Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Мобільний застосунок для створення каталогу домашньої бібліотеки

(тема)

Виконав:
здобувач 3 року навчання,
групи КІУКІу-22-1
Леонід БОКЛАГ
(власне ім’я, прізвище)

Спеціальність
123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп’ютерна інженерія
(повна назва освітньої програми)

Керівник: ас. Віталій СІТНІКОВ
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім’я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Боклагу Леоніду Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи Мобільний застосунок для створення каталогу домашньої бібліотеки

затверджена наказом по університету від “ 25 ” травня 2025 р. № 425 СТ

2. Термін подання здобувачем роботи до екзаменаційної комісії 14 липня 2025 р.

3. Вхідні дані до роботи 1) Система автоматизованого збирання Gradle
1) середовище розробки Android Studio;
2) мова програмування Kotlin;
3) система автоматизованого збирання Gradle.

4. Перелік питань, що потрібно опрацювати у роботі
1) аналіз існуючих рішень;
2) обґрунтування та вибір технологій розробки;
3) розробка алгоритмічного забезпечення;
4) розробка програмних модулів;
5) відлагодження програмних модулів;
6) висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд-презентація – 10 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз існуючих методів вирішення задачі	10.06.2025-13.06.25	
2	Вибір програмного забезпечення та	14.06.2025-17.06.25	
	інструментів розробки		
3	Проєктування архітектури застосунку	18.06.25-21.06.25	
4	Розробка логіки застосунку	22.06.25-28.06.25	
5	Розробка графічного інтерфейсу користувача	29.06.25-02.07.25	
6	Тестування застосунку	03.07.25-05.07.25	
7	Подання кваліфікаційної роботи керівникам	06.07.25-09.07.25	
	для попереднього захисту		
8	Подання кваліфікаційної роботи на	10.07.25-11.07.25	
	рецензування		

Дата видачі завдання " 09 " червня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Віталій СІТНІКОВ
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 66 с., 18 рис., 1 дод., 9 джерел.

КАТАЛОГ, РЕЦЕНЗІЯ, МОБІЛЬНИЙ ЗАСТОСУНОК, ІНТЕРФЕЙС КОРИСТУВАЧА, ДОСВІД КОРИСТУВАЧА, ПОШУК, СОРТУВАННЯ, АВТОРИЗАЦІЯ, РЕЄСТРАЦІЯ.

Метою роботи є створення мобільного застосунку для платформи Android, який забезпечує зручну організацію особистої бібліотеки, обмін враженнями від прочитаних книг, а також підтримує базову соціальну взаємодію між користувачами. Для досягнення поставленої мети були застосовані методи: аналіз існуючих програм-аналогів, вивчення інструментів розробки для Android, проєктування архітектури застосунку, реалізація програмної частини та проведення тестування (функціонального, інтеграційного та UI-тестування).

Предметом дослідження є процес розробки мобільного застосунку для ведення персональної електронної бібліотеки користувача. Об’єктом дослідження виступає програмне забезпечення, що дозволяє здійснювати пошук, збереження, рецензування книг, а також взаємодію між користувачами в межах книжкової спільноти.

У результаті розроблено мобільний застосунок, що дозволяє користувачам шукати книги, зберігати їх на віртуальну книжкову полицю, залишати рецензії, оцінювати рецензії інших користувачів та переглядати бібліотеки інших учасників. Також реалізовано можливість доступу до облікового запису з різних мобільних пристроїв. Розроблене програмне забезпечення є повністю працездатним і готовим до подальшого вдосконалення.

ABSTRACT

Bachelor’s thesis: 66 pages, 18 figures, 1 appendix, 9 sources.

CATALOGUE, REVIEW, MOBILE APPLICATION, USER INTERFACE, USER EXPERIENCE, SEARCH, SORTING, AUTHORIZATION, REGISTRATION.

The purpose of the work is to create a mobile application for the Android platform, which provides convenient organization of a personal library, exchange of impressions from read books, and also supports basic social interaction between users. To achieve the goal, the following methods were used: analysis of existing similar programs, study of development tools for Android, design of the application architecture, implementation of the software part and testing (functional, integration and UI testing).

The subject of the study is the process of developing a mobile application for maintaining a user's personal electronic library. The object of the study is software that allows searching, saving, reviewing books, as well as interaction between users within the book community.

As a result, a mobile application was developed that allows users to search for books, save them to a virtual bookshelf, leave reviews, rate other users' reviews, and browse other participants' libraries. The ability to access the account from different mobile devices was also implemented. The developed software is fully functional and ready for further improvement.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API – програмний інтерфейс програми (англ., Application Programming Interface)

DAO – об'єкт доступу до даних (англ., Data Access Object)

IDE – інтегроване середовище розробки (англ., Integrated Development Environment)

JVM – віртуальна машина Java (англ., Java Virtual Machine)

MVVM – архітектура Модель-Подання-Модель Подання (англ., Model-View-ViewModel)

RAD – концепція організації процесу розробки, орієнтована на швидке отримання результату за умов обмежень за термінами (англ., Rapid Application Development)

URL – єдиний покажчик ресурсу (англ., Uniform Resource Locator)

XML – мова розмітки, що розширюється (англ., eXtensible Markup Language)

ВСТУП

Створення каталогу домашньої бібліотеки – це практичне рішення, яке приносить одразу кілька суттєвих переваг. Насамперед, він дозволяє систематизувати наявні книги, упорядкувати їх за жанрами, авторами, темами або іншими критеріями, що значно полегшує пошук потрібної книги, особливо якщо колекція вже налічує десятки чи сотні екземплярів.

Використання каталогу допомагає уникнути дублювання при нових покупках. Іноді може виникнути ситуація, що людина купує книги, які вже наявної в домашній колекції, просто тому, що вона забула про її наявність. Такого не буде, якщо є каталог.

Також каталогізація сприяє свідомому читанню. Використовуючи каталог можна відзначати книги, які вже прочитані, а також сформувати для себе перелік тих, які людина бажає прочитати. Це зручно для планування, наприклад, можна скласти список літератури на рік або виділити книги певної тематики.

Ще один корисний аспект – це можливість ділитися інформацією про свою бібліотеку з друзями чи родичами. Каталог можна легко відправити в електронному вигляді або показати, відмітити того, хто має бажання взяти книгу на якийсь час. При цьому за допомогою каталогу зручно фіксувати, кому і коли було позичено ту чи іншу книгу, щоб нічого не втратити.

Наявність каталогу також особливо цінна при переїздах, перестановках та прибиранні. Він допомагає зберегти порядок та швидко відновити розташування книг на полицях. Іноді людина переміщується між кількома місцями, наприклад, дача, офіс, квартира. І слідкування за книгами в даному випадку – це важлива задача.

Для колекціонерів та бібліофілів каталог стає ще й інструментом оцінки своєї колекції: можна відслідковувати, яких авторів чи жанрів більше, які книги рідкісні, дорогі чи видані в унікальних форматах.

Крім того, ведення каталогу розвиває уважність, дисципліну та любов до порядку. Цей процес може бути захоплюючим хобі саме по собі, особливо якщо користувач доповнює каталог своїми анотаціями, особистими враженнями від книг або рейтингами.

Таким чином, каталогізація домашньої бібліотеки – це не просто спосіб упорядкувати книги, а повноцінний інструмент для організації інтелектуального життя, збереження пам'яті, планування читання та спілкування з однодумцями.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Види та місце каталогу у бібліотеці

Каталог – це мозок бібліотеки, її пам'ять, де зберігаються відомості про кожне видання, прізвище автора, назву книги, місто, де вона надрукована, видавництво та рік видання.

Бібліотечні каталоги виникли у давнину. Бібліотеки Стародавнього світу, як і сучасні, мали величезну кількість документів і орієнтуватися в них без каталогів було неможливо. Свідченням наявності таких каталогів є фрагменти, що дійшли до нас: написи на стінах стародавніх храмових бібліотек, глиняні таблички з каталожними записами та ін.

«Класичний» паперовий бібліотечний каталог складається з каталожних карток, на які записують кожну книгу, що надійшла до бібліотеки. Каталожна картка несе інформацію про книгу та зміст: відомості записуються у порядку, який має назву «бібліографічний опис». На каталожній картці вказують прізвище, ім'я та по батькові автора, назву книги, місце видання, видавництво, рік видання, кількість сторінок, наявність ілюстрацій та інші дані. Ці дані, як правило, беруться із титульного аркуша книги. Але, крім цих відомостей, необхідно знати місце книги на полиці, тобто її адресу. Такою адресою виступає шифр книги, який на каталожній картці міститься у верхньому лівому кутку і записується як дроб, де у чисельнику ставиться індекс, а в знаменнику – авторський знак. Верхня частина шифру (індекс) вказує відділ, де знаходиться книга, а нижня частина – місце всередині відділу. Таким чином, кожна книга отримує своє строго певне місце.

За допомогою такого каталогу можливо дізнатися:

- наявність книги у бібліотеці;
- рік видання, обсяг книги, видавництво;

- уточнити назву книжки, якщо автор відомий;
- які твори автора є у бібліотеці;
- з анотації є можливість дізнатися короткий зміст книги.

Найбільш популярними в бібліотеці є алфавітні, систематичні та електронні каталоги. Але існують і інші.

Алфавітний каталог – це бібліотечний каталог, у якому бібліографічні записи розміщуються в алфавітному порядку імен осіб, найменувань організацій та/або назв документів.

Архівний каталог – це довідник, у якому знаходиться інформація про архівні документи, що розміщені відповідно до обраної схеми класифікації.

Генеральний каталог – це каталог, що відображає весь бібліотечний фонд, за винятком обмінних та резервних фондів.

Географічний каталог – це каталог, у якому записи розміщуються відповідно до місця видання документів у алфавітному порядку.

Видавничий каталог – це каталог, що містить список видань, випущених видавництвом за певний період.

Краєзнавчий каталог – це регіональний бібліотечний каталог, який відображає документи краєзнавчого змісту.

Систематичний каталог – це бібліотечний каталог, у якому бібліографічні записи розміщуються за галузями знання відповідно до певної системи класифікації документів.

Країнознавчий каталог – це регіональний бібліотечний каталог, що відображає документи, що належать до змісту для певної країни.

Топографічний каталог – це бібліотечний каталог, у якому записи розміщуються відповідно до розміщення документів на полицях.

Розглянемо докладніше два каталоги, які на даний час найбільш затребувані у бібліотеках: алфавітний та систематичний.

Основна функція алфавітного каталогу – це інформування про те, чи є той чи інший твір друку чи інший документ у фонді бібліотеки, а також які твори певного автора (індивідуального чи колективного) можна знайти у

фонді. Всі картки в алфавітному каталозі розставлені в єдиному алфавіті прізвищ авторів та заголовків книг, незалежно від їх змісту. Такий каталог дозволяє зібрати в одному місці всі твори одного автору. Цінна властивість алфавітного каталогу полягає у простоті роботи з ним у випадку, якщо відомий автор або назва твору. Користуватися алфавітним каталогом зовсім нескладно. Для того, щоб здійснити пошук книги необхідно пам’ятати алфавіт та знати декілька основних правил розташування карток:

- картки розставлені в строгому алфавіті авторів, заголовків книг, причому дотримується алфавіт першої, другої, третьої та інших літер;

- при збігу перших слів, розстановка проводиться у разі алфавіту других, третіх та наступних слів;

- картки на книги авторів-однофамільців розставляються в алфавіті їхніх ініціалів;

- картки на книги авторів з подвійними прізвищами ставляться після книги автора з одним прізвищем;

- книги, які не мають ні індивідуального, ні колективного автора, тобто описані під назвою (книги чотирьох та більше авторів, збірники), розставляються в алфавітної послідовності слів, що входять в їх назву.

Пошук потрібної літери, слова, прізвища полегшують роздільники, які показують порядок розташування описів, що дозволяє наочно продемонструвати правила розміщення.

До систематичного каталогу звертаються, щоб знайти в картотеці необхідні статті на тему. Кожна скринька систематичного каталогу має етикетку з переліком основних розділів каталогу, поміщених у дану скриню.

Від алфавітного каталогу систематичний каталог відрізняється тим, що в кожному з його розділів відображається література з питань однієї галузі.

Впровадження в бібліотеку автоматизованої інформаційної системи спрощує пошук необхідної літератури, оскільки електронний каталог об'єднує пошукові можливості алфавітного, систематичного та предметного каталогів. Він реалізує функції:

- пошуку інформації за разовими запитами;
- друкування бібліографічних даних у вигляді каталожних карток;
- друкування тематичних бібліографічних списків, покажчиків та ін.

На даний час прогрес не стоїть на місці і, мабуть незабаром, паперові каталоги зникнуть з бібліотек зовсім, але досвід, який було ними принесено до бібліотеки залишиться і з робітниками бібліотек, і з користувачами.

1.2 Методи створення каталогу

Питання створення каталогу домашньої бібліотеки на певному етапі постає перед кожним книголюбом. У міру того, як зростає кількість книг у бібліотеці, це питання стає все більш актуальним. Загалом, каталог – це засіб контролю за своєю домашньою бібліотекою та підтримки її в порядку. Розглянемо, які на даний час існують методи для створення каталогу.

1.2.1 Паперовий каталог

Один із традиційних та найпростіших способів створення каталогу домашньої бібліотеки – це ведення його у паперовій формі. Такий підхід не потребує жодних цифрових засобів і може реалізовуватись за допомогою звичайного зошита, альбому, щоденника або навіть набору карток. У кожному записі фіксується базова інформація про книгу: автор, назва, рік видання, жанр, статус прочитання, іноді коротка анотація чи власні нотатки. Записи можна впорядковувати в алфавітному порядку або за тематичними розділами (окремо зберігати книги з історії, художню літературу чи наукову).

