Міністерство освіти і науки України
Державний університет «Одеська політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра системного програмного забезпечення

Афанасьєв Богдан Володимирович
студент групи АС-161

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Мобільний застосунок з системою рекомендацій для формування множини мотиваційних цитат

Спеціальність:
121 – Інженерія програмного забезпечення
Освітня програма:
Інженерія програмного забезпечення

Керівник:
Зіноватна Світлана Леонідівна,
кандидат технічних наук, доцент

Одеса – 2021

ЗМІСТ

Міністерство освіти і науки України
Державний університет «Одеська політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра системного програмного забезпечення

Рівень вищої освіти: другий (магістерський)
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____проф. Любченко В.В.
"_____" _____ 2021 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Афансьєву Богдану Володимировичу, АС-161
(прізвище, ім'я та по батькові)

1. Тема проекту (роботи) Мобільний застосунок з системою рекомендацій для формування множини мотиваційних цитат
Керівник проекту Зіноватна Світлана Леонідівна, канд. техн. наук, доцент
затверджені наказом ректора від "25"жовтня 2021 р.№374-в
2. Строк подання студентом проекту (роботи) 07.12.2021
3. Вихідні дані по проекту (роботі) Технічне завдання
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належіть розробити)
Вступ. Вивчення особливостей предметної області. Вимоги до програми для зберігання множини мотиваційних цитат відповідно особистим перевагам. Архітектура програмного продукту. Структура загальної бази даних. Опис класів. Інтерфейс користувача. Тестування програми. Висновки. Список літератури
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Мета та задачі роботи. Порівняння функціональних характеристик продуктів-аналогів. Сервіс рекомендацій. Діаграма варіантів використання. Струкутра загальної БД. Архітектура програми. Структура локальної БД. Діаграма класів. Приклади екранних форм програми. Тестові випадки. Висновки

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Результат прийняв

7. Дата видачі завдання 28.08.2021

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1.	Вивчення предметної області та аналіз програм-аналогів	10.09.2021	виконано
2.	Формування вимог	21.09.2021	виконано
4.	проектування структури бази даних	05.10.2021	виконано
5.	Розробка класів	22.10.2021	виконано
6.	Реалізація інтерфейсу програми	04.11.2021	виконано
7.	Тестування програми	12.11.2021	виконано
8.	Оформлення документації	15.11.2021	виконано

Здобувач вищої освіти _____ *Б. В. Афанасьєв*

Керівник роботи _____ *С. Л. Зіноватна*

РЕФЕРАТ

Метою роботи є скорочення часу на пошук мотиваційних цитат за рахунок створення власного сховища даних, асоційованого з мобільним застосунком.

У роботі поставлено і виконано завдання розробки мобільного застосунку для показу мотиваційних цитат, з можливостю створення персональних рекомендацій на основі дій користувача.

В роботі проаналізовані особливості предметної області, сформовані функціональні вимоги до програмного забезпечення. Створено структуру загальної та локальної баз даних. Розроблено архітектуру застосунку. Спроектований програмний інтерфейс користувача.

Інтерфейс користувача реалізований засобами мов Kotlin та тексту розмітки XML. Віддалений серер реалізований засобами мови Java. Інформація, необхідна для роботи застосунку зберігається у двох базах даних – SQLite та PostgeSQL.

Ключові слова: мобільний застосунок, локальна база даних, Kotlin, Java, Android, JetPack, Spring, Pyton, матриця кореляцій.

ABSTRACT

The aim of the work is to reduce the time to search for motivational quotes by creating own data warehouse associated with the mobile application.

The task of development of the mobile application for display of motivational quotations, with a possibility of creation of personal recommendations on the basis of user actions is set and executed in work.

The peculiarities of the subject area are analyzed in the work, the functional requirements to the software are formed. The structure of general and local databases is created. Developed application architecture. Designed software user interface.

The user interface is implemented by means of Kotlin languages and XML markup text. The remote server is implemented by means of the Java language. The information required for the application to work is stored in two databases - SQLite and PostgeSQL.

Keywords: mobile application, local database, Kotlin, Java, Android, JetPack, Spring, Pyton, correlation matrix.

ВСТУП

На сьогоднішній день більшість людей у світи використовують мобільні пристрої. В [1] названі такі фактори популярності мобільних пристроїв: функціонал, доступ до всіх засобів зв’язку, невеликий об’єм пристрою. Також визначений основний недолік використання мобільної техніки: «з'являються деякі труднощі, які полягають у відстеженні всіх потоків інформації на всіх пристроях».

В [2] вказано, що «продаж смартфонів росте із блискавичною швидкістю, а компанії-виробники вкладають надзвичайні суми в мобільні технології, їхній розвиток і популяризацію на ринку. Мобільна розробка - дуже стрімко зростаюча область програмування, адже кількість мобільних пристроїв значно перевищує кількість персональних комп'ютерів, і ця тенденція буде тільки зростати».

Також постійно зростає кількість завантажень застосунків для мобільних пристроїв. Наприклад, в 2020 році по усьому світі скачали більше чим 204 млрд. застосунків (без обліку повторних установок і оновлень), що на 45% більше, чим у 2016 році [4].

Застосунки для смартфонів стали невід’ємною частиною життя сучасної людини. Різні типи мобільних застосунків дають можливість отримати допомогу в організації роботи й відпочинку, доступ до останніх новин, спілкування із колегами та друзями. Створення якісних сервісів вигідно всім учасникам ринку мобільного програмного забезпечення. Користувачі одержують інструменти для рішення повсякденних задач. Розроблювачі отримують прибуток за рахунок: продажу рекламних блоків; платних додаткових функцій (у безкоштовній програмі); платного поширення програм; створення контенту на замовлення [5].

У поняття розробки мобільних застосунків для смартфонів, планшетів і інших мобільних пристроїв входить написання програмного коду з метою створення програм, які будуть працювати на певних мобільних платформах

(на сьогоднішній день існує 2 основні платформи мобільних операційних систем - Android і iOS, і менш популярні Windows Phone і Symbian). Ці програми й застосунки можуть попередньо встановлюватися на мобільні телефони, персональні цифрові помічники, корпоративні цифрові асистенти, смартфони та інші мобільні пристрої до того, як пристрою потраплять у руки користувачеві, або завантажуватися користувачами в пристрій безпосередньо в процесі використання [2, 3].

Мобільні застосунки можна розділити на 4 категорії відповідно до призначення: гри; промо-застосунки; контентні сервіси; соціальні мережі. Існують також інші типи програм, менш затребувані, чим перераховані групи. Наприклад, сервіси для ведення бізнесу, створені для керування компаніями [5]. Ігрові застосунки включають мобільні ігри різних жанрів; дитячі, дорослі, сімейні ігри. Промо-програми створюються на замовлення для бізнесу, служать для просування бренда. Головна мета розроблювачів - це максимальне охоплення клієнтів, які можуть користуватися різними моделями пристроїв. Соціальні мережі - сервіси дозволяють спілкуватися в соціальних мережах через мобільний пристрій. Контентні застосунки - це програми, створені для швидкого доступу до певного контенту. Можуть містити, наприклад, інформацію:

- публікації новинних видань;
- мотивуючі цитати;
- актуальні курси валют;
- системи схуднення або тренувань;
- навчальні мовні курси.

Перевага роботи в сфері розробки мобільних додатків - затребуваність контента для мобільних телефонів, користувачам потрібні зручні програми, бізнесу потрібні інструменти для просування. Недоліки: 1) пропозиція перевищує попит, для рішення кожного завдання вже можуть існувати десятки програм, потрібний інноваційний контент або якісно новий рівень реалізації

існуючої ідеї; 2) необхідність постійних оновлень, оскільки випускаються нові моделі пристроїв, нові версії операційних систем; 2) програма повинна підбудовуватися під запити користувачів, необхідно постійно обновляти функції, додавати новий контент, щоб зберігати охоплення аудиторії й лояльність користувачів [5].

Як сказано вище, одним з варіантів змісту контентних систем є мотивуючі цитати.

Мотиваційні цитати працюють, змушуючи людини покинути зону комфорту й почати рухатися в напрямку своєї мрії. Як правило, людина звертає увагу на те, що відповідає його переживанням і думкам у певний момент часу. Це пояснює ефект мотиваційних цитат. Вони демонструють альтернативний погляд на існуючу проблему й допомагають зробити правильний вибір для її рішення [6].

Тобто задача наявності застосунку для зберігання множини мотиваційних цитат відповідно особистим перевагам є актуальною задачею.

Метою роботи є скорочення часу на пошук мотиваційних цитат за рахунок створення власного сховища даних, асоційованого з мобільним застосунком.

Для досягнення мети в роботі потрібно вирішити наступні задачі:

– визначити особливості створення облікових систем у вигляді мобільного застосунку;

– проаналізувати функціональні характеристики програм для зберігання власної інформації на мобільному пристрої у вигляді бази даних;

– розробити структуру загальної та локальної баз даних;

– розробити архітектуру застосунку;

– розробити програмний інтерфейс для можливості наповнення загальної та локальної баз даних та пошуку необхідної інформації.

Об'єктом дослідження є мобільні застосунки з власною базою даних.

Предметом дослідження є засоби збереження даних на мобільному пристрої та взаємодії баз даних мобільного пристрою да хмарного сховища даних.

В першому розділі розглянуто основні характеристики мобільних застосунків, перелічені їх категорії. Визначені основні етапи розробки мобільних застосунків, порівняно перелік етапів за різними джерелами. Описані та порівняні різні види застосунків для перегляду цитат.

У другому розділі визначені вимоги до програми для зберігання множини мотиваційних цитат відповідно особистим перевагам, зокрема, функціональні вимоги у вигляді варіантів використання, а також описані нефункціональні вимоги відповідно зазначеним типам.

У третьому розділі визначено інструменти та засоби розробки програмного забезпечення, проаналізовано різні шаблони архітектури, та вибрано найбільш відповідну для цілей розробки. На основі вибраної архітектури побудовано та описано структури концептуальних класів та структура загальної та локальної баз даних.

В четвертому розділі описано програмні класи, які створені на основі концептуальних класів. Надано детальний опис методів класів, які розподілені на частини відповідно архітектурному шаблону. Також визначений принцип роботи сервісу рекомендацій. Описано графічний інтерфейс застосунку у вигляді інструкції використання.

В пятому розділі описано способ тестування програмнго продукту з використанням тестових випадків, наведені детальні описи тестових випадків для окремих варіантів використання. Описано експеримент, який доказав досягнення мети розробки.

Результати роботи доповідалися й обговорювалися на XIV Міжнародної науково-практичної конференції «Інформаційні технології і автоматизація – 2021», Одеса, 21 - 22 жовтня 2021 р.

1 ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Ринок мобільних застосунків

У сучасному світі використовується все більше мобільних застосунків. З кожним роком збільшується розрив між трафіком з десктопних і мобільних пристроїв. Наприклад, в 2018 році американці проводили в середньому 3.6 годин у мобільному (що вже в 12 разів більше, ніж показник 2008 року) і всього 2 години на день із комп'ютера (цей показник стабільний). За підсумками першого півріччя 2020, в Google Play було доступно до скачування майже 3 млн. застосунків; а в Apple App Store - близько 4,4 млн. позицій. У середньому, щомісяця в Google Play випускається більше 100 000 нових застосунків, а в Apple App Store - більше 30 000 [4].

В [7] визначено, що завдяки пандемії короновірусу 2020-й рік значно прискорив розвиток мобільного ринку. Особливостями минулого року є різке зростання витрат мобільних користувачів по усьому світі. Зокрема, відповідно даним дослідницької компанії SensorTower, споживчі витрати в мобільних застосунках і іграх, а також витрати на їхню покупку, перевищили $100 млрд до початку грудня, що на 30% більше показників 2019 року.

В [8] описано, на які види мобільних застосунків люди витрачають найбільше часу (табл.1.1).

Таблиця 1.1 – Витрати часу за видами мобільних застосунків

Вид застосунку	Опис	Зростання часу, %
Персоналізація	Відхід від шаблонності й прагнення виділитися, завантаження нестандартних іконок для меню, бекграундів, екранів блокування, лаунчерів	Не визначено

Продовження табл. 1.1

Вид застосунку	Опис	Зростання часу, %
Електронні журнали	Читання газет й журналів в іншому середовищі - мобільному застосунку	135
Оптимізація пристрою	Завантаження застосунків, які підвищують продуктивність, прохолоджують смартфони, очищають сміття й бережуть батарею	Не визначено
Корисні поради для життя	Щоденні підказки по шопінгу, стилю життя й лайфхакам	81
Месенджери, сервіси для подорожей і спорту	Забезпечує швидке спілкування незалежно від географічного місця знаходження, спрощує планування подорожі тощо	більше 50

В [9] наведені такі дані: протягом 2020 року найбільший ріст аудиторії користувачів спостерігався в категоріях бізнес-застосунків (збільшення бази встановлених застосунків склало 134%), медичних додатків (66%), застосунків для фітнеса (50%), і застосунків для освіти (41%).

В [9] наведені дані про зростання обсягів ринку мобільних додатків різного типу (рис.1.1) в грошовому вираженні оплати за підписки та у відсотках у порівнянні даних четвертого кварталу 2019 року з даними другого кварталу 2021 року.

В [10] також надані результати аналізу ринку мобільних додатків за 2020 рік у розрізі змін у зв’язку з пандемією:

– доходи мобільних ігор в 2020 склали більше 69% від всієї галузі в Google Play і 71% - в App Store;

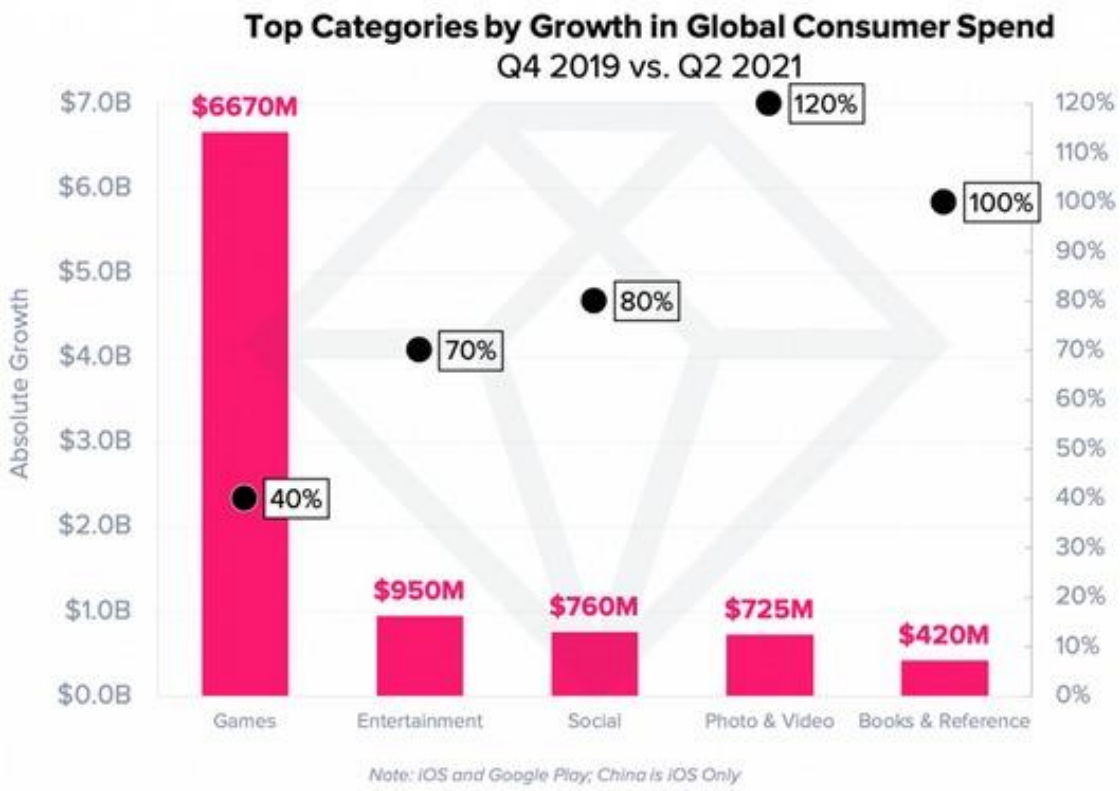

Рисунок 1.1 – Зростання обсягів ринку мобільних додатків [9]

– зростання мобільної комерції, коли під час пандемії офлайн продажі недоступні, користувачі стали використати смартфони як основний спосіб здійснення покупок;

– кількість застосунків для покупок продуктів харчування й замовлень їжі з ресторанів зросло до 200%;

– багато освітніх установ перейшло в онлайн режим, відповідно виросла кількість продажів онлайн курсів, внутрішніх покупок в освітніх застосунках;

– зростання продажів застосунків для видаленої роботи, оскільки офісні працівники перейшли в режим видаленої роботи, при цьому багато компаній зрозуміли, що робота з дому може бути не менш ефективною, відсутність витрат на офіс компенсує незначне зниження ефективності роботи вилучених співробітників.

1.2 Особливості мобільних застосунків та основні етапи їх розробки

В [2] дано неформальне визначення розробника мобільних застосунків: «це ті люди, які вдихають життя в бездушні мобільні коробочки, якісь шамани-заклинателі, які говорять іноземною мовою й можуть практично всі,… це ті люди, які рухають прогрес уперед, створюючи усе більше зроблені й складні програми». Відзначено, що гарний програміст повинен знати мови та технології, мати навики системного мислення, вміти працювати в команді, вміти спілкуватися та розуміти замовника.

Відповідно [2], мобільних розробників можна поділити на дві категорії в залежності від програмного забезпечення, для якого створюється застосунок: iOS-розробники та Android-розробники.

Головною задачею розроблювача є створення зручного мобільного застосунку, який буде безвідмовно працювати, а також буде інтуїтивно зрозумілим, корисним й багатофункціональним для кінцевого користувача, і прибутковим для замовника [2].

Особливостями розробки мобільного застосунку є невеликий розмір екрану, велика розмаїтість пристроїв, під які потрібно писати програму, постійна взаємодія з Інтернетом, відносно невисока обчислювальна потужність. Більша частина обчислювальної логіки, що є в застосунку, є взаємодією з Інтернетом і із хмарними сервісами [2].

Багато джерел описує етапи розробки мобільного застосунку (табл. 1.2).

Таблиця 1.2 – Етапи розробки мобільного застосунку за різними джерелами

Етапи розробки	Джерело
Пошук основної ідеї застосунка. Створення базового функціонала. Визначення базової моделі поширення.	[2]

Продовження табл. 1.2

Етапи розробки	Джерело
UI/UX design. Валідація прототипів. Просування. Повторне тестування функціональності застосунка й пошук помилок. Реліз.	[2]
Аналітика. Технічне завдання. Проектування й дизайн. Розробка. Тестування й стабілізація. Публікація в сторах. Підтримка й розвиток	[11]
Технічне завдання (результат: опис базових функцій мобільного застосунка; вибір платформи: iOS, Android або крос-платформа; вибір методології: Agile або Waterfall). Планування й оцінка (результат: перелік завдань; бюджет проекту). Аналітика (результат: специфікація функціональних вимог; специфікація нефункціональних вимог; основа графічного інтерфейсу; план проекту; детальний бюджет). Дизайн додатка (результат: карта екранів; дизайн застосунка; привабливий UI й зручний UX). Розробка (результат: версія застосунка, готова до тестування; коректування дизайну). Тестування й усунення помилок (результат: помилки зведені до мінімуму; передрелізна версія застосунка).	[12]

Продовження табл. 1.2

Етапи розробки	Джерело
Реліз (результат: застосунок у сторі). Техпідтримка й розвиток (результат: гарантія на усунення помилок; договір супроводу).	[12]

1.3 Використання мотиваційних цитат

Мотиваційні цитати допомагають людині домагатися мети й вірити в себе, змушують рухатися вперед, дають сили не зупинятися на півшляху й не шукати виправдань. Мотивація важлива для самовдосконалення. Саме короткі мотивуючі фрази допоможуть надихнутися й випливати до наміченої мети. Однак без знань, досвіду й працьовитості навіть сама мотивована людина навряд чи зможе домогтися результату [13].

