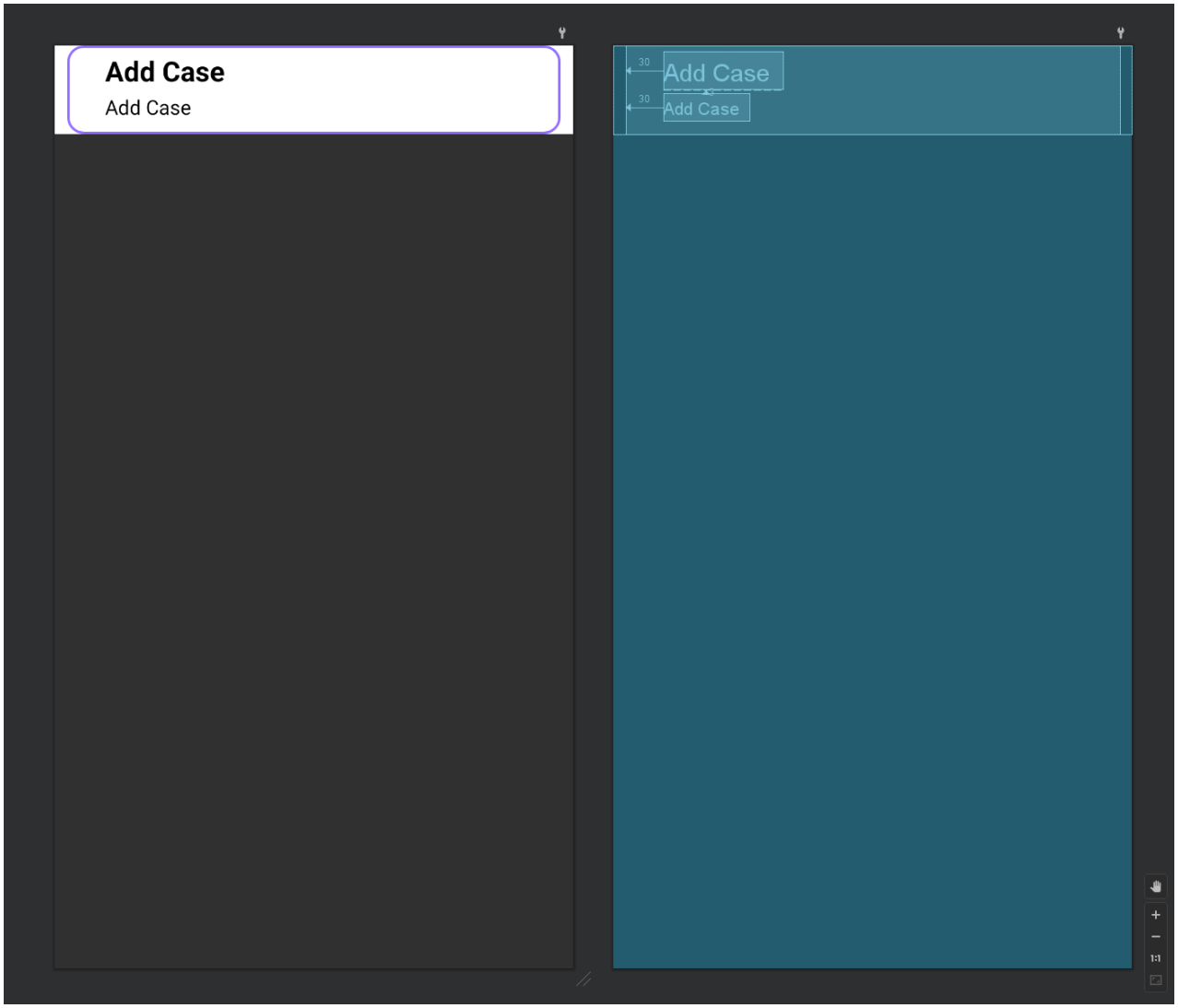


Рисунок 3.3.2 – Об’єкт доданого проекту

Створили фрагмент (рисунок 3.3.1) де після додавання в базу даних нової інформації, на цій сторінці будуть з’являтись нові об’єкти (рисунок 3.3.2) у вигляді списку. При натисканні на один із елементів ми переходимо на іншу сторінку.

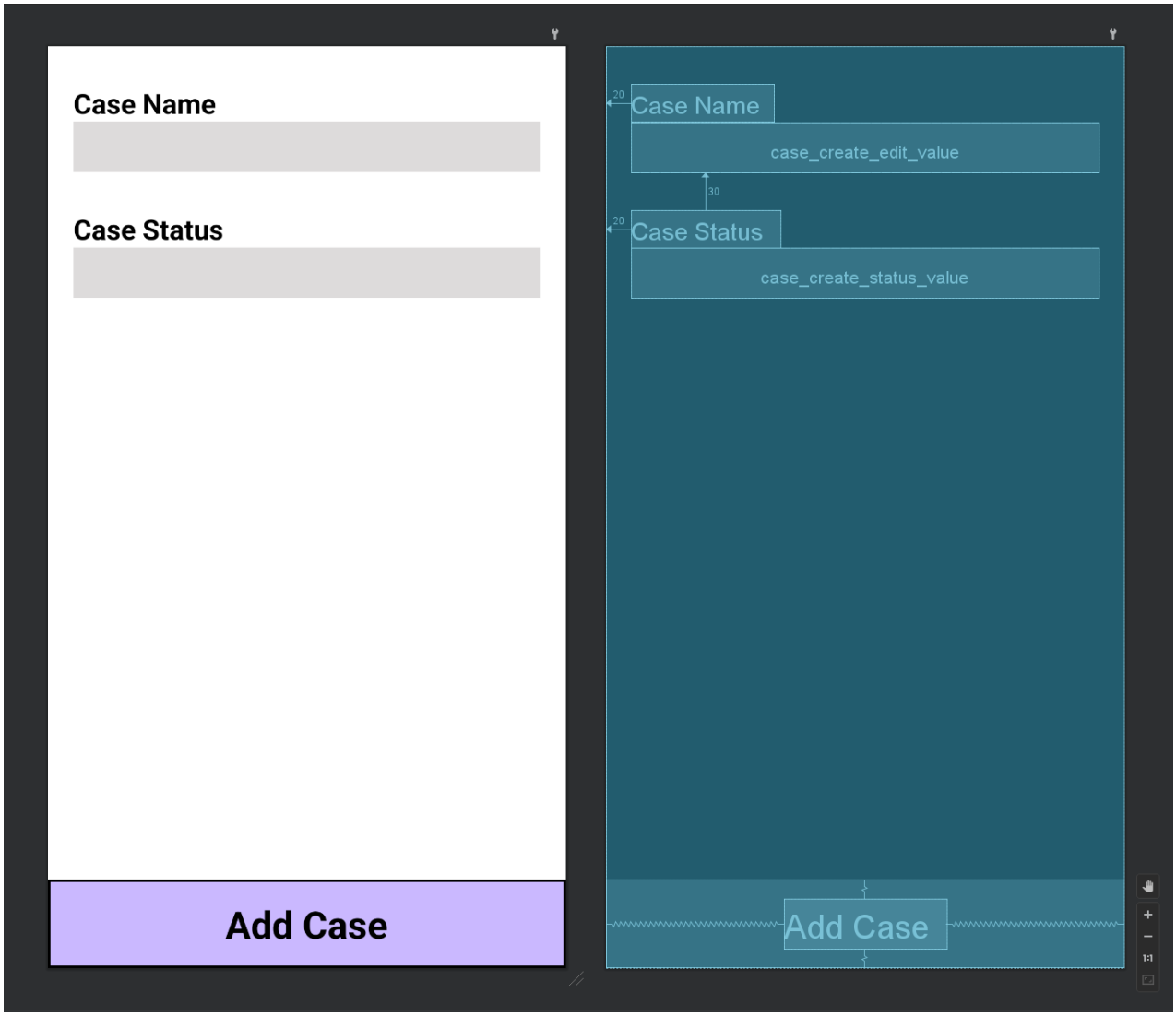


Рисунок 3.3.3 – Фрагмент створення проекту

Якщо натиснути на кнопку “Створити новий проект”, ми перейдемо на наступний фрагмент, де нам пропонують створити проект, вписавши його назву та визначити статус виконання.

Одразу після натискання кнопки “Додати проект”, ми повернемось на фрагмент з списком проектів на якому вже буде створений новий об’єкт з інформацією, яку ми щойно записали. Інформація була додана до локальної бази даних RoomDB (Рисунок 3.3.4).

```
@Database(entities = {CaseItem.class,TaskItem.class},version = 1,exportSchema = false)
public abstract class DatabaseList extends RoomDatabase {
    private static DatabaseList databaseList;
    public abstract DAO_List dao_list();
    public static DatabaseList getDatabaseList(final Context context){
        if (databaseList==null){
            synchronized (DatabaseList.class){
                if (databaseList==null){
                    databaseList= Room.databaseBuilder(context.getApplicationContext(),DatabaseList.class, name: "List_Database").build();
                }
            }
        }
        return databaseList;
    }
}
```

Рисунок 3.3.4 – Створення локальної бази даних

```
@Query("SELECT * FROM Cases_Table")
LiveData<List<CaseItem>> getAllCases();
@Query("SELECT * FROM Tasks_Table WHERE task_case LIKE :task_case")
LiveData<List<TaskItem>> getAllCaseTasks(String task_case);
```

Рисунок 3.3.5 – Запит до бази даних

```
@Entity(tableName = "Cases_Table")
public class CaseItem {
    @PrimaryKey
    @NonNull
    @ColumnInfo(name = "case_name")
    private String case_name;
    @NonNull
    @ColumnInfo(name = "case_status")
    private String case_status;

    public CaseItem(@NonNull String case_name, @NonNull String case_status) {
        this.case_name = case_name;
        this.case_status = case_status;
    }

    @NonNull
    public String getCase_name() { return case_name; }

    public void setCase_name(@NonNull String case_name) { this.case_name = case_name; }

    @NonNull
    public String getCase_status() { return case_status; }

    public void setCase_status(@NonNull String case_status) { this.case_status = case_status; }
}
```

Рисунок 3.3.6 – Створення об’єкту

Також робимо запит до нашої бази даних (рисунок 3.3.5), для створення об’єкту та відображення на фрагменті з списком проектів.

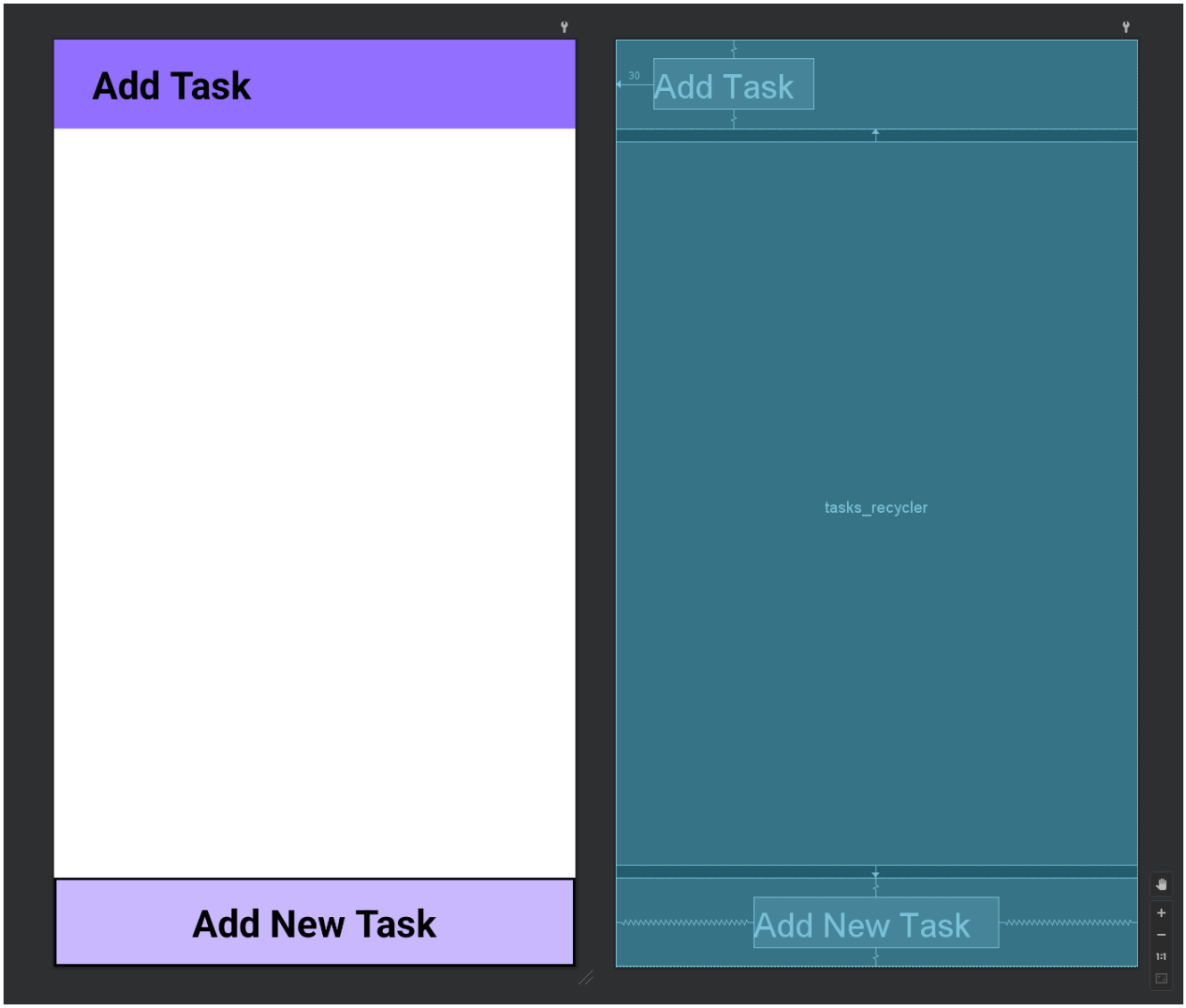


Рисунок 3.3.6 – Фрагмент з списком завдань проекта

Для кожного доданого проекта буде створений лист завдань (рисунок 3.3.6), які можна додавати за схожим принципом що і самі проекти.

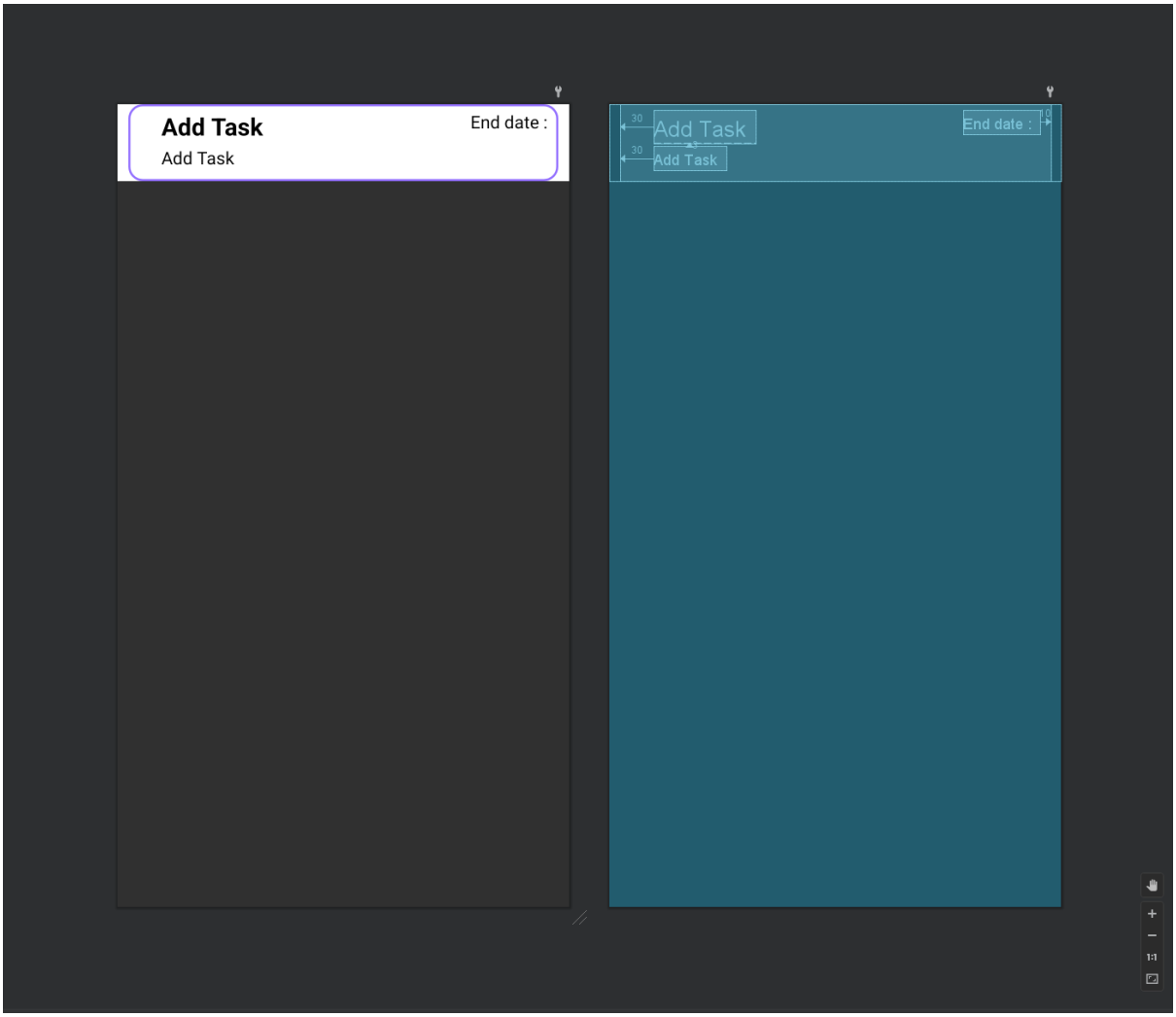


Рисунок 3.3.7 – Об’єкт доданого завдання

Такий вигляд матиме створений об’єкт завдання (рисунок 3.3.7), інформація для якого береться із бази даних (рисунок 3.3.5).