Переваги такого підходу:

- не потребує технічних навичок – каталог може вести будь-хто, навіть без досвіду користування комп’ютером або смартфоном;
- не залежить від електроенергії, інтернету чи батареї пристроїв, працює завжди;

- підходить для прихильників ручної роботи – процес запису може бути медитативним та приємним;

- має естетичну цінність – охайні записи, оформлені від руки, створюють відчуття затишку та унікальності;

- зручний для людей похилого віку або тих, хто не хоче мати справу з технологіями.

Однак, паперовий каталог має й суттєві недоліки:

- повільний пошук інформації: щоб знайти книгу, іноді доводиться переглянути десятки сторінок;

- неможливість швидкого сортування або фільтрації за різними параметрами;

- редагування незручне: видалення чи оновлення інформації часто призводить до закреслень, помилок або необхідності переписувати сторінки;

- фізична вразливість: папір може порватися, зіпсуватися, загубитися або постраждати від води;

- каталог важко масштабувати: велика бібліотека займає багато місця, і паперовий варіант стає незручним у використанні;

- відсутність можливості дистанційного доступу або обміну з іншими користувачами.

Як висновок, паперовий каталог залишається привабливим для ентузіастів ручної праці, шанувальників аналогового стилю та користувачів без цифрових навичок. Проте він суттєво обмежений у функціональності, гнучкості та зручності, особливо коли йдеться про великі бібліотеки або потребу швидко орієнтуватися в колекції.

1.2.2 Створення каталогу в офісних програмах

Другий дуже поширений спосіб – це створити каталог книг в офісних програмах: табличних або текстових редакторах. Прикладами таких програм є Word або Excel. Word – це текстовий редактор, і він не дуже зручний для

створення великих каталогів, тому альтернативою залишається Excel. На даний час створення каталогу в таблиці – це популярний та зручний спосіб. Таблиця може включати багато колонок: автор, назва, жанр, рік, статус прочитання, формат книги та ін. При створенні каталогу за допомогою Excel можна використовувати різноманітні фільтри, сортування, кольорові позначки, формули та навіть посилання на електронні файли.

Перевагами використання Excel є:

- простота створення та редагування;
- швидкий пошук та сортування за будь-якими полями;
- можливість використовувати каталог як на комп'ютері, так і на смартфоні (наприклад, Google таблиці дозволяють синхронізувати доступ між пристроями та користувачами);
- легкість додавання, видаляння чи редагування записів;
- зручний експорт та можливість створення резервних копій;
- є можливість візуального аналізу через діаграми та фільтри.

Але такий спосіб створення каталогу має і недоліки:

- потрібне базове знання роботи з таблицями;
- може бути незручно для візуальних людей;
- немає вбудованих обкладинок, анотацій чи візуального оформлення;
- при великому обсязі даних таблиця може "гальмувати";
- складнощі при мобільному доступі на старих пристроях або в місцях без Інтернету.

В якості висновку можливо зазначити, що це універсальний та практичний спосіб для більшості користувачів. Він підходить тим, хто хоче гнучкості, простоти в обліку та зручного доступу.

1.2.3 Використання комп'ютерних програм та мобільних застосунків

Розробники програмного забезпечення теж у курсі проблеми складання каталогу книг домашньої бібліотеки та спеціально для цих цілей створюють

низку спеціальних програм. Серйозною перевагою цих програм є можливість автоматичного завантаження даних про книгу (включаючи обкладинку) за ISBN або за посиланням на магазин. Серед плюсів – зручна оболонка та пошук.

Перевагами використання таких програм є:

- швидке додавання книг через сканер або пошук бази;
- естетичне оформлення: обкладинки, списки, рейтинги, відгуки;
- зручна синхронізація між пристроями;
- можливість ділитися бібліотекою онлайн;
- деякі програми дозволяють відзначати книги, позичені друзям;
- часто присутні функції нагадувань, списків, тегів та ін.

Як недоліки можливо зазначити:

- залежність від підтримки застосунків розробником (якщо проєкт не підтримується, то можливо втрата даних);
- часто потрібні платні версії для розширених функцій;
- не всі бази містять укромовні книжки;
- деякі програми погано працюють в автономному режимі;
- потрібна реєстрація, доступ до Інтернету та технічна грамотність;
- проблеми з експортом даних під час переходу на іншу платформу.

Використання таких програм для організації каталогів – це відмінне рішення для просунутих користувачів, колекціонерів та любителів мобільних рішень. Особливо добре цей спосіб підходить тим, хто часто купує нові книги та хоче все тримати під рукою.

1.2.4 Використання самописної бази даних

Для тих, хто любить більше контролю, можна створити власну систему обліку – базу даних, таблицю в Notion або навіть веб-інтерфейс із власними фільтрами та логікою. Це вимагає часу, але дозволяє налаштувати все під себе. СУБД Access [1] – це спеціалізована програма для створення бази

даних, яка має найширші можливості. Але слід зазначити, що для користування Access потрібні навички, і для багатьох людей ця програма виявляється занадто складною, а розбиратися в ній лише заради створення каталогів домашньої бібліотеки готові не всі.

Перевагами використання способу, орієнтованого на написання БД є:

- повна кастомізація структури та полів;
- можливість підключення автоматичних розрахунків, тегів, посилань та міток;
- добре підходить для сучасних користувачів;
- можна вести облік різних форматів книг, навіть створювати візуальні дошки;
- у Notion можна створити галерею з обкладинками та фільтрами на теми.

Але запропонований метод має низку недоліків:

- потрібен час на розробку та налаштування;
- потрібне розуміння принципів баз даних чи хоча б логіки систем;
- можливо надмірно складно для невеликої бібліотеки;
- не всі платформи є зручними для мобільного використання;
- часто немає стандартної підтримки імпорту даних із інших джерел.

В якості висновку можливо зазначити, що цей спосіб ідеальний для ентузіастів, які хочуть максимального контролю над своїм каталогом та не бояться технічного налаштування, але для звичайного користувача він може здатися громіздким.

1.3 Аналіз існуючих аналогів для створення каталогу бібліотеки

У сучасному світі, де кількість доступної літератури стрімко зростає, зберігати порядок у власній бібліотеці стає все складніше. Для цього створено безліч цифрових інструментів – від простих мобільних застосунків до потужних онлайн-сервісів. Кожен з них має свої особливості, сильні

сторони та недоліки, залежно від потреб користувача. Нижче наведено аналіз найбільш популярних міжнародних аналогів для створення каталогів домашньої бібліотеки, який допоможе оцінити їх функціональність, зручність та відповідність різним сценаріям використання.

1.3.1 Соціальна мережа Goodreads

Соціальна мережа Goodreads (рисунок 1.1) є одним із найвідоміших міжнародних сервісів для читання та каталогізації книг, який належить Amazon. Дане рішення дозволяє додавати книги до своєї бібліотеки, ставити оцінки, писати рецензії, вести списки та брати участь у книжкових клубах.

Рисунок 1.1 – Соціальна мережа Goodreads

Goodreads – це цілісна екосистема для взаємодії книголюбів, письменників та видавців. У Goodreads існує соціальна ієрархія: користувачі,

які розмістили у своєму профілі понад 50 книг, мають право стати бібліотекарями. Власники цього статусу можуть редагувати дані про книги та авторів, складати списки, поповнювати каталог.

Плюсами використання Goodreads є наявність величезної бази даних книг, яка охоплює не лише популярні, але й рідкісні та зарубіжні видання. Важливою перевагою є соціальна складова платформи: користувачі можуть додавати друзів, брати участь в обговореннях, отримувати персональні рекомендації. Goodreads дозволяє зручно вести списки книг за статусами: «прочитано», «читаю», «хочу прочитати», що допомагає організувати читання. Сервіс також підтримує синхронізацію з Kindle та іншими платформами Amazon, що полегшує доступ до куплених або завантажених книг. Додатково користувачам доступні як мобільний додаток, так і вебверсія.

Водночас, Goodreads має і свої недоліки. Інтерфейс ресурсу виглядає застарілим і часто перевантажений елементами, що ускладнює навігацію. Функціонал сортування та фільтрації в особистій бібліотеці обмежений, що створює труднощі при роботі з великими колекціями. Немає повноцінної офлайн-версії, що може бути незручно при відсутності доступу до Інтернету. Основний контент сайту нажаль англомовний: переважна більшість книг, рецензій та рекомендацій представлена англійською, що знижує зручність для україномовних користувачів. Крім того, сервіс тісно пов’язаний з Amazon, а це обмежує персоналізацію та створює відчуття комерційної залежності.

1.3.2 Застосунок для каталогізації книг LibraryThing

LibraryThing – це соціальна каталогізація у вигляді вебзастосунку (рисунок 1.2) для зберігання та обміну книгами, каталогами та різними типами метаданих. Застосунок запущено у 2005 року і станом на березень 2024 року вона налічувала 2 842 000 користувачів, понад 148 мільйонів книг

було каталогізовано.

LibraryThing – це бібліотечний каталог для колекціонерів та бібліофілів, який дозволяє додавати книги вручну або за допомогою ISBN, сортувати за багатьма критеріями, вести статистику.

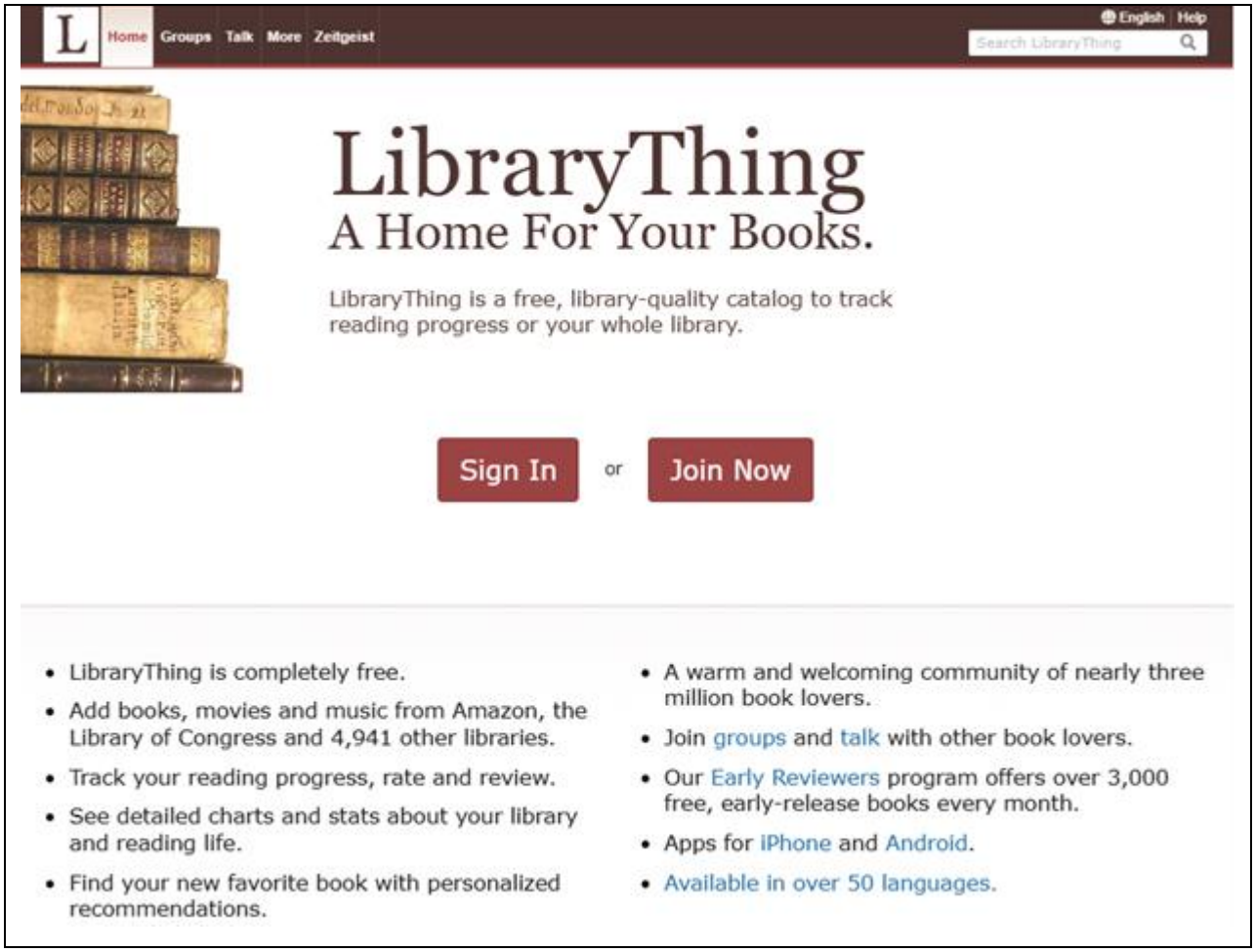


Рисунок 1.2 – LibraryThing

Плюсами використання LibraryThing є те, що цей сервіс пропонує потужну систему тегів, метаданих та категорій, що дозволяє гнучко структурувати бібліотеку. Він підтримує експорт та імпорт даних, що зручно для перенесення чи резервного копіювання інформації. Додатковою перевагою є розширена статистика: можна аналізувати колекцію за жанрами, авторами, датами тощо. LibraryThing також підтримує бібліографічні формати, зокрема MARC-записи, що робить його придатним навіть для професійних бібліотек. Завдяки високій точності каталогізації цей інструмент

цінується серед колекціонерів та бібліотекарів.

Однак є й мінуси. Інтерфейс LibraryThing доволі складний та візуально застарілий, що може ускладнити його освоєння для новачків. Мобільний застосунок має обмежену функціональність та є незручним у використанні. Налаштування системи вимагає часу та терпіння, особливо при великій кількості книг. Крім того, основна мова інтерфейсу й бази даних англійська, а україномовних книг у базі відносно небагато, що може бути проблемою для користувачів, які читають переважно українською.