Думки людини мають велике значення й вплив на рівень власного успіху. Усе, що думає людина, має значення. Тому важливо заряджати себе правильними думками й підтримувати позитивний настрій, незважаючи на труднощі. Іноді кожному потрібна додаткова мотивація, тому що справи складаються не так, як хочеться, або шлях успіху виявляється довше, ніж очікувалося. У такому випадку можуть допомогти мотивуючі цитати, які додають енергії й бажання продовжувати рухатися до наміченої мети [14].

Можна розподілити мотиваційні цитати на багато категорій, наприклад, можна побачити такі запити у пошуковій системі Google:

- мотивуючі цитати для роботи;
- короткі мотиваційні цитати;
- цитати про мотивації в житті;
- мотиваційні цитати для жінок;
- мотиваційні цитати про успіх і ціль;
- мотиваційні цитати із книг;
- мотивуючі цитати на щодня.

1.4 Огляд застосунків для мотиваційних цитат

Платформа Motivetica. Застосування замислювалося як набір цитат про дизайн від дизайнерів і архітекторів. Застосунок дає можливість будь-яку цитату оформити в пропонований дизайн і встановити на телефон. Має два режими: 1) переглянути цитати, цитату, яка сподобалася, можна зберегти як wallpaper; 2) можна написати будь-яку фразу й зберегти як wallpaper. Пропонується зберегти в 2-х інверсійних кольорах – чорному та білому [15] (рис. 1.1).

Рисунок 1.1 – Приклад роботи програми motivetica

Програма QUOTATO. Надзвичайно простий мінімалістичний застосунок для збору улюблених цитат [16] (рис.1.2).

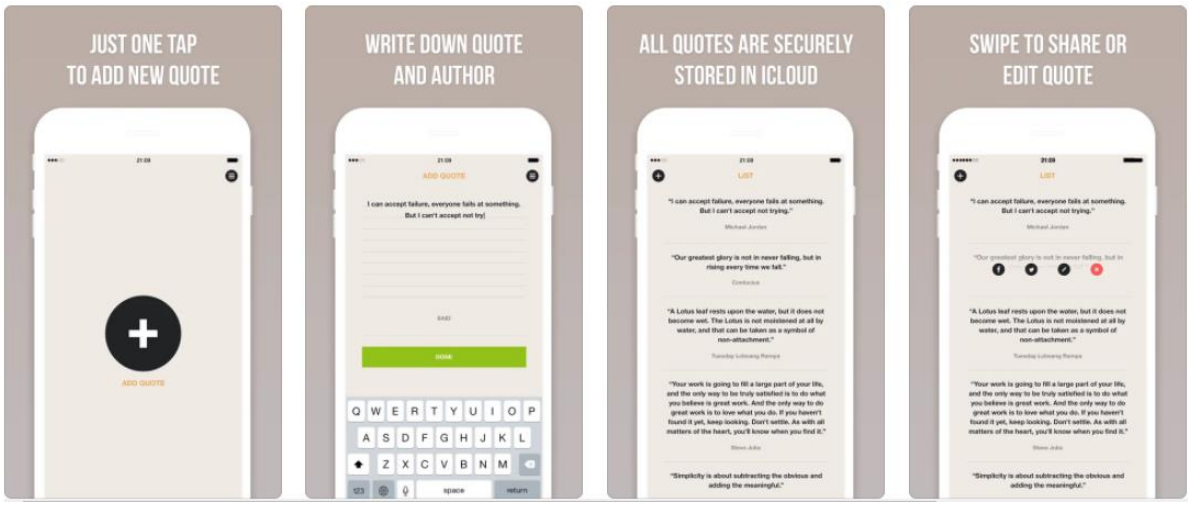

Рисунок 1.2 – Приклад роботи програми QUOTATO

Програма Canva. Програма в першу чергу орієнтована на роботу користувача з різними видами дизайну. Надає можливість застосовувати прийоми оформлення тексту до різних видів публікацій, один з видів публікації – цитати [17], де цитати для мотивації є окремою категорією, яку вже неможливо розбивати на частини (рис. 1.3).

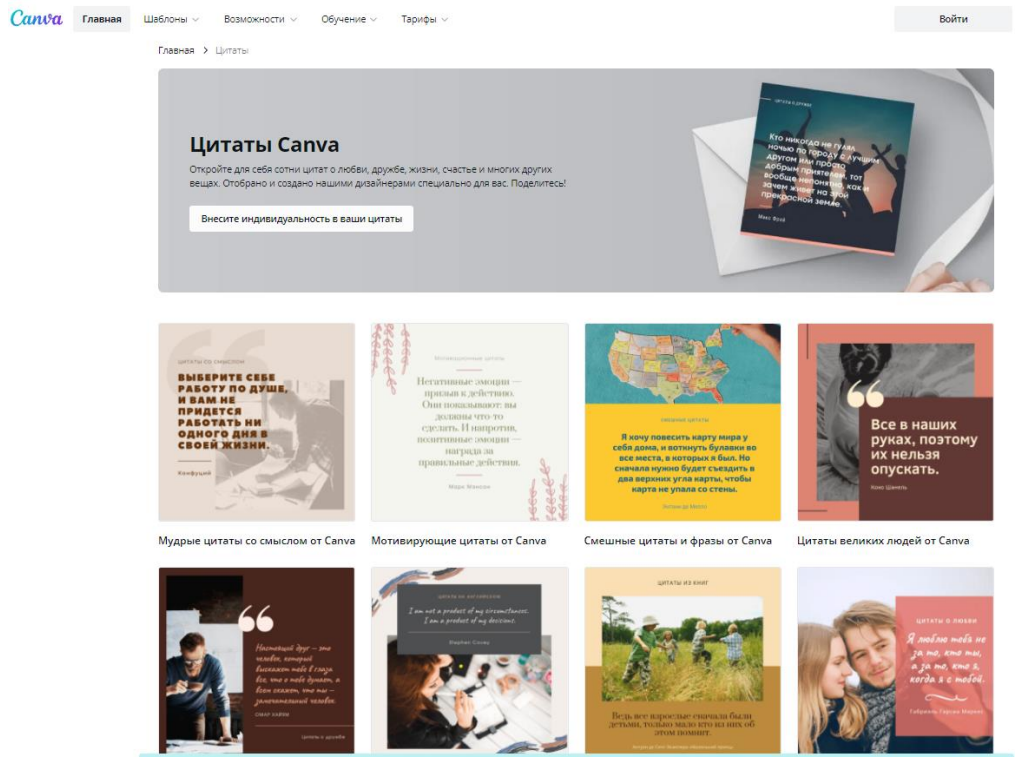

Рисунок 1.3 – Приклад роботи програми Canva

Сайт ДивенСвіт. Містить набор цитат релігійної направленості. Пропонує встановити мобільний застосунок «Цитати Любомира Гузара» [18]. (рис.1.4).

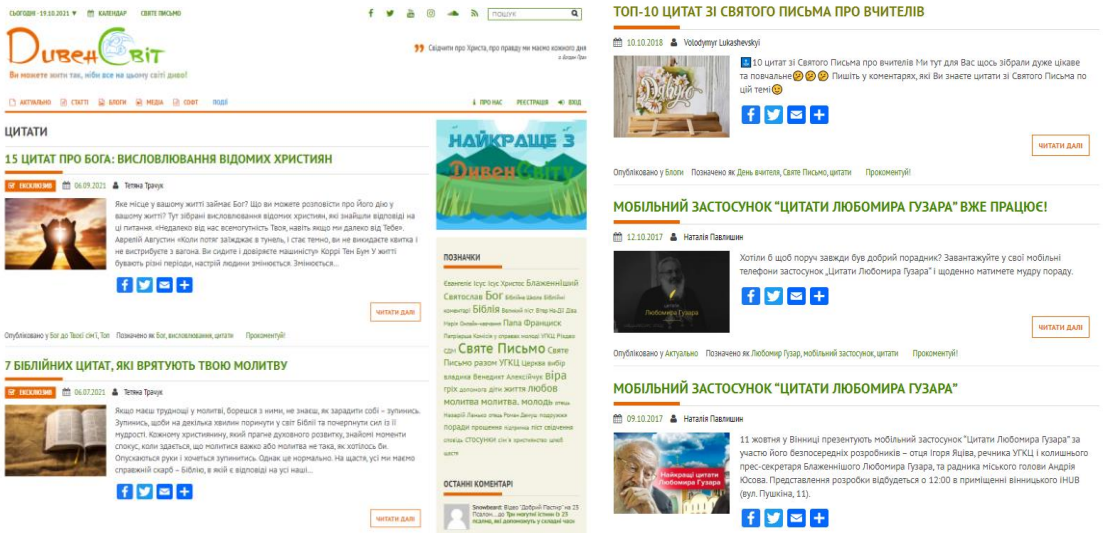

Рисунок 1.4 – Сайт ДивенСвіт

Таблиця 1.3 – Порівняння характеристик застосунків для мотиваційних цитат

Характеристики	Програма				
	Canva	Motivetica	QUOTATO	Дивен Світ	Motivation
Додавання цитати	-	+	+	-	+
Додавання категорії	-	-	-	-	+
Зміна оформлення	+	-	-	-	+
Редагування тексту цитати	+	-	+	-	+

Продовження табл. 1.3

Характеристики	Програма				
	Canva	Motivetica	QUOTATO	Дивен Світ	Motivation
Надлишкові функції	+	-	-	-	-
Мова	Російська	Англ.	Англ.	Укр.	Укр
Операційна система	Web	iOS	iOS	Будь-яка	Android
Персоналізація	+	-	-	-	+
Додавання реакції	-	-	-	-	+
Персональні рекомендації	-	-	-	-	+

Таким чином, можна зробити висновок, що розробка мобільного застосунку для зберігання множини мотиваційних цитат відповідно особистим перевагам є актуальною задачею.

1.5 Висновки до розділу

В розділі розглянуто основні характеристики мобільних застосунків, перелічені їх категорії. Визначені основні етапи розробки мобільних застосунків, порівняно перелік етапів за різними джерелами. Описані та порівняні різні види застосунків для перегляду цитат.

Таким чином, метою роботи є скорочення часу на пошук мотиваційних цитат за рахунок створення власного сховища даних, асоційованого з мобільним застосунком.

Для досягнення мети в роботі потрібно вирішити наступні задачі:

– визначити особливості створення облікових систем у вигляді мобільного застосунку;

– проаналізувати функціональні характеристики програм для зберігання власної інформації на мобільному пристрої у вигляді бази даних;

– розробити структуру загальної та локальної баз даних;

– розробити архітектуру застосунку;

– розробити програмний інтерфейс для можливості наповнення загальної та локальної баз даних та пошуку необхідної інформації.

2 ВИМОГИ ДО ПРОГРАМИ ДЛЯ ЗБЕРІГАННЯ МНОЖИНИ МОТИВАЦІЙНИХ ЦИТАТ ВІДПОВІДНО ОСОБИСТИМ ПЕРЕВАГАМ

2.1 Прецеденти для опису функціональних вимог

Use Case (варіант використання, прецедент, юскейс) - це сценарна техніка опису взаємодії. За допомогою Use Case можна описати користувальницьку вимогу, вимога до взаємодії систем, взаємодію людей і компаній у реальному житті. У загальному випадку, за допомогою Use Case може описуватися взаємодія двох або більшої кількості учасників, які мають конкретну мету [19].

Прецеденти використовується для візуального зображення вимог у моделі системи, яка уточнюється і доповнюється новими сценаріями для одержання остаточних логічної і фізичної моделей системи [20].

Прецедент - опис множини послідовностей виконуваних системою дій (транзакцій) для досягнення певного результату, який має цінність для зазначеного зовнішнього об’єкта (актора). «Одна із послідовностей дій системи спрямована на виконання головної цільової функції даного прецеденту і визначається як основний потік подій (сценарій). Альтернативні потоки описують поведінку системи у виняткових ситуаціях, наприклад при помилках» [21].

Моделювання прецедентів – це форма опису вимог. На етапі аналізу вимог прецедент розглядається як «чорна скринька», розробник повинен вирішити питання. що повинен виконати прецедент, а не як він повинен це зробити [22].

Моделювання прецедентів зазвичай відбувається наступним чином:

- встановлюються межі потенційної системи;
- виявляються актори;
- виявляються прецеденти:
    a) визначається прецедент;
    b) визначаються основний та альтернативні потоки.

До програми для зберігання множини мотиваційних цитат визначені наступні прецеденти (рис.2.1).

Рисунок 2.1 – Діаграма прецедентів

2.2 Сценарії прецедентів

Для прецедентів визначені наступні сценарії.

Прецедент «Отримання цитат з загальної БД»

Актори: Модератор загальної БД, Власник локальної БД

Мета: Переглянути цитату, до якої є зацікавленість

Передумова: Користувач авторизований у застосунку.

Успішний сценарій:

1. Користувач хоче переглянути цитати.

2. Система відкриває вікно зі списком цитат.

3. Користувач обирає потрібну цитату.

4. Система формує вікно з повним текстом цитати.

Результат: Користувач має можливість отримати мотивацію завдяки переглянутій цитаті.

Прецедент «Перевірка цитати в загальній БД»

Актори: Модератор загальної БД

Мета: Отримати коректний текст цитати у загальній БД

Передумова: Користувач авторизований, знаходиться в режимі перегляду цитат.

Успішний сценарій:

1. Користувач хоче перевірити цитати.

2. Система відкриває вікно зі списком неперевірених цитат.

3. Користувач обирає потрібну цитату.

4. Система формує вікно з повним текстом цитати з можливістю редагування.

5. Користувач підтверджує можливість публікації цитати.

6. Система помічає цитату як перевірену та повертається до списку неперевірених цитат.

Альтернативні сценарії

Сценарій 1

1а. Користувач хоче перевірити цитати.

2а. Система перевіряє наявність неперевірених цитат, визначає їх відсутність та виводить повідомлення.

Сценарій 2

5б. Користувач знаходить орфографічні помилки, виправляє їх, хоче зберегти зміни та підтверджує можливість публікації цитати.

6б. Система помічає цитату як перевірену та повертається до списку неперевірених цитат

Сценарій 3

5в. Користувач вважає можливим публікацію цитати.

6в. Система помічає цитату як заборонену до публікації та повертається до списку неперевірених цитат.

Результат: Користувачі застосунку бачать лише коректні цитати у загальній БД.

Прецедент «Додавання цитати до загальної БД»

Актори: Модератор загальної БД

Мета: Отримати нову мотиваційну цитату у загальній БД

Передумова: Користувач авторизований, знаходиться в режимі перегляду цитат.

Успішний сценарій:

1. Користувач хоче додати цитату.

2. Система відкриває вікно для введення тексту цитати, її автора та категорій, до яких потрібно віднести цитату.

3. Користувач вносить дані та хоче зберегти цитату.

4. Система перевіряє заповнення обов’язкових полів та зберігає цитату в загальній БД.

5. Система закриває вікно для введення даних та повертається до перегляду списку цитат.

Альтернативні сценарії

Сценарій 1

4а. Користувач не заповнив обов’язкові поля.

5а. Система повертається до перегляду списку цитат.

Сценарій 2

4б. Користувач не хоче додавати цитату та скасовує дію.

5б. Система закриває вікно для введення даних та повертається до перегляду списку цитат

Результат: Користувачі застосунку мають доступ до оновленого списку цитат у загальній БД.

Прецедент «Додавання категорії»

Актори: Модератор загальної БД, Власник БД

Мета: Створити нову категорію.

Передумова: Користувач авторизований, знаходиться в режимі перегляду категорій.

Успішний сценарій:

1. Користувач хоче додати категорію.

2. Система відкриває вікно для введення назви категорії.

3. Користувач вносить назву категорії та хоче зберегти її.

4. Система перевіряє наявність внесеної назви та зберігає категорію у БД, яка відповідає поточному користувачу.

5. Система закриває вікно для введення даних та повертається до перегляду списку категорій.

Альтернативні сценарії

Сценарій 1

4а. Користувач не вказав назву категорії.

5а. Система виводить повідомлення про необхідність введення назви.

Сценарій 2

3б. Користувач не хоче додавати категорію та скасовує дію.

4б. Система закриває вікно для введення даних та повертається до перегляду списку категорій.

Сценарій 3

4б. Система знаходить категорію з такою назвою серед вже існуючих категорій.

5б. Система виводить повідомлення про необхідність змінити назву.

Результат: Користувачі застосунку мають список категорій, що задовольняє їх потребам.

Прецедент «Зміна стилю застосунку»

Актори: Власник БД

Мета: Мати зручний інтерфейс.

Передумова: Користувач авторизований.

Успішний сценарій:

1. Користувач хоче змінити стиль інтерфейсу.

2. Система виводить список можливих стилів.

3. Користувач обирає стиль.

4. Система запам’ятовує обраний стиль та змінює вигляд інтерфейсу користувача.

Результат: Користувач отримує інтерфейс, який відповідає обраному стилю.

Прецедент «Авторизація для загальної БД»

Актори: Модератор загальної БД, Власник БД

Мета: Мати доступ до функцій загальної частини застосунку.

Передумова: Застосунок відкритий.

Успішний сценарій:

1. Користувач хоче почати працювати з функціями програми.

2. Система відкриває вікно для введення паролю.

3. Користувач вносить пароль та хоче війти в програму.

4. Система перевіряє правильність паролю та відкриває вікно з доступними функціями відповідно статусу користувача.

Альтернативні сценарії

Сценарій 1

4а. Користувача із зазначеним паролем не існує.

5а. Система пропонує зареєструватися.

Результат: Користувачі може виконувати бажані дії.

Прецедент «Додавання цитати до власної БД»

Актори: Власник БД

Мета: Мати в списку цитат всі мотивуючі себе цитати.

Передумова: Користувач авторизований, знаходиться в режимі перегляду цитат.

Успішний сценарій:

1. Користувач хоче додати цитату.

2. Система відкриває вікно для введення тексту цитати, її автора та категорій, до яких потрібно віднести цитату.

3. Користувач вносить дані та хоче зберегти цитату.

4. Система перевіряє заповнення обов’язкових полів та зберігає цитату в загальній БД.

5. Система закриває вікно для введення даних та повертається до перегляду списку цитат.

Альтернативні сценарії

Сценарій 1

4а. Користувач не заповнив обов’язкові поля.

5а. Система виводить повідомлення та повертається до вікна введення даних.

Сценарій 2

4б. Користувач не хоче додавати цитату та скасовує дію.

5б. Система закриває вікно для введення даних та повертається до перегляду списку цитат

Результат: Користувач застосунку має актуальний список цитат для власного мотивування.

Прецедент «Фільтр цитат з загальної БД за категорією»

Актори: Модератор загальної БД, Власник БД

Мета: Скоротити список цитат для перегляду.