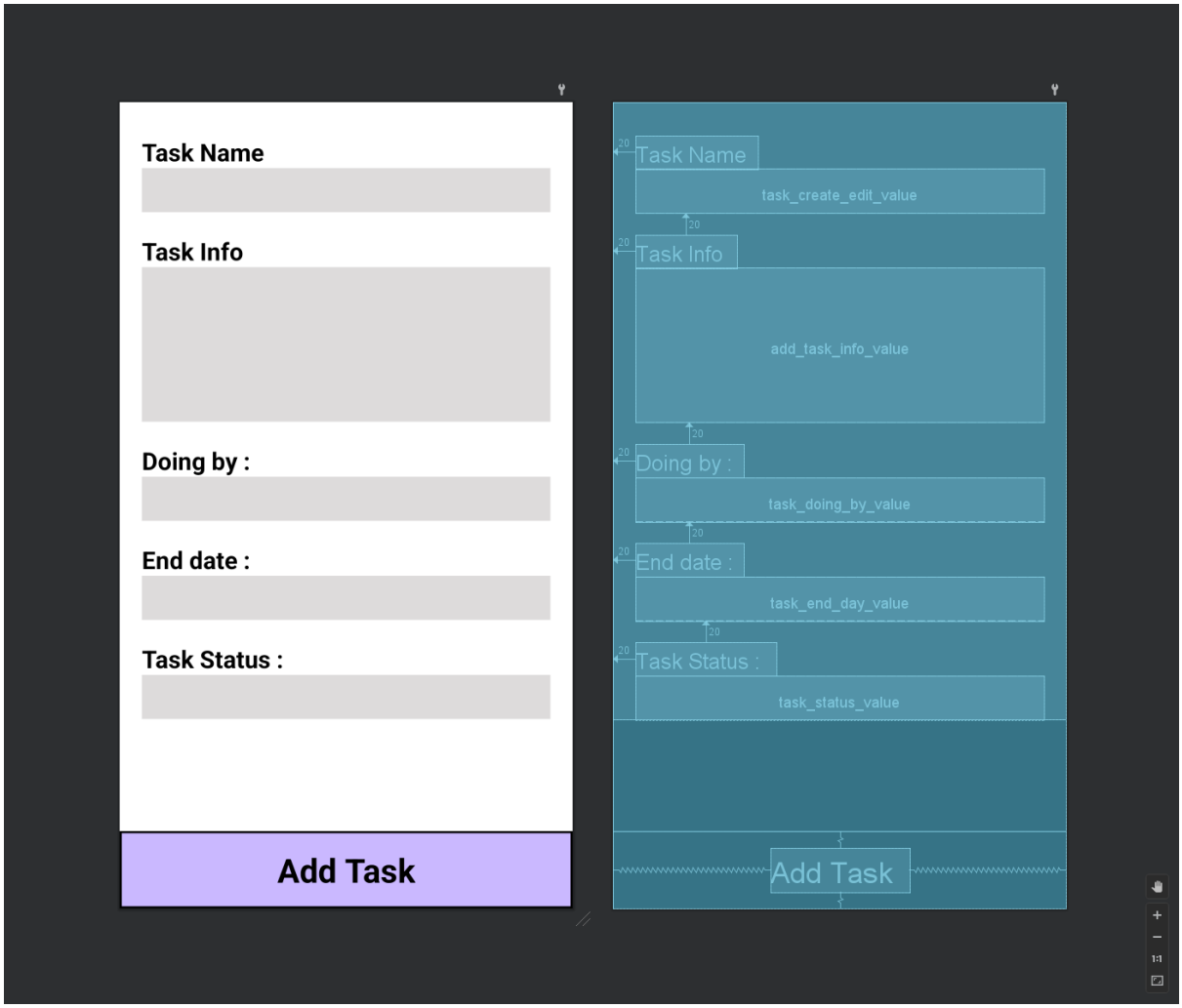
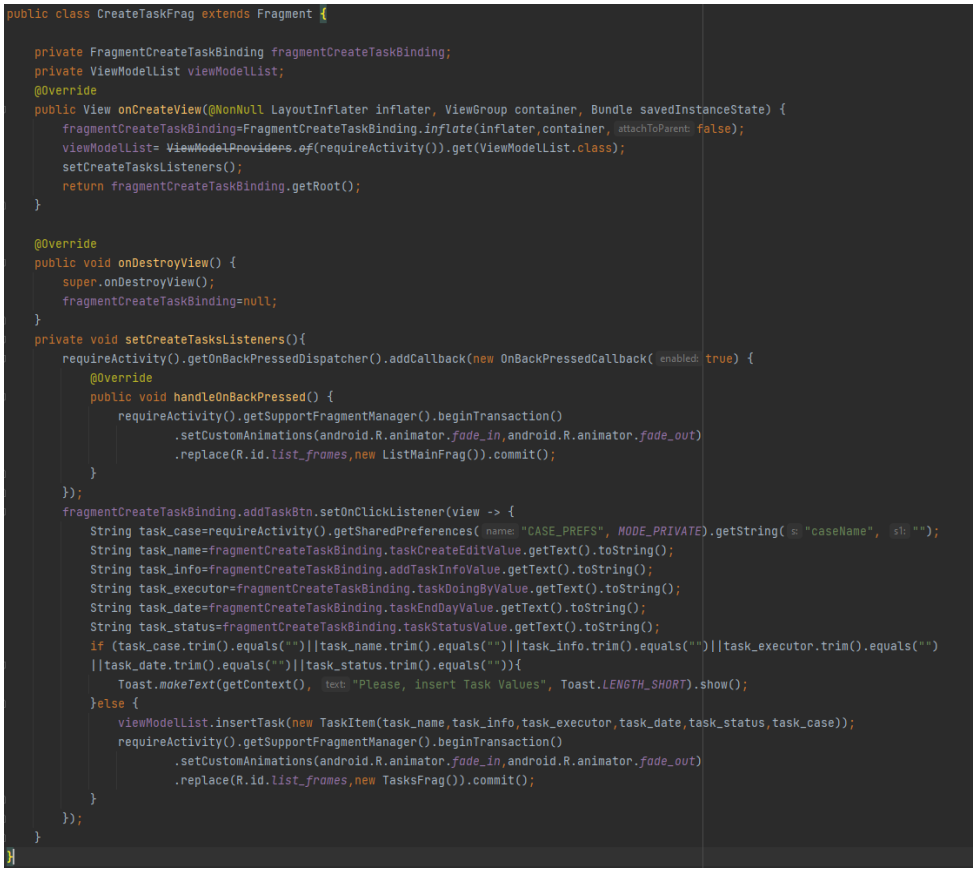


Рисунок 3.3.8 – Фрагмент створення нового завдання

Після взаємодії з кнопкою “Додати нове завдання” ми переходимо до вікна з полями для заповнення інформації (рисунок 3.3.8), де потрібно задати назву завдання, опис завдання, ким виконується, кінцева дата завершення та поточний статус виконання, оскільки це вважається основною інформацією для зручного відстежування. Потім ця інформація зчитується та записується у локальній базі даних (рисунок 3.3.9).



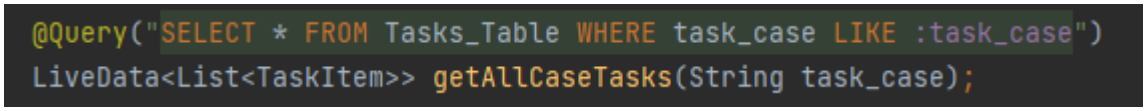
```
public class CreateTaskFrag extends Fragment {

    private FragmentCreateTaskBinding fragmentCreateTaskBinding;
    private ViewModelList viewModelList;
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        fragmentCreateTaskBinding=FragmentCreateTaskBinding.inflate(inflater,container, attachToParent: false);
        viewModelList= ViewModelProviders.of(requireActivity()).get(ViewModelList.class);
        setCreateTasksListeners();
        return fragmentCreateTaskBinding.getRoot();
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        fragmentCreateTaskBinding=null;
    }
    private void setCreateTasksListeners(){
        requireActivity().getOnBackPressedDispatcher().addCallback(new OnBackPressedCallback( enabled: true) {
            @Override
            public void handleOnBackPressed() {
                requireActivity().getSupportFragmentManager().beginTransaction()
                        .setCustomAnimations(android.R.animator.fade_in,android.R.animator.fade_out)
                        .replace(R.id.list_frames,new ListMainFrag()).commit();
            }
        });
        fragmentCreateTaskBinding.addTaskBtn.setOnClickListener(view -> {
            String task_case=requireActivity().getSharedPreferences( name: "CASE_PREFS", MODE_PRIVATE).getString( s: "caseName",  s1: "");
            String task_name=fragmentCreateTaskBinding.taskCreateEditValue.getText().toString();
            String task_info=fragmentCreateTaskBinding.addTaskInfoValue.getText().toString();
            String task_executor=fragmentCreateTaskBinding.taskDoingByValue.getText().toString();
            String task_date=fragmentCreateTaskBinding.taskEndDayValue.getText().toString();
            String task_status=fragmentCreateTaskBinding.taskStatusValue.getText().toString();
            if (task_case.trim().equals("")||task_name.trim().equals("")||task_info.trim().equals("")||task_executor.trim().equals("")
            ||task_date.trim().equals("")||task_status.trim().equals("")){
                Toast.makeText(getContext(),  text: "Please, insert Task Values", Toast.LENGTH_SHORT).show();
            }else {
                viewModelList.insertTask(new TaskItem(task_name,task_info,task_executor,task_date,task_status,task_case));
                requireActivity().getSupportFragmentManager().beginTransaction()
                        .setCustomAnimations(android.R.animator.fade_in,android.R.animator.fade_out)
                        .replace(R.id.list_frames,new TasksFrag()).commit();
            }
        });
    }
}
```

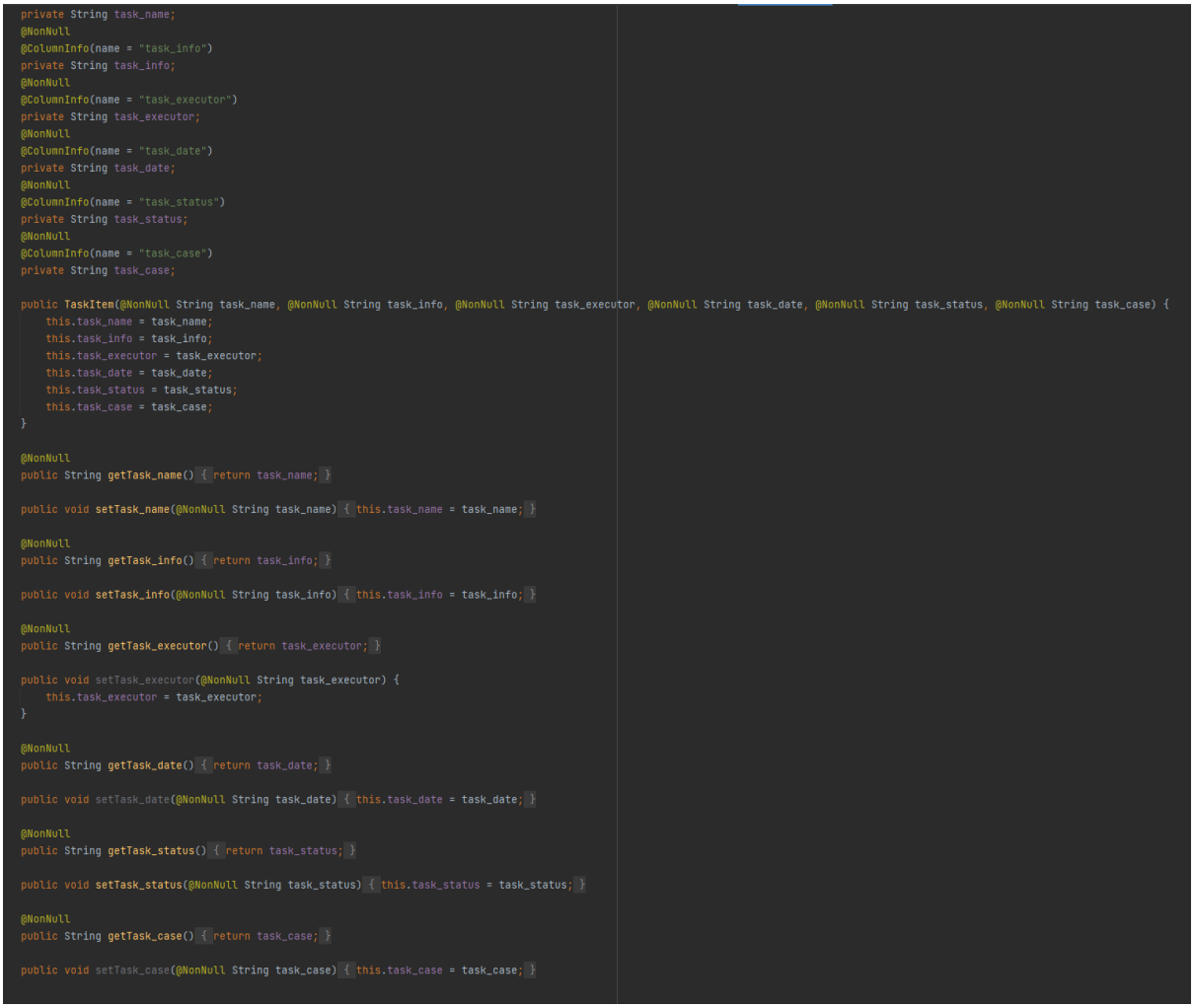
Рисунок 3.3.9 – Зчитування та зберігання інформації із текст боксів

Виконуємо запит до бази даних (рисунок 3.3.10) на отримання інформації щодо завдань, які відносяться до цього проекту. Зчитуємо інформацію, на їх основі створюємо об’єкти, які містять у собі отриману інформацію.



```
@Query("SELECT * FROM Tasks_Table WHERE task_case LIKE :task_case")
LiveData<List<TaskItem>> getAllCaseTasks(String task_case);
```

Рисунок 3.3.10 – Запит до бази даних



```
private String task_name;
@NonNull
@ColumnInfo(name = "task_info")
private String task_info;
@NonNull
@ColumnInfo(name = "task_executor")
private String task_executor;
@NonNull
@ColumnInfo(name = "task_date")
private String task_date;
@NonNull
@ColumnInfo(name = "task_status")
private String task_status;
@NonNull
@ColumnInfo(name = "task_case")
private String task_case;

public TaskItem(@NonNull String task_name, @NonNull String task_info, @NonNull String task_executor, @NonNull String task_date, @NonNull String task_status, @NonNull String task_case) {
    this.task_name = task_name;
    this.task_info = task_info;
    this.task_executor = task_executor;
    this.task_date = task_date;
    this.task_status = task_status;
    this.task_case = task_case;
}

@NonNull
public String getTask_name() { return task_name; }

public void setTask_name(@NonNull String task_name) { this.task_name = task_name; }

@NonNull
public String getTask_info() { return task_info; }

public void setTask_info(@NonNull String task_info) { this.task_info = task_info; }

@NonNull
public String getTask_executor() { return task_executor; }

public void setTask_executor(@NonNull String task_executor) {
    this.task_executor = task_executor;
}

@NonNull
public String getTask_date() { return task_date; }

public void setTask_date(@NonNull String task_date) { this.task_date = task_date; }

@NonNull
public String getTask_status() { return task_status; }

public void setTask_status(@NonNull String task_status) { this.task_status = task_status; }

@NonNull
public String getTask_case() { return task_case; }

public void setTask_case(@NonNull String task_case) { this.task_case = task_case; }
```

Рисунок 3.3.11 – Створення об’єктів

Спираючись на кількість отриманої інформації із бази даних, створюється відповідна кількість об’єктів (рисунок 3.3.11), які будують список на відповідному фрагменті для відображення усієї інформації взятої із бази даних.