1.3.3 Хмарне рішення для обліку книг Libib

Libib – це хмарне рішення для обліку книг, фільмів, музики та ігор, яке підходить як для особистого користування, так і для невеликих публічних колекцій (рисунок 1.3). Сервіс пропонує як безкоштовну, так і платну версію, залежно від потреб користувача. До переваг також належить наявність мобільного застосунку та зручного вебінтерфейсу, що дозволяє працювати з бібліотекою з будь-якого пристрою.

Серед основних плюсів Libib варто відзначити зручний та мінімалістичний інтерфейс, який не перевантажений зайвими елементами. Програма підтримує сканування штрих-кодів книг прямо з мобільного застосунку, що значно пришвидшує процес додавання нових позицій. Користувач може створювати власні категорії, мітки та сортувати матеріали за власними критеріями. Libib також дозволяє імпортувати та експортувати бібліотеку у форматі CSV, що є корисним для резервного копіювання або перенесення даних. Окремо варто згадати функцію розмежування на особисту та публічну бібліотеки: можна, наприклад, вести внутрішній каталог і паралельно мати відкриту версію для друзів чи колег.

Проте сервіс має й недоліки. Деякі розширені можливості, такі як поглиблений аналіз або статистика використання, доступні лише у платній версії. Також спостерігається обмежена підтримка україномовних видань, що

може ускладнити пошук та додавання деяких книг. Крім того, у безкоштовній версії можна створити не більше п’яти окремих бібліотек, що може стати обмеженням для користувачів з великою або тематично розділеною колекцією.

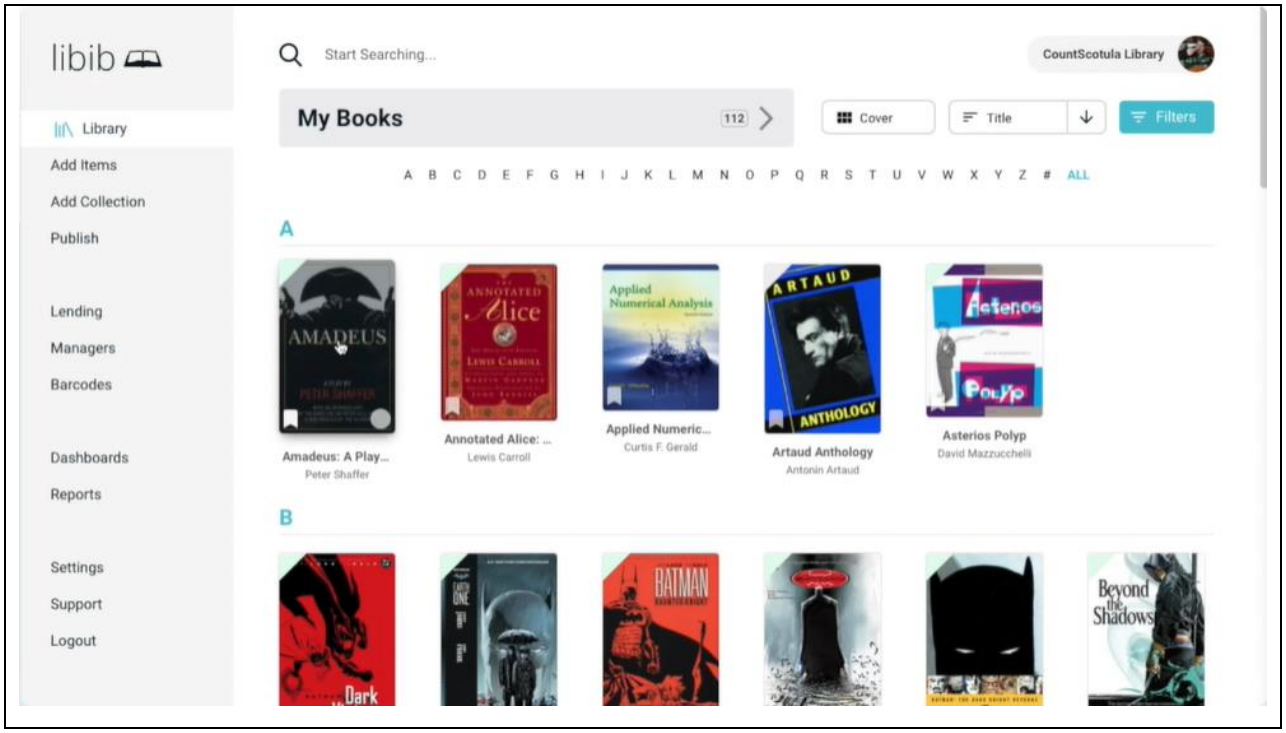


Рисунок 1.3 – Хмарне рішення для обліку книг Libib

1.3.4 Мобільний застосунок BookBuddy

BookBuddy (рисунок 1.4) – зручний мобільний застосунок для обліку домашньої бібліотеки, який орієнтовано на користувачів пристроїв Apple. Його перевага – це простота у використанні та швидкість роботи, що робить його одним із найпопулярніших рішень серед власників iPhone та iPad.

Серед плюсів BookBuddy варто відзначити дуже зручний та інтуїтивно зрозумілий мобільний інтерфейс, який не перевантажений зайвими елементами. Застосунок дозволяє швидко сканувати ISBN-коди книг та додавати їх у бібліотеку практично миттєво. Користувачам доступні різноманітні фільтри, теги, статуси читання, а також можливість залишати нотатки до кожної книзі. Крім того, є функція експорту бібліотеки у форматі

CSV, що зручно для створення резервних копій або роботи з каталогом на комп’ютері. Інтерфейс BookBuddy приємний на вигляд і добре адаптований під мобільні пристрої Apple.

Рисунок 1.4 – Мобільний застосунок BookBuddy

Втім, є й мінуси. Застосунок доступний виключно для пристроїв на базі iOS, тому користувачі Android не зможуть ним скористатися. Деякі корисні функції, зокрема розширене сортування, резервне копіювання в хмару та безрекламний режим, доступні лише у платній версії BookBuddy Pro. Інтерфейс програми повністю англомовний, а база даних книг має англоцентричний ухил, що може створити незручності для користувачів, які читають українською чи іншими мовами. Також варто враховувати, що застосунок не має вебверсії та не підтримує синхронізацію з іншими платформами, що обмежує можливості багатоплатформного доступу до бібліотеки.

1.3.5 Застосунок Delicious Library 3

Delicious Library 3 – це настільний додаток для macOS, який дозволяє створювати детальний цифровий каталог фізичних колекцій: книг, фільмів, відеоігор, музики, ґаджетів та іншого. Програма вирізняється стильним

оформленням та орієнтована на візуально привабливий облік великих приватних колекцій.

Рисунок 1.5 – Настільний застосунок Delicious Library 3

Серед основних переваг Delicious Library 3 варто виділити багаті візуальні можливості: у застосунку реалізовано 3D-подання полиць, яскраві обкладинки та категорії, що створює ефект справжньої домашньої бібліотеки. Інтелектуальний пошук та потужні засоби сортування дозволяють легко орієнтуватися у великій кількості предметів. Є функція сканування штрих-кодів через камеру комп’ютера або сумісного пристрою, що значно пришвидшує додавання нових об’єктів. Також доступна підтримка міток, нотаток та відміток про стан книги або іншого носія. Програма добре підходить для ведення великих та різноманітних за форматом колекцій.

Проте є й недоліки. Delicious Library 3 працює виключно на macOS, тому користувачі інших платформ (Windows, Linux) скористатися нею не зможуть. Програма є платною, і безкоштовної версії нажаль не передбачено. Ще один недолік – обмежена підтримка неангломовних джерел: база даних

найкраще працює з англомовним контентом, що може бути незручно для користувачів, які збирають книги або фільми іншими мовами. Крім того, застосунок не підтримує синхронізацію з популярними онлайн-сервісами, такими як Goodreads чи Libib, що обмежує інтеграцію з іншими платформами та обліковими записами.

1.3.6 Мобільний застосунок MyLibrary

MyLibrary – це мінімалістичний мобільний застосунок, доступний для iOS та Android, який призначений для ведення особистого каталогу книг. Користувачі можуть додавати книги вручну або за допомогою сканера штрих-коду, зберігати інформацію локально на пристрої, ділитися нею з іншими та експортувати у зручному форматі.

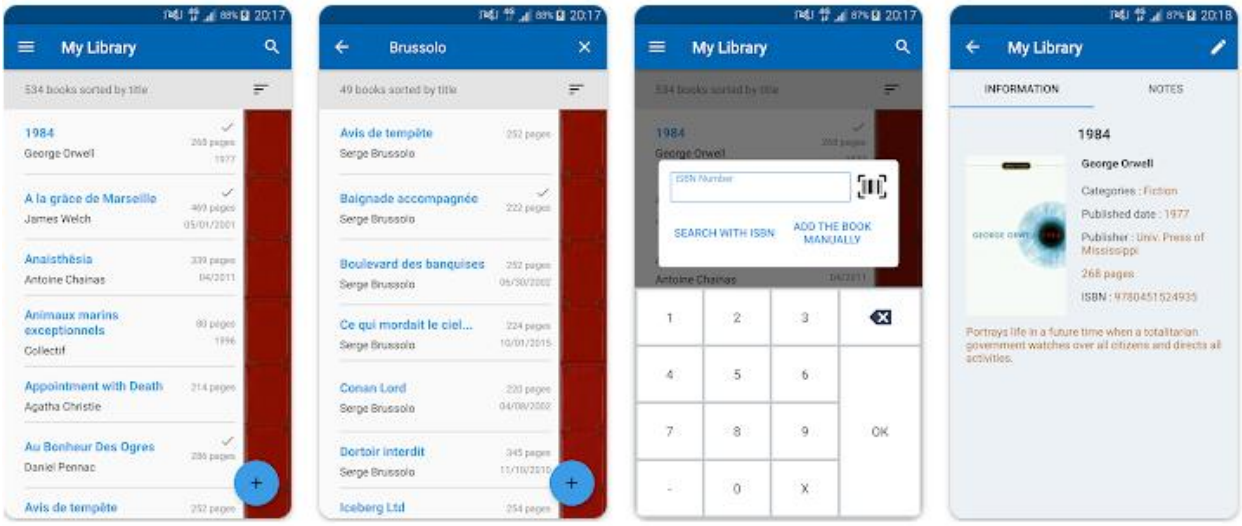


Рисунок 1.6 – Мобільний застосунок MyLibrary

Серед основних плюсів простота та легкість у використанні. Інтерфейс застосунку інтуїтивно зрозумілий, що робить його доступним навіть для тих, хто вперше користується подібними сервісами. Візуалізація бібліотеки виглядає привабливо: книги представлені у вигляді обкладинок, з можливістю сортування за жанрами. MyLibrary підтримує експорт бібліотеки

у формати Excel або CSV, що дозволяє створити резервну копію або обробляти дані на комп’ютері. Також є корисна функція створення списку бажаних книг та поділу на статуси: «прочитано», «читаю», «не прочитано». Важливо зазначити, що застосунок не вимагає реєстрації та не містить нав’язливої реклами, що додає комфорту при щоденному використанні.

Проте застосунок має й мінуси. Він не завжди успішно знаходить книжки українською мовою, що може створити труднощі для користувачів з відповідною колекцією. Також відсутня функція синхронізації між пристроями: дані зберігаються лише на одному телефоні, і для перенесення потрібне ручне копіювання. Додаткові функції обліку та аналітики практично відсутні: реалізовано лише базовий функціонал. Крім того, резервне копіювання також здійснюється лише вручну, без інтеграції з хмарними сервісами, що може бути незручно у випадку втрати пристрою.

1.4 Висновки по розділу

На основі проведеного аналізу можна зробити висновок, що жоден із існуючих застосунків не задовольняє одночасно всім ключовим потребам сучасного користувача, особливо з урахуванням специфіки україномовного ринку. Кожне з рішень має свої сильні сторони, але також має і суттєві обмеження, які можуть стати причиною для розробки власного, більш збалансованого мобільного застосунку.

Основні слабкі місця існуючих рішень:

- Goodreads – сильна соціальна складова, але слабка каталогізація, застарілий інтерфейс та англомовна орієнтація;
- LibraryThing – професійна точність, але складність інтерфейсу, низька мобільність та обмеженість з українською мовою;
- Libib – зручність та мобільність, але частина функцій лише у платній версії та недостатня підтримка локальних книг;
- BookBuddy – зручний, але тільки для iOS, без синхронізації та без

наявності вебверсії;

- Delicious Library 3 – візуально привабливий, але обмежений: лише до macOS, платний, без інтеграції з іншими сервісами;
- MyLibrary – простий та доступний, але з мінімальним функціоналом, ручним резервним копіюванням та відсутністю синхронізації.

Проведений аналіз показав, що переважаючими недоліками сервісів є:

- відсутність підтримки української мови та локалізованих баз даних;
- обмеження за платформами (тільки iOS або тільки macOS);
- застарілий або перевантажений інтерфейс;
- відсутність або платність ключових функцій (синхронізація, статистика, резервні копії);
- слабка персоналізація та інструменти аналітики.

На основі виявлених недоліків є доцільним створення власного мобільного застосунку, який би поєднував:

- простоту інтерфейсу MyLibrary та Libib;
- точність каталогізації LibraryThing;
- базову соціальну функціональність Goodreads (наприклад, обмін рекомендаціями);
- повну українську локалізацію та підтримку книжок українською;
- безкоштовний доступ до базових функцій із можливістю розширення за бажанням.

Такий застосунок зміг би закрити нішу, яка зараз недостатньо охоплена: мобільний, простий, локалізований сервіс з акцентом на персональну бібліотеку та зручне управління нею.