Передумова: Користувач авторизований, знаходиться в режимі перегляду цитат.

Успішний сценарій:

1. Користувач хоче переглянути цитати лише окремої категорії.

2. Система відкриває вікно зі списком категорій.

3. Користувач обирає категорію.

4. Система виводить список цитат, які відносяться до обраної категорії.

Результат: Користувач застосунку переглядає лише ті цитати, які цікавлять його на даний момент часу.

Прецедент «Налаштування застосунку»

Актори: Власник БД

Мета: Отримувати цитати, відповідні персональним характеристикам.

Передумова: Користувач авторизований.

Успішний сценарій:

1. Користувач хоче налаштувати застосунок.

2. Система відкриває вікно з полями для введення основних персональних характеристик.

3. Користувач вносить дані.

4. Система зберігає внесені дані та виводить вікно для встановлення часових характеристик сповіщень.

5. Користувач вносить дані.

6. Система зберігає дані

Результат: Користувач застосунку отримує сповіщення відповідно налаштуванням.

Прецедент «Додавання коментаря»

Актори: Власник БД

Мета: Мати можливість фіксувати думки з приводу тексту цитати.

Передумова: Користувач авторизований, знаходиться в режимі перегляду цитат.

Успішний сценарій:

1. Користувач хоче додати коментар.

2. Система відкриває вікно для введення тексту коментаря.

3. Користувач вносить текст та хоче зберегти коментар.

4. Система перевіряє наявність тексту та зберігає коментар в БД.

5. Система закриває вікно для введення даних та повертається до перегляду списку цитат.

Результат: Користувач застосунку зафіксував власну думку відносно обраної цитати.

Прецедент «Редагування цитати»

Актори: Модератор загальної БД, Власник БД

Мета: Містити у БД цитати без помилок.

Передумова: Користувач авторизований, знаходиться в режимі перегляду цитат.

Успішний сценарій:

1. Користувач хоче змінити цитату.

2. Система відкриває вікно з полям для редагування, які містять текст обраної цитати, її автора та категорії, до яких цитата відноситься.

3. Користувач вносить дані та хоче зберегти відкоректовану цитату.

4. Система перевіряє заповнення обов’язкових полів та зберігає цитату в БД.

5. Система закриває вікно для редагування даних та повертається до перегляду списку цитат.

Альтернативні сценарії

Сценарій 1

4а. Користувач не заповнив обов’язкові поля.

5а. Система виводить повідомлення та повертається до вікна редагування.

Сценарій 2

4б. Користувач не хоче редагувати цитату та скасовує дію.

5б. Система закриває вікно для редагування даних та повертається до перегляду списку цитат

Результат: Користувач застосунку має актуальний список цитат для власного мотивування.

Прецедент «Реєстрація для загальної частини»

Актори: Власник БД

Мета: Мати можливість додавати дані до загальної БД.

Передумова: Користувач відкрив застосунок.

Успішний сценарій:

1. Користувач хоче зареєструватися в програмі.

2. Система відкриває вікно з полям для введення логіну та паролю.

3. Користувач вносить дані та хоче виконати реєстрацію.

4. Система перевіряє заповнення обов’язкових полів та зберігає дані про нового користувача.

5. Система закриває вікно для доступу до функцій зареєстрованого користувача.

Альтернативні сценарії

Сценарій 1

4а. Користувач не заповнив обов’язкові поля.

5а. Система виводить повідомлення та повертається до вікна введення даних.

Сценарій 2

4б. Користувач не хоче редагувати цитату та скасовує дію.

5б. Система закриває вікно для введення даних та повертається до стартового вікна застосунку

Сценарій 3

4в. Користувач ввів логін, який вже існує у БД.

5в. Система виводить повідомлення та повертається до вікна введення даних.

Результат: Користувач зареєстрований та отримує можливість користатися функціоналом загальної частини програми.

Прецедент «Додавання категорії до власної БД»

Актори: Власник БД

Мета: Мати зручний персональний набор категорій.

Передумова: Користувач авторизований, знаходиться в режимі перегляду категорій.

Успішний сценарій:

1. Користувач хоче додати категорію.

2. Система відкриває вікно для введення назви категорії.

3. Користувач вносить назву та хоче зберегти категорію.

4. Система перевіряє наявність тексту та зберігає категорію в БД.

5. Система закриває вікно для введення даних та повертається до перегляду списку категорій. Відображає нову категорію у списку

Результат: Користувач отримав оновлений список категорій.

2.3 Нефункціональні вимоги

Нефункціональна вимога, на відмінність від функціональної вимоги, відповідає на запитання, якою повинна бути система, тоді як функціональна вимога відповідає на питання, що повинна робити система.

У [23] перелічені таки типи нефункціональних вимог:

– продуктивність - перевіряється пропускна здатність системи в нормальних умовах навантаження;

– юзабіліті - наприклад, якщо користувачеві для введення даних доводиться переглядати кілька екранів, та опція введення даних відображається в невеликих текстових полях, які користувач не легко бачить, зручність використання програми буде дуже низькою;

– технічне обслуговування – позначає ремонтопридатність, яка може вимірюватися на рівні коду, з використанням цикломатичної складності; цикломатична складність означає, що чим менш складний код, тим легше підтримувати програмне забезпечення;

– надійність - доступність системи за певних умов;

– переносимість - здатність програми працювати в інших середовищах, при незмінності основного фреймворку;

– підтримка - здатність експерта встановлювати програмну систему в реальному часі, контролювати систему під час її роботи, виявляти будь-які технічні проблеми в системі та надавати рішення для вирішення проблеми;

– адаптованість - здатність програмної системи пристосовуватися до змін у середовищі без будь-яких змін у її поведінці.

На додаток до перелічених вище нефункціональних вимог, є також багато додаткових вимог: доступність, резервне копіювання, ємність, відповідність, цілісність даних, збереження даних, залежність, розгортання, документація, довговічність, ефективність, експлуатаційність, розширюваність, управління відмовами, відмовостійкість, сумісність, модифікація, працездатність, конфіденційність, читабельність, звітування, стійкість, повторне використання, надійність, масштабованість, стабільність, перевіряність, пропускна здатність, прозорість, інтегрованість.

До програми Motivation пропонуються наступні нефункціональні вимоги.

Продуктивність – мобільний застосунок працює без помітного зниження швидкості реакції при одночасному підключенню до загальної частини 5 тис. користувачів.

Юзабіліті – програма має інтерфейс з мінімальними переходами між екранними формами, на формах містяться підказки та назви, які повідомляють користувачу, які дії він повинен виконати, або виконує в поточний момент часу.

Технічне обслуговування – програма не має містити зовнішніх посилань, при програмуванні використовуються принципи об’єктно-орієнтованого проектування, що дозволяє виконувати зміни в коді на рівні методів класів.

Надійність – система є безперервно доступною.

Переносимість – програма може працювати на різних версіях Android, на різних типах мобільних пристроїв.

Підтримка – програма є мобільним застосунком, може бути встановлена у будь-який час на мобільний пристрій.

Резервне копіювання – для загальної БД створюється архівна копія один раз на добу.

Розширюваність – функціонал програми може бути розширений та наданий користувачеві у вигляді оновлення версії програми.

Конфіденційність – забезпечується за допомогою шифрування паролів.

2.4 Висновки до розділу

Визначені вимоги до програми для зберігання множини мотиваційних цитат відповідно особистим перевагам, зокрема, функціональні вимоги у вигляді варіантів використання, а також описані нефункціональні вимоги відповідно зазначеним типам.

3 ПРОЕКТУВАННЯ ПРОГРАМИ ДЛЯ ЗБЕРІГАННЯ МНОЖИНИ МОТИВАЦІЙНИХ ЦИТАТ ВІДПОВІДНО ОСОБИСТИМ ПЕРЕВАГАМ

3.1 Інструментальні засоби для розробки програмної системи

Для розробки застосунку використовувано Android Studio - інтегроване середовище розробки (IDE), створене на базі IntelliJ IDEA для платформи Android. Середовище розробки адаптоване для виконання типових завдань, що вирішуються в процесі розробки застосунків для платформи Android. У тому числі у середовище включені інструменти для проєктування застосунків, що працюють на пристроях з екранами різної роздільності. Крім можливостей, присутніх в IntelliJ IDEA, в Android Studio реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування і розгортання застосунків, заснована на складальному інструментарії Gradle, що підтримує використання засобів безперервної інтеграції [25].

Компанія Google, яка розповсюджує і підтримує Android, офіційно заявила, що найкраща мова для написання програм на базі андроїд - це Kotlin [26]. Він отримує найшвидшу і кращу підтримку в інтеграції з Android і Jetpack, тому код програми був написаний саме цією мовою. Kotlin (Котлін) — це статично типізована мова програмування, що працює поверх JVM і також компілюється в JavaScript.

Також Android Studio підтримує фреймворк Android Jetpack. Jetpack - це набір бібліотек, які допомагають розробникам слідувати передовим методам, скорочувати шаблонний код і писати код, який працює злагоджено на всіх версіях Android і на всіх пристроях. Наприклад, бібліотеки Android ViewModel і LiveData допомагають розробнику реалізувати архітектурний патерн MVVM, використовуючи патерн «Спостерігач».

При проектуванні також використана бібліотека Room – це високорівневий інтерфейс для низькорівневих прив'язок SQLite, вбудованих в Android. Тому база даних застосунку була написана з використанням СКБД SQLite.

Для розробки віддаленого REST-сервісу використовувалась IntelliJ IDEA - інтегроване середовище розробки для різних мов програмування (Java, Python, Scala та ін.). IntelliJ IDEA підтримує інструменти для проведення тестування, системи контролю версій, засоби складання, з підтримкою технологій Java EE, UML-діаграм, з підрахунком покриття коду та можливістю роботи з фреймворками (Spring Framework і Hibernate).

Для реалізації REST-сервісу використана мова програмування Java, фреймворк Spring Framework та бібліотека Hibernate. Java - об'єктно-орієнтована мова програмування. В офіційній реалізації Java-програми компілюються в байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи. Spring Framework – це розширення для створення веб застосунків на платформі Java EE. Spring Framework також має в собі реалізацію серверу Tomcat для розгортання програм у веб, а також бібліотеки для реалізації інверсії управління і програмний каркас MVC на основі HTTP сервлета що спрощує створення веб застосунків і веб служб з підходом RESTful. Hibernate також є частиною Spring Framework. Hibernate - це реалізація інтерфейсу Jpa для підтримки ORM підходу.

Для розробки віддаленої бази даних використана СКБД PostgreSQL, яка має безплатну ліцензію та підтримку в безплатній версії хмарної PaaS-платформи Heroku.

Для розробки віддаленого сервісу рекомендацій використовувалась PyCharm — інтегроване середовище розробки для мови програмування Python. PyCharm надає засоби для аналізу коду, інструмент для запуску модульних тестів і підтримує веброзробку на Django.

Python - інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Python підтримує велику кількість бібліотек для роботи з маніпулюванням даними та їхнього аналізу. Серед них була вибрана бібліотека pandas, завдяки якій можна побудувати матриці кореляцій для персональних рекомендацій користувачам.

3.2 Архітектура програмного продукту

В даній кваліфікаційній роботі розроблено мобільне застосунок для операційної системи Android. Щоб реалізувати такий застосунок, з огляду на специфіку роботи системи Android, а саме те, що програми працюють в декількох потоках, головний з яких відводиться зображенню інтерфейсу користувача, використовувати архітектурний підхід MVC є неможливим, тому, що вся логіка буде викликатися в класах, що відповідають за інтерфейс користувача. Розв'язувати цю проблему можна за допомогою двох інших архітектурних підходів - MVP і MVVM, їх загальна риса в тому, що великовагові функції не викликаються безпосередньо з класів інтерфейсу. Замість цього вони лише підписуються на зміни в інших класах, у класах типу «представник» в разі MVP підходу, і класах типу «модель представлення» у випадку MVVM. Між двох цих підходів був обраний MVVM, тому, що його використання рекомендує компанія Google, виробник і розповсюджувач платформи Android, а також надає спрощену реалізацію засобами бібліотеки JetPack ViewModel. На рисунку 3.1 представлено загальну схему реалізації MVVM у системі Android [27].

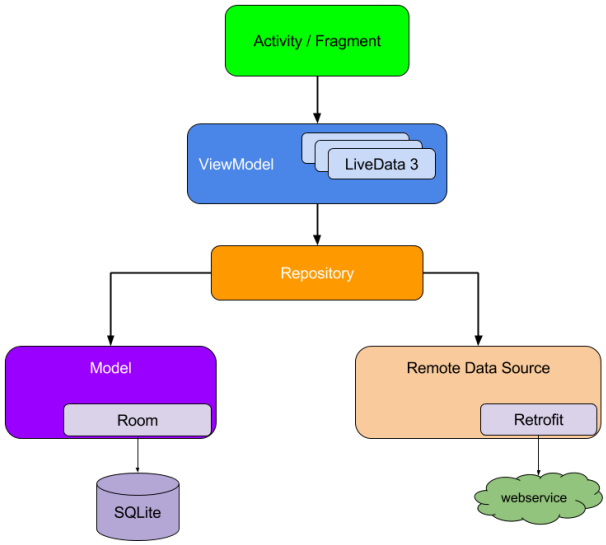

Рисунок 3.1 – Загальна реалізації MVVM у системі Android

MVVM (Model-View-ViewModel) - шаблон проектування архітектури застосувань, що використовується для поділу моделі і її представлення, що необхідно для їх зміни окремо один від одного. У таких платформах, як Android, є концепція «зв'язування даних», що дозволяє зв'язувати дані з візуальними елементами в обидві сторони. Отже, при реалізації цієї концепції використання архітектурного підходу MVC стає неможливим через те, що прив'язка даних до представлення безпосередньо, не вкладається в концепцію MVC. Шаблон MVVM ділиться на три частини. Модель - являє собою логіку роботи з даними та опис фундаментальних даних, необхідних для роботи програми. Представлення - графічний інтерфейс. Виступає підписником на події зміни значень параметрів, що надаються моделлю представлення. У разі, якщо в моделі представлення змінилася якась властивість, то вона сповіщає всіх підписників про це, і представлення, своєю чергою, запитує оновлене значення властивості з моделі представлення. Модель представлення - з одного боку, абстракція представлення, а з іншого - обгортка даних з моделі, що підлягають зв'язуванню [24].

На рисунку 3.2 представлена загальна схема архітектури. Розглянемо кожен блок окремо.

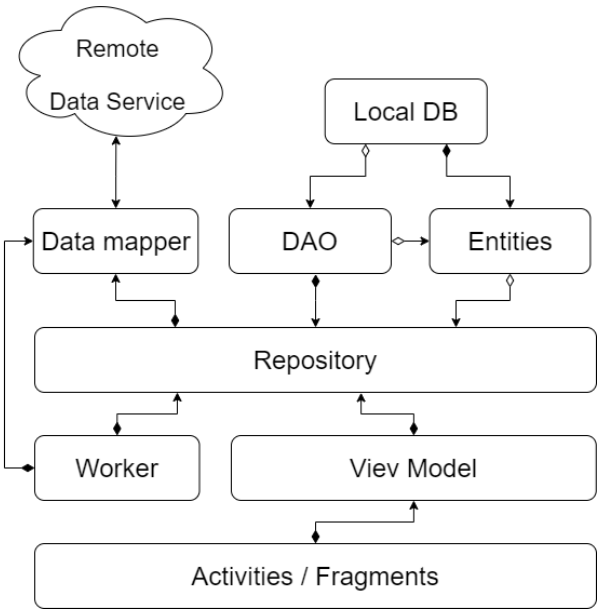

Рисунок 3.2 – Схема архітектури застосунку

Блоки data mapper, local DB, DAO, entities, repository - становлять собою частину "модель". Data mapper - це набір класів, відповідальних за отримання та посилання даних на віддалений сервер. Для взаємодії з сервером був обраний RESTful підхід, завдяки чому дані передаються в форматі json. Тому data mapper при отриманні даних спочатку перетворює їх з json в сутності програми (entities) або, навпаки, при посиланні перетворює сутності в json, а потім за допомогою функцій репозиторію (repository) зберігає їх у локальній базі даних (local DB). Local DB - це класи-службовці для зв'язування програми з локальною базою даних і відповідають за перетворення сутностей програми в SQL синтаксис і назад. DAO - data access object - об'єкти доступу - цей набір класів відповідає за запити в базу даних, вони виступають як функції бази даних. Entities - це сутності програми, вони використовуються як логічні контейнери для даних. Вони відповідають таблицям бази даних і json схемам. Repository - набір класів для взаємозв'язку частини "модель" з іншою програмою, ці класи отримують дані у вигляді сутностей програми з бази даних і віддаленого сервера, використовуючи для цього методи DAO. Також репозиторій відповідає за кінцеве перетворення даних, наприклад, сортування і фільтрацію даних.

Блок viewmodel - це набір класів, відповідальних за частину "модель представлення". Ці класи викликають методи репозиторію та через них отримують дані. Так само viewmodel являє собою логічну модель частини "представлення" - activities / fragments.

Блок "activities / fragments" - це набір класів, відповідальних за частину "представлення". Це класи візуального представлення. Кожен клас відповідає за один екран або контейнер інтерфейсу. Кожен клас пов'язаний з XML розміткою, яка є верстkою та графічним зображенням даних. Кожен клас блоку представлення підписаний на зміни в моделях представлення і в залежності від їх конфігурації оновлює візуальну складову.

Блок worker - це реалізація бібліотеки worker фреймворку android framework. Цей блок здійснює свою роботу із заданою періодичністю, навіть коли програма згорнута або закрита.

3.4 Опис концептуальних класів

Тепер, коли обраний архітектурний підхід, і визначені основні блоки програмного забезпечення, розширимо архітектуру до діаграми концептуальних класів, зображену на рис. 3.3 та 3.4.