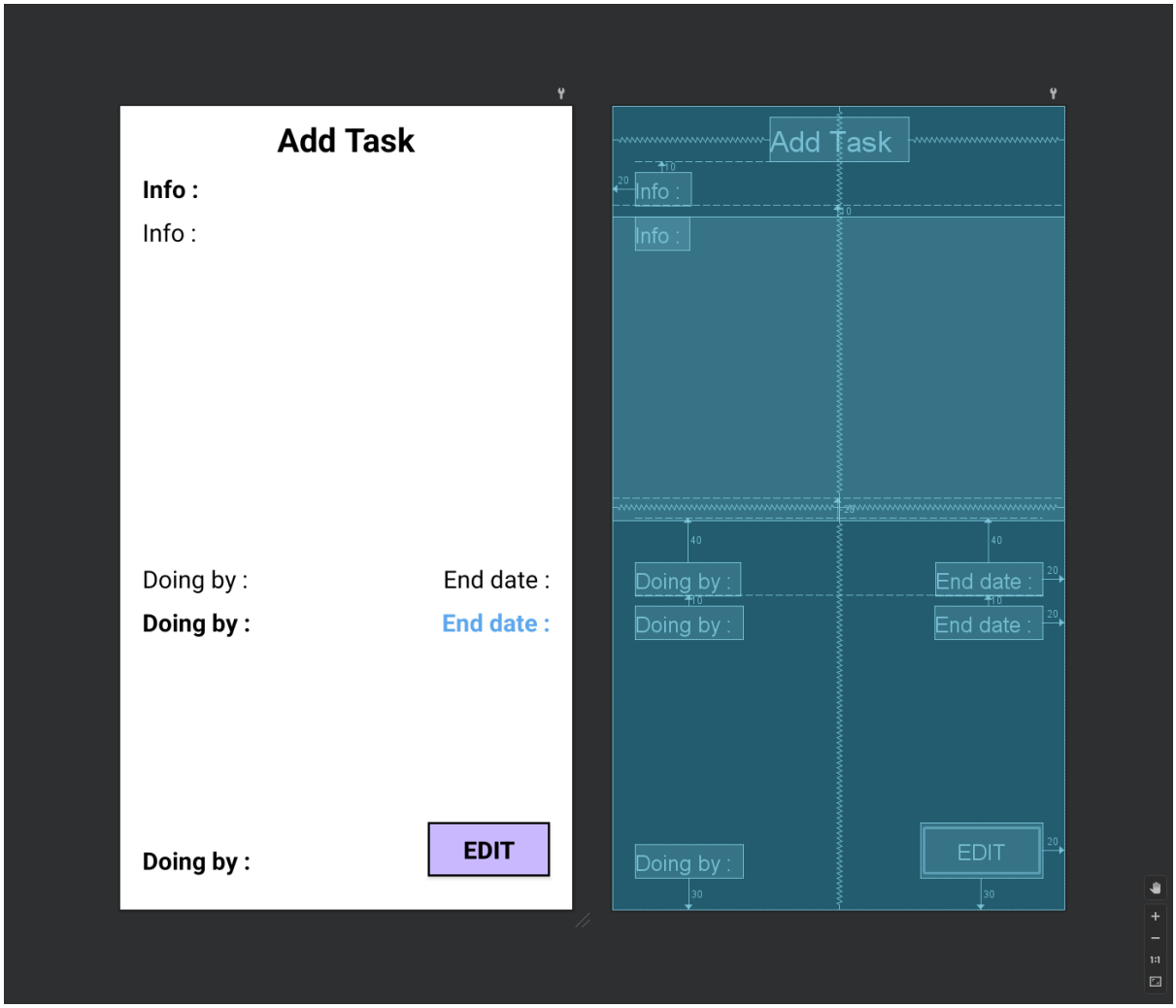


Рисунок 3.3.12 – Фрагмент інформації про завдання

При взаємодії з одним із створених об’єктів, переходимо до фрагменту, де відображається уся взята інформація після запиту до бази даних (рисунок 3.3.12). На цьому фрагменті відображена уся збережена раніше інформація, а дата виділена іншим кольором для зручності пошуку.

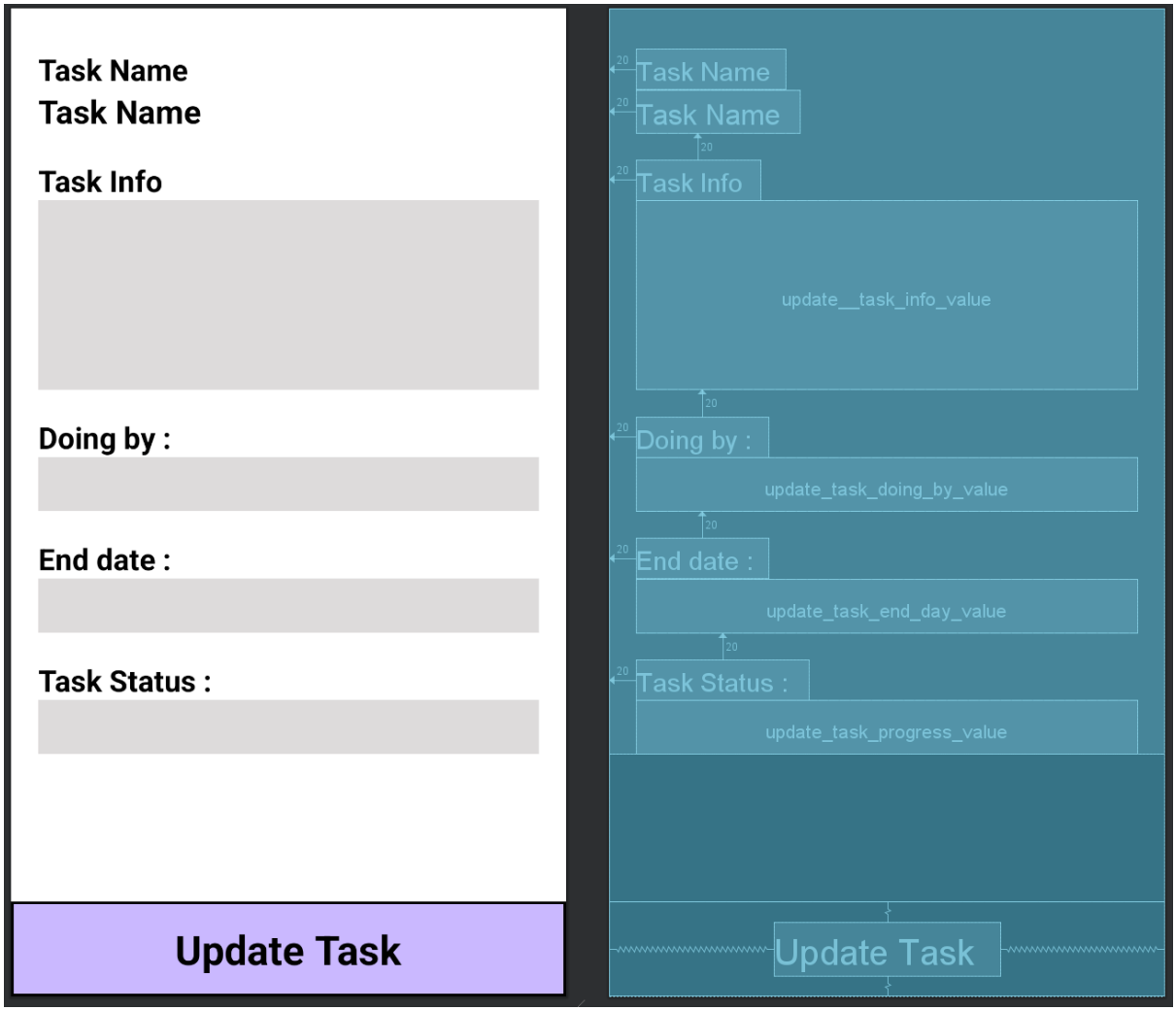


Рисунок 3.3.13 – Фрагмент зміни інформації завдання

Інформацію, що знаходиться у фрагменті з описом завдання, можна оновити. Зробити це можна завдяки відповідній кнопці у нижній частині екрана.

Переходимо у фрагмент для зміни інформації у базі даних (рисунок 3.3.13). За бажанням оновлюємо інформацію, людину що відповідальна за виконання чи статус виконання.

```
public class UpdateTaskFrag extends Fragment {

    private FragmentUpdateTaskBinding fragmentUpdateTaskBinding;
    private String task_case,task_name,task_status,task_info,task_executor,task_date;
    private ViewModelList viewModelList;
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        fragmentUpdateTaskBinding=FragmentUpdateTaskBinding.inflate(inflater,container, attachToParent: false);
        viewModelList= ViewModelProviders.of(requireActivity()).get(ViewModelList.class);
        setUpdate();
        return fragmentUpdateTaskBinding.getRoot();
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        fragmentUpdateTaskBinding=null;
    }
    private void setUpdate(){
        requireActivity().getOnBackPressedDispatcher().addCallback(new OnBackPressedCallback( enabled: true) {
            @Override
            public void handleOnBackPressed() {
                requireActivity().getSupportFragmentManager().beginTransaction()
                        .setCustomAnimations(android.R.animator.fade_in,android.R.animator.fade_out)
                        .replace(R.id.list_frames,new TasksFrag()).commit();
            }
        });
        task_case=requireActivity().getSharedPreferences( name: "CASE_PREFS", MODE_PRIVATE).getString( s: "caseName",  s1: "");
        task_name=requireActivity().getSharedPreferences( name: "CASE_PREFS", MODE_PRIVATE).getString( s: "task_name",  s1: "");
        task_status=requireActivity().getSharedPreferences( name: "CASE_PREFS", MODE_PRIVATE).getString( s: "task_status",  s1: "");
        task_info=requireActivity().getSharedPreferences( name: "CASE_PREFS", MODE_PRIVATE).getString( s: "task_info",  s1: "");
        task_executor=requireActivity().getSharedPreferences( name: "CASE_PREFS", MODE_PRIVATE).getString( s: "task_executor",  s1: "");
        task_date=requireActivity().getSharedPreferences( name: "CASE_PREFS", MODE_PRIVATE).getString( s: "task_date",  s1: "");

        fragmentUpdateTaskBinding.taskUpdateEditValue.setText(task_name);
        fragmentUpdateTaskBinding.updateTaskInfoValue.setText(task_info);
        fragmentUpdateTaskBinding.updateTaskDoingByValue.setText(task_executor);
        fragmentUpdateTaskBinding.updateTaskEndDayValue.setText(task_date);
        fragmentUpdateTaskBinding.updateTaskProgressValue.setText(task_status);

        fragmentUpdateTaskBinding.updateTaskBtn.setOnClickListener(view -> {
            task_status=fragmentUpdateTaskBinding.updateTaskProgressValue.getText().toString();
            task_info=fragmentUpdateTaskBinding.updateTaskInfoValue.getText().toString();
            task_executor=fragmentUpdateTaskBinding.updateTaskDoingByValue.getText().toString();
            task_date=fragmentUpdateTaskBinding.updateTaskEndDayValue.getText().toString();

            if (task_case.trim().equals("")||task_name.trim().equals("")||task_info.trim().equals("")||task_executor.trim().equals("")
                    ||task_date.trim().equals("")||task_status.trim().equals("")){
                Toast.makeText(getContext(),  text: "Please, insert Task Values", Toast.LENGTH_SHORT).show();
            }else {
                AsyncTask.execute(() -> viewModelList.updateSinfleTask(task_name,task_info,task_executor,task_date,task_status));
                requireActivity().getSupportFragmentManager().beginTransaction()
                        .setCustomAnimations(android.R.animator.fade_in,android.R.animator.fade_out)
                        .replace(R.id.list_frames,new TasksFrag()).commit();
            }
        });
```

Рисунок 3.3.14 – Процес зміни інформації у базі даних

Після проведених змін, вилучаємо нові дані, та проводимо заміну у відповідних частинах бази даних (рисунок 3.3.14).

3.4 Проведення тестування

Для перевірки був обраний спосіб через емулятор операційної системи Android - NoxPlayer.

Встановлення емулятора та завантаження створеного мобільного застосунку (рисунок 2.3.1)

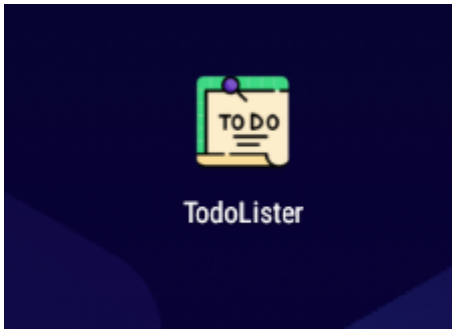


Рисунок 3.4.1 – Встановлений застосунок

Переходимо до вікна створення нового проекту (рисунок 3.4.2)

Рисунок 3.4.2 – Створення проекту

У нашому списку з’явився блок з назвою і статусом проекта (рисунок 3.4.3)

Рисунок 3.4.3 – Створений проект

Потім переходжу у проект і там створюю декілька нових задач для нашого проекту “Диплом” (рисунок 3.4.4). Створюю 3 завдання з різними даними.

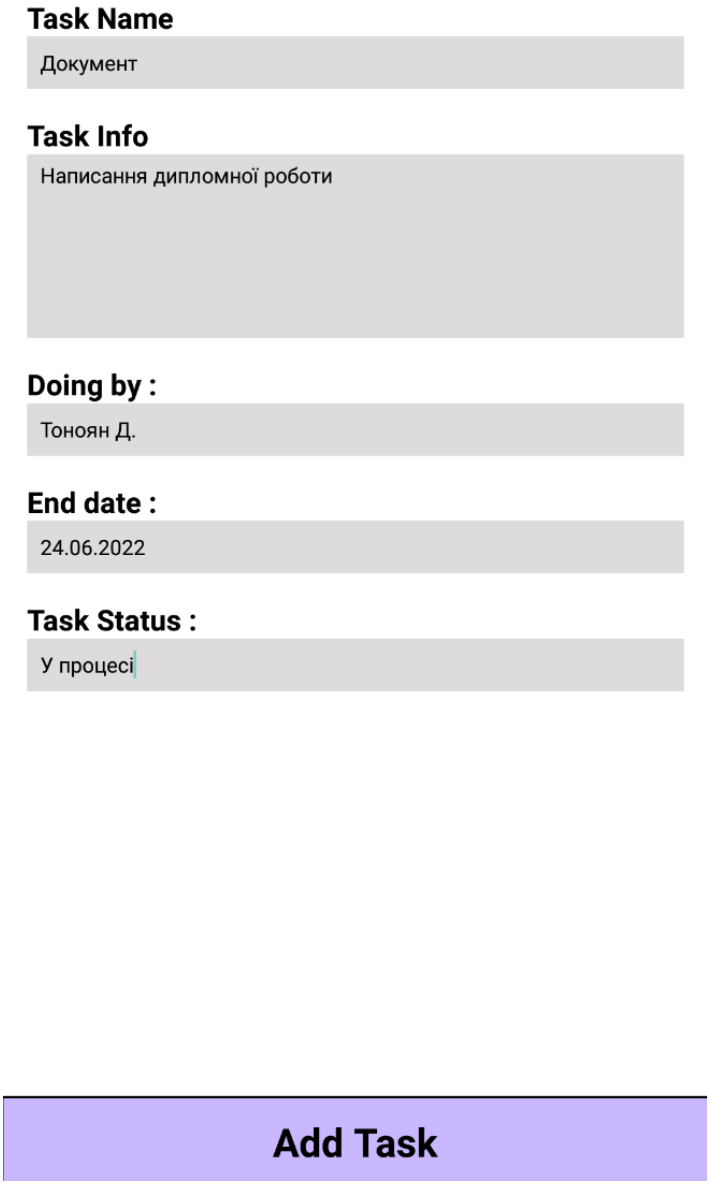


Рисунок 3.4.4 – Приклад створення задачі

Вказую назву завдання, що буде відображатись у голові створеного об’єкта. Далі детальний опис поставленої задачі, вказання нюансів чи можливих перешкод, і саме мета завдання. Потрібно також вказати людину, яка буде відповідальна за виконання поставленої задачі у потрібний срок, який я теж потрібно визначити. Обов’язковим полем є статус виконання, завдяки якому легко відстежувати на якому етапі виконання завдання.