2 ОБГРУНТУВАННЯ ВИБОРУ ІНСТРУМЕНТАРІЮ ДЛЯ РОЗРОБКИ МОБІЛЬНОГО ЗАСТОСУНКУ

2.1 Постановка задачі проєктування

Під час виконання кваліфікаційної роботи необхідно створити програмне забезпечення, яке орієнтоване на створення каталогу домашньої бібліотеки у вигляді мобільного застосунку для операційної системи Android та включати систему надання користувачеві рецензій, а також мати функціонал для взаємодії з каталогом книг [2]. Під взаємодією мається на увазі можливість додавати книги до віртуальної бібліотеки, змінювати інформацію про вже існуючі в ній книги, проводити розрахунок статистики для прочитаних книг. Джерелом даних для пошуку інформації є API «Google Books», джерелом даних для створення статистики є існуючі дані, які були внесені користувачем. Окремою задачею є реалізація графічного інтерфейсу, з яким взаємодіятиме користувач. Він повинен складатися з окремих вікон для роботи з окремими розділами програми [3].

2.2 Перелік вимог до програмного забезпечення

У застосунку необхідно реалізувати наступні ролі користувачів:

- неавторизований користувач;
- авторизований користувач (має обліковий запис у застосунку).

Неавторизований користувач має доступ лише до двох розділів: реєстрації та авторизації.

Авторизований користувач має права до:

- розділу «Список книг» – перегляд, редагування, створення запиту;
- розділу «Список рецензій» – перегляд, оцінка;
- розділу «Персональна статистика» – перегляд, редагування;
- розділу «Сповіщення» – перегляд;
- розділів соціального функціоналу (профіль користувача, стрічка

передплат) – перегляд, редагування.

У застосунку повинно бути реалізовано декілька підсистем. Підсистема реєстрації дій користувача дозволяє реєструвати факти взаємодії з програмою. Підсистема зберігання даних зберігає дані у створеній базі даних. Підсистема розрахунку статистики розраховує статистику прочитаних книг та сумарного часу читання для користувача за певний проміжок часу. Підсистема повідомлення формує та надсилає повідомлення користувачу про активність інших користувачів. Підсистема редагування інформації дозволяє налаштувати графік читання книги, змінити особисту інформацію у профілі користувача. Підсистема пошуку інформації здійснює пошук книг та користувачів.

Мобільний застосунок, що розробляється, повинен коректно працювати на смартфонах, що мають наступні характеристики:

- мінімальна версія операційної системи Android, що підтримується: Android 5.0;
- максимальна версія операційної системи Android, що підтримується: Android 11.

Підтримка альбомної орієнтації або функції розділення екрана у застосунку, що розробляється, не передбачена.

Застосунок повинен мати інтерфейс, що відповідає наступним вимогам:

- взаємодія програми та користувача повинна здійснюватися українською або англійською мовами;
- необхідно реалізувати відображення на екрані лише тих можливостей, які доступні користувачеві відповідно до наданих йому прав доступу;
- всі екранні форми інтерфейсу користувача повинні містити однакове розташування основних елементів управління та навігації;
- повідомлення про помилки ПЗ або помилкові дії користувача повинні супроводжуватися інформацією про причини помилки та з підказкою про подальші дії, необхідні для усунення проблеми;

- виконання задачі, ініційованої користувачем, у разі відсутності очевидного результату має супроводжуватись відповідною зворотною реакцією з боку системи, що сигналізує про результат виконання.

Дизайн інтерфейсу користувача повинен адаптуватися під основні дозволи екранів мобільних пристроїв. Максимальна ширина екрану пристрою – 1440 px, мінімальна – 1080 px. Інтерфейс програми, включаючи графіки та діаграми, повинен адаптуватися під роздільну здатність екрана. Елементи інтерфейсу (пункти меню, кнопки, поля введення у формах, списки, що розкриваються) повинні адаптуватися за розміром під екран пристрою. Навігаційні елементи інтерфейсу ПЗ повинні забезпечувати однозначне розуміння користувачем їхнього сенсу, надавати навігацію по всіх доступних користувачеві розділах програми та відображати відповідну інформацію.

2.3 Методологія розробки

Для реалізації проєкту було обрано концепцію «швидкої розробки додатків» (RAD), що стала ефективною альтернативою традиційним, лінійним підходам (зокрема «водоспадній моделі»), які часто не враховують змінність вимог на пізніх етапах розробки. У випадку «водоспадної моделі» внесення змін після завершення етапу проєктування є складним та ресурсоємним, що може призвести до створення продукту, який не відповідає актуальним потребам користувачів.

RAD орієнтується на гнучкість, швидкість та безперервний зворотний зв’язок із кінцевим користувачем. Розробка відбувається через низку швидких ітерацій (інкрементів), під час яких створюються функціональні прототипи застосунку, що дозволяє ще на ранніх етапах оцінити зручність інтерфейсу, виявити помилки, отримати відгуки від потенційних користувачів та адаптувати функціонал згідно з актуальними потребами ринку.

Ключові принципи методології RAD, які були застосовані під час

розробки мобільного застосунку:

- висока швидкість розробки ПЗ: завдяки використанню інструментів візуального програмування, шаблонів інтерфейсу та компонентів, що повторно використовуються, вдалося значно скоротити час реалізації базового функціоналу;

- залучення користувача на всіх етапах: постійна комунікація з цільовою аудиторією забезпечила створення інтерфейсу, що відповідає очікуванням та є інтуїтивно зрозумілим;

- гнучкість та адаптивність: зміни у вимогах, що виникали в ході розробки, швидко враховувалися без необхідності масштабного перепроєктування;

- простота масштабування та модифікації: архітектура застосунку була побудована таким чином, щоб у майбутньому легко додавати нові функції без ризику порушення стабільності роботи системи.

Таким чином, використання методології RAD дало змогу не лише пришвидшити процес розробки, а й створити якісний, зручний для користувача продукт, готовий до подальшого розвитку та адаптації під ринкові умови.

2.4 Обґрунтування вибору мови програмування та засобів розробки

При розробці мобільного застосунку була використана мова програмування Kotlin, а як середовище розробки Android Studio, тому що ці інструменти є офіційно рекомендованими Google для створення Android-застосунків. Kotlin забезпечує сучасний синтаксис, високу безпеку від помилок (null safety) та хорошу інтеграцію з існуючим Java-кодом. Android Studio, у свою чергу, пропонує потужні інструменти для налагодження, тестування та проєктування інтерфейсу, що значно прискорює процес розробки та підвищує якість кінцевого продукту.

2.4.1 Мова програмування Kotlin

Kotlin [4] являє собою статично типізовану мову програмування, будучи однією з тих, що швидко розвиваються. Він підтримує як об'єктно-орієнтоване, так і процедурне програмування. Офіційний реліз мови відбувся у 2011 році, розробником виступає компанія JetBrains. Найпопулярнішим напрямком, у якому застосовується Kotlin, є розробка програм під операційну систему Android. У 2017 році мова отримала офіційну підтримку на конференції Google I/O. Після цього інструменти для роботи з Kotlin були включені до функціоналу середовища розробки Android Studio.

Основними можливостями та перевагами Kotlin є:

- відкритий вихідний код;
- доступна автоматична конвертація Java-коду в Kotlin;
- програми можуть використовувати усі існуючі Java-бібліотеки;
- мову можна використовувати з багатьма існуючими системами збирання програм, у тому числі Maven та Gradle;
- код компілюється в байткод JVM.

З впровадженням мови Kotlin було вирішено низку проблем, пов'язаних як із безпекою, так і зі складністю синтаксису, що були у Java. На відміну від Java, в Kotlin є класи даних, що призначені спеціально для зберігання даних, також тут використовується автоматичне приведення типів та вирішена проблема з виникненням виключення NullPointerException. Крім того, Kotlin надає велику кількість корисних можливостей для функціонального програмування, наприклад, лямбда-вирази та функції вищого порядку.

2.4.2 Система автоматизованого збирання Gradle

Gradle – це сучасна система автоматизованого збирання проєктів, яка широко використовується в середовищі Android-розробки, зокрема в Android Studio. Вона дозволяє ефективно керувати процесом компіляції, збірки, тестування, запуску та публікації застосунків. Gradle підтримує мови Java,

Kotlin, Groovy та інші, а також легко масштабується: від невеликих проєктів до великих, модульних систем. Основними модулями, що призначені для розробки та розгортання є Scala, Groovy та Java застосунків [5].

З Gradle можна автоматизувати компіляцію, тестування та розгортання програмного забезпечення чи будь-яких інших типів проєктів. На даний час Gradle вже використовується великими проєктами з відкритим вихідним кодом, такими як Spring (універсальний фреймворк з відкритим вихідним кодом Java-платформи), Hibernate (бібліотека для мови програмування Java) та Grails (фреймворк для створення вебзастосунків) [5].

Основними перевагами Gradle є:

- гнучкість: можна легко налаштувати правила складання, залежності, скрипти для автоматизації будь-яких етапів розробки;
- підтримка плагінів: Android-плагін для Gradle забезпечує повну інтеграцію з Android SDK і спрощує створення APK/AAB-файлів;
- можливість роботи з залежностями: автоматичне підключення зовнішніх бібліотек із репозиторіїв (наприклад, Maven Central або JCenter);
- інкрементальна збірка: збільшує швидкість збирання, виконуючи лише ті завдання, які дійсно потребують оновлення.

Gradle є важливою частиною екосистеми Android і дозволяє значно оптимізувати життєвий цикл проєкту, підвищуючи ефективність командної роботи та якість коду.

2.4.3 Середовище розробки Android Studio

Android Studio [6] – це офіційне інтегроване середовище розробки (IDE) від компанії Google для створення застосунків під операційну систему Android. Побудоване на базі IntelliJ IDEA, Android Studio поєднує в собі потужний набір інструментів, які дозволяють розробникам ефективно створювати, тестувати та оптимізувати мобільні додатки.

Однією з головних переваг Android Studio є глибока інтеграція з Android SDK, що забезпечує швидкий доступ до емуляторів,

відлагоджувачів, засобів профілювання продуктивності та інших важливих компонентів. Розробники можуть використовувати мови програмування Kotlin, Java або C++, а також розширювати проєкти за допомогою зовнішніх бібліотек і фреймворків.

Серед ключових можливостей Android Studio:

- візуальний редактор інтерфейсу (Layout Editor), який дозволяє створювати інтерфейси перетягуванням елементів;
- інструменти налагодження та аналізу продуктивності (Logcat, Profiler, Memory Monitor тощо);
- потужна система складання на основі Gradle, яка автоматизує процеси збирання, тестування та розгортання застосунків;
- інтегрований емулятор Android, що дає змогу запускати застосунок без фізичного пристрою;
- підтримка версійності та тестування: можна створювати різні версії програми для різних пристроїв та умов.

Android Studio також підтримує автоматичну перевірку помилок, інтелектуальні підказки коду (IntelliSense), систему контролю версій (наприклад, Git), що робить роботу над проєктами зручною як для індивідуального розробника, так і для команд.

Середовище постійно оновлюється, підтримуючи найновіші версії Android API, нові технології та сучасні підходи до розробки, такі як Jetpack Compose. Завдяки Android Studio розробка застосунків стає доступною, структурованою та ефективною незалежно від рівня досвіду програміста.

3 ПРОЄКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Структура програми

Застосунок, що розробляється, складається із декількох компонентів, кожен з яких оголошується у файлі маніфесту. ОС Android використовує цей файл, щоб інтегрувати програму в інтерфейс користувача. Крім того, мобільний пристрій має обмеження у вільних для використання ресурсах, тому ОС може переривати деякі застосунки, щоб звільнити місце для нових.

З урахуванням цих обставин було обрано шаблон проектування архітектури MVVM [7]. Таким чином, шар «Подання» не буде зберігати дані та стан об'єктів програми, а служитиме лише для відображення інформації. А всі дані після перезавантаження «Подання» знову завантажуватимуться з «Моделі». Такий підхід називається «Поділ відповідальності». Класи інтерфейсу користувача містять лише логіку обробки дій користувача, а потім повідомляють «Модель» про ці дії. «Модель», своєю чергою, пов'язана з класами «Подання» і може обробляти дані незалежно від того, який клас до неї звернувся [3].

Це рішення спрощує навантаження на ресурси пристрою та взаємодію користувача з застосунком, оскільки класи «Уявлення» мають свій «життєвий цикл», який залежить від багатьох дій користувача та системи. У таких умовах усі об'єкти класів Activity та Fragment піддаються змінам при відкритті, згортанні, закритті програми, при перевороті екрана та при взаємодії різних компонентів програми один з одним.

На рисунку 2.1 представлено схему життєвого циклу класу Activity з вказанням відповідних дій:

- зв'язок 1 – інша активність вийшла на передній план;
- зв'язок 2 – активність більше не видно;
- зв'язок 3 – активність припинила роботу або була знищена системою;
- зв'язок 4 – застосунку із високим пріоритетом потрібна пам'ять;

- зв'язок 5 – користувач переходить на активність;
- зв'язок 6 – користувач повернувся до активності;
- зв'язок 7 – користувач переходить на активність.