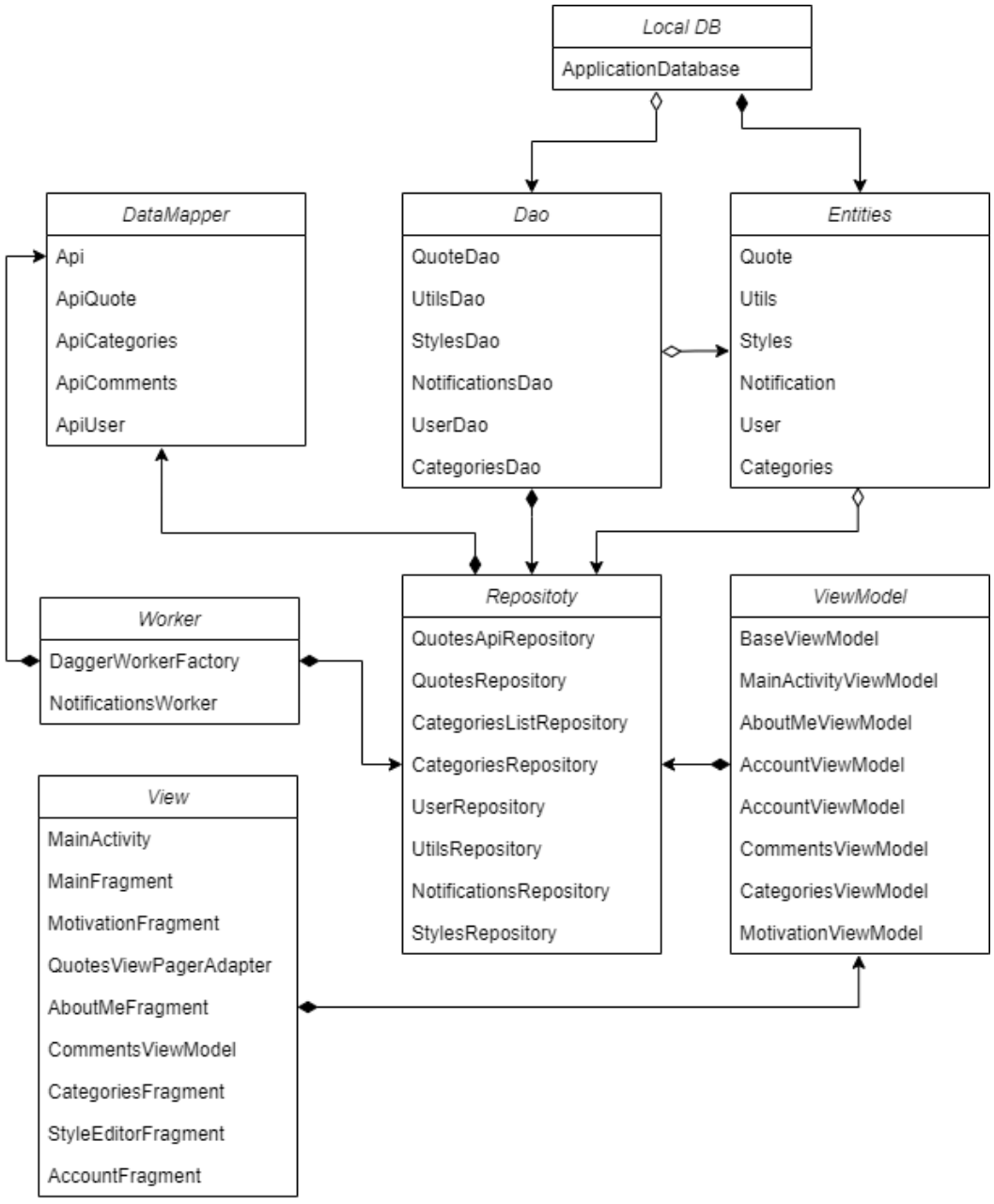

Рисунок 3.3 – Діаграма концептуальних класів Android застосунку

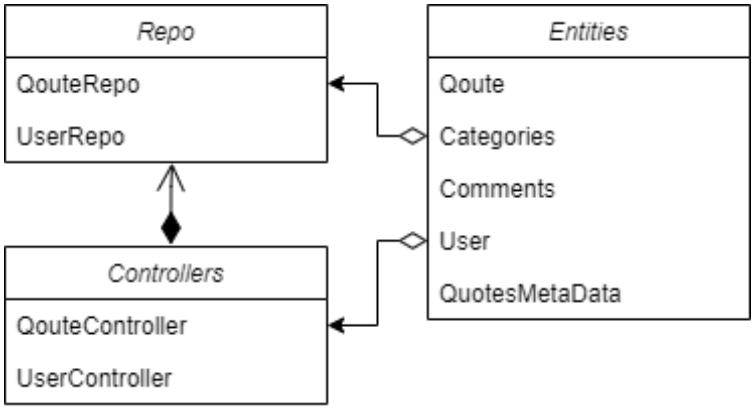

Рисунок 3.4 – Діаграми концептуальних класів серверу

Розглянемо кожен блок окремо.

Концептуальний клас сутностей (entities) включає в себе наступні класи - Quote - клас цитати, основного елемента програми, який демонструється користувачу. Categories - зберігає в собі список можливих категорій цитат. Notification - в цій сутності зберігаються дані про параметри повідомлень, які обрав користувач. Utils - в цій сутності зберігаються прапори, відповідні етапу виконання програми, наприклад, екран вітання показується тільки при першому запуску, для цього служить окремий прапор. Styles - в цьому класі зберігається поточний тип обраної теми програми. User - клас користувача, в ньому зберігається вся інформація про користувача: його персональні дані, посилання на його коментарі та улюблені цитати.

Концептуальний клас DAO містить наступні класи QuoteDao, UtilsDao, StylesDao, NotificationsDao, UserDao, CategoriesDao. Кожен з них реалізує абстракцію звернення до бази даних. Кожен клас відповідно співвідноситися з класами сутностей у вигляді взаємозв'язку агрегації.

Концептуальний клас локальної бази даних (local DB) містить клас ApplicationDatabase, який містить сутності як поля, тобто у вигляді взаємозв'язку композиції. І об'єкти DAO в якості типу повернення абстрактних функцій, тобто у вигляді взаємозв'язку агрегації.

Концептуальний клас data mapper містить наступні класи - Api, ApiQuote, ApiCategories, ApiComments, ApiUser. Клас Api реалізує отримання і посилання запитів на віддалений сервер у вигляді json формату і використовує інші класи для перетворення його по суті програми типу Api, а потім по суті використовуються в іншій частині програми.

Концептуальний клас репозиторій (repository) містить класи QuotesApiRepository, QuotesRepository, CategoriesListRepository, CategoriesRepository, UserRepository, UtilsRepository, NotificationsRepository, StylesRepository. Кожен з цих класів містить відповідні за назвою об'єкти DAO в якості аргументів, тобто вигляді взаємозв'язку композиції та використовують їх методи для отримання даних у вигляді сутностей.

Концептуальний клас viewmodel містить клас BaseViewModel, який є батьківським для інших класів і містить базову реалізацію, а також класи-спадкоємці MainActivityViewModel, AboutMeViewModel, AccountViewModel, CategoriesViewModel, CommentsViewModel, CategoriesViewModel, MotivationViewModel, CategoriesPickerViewModel, StyleEditorViewModel, YouViewModel, NotificationSettingsViewModel, кожен з яких є моделлю відповідних класів уявлення, а також містить відповідні за назвою об'єкти сховища в якості аргументів, тобто композиційним типом взаємозв'язку та використовують їх методи для отримання кінцевих даних у вигляді сутностей.

Концептуальний клас представлення (view) містить класи MainActivity, MotivationFragment, QuotesViewPagerAdapter, AboutMeFragment, AboutMeRecyclerViewAdapter, CategoriesFragment, MainFragment, StyleEditorFragment, NotificationSettingsFragment, AccountFragment кожен з яких відповідає за один екран або контейнер інтерфейсу, а так само зберігає в собі поле відповідного за назвою viewmodel для отримання даних через їх методи.

Концептуальний клас worker містить класи DaggerWorkerFactory і NotificationsWorker. Вони служать для посилання повідомлень, роботи віджету, а також актуалізації даних.

Тепер розглянемо діаграму концептуальних класів віддаленого сервера.

Програма має отримувати та відправляти дані користувачів і цитат. Бек енд написаний на мові Java, використовуючи фреймворк Spring Framework, зокрема Spring MVC. З огляду на сформульовані вимоги спроектовано діаграму концептуальних класів віддаленого сервера. Розглянемо кожен блок окремо.

Концептуальний клас сутностей (entities) включається в себе наступні класи - Quote - клас цитати, основного елемента програми, який демонструється користувачу. Categories - зберігає в собі список можливих категорій цитат. Comments - зберігає в собі список коментарів до цитат. QuotesMetaData - зберігає в собі пару ключ - значення користувача та його ставлення до цитати в числовому вигляді. User - клас користувача, в ньому зберігається вся інформація про користувача: його персональні дані, посилання на його коментарі та улюблені цитати.

Концептуальний клас репозиторію (repo) включає в себе наступні класи - QouteRepo, UserRepo - кожен з них реалізує абстракцію звернення до бази даних. Кожен клас відповідно співвідноситися з класами сутностей у вигляді взаємозв'язку агрегації.

Концептуальний клас контролера (controllers) включає в себе наступні класи - QouteController, UserController кожен з них пов'язує відповідні сутності та абстракції звернення до бази даних. Реалізує функції зворотного виклику, які реагують на http запит, що приходить на сервер.

3.5 Структура загальної бази даних

Загальна база даних розташована на сервері і зберігає в собі інформацію, необхідну всім користувачам системи – всі можливі цитати, а також інформацію про кожного користувача, для відновлення їх даних. Доступ до даних відбувається через http запити, які в свою чергу перетворюються на sql

запити. Загальна БД зображена на рисунку 3.5 і містить дані в вигляді наступних таблиць.

Таблиця Users містить список зареєстрованих користувачів. Структура таблиці наведена в табл. 3.1.

Таблиця 3.1 – Структура таблиці Users

Назва	Семантика	Тип даних	Обмеження
id	Ідентифікатор користувача	int	Первинний ключ
name	Ім’я користувача	text (20)	
age	Вік користувача	int	
sex	Стать користувача	text (10)	
login	Логін користувача	text (20)	
password	Пароль користувача	text (20)	
dateReg	Дата реєстрації	date	

Таблиця Quotes містить список всіх цитат. Структура таблиці наведена в табл. 3.2.

Таблиця 3.2 – Структура таблиці Quotes

Назва	Семантика	Тип даних	Обмеження
id	Ідентифікатор цитати	int	Первинний ключ
content	Текст цитати	text (1000)	
author	Автор цитати	text (50)	
user	Користувача, який вніс цитату	int	Зовнішній ключ
isControl	Чи перевірена цитата	bool	
createdAt	Час, коли була створенна цитата	date	
updatedAt	Час обновлення цитати	date	

Таблиця Category містить список категорій, до яких можуть бути віднесені цитати. Структура таблиці наведена в табл. 3.3.

Таблиця 3.3 – Структура таблиці Category

Назва	Семантика	Тип даних	Обмеження
id	Ідентифікатор категорії	int	Первинний ключ
category	Назва категорії	text (50)	

Таблиця QuoteCategory містить співвідношення категорій та цитат. Цю структуру отримує користуач застосунку, коли вибирає, які цитати хоче переглядати. Структура таблиці наведена в табл. 3.4.

Таблиця 3.4 – Структура таблиці QuoteCategory

Назва	Семантика	Тип даних	Обмеження
id	Ідентифікатор рядка	int	Первинний ключ
quote	Ідентифікатор цитати	int	Зовнішній ключ
category	Ідентифікатор категорії	int	Зовнішній ключ

Таблиця Comments містить список всіх коментарів, які всі користувачі додали до цитат. Структура таблиці наведена в табл. 3.5.

Таблиця 3.5 – Структура таблиці Comments

Назва	Семантика	Тип даних	Обмеження
id	Ідентифікатор коментаря	int	Первинний ключ
comment	Зміст коментаря	text (400)	
quote	Цитата, до якої був залишенний коментар	int	Зовнішній ключ
user	Користувач, який додав коментар	int	Зовнішній ключ
isControl	Чи перевірений коментар	bool	

Таблиця QouteMetaData містить інформацію про відношення всіх користувачів до цитат, що формуються с коментарів, додавання до улюбленних, часом переглядання та використуається для формування матриць кореляцій для системи персональних рекомендацій. Структура таблиці наведена в табл. 3.6.

Таблиця 3.6 – Структура таблиці QuoteMetaData

Назва	Семантика	Тип даних	Обмеження
id	Ідентифікатор рядка	int	Первинний ключ
user	Ідентифікатор користувача	int	Зовнішній ключ
quote	Ідентифікатор категорії	int	Зовнішній ключ
metaData	Числове значення відношення користувача до цитати	int	

Схема всіх таблиць загальної БД та зв'язків між ними зображена на рис. 3.5.

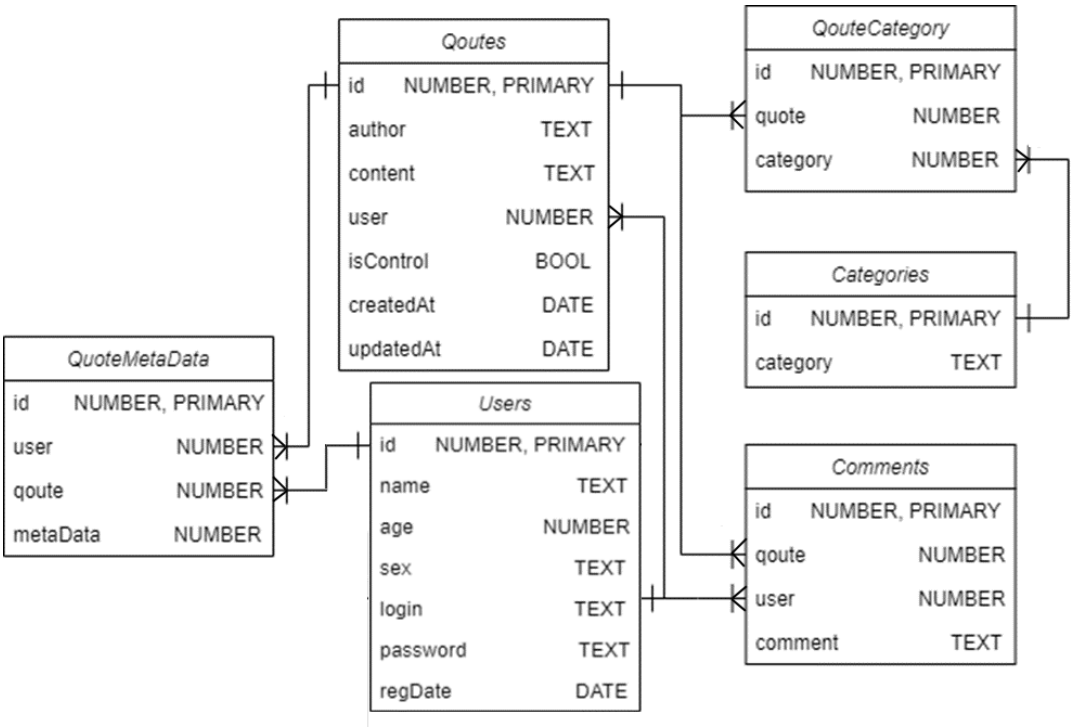

Рисунок 3.5 – Схема загальної БД

3.6 Структура локальної бази даних

Таблиця User містить інформацію про авторизованого користувача. Структура таблиці наведена в табл. 3.7.

Таблиця 3.7 – Структура таблиці User

Назва	Семантика	Тип даних	Обмеження
id	Ідентифікатор користувача	int	Первинний ключ
name	Ім’я користувача	text (20)	
age	Вік користувача	int	
sex	Стать користувача	text (10)	
login	Логін користувача	text (20)	
password	Пароль користувача	text (20)	

Таблиця Quotes містить список цитат, які користувач додав до локальної БД. Структура таблиці наведена в табл. 3.8.

Таблиця 3.8 – Структура таблиці Quotes

Назва	Семантика	Тип даних	Обмеження
id	Ідентифікатор цитати	int	Первинний ключ
content	Текст цитати	text (1000)	
author	Автор цитати	text (200)	
isFavorite	Чи відноситься до улюблених	bool	

Таблиця Category містить список категорій, до яких можуть бути віднесені цитати. Структура таблиці наведена в табл. 3.9.

Таблиця 3.9 – Структура таблиці Category

Назва	Семантика атрибуту	Тип даних	Обмеження
id	Ідентифікатор категорії	int	Первинний ключ
category	Назва категорії	text (200)	

Таблиця Comments містить список всіх коментарів, які користувач додав до цитат. Структура таблиці наведена в табл. 3.10.

Таблиця 3.10 – Структура таблиці Comments

Назва	Семантика атрибуту	Тип даних	Обмеження
id	Ідентифікатор коментаря	int	Первинний ключ
comment	Зміст коментаря	text (200)	
quote	Ідентифікатор цитати	int	Зовнішній ключ

Таблиця QuoteCategory містить співвідношення категорій та цитат. Структура таблиці наведена в табл. 3.11.

Таблиця 3.11 – Структура таблиці QuoteCategory

Назва атрибуту	Семантика атрибуту	Тип даних	Обмеження
id	Ідентифікатор рядка	int	Первинний ключ
quote	Ідентифікатор цитати	int	Зовнішній ключ
category	Ідентифікатор категорії	int	Зовнішній ключ

Таблиця Utils містить список прапорів виконання програми. Структура таблиці наведена в табл. 3.12.

Таблиця 3.12 – Структура таблиці Utils

Назва атрибуту	Семантика атрибуту	Тип даних	Обмеження
id	Ідентифікатор прапорів	int	Первинний ключ
isSettingPassed	Чи виконані налаштування	bool	
isPopupPassed	Чи отримані підказки	bool	
isFavoriteTabOpen	Флаг відкриття улюблених цитат	bool	

Таблиця Notifications містить інформацію про налаштування повідомлень користувачу. Структура таблиці наведена в табл. 3.13.

Таблиця 3.13 – Структура таблиці Notifications

Назва атрибуту	Семантика атрибуту	Тип даних	Обмеження
id	Ідентифікатор повідомлення	int	Первинний ключ
quantity	Кількість повідомлень	int	
startTime	Час початку	int	
endTime	Час закінчення	int	

Таблиця Styles містить список стилів, що можуть бути використані для оформлення зовнішнього вигляду застосунку. Структура таблиці наведена в табл. 3.14.

Таблиця 3.14 – Структура таблиці Styles

Назва атрибуту	Семантика атрибуту	Тип даних	Обмеження
id	Ідентифікатор стилю	int	Первинний ключ
style	Назва стилю	text (200)	

Таблиця QouteMetaData містить інформацію про відношення користувача до цитат. Структура таблиці наведена в табл. 3.15.

Таблиця 3.15 – Структура таблиці QuoteMetaData

Назва	Семантика	Тип даних	Обмеження
id	Ідентифікатор рядка	int	Первинний ключ
quote	Ідентифікатор категорії	int	Зовнішній ключ
metaData	Значення відношення користувача до цитати	int	

Локальна база даних розташована в застосунку і зберігає в собі інформацію необхідну користувачу цього екземпляру застосунку – цитати, що вибрав користувач, інформація про нього, а також дані про системні

налаштування користувача. Схема всіх таблиць локальної БД та зв'язків між ними зображена на рис. 3.6.