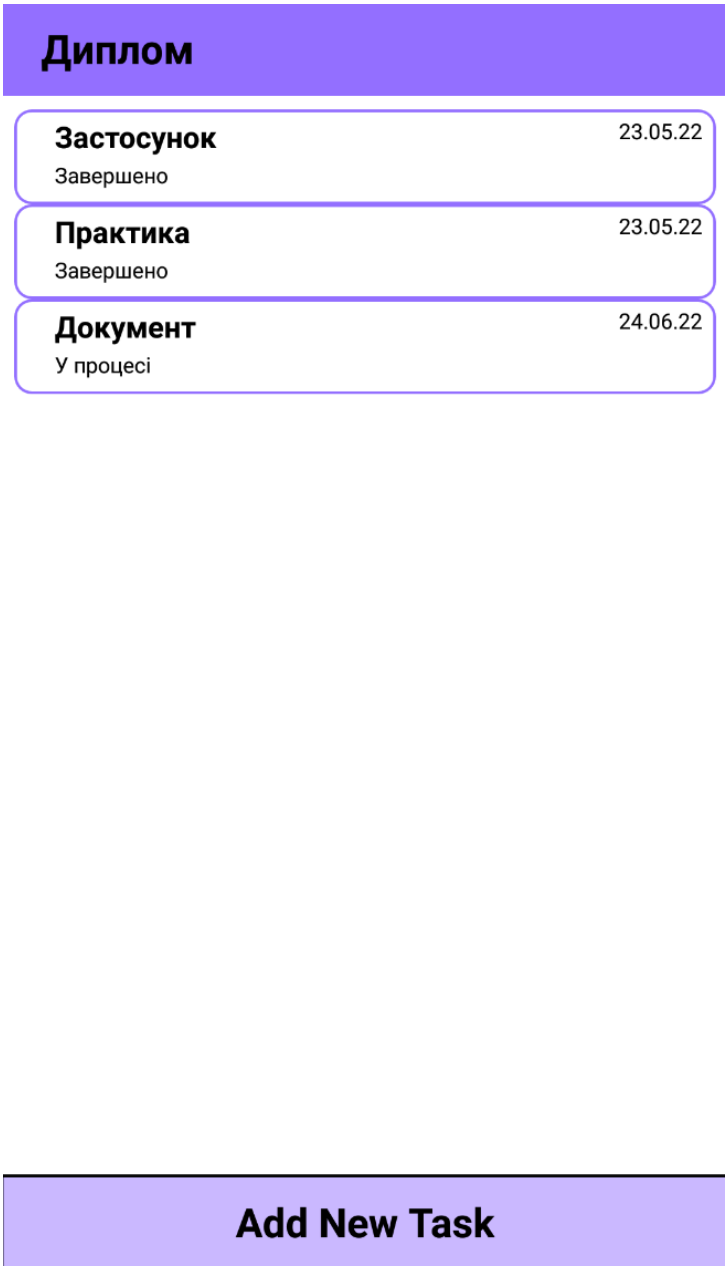


Рисунок 3.4.5 – Список поставлених задач

У результаті отримали список із 3 об’єктів, які можна відкрити і подивитись наявну в них інформацію, або відредагувати її.

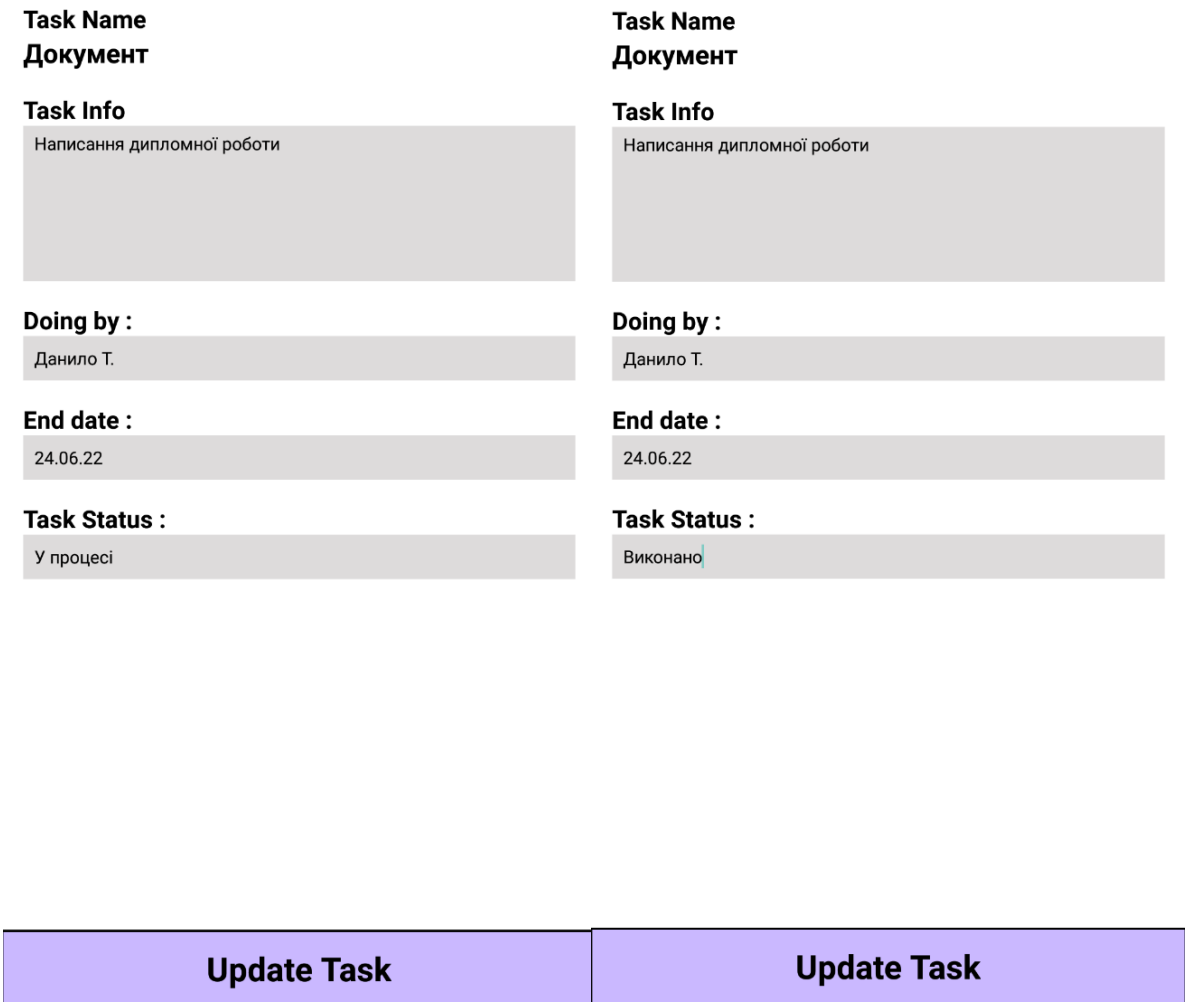


Рисунок 3.4.6 – Редагування вже існуючого завдання

Робимо зміни у статусі, змінюючи його з “У процесі” на “Виконано”. Також можемо змінити кінцеву дату нашого завдання, чи відразу змінити відповідального. Зберігаємо, і перевіряємо результат.

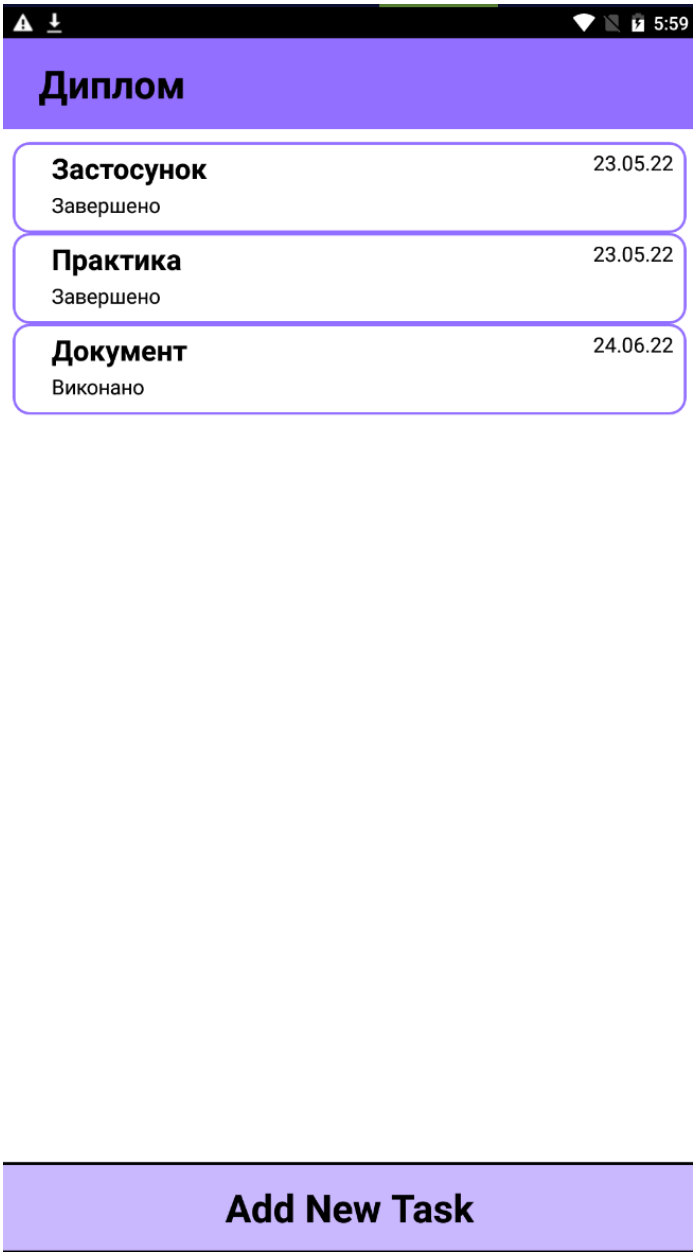


Рисунок 3.4.7 – Список завдань

Як можна побачити, зміни пройшли успішно.

Пізніше такий же тест було проведено на власному мобільному пристрої ASUS ZenPhone, ніяких збоїв не спостерігалось, тестування пройшло успішно.

Висновки до розділу

У третьому розділі був зроблений опис побудови архітектури мобільного застосунку на операційній базі Android.

Також за допомогою додаткових програм для створення та редагування концепт дизайнів, був побудований макет графічного користувацького інтерфейсу.

Виконана робота з програмним кодом за допомогою обраної раніше середи розробки мобільних застосунків на базі операційної системи Android, а також використаний інструментарій, що був проаналізований.

Проведено тестування готового застосунку за допомогою емулятора ОС Android та на власному мобільному пристрої.

ВИСНОВКИ

У результаті виконаної дипломної роботи були досягнуті поставлені цілі та мета. Серед цілей можна виділити проведення аналізу та порівняння різноманітних платформ та операційних систем, та наскільки сьогодні вигідно.

Реалізований зручний мобільний застосунок з функціоналом: створення проектів/задач, редагування та відстеження. Надалі застосунок буде поповнюватись новим функціоналом для набуватиме все більш зручний та зрозумілий графічний інтерфейс.

Також було проведене тестування мобільного застосунку, як на емуляторі так і на звичайному мобільному пристрої що підтримує операційну систему Android.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розробка мобільних додатків від А до Я: повний гайд URL: https://dan-it.com.ua/uk/rozrobka-mobilnih-dodatkiv-vid-a-do-ja-povnij-gajd/
2. Android Studio Guide URL: https://developer.android.com/studio/intro
3. Java tutorial URL: https://www.w3schools.com/java/default.asp
4. Документація по Java URL: https://www.tutorialspoint.com/java/index.htm
5. Kotlin документація URL: https://kotlinlang.org/docs/home.html
6. Руководство по Kotlin URL: https://metanit.com/kotlin/tutorial/
7. RoomDB using URL: https://developer.android.com/training/data-storage/room
8. Розробка дизайну мобільних застосунків. Сім тонкостей, про котрі ви повинні знати URL: https://artjoker.ua/ru/blog/razrabotka-dizayna-mobilnykhprilozheniy-7-tonkostey-o-kotorykh-vy-dolzhny-znat/
9. Роджерс Р., Ломбардо Д. Android. Розробка застосунків, Роджерс Р., Ломбардо Д. – М.: ЕКОМ Паблішерз, 2010. – 400с.
10. Брайн Харді, Білл Філліпс Android. Програмування для професіоналів. - СПб .:Пітер, 2016. – 640 с.
11. Кармен Делессіо, Лорен Дерсі, Шейн Кондер Створення додатків для Android за 24 години – М .: Ескмо, 2015. – 528с
12. Collis B. Flexible learning in a digital world: experiences and expectations, Betty Collis, Jef Moonen. – London : Kogan Page Limited, 2011. – 231 p.
13. Nilanchala Panigrahy. Xamarin Mobile Application Development for Android Second Edition, Panigrahy N. – Birmingham : Packt, 2015. – 296 c

14..Рік Роджерс — Android. Розробка застосунків, Роджерс Рік. — М.: Еком, 2010. — 130 с.

15.Цехнер Маріо — Програмування під Android, Маріо Цехнер. — М.: Пітер, 2012. — 688 с.

16.П. Дейтел — Android для програмістів. Створюємо застосунки / П.Дейтел — М.: Пітер, 2012. — 560 с.

ДОДАТКИ

ДОДАТОК А

MainActivity.java:

```
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import com.danylo.todolister.com.ListFrags.ListMainFrag;

public class MainActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
     getSupportFragmentManager().beginTransaction()
           .setCustomAnimations(android.R.animator.fade_in,android.R.animator.fade_out)
           .replace(R.id.list_frames,new ListMainFrag()).commit();
  }
}
```

ListMainFrag.java:

```
package com.danylo.todolister.com.ListFrags;

import android.os.Bundle;

import androidx.activity.OnBackPressedCallback;
import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProviders;
import androidx.recyclerview.widget.LinearLayoutManager;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.danylo.todolister.com.ListDatabasePack.CaseAdapter;
import com.danylo.todolister.com.ListDatabasePack.ViewModelList;
import com.danylo.todolister.com.R;
import com.danylo.todolister.com.databinding.FragmentListMainBinding;


public class ListMainFrag extends Fragment {


  private ViewModelList viewModelList;
  private CaseAdapter caseAdapter;
  private FragmentListMainBinding fragmentListMainBinding;
  @Override
  public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
     fragmentListMainBinding=FragmentListMainBinding.inflate(inflater,container,false);
     setCaseListeners();
     return fragmentListMainBinding.getRoot();
  }

  @Override
  public void onDestroyView() {
     super.onDestroyView();
     fragmentListMainBinding=null;
  }
  private void setCaseListeners(){
     requireActivity().getOnBackPressedDispatcher().addCallback(new OnBackPressedCallback(true) {
        @Override
        public void handleOnBackPressed() {
           requireActivity().finish();
        }
     });
     fragmentListMainBinding.caseButtonsRel.setOnClickListener(view ->
           requireActivity().getSupportFragmentManager().beginTransaction()
           .setCustomAnimations(android.R.animator.fade_in,android.R.animator.fade_out)
           .replace(R.id.list_frames,new CreateCaseFrag()).commit());
     fragmentListMainBinding.casesRecycler.setHasFixedSize(true);
     fragmentListMainBinding.casesRecycler.setLayoutManager(new LinearLayoutManager(getContext()));
```

```
        caseAdapter=new CaseAdapter(getContext());
        viewModelList= ViewModelProviders.of(requireActivity()).get(ViewModelList.class);
        viewModelList.getAllCaseItems().observe(getViewLifecycleOwner(),caseItems -> caseAdapter.setCaseItems(caseItems));
        fragmentListMainBinding.casesRecycler.setAdapter(caseAdapter);
    }
}
```