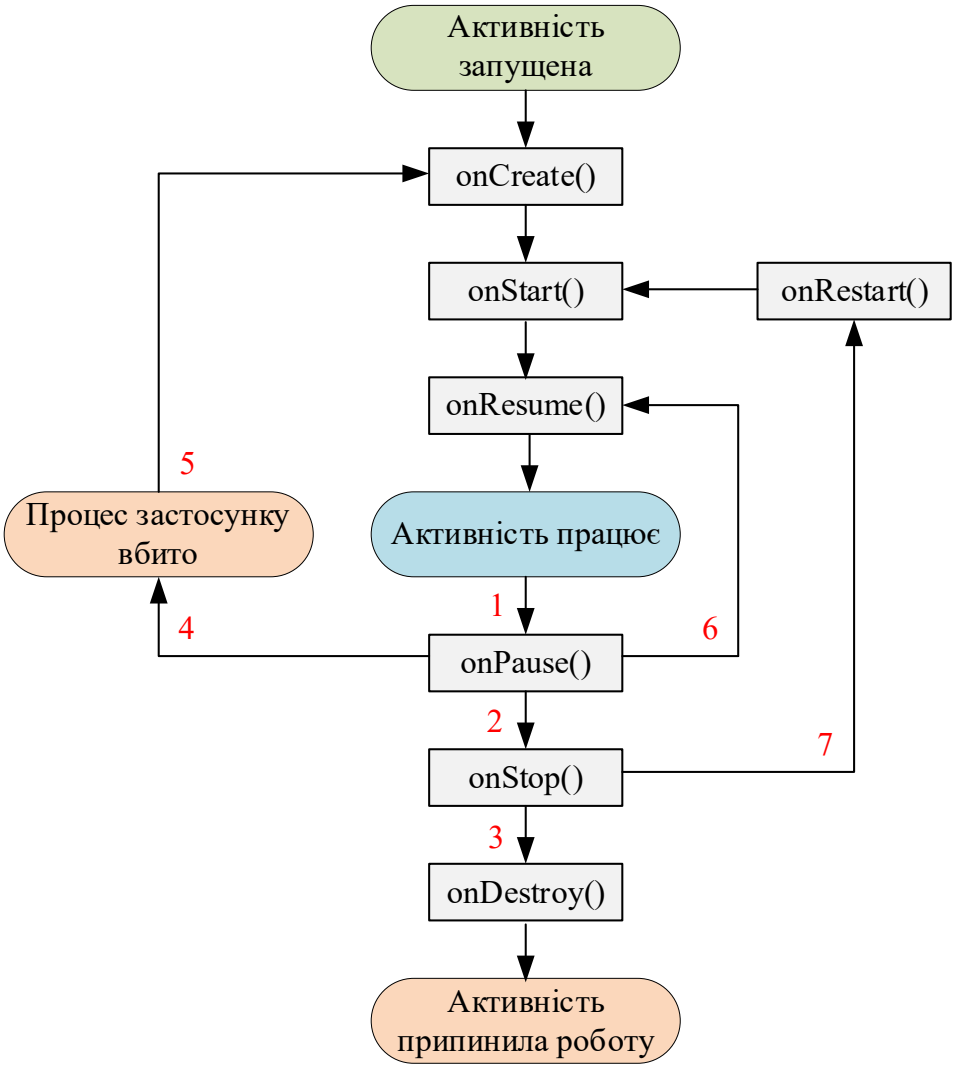


Рисунок 3.1 – Схема життєвого циклу класу Activity

Вся програма поділена на 4 логічні групи розділів застосунку [8]: список книг, список рецензій, персональна статистика, повідомлення. Навігація між групами здійснюється за допомогою нижньої навігаційної панелі (BottomNavigationView), що реалізує перемикання між екранами. Схема екранів системи та їх взаємодії представлена рисунку 2.2.

Кожен екран мобільного застосунку реалізується окремим класом, що успадковується від класу Fragment.

Модуль «Список книг» містить екрани списку книг (BookListFragment) та перегляду детальної інформації про книгу (BookDetailsFragment). Модуль «Список рецензій» містить екрани списку рецензій (ReviewFragment), перегляду певної рецензії (ReviewDetailsFragment) та створення нової рецензії (CreateReviewFragment). Модуль «Персональна статистика» містить екрани перегляду загальної статистики (TimelineFragment) та перегляду статистики за книгою (CurrentTimelineFragment). Модуль сповіщення містить екран з відображенням сповіщень (NotificationsFragment).

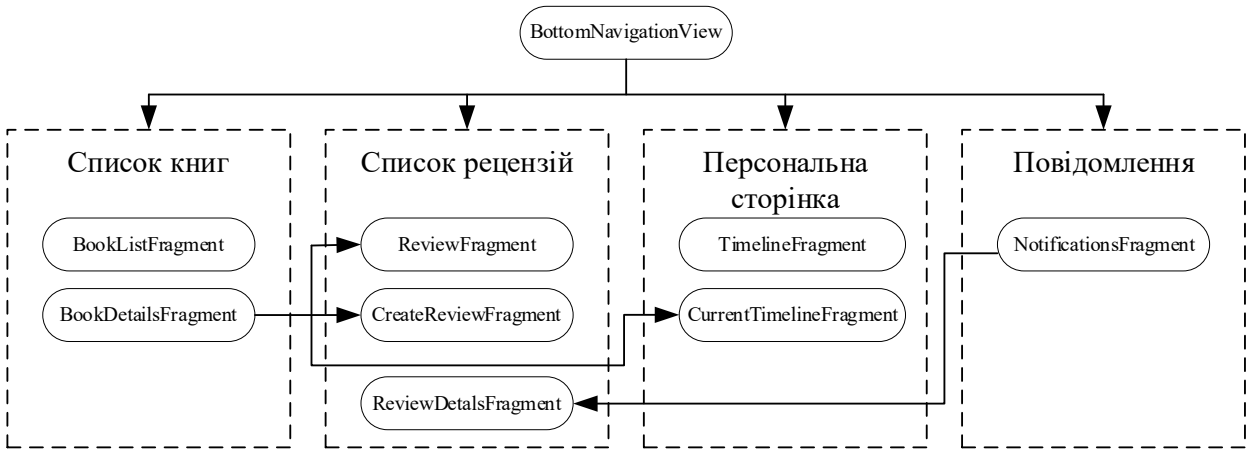


Рисунок 3.2 – Екрани системи та їх взаємодія

Крім основного класу, кожен екран може бути розбитий на кілька допоміжних класів, які реалізують деяку частину інтерфейсу екрану та логіку, пов'язану з цією частиною.

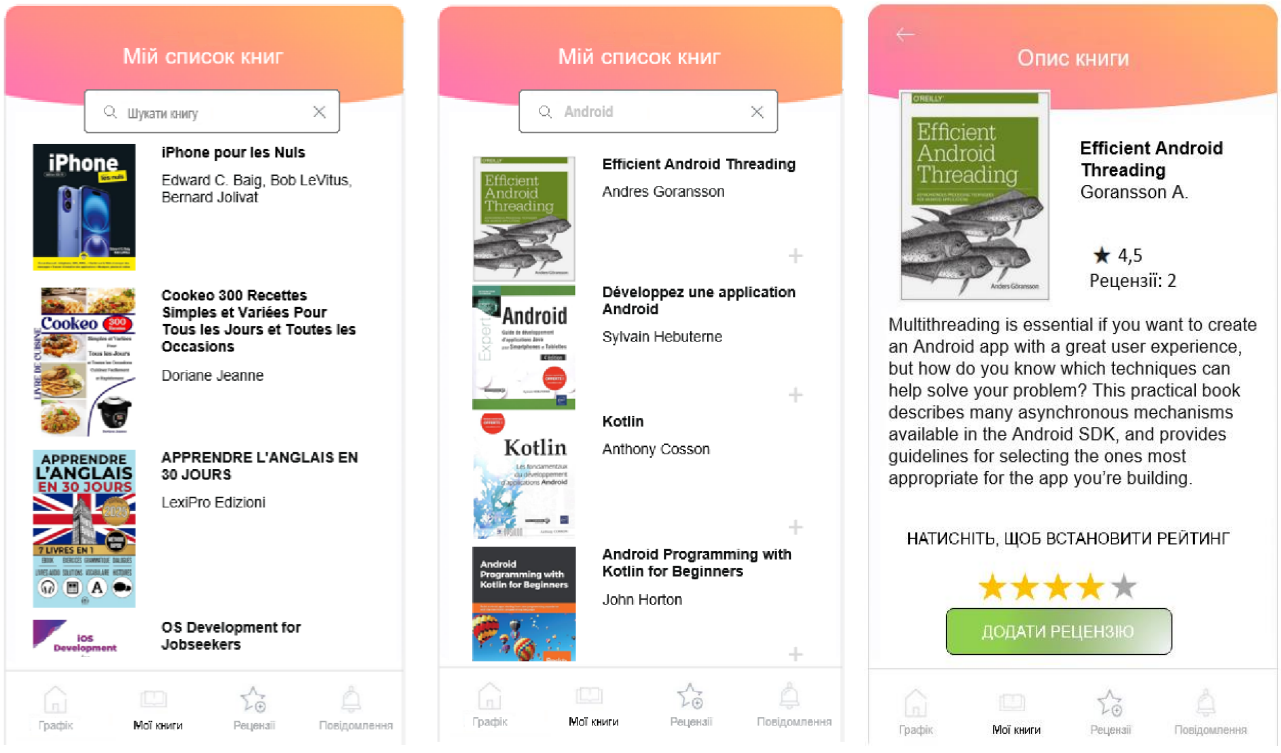
3.2 Проєктування інтерфейсу користувача

Для виконання задачі, яка була сформульована, необхідно розробити графічний інтерфейс користувача мобільного застосунку. В якості інтерфейсу виступають файли, що містять XML-розмітку та класи «Подання». Інтерфейс повинен надавати користувачеві можливість переглядати розділи мобільного застосунку, що містять список книг, список рецензій, персональну статистику та повідомлення про дії інших

користувачів. Для швидкої навігації використовується меню, кожен пункт якого відповідає за один із розділів. Зовнішній вигляд меню представлений рисунку 2.3.

Рисунок 2.3 – Зовнішній вигляд меню користувача

Розділ «Мої книги» містить форму виведення даних, які будуть надіслані у запиті на сервер. Наприклад, якщо перейти до розділу «Мій список книг», то користувач побачить перелік книг, які є у застосунку (рисунок 2.3 а). Якщо користувач забажає здійснити так зване сортування за окремою тематикою, то результатом буде вибірка стосовно відповідного запиту. Наприклад, користувач вводить у поле пошуку «Android», в якості відповіді з’являється список, який містить це слово (рисунок 2.4 б).

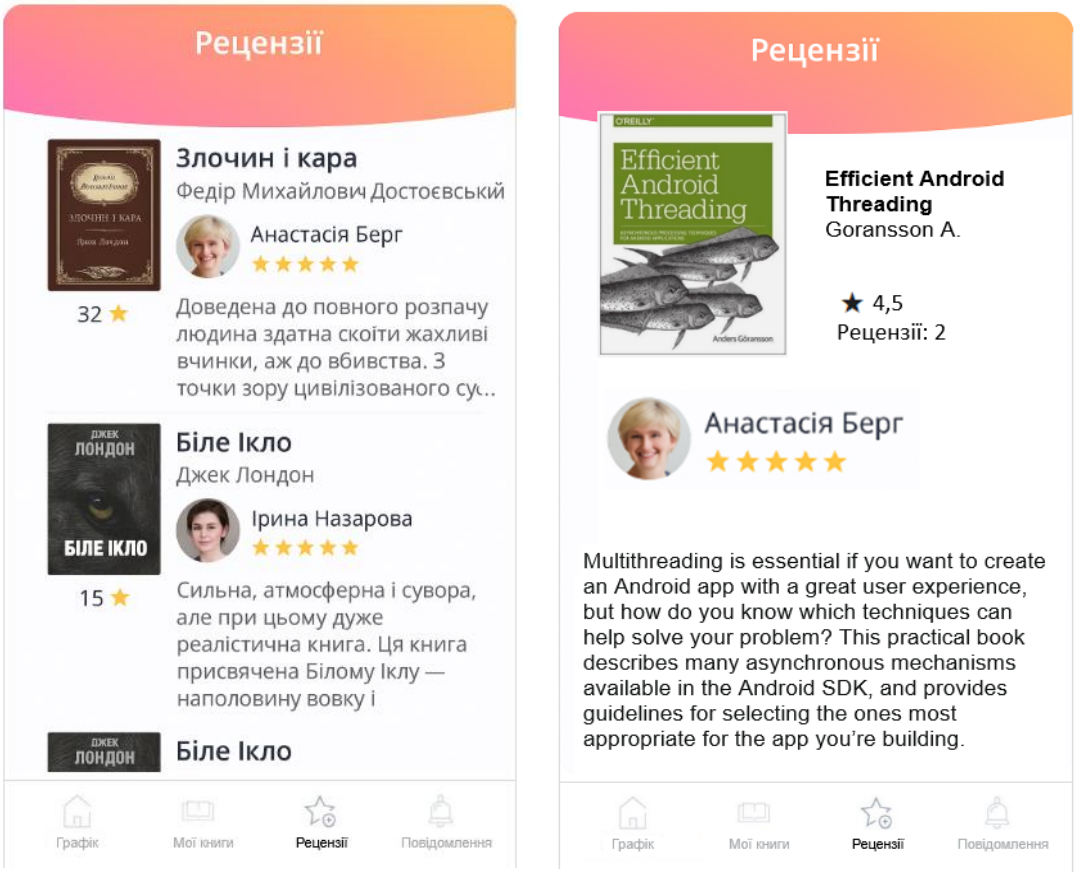


а) б) в)

Рисунок 2.4 – Зовнішній вигляд розділу «Мої книги»

У розділі також відображається інформація, що надійшла з відповіддю на останній запит, отриманий від сервера, або повідомлення про помилку. Для перегляду всіх даних про книгу створено окрему сторінку, на яку можна перейти, вибравши елемент списку. Приклад такої сторінки представлено рисунку 2.4 в.

Розділ «Рецензії» представлений списком рецензій із зазначенням авторів, є можливість перейти до перегляду детальної інформації та оцінки рецензії, а також інформації про автора. Приклад сторінок розділу представлено рисунку 2.5.