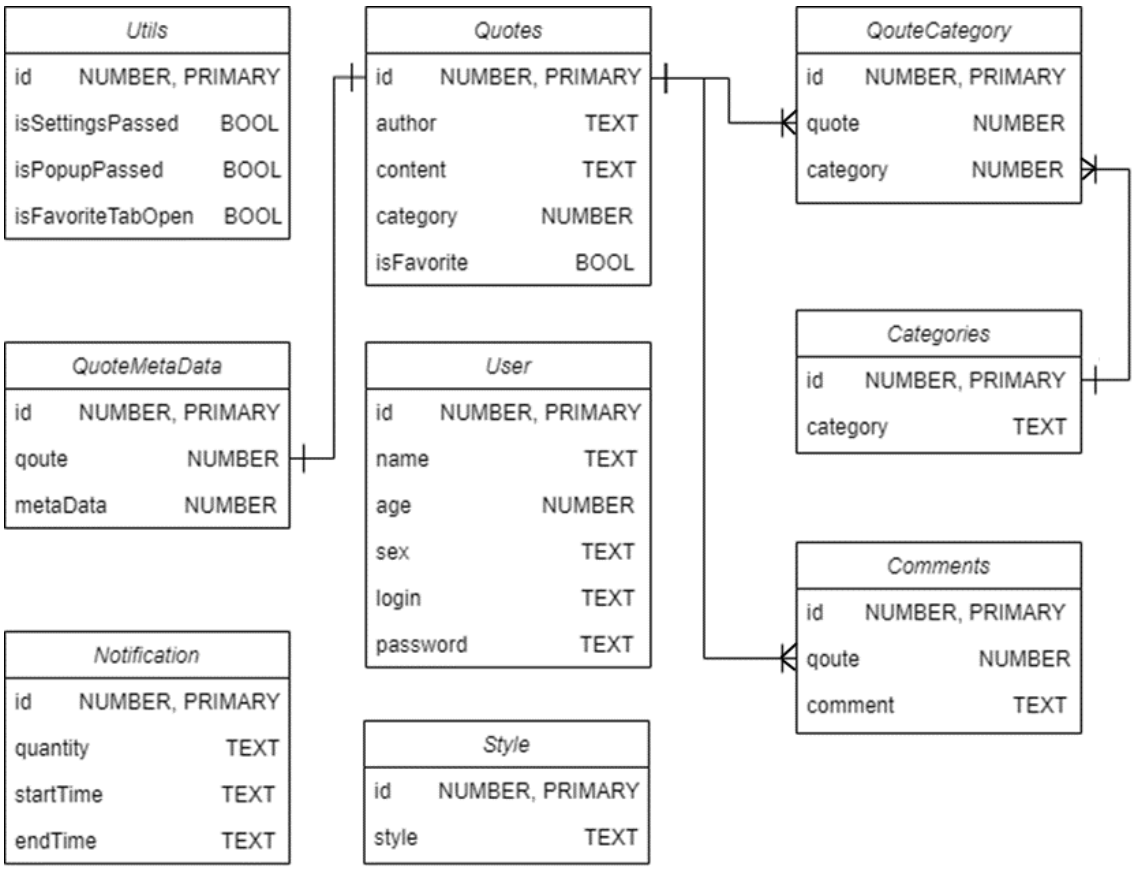

Рисунок 3.6 – Схема локальної БД

3.7 Висновки до розділу

У розділі визначено інструменти та засоби розробки програмного забезпечення, проаналізовано різні шаблони архітектури, та вибрано найбільш відповідну для цілей розробки. На основі вибраної архітектури побудовано та описано структури концептуальних класів та структура загальної та локальної баз даних.

4 РЕАЛІЗАЦІЯ ПРОГРАМИ ДЛЯ ЗБЕРІГАННЯ МНОЖИНИ МОТИВАЦІЙНИХ ЦИТАТ ВІДПОВІДНО ОСОБИСТИМ ПЕРЕВАГАМ

4.1 Опис класів

Нижче описано поля і методи класів, а також зв'язки з іншими класами. На основі цього сформована діаграма програмних класів, частини якої зображені на рис. 4.1-4.4. Розглянемо кожен клас окремо.

Клас ApplicationDataBase, містить сутності Quote, User, Notifications, Utils, Styles і Categories як поля, тобто у вигляді взаємозв'язку композиції. І об'єкти DAO в якості типу повернення абстрактних функцій, тобто у вигляді взаємозв'язку агрегації - quotesDao(): QuotesDao, notificationDao(): NotificationDao, utilsDao(): UtilsDao, styleDao(): StyleDao, themesDao(): CategoriesDao, userDao(): UserDao.

Клас Quote, як і інші класи типу сутностей, не мають методів, а лише поля для зберігання даних. Клас Quote зберігає дані для маніпуляцій і зображення цитат, а саме id - ідентифікаційний номер, який служить первинним ключем, author - строкове значення містить автора цитати, може бути порожнім, content - строкове значення, що містить сам текст цитати, category - тип до якого належить цитата, comments - посилання на всі коментарі, що належать до цієї цитати, isFavorite - логічне значення, що визначає чи подобається цитата користувачеві чи ні.

Клас User зберігає інформацію про користувача програмного продукту, а саме такі поля - id - ідентифікаційний номер, який служить первинним ключем, name - ім'я користувача, age - вік користувача, sex - стать користувача, login - логін користувача, password - пароль користувача.

Клас Utils зберігає прапори, відповідні етапу виконання програми - id - ідентифікаційний номер, який служить первинним ключем, isSettingsPassed - логічне значення, прапор визначає, чи пройшов користувач стартовий екран налаштувань, і блокує його показ, якщо він вже пройдений, isPopupPassed -

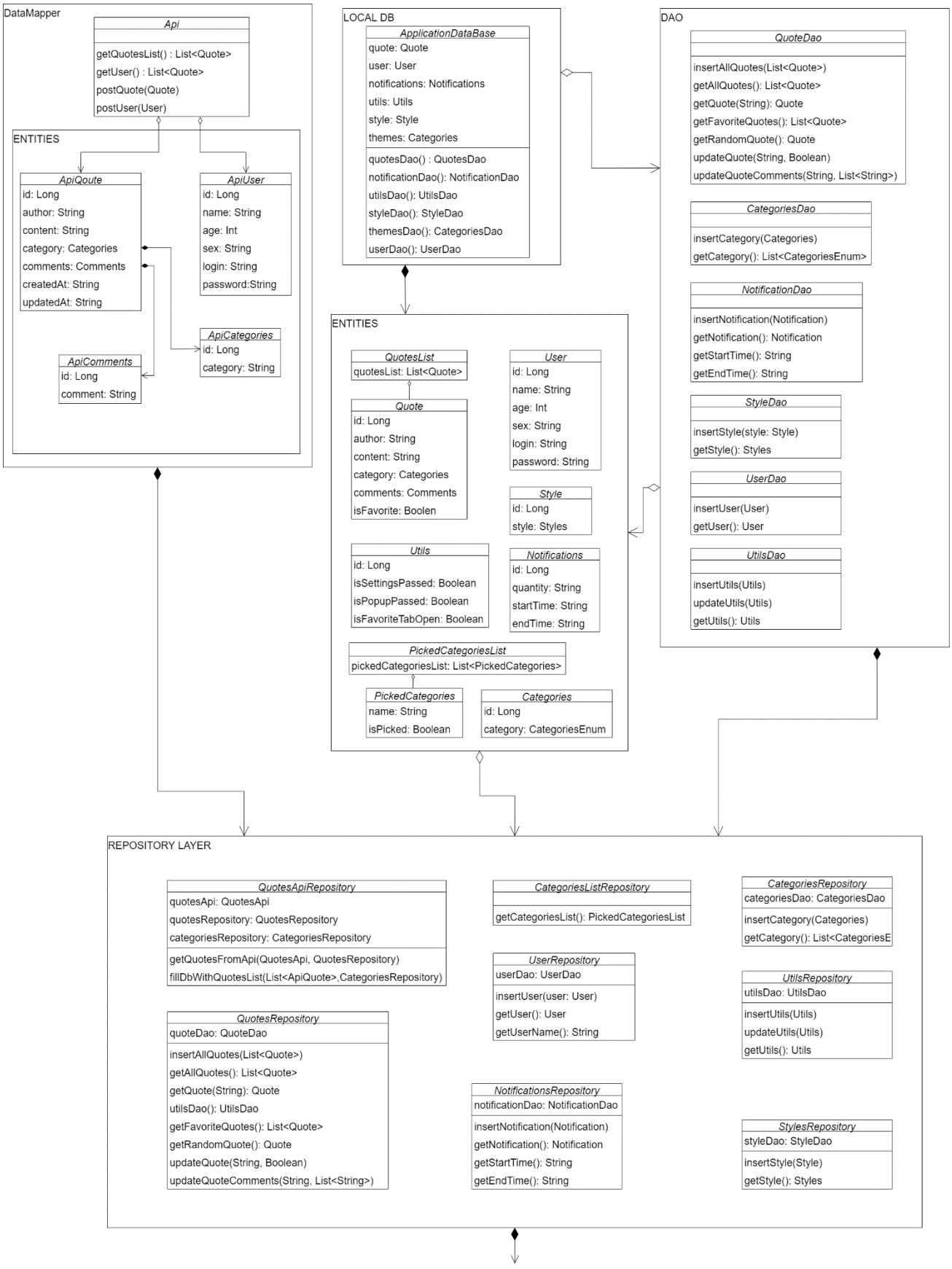

Рисунок 4.1 – Частина «модель» діаграми програмних класів

логічне значення, прапор визначає, чи пройшов користувач спливаюче діалогове меню з підказками, і блокує його показ, якщо він вже пройдений, isFavoriteTabOpen - логічне значення, прапор визначає, показувати користувачеві всі цитати або тільки його улюблені.

Клас Notification зберігає повідомлення, які обрав користувач у вигляді наступних полів: id - ідентифікаційний номер, який служить первинним ключем, quantity - числове значення, кількість повідомлень яку хоче отримати користувач, startTime - час початку показу повідомлень, endTime - час кінця показу повідомлень.

Клас Categories зберігає в собі список можливих категорій цитат у вигляді наступних полів: id - ідентифікаційний номер, який служить первинним ключем, і category - сам список.

Клас Styles - в цьому класі зберігається поточний тип обраної теми програми у вигляді наступних полів: id - ідентифікаційний номер, який служить первинним ключем, і style - назва стилю.

Клас QuoteDao, як і інші класи типу DAO, не мають полів, а лише методи абстракцій звернення до бази даних. Клас QuoteDao має наступні методи: insertAllQuotes(List <Quote>) - приймає на вхід список цитат і записує їх в базу даних, getAllQuotes (): List <Quote> - запитує всі цитати з бази даних і повертає їх у вигляді списку об'єктів цитат, getQuote (String): Quote - запитує одну цитату по її вмісту з бази даних і повертає у вигляді об'єкта цитати, getFavoriteQuotes(): List <Quote> - запитує всі цитати з бази даних зі значенням "улюблені" і повертає їх у вигляді списку об'єктів цитат, getRandomQuote (): Quote - запитує одну випадкову цитату з бази даних і повертає у вигляді об'єкта цитати, updateQuote(String, Boolean) - перезаписує цитату в базі даних, змінюючи її значення в "улюблені" або навпаки, updateQuoteComments(String, List < String>) - перезаписує цитату в базі даних, видаляючи або додаючи поля з коментарями.

Клас UserDao має наступні методи: insertUser(User) - приймає на вхід об'єкт користувача та записує його в базу даних, getUser(): User - запитує користувача з бази даних і повертає у вигляді об'єкта користувача.

Клас CategoriesDao має наступні методи: insertCategory(Categories) - приймає на вхід об'єкт категорії та записує його в базу даних, getCategory(): List <CategoriesEnum> - запитує всі категорії з бази даних і повертає у вигляді списку об'єктів категорій.

Клас StylesDao має наступні методи: insertStyle(style: Style) - приймає на вхід об'єкт стилю і записує його в базу даних, getStyle(): Styles - запитує стиль з бази даних і повертає у вигляді об'єкта стилю.

Клас NotificationDao має наступні методи: insertNotification(Notification) - приймає на вхід об'єкт повідомлень і записує його в базу даних, getNotification(): Notification - запитує повідомлення з бази даних і повертає у вигляді об'єкта повідомлень, getStartTime(): String - запитує час початку посилання повідомлень з бази даних і повертає у вигляді рядка, getEndTime(): String - запитує час кінця посилання повідомлень з бази даних і повертає у вигляді рядка.

Клас UtilsDao має наступні методи: insertUtils(Utils) - приймає на вхід об'єкт прапорів і записує його в базу даних, updateUtils(Utils) - приймає на вхід об'єкт прапорів і перезаписує його значення в базі даних, getUtils(): Utils - запитує прапори з бази даних і повертає у вигляді об'єкта прапорів.

Клас Api служить для отримання і посилання даних на віддалений сервер, для цього він використовує такі методи - getQuotesList(): List <Quote> - відправляє запит по http посиланню і отримує відповідь у вигляді json об'єктів які зберігаються у вигляді об'єктів типу Quote, getUser(): User - відправляє запит по http посиланням і отримує відповідь у вигляді json об'єктів які зберігаються у вигляді об'єктів типу User, postQuote(Quote) - відправляє об'єкт типу Quote у вигляді json об'єкта по http посиланню та отримує відповідь у вигляді коду підтвердження, postUser(User) - відправляє об'єкт типу User у вигляді json об'єкта по http посиланню і отримує відповідь у вигляді коду

підтвердження. Як вхідні та вихідні параметри ці функції використовують класи ApiQuote, ApiCategories, ApiComments, ApiUser, які дублюють вміст аналогічних класів з пакета сутностей, і служать контейнером для даних одержуваних з сервера.

Клас BaseViewModel служить батьківським для інших класів типу viewmodel і має поле state: MutableLiveData <State>() - це поле-контейнер типу liveData, яке успадковують інші класи. Його особливість в тому, що в нього можна поміщати необмежену кількість різних типів даних і отримувати їх за допомогою слухача.

Клас MainActivityViewModel як поля зберігає в собі UtilsRepository і StylesRepository, і реалізує їх методи saveUtils(Utils) і readCurrentStyle().

Клас AboutMeViewModel як поля зберігає в собі UserRepository, і реалізує його методи getUser(): User і insertUser(User).

Клас AccountViewModel як поля зберігає в собі UserRepository і QuotesRepository, і реалізує їх методи getUser(), getFavQuotes(), getAllQuotes(), getUserName().

Клас CategoriesViewModel як поля зберігає в собі UtilsRepository і QuotesRepository, і реалізує їх методи readFavoriteQuotes(), updateUtils(Utils)

Клас CommentsViewModel як поля зберігає в собі QuotesRepository і UserRepository, і реалізує їх методи getQuote(String), updateQuoteComments (String, comments: List <String>), getUserName().

Клас MainViewModel як поля зберігає в собі UtilsRepository, QuotesRepository, NotificationsRepository, і реалізує їх методи readUtils(), readNotification(), getQuotesFromApi().

Клас MotivationViewModel як поля зберігає в собі UtilsRepository і QuotesRepository, і реалізує їх методи readUtils(), readAllQuotesFromQuotesDb(), readFavouriteQuotesFromQuotesDb(), updateUtils(Utils), updateQuote(String, Boolean).

Клас YouViewModel як полів зберігає в собі UserRepository, і реалізує його метод insertUser(User).

Клас NotificationSettingsViewModel як полів зберігає в собі NotificationsRepository, і реалізує його метод saveNotification(Notification).

Клас StyleEditorViewModel як поля зберігає в собі StylesRepository реалізує його метод saveCurrentStyle(Style).

Клас CategoriesPickerViewModel як поля зберігає в собі UtilsRepository, CategoriesListRepository і CategoriesRepository і реалізує їх методи getThemeList(), updateUtils(Utils) і insertTheme(Categories).

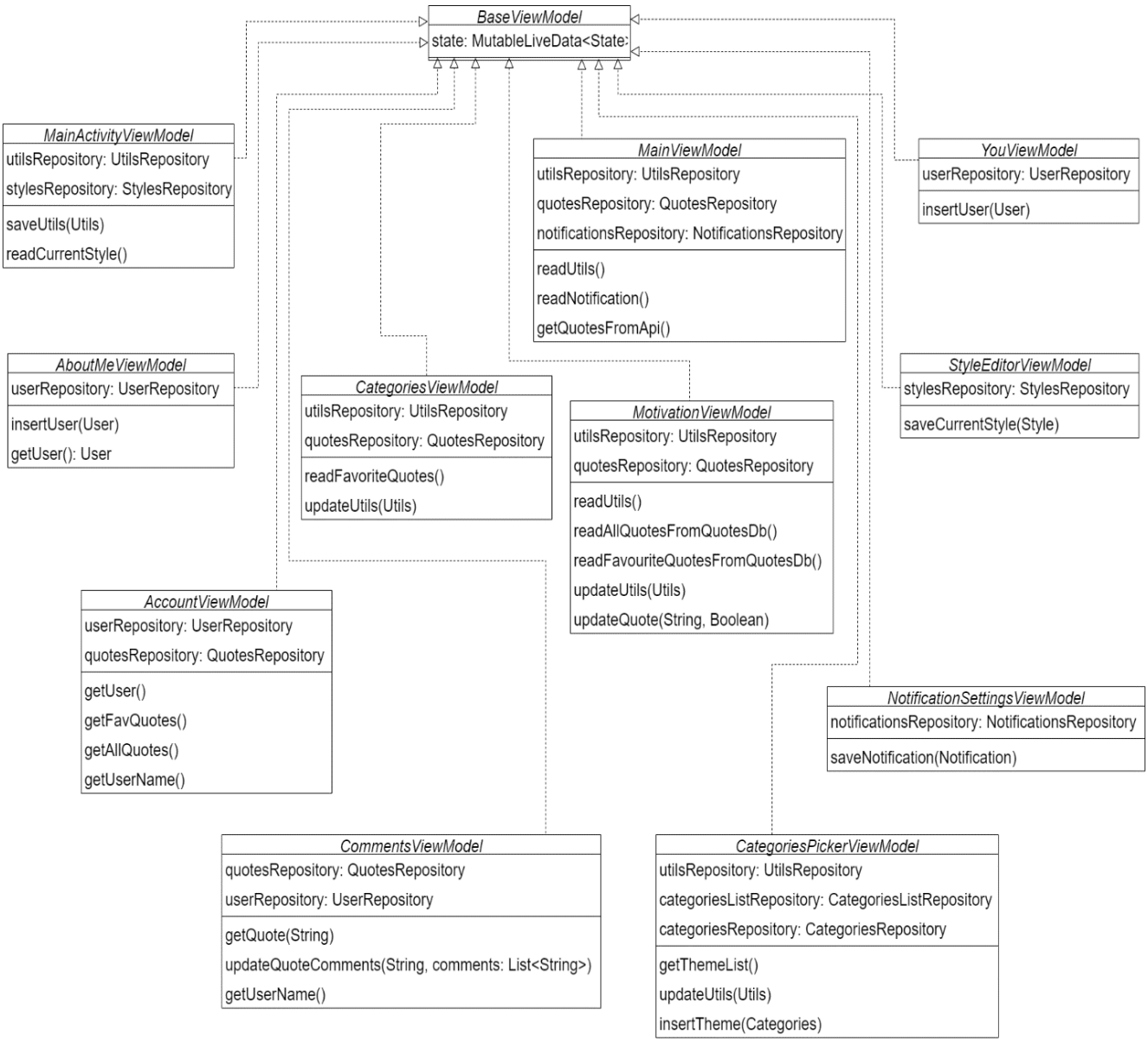

Рисунок 4.2 – Частина «модель представлення» діаграми програмних класів

Клас MainActivity є контейнером для всіх інших вікон програми. MainActivity як поля зберігає в собі MainActivityViewModel - модель представлення для цієї активності та ActivityMainBinding - клас, що зв'язує XML-розмітку з відповідним програмним класом. Методи цього класу - onCreate() - метод життєвого циклу активності, що спрацьовує при відкритті екрану і викликає метод initializeObserver() - слухач змін моделі представлення та navGrap(AboutMeViewModel, AccountFragment, CategoriesFragment, CommentsFragment, MainFragment, MotivationFragment, NotificationSettingsFragment, StyleEditorFragment, CategoriesPickerFragment, YouFragment) - навігаційний граф, що керує стеком переходів між усіма екранами застосунку.