CreateCaseFrag.java:

```
package com.danylo.todolister.com.ListFrags;

import android.os.Bundle;

import androidx.activity.OnBackPressedCallback;
import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProviders;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.danylo.todolister.com.ListDatabasePack.CaseItem;
import com.danylo.todolister.com.ListDatabasePack.ViewModelList;
import com.danylo.todolister.com.R;
import com.danylo.todolister.com.databinding.FragmentCreateCaseBinding;


public class CreateCaseFrag extends Fragment {

    private FragmentCreateCaseBinding fragmentCreateCaseBinding;
    private ViewModelList viewModelList;

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        fragmentCreateCaseBinding=FragmentCreateCaseBinding.inflate(inflater,container,false);
        viewModelList=ViewModelProviders.of(requireActivity()).get(ViewModelList.class);
        setCreateCaseListeners();
        return fragmentCreateCaseBinding.getRoot();
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        fragmentCreateCaseBinding=null;
    }
    private void setCreateCaseListeners(){
        requireActivity().getOnBackPressedDispatcher().addCallback(new OnBackPressedCallback(true) {
            @Override
            public void handleOnBackPressed() {
                requireActivity().getSupportFragmentManager().beginTransaction()
                        .setCustomAnimations(android.R.animator.fade_in,android.R.animator.fade_out)
                        .replace(R.id.list_frames,new ListMainFrag()).commit();
            }
        });
        fragmentCreateCaseBinding.addCaseBtn.setOnClickListener(view -> {
            String caseName=fragmentCreateCaseBinding.caseCreateEditValue.getText().toString();
            String caseStatus=fragmentCreateCaseBinding.caseCreateStatusValue.getText().toString();
            if (caseName.trim().equals("")||caseStatus.trim().equals("")){
                Toast.makeText(getContext(), "Please, insert Case Values", Toast.LENGTH_SHORT).show();
            }else {
                viewModelList.insertCase(new CaseItem(caseName,caseStatus));
                requireActivity().getSupportFragmentManager().beginTransaction()
                        .setCustomAnimations(android.R.animator.fade_in,android.R.animator.fade_out)
                        .replace(R.id.list_frames,new ListMainFrag()).commit();
            }
        });
    }
}
```

TaskFrag.java

```
import static android.content.Context.MODE_PRIVATE;

import android.os.Bundle;

import androidx.activity.OnBackPressedCallback;
import androidx.annotation.NonNull;
```

```
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProviders;
import androidx.recyclerview.widget.LinearLayoutManager;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.danylo.todolister.com.ListDatabasePack.CaseAdapter;
import com.danylo.todolister.com.ListDatabasePack.TaskAdapter;
import com.danylo.todolister.com.ListDatabasePack.ViewModelList;
import com.danylo.todolister.com.R;
import com.danylo.todolister.com.databinding.FragmentTasksBinding;


public class TasksFrag extends Fragment {

  private ViewModelList viewModelList;
  private TaskAdapter taskAdapter;
  private FragmentTasksBinding fragmentTasksBinding;

  @Override
  public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
     fragmentTasksBinding=FragmentTasksBinding.inflate(inflater,container,false);
     setTaskFrag();
     return fragmentTasksBinding.getRoot();
  }

  @Override
  public void onDestroyView() {
     super.onDestroyView();
     fragmentTasksBinding=null;
  }
  private void setTaskFrag(){
     String topic=requireActivity().getSharedPreferences("CASE_PREFS", MODE_PRIVATE).getString("caseName", "");
     requireActivity().getOnBackPressedDispatcher().addCallback(new OnBackPressedCallback(true) {
        @Override
        public void handleOnBackPressed() {
           requireActivity().getSupportFragmentManager().beginTransaction()
                 .setCustomAnimations(android.R.animator.fade_in,android.R.animator.fade_out)
                 .replace(R.id.list_frames,new ListMainFrag()).commit();
        }
     });

     fragmentTasksBinding.caseTopicText.setText(topic);
     fragmentTasksBinding.addTasksBtn.setOnClickListener(view ->
           requireActivity().getSupportFragmentManager().beginTransaction()
                 .setCustomAnimations(android.R.animator.fade_in,android.R.animator.fade_out)
                 .replace(R.id.list_frames,new CreateTaskFrag()).commit());
     fragmentTasksBinding.tasksRecycler.setHasFixedSize(true);
     fragmentTasksBinding.tasksRecycler.setLayoutManager(new LinearLayoutManager(getContext()));
     taskAdapter=new TaskAdapter(getContext());
     viewModelList= ViewModelProviders.of(requireActivity()).get(ViewModelList.class);
     viewModelList.getAllCaseTasks(topic).observe(getViewLifecycleOwner(),taskItems -> taskAdapter.setTaskItems(taskItems));
     fragmentTasksBinding.tasksRecycler.setAdapter(taskAdapter);
  }
}
CreateTaskFrag.java
import static android.content.Context.MODE_PRIVATE;

import android.os.Bundle;

import androidx.activity.OnBackPressedCallback;
import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProviders;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.danylo.todolister.com.ListDatabasePack.CaseItem;
import com.danylo.todolister.com.ListDatabasePack.TaskItem;
import com.danylo.todolister.com.ListDatabasePack.ViewModelList;
```

```
import com.danylo.todolister.com.R;
import com.danylo.todolister.com.databinding.FragmentCreateTaskBinding;


public class CreateTaskFrag extends Fragment {

  private FragmentCreateTaskBinding fragmentCreateTaskBinding;
  private ViewModelList viewModelList;
  @Override
  public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
     fragmentCreateTaskBinding=FragmentCreateTaskBinding.inflate(inflater,container,false);
     viewModelList= ViewModelProviders.of(requireActivity()).get(ViewModelList.class);
     setCreateTasksListeners();
     return fragmentCreateTaskBinding.getRoot();
  }

  @Override
  public void onDestroyView() {
     super.onDestroyView();
     fragmentCreateTaskBinding=null;
  }
  private void setCreateTasksListeners(){
     requireActivity().getOnBackPressedDispatcher().addCallback(new OnBackPressedCallback(true) {
        @Override
        public void handleOnBackPressed() {
           requireActivity().getSupportFragmentManager().beginTransaction()
                 .setCustomAnimations(android.R.animator.fade_in,android.R.animator.fade_out)
                 .replace(R.id.list_frames,new ListMainFrag()).commit();
        }
     });
     fragmentCreateTaskBinding.addTaskBtn.setOnClickListener(view -> {
        String task_case=requireActivity().getSharedPreferences("CASE_PREFS", MODE_PRIVATE).getString("caseName", "");
        String task_name=fragmentCreateTaskBinding.taskCreateEditValue.getText().toString();
        String task_info=fragmentCreateTaskBinding.addTaskInfoValue.getText().toString();
        String task_executor=fragmentCreateTaskBinding.taskDoingByValue.getText().toString();
        String task_date=fragmentCreateTaskBinding.taskEndDayValue.getText().toString();
        String task_status=fragmentCreateTaskBinding.taskStatusValue.getText().toString();
        if (task_case.trim().equals("")||task_name.trim().equals("")||task_info.trim().equals("")||task_executor.trim().equals("")
        ||task_date.trim().equals("")||task_status.trim().equals("")){
           Toast.makeText(getContext(), "Please, insert Task Values", Toast.LENGTH_SHORT).show();
        }else {
           viewModelList.insertTask(new TaskItem(task_name,task_info,task_executor,task_date,task_status,task_case));
           requireActivity().getSupportFragmentManager().beginTransaction()
                 .setCustomAnimations(android.R.animator.fade_in,android.R.animator.fade_out)
                 .replace(R.id.list_frames,new TasksFrag()).commit();
        }
     });
  }
}
TaskDetailFrag.java
import static android.content.Context.MODE_PRIVATE;

import android.os.Bundle;

import androidx.activity.OnBackPressedCallback;
import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.danylo.todolister.com.ListDatabasePack.TaskItem;
import com.danylo.todolister.com.ListDatabasePack.ViewModelList;
import com.danylo.todolister.com.R;
import com.danylo.todolister.com.databinding.FragmentTaskDetailBinding;


public class TaskDetailFrag extends Fragment {

  private FragmentTaskDetailBinding fragmentTaskDetailBinding;

  @Override
  public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
```

```
        fragmentTaskDetailBinding=FragmentTaskDetailBinding.inflate(inflater,container,false);
        setDetailFrag();
        return fragmentTaskDetailBinding.getRoot();
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        fragmentTaskDetailBinding=null;
    }
    private void setDetailFrag(){
        requireActivity().getOnBackPressedDispatcher().addCallback(new OnBackPressedCallback(true) {
            @Override
            public void handleOnBackPressed() {
                requireActivity().getSupportFragmentManager().beginTransaction()
                        .setCustomAnimations(android.R.animator.fade_in,android.R.animator.fade_out)
                        .replace(R.id.list_frames,new TasksFrag()).commit();
            }
        });
        String task_name = requireActivity().getSharedPreferences("CASE_PREFS", MODE_PRIVATE).getString("task_name", "");
        String task_status = requireActivity().getSharedPreferences("CASE_PREFS", MODE_PRIVATE).getString("task_status", "");
        String task_info = requireActivity().getSharedPreferences("CASE_PREFS", MODE_PRIVATE).getString("task_info", "");
        String task_executor = requireActivity().getSharedPreferences("CASE_PREFS", MODE_PRIVATE).getString("task_executor", "");
        String task_date = requireActivity().getSharedPreferences("CASE_PREFS", MODE_PRIVATE).getString("task_date", "");

        fragmentTaskDetailBinding.taskDetailTopic.setText(task_name);
        fragmentTaskDetailBinding.taskDetailText.setText(task_info);
        fragmentTaskDetailBinding.taskDoingByValue.setText(task_executor);
        fragmentTaskDetailBinding.taskEndValue.setText(task_date);
        fragmentTaskDetailBinding.taskStatus.setText(task_status);

        fragmentTaskDetailBinding.taskEditBtn.setOnClickListener(view ->
                requireActivity().getSupportFragmentManager().beginTransaction()
                .setCustomAnimations(android.R.animator.fade_in,android.R.animator.fade_out)
                .replace(R.id.list_frames,new UpdateTaskFrag()).commit());
    }
}
```

UpdateTaskFrag.java

```
import static android.content.Context.MODE_PRIVATE;

import android.os.AsyncTask;
import android.os.Bundle;

import androidx.activity.OnBackPressedCallback;
import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProviders;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.danylo.todolister.com.ListDatabasePack.TaskItem;
import com.danylo.todolister.com.ListDatabasePack.ViewModelList;
import com.danylo.todolister.com.R;
import com.danylo.todolister.com.databinding.FragmentUpdateTaskBinding;


public class UpdateTaskFrag extends Fragment {

    private FragmentUpdateTaskBinding fragmentUpdateTaskBinding;
    private String task_case,task_name,task_status,task_info,task_executor,task_date;
    private ViewModelList viewModelList;
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        fragmentUpdateTaskBinding=FragmentUpdateTaskBinding.inflate(inflater,container,false);
        viewModelList= ViewModelProviders.of(requireActivity()).get(ViewModelList.class);
        setUpdate();
        return fragmentUpdateTaskBinding.getRoot();
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
```

```
        fragmentUpdateTaskBinding=null;
    }
    private void setUpdate(){
        requireActivity().getOnBackPressedDispatcher().addCallback(new OnBackPressedCallback(true) {
            @Override
            public void handleOnBackPressed() {
                requireActivity().getSupportFragmentManager().beginTransaction()
                        .setCustomAnimations(android.R.animator.fade_in,android.R.animator.fade_out)
                        .replace(R.id.list_frames,new TasksFrag()).commit();
            }
        });
        task_case=requireActivity().getSharedPreferences("CASE_PREFS", MODE_PRIVATE).getString("caseName", "");
        task_name=requireActivity().getSharedPreferences("CASE_PREFS", MODE_PRIVATE).getString("task_name", "");
        task_status=requireActivity().getSharedPreferences("CASE_PREFS", MODE_PRIVATE).getString("task_status", "");
        task_info=requireActivity().getSharedPreferences("CASE_PREFS", MODE_PRIVATE).getString("task_info", "");
        task_executor=requireActivity().getSharedPreferences("CASE_PREFS", MODE_PRIVATE).getString("task_executor", "");
        task_date=requireActivity().getSharedPreferences("CASE_PREFS", MODE_PRIVATE).getString("task_date", "");

        fragmentUpdateTaskBinding.taskUpdateEditValue.setText(task_name);
        fragmentUpdateTaskBinding.updateTaskInfoValue.setText(task_info);
        fragmentUpdateTaskBinding.updateTaskDoingByValue.setText(task_executor);
        fragmentUpdateTaskBinding.updateTaskEndDayValue.setText(task_date);
        fragmentUpdateTaskBinding.updateTaskProgressValue.setText(task_status);

        fragmentUpdateTaskBinding.updateTaskBtn.setOnClickListener(view -> {
            task_status=fragmentUpdateTaskBinding.updateTaskProgressValue.getText().toString();
            task_info=fragmentUpdateTaskBinding.updateTaskInfoValue.getText().toString();
            task_executor=fragmentUpdateTaskBinding.updateTaskDoingByValue.getText().toString();
            task_date=fragmentUpdateTaskBinding.updateTaskEndDayValue.getText().toString();

            if (task_case.trim().equals("")||task_name.trim().equals("")||task_info.trim().equals("")||task_executor.trim().equals("")
                    ||task_date.trim().equals("")||task_status.trim().equals("")){
                Toast.makeText(getContext(), "Please, insert Task Values", Toast.LENGTH_SHORT).show();
            }else {
                AsyncTask.execute(() -> viewModelList.updateSinfleTask(task_name,task_info,task_executor,task_date,task_status));
                requireActivity().getSupportFragmentManager().beginTransaction()
                        .setCustomAnimations(android.R.animator.fade_in,android.R.animator.fade_out)
                        .replace(R.id.list_frames,new TasksFrag()).commit();
            }
        });
    }
}
```

activity_main.xml

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

fragment_list_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ListFrags.ListMainFrag"
    android:background="@color/white">
    <RelativeLayout
        android:id="@+id/cases_topic_rel"
        android:layout_width="match_parent"
        android:layout_height="70dp"
        android:layout_alignParentTop="true"
        android:background="@color/dark_pink">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
        android:textSize="30sp"
        android:textColor="@color/black"
        android:text="@string/case_list"
        android:layout_centerVertical="true"
        android:layout_alignParentStart="true"
        android:layout_marginStart="30dp"
        android:textStyle="bold"/>
  </RelativeLayout>

  <androidx.recyclerview.widget.RecyclerView
     android:id="@+id/cases_recycler"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     android:layout_above="@+id/case_buttons_rel"
     android:layout_below="@+id/cases_topic_rel"
     android:layout_marginVertical="10dp"
     android:fadeScrollbars="false"
     android:scrollbarSize="12dp"
     android:scrollbarStyle="outsideOverlay"
     android:scrollbars="horizontal"
     android:splitMotionEvents="false" />
     <RelativeLayout
        android:id="@+id/case_buttons_rel"
        android:layout_width="match_parent"
        android:layout_height="70dp"
        android:layout_alignParentBottom="true"
        android:background="@drawable/btn_shape">
        <TextView
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:textSize="30sp"
           android:textColor="@color/black"
           android:layout_centerInParent="true"
           android:gravity="center"
           android:text="@string/add_new_case"
           android:textStyle="bold"/>
     </RelativeLayout>

</RelativeLayout>

fragment_create_case.xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".ListFrags.CreateCaseFrag"
  android:background="@color/white">
  <TextView
     android:id="@+id/case_create_text"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:textSize="22sp"
     android:text="@string/case_name"
     android:textColor="@color/black"
     android:layout_marginTop="30dp"
     android:layout_alignParentStart="true"
     android:layout_marginStart="20dp"
     android:textStyle="bold"/>
  <EditText
     android:id="@+id/case_create_edit_value"
     android:layout_width="match_parent"
     android:layout_height="40dp"
     android:maxLength="20"
     android:textColor="@color/black"
     android:paddingStart="10dp"
     android:textAlignment="textStart"
     android:textSize="16sp"
     android:layout_below="@+id/case_create_text"
     android:layout_marginHorizontal="20dp"
     android:background="#DDDBDB"
     tools:ignore="RtlSymmetry" />
  <TextView
     android:id="@+id/case_create_status_text"
```

```
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="22sp"
      android:text="@string/case_status"
      android:textColor="@color/black"
      android:layout_below="@+id/case_create_edit_value"
      android:layout_marginTop="30dp"
      android:layout_alignParentStart="true"
      android:layout_marginStart="20dp"
      android:textStyle="bold"/>
   <EditText
      android:id="@+id/case_create_status_value"
      android:layout_width="match_parent"
      android:layout_height="40dp"
      android:maxLength="20"
      android:textColor="@color/black"
      android:paddingStart="10dp"
      android:textAlignment="textStart"
      android:textSize="16sp"
      android:layout_below="@+id/case_create_status_text"
      android:layout_marginHorizontal="20dp"
      android:background="#DDDBDB"
      tools:ignore="RtlSymmetry" />

   <RelativeLayout
      android:id="@+id/add_case_btn"
      android:layout_width="match_parent"
      android:layout_height="70dp"
      android:layout_alignParentBottom="true"
      android:background="@drawable/btn_shape">
      <TextView
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:textSize="30sp"
         android:textColor="@color/black"
         android:layout_centerInParent="true"
         android:gravity="center"
         android:text="@string/add_case"
         android:textStyle="bold"/>
   </RelativeLayout>