а) б)

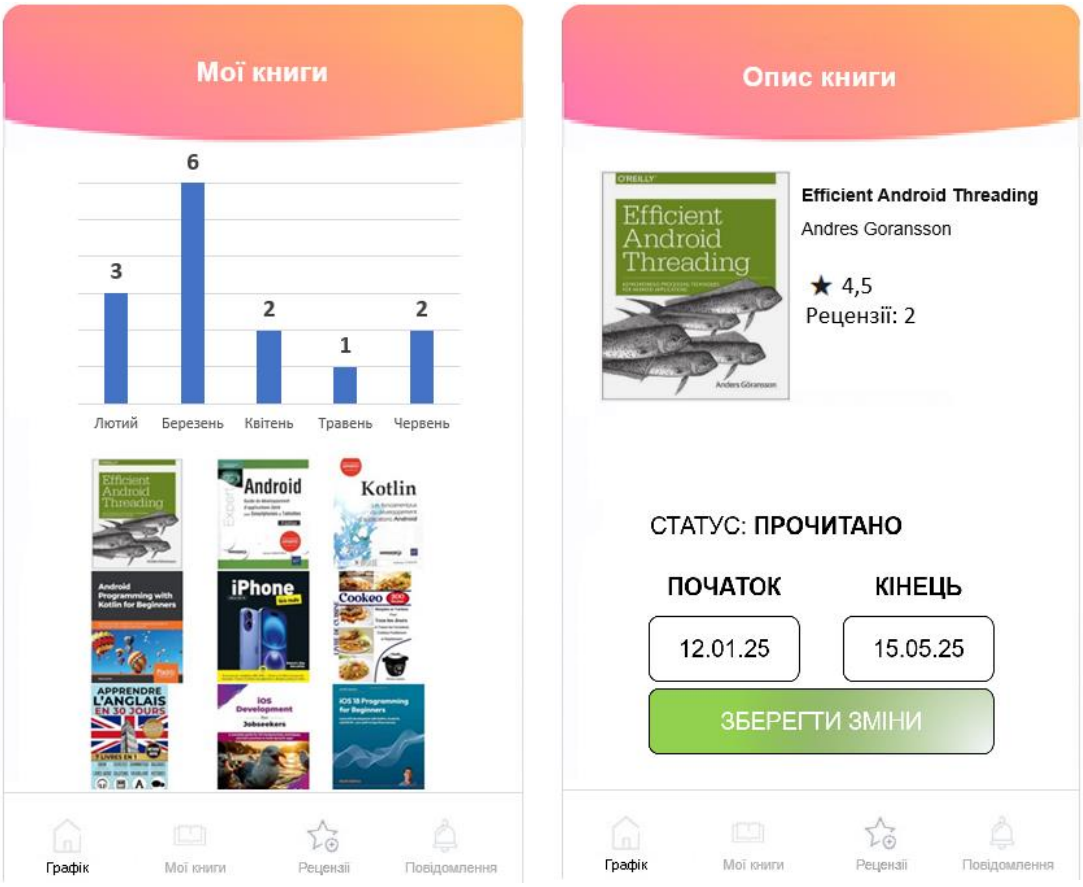
Рисунок 2.5 – Зовнішній вигляд розділу «Рецензії»

Розділ застосунку «Графік» є важливою складовою застосунку, яка забезпечує користувача зручною візуалізацією його читацької активності. У цьому розділі представлено графічну статистику, що відображає кількість

прочитаних книг користувачем за останній період (кожен місяць подається окремо у вигляді стовпчикової діаграми). Така форма подання дозволяє швидко оцінити динаміку читання, виявити найбільш активні періоди, а також мотивує користувача підтримувати або покращувати свій прогрес.

Окрім графіка, на сторінці також розміщено список усіх книг, які було прочитано користувачем. Цей список створено для швидкого доступу до прочитаних матеріалів у разі потреби. Наприклад, щоб переглянути деталі книги або повторно її перечитати.

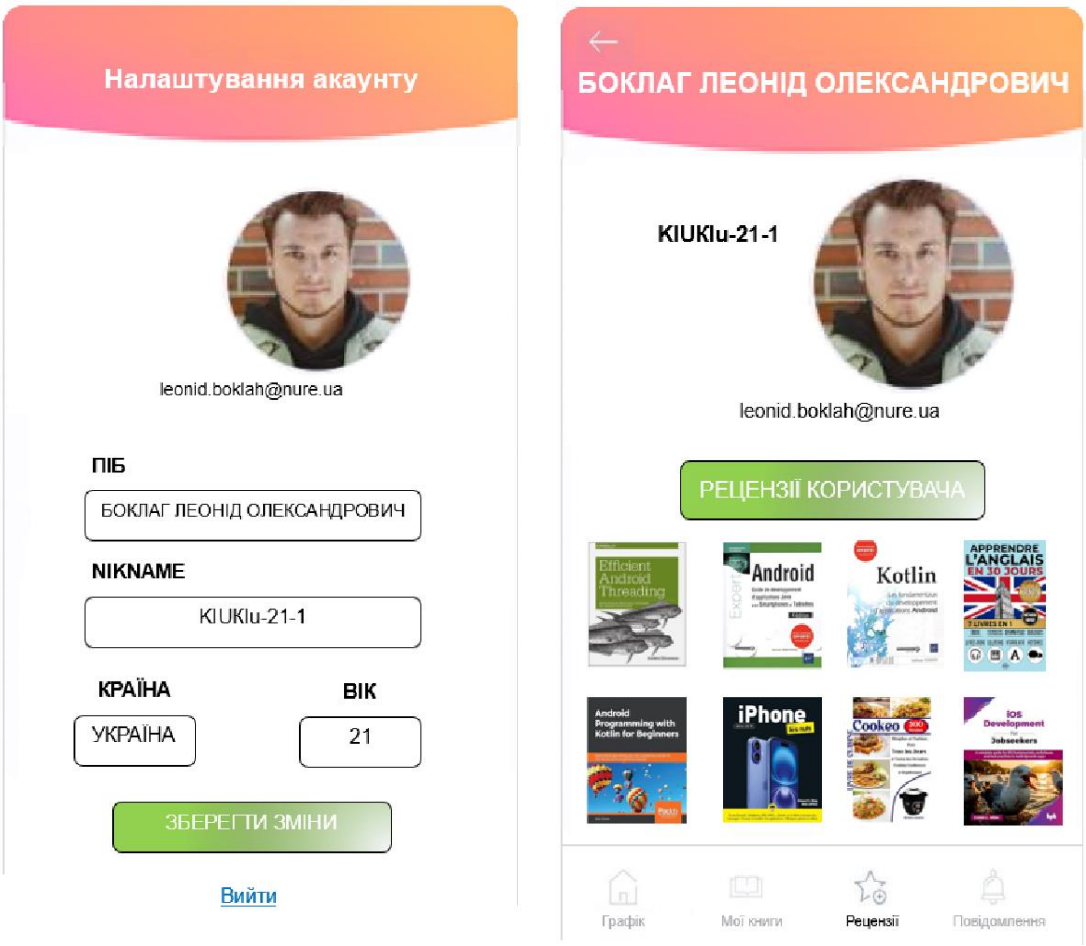
Для кожної книги передбачено можливість редагування інформації, зокрема вказання дати початку та завершення читання. На підставі цих даних автоматично визначається статус книги: «прочитано», «очікує читання» або інші можливі стани (наприклад, «у процесі читання»). Приклад редагування інформації про книгу наведено на рисунку 2.6.



а) б)

Рисунок 2.6 – Зовнішній вигляд розділу «Графік»

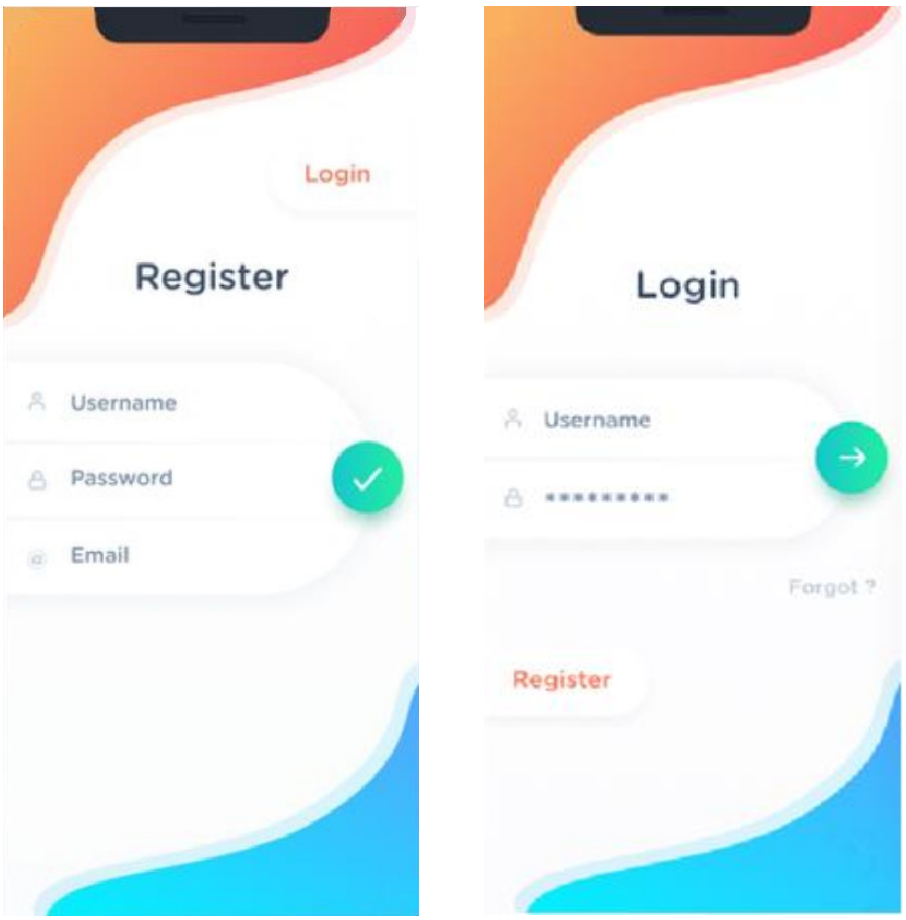
Крім розділу «Графік», застосунок також містить інші функціональні сторінки, які забезпечують повноцінну взаємодію користувача з платформою. Зокрема, це сторінка профілю користувача, де зберігається його персональна інформація, історія активності, а також налаштування облікового запису, які дозволяють змінювати пароль, налаштовувати сповіщення тощо.



а) б)

Рисунок 3.7 – Зовнішній вигляд сторінок користувача

Окрему увагу приділено процесам реєстрації та авторизації, які є обов’язковими для доступу до персоналізованих функцій застосунку. Сучасний інтерфейс сторінок реєстрації та входу в обліковий запис розроблено з урахуванням зручності користувача, що представлено на рисунках 2.7 та 2.8.



а) б)

Рисунок 3.8 – Зовнішній вигляд сторінок реєстрації та авторизації

3.3 Контекстний опис класів

Оскільки застосунок, що розробляється, реалізує архітектуру MVVM, тому він ділиться на три складові: «Model», «View» та «ViewModel». Для відображення служать XML-файли, а «Model» та «ViewModel» реалізуються за допомогою Kotlin-класів [3].

Основна логіка роботи програми міститься у класах «ViewModel». Вони обробляють запити, що надходять від користувача, отримують відповіді сервера, формують дані із відповідної «Model» та поміщають їх у «View». На кожен XML-файл припадає щонайменше один клас для його представлення. Також є класи для обробки запитів з пошуку потрібної інформації на сервері [9].

4 РОЗРОБКА ТА ТЕСТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

4.1 Архітектура мобільного застосунку

Під час розробки програми використовується шаблон проектування архітектури MVVM. Його перевагою є розподіл логіки та інтерфейсу програми. Структура шаблону представлена на рисунку 4.1.

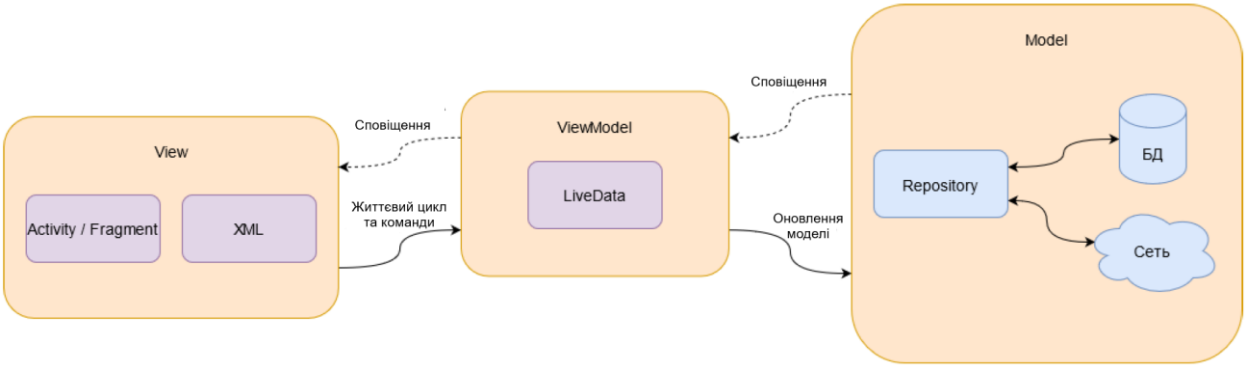


Рисунок 4.1 – Шаблон проектування MVVM

Програма має три шари: «View», «Model» та «ViewModel». Із застосуванням такої структури шар «View» не зберігатиме дані та стан об'єктів застосунку, а служитиме лише для відображення інформації. А всі дані після перезавантаження «View» знову будуть завантажуватимуться з «Model».

Функції відображення даних та надсилання подій користувача виконує шар «View». Шар «ViewModel», на відміну від «View», не обмежений «життєвим циклом» класів Fragment та Activity, тому зберігає дані в той момент, коли шар «View» знищений, а потім знову надає їх. Порівняння «життєвих циклів» обох шарів зображено рисунку 4.2. З цієї причини шар «ViewModel» не містить посилань на «View», тому що при використанні посилання на знищене «View» є ризик виникнення витоків пам'яті та збоїв у роботі програми.