Клас CategoriesPickerFragment відповідає за екран на якому користувач може вибрати цитати яких категорій він хоче переглядати. Як поля зберігає в собі CategoriesPickerViewModel - модель представлення для цього фрагмента, FragmentCategoriesPickerBinding - клас, що зв'язує XML-розмітку з відповідним програмним класом, ThemesRecyclerViewAdapter - клас-адаптер, що слугує для представлення списку даних у вигляді однотипних елементів на екрані з можливістю їх перегортання. Методи цього класу -onViewCreated() - метод життєвого циклу фрагмента в якому оголошуються всі параметри конфігурації, onCreateView() - метод життєвого циклу фрагмента в якому викликаються інші методи активності. Методи життєвого циклу схожі за дією у всіх інших класах фрагментах. setAnimations() - метод в якому відбувається установка і запуск анімацій активності, initializeObserver() - слухач змін моделі представлення, initializeRecyclerView() - метод ініціалізації класу-адаптер, onBntContinueClicked() - метод в якому оголошується слухач натискання на кнопку.

Клас MotivationFragment відповідає за екран, на якому безпосередньо відбувається показ цитат. Як поля зберігає в собі MotivationViewModel - модель представлення для цього фрагмента, FragmentMotivationBinding - клас, що зв'язує XML-розмітку з відповідним програмним класом,

QuotesViewPagerAdapter - клас-адаптер, що слугує для представлення списку даних у вигляді однотипних екранів з можливістю їх перегортання, MotivationFragmentArgs - список вхідних параметрів класу, які були передані йому при переході з іншого класу, position: Int - числове значення що показує позицію поточного елемента на екрані. Методи цього класу – initializeObserver() - метод, який спостерігає за змінами в моделі представлення, getQuotes(isFavoriteTabOpen: Boolean) - метод, який визначає які цитати виводити на екран, все або тільки улюблені, initializePopup(utils: Utils) - метод виводить на екран спливаюче вікно з підказками, переді цим контрольний прапор, initializeViewPager() - метод в якому відбувається ініціалізація класу адаптера, changeData() - оскільки цитат в базі даних тисячі, зобразити їх всіх відразу затратно по часу, тому в залежності від значення змінної position, якщо вона перевищує 32 метод виконує перезавантаження даних.

Клас AboutMeFragment відповідає за екран, де користувач може переглянути та змінити особисту інформацію, а також подивитися улюблені цитати та залишені коментарі. Як поля зберігає в собі AboutMeViewModel - модель представлення для цього фрагмента, FragmentAboutMeBinding - клас, що зв'язує XML-розмітку з відповідним програмним класом, AboutMeRecyclerAdapter - клас-адаптер, що слугує для представлення списку даних у вигляді однотипних екранів з можливістю їх перегортання, CommentsFragmentArgs - список вхідних параметрів класу, які були передані йому при переході з іншого класу. Методи цього класу - initializeObserver() - метод, який спостерігає за змінами в моделі представлення, specifyNavArgs() - метод визначає який з методів установки запустити в залежності від параметрів класу, setUpUserInfoView() - метод установки виду даних користувача, setUpFavoriteQuotesView() - метод установки виду улюблених цитат користувача, setUpUserComments() - метод установки всіх коментарів користувача, onClickChangeProfile() - метод запуску спливаючого вікна з

редагуванням інформації користувача, initializeRecyclerView() - метод ініціалізації класу адаптера.

Клас CommentsFragment відповідає за екран на якому користувач може залишити коментар. Як поля зберігає в собі CategoriesViewModel - модель представлення для цього фрагмента, FragmentCategoriesBinding - клас, що зв'язує XML-розмітку з відповідним програмним класом. Методи цього класу - initializeObserver() - метод, який спостерігає за змінами в моделі представлення, initializeRecyclerView() - метод ініціалізації класу адаптера, onSendButtonClicked () - метод установки слухача натискань на кнопку.

Клас YouFragment відповідає за екран, де користувач повинен зареєструватися або авторизуватися. Як поля зберігає в собі YouViewModel - модель представлення для цього фрагмента, FragmentYouBinding - клас, що зв'язує XML-розмітку з відповідним програмним класом. Методи цього класу - disableKeyboard() - метод спрацьовує при натисканні на будь-яке місце на екрані, згортає системну клавіатуру, initSpinner() - метод ініціалізації списку, setAnimations() - метод установки та запуску анімацій активності, onClickBtnGetStarted() - метод установки слухача натискань на кнопку переходу на наступний екран.

Клас AccountFragment як поля зберігає в собі AccountViewModel - модель представлення для цього фрагмента, FragmentAccountBinding - клас, що зв'язує XML-розмітку з відповідним програмним класом. Методи цього класу - initObserver() - метод, який спостерігає за змінами в моделі в моделі представлення, setOnClickListeners() - метод установки слухача натискань на кнопки.

Клас StyleEditorFragment як поля зберігає в собі StyleEditorViewModel - модель представлення для цього фрагмента, FragmentStyleEditorBinding - клас, що зв'язує XML-розмітку з відповідним програмним класом. Методи цього класу - onBtnLightClicked(), onBtnDarkClicked(), onBtnBlueClicked(), onBtnMixClicked() - методи установки слухача натискань на кнопки відповідають за вибір тем для програми.

Клас MainFragment як поля зберігає в собі MainViewModel - модель представлення для цього фрагмента, FragmentMainBinding - клас, що зв'язує XML-розмітку з відповідним програмним класом. Методи цього класу - initializeObserver() - метод, який спостерігає за змінами в моделі представлення, chooseActivityToOpen(Utils) - метод, що вибирає який екран відкрити наступним залежно від значення прапора, openMotivationFragment () - метод відкриття фрагмента, setNotificationWorker(Notification) - метод настройки роботи класу Worker, що відповідає за посилання повідомлень, а також регулярної відправки інформації про відношення користувача до цитат для формування матриць кореляцій для системи персональних рекомендацій.

Клас CategoriesFragment як поля зберігає в собі CategoriesPickerViewModel - модель представлення для цього фрагмента, FragmentCategoriesPickerBinding - клас, що зв'язує XML-розмітку з відповідним програмним класом, ThemesRecyclerViewAdapter - клас-адаптер, що слугує для представлення списку даних у вигляді однотипних екранів з можливістю їх перегортання. Методи цього класу - setAnimations() - метод встановлює і запускає анімації активностей, initializeObserver() - ініціалізація слухача змін моделі представлення, initializeRecyclerView() - ініціалізація класу адаптера, onBntContinueClicked() - метод установки слухача натискань на кнопку переходу на наступний екран та збереження в базі даних інформацію щодо вибраних категорій.

Клас NotificationSettingsFragment як поля зберігає в собі NotificationSettingsViewModel - модель представлення для цього фрагмента, FragmentNotificationSettingsBinding - клас, що зв'язує XML-розмітку з відповідним програмним класом, isStartTimer: Boolen - логічне значення, необхідне для визначення який з таймерів запущений, notificationQuantity: Int - числове значення кількості повідомлень, hour: Int - числове значення кількості годин, minute: Int - числове значення кількості хвилин. Методи цього класу - setAnimations() - метод встановлює і запускає анімації активностей, onClickButtons() - метод установки слухача натискань всіх кнопок на екрані,

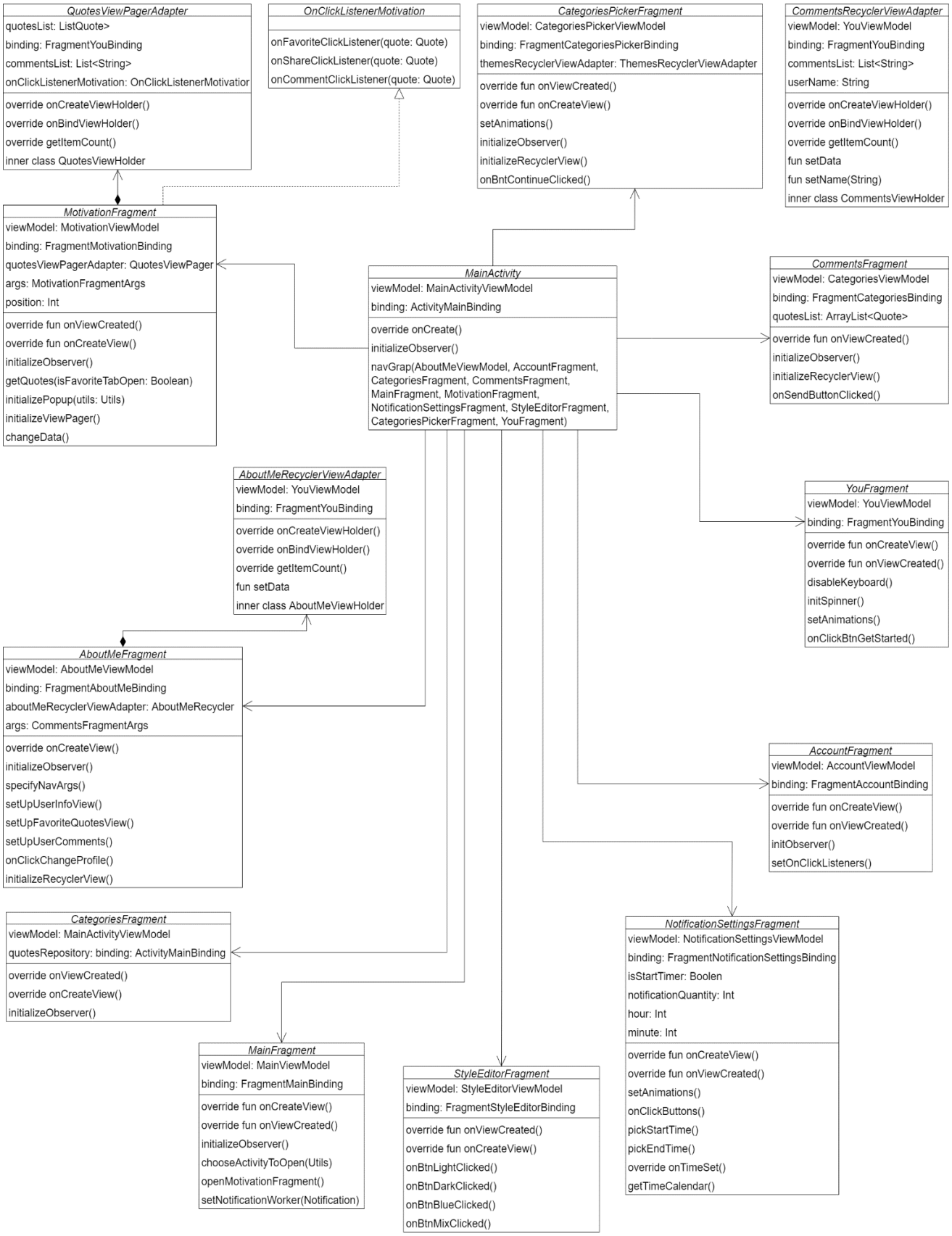

Рисунок 4.3 – Частина «представлення» діаграми програмних класів

pickStartTime() - слухач змін часу таймера початку посилання повідомлень, pickEndTime() - слухач змін часу таймера кінця посилання повідомлень, onTimeSet() - метод збереження і зображення змін в таймерах, getTimeCalendar() - метод отримання поточного часу, onBntContinueClicked() - метод установки слухача натискань на кнопку переходу на наступний екран та збереження в базі даних інформацію щодо налаштування повідомлень. Клас Users зберігає інформацію про всіх користувачів системи, а саме такі поля - id - ідентифікаційний номер, який служить первинним ключем, name - ім'я користувача, age - вік користувача, sex - стать користувача, login - логін користувача, password - пароль користувача.

Тепер розглянемо діаграму програмних класів віддаленого сервера. Вона зображена на рис. 4.4.

Клас Quotes, як і інші класи сутностей, має структуру даних, що відповідає таблиці загальної БД. Клас Quotes зберігає дані для цитат, для відправки їх користувачам, а саме id - ідентифікаційний номер, який служить первинним ключем, author - строкове значення містить автора цитати, може бути порожнім, content - строкове значення, що містить сам текст цитати, category - тип до якого належить цитата, comments - посилання на всі коментарі, що належать до цієї цитати, metadata - містить інформацію про відношення користувача до цитат.

Класи QuotesRepo і UserRepo реалізують клас JpaRepo, що входить в склад Spring фреймворку і надає готове рішення для реалізації DAO класів. Ці класи мають однакові методи, однак працюють з різними типами даних. Це методи – findById() – знаходить запис в БД за ідентифікаційним ключом, findAll – знаходить всі записи в БД, save() – зберігає вказаний елемент в БД, saveAll() – зберігає всі записи в БД, delete() – видаляє вказаний елемент з БД, deleteAll() – видаляє всі записи в БД.

Методи класів типу контролер не викликаються у програмі а мають реалізацію зворотного виклику і реагують на http запити. Кожен метод має прив’язку до своєї унікальної http адреси, також кожен метод має мітку на який

тип запиту треба реагувати – get, post, put, delete тощо. Разом з вихідними даними такий метод повертає код відповіді, який може бути дво- , трьох-, чотирьох, п’яти- сотого рівня та вказувати на коректність результату або помилку та точну вказівку її причини.

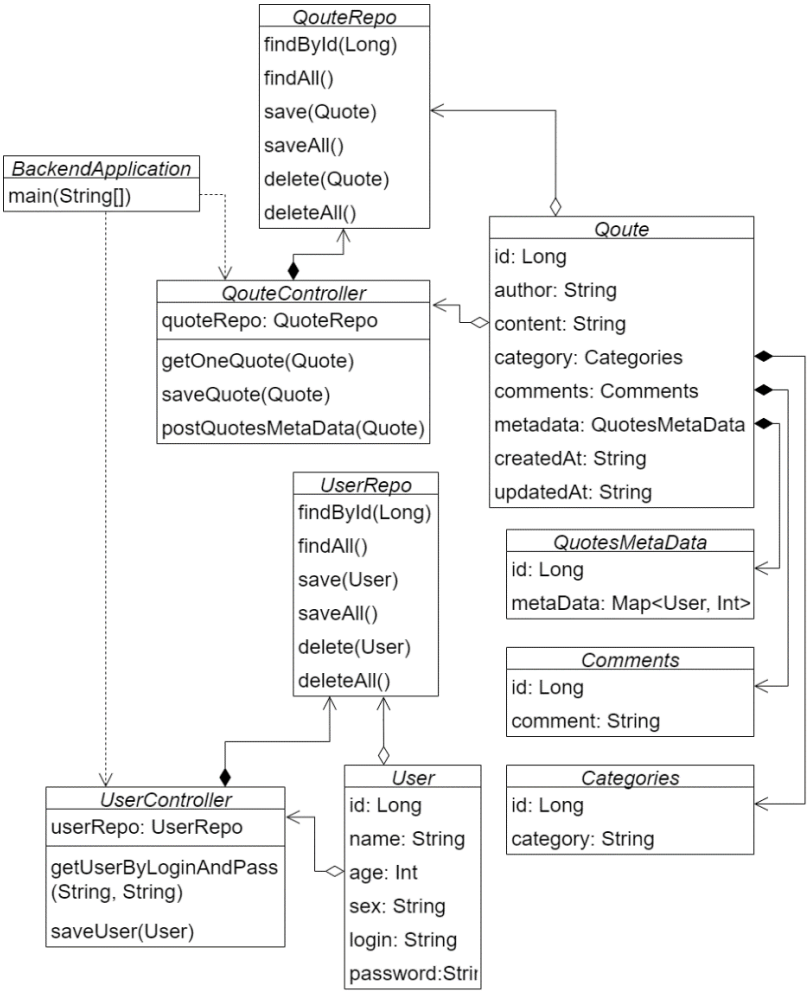

Рисунок 4.4 – Діаграма програмних класів віддаленого сервера

Клас UserController зберігає клас UserRepo як поле, та розширює його методи до необхідних для роботи з даними. Метод getUserByLoginAndPass(String, String) – приймає на вхід логін і пароль користувача, і якщо такий користувач зберігається в БД, повертає об’єкт з його даними. Метод saveUser(User) – зберігає нового користувача у БД.

Клас QouteController зберігає клас QuotesRepo як поле, та розширює його методи до необхідних для роботи з даними. Метод getAllQuotes - повертає

всі записи про цитати в БД, метод getOneQuote – знаходить на повертає одну цитату з БД, метод saveQuote - зберігає вказану цитату в БД, метод postQuoteMetaData – відправляє інформацію про відношення користувача до цитат до сервісу рекомендацій.

4.2 Сервіс рекомендацій

В загальній базі даних є близько десяти тисяч мотиваційних цитат. Незалежно від того скільки і які теми вибрав користувач, йому на пристрій буде завантажено не більше двох тисяч цитат. Крім того, навіть якщо два користувачі виберуть однакові теми, їм будуть додані різні цитати. Кожна цитата має її рейтинг, або так звані метадані. Рейтинг формується залежно від реакції користувача та вибраних ваг. Як реакції мається на увазі додавання цитати в категорію " подобається ", чи поділився користувач цитатою і скільки разів він це зробив, а також скільки часу сумарно користувач переглядав цю цитату. Назвемо дію «подобається» *like*, яка може приймати значення 0 або 1, вагу для неї назвемо $w1$, дію «поділитися» назвемо *share*, вона дорівнює кількості разів, що користувач поділися цитатою, вагу для неї назвемо $w2$, а кількість часу перегляду назвемо *minute*, і вона дорівнює кількості хвилин перегляду, вагу для неї назвемо $w3$. Результат функції кореляції назвемо r. Кореляція вподобання повинна бути в діапазоні від 0 до 1, тому отриманий результат ділиться на сто, якщо результат становить більше ста, то він прирівнюється до ста. Отже, загальна формула буде:

$$r = (w1 * like + w2 * share + w3 * minute)/100$$

Оскільки дія «поділитися» є проявом найсильнішої симпатії користувача, вага була обрана 15. Для додавання в категорію “ подобається ” була обрана вага 10. Для часу перегляду вага 4 за кожну хвилину. Для кожної цитати, для кожного користувача рейтинг рахується окремо. Для створення

матриці кореляцій, а також для аналізу вхідних та отриманих даних використовувалася бібліотека Pandas corr. Дії, які були виконані для обробки даних: спочатку були зібрані дані, які будуть використовуватися для кореляційної матриці. Потім було створено DataFrame , щоб передати вказаний вище набір даних у Python. Потім було згенеровано кореляційну матрицю, використовуючи шаблон Pandas corr. Також були використані пакети seaborn і matplotlib, щоб отримати візуальне подання кореляційної матриці. На рис. 4.5 зображено візуалізацію найпростішої кореляційної матриці на десять цитат. Раз на день модуль worker. Звертається на віддалений сервер та запитує нові цитати. Сервер у свою чергу звертається до сервісу рекомендацій та отримує з нього дата кадру матриці кореляцій. Завдяки йому відбувається заміна цитат на пристрої користувачів. Цитати зі значенням менше 0,5 видаляються, а більше 0,5 додаються, але так, щоб у користувача було завантажено не більше двох тисяч цитат.