</RelativeLayout>

case_item.xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:background="@color/white">
   <RelativeLayout
      android:id="@+id/tasks_topic_rel"
      android:layout_width="match_parent"
      android:layout_height="70dp"
      android:layout_alignParentTop="true"
      android:background="@drawable/item_shape"
      android:layout_marginHorizontal="10dp">

      <TextView
         android:id="@+id/case_name_text"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignParentStart="true"
         android:layout_marginStart="30dp"
         android:layout_marginTop="5dp"
         android:text="@string/add_case"
         android:textColor="@color/black"
         android:textSize="22sp"
         android:textStyle="bold" />

      <TextView
         android:id="@+id/case_status_text"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_below="@+id/case_name_text"
         android:layout_alignParentStart="true"
```

```
        android:layout_marginStart="30dp"
        android:layout_marginTop="3dp"
        android:text="@string/add_case"
        android:textColor="@color/black"
        android:textSize="16sp" />
  </RelativeLayout>
</RelativeLayout>
```

fragment_tasks.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".ListFrags.TasksFrag"
  android:background="@color/white">
  <RelativeLayout
    android:id="@+id/tasks_topic_rel"
    android:layout_width="match_parent"
    android:layout_height="70dp"
    android:layout_alignParentTop="true"
    android:background="@color/dark_pink">
    <TextView
      android:id="@+id/case_topic_text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="30sp"
      android:text="@string/add_task"
      android:textColor="@color/black"
      android:layout_centerVertical="true"
      android:layout_alignParentStart="true"
      android:layout_marginStart="30dp"
      android:textStyle="bold"/>
  </RelativeLayout>
  <androidx.recyclerview.widget.RecyclerView
    android:id="@+id/tasks_recycler"
    android:layout_below="@+id/tasks_topic_rel"
    android:splitMotionEvents="false"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scrollbars="horizontal"
    android:layout_marginVertical="10dp"
    android:fadeScrollbars="false"
    android:scrollbarSize="12dp"
    android:scrollbarStyle="outsideOverlay"
    android:paddingBottom="30dp"
    android:layout_above="@+id/add_tasks_btn"/>
    <RelativeLayout
      android:id="@+id/add_tasks_btn"
      android:layout_width="match_parent"
      android:layout_height="70dp"
      android:layout_alignParentBottom="true"
      android:background="@drawable/btn_shape">
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30sp"
        android:textColor="@color/black"
        android:layout_centerInParent="true"
        android:gravity="center"
        android:text="@string/add_new_task"
        android:textStyle="bold"/>
    </RelativeLayout>
</RelativeLayout>
```

fragment_create_task.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".ListFrags.CreateTaskFrag"
  android:background="@color/white">
  <ScrollView
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_above="@+id/add_task_btn">
  <RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <TextView
      android:id="@+id/task_create_text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="22sp"
      android:text="@string/task_name"
      android:textColor="@color/black"
      android:layout_marginTop="30dp"
      android:layout_alignParentStart="true"
      android:layout_marginStart="20dp"
      android:textStyle="bold"/>
    <EditText
      android:id="@+id/task_create_edit_value"
      android:layout_width="match_parent"
      android:layout_height="40dp"
      android:maxLength="15"
      android:textColor="@color/black"
      android:paddingStart="10dp"
      android:textAlignment="textStart"
      android:textSize="16sp"
      android:layout_below="@+id/task_create_text"
      android:layout_marginHorizontal="20dp"
      android:background="#DDDBDB"
      tools:ignore="RtlSymmetry" />
    <TextView
      android:id="@+id/add_task_info"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="22sp"
      android:text="@string/task_info"
      android:textColor="@color/black"
      android:layout_below="@+id/task_create_edit_value"
      android:layout_marginTop="20dp"
      android:layout_alignParentStart="true"
      android:layout_marginStart="20dp"
      android:textStyle="bold"/>
    <EditText
      android:id="@+id/add_task_info_value"
      android:layout_width="match_parent"
      android:layout_height="140dp"
      android:textColor="@color/black"
      android:paddingStart="10dp"
      android:padding="5dp"
      android:gravity="start"
      android:textAlignment="textStart"
      android:textSize="16sp"
      android:layout_below="@+id/add_task_info"
      android:layout_marginHorizontal="20dp"
      android:background="#DDDBDB"
      tools:ignore="RtlSymmetry" />
    <TextView
      android:id="@+id/task_doing_by"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="22sp"
      android:text="@string/doing"
      android:textColor="@color/black"
      android:layout_below="@+id/add_task_info_value"
      android:layout_marginTop="20dp"
      android:layout_alignParentStart="true"
      android:layout_marginStart="20dp"
      android:textStyle="bold"/>
    <EditText
      android:id="@+id/task_doing_by_value"
      android:layout_width="match_parent"
      android:layout_height="40dp"
      android:maxLength="15"
      android:textColor="@color/black"
      android:paddingStart="10dp"
```

```
        android:textAlignment="textStart"
        android:textSize="16sp"
        android:layout_below="@+id/task_doing_by"
        android:layout_marginHorizontal="20dp"
        android:background="#DDDBDB"
        tools:ignore="RtlSymmetry" />
    <TextView
        android:id="@+id/task_end_day_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="22sp"
        android:text="@string/end"
        android:textColor="@color/black"
        android:layout_below="@+id/task_doing_by_value"
        android:layout_marginTop="20dp"
        android:layout_alignParentStart="true"
        android:layout_marginStart="20dp"
        android:textStyle="bold"/>
    <EditText
        android:id="@+id/task_end_day_value"
        android:layout_width="match_parent"
        android:layout_height="40dp"
        android:maxLength="10"
        android:textColor="@color/black"
        android:paddingStart="10dp"
        android:textAlignment="textStart"
        android:textSize="16sp"
        android:layout_below="@+id/task_end_day_text"
        android:layout_marginHorizontal="20dp"
        android:background="#DDDBDB"
        tools:ignore="RtlSymmetry" />
    <TextView
        android:id="@+id/task_progress_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="22sp"
        android:text="@string/tast_status"
        android:textColor="@color/black"
        android:layout_below="@+id/task_end_day_value"
        android:layout_marginTop="20dp"
        android:layout_alignParentStart="true"
        android:layout_marginStart="20dp"
        android:textStyle="bold"/>
    <EditText
        android:id="@+id/task_status_value"
        android:layout_width="match_parent"
        android:layout_height="40dp"
        android:maxLength="10"
        android:textColor="@color/black"
        android:paddingStart="10dp"
        android:textAlignment="textStart"
        android:textSize="16sp"
        android:layout_below="@+id/task_progress_text"
        android:layout_marginHorizontal="20dp"
        android:background="#DDDBDB"
        tools:ignore="RtlSymmetry" />
  </RelativeLayout>
  </ScrollView>
  <RelativeLayout
    android:id="@+id/add_task_btn"
    android:layout_width="match_parent"
    android:layout_height="70dp"
    android:layout_alignParentBottom="true"
    android:background="@drawable/btn_shape">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30sp"
        android:textColor="@color/black"
        android:layout_centerInParent="true"
        android:gravity="center"
        android:text="@string/add_task"
        android:textStyle="bold"/>
  </RelativeLayout>
</RelativeLayout>
```

task_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:background="@color/white">
  <RelativeLayout
    android:id="@+id/tasks_topic_rel"
    android:layout_width="match_parent"
    android:layout_height="70dp"
    android:layout_alignParentTop="true"
    android:background="@drawable/item_shape"
    android:layout_marginHorizontal="10dp">
    <TextView
      android:id="@+id/task_name_text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="22sp"
      android:textColor="@color/black"
      android:layout_marginTop="5dp"
      android:layout_alignParentStart="true"
      android:text="@string/add_task"
      android:layout_marginStart="30dp"
      android:textStyle="bold"/>
    <TextView
      android:id="@+id/task_status_text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@+id/task_name_text"
      android:textSize="16sp"
      android:layout_marginTop="3dp"
      android:textColor="@color/black"
      android:layout_alignParentStart="true"
      android:text="@string/add_task"
      android:layout_marginStart="30dp" />
    <TextView
      android:id="@+id/task_end_text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="16sp"
      android:layout_marginTop="5dp"
      android:textColor="@color/black"
      android:layout_alignParentEnd="true"
      android:layout_marginEnd="10dp"
      android:text="@string/end"
      android:layout_marginStart="30dp"/>
  </RelativeLayout>
</RelativeLayout>
```

fragment_task_detail.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".ListFrags.TaskDetailFrag"
  android:background="@color/white">
  <TextView
    android:id="@+id/task_detail_topic"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="30sp"
    android:text="@string/add_task"
    android:textColor="@color/black"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="10dp"
    android:textStyle="bold"/>
  <TextView
    android:id="@+id/task_detail_info"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="22sp"
    android:text="@string/info"
```

```
    android:textColor="@color/black"
    android:layout_below="@+id/task_detail_topic"
    android:layout_marginTop="10dp"
    android:layout_alignParentStart="true"
    android:layout_marginStart="20dp"
    android:textStyle="bold"/>
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingBottom="15dp"
    android:layout_below="@+id/task_detail_info"
    android:layout_marginBottom="-20dp"
    android:layout_marginTop="10dp"
    android:layout_above="@+id/center_detail_task">
    <TextView
        android:id="@+id/task_detail_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="22sp"
        android:text="@string/info"
        android:textColor="@color/black"
        android:layout_marginStart="20dp"/>
</ScrollView>
<TextView
    android:id="@+id/center_detail_task"
    android:layout_centerInParent="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
<TextView
    android:id="@+id/task_doing_by"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="22sp"
    android:text="@string/doing"
    android:layout_below="@+id/center_detail_task"
    android:layout_marginTop="40dp"
    android:textColor="@color/black"
    android:layout_marginStart="20dp"/>
<TextView
    android:id="@+id/task_end"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="22sp"
    android:text="@string/end"
    android:layout_marginTop="40dp"
    android:layout_below="@+id/center_detail_task"
    android:textColor="@color/black"
    android:layout_alignParentEnd="true"
    android:layout_marginEnd="20dp"/>
<TextView
    android:id="@+id/task_doing_by_value"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="22sp"
    android:text="@string/doing"
    android:layout_below="@+id/task_end"
    android:layout_marginTop="10dp"
    android:textColor="@color/black"
    android:textStyle="bold"
    android:layout_marginStart="20dp"/>
<TextView
    android:id="@+id/task_end_value"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="22sp"
    android:text="@string/end"
    android:layout_marginTop="10dp"
    android:textStyle="bold"
    android:layout_below="@+id/task_end"
    android:textColor="#5DA7ED"
    android:layout_alignParentEnd="true"
    android:layout_marginEnd="20dp"/>
<Button
    android:id="@+id/task_edit_btn"
    android:layout_width="wrap_content"
```

```
      android:layout_height="wrap_content"
      android:textSize="22sp"
      android:text="@string/edit"
      android:layout_marginBottom="30dp"
      android:textStyle="bold"
      android:paddingHorizontal="32dp"
      android:paddingVertical="10dp"
      android:background="@drawable/btn_shape"
      android:layout_alignParentBottom="true"
      android:textColor="@color/black"
      android:layout_alignParentEnd="true"
      android:layout_marginEnd="20dp"/>
   <TextView
      android:id="@+id/task_status"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="22sp"
      android:text="@string/doing"
      android:layout_alignParentBottom="true"
      android:layout_marginBottom="30dp"
      android:textColor="@color/black"
      android:textStyle="bold"
      android:layout_marginStart="20dp"/>
</RelativeLayout>

fragment_update_task.xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".ListFrags.UpdateTaskFrag"
   android:background="@color/white">

   <ScrollView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_above="@+id/update_task_btn">
      <RelativeLayout
         android:layout_width="match_parent"
         android:layout_height="wrap_content">
         <TextView
            android:id="@+id/task_update_text"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="22sp"
            android:text="@string/task_name"
            android:textColor="@color/black"
            android:layout_marginTop="30dp"
            android:layout_alignParentStart="true"
            android:layout_marginStart="20dp"
            android:textStyle="bold"/>
         <TextView
            android:id="@+id/task_update_edit_value"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@color/black"
            android:layout_alignParentStart="true"
            android:layout_marginStart="20dp"
            android:textStyle="bold"
            android:text="@string/task_name"
            android:textSize="24sp"
            android:layout_below="@+id/task_update_text"
            android:layout_marginHorizontal="20dp"
            tools:ignore="RtlSymmetry" />
         <TextView
            android:id="@+id/update_task_info"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="22sp"
            android:text="@string/task_info"
            android:textColor="@color/black"
            android:layout_below="@+id/task_update_edit_value"
            android:layout_marginTop="20dp"
```

```
    android:layout_alignParentStart="true"
    android:layout_marginStart="20dp"
    android:textStyle="bold"/>
  <EditText
    android:id="@+id/update__task_info_value"
    android:layout_width="match_parent"
    android:layout_height="140dp"
    android:textColor="@color/black"
    android:paddingStart="10dp"
    android:padding="5dp"
    android:gravity="start"
    android:textAlignment="textStart"
    android:textSize="16sp"
    android:layout_below="@+id/update_task_info"
    android:layout_marginHorizontal="20dp"
    android:background="#DDDBDB"
    tools:ignore="RtlSymmetry" />
  <TextView
    android:id="@+id/update_task_doing_by"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="22sp"
    android:text="@string/doing"
    android:textColor="@color/black"
    android:layout_below="@+id/update__task_info_value"
    android:layout_marginTop="20dp"
    android:layout_alignParentStart="true"
    android:layout_marginStart="20dp"
    android:textStyle="bold"/>
  <EditText
    android:id="@+id/update_task_doing_by_value"
    android:layout_width="match_parent"
    android:layout_height="40dp"
    android:maxLength="15"
    android:textColor="@color/black"
    android:paddingStart="10dp"
    android:textAlignment="textStart"
    android:textSize="16sp"
    android:layout_below="@+id/update_task_doing_by"
    android:layout_marginHorizontal="20dp"
    android:background="#DDDBDB"
    tools:ignore="RtlSymmetry" />
  <TextView
    android:id="@+id/update_task_end_day_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="22sp"
    android:text="@string/end"
    android:textColor="@color/black"
    android:layout_below="@+id/update_task_doing_by_value"
    android:layout_marginTop="20dp"
    android:layout_alignParentStart="true"
    android:layout_marginStart="20dp"
    android:textStyle="bold"/>
  <EditText
    android:id="@+id/update_task_end_day_value"
    android:layout_width="match_parent"
    android:layout_height="40dp"
    android:maxLength="10"
    android:textColor="@color/black"
    android:paddingStart="10dp"
    android:textAlignment="textStart"
    android:textSize="16sp"
    android:layout_below="@+id/update_task_end_day_text"
    android:layout_marginHorizontal="20dp"
    android:background="#DDDBDB"
    tools:ignore="RtlSymmetry" />
  <TextView
    android:id="@+id/update_task_progress_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="22sp"
    android:text="@string/tast_status"
    android:textColor="@color/black"
    android:layout_below="@+id/update_task_end_day_value"
```

```
            android:layout_marginTop="20dp"
            android:layout_alignParentStart="true"
            android:layout_marginStart="20dp"
            android:textStyle="bold"/>
        <EditText
            android:id="@+id/update_task_progress_value"
            android:layout_width="match_parent"
            android:layout_height="40dp"
            android:maxLength="10"
            android:textColor="@color/black"
            android:paddingStart="10dp"
            android:textAlignment="textStart"
            android:textSize="16sp"
            android:layout_below="@+id/update_task_progress_text"
            android:layout_marginHorizontal="20dp"
            android:background="#DDDBDB"
            tools:ignore="RtlSymmetry" />
    </RelativeLayout>
  </ScrollView>
  <RelativeLayout
    android:id="@+id/update_task_btn"
    android:layout_width="match_parent"
    android:layout_height="70dp"
    android:layout_alignParentBottom="true"
    android:background="@drawable/btn_shape">
    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="30sp"
      android:textColor="@color/black"
      android:layout_centerInParent="true"
      android:gravity="center"
      android:text="@string/update_task"
      android:textStyle="bold"/>
  </RelativeLayout>

</RelativeLayout>
```

DAO_List_Impl.java

```
import android.database.Cursor;
import androidx.lifecycle.LiveData;
import androidx.room.EntityDeletionOrUpdateAdapter;
import androidx.room.EntityInsertionAdapter;
import androidx.room.RoomDatabase;
import androidx.room.RoomSQLiteQuery;
import androidx.room.SharedSQLiteStatement;
import androidx.room.util.CursorUtil;
import androidx.room.util.DBUtil;
import androidx.sqlite.db.SupportSQLiteStatement;
import java.lang.Class;
import java.lang.Exception;
import java.lang.Override;
import java.lang.String;
import java.lang.SuppressWarnings;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.concurrent.Callable;