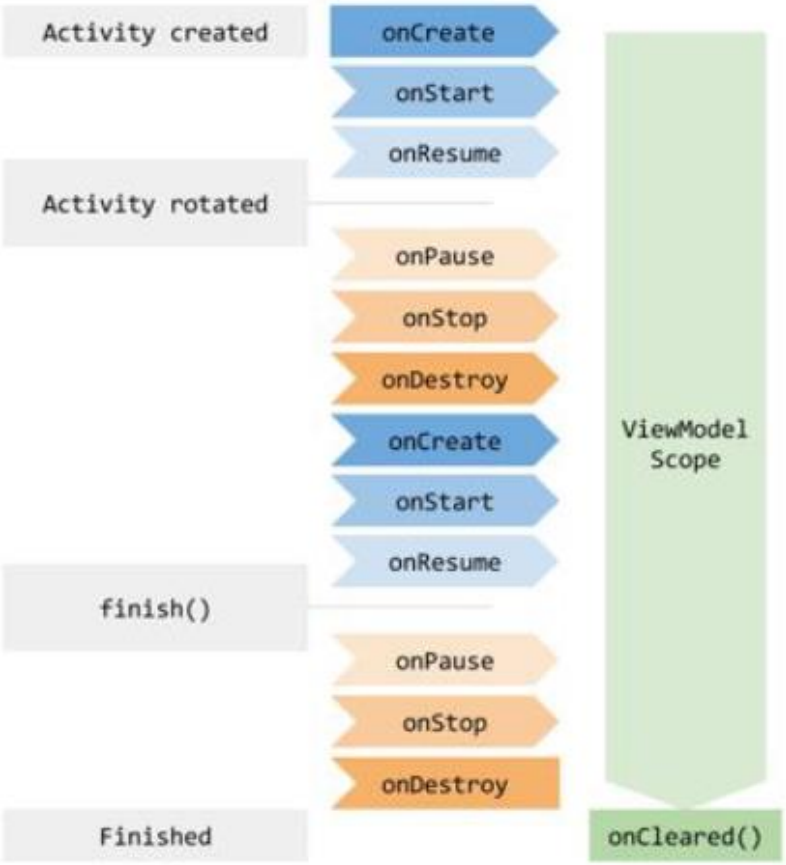


Рисунок 4.2 – Життєвий цикл класів ViewModel та Activity

Для взаємодії між «VeiwModel» та «View» використовується шаблон проектування «Спостерігач» (Observer), що використовує об'єкти LiveData, які спостерігаються із ViewModel. UML-діаграма шаблону представлена на рисунку 4.3 [7].

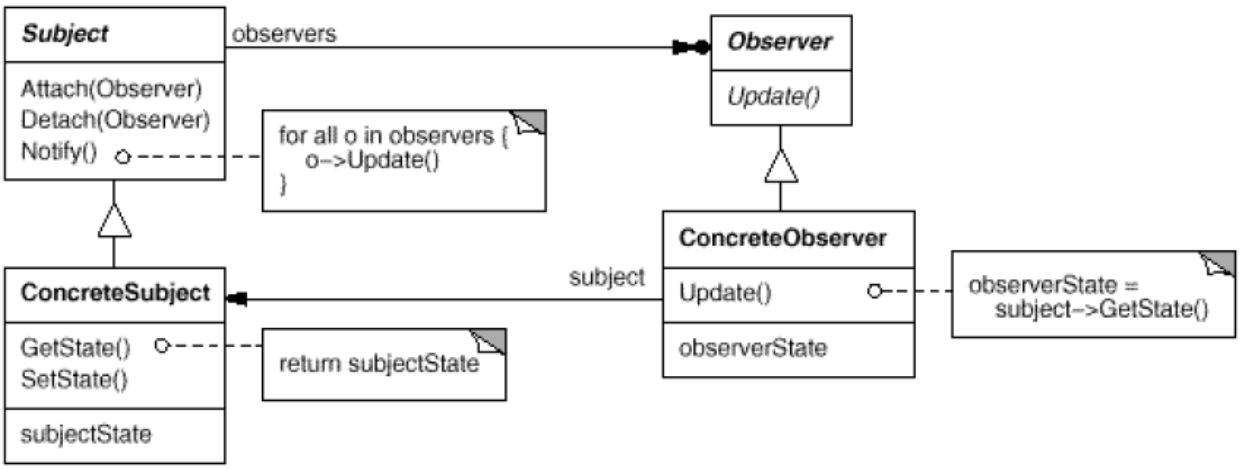


Рисунок 4.3 – UML-діаграма шаблону проектування «Спостерігач»

При реалізації взаємодії «View» підписується на зміни LiveData у «ViewModel». У даного підходу є ряд переваг. По-перше «ViewModel» зберігається при зміні конфігурації, тому немає необхідності повторно запитувати дані зовнішнього джерела (наприклад, БД або мережі). Коли завершуються тривалі операції, дані, що спостерігаються в «ViewModel» оновлюються. Так як«ViewModel» не посилається на «View», тому ризик витоку пам'яті менший.

У лістингу 4.1 наведено приклад підписки на поле даних «ViewModel» із класів Activity або Fragment.

Лістинг 4.1 – Підписка на поле даних «ViewModel» із класів Activity або Fragment

```
private fun subscribeToModel() {
// Observe book data
viewModel.getObservableBook().observe(this, object :
Observer<Book?>() {
fun onChanged(@Nullable book: Book)
mTitle.setText(book.title) }
})}
```

Програма має доступ до кількох джерел даних: мережа, локальні дані та кеш у пам'яті. Для роботи з вилученням та відправкою даних використовується один із найбільш поширених шаблонів проектування – Repository [7]. Репозиторій полегшує поділ бізнес-логіки та рівнів доступу до даних у застосунку. Він залежить від постійної моделі даних та віддаленого джерела даних. Таким чином, якщо користувач повертається до програми, то бачить інформацію, збережену локально. Якщо дані застаріли, модуль репозиторію програми починає оновлювати дані у фоновому режимі.

4.2 Модель внутрішнього сховища даних програми

На рисунку 4.4 представлена схема моделі внутрішнього сховища даних у програмному забезпеченні, що розробляється.

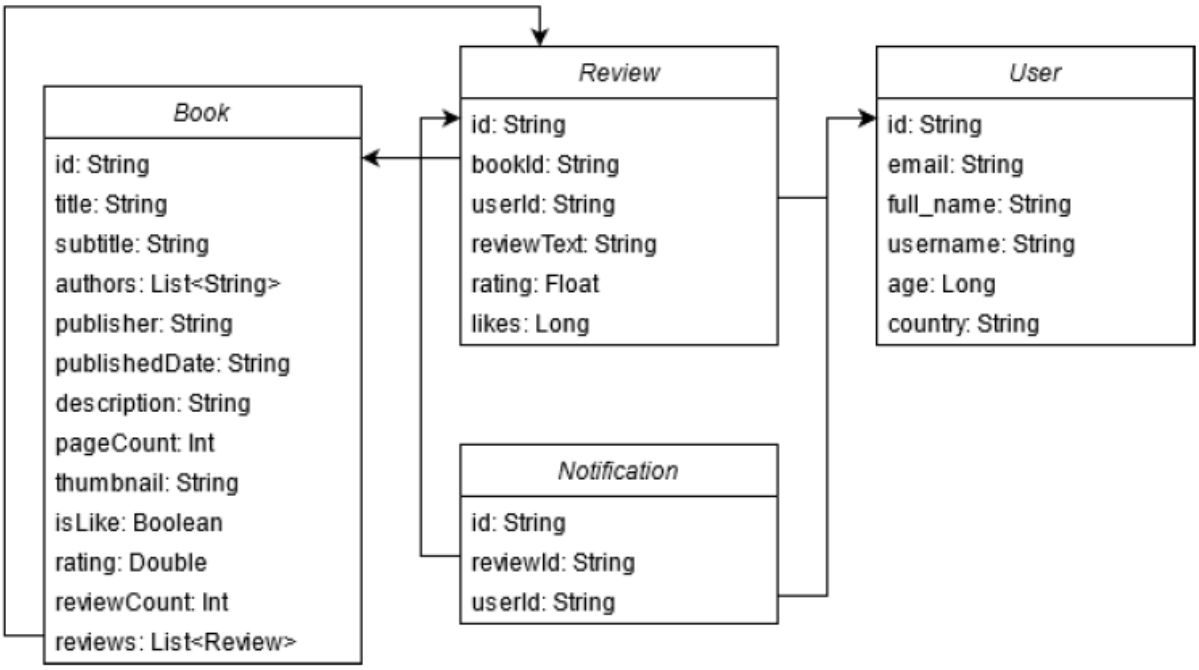


Рисунок 4.4 – Схема моделі внутрішнього сховища даних програми

Система містить 4 різні структури даних: User, Book, Review, Notification. Усі дані отримують оновлення з сервера щоразу перед відображенням на екрані користувача. Поле reviews моделі Book є масивом моделей Review. Поля userId та bookId моделі Review зберігають лише id користувачів та книг, але не самі моделі.

4.3 Реалізація класів-репозиторіїв

4.3.1 Клас AuthAppRepository

Клас AuthAppRepository призначений для реєстрації та авторизації користувача в застосунку, а також відповідає за надання особистих даних користувача шару подання. Клас містить поля, що зберігають ім'я користувача, нікнейм, електронну пошту, країну, вік та фотографію профілю. Для реєстрації та авторизації програма звертається до сервісу Firebase, який надає серверні служби для аутентифікації користувача за допомогою електронної пошти, телефону та низки соціальних мереж. Також клас

реалізує методи для виходу користувача з програми та відправлення оновлених даних на сервер.

У лістингу 4.2 наведено приклад одного з методів, що здійснюють взаємодію з Firebase, який реалізує реєстрацію користувача.

Лістинг 4.2 – Реалізація користувача

```
suspend fun login(email: String?, password: String?) {
withContext(Dispatchers.IO) {
firebaseAuth.signInWithEmailAndPassword(email!!, password!!)
.addOnCompleteListener(
ContextCompat.getMainExecutor(application.applicationContext),
{ task ->
if (task.isSuccessful) {
userLiveData.postValue(firebaseAuth.currentUser)
} else {
Toast.makeText(
application.applicationContext,
"Login Failure: " + task.exception?.message,
Toast.LENGTH_SHORT
).show() } }) } }
```

При успішному завершенні операції метод оновлює значення LiveData, яке буде використовуватися застосунком надалі без звернення до сервера. У разі виникнення помилки користувач бачить повідомлення із зазначенням причини її виникнення.

4.3.2 Клас BooksRepository

Клас BooksRepository призначений для отримання списку книг на запит користувача, для оновлення списку книг на віртуальній полиці користувача, а також для обробки додавання та видалення елементів з цього списку.

При реалізації перерахованих методів використовується не тільки клас BooksRepository, а й додаткові класи до роботи з мережею. Перший з них – DAO клас BooksApi, що описує шляхи даних.

На лістингу 4.3 наведено приклад методів searchBook(query: String) та searchCurrentBook(bookId: String) з цього класу.

Лістинг 4.3 – Реалізація методів searchBook та searchCurrentBook

```
@GET("v1/volumes")
suspend fun searchBook(
@Query("api_key") apiKey: String,
@Query("q") query: String, 36
@Query("page") page: Int = 1
): SearchBookResponse
@GET("v1/volumes/{id}")
suspend fun searchCurrentBook(
@Path("id") id: String,
@Query("api_key") apiKey: String
): BookNetworkModel
```

Анотація @GET вказує команду для запиту до вебсервісу та дозволяє вказати в якості параметру шлях до сервісу. У цьому випадку базова URL-адреса зберігається як константа, до якої під час запиту приєднується шлях, вказаний у параметрі. Анотації @Query та @Path означають параметри запиту та шлях запиту, що змінюється відповідно.

Другий клас, до якого винесена ініціалізація об'єкта класу Retrofit та самого класу BooksRepository – це клас AppComponent. У лістингу 4.4 наведено фрагмент даної ініціалізації. У конструктор класу Retrofit як параметри передаються: базова URL-адреса, GsonConverterFactory (призначена для зміни формату JSON-об'єктів) та описаний раніше клас BooksApi для формування запитів.

Лістинг 4.4 – Фрагмент ініціалізації класу Retrofit

```
init {
val api = Retrofit.Builder()
.baseUrl(BuildConfig.BASE_URL)
.addConverterFactory(GsonConverterFactory.create())
.build()
.create(BooksApi::class.java)
booksRepo = BooksRepository(api) }
```

4.3.3 Клас ReviewRepository

Клас ReviewRepository призначений для отримання списку рецензій та збереження нових рецензій на сервері. Оскільки програма розрахована на

багато користувачів, звернення до сервера здійснюється щоразу при оновленні списку рецензій для показу користувачеві. Клас реалізує методи setReview(review: Review) та getReviews(). Приклад звернення до бази даних рецензій для оновлення списку наведено у лістингу 4.5.

Лістинг 4.5 – Звернення до бази даних рецензій для оновлення списку

```
private val reviewsReference =
firebaseFirestore.collection("all_reviews")
reviewsReference.get().addOnSuccessListener
{ result ->
for (document in result) {
list.add(document.toObject())
}
}
```

Крім збереження нових рецензій на сервері, методи класу реалізують оновлення параметрів всіх даних, із якими пов'язаний створений елемент.

4.4 Реалізація інших класів програми

4.4.1 Клас AppComponent

Клас AppComponent реалізує ініціалізацію об'єкта бібліотеки Retrofit та об'єкта класу BooksRepository для роботи з мережею. Цей метод ініціалізації викликається з головного класу програми BookApp, успадкованого від Application. Виклик цього методу наведено у лістингу 4.6.

Лістинг 4.6 – Виклик методу ініціалізації

```
val myComponent: AppComponent by lazy { AppComponent(this) }
```

Тут ключове слово by lazy служить для відкладеної ініціалізації. Таким чином, при першому доступі функція використовується для ініціалізації, а при повторних зверненнях повертається вже отримане раніше значення.