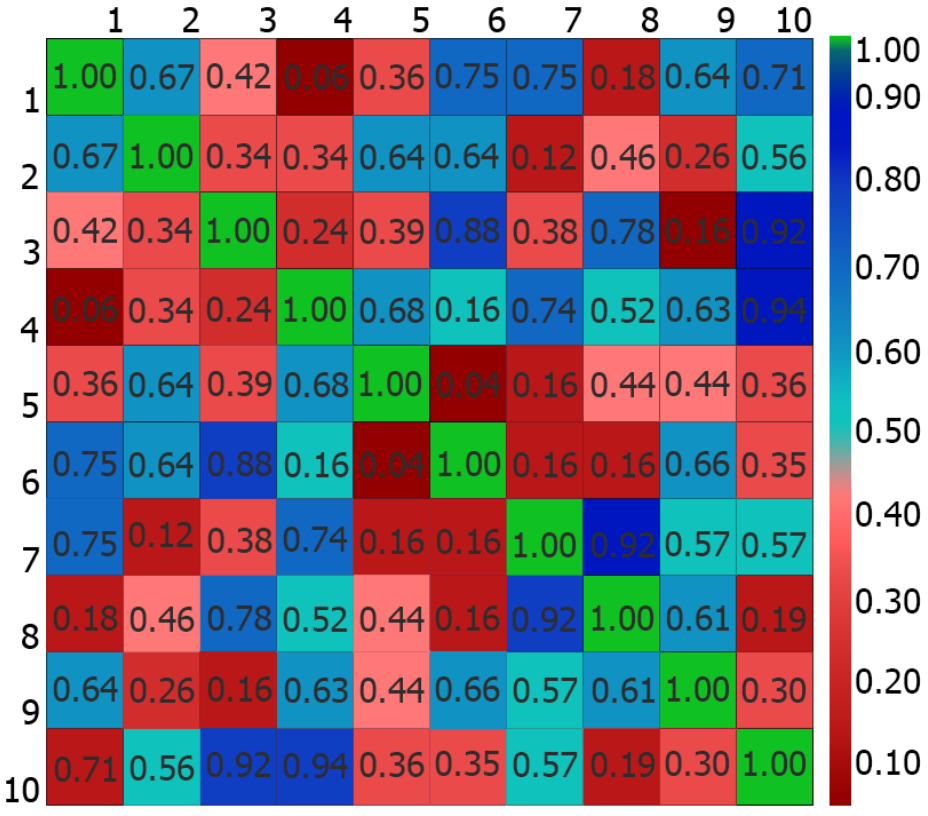

Рисунок 4.5 – Матриця кореляцій для десяти цитат

4.3 Інтерфейс користувача

Після установки і запуску мобільного застосунку відкривається ряд вікон, що відповідають за ознайомлення користувача зі застосунком, реєстрацією і первинним налаштуванням застосунку. Вони показуються тільки один раз при першому запуску. Однак якщо користувач закрив застосунок, не пройшовши до кінця ці вікна, вони будуть показані заново при наступному відкритті.

Перший екран (рис. 4.6) служить для вітання користувача, він містить текст з інформацією о цілі застосування і всього одну кнопку переходу на наступний екран.

Рисунок 4.6

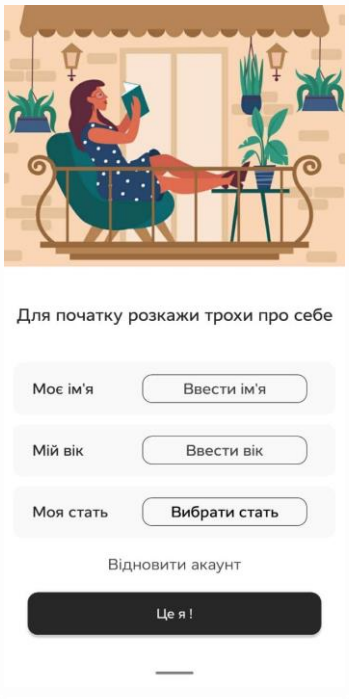

Рисунок 4.7

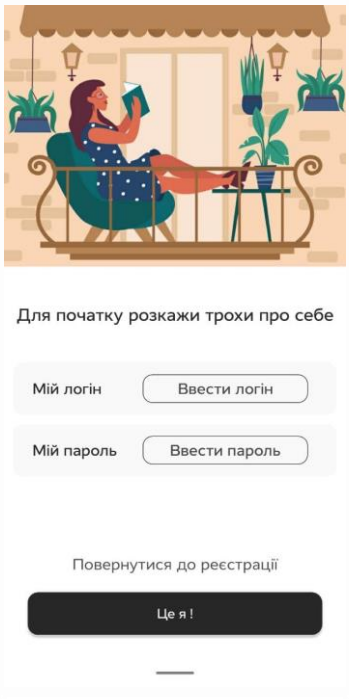

Рисунок 4.8

Екран реєстрації та авторизації показані на рис. 4.7 та 4.8 відповідно. На екрані користувачеві пропонується ввести свої персональні дані, а саме ім'я, стать і вік, для кращих рекомендацій цитат для нього в подальшому. Поки користувач не заповнить всі три поля, він не зможе перейти на наступні екрани. Якщо всі поля заповнені, то при натисканні на кнопку користувач буде

зареєстрований, і перейде на наступний екран. Також якщо користувач користувався застосунком з іншого пристрою, він може натиснути на кнопку "відновити акаунт" і замість полів з персональними даними з'являться поля для введення логіна і пароля, якщо вони введені правильно, при натисканні на кнопку, користувач авторизується і зможе перейти на наступний екран.

Застосунок може відправляти користувачеві цитати прямо в центр повідомлень. Екран налаштування отримування повідомлень показаний на рис. 4.9. На екрані є три функціональних елементи. На першому можна налаштувати кількість одержуваних повідомлень, на другому - з якого часу починати отримувати повідомлення, і третій - в який момент припиняти показувати повідомлення. Однак повідомлення не будуть надсилатися частіше, ніж один раз в п'ятнадцять хвилин. При натисканні на кнопку буде відкритий наступний екран.

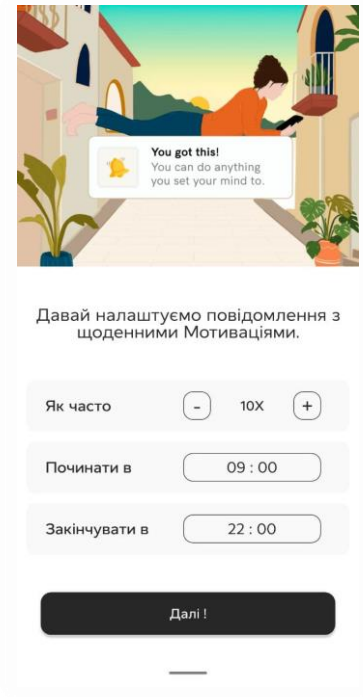

Рисунок 4.9

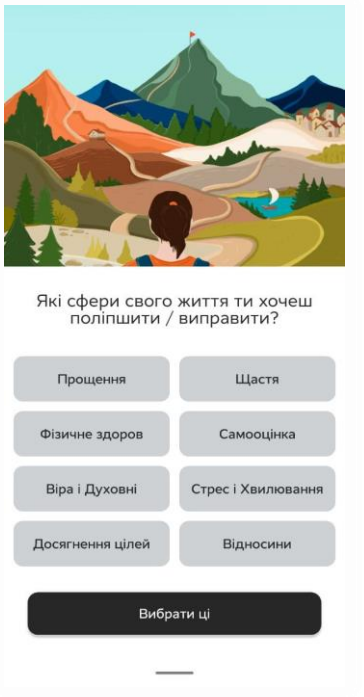

Рисунок 4.10

Останнім з екранів початкового налаштування є екран вибору категорій (рис. 4.10). На даний момент цитати в застосунку згруповані на вісім категорій,

що відрізняються за посилом і настроєм. На цьому екрані користувач може зробити первинну настройку персоніфікації і вибрати тільки теми, що його цікавлять. Після цього він може натиснути на кнопку і перейти на наступний екран. Якщо жодна тема не обрана, користувач не зможе натиснути на кнопку і на екрані з’явиться підказка про необхідність вибору теми.

Екрани для роботи з мотиваційними цитатами показано на рис. 4.11 та 4.12, відповідно. Це головний екран застосунку. Крім першого запуску і початкового налаштування робота застосунку буде починатися саме з цього екрану.

Рисунок 4.11

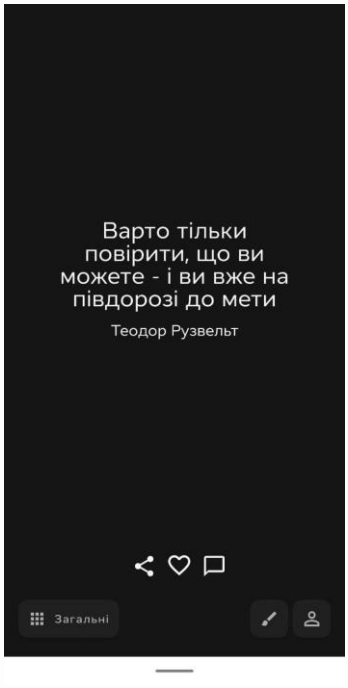

Рисунок 4.12

На екрані присутній сам текст цитати, а також три кнопки, що відносяться до цієї цитати, і три кнопки переходу на інші екрани застосунку. Під час перегляду екрана вгору або вниз цитати і пов'язані з ним кнопки будуть заміняти один одного. При натисканні на кнопку з символом "Поділитися" система Android знайде всі програми на телефоні, які підтримують функції передачі тексту, і сформує з них список, запропонувавши користувачеві

вибрати одне з них для відправки цитати (рис. 4.13). Це більшість месенджерів, сервісів електронної пошти, повідомлення, замітки і так далі. При натисканні на кнопку з символом серця, вона змінить свій колір на червоний, а сама цитата додасться в окремий список "улюблених цитат" (рис. 4.14). При натисканні на кнопку з символом коментаря буде відкритий екран, щоб залишити коментар (рис. 4.15). При натисканні на кнопку з символом плитки буде відкритий екран вибору категорій цитат. При натисканні на кнопку з символом кисті буде відкритий екран вибору тем програми. При натисканні на кнопку з символом "Акаунт" буде відкритий екран особистого кабінету.

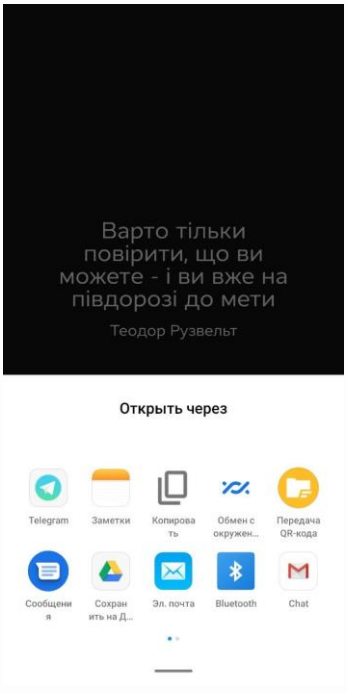

Рисунок 4.13

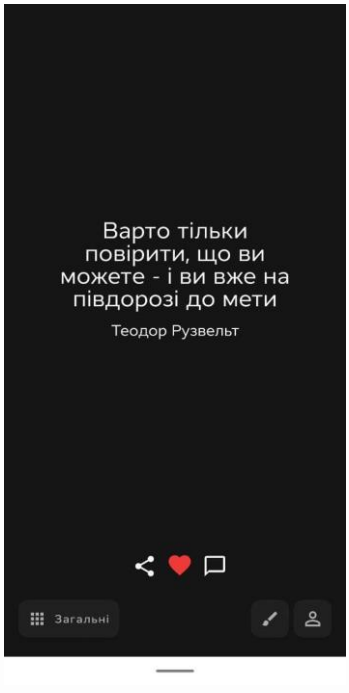

Рисунок 4.14

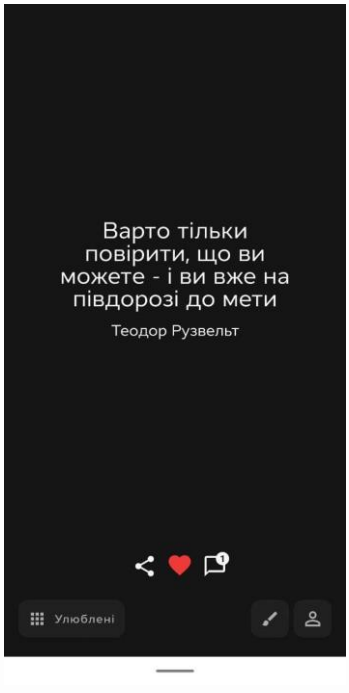

Рисунок 4.15

Екран коментарів (рис. 4.16) має такі елементи. Знизу екрану присутнє поле введення і кнопка "Відправити". Після введення тексту і натиснення на кнопку на екрані з'явитися блок з ім'ям автора і текстом його коментаря. Також якщо цій цитаті залишали коментарі інші користувачі, вони так само будуть відображені тут.

На екрані вибору категорій (рис. 4.17) присутні дві кнопки вибору категорій: "Загальні" - це все цитати вибрані користувачем; "Улюблені" - список цитат, які користувач відзначив як улюблені.

На екрані вибору стилів (рис. 4.18) присутні чотири кнопки з різним кольорами, відповідними кольорам стилів, на які вони змінюють інтерфейс застосунку.

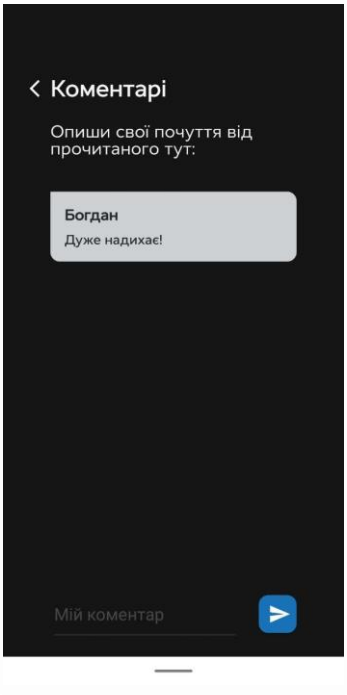

Рисунок 4.16

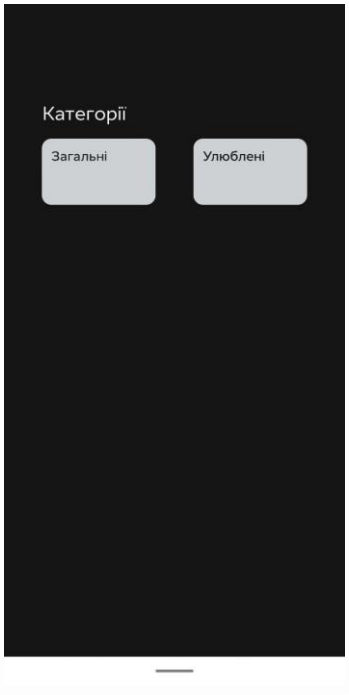

Рисунок 4.17

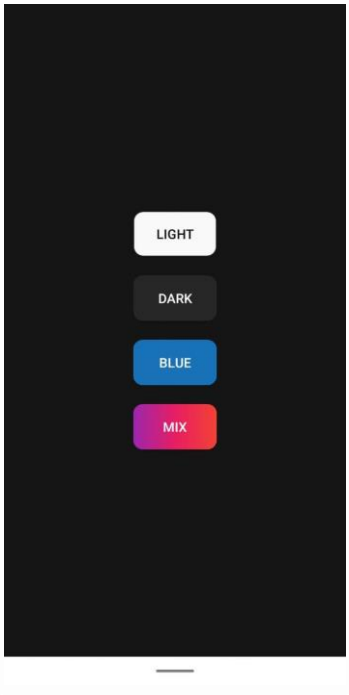

Рисунок 4.18

На екрані особистого кабінету (рис. 4.19) присутні наступні кнопки. Кнопка "Інформація про мене" відкриває список особистих даних користувача, які він вказував при реєстрації: ім'я, стать і вік (рис. 4.20). Кнопка "Змінити дані" відкриває діалогове вікно, в якому можна редагувати дані, а також додати логін і пароль (рис. 4.21). Кнопка "Улюблені цитати" відкриває список всіх улюблених цитат користувача, надає той же функціонал, що і розділ "Улюблені", проте відображає цитати у вигляді списку, що полегшує читання і пошук. Кнопка "Мої коментарі" відкриває список всіх цитат користувача, залишених їм під усіма коментарями.

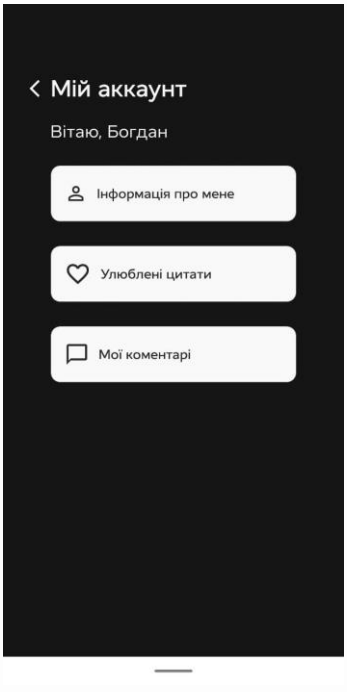

Рисунок 4.19

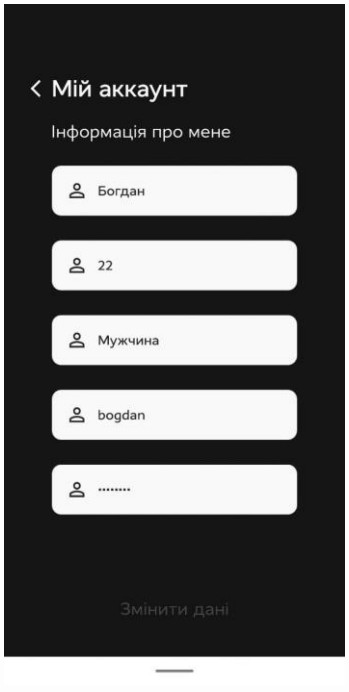

Рисунок 4.20

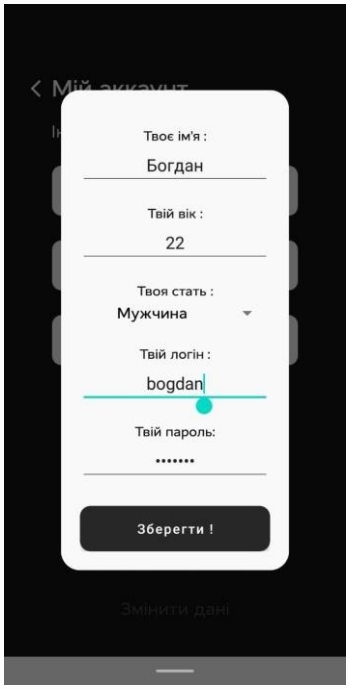

Рисунок 4.21

4.4 Висновки до розділу

В розділі описано програмні класи, які створені на основі концептуальних класів. Надано детальний опис методів класів, які розподілені на частини відповідно архітектурному шаблону. Також визначений принцип роботи сервісу рекомендацій. Описано графічний інтерфейс застосунку у вигляді інструкції використання.

5 ТЕСТУВАННЯ ПРОГРАМИ ДЛЯ ЗБЕРІГАННЯ МНОЖИНИ МОТИВАЦІЙНИХ ЦИТАТ ВІДПОВІДНО ОСОБИСТИМ ПЕРЕВАГАМ

5.1 Тестові випадки

Експлуатація розподілених інформаційних систем потребуэ посилених вимог до перевірки їх функціонування [27]. Тестування містять тестові приклади, які перевіряють роботу кожної одиниці коду (окремих класів або малих груп пов’язаних функцій) незалежно від його середовища. Такі тестові випадки не повинні мати зовнішніх залежностей. Вони зосереджені на одній, відокремленій від усього, функціональності. Це дозволяє їм забезпечити вичерпне покриття низькорівневого коду, недоступного для більших тестів [29].