@SuppressWarnings({"unchecked", "deprecation"})
public final class DAO_List_Impl implements DAO_List {
 private final RoomDatabase __db;

 private final EntityInsertionAdapter<CaseItem> __insertionAdapterOfCaseItem;

 private final EntityInsertionAdapter<TaskItem> __insertionAdapterOfTaskItem;

 private final EntityDeletionOrUpdateAdapter<TaskItem> __updateAdapterOfTaskItem;

 private final SharedSQLiteStatement __preparedStmtOfUpdateSingleTask;

 public DAO_List_Impl(RoomDatabase __db) {
   this.__db = __db;
   this.__insertionAdapterOfCaseItem = new EntityInsertionAdapter<CaseItem>(__db) {
     @Override
     public String createQuery() {
```

```
      return "INSERT OR IGNORE INTO `Cases_Table` (`case_name`,`case_status`) VALUES (?,?)";
    }

    @Override
    public void bind(SupportSQLiteStatement stmt, CaseItem value) {
      if (value.getCase_name() == null) {
        stmt.bindNull(1);
      } else {
        stmt.bindString(1, value.getCase_name());
      }
      if (value.getCase_status() == null) {
        stmt.bindNull(2);
      } else {
        stmt.bindString(2, value.getCase_status());
      }
    }
  };
  this.__insertionAdapterOfTaskItem = new EntityInsertionAdapter<TaskItem>(__db) {
    @Override
    public String createQuery() {
      return "INSERT OR IGNORE INTO `Tasks_Table` (`task_name`,`task_info`,`task_executor`,`task_date`,`task_status`,`task_case`)
VALUES (?,?,?,?,?,?,?)";
    }

    @Override
    public void bind(SupportSQLiteStatement stmt, TaskItem value) {
      if (value.getTask_name() == null) {
        stmt.bindNull(1);
      } else {
        stmt.bindString(1, value.getTask_name());
      }
      if (value.getTask_info() == null) {
        stmt.bindNull(2);
      } else {
        stmt.bindString(2, value.getTask_info());
      }
      if (value.getTask_executor() == null) {
        stmt.bindNull(3);
      } else {
        stmt.bindString(3, value.getTask_executor());
      }
      if (value.getTask_date() == null) {
        stmt.bindNull(4);
      } else {
        stmt.bindString(4, value.getTask_date());
      }
      if (value.getTask_status() == null) {
        stmt.bindNull(5);
      } else {
        stmt.bindString(5, value.getTask_status());
      }
      if (value.getTask_case() == null) {
        stmt.bindNull(6);
      } else {
        stmt.bindString(6, value.getTask_case());
      }
    }
  };
  this.__updateAdapterOfTaskItem = new EntityDeletionOrUpdateAdapter<TaskItem>(__db) {
    @Override
    public String createQuery() {
      return "UPDATE OR ABORT `Tasks_Table` SET `task_name` = ?,`task_info` = ?,`task_executor` = ?,`task_date` = ?,`task_status` =
?,`task_case` = ? WHERE `task_name` = ?";
    }

    @Override
    public void bind(SupportSQLiteStatement stmt, TaskItem value) {
      if (value.getTask_name() == null) {
        stmt.bindNull(1);
      } else {
        stmt.bindString(1, value.getTask_name());
      }
      if (value.getTask_info() == null) {
        stmt.bindNull(2);
      } else {
```

```
        stmt.bindString(2, value.getTask_info());
      }
      if (value.getTask_executor() == null) {
        stmt.bindNull(3);
      } else {
        stmt.bindString(3, value.getTask_executor());
      }
      if (value.getTask_date() == null) {
        stmt.bindNull(4);
      } else {
        stmt.bindString(4, value.getTask_date());
      }
      if (value.getTask_status() == null) {
        stmt.bindNull(5);
      } else {
        stmt.bindString(5, value.getTask_status());
      }
      if (value.getTask_case() == null) {
        stmt.bindNull(6);
      } else {
        stmt.bindString(6, value.getTask_case());
      }
      if (value.getTask_name() == null) {
        stmt.bindNull(7);
      } else {
        stmt.bindString(7, value.getTask_name());
      }
    }
  };
  this.__preparedStmtOfUpdateSingleTask = new SharedSQLiteStatement(__db) {
    @Override
    public String createQuery() {
      final String _query = "UPDATE tasks_table SET task_name= ?,task_info= ?,task_executor= ?,task_date= ?,task_status= ? WHERE
task_name LIKE ?";
      return _query;
    }
  };
 }

 @Override
 public void insertCase(final CaseItem caseItem) {
  __db.assertNotSuspendingTransaction();
  __db.beginTransaction();
  try {
    __insertionAdapterOfCaseItem.insert(caseItem);
    __db.setTransactionSuccessful();
  } finally {
    __db.endTransaction();
  }
 }

 @Override
 public void insertTask(final TaskItem taskItem) {
  __db.assertNotSuspendingTransaction();
  __db.beginTransaction();
  try {
    __insertionAdapterOfTaskItem.insert(taskItem);
    __db.setTransactionSuccessful();
  } finally {
    __db.endTransaction();
  }
 }

 @Override
 public void updateTask(final TaskItem taskItem) {
  __db.assertNotSuspendingTransaction();
  __db.beginTransaction();
  try {
    __updateAdapterOfTaskItem.handle(taskItem);
    __db.setTransactionSuccessful();
  } finally {
    __db.endTransaction();
  }
 }
```

```
@Override
public void updateSingleTask(final String task_name, final String task_info,
    final String task_executor, final String task_date, final String task_status) {
  __db.assertNotSuspendingTransaction();
  final SupportSQLiteStatement _stmt = __preparedStmtOfUpdateSingleTask.acquire();
  int _argIndex = 1;
  if (task_name == null) {
    _stmt.bindNull(_argIndex);
  } else {
    _stmt.bindString(_argIndex, task_name);
  }
  _argIndex = 2;
  if (task_info == null) {
    _stmt.bindNull(_argIndex);
  } else {
    _stmt.bindString(_argIndex, task_info);
  }
  _argIndex = 3;
  if (task_executor == null) {
    _stmt.bindNull(_argIndex);
  } else {
    _stmt.bindString(_argIndex, task_executor);
  }
  _argIndex = 4;
  if (task_date == null) {
    _stmt.bindNull(_argIndex);
  } else {
    _stmt.bindString(_argIndex, task_date);
  }
  _argIndex = 5;
  if (task_status == null) {
    _stmt.bindNull(_argIndex);
  } else {
    _stmt.bindString(_argIndex, task_status);
  }
  _argIndex = 6;
  if (task_name == null) {
    _stmt.bindNull(_argIndex);
  } else {
    _stmt.bindString(_argIndex, task_name);
  }
  __db.beginTransaction();
  try {
    _stmt.executeUpdateDelete();
    __db.setTransactionSuccessful();
  } finally {
    __db.endTransaction();
    __preparedStmtOfUpdateSingleTask.release(_stmt);
  }
}

@Override
public LiveData<List<CaseItem>> getAllCases() {
  final String _sql = "SELECT * FROM Cases_Table";
  final RoomSQLiteQuery _statement = RoomSQLiteQuery.acquire(_sql, 0);
  return __db.getInvalidationTracker().createLiveData(new String[]{"Cases_Table"}, false, new Callable<List<CaseItem>>() {
    @Override
    public List<CaseItem> call() throws Exception {
      final Cursor _cursor = DBUtil.query(__db, _statement, false, null);
      try {
        final int _cursorIndexOfCaseName = CursorUtil.getColumnIndexOrThrow(_cursor, "case_name");
        final int _cursorIndexOfCaseStatus = CursorUtil.getColumnIndexOrThrow(_cursor, "case_status");
        final List<CaseItem> _result = new ArrayList<CaseItem>(_cursor.getCount());
        while(_cursor.moveToNext()) {
          final CaseItem _item;
          final String _tmpCase_name;
          if (_cursor.isNull(_cursorIndexOfCaseName)) {
            _tmpCase_name = null;
          } else {
            _tmpCase_name = _cursor.getString(_cursorIndexOfCaseName);
          }
          final String _tmpCase_status;
          if (_cursor.isNull(_cursorIndexOfCaseStatus)) {
            _tmpCase_status = null;
          } else {
```

```
            _tmpCase_status = _cursor.getString(_cursorIndexOfCaseStatus);
          }
          _item = new CaseItem(_tmpCase_name,_tmpCase_status);
          _result.add(_item);
        }
        return _result;
      } finally {
        _cursor.close();
      }
    }

    @Override
    protected void finalize() {
      _statement.release();
    }
  });
}

@Override
public LiveData<List<TaskItem>> getAllCaseTasks(final String task_case) {
  final String _sql = "SELECT * FROM Tasks_Table WHERE task_case LIKE ?";
  final RoomSQLiteQuery _statement = RoomSQLiteQuery.acquire(_sql, 1);
  int _argIndex = 1;
  if (task_case == null) {
    _statement.bindNull(_argIndex);
  } else {
    _statement.bindString(_argIndex, task_case);
  }
  return __db.getInvalidationTracker().createLiveData(new String[]{"Tasks_Table"}, false, new Callable<List<TaskItem>>() {
    @Override
    public List<TaskItem> call() throws Exception {
      final Cursor _cursor = DBUtil.query(__db, _statement, false, null);
      try {
        final int _cursorIndexOfTaskName = CursorUtil.getColumnIndexOrThrow(_cursor, "task_name");
        final int _cursorIndexOfTaskInfo = CursorUtil.getColumnIndexOrThrow(_cursor, "task_info");
        final int _cursorIndexOfTaskExecutor = CursorUtil.getColumnIndexOrThrow(_cursor, "task_executor");
        final int _cursorIndexOfTaskDate = CursorUtil.getColumnIndexOrThrow(_cursor, "task_date");
        final int _cursorIndexOfTaskStatus = CursorUtil.getColumnIndexOrThrow(_cursor, "task_status");
        final int _cursorIndexOfTaskCase = CursorUtil.getColumnIndexOrThrow(_cursor, "task_case");
        final List<TaskItem> _result = new ArrayList<TaskItem>(_cursor.getCount());
        while(_cursor.moveToNext()) {
          final TaskItem _item;
          final String _tmpTask_name;
          if (_cursor.isNull(_cursorIndexOfTaskName)) {
            _tmpTask_name = null;
          } else {
            _tmpTask_name = _cursor.getString(_cursorIndexOfTaskName);
          }
          final String _tmpTask_info;
          if (_cursor.isNull(_cursorIndexOfTaskInfo)) {
            _tmpTask_info = null;
          } else {
            _tmpTask_info = _cursor.getString(_cursorIndexOfTaskInfo);
          }
          final String _tmpTask_executor;
          if (_cursor.isNull(_cursorIndexOfTaskExecutor)) {
            _tmpTask_executor = null;
          } else {
            _tmpTask_executor = _cursor.getString(_cursorIndexOfTaskExecutor);
          }
          final String _tmpTask_date;
          if (_cursor.isNull(_cursorIndexOfTaskDate)) {
            _tmpTask_date = null;
          } else {
            _tmpTask_date = _cursor.getString(_cursorIndexOfTaskDate);
          }
          final String _tmpTask_status;
          if (_cursor.isNull(_cursorIndexOfTaskStatus)) {
            _tmpTask_status = null;
          } else {
            _tmpTask_status = _cursor.getString(_cursorIndexOfTaskStatus);
          }
          final String _tmpTask_case;
          if (_cursor.isNull(_cursorIndexOfTaskCase)) {
            _tmpTask_case = null;
```

```
        } else {
          _tmpTask_case = _cursor.getString(_cursorIndexOfTaskCase);
        }
        _item = new TaskItem(_tmpTask_name,_tmpTask_info,_tmpTask_executor,_tmpTask_date,_tmpTask_status,_tmpTask_case);
        _result.add(_item);
      }
      return _result;
    } finally {
      _cursor.close();
    }
  }

  @Override
  protected void finalize() {
    _statement.release();
  }
 });
}

public static List<Class<?>> getRequiredConverters() {
  return Collections.emptyList();
}
}
```

DatabaseList_Impl.java

```
import androidx.room.DatabaseConfiguration;
import androidx.room.InvalidationTracker;
import androidx.room.RoomOpenHelper;
import androidx.room.RoomOpenHelper.Delegate;
import androidx.room.RoomOpenHelper.ValidationResult;
import androidx.room.util.DBUtil;
import androidx.room.util.TableInfo;
import androidx.room.util.TableInfo.Column;
import androidx.room.util.TableInfo.ForeignKey;
import androidx.room.util.TableInfo.Index;
import androidx.sqlite.db.SupportSQLiteDatabase;
import androidx.sqlite.db.SupportSQLiteOpenHelper;
import androidx.sqlite.db.SupportSQLiteOpenHelper.Callback;
import androidx.sqlite.db.SupportSQLiteOpenHelper.Configuration;
import java.lang.Class;
import java.lang.Override;
import java.lang.String;
import java.lang.SuppressWarnings;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;

@SuppressWarnings({"unchecked", "deprecation"})
public final class DatabaseList_Impl extends DatabaseList {
 private volatile DAO_List _dAOList;

 @Override
 protected SupportSQLiteOpenHelper createOpenHelper(DatabaseConfiguration configuration) {
  final SupportSQLiteOpenHelper.Callback _openCallback = new RoomOpenHelper(configuration, new RoomOpenHelper.Delegate(1) {
   @Override
   public void createAllTables(SupportSQLiteDatabase _db) {
    _db.execSQL("CREATE TABLE IF NOT EXISTS `Cases_Table` (`case_name` TEXT NOT NULL, `case_status` TEXT NOT NULL,
PRIMARY KEY(`case_name`))");
    _db.execSQL("CREATE TABLE IF NOT EXISTS `Tasks_Table` (`task_name` TEXT NOT NULL, `task_info` TEXT NOT NULL,
`task_executor` TEXT NOT NULL, `task_date` TEXT NOT NULL, `task_status` TEXT NOT NULL, `task_case` TEXT NOT NULL,
PRIMARY KEY(`task_name`))");
    _db.execSQL("CREATE TABLE IF NOT EXISTS room_master_table (id INTEGER PRIMARY KEY,identity_hash TEXT)");
    _db.execSQL("INSERT OR REPLACE INTO room_master_table (id,identity_hash) VALUES(42,
'c7c964e3f9e30877cf178d05a5a3c1a1')");
   }

   @Override
   public void dropAllTables(SupportSQLiteDatabase _db) {
    _db.execSQL("DROP TABLE IF EXISTS `Cases_Table`");
    _db.execSQL("DROP TABLE IF EXISTS `Tasks_Table`");
    if (mCallbacks != null) {
     for (int _i = 0, _size = mCallbacks.size(); _i < _size; _i++) {
      mCallbacks.get(_i).onDestructiveMigration(_db);
     }
```

```
    }
  }

  @Override
  protected void onCreate(SupportSQLiteDatabase _db) {
    if (mCallbacks != null) {
      for (int _i = 0, _size = mCallbacks.size(); _i < _size; _i++) {
        mCallbacks.get(_i).onCreate(_db);
      }
    }
  }

  @Override
  public void onOpen(SupportSQLiteDatabase _db) {
    mDatabase = _db;
    internalInitInvalidationTracker(_db);
    if (mCallbacks != null) {
      for (int _i = 0, _size = mCallbacks.size(); _i < _size; _i++) {
        mCallbacks.get(_i).onOpen(_db);
      }
    }
  }

  @Override
  public void onPreMigrate(SupportSQLiteDatabase _db) {
    DBUtil.dropFtsSyncTriggers(_db);
  }