4.4.2 Клас BookNetworkModel

Клас BookNetworkModel служить для зберігання даних, які отримують від сервера у форматі JSON [9]. Для перетворення даних у класі використовуються анотації @SerializedName, що приймаються як параметр імен полів з даними у форматі JSON. У лістингу 4.7 представлені поля класу.

Лістинг 4.7 – Поля класу

```
@SerializedName("id")
val id: String,
@SerializedName("volumeInfo")
var volumeInfo: VolumeInfo
```

Аналогічно реалізований клас VolumeInfo, що містить дочірні поля з прийнятими даними (лістинг 4.8).

Лістинг 4.8 – Реалізація класу VolumeInfo

```
@SerializedName("title")
var title: String,
@SerializedName("subtitle")
var subtitle: String,
@SerializedName("authors")
var authors: List<String>,
@SerializedName("publisher")
var publisher: String,
@SerializedName("publishedDate")
var publishedDate: String,
@SerializedName("description")
var description: String,
@SerializedName("pageCount")
var pageCount: Int,
@SerializedName("imageLinks")
var imageLinks: Cover?
```

4.4.3 Клас MainActivity

Клас MainActivity являє собою основний клас програми, що формує інтерфейс користувача та здійснює навігацію між різними екранами програми, які представлені класами Fragment. У лістингу 4.9 наведено

фрагмент коду, який містить логіку переходу між екранами. Навігація між екранами здійснюється користувачем за допомогою об'єкта класу BottomNavigationView – нижньої навігаційної панелі.

Лістинг 4.9 – Реалізація логіки переходу між екранами

```
bottomNavigationView?.setOnNavigationItemSelectedListener { item
->
when (item.itemId) {
R.id.my_books -> {
supportFragmentManager.beginTransaction()
.replace(R.id.container, BookListFragment.newInstance())
.commitNow() }
R.id.timeline -> {
supportFragmentManager.beginTransaction()
.replace(R.id.container, TimelineFragment.newInstance())
.commitNow() }
R.id.add_review -> {
supportFragmentManager.beginTransaction()
.replace(R.id.container, ReviewFragment.newInstance())
.commitNow() }
R.id.notifications -> {
supportFragmentManager.beginTransaction()
.replace(R.id.container,
NotificationsFragment.newInstance())
.commitNow() }
}
true
}
```

4.4.4 Клас UserBooksAdapter

Клас UserBooksAdapter відповідає за відображення списку книг на екрані. Клас репозиторію передає список даних уявленню, яке потім викликає метод bindBooks(booksList: List<Book>) адаптера для відтворення елементів списку на екрані (лістинг 4.10).

Лістинг 4.10 – Відтворення елементів списку на екрані

```
fun bindBooks(booksList: List<Book>) {
books = booksList
notifyDataSetChanged()
}
```

Функція notifyDataSetChanged() повідомляє адаптер про те, що список даних було оновлено. Також клас містить вкладений клас UserBooksViewHolder, який знаходить елементи на екрані та встановлює в них необхідні дані (лістинг 4.11).

Лістинг 4.11 – Реалізація класу UserBooksViewHolder

```
inner class UserBooksViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
fun onBind(book: Book) {
itemView.findViewById<ImageView>(R.id.book_cover).load(book.thum
bnail)
}
}
```

4.4.5 Клас StringListConverter

Клас StringListConverter призначений для конвертації списку рядків в один рядок при збереженні бази даних, а також для зворотного перетворення. У класі оголошено дві функції, які позначені анотацією @TypeConverter (лістинг 4.12). Цей клас передається як параметр для класу Book з анотацією @TypeConverters.

Лістинг 4.12 – Оголошення функцій

```
@TypeConverter
fun stringToList(s: String?) : List<String>? = s?.split(",")
@TypeConverter
fun listToString(list: List<String>?) : String? =
list?.joinToString()
```

4.5 Вибір способів тестування програми

Використання процесу тестування під час розробки мобільних застосунків має низку важливих переваг, які безпосередньо впливають на якість кінцевого продукту, задоволеність користувачів та і успішність самого застосунку. Ось основні переваги застосування тестування:

- підвищення якості програмного забезпечення: тестування дозволяє виявити помилки та недоліки в роботі застосунку ще до його публікації. Це забезпечує стабільну, передбачувану та надійну роботу продукту, що зменшує кількість збоїв та помилок під час використання;

- покращення досвіду користувача (UX): UI- та UX-тестування допомагають переконатися, що інтерфейс є інтуїтивно зрозумілим, зручним та приємним у використанні. Це особливо важливо для мобільних застосунків, де простота та швидкість взаємодії критично важливі;

- забезпечення сумісності з різними пристроями та платформами: мобільні застосунки повинні коректно працювати на широкому спектрі пристроїв з різними розмірами екранів, версіями операційної системи (Android, iOS) та технічними характеристиками. Тестування дозволяє перевірити сумісність та адаптивність інтерфейсу;

- зниження витрат на подальшу підтримку: виявлення та виправлення помилок на ранніх етапах розробки значно дешевше та простіше, ніж після випуску застосунку. Це зменшує витрати на технічну підтримку та оновлення;

- підвищення репутації продукту та бренду: якісний, добре протестований застосунок викликає більше довіри у користувачів. Це сприяє позитивним відгукам, вищому рейтингу в маркетплейсах та зростанню кількості завантажень;

- організація безпеки та захисту даних користувачів: тестування допомагає виявляти потенційні уразливості, які можуть призвести до витоку особистої інформації. Особливо це важливо у застосунках, що працюють з персональними або фінансовими даними;

- підтримка відповідності технічним та бізнес-вимогам: завдяки тестуванню можна переконатися, що всі бізнес-функції реалізовано згідно з технічним завданням, а застосунок виконує всі необхідні сценарії використання.

При розробці програмного забезпечення було проведено повний цикл тестування, що включав функціональне, інтеграційне та UI-тестування.

4.5.1 Функціональне тестування

Функціональне тестування – це тестування програмного забезпечення з метою перевірки реалізованості функціональних вимог, тобто можливості програмного забезпечення за певних умов вирішувати завдання, необхідних користувачеві. Для проведення функціонального тестування було розроблено автоматизовані тести з використанням бібліотеки Mockito, яка використовується для заміни об'єктів, пов'язаних із тестованим.

Приклад автоматизованого тесту для класу AuthAppRepository наведено на лістингу 4.13.

Лістинг 4.13 – Автоматизований тест для класу AuthAppRepository

```
public class AuthAppRepositoryTest {
AuthAppRepository authAppRepository = new AuthAppRepository();
@Test
public void getEmail() throws Exception {
String email = authAppRepository.email.observe(this, {
assertThat(email).isEqualTo(currentUser.email);
});
}}
```

В результаті тестування було визначено, що функції програми працюють згідно із заданими вимогами. Усі виявлені помилки усунуто.

4.5.2 UI-тестування

UI-тестування – це тестування графічного інтерфейсу користувача, яке дозволяє перевірити більшу частину дій користувача, а також взаємодію компонентів програми. Для проведення UI-тестування використаний фреймворк Espresso, що складається з 4 класів: Espresso (основний клас), ViewMatchers (здійснює пошук об'єкта на екрані), ViewActions (здійснює

взаємодію з об'єктом) та ViewAssertions (перевіряє стан об'єкта). Приклад автоматизованого UI-тесту для класу UserFragment наведено на лістингу 4.14.

Лістинг 4.14 – Автоматизований UI-тест для класу UserFragment

```
@RunWith(AndroidJUnit4.class)
public class UserFragmentTest {
@Rule
public final RuleChain rules = RuleChain
.outerRule(new CreateFileRule(getTestFile(), "{name :
$userName}"))
.around(new FragmentTestRule<>(MainActivity.class, new
UserFragment()));
@Test
public void nameDisplayed() {
onView(withText(userName)).check(matches(isDisplayed()));
}
}
```

У процесі UI-тестування було виявлено деякі недоробки у розробленому застосунку. В результаті їх усунення для всіх виявлених помилок під час використання програми були додані інформаційні повідомлення для користувача із зазначенням типу помилки та причини.

4.5.3 Інтеграційне тестування

Інтеграційне тестування – це повна перевірка ПЗ після його збирання з метою виявлення помилок, що виникають у процесі інтеграції програмних модулів чи компонентів.

Застосунок перевірено на всіх можливих дозволах екрана з діапазону, зазначеного у вимогах до апаратного забезпечення, щодо некоректності відображення елементів інтерфейсу. За результатами тестування, всі елементи відображаються коректно на всіх заявлених дозволах екрану.

Програма перевірена на всіх версіях операційної системи Android, а саме Android 5.0 – Android 11. У ході тестування також не виявлено помилок, програмне забезпечення працює коректно на всіх підтримуваних версіях операційної системи.

Завдяки комплексному підходу до тестування вдалося виявити й усунути критичні помилки ще на ранніх етапах розробки, що суттєво підвищило якість і стабільність фінального продукту.

Функціональне тестування дозволило переконатися, що всі окремі компоненти та функції застосунку працюють відповідно до заданих вимог – зокрема, правильність обробки даних, коректна робота форм, кнопок, логіки зміни статусів книг тощо.

Інтеграційне тестування було спрямоване на перевірку взаємодії між різними модулями системи, наприклад, між інтерфейсом користувача та базою даних, або між різними екранами застосунку, щоб гарантувати безперебійне функціонування всієї програми в цілому.

UI-тестування (тестування інтерфейсу користувача) забезпечило перевірку зовнішнього вигляду та зручності користування застосунком, включаючи відповідність елементів дизайну, читабельність текстів, зручність навігації, адаптивність під різні пристрої та загальну ергономіку.

ВИСНОВКИ

У роботі розглядається значення бібліотечного каталогу як основного інструменту для систематизації та пошуку книг у бібліотеці, наведена класифікація їх за типами (алфавітний, систематичний та електронний), а також виявлено особливості у їх використанні та призначенні. На даний час існує декілька методів створення каталогів для домашніх бібліотек. В роботі наведено 4 способи: ведення паперового каталогу, використання офісних програм, застосування спеціалізованих програм і застосунків, а також створення власної бази даних. Кожен метод має свої переваги й недоліки: від простоти та доступності до високої гнучкості та технологічності. Вибір залежить від технічної підготовки користувача, кількості книг і потреб у швидкості доступу та аналізу інформації. Усі способи мають на меті зберегти порядок у бібліотеці та полегшити пошук літератури.

Для створення каталогу домашньої бібліотеки було обрано мову програмування Kotlin, систему збірки Gradle та середовище розробки Android Studio, оскільки ці інструменти забезпечують сучасний, ефективний та зручний процес розробки під платформу Android. Kotlin є мовою для Android і поєднує в собі лаконічність синтаксису, безпечність при роботі з нульовими значеннями та повну сумісність з Java, що дозволяє легко інтегрувати сторонні бібліотеки або адаптувати наявний код. Використання Gradle як системи автоматизованого збирання дозволяє гнучко керувати залежностями, конфігураціями і процесом складання застосунку, що особливо корисно в умовах змінного або зростаючого функціоналу. Android Studio, як офіційне середовище розробки, забезпечує розширені можливості для налагодження, візуального дизайну інтерфейсу, а також інтеграцію з емулюючими пристроями, що значно прискорює тестування і розгортання програми. Комбінація цих технологій дозволяє ефективно реалізувати інтуїтивний і стабільний застосунок для ведення домашньої бібліотеки з урахуванням потреб користувача та специфіки мобільної платформи.

У процесі виконання даної кваліфікаційної роботи було реалізовано мобільний застосунок, який дозволяє спростити пошук книг та впорядкувати особисту бібліотеку за допомогою створення своєї віртуальної полиці книг, а також надати можливість ділитися враженнями від прочитаного. Для досягнення цієї мети було вирішено такі задачі:

- проведено аналіз існуючих аналогів для ПЗ, що розробляється;
- вивчені засоби розробки для операційної системи Android;
- визначено вимоги та спроектовано мобільний застосунок;
- реалізовано та протестовано мобільний застосунок.

Вимоги, що пред'являються до програмного забезпечення, що розробляється, виконані. Реалізовані можливості програми дозволяють проводити пошук книг, їх збереження в особисту віртуальну бібліотеку, писати рецензії, оцінювати рецензії інших користувачів, а також знайомитись з книжковою полицею інших користувачів. Крім того, можливий доступ до облікового запису з різних мобільних пристроїв.

В результаті роботи розроблено програмне забезпечення, яке готове до практичного застосування. При подальшій розробці програми можуть бути внесені такі доробки:

- реалізація функції експорту даних у формат CSV;
- реалізація функції обговорення користувачами рецензій;
- використання розділу статей з добірками книг.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Alexander M., Kusleika R. Access 2019 Bible. Publisher: John Wiley & Sons Inc, 2018. 1136 p.

2. Tanenbaum A.S., Bos H. Modern Operating Systems, Global Edition [Print Replica]. Publisher: Pearson, 5 th edition, 2023. 21.7 MB

3. Guide to app architecture. URL: https://developer.android.com/jetpack/guide

4. Elizarov R., Isakova S., Aigner S., Jemerov D. Kotlin in Action. Publisher: Manning Publications, 2 nd Edition, 2024. 560 p.

5. Hubert Klein Ikkink. Gradle Effective Implementation Guide. Publisher: Packt Publishing Ltd., 2012. 382 p.

6. Smyth N. Android Studio Ladybug Essentials – Kotlin Edition: Developing Android Apps Using Android Studio Ladybug and Kotlin [Kindle Edition]. Publisher: Payload Media Inc, 1 st Edition, 2024. 65.1 MB

7. Gamma E., Helm R., Johnson R., Vlissides J., Booch G. Design Patterns: Elements of Reusable Object-Oriented Software. Publisher: Addison-Wesley Professional, 1 st edition, 1994. 416 p.

8. Goransson A. Efficient Android Threading. Publisher: O′Reilly, 2014. 280 p.

9. OkHttp Recipes. URL: https://square.github.io/okhttp/recipes