Обмежене покриття та відсутність зовнішніх залежностей означають, що просте тестування можна виконувати дуже швидко. Тому їх часто можна запускати і швидко знаходити помилки.

Тестовий випадок (тест-кейс) – це певний перелік дій, переважних умов і функцій, які необхідні для перевірки того, наскільки готова до використання функціональність, що тестується, або певна окрема функція [30].

Тестовий приклад містить компоненти, які описують введення, дію та очікувану відповідь, щоб визначити, чи правильно працює функція програми.

Тестовий приклад — це набір інструкцій щодо «як» для перевірки конкретної мети/цілі тесту, які, якщо їм слідувати, покажуть, чи задовольняється очікувана поведінка системи [30].

В загальному випадку тестові випадки мають таку структуру [31]:

- назва;
- короткий опис;
- передумови;
- кроки для розмноження;
- очікуваний результат.

Передумовами є поєднання всіх необхідних підготовчих кроків (налаштувань програми, середовища тестування), необхідних для виконання окремого тестового прикладу.

Тестовий випадок для варіанту використання «Додавання цитати до загальної БД», який представляє позитивний тест, має вигляд, як показано в табл. 5.1.

Таблиця 5.1 – Позитивний тестовий випадок для варіанту використання «Додавання цитати до загальної БД»

Назва	Нова цитата в загальній БД. Позитивний тест		
Опис	Перевірка варіанту використання «Додавання цитати до загальної БД»		
Передумова	Користувач авторизований як модератор, знаходиться у формі перегляду цитат		
Крок	Опис	Очікуваний результат	Виконання
1	Натиснути кнопку Додати цитату	На екрані створена форма, яка містить поля для вказівки тексту цитати та її автора, а також список можливих категорій	Успішно
2	Встановити фокус на поле тексту цитати	На екрані з’явилося клавіатура для введення тексту	Успішно
3	Ввести текст цитати	У полі відображається введений текст	Успішно
4	Встановити фокус на поле автору цитати	На екрані з’явилося клавіатура для введення тексту	Успішно

Продовження табл. 5.1

Крок	Опис	Очікуваний результат	Виконано
5	Ввести автора цитати	У полі відображається введений текст	Успішно
6	Відкрити список категорій	На екрані з’явилися назви категорій	Успішно
7	Обрати категорію	Обрана категорія відображається іншим кольором	Успішно
8	Натиснути кнопку Зберегти	На екрані відображається список цитат, додана цитата присутня у списку	Успішно

Тестовий випадок для варіанту використання «Додавання цитати до загальної БД» », який представляє негативний тест, має вигляд, як показано в табл. 5.2.

Таблиця 5.2 – Негативний тестовий випадок для варіанту використання «Додавання цитати до загальної БД»

Назва		Нова цитата в загальній БД. Негативний тест	
Опис		Перевірка варіанту використання «Додавання цитати до загальної БД» щодо валідації помилок	
Передумова		Користувач авторизований як модератор, знаходиться у формі додавання цитати	
Крок	Опис	Очікуваний результат	Виконання
1	Натиснути кнопку Зберегти	На екрані з’явилося повідомлення про відсутність тексту цитати	Успішно

Продовження табл. 5.2

Крок	Опис	Очікуваний результат	Виконання
2	Встановити фокус на поле тексту цитати	На екрані з’явилося клавіатура для введення тексту	Успішно
3	Ввести текст цитати	У полі відображається введений текст	Успішно
8	Натиснути кнопку Зберегти	На екрані з’явилося повідомлення про відсутність автору цитати	Успішно
4	Встановити фокус на поле автору цитати	На екрані з’явилося клавіатура для введення тексту	Успішно
5	Ввести автора цитати	У полі відображається введений текст	Успішно
8	Натиснути кнопку Зберегти	На екрані з’явилося повідомлення про відсутність обраної категорії	Успішно

Тестовий випадок для варіанту використання «Зміна стилю застосунку» має вигляд, як показано в табл. 5.3.

Таблиця 5.3 – Тестовий випадок для варіанту використання «Зміна стилю застосунку»

Назва	Нова стиль відображення цитат		
Опис	Перевірка варіанту використання «Зміна стилю застосунку»		
Передумова	Користувач авторизований як власник БД		
Крок	Опис	Очікуваний результат	Виконання
1	Натиснути кнопку Light	Відображення на екрані змінили колір на світлі тона	Успішно

Продовження табл. 5.3

Крок	Опис	Очікуваний результат	Виконання
2	Натиснути кнопку Dark	Відображення на екрані змінили колір на темні тона	Успішно
3	Натиснути кнопку Dark	Відображення на екрані змінили колір на блакитні тона	Успішно
4	Натиснути кнопку Mix	Відображення на екрані змінили колір на багатокольорові тона	Успішно

Тестовий випадок для варіанту використання «Авторизація для загальної БД», який представляє негативний тест, має вигляд, як показано в табл. 5.4.

Таблиця 5.4 – Негативний тестовий випадок для варіанту використання «Авторизація для загальної БД»

Назва	Авторизація для загальної БД. Негативний тест		
Опис	Перевірка варіанту використання «Авторизація для загальної БД» щодо валідації помилок		
Передумова	Користувач відкрив застосунок		
Крок	Опис	Очікуваний результат	Виконання
1	Натиснути кнопку «Авторизуватися»	На екрані з’явилися поля для введення логіну та паролю	Успішно
2	Натиснути кнопку «Ввійти»	На екрані з’явилося повідомлення про відсутність логіну	Успішно

Продовження табл. 5.4

Крок	Опис	Очікуваний результат	Виконання
3	Встановити фокус на поле введення логіну	На екрані з’явилося клавіатура для введення тексту	Успішно
4	Ввести невірний логін	У полі відображається введений текст	Успішно
5	Натиснути кнопку «Ввійти»	На екрані з’явилося повідомлення про невірний логін	Успішно
6	Встановити фокус на поле введення логіну	На екрані з’явилося клавіатура для введення тексту	Успішно
7	Ввести вірний логін	У полі відображається введений текст	Успішно
8	Натиснути кнопку «Ввійти»	На екрані з’явилося повідомлення про відсутність паролю	Успішно
9	Встановити фокус на поле введення паролю	На екрані з’явилося клавіатура для введення тексту	Успішно
10	Ввести невірний пароль	У полі відображається введений текст у скритому вигляді	Успішно
11	Натиснути кнопку «Ввійти»	На екрані з’явилося повідомлення про невірний пароль	Успішно

Таким чином, розроблені тестові випадки дають можливість перевірити готовність до випуску функцій програми, які відповідають варіантам використання.

5.2 Експериментальне дослідженння роботи застосунку

5.2.1 Дослідження сервісу рекомендацій. Для дослідження системи рекомендацій проведено ряд експериментів, щоб з’ясувати, скільки потрібно надати системі цитат, даних з відношенням користувачів, які ваги встановлювати на кожен з параметрів, та при яких результатах функції рекомендацій довавати чи видаляти цитати.

Для проведення дослідження трьом людям було запропонованно різні версії застосунку з різними налаштуваннями та різним набіром цитат. Тестувальники користувались кожною версією застосунку протягом двох тижнів, кожен день реагуючи мінімум сто цитат. Параметри та умови налаштування наведено в таблиці 5.5.

Таблиця 5.5 – Визначення параметрів та умов налаштування

Умови налаштування		Параметри налаштування				
Користувач	Версія	Кількість цитат в базі даних	Кількість цитат з рекціями	Ваги		
				like	*share*	*minute*
1	1	5 000	1980	15	10	4
	2	7 000	1554	20	12	4
	3	10 000	2154	8	6	3
	4	3 000	2080	9	5	3
2	1	10 000	1685	15	10	4
	2	5 000	1808	20	12	4
	3	3 000	1693	8	6	5
	4	7 000	2029	9	5	3
3	1	3 000	1625	15	10	4
	2	10 000	1867	20	12	4

Продовження табл. 5.5

Користувач	Версія	Кількість цитат в базі даних	Кількість цитат з рекціями	Ваги		
				like	*share*	*minute*
3	3	7 000	1744	8	6	3
	4	5 000	1611	9	5	3

Далі всі версії застосунків передали трьом іншим користувачам та запропонували їм використувувати застосунок в зручному для них режимі, але кожного дня. Раз в три дні на сервісу рекомендацій змінювали значення, при якому результати функції рекомендацій необхідно довавати чи видаляти цитати. Наприкінці користування кожен користувач надавав субъективну оцінку за стобальною шкалою, наскільки він задоволений показаними цитатами. Далі розраховано середнє арифметичне оцінок. Результат реакцій користувачів наведено в таблиці 5.6.

Таблиця 5.6 – Порівняння оцінок користувачів

Версія	Значення фукціїї, при якому цитати додаються користувачу	Оцінка
1	> 0.4	78
	> 0.5	94
	> 0.6	82
2	> 0.4	52
	> 0.5	65
	> 0.6	59
3	> 0.4	74
	> 0.5	87
	> 0.6	82
4	> 0.4	64
	> 0.5	79
	> 0.6	78

Проаналізувавши дану таблицю, можемо зробити висновок, що оптимальною версією є версія з вагами 15, 10 та 4, а також с максимальною кількітю цитат – 10 000. Також користувачі обрали версію зі значенням функції рекомендації більше 0.5, тому що при меньшому значенні якість цитат знижується, а при більшому кількість повторювання цитат різко зростає.

5.2.2 Порівняльний аналіз якості систем. Для підтвердження виконання поставленої мети треба провести порівняльний аналіз застосунків в плані витрати часу. Повний функціонал розробленого застосунку не покриває жодне з існуючих застосувань, тому порівняння проводилося між різними функціональними частинами та різними застосунками.

Для порівняння основного функціоналу використано сервіси Motivetica та Quatato.

Якщо не використовувати застосування з кваліфікаційної роботи, то алгоритм дій виглядає наступним чином.

1. Завантажити застосування Motivetica.
2. Переглядати цитати, поки не знайдеться та, що подобається.
3. Скопіювати текст цитати.
4. Завантажити застосування Quatato.
5. Додати цитату.
6. Переглядати, додавати свої, поділитися.

Якщо використовувати застосунок з кваліфікаційної роботи, то алгоритм дій виглядає наступним чином.

1. Завантажити застосування.
2. Обрати теми, що цікавлять.
3. Переглядати цитати .
4. Додати цитату до улюблених, подади коментар, поділитися.

Обидва алгоритми дій були протестовані групою студентів за двома критеріями: швидкість взаємодії та якість цитат. Для розрахунку швидкості користувачам запропонували знайти десять цитат, що їм подобаються, додати

до улюблених та поділитися. В кваліфікаційній роботі увесь необхідний функціонал знаходиться в мобільному застосунку, тому весь процес тестування відбувається безпосередньо в ньому. Таких можливості у аналогів немає, тому весь етап переглядання цитат відбувався у застосунку Motivetica, а перегляд, додавання своїх цитат та розповсюждення в застосуванні Quatato, що потребує більше дії та займає більше часу. Детальні результати представлені в таблиці 5.7.

Таблиця 5.7 – Порівняння часових характеристик застосунків

Характеристика програми / Назва програми	Перегляд	Перенос	Додавання	Поділитися	Всього часу
Motivetica+ Quatato	12 хв	4 хв	2 хв	1 хв	19 хв
Motivation	7 хв	-	1 хв	1 хв	9 хв

5.3 Висновки до розділу

В розділі описано спосіб тестування програмного продукту з використанням тестових випадків, наведені детальні описи тестових випадків для окремих варіантів використання.

В результати експеримента було з’ясованно що, використовуючи застосунок з кваліфкаційної роботи, вдалося зменшити час на отримання інформації в 2,1 раз, а також зменшити кількість дії та покращити якість показаних цитат. Таким чином, поставлена мета кваліфікаційної роботи досягнута в повному обсязі.

ВИСНОВКИ

В роботі вивчені існуючі на ринку програмні засоби з функціоналом для доступу до мотиваційних цитат, виконаний порівняльний аналіз з розробленим застосунком.

Описані вимоги до застосунку, функціональні, у вигляді варіантів використання, та нефункціональні.

Визначена архітектура програми. Розроблено концептуальні, а далі програмні класи.

Створено зручний інтерфейс для отримання доступу до функціоналу програми.

Під час розробки використані сучасні засоби розробки, зокрема Kotlin, Java, Android, JetPack, Spring, Pyton.

Таким чином, як результат роботи реалізований мобільний застосунок для підтримки множини цитат, які є мотиваційними для окремої персони.

Розроблена програма підтримується операційною системою Android, може бути встановлена на мобільний пристрій для легкого доступу до нею у будь-який час.

Застосунок може працювати локально, з власним збереженим набором цитат, або віддалено, отримуючи доступ до більш широкої множини цитат.

Використовуючи застосунок з кваліфікаційної роботи, вдалося зменшити час на отримання інформації в 2,1 раз, а також зменшити кількість дій та покращити якість показіних цитат. Таким чином, поставлена мета кваліфікаційної роботи досягнута в повному обсязі.

Результати роботи апробовані на XIV Міжнародної науково-практичної конференції «Інформаційні технології і автоматизація – 2021», Одеса, 21 - 22 жовтня 2021 р.

СПИСОК ЛІТЕРАТУРИ

1. Болтышев И. А Шелягович А.С. Разработка мобильных приложений на языке программирования JAVA. 57-я научная конференция аспирантов, магистрантов и студентов БГУИР, 2021 г. URL: https://libeldoc.bsuir.by/bitstream/123456789/43612/1/Boltyshev_Razrabotka.pdf

2. Разработка мобильных приложений от А до Я: полный гайд. URL: https://dan-it.com.ua/blog/razrabotka-mobilnyh-prilozhenij-ot-a-do-ja-polnyj-gajd/01.01.20

3. Martynyuk A. M., Ahmesh Tamem, Thuong Bui Van, Drozd O. V. Multilevel Behavioral Testing of Distributed Information Systems. Herald of Advanced Information Technology. 2019; Vol. 2 No.4: p. 298–309. DOI: http://doi.org/10.15276/hait.04.2019.6

4. Статистика мобильных приложений 2021: загрузки, тренды и доходность индустрии. URL: https://vc.ru/marketing/245003-statistika-mobilnyh-prilozheniy-2021-zagruzki-trendy-i-dohodnost-industrii

5. Типы мобильных приложений URL: https://punicapp.com/blog/pages/1046/tipy-mobilnyh-prilozhenij

6. 215 лучших мотивационных цитат. URL: https://www.plerdy.com/ru/blog/top-215-motivational-quotes/

7. Ромашов В. Рынок мобильных приложений в 2020 году. URL: http://android.mobile-review.com/articles/67321/

8. Перспективы рынка мобильных приложений. URL: https://artjoker.ua/ru/blog/perspektivy-rynka-mobilnykh-prilozheniy/

9. Sydow L. Signs of a Recovery: Mobile App Usage Reveals Which Sectors are Rebounding and Which Mobile Habits Are Here to Stay. URL: https://www.appannie.com/en/insights/market-data/mobile-market-pandemic-recovery-signs/

10. Одноroженко Т. Дайджест: рынок мобильных приложений в 2020 году. URL: https://asomobile.net/blog/digest2020/

11. Чёрная Е. Этапы разработки мобильного приложения: аналитика и техническое задание. URL: https://vc.ru/dev/142571-etapy-razrabotki-mobilnogo-prilozheniya-analitika-i-tehnicheskoe-zadanie 30 июл 2020

12. Казовская Д. Этапы разработки мобильного приложения: пьеса в 7 действиях. URL: https://www.azoft.ru/blog/app-development-process/

13. Мотивирующие цитаты и фразы. URL: https://resheto.net/raznosti/75-motiviruyushchie-tsitaty

14. Мотивирующие цитаты: 100+ цитат, которые вдохновляют. URL: https://lafounder.com/article/motivaciya-citaty

15. Мотивирующие цитаты от Canva. URL: https://www.canva.com/ru_ru/citaty/motivaciya/

16. Motivetica is an app that gives you everyday motivation in Helvetica. URL: https://motivetica.com/

17. QUOTATO. Simple notebook for quotes. URL: https://apps.apple.com/ru/app/quotato/id642037182?l=en

18. ДивенСвіт. URL: https://dyvensvit.org/tag/tsytaty/

19. Абрамова А. USE CASES. Что это такое и зачем они нужны? URL: https://systems.education/use-case

20. Об’єктно-орієнтований підхід до розробки вимог. URL: https://infopedia.su/14x247.html

21. Функціональне моделювання. URL: https://posibniki.com.ua/post-funkcionalne-modelyuvannya

22. Визначення вимог через прецеденти. URL: https://zdamsam.ru/a68258.html

23. Features Functional Requirements. URL: https://uk.myservername.com/features-functional-requirements#Types_Of_Non-functional_Requirements

24. Паттерн MVVM. Определение паттерна MVVM. URL: https://metanit.com/sharp/wpf/22.1.php\

25. Как пользоваться Android Studio. URL: https://timeweb.com/ru/community/articles/kak-polzovatsya-android-studio

26. Kotlin is now Google’s preferred language for Android app development. URL: https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/

27. Kravchenko I. A. Speranskiy V. A. Cross-Platform Practices for Mobile Application Development of Automated Trade Accounting. Applied Aspects of Information Technology. 2018; Vol. 1 No.1: 48–58. DOI: https://doi.org/10.15276/aait.01.2018.3.

28. Знакомство с android architecture components и MVVM. URL: https://medium.com/@nyavorskii/знакомство-с-android-architecture-components-и-mvvm-перевод-29654672f4ab

29. How to prepare test run. URL: https://geteasyqa.com/qa/prepare-test-run/

30. Why we should use preconditions when designing test cases. URL: https://testmatick.com/why-we-should-use-preconditions-when-designing-test-cases/

31. How To Write Test Cases: The Ultimate Guide With Examples. URL: https://www.softwaretestinghelp.com/how-to-write-effective-test-cases-test-cases-procedures-and-definitions/

32. Test case structure & example in software qa outsourcing. URL: https://testmatick.com/test-case-structure-example-in-software-qa-outsourcing/

ДОДАТОК А КОД ПРОГРАМИ

Код отримання цитат

```
interface QuotesApi {
    @GET("./")
    suspend fun getQuotesList(): Response<List<ApiQuote>>
}
class QuotesApiRepository @Inject constructor(
    private val quotesApi: QuotesApi,
    private val quotesRepository: QuotesRepository,
    private val categoriesRepository: CategoriesRepository
) {

    suspend fun getQuotesFromApi() {
        try {
            val response = quotesApi.getQuotesList()
            if (response.isSuccessful) {
                val apiList = response.body()
                val quotesList = fillQuotesList(apiList)
                quotesRepository.insertAllQuotes(quotesList)
            } else {
                Log.w("DEBUG", "getQuotesFromApi() Wrong Response"
                   + response.errorBody().toString())
            }
        } catch (e: Exception) {
            Log.w("DEBUG", "getQuotesFromApi() Wrong Mapping $e")
        }
    }

    private suspend fun fillQuotesList(apiList: List<ApiQuote>?)
       : List<Quote> {
        val quotesList = ArrayList<Quote>()
        val pickedThemes = categoriesRepository.getCategory()
        apiList?.forEach {
            try {
                if (pickedThemes.contains(CategoriesEnum.valueOf
                   (it.theme ?: CategoriesEnum.motavation.name))) {
                    quotesList.add(
                        Quote(
                            0,
                            it.content ?: "",
```