  @Override
  public void onPostMigrate(SupportSQLiteDatabase _db) {
  }

  @Override
  protected RoomOpenHelper.ValidationResult onValidateSchema(SupportSQLiteDatabase _db) {
    final HashMap<String, TableInfo.Column> _columnsCasesTable = new HashMap<String, TableInfo.Column>(2);
    _columnsCasesTable.put("case_name", new TableInfo.Column("case_name", "TEXT", true, 1, null,
TableInfo.CREATED_FROM_ENTITY));
    _columnsCasesTable.put("case_status", new TableInfo.Column("case_status", "TEXT", true, 0, null,
TableInfo.CREATED_FROM_ENTITY));
    final HashSet<TableInfo.ForeignKey> _foreignKeysCasesTable = new HashSet<TableInfo.ForeignKey>(0);
    final HashSet<TableInfo.Index> _indicesCasesTable = new HashSet<TableInfo.Index>(0);
    final TableInfo _infoCasesTable = new TableInfo("Cases_Table", _columnsCasesTable, _foreignKeysCasesTable, _indicesCasesTable);
    final TableInfo _existingCasesTable = TableInfo.read(_db, "Cases_Table");
    if (! _infoCasesTable.equals(_existingCasesTable)) {
      return new RoomOpenHelper.ValidationResult(false, "Cases_Table(com.danylo.todolister.com.ListDatabasePack.CaseItem).\n"
            + " Expected:\n" + _infoCasesTable + "\n"
            + " Found:\n" + _existingCasesTable);
    }
    final HashMap<String, TableInfo.Column> _columnsTasksTable = new HashMap<String, TableInfo.Column>(6);
    _columnsTasksTable.put("task_name", new TableInfo.Column("task_name", "TEXT", true, 1, null,
TableInfo.CREATED_FROM_ENTITY));
    _columnsTasksTable.put("task_info", new TableInfo.Column("task_info", "TEXT", true, 0, null,
TableInfo.CREATED_FROM_ENTITY));
    _columnsTasksTable.put("task_executor", new TableInfo.Column("task_executor", "TEXT", true, 0, null,
TableInfo.CREATED_FROM_ENTITY));
    _columnsTasksTable.put("task_date", new TableInfo.Column("task_date", "TEXT", true, 0, null,
TableInfo.CREATED_FROM_ENTITY));
    _columnsTasksTable.put("task_status", new TableInfo.Column("task_status", "TEXT", true, 0, null,
TableInfo.CREATED_FROM_ENTITY));
    _columnsTasksTable.put("task_case", new TableInfo.Column("task_case", "TEXT", true, 0, null,
TableInfo.CREATED_FROM_ENTITY));
    final HashSet<TableInfo.ForeignKey> _foreignKeysTasksTable = new HashSet<TableInfo.ForeignKey>(0);
    final HashSet<TableInfo.Index> _indicesTasksTable = new HashSet<TableInfo.Index>(0);
    final TableInfo _infoTasksTable = new TableInfo("Tasks_Table", _columnsTasksTable, _foreignKeysTasksTable, _indicesTasksTable);
    final TableInfo _existingTasksTable = TableInfo.read(_db, "Tasks_Table");
    if (! _infoTasksTable.equals(_existingTasksTable)) {
      return new RoomOpenHelper.ValidationResult(false, "Tasks_Table(com.danylo.todolister.com.ListDatabasePack.TaskItem).\n"
            + " Expected:\n" + _infoTasksTable + "\n"
            + " Found:\n" + _existingTasksTable);
    }
    return new RoomOpenHelper.ValidationResult(true, null);
  }
}, "c7c964e3f9e30877cf178d05a5a3c1a1", "5b63b24f39a177565f15b290c574464b");
final SupportSQLiteOpenHelper.Configuration _sqliteConfig = SupportSQLiteOpenHelper.Configuration.builder(configuration.context)
  .name(configuration.name)
```

```
      .callback(_openCallback)
      .build();
  final SupportSQLiteOpenHelper _helper = configuration.sqliteOpenHelperFactory.create(_sqliteConfig);
  return _helper;
 }

 @Override
 protected InvalidationTracker createInvalidationTracker() {
  final HashMap<String, String> _shadowTablesMap = new HashMap<String, String>(0);
  HashMap<String, Set<String>> _viewTables = new HashMap<String, Set<String>>(0);
  return new InvalidationTracker(this, _shadowTablesMap, _viewTables, "Cases_Table","Tasks_Table");
 }

 @Override
 public void clearAllTables() {
  super.assertNotMainThread();
  final SupportSQLiteDatabase _db = super.getOpenHelper().getWritableDatabase();
  try {
    super.beginTransaction();
    _db.execSQL("DELETE FROM `Cases_Table`");
    _db.execSQL("DELETE FROM `Tasks_Table`");
    super.setTransactionSuccessful();
  } finally {
    super.endTransaction();
    _db.query("PRAGMA wal_checkpoint(FULL)").close();
    if (!_db.inTransaction()) {
      _db.execSQL("VACUUM");
    }
  }
 }

 @Override
 protected Map<Class<?>, List<Class<?>>> getRequiredTypeConverters() {
  final HashMap<Class<?>, List<Class<?>>> _typeConvertersMap = new HashMap<Class<?>, List<Class<?>>>();
  _typeConvertersMap.put(DAO_List.class, DAO_List_Impl.getRequiredConverters());
  return _typeConvertersMap;
 }

 @Override
 public DAO_List dao_list() {
  if (_dAOList != null) {
    return _dAOList;
  } else {
    synchronized(this) {
      if(_dAOList == null) {
        _dAOList = new DAO_List_Impl(this);
      }
      return _dAOList;
    }
  }
 }
}
```

ViewModelList.java

```
import android.app.Application;
import android.os.AsyncTask;

import androidx.annotation.NonNull;
import androidx.lifecycle.AndroidViewModel;
import androidx.lifecycle.LiveData;

import java.util.List;

public class ViewModelList extends AndroidViewModel {

  private DAO_List dao_list;
  private LiveData<List<CaseItem>> allCaseItems;
  private DatabaseList databaseList;
  public ViewModelList(@NonNull Application application) {
      super(application);
      databaseList=DatabaseList.getDatabaseList(application);
      dao_list=databaseList.dao_list();
      allCaseItems=dao_list.getAllCases();
  }
  public void insertTask(TaskItem taskItem){
      new InsertTaskClass(dao_list).execute(taskItem);
```

```
    }
    public void insertCase(CaseItem caseItem){
        new InsertCaseClass(dao_list).execute(caseItem);
    }
    public void updateTask(TaskItem taskItem){
        new UpdateTaskClass(dao_list).execute(taskItem);
    }
    public LiveData<List<CaseItem>> getAllCaseItems(){
        return allCaseItems;
    }
    public LiveData<List<TaskItem>> getAllCaseTasks(String task_case){
        return dao_list.getAllCaseTasks(task_case);
    }
    public void updateSinfleTask(String task_name,String task_info,String task_executor,String task_date,String task_status){
        dao_list.updateSingleTask(task_name,task_info,task_executor,task_date,task_status);
    }

    private class InsertCaseClass extends AsyncTask<CaseItem,Void,Void>{
        DAO_List dao_list;
        public InsertCaseClass(DAO_List dao_list) {
            this.dao_list = dao_list;
        }
        @Override
        protected Void doInBackground(CaseItem... caseItems) {
            dao_list.insertCase(caseItems[0]);
            return null;
        }
    }
    private class InsertTaskClass extends AsyncTask<TaskItem,Void,Void>{
        DAO_List dao_list;

        public InsertTaskClass(DAO_List dao_list) {
            this.dao_list = dao_list;
        }
        @Override
        protected Void doInBackground(TaskItem... taskItems) {
            dao_list.insertTask(taskItems[0]);
            return null;
        }
    }

    private class UpdateTaskClass extends AsyncTask<TaskItem,Void,Void>{
        DAO_List dao_list;

        public UpdateTaskClass(DAO_List dao_list) {
            this.dao_list = dao_list;
        }
        @Override
        protected Void doInBackground(TaskItem... taskItems) {
            dao_list.updateTask(taskItems[0]);
            return null;
        }
    }

}
```

CaseItem.java

```
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.room.ColumnInfo;
import androidx.room.Entity;
import androidx.room.PrimaryKey;

@Entity(tableName = "Cases_Table")
public class CaseItem {
    @PrimaryKey
    @NonNull
    @ColumnInfo(name = "case_name")
    private String case_name;
    @NonNull
    @ColumnInfo(name = "case_status")
    private String case_status;

    public CaseItem(@NonNull String case_name, @NonNull String case_status) {
        this.case_name = case_name;
```

```
        this.case_status = case_status;
    }

    @NonNull
    public String getCase_name() {
        return case_name;
    }

    public void setCase_name(@NonNull String case_name) {
        this.case_name = case_name;
    }

    @NonNull
    public String getCase_status() {
        return case_status;
    }

    public void setCase_status(@NonNull String case_status) {
        this.case_status = case_status;
    }
}
```

CaseAdapter.java

```
import static android.content.Context.MODE_PRIVATE;

import android.annotation.SuppressLint;
import android.content.Context;
import android.content.SharedPreferences;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.RecyclerView;

import com.danylo.todolister.com.ListFrags.TasksFrag;
import com.danylo.todolister.com.R;

import java.util.List;

public class CaseAdapter extends RecyclerView.Adapter<CaseAdapter.CaseViewHolder> {

    private Context context;
    private List<CaseItem> caseItems;
    public static final String CASE_PREFS ="CASE_PREFS";
    private SharedPreferences prefs;
    private SharedPreferences.Editor editor;

    public CaseAdapter(Context context) {
        this.context = context;
    }

    @NonNull
    @Override
    public CaseViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view=LayoutInflater.from(context).inflate(R.layout.case_item,parent,false);
        return new CaseViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull CaseViewHolder holder, int position) {
        final CaseItem caseItem=caseItems.get(position);
        String caseName=caseItem.getCase_name();
        String caseStatus=caseItem.getCase_status();
        holder.case_name_text.setText(caseName);
        holder.case_status_text.setText(caseStatus);
        holder.itemView.setOnClickListener(view -> {
            AppCompatActivity compatActivity=(AppCompatActivity) view.getContext();
            prefs = compatActivity.getSharedPreferences(CASE_PREFS,MODE_PRIVATE);
            editor = prefs.edit();
            editor.putString("caseName",caseName);
            editor.apply();
            compatActivity.getSupportFragmentManager().beginTransaction()
```

```
                .setCustomAnimations(android.R.animator.fade_in,android.R.animator.fade_out)
                .replace(R.id.list_frames,new TasksFrag()).commit();
        });
    }

    @Override
    public int getItemCount() {
        if (caseItems !=null){
            return caseItems.size();
        }
        else return 0;
    }
    @SuppressLint("NotifyDataSetChanged")
    public void setCaseItems(List<CaseItem> caseItems){
        this.caseItems=caseItems;
        notifyDataSetChanged();
    }

    public static class CaseViewHolder extends RecyclerView.ViewHolder {
        private TextView case_name_text,case_status_text;
        public CaseViewHolder(@NonNull View itemView) {
            super(itemView);
            case_name_text=itemView.findViewById(R.id.case_name_text);
            case_status_text=itemView.findViewById(R.id.case_status_text);
        }
    }
}
```

TaskItem.java

```
import androidx.annotation.NonNull;
import androidx.room.ColumnInfo;
import androidx.room.Entity;
import androidx.room.PrimaryKey;

@Entity(tableName = "Tasks_Table")
public class TaskItem {
    @PrimaryKey
    @NonNull
    @ColumnInfo(name = "task_name")
    private String task_name;
    @NonNull
    @ColumnInfo(name = "task_info")
    private String task_info;
    @NonNull
    @ColumnInfo(name = "task_executor")
    private String task_executor;
    @NonNull
    @ColumnInfo(name = "task_date")
    private String task_date;
    @NonNull
    @ColumnInfo(name = "task_status")
    private String task_status;
    @NonNull
    @ColumnInfo(name = "task_case")
    private String task_case;

    public TaskItem(@NonNull String task_name, @NonNull String task_info, @NonNull String task_executor, @NonNull String task_date,
@NonNull String task_status, @NonNull String task_case) {
        this.task_name = task_name;
        this.task_info = task_info;
        this.task_executor = task_executor;
        this.task_date = task_date;
        this.task_status = task_status;
        this.task_case = task_case;
    }

    @NonNull
    public String getTask_name() {
        return task_name;
    }

    public void setTask_name(@NonNull String task_name) {
        this.task_name = task_name;
    }
```

```
    @NonNull
    public String getTask_info() {
        return task_info;
    }

    public void setTask_info(@NonNull String task_info) {
        this.task_info = task_info;
    }

    @NonNull
    public String getTask_executor() {
        return task_executor;
    }

    public void setTask_executor(@NonNull String task_executor) {
        this.task_executor = task_executor;
    }

    @NonNull
    public String getTask_date() {
        return task_date;
    }

    public void setTask_date(@NonNull String task_date) {
        this.task_date = task_date;
    }

    @NonNull
    public String getTask_status() {
        return task_status;
    }

    public void setTask_status(@NonNull String task_status) {
        this.task_status = task_status;
    }

    @NonNull
    public String getTask_case() {
        return task_case;
    }

    public void setTask_case(@NonNull String task_case) {
        this.task_case = task_case;
    }
}
```

TaskAdapter.java

```
import static android.content.Context.MODE_PRIVATE;

import static com.danylo.todolister.com.ListDatabasePack.CaseAdapter.CASE_PREFS;

import android.annotation.SuppressLint;
import android.content.Context;
import android.content.SharedPreferences;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.RecyclerView;

import com.danylo.todolister.com.ListFrags.TaskDetailFrag;
import com.danylo.todolister.com.ListFrags.TasksFrag;
import com.danylo.todolister.com.R;

import java.util.List;

public class TaskAdapter extends RecyclerView.Adapter<TaskAdapter.TaskViewHolder> {
    private Context context;
    private List<TaskItem> taskItems;
    private SharedPreferences prefs;
    private SharedPreferences.Editor editor;
```

```
  public TaskAdapter(Context context) {
     this.context = context;
  }

  @NonNull
  @Override
  public TaskViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
     View view= LayoutInflater.from(context).inflate(R.layout.task_item,parent,false);
     return new TaskViewHolder(view);
  }

  @Override
  public void onBindViewHolder(@NonNull TaskViewHolder holder, int position) {
     final TaskItem taskItem=taskItems.get(position);
     String task_name=taskItem.getTask_name();
     String task_status=taskItem.getTask_status();
     String task_date=taskItem.getTask_date();
     String task_info=taskItem.getTask_info();
     String task_executor=taskItem.getTask_executor();
     holder.task_name_text.setText(task_name);
     holder.task_status_text.setText(task_status);
     holder.task_end_text.setText(task_date);
     holder.itemView.setOnClickListener(view -> {
        AppCompatActivity compatActivity=(AppCompatActivity) view.getContext();
        prefs = compatActivity.getSharedPreferences(CASE_PREFS,MODE_PRIVATE);
        editor = prefs.edit();
        editor.putString("task_name",task_name);
        editor.putString("task_status",task_status);
        editor.putString("task_date",task_date);
        editor.putString("task_info",task_info);
        editor.putString("task_executor",task_executor);
        editor.apply();
        compatActivity.getSupportFragmentManager().beginTransaction()
              .setCustomAnimations(android.R.animator.fade_in,android.R.animator.fade_out)
              .replace(R.id.list_frames,new TaskDetailFrag()).commit();
     });
  }

  @Override
  public int getItemCount() {
     if (taskItems !=null){
        return taskItems.size();
     }
     else return 0;
  }
  @SuppressLint("NotifyDataSetChanged")
  public void setTaskItems(List<TaskItem> taskItems){
     this.taskItems=taskItems;
     notifyDataSetChanged();
  }

  public static class TaskViewHolder extends RecyclerView.ViewHolder {
     private TextView task_name_text,task_status_text,task_end_text;
     public TaskViewHolder(@NonNull View itemView) {
        super(itemView);
        task_name_text=itemView.findViewById(R.id.task_name_text);
        task_status_text=itemView.findViewById(R.id.task_status_text);
        task_end_text=itemView.findViewById(R.id.task_end_text);
     }
  }
}
```

DatabaseList.java

```
import android.content.Context;

import androidx.room.Database;
import androidx.room.Room;
import androidx.room.RoomDatabase;

@Database(entities = {CaseItem.class,TaskItem.class},version = 1,exportSchema = false)
public abstract class DatabaseList extends RoomDatabase {
  private static DatabaseList databaseList;
  public abstract DAO_List dao_list();
  public static DatabaseList getDatabaseList(final Context context){
```

```
    if (databaseList==null){
        synchronized (DatabaseList.class){
            if (databaseList==null){
                databaseList= Room.databaseBuilder(context.getApplicationContext(),DatabaseList.class,"List_Database").build();
            }
        }
    }
    return databaseList;
  }
}
```

DAO_List.java

```
import androidx.lifecycle.LiveData;
import androidx.room.Dao;
import androidx.room.Delete;
import androidx.room.Insert;
import androidx.room.OnConflictStrategy;
import androidx.room.Query;
import androidx.room.Update;

import java.util.List;

@Dao
public interface DAO_List {
  @Insert(onConflict= OnConflictStrategy.IGNORE)
  void insertCase(CaseItem caseItem);
  @Insert(onConflict= OnConflictStrategy.IGNORE)
  void insertTask(TaskItem taskItem);

  @Query("SELECT * FROM Cases_Table")
  LiveData<List<CaseItem>> getAllCases();
  @Query("SELECT * FROM Tasks_Table WHERE task_case LIKE :task_case")
  LiveData<List<TaskItem>> getAllCaseTasks(String task_case);

  @Update
  void updateTask(TaskItem taskItem);
  @Query("UPDATE tasks_table SET task_name= :task_name,task_info= :task_info,task_executor= :task_executor,task_date=
:task_date,task_status= :task_status WHERE task_name LIKE :task_name")
  void updateSingleTask(String task_name,String task_info,String task_executor,String task_date,String task_status);
}
```