УДК 004.4
DOI: https://doi.org/ 10.17721/3041-2323.2025.330-348

Євгеній ТОПОЛЬСЬКОВ, канд.техн.наук, доц.
ORCID ID: 0000-0001-5587-3069
e-mail: y.topolskov@knu.ua
Київський національний університет
імені Тараса Шевченка, Київ, Україна

Вадим РИБЧИНЧУК, студент
ORCID ID: 0009-0002-5868-9608
e-mail: afecn777@gmail.com
Державний університет
«Київський авіаційний інститут», Київ, Україна

УДОСКОНАЛЕНА АРХІТЕКТУРА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ЧИТАННЯ ЕЛЕКТРОННИХ КНИГ

У роботі проведено аналіз тенденцій розвитку мобільних застосунків для читання електронних книг, зокрема їх поширення у світі, представлено класифікацію основних типів таких застосунків і здійснено огляд сучасних архітектурних принципів їхньої побудови. Окрему увагу приділено архітектурним принципам, які забезпечують персоналізацію, масштабованість, офлайн-доступ, модульність, аналітику користувацьких дій, адаптивність інтерфейсу, а також інтеграцію з хмарними та соціальними сервісами. Запропоновано удосконалений варіант архітектури MVVM з елементами Clean Architecture для розділення UI та бізнес-логіки за допомогою реактивних прив'язок даних та забезпечення незалежності застосунку від конкретних UI-технологій, баз даних або зовнішніх API. Результати дослідження є корисними для фахівців та розробників у сфері цифрових технологій, програмної інженерії, електронного видавництва та освіти.

К л ю ч о в і с л о в а : ***мобільний застосунок, е-книга, читання книг, електронна бібліотека, архітектура програмного забезпечення.***

Вступ

Стрімкий розвиток інформаційно-комунікаційних технологій змінив способи взаємодії користувачів з текстовим контентом. Зростаючі обсяги інформації вимагають удосконалення інструментів для її пошуку та систематизації. Мобільні застосунки для пошуку і читання книг стали головним каналом

споживання літератури для мільйонів користувачів, а також дозволяють інтегрувати інструменти рекомендацій, аналізу користувацьких вподобань і забезпечення персоналізованого досвіду. Зважаючи на різноманіття функціональних потреб – від перегляду художньої літератури до читання технічної документації та підручників – виникає потреба в ефективних архітектурних рішеннях, здатних забезпечити надійність, масштабованість і адаптивність таких застосунків. Водночас ринкові тенденції, що характеризуються попитом на інноваційні рішення, висувають більш широкі функціональні і нефункціональні вимоги до мобільних застосунків для читання книг. Для задоволення цих вимог доцільно провести аналіз існуючих архітектурних принципів побудови мобільних застосунків та розробити удосконалений варіант архітектури мобільного застосунку для читання книг.

Аналіз останніх досліджень і публікацій

Активний розвиток цифрових технологій та зручність мобільних пристроїв кардинально змінили ландшафт споживання текстового контенту (Chavez-Helaconde et al., 2020). Мобільні застосунки для читання книг стали невід'ємною частиною повсякденного життя мільйонів користувачів по всьому світу. Згідно з даними Statista (Statista. Global: eBooks number of users, 2025) та наукових досліджень кількість користувачів мобільних застосунків для читання книг досягла майже 1,1 млрд в 2024 році з темпами зростання приблизно 9% щороку. У США та Європі спостерігається значне проникнення електронного читання: наприклад, у США близько 75% населення читає книги у будь-якому форматі, причому електронні книги та аудіокниги відіграють значну роль. Ринок електронного читання демонструє стабільне зростання, прогнозуючи збільшення обсягу до 22.76 мільярдів доларів США до 2030 року. В Азії, особливо в Китаї та Індії, мобільні телефони є основним засобом доступу до електронних книг (Statista. eBooks – Worldwide, 2025). Наприклад, у Китаї понад 340 мільйонів мобільних читачів було зафіксовано ще у 2017 році, що свідчить про високу адаптацію до мобільного читання. У Західній Європі та Південній Кореї також спостерігається зростання популярності застосунків із

соціальними функціями, дозволяючи користувачам обмінюватись думками, рецензіями та взаємодіяти в читацьких спільнотах (Statista. eBooks – Worldwide, 2025). Додатково Африка й Латинська Америка демонструють підвищений інтерес до безкоштовних освітніх платформ, зокрема Worldreader, які надають доступ до тисяч електронних книг через мобільні телефони, сприяючи цифровій освіті та підвищенню грамотності у регіонах з обмеженим доступом до друкованої літератури (Revankar, 2025).

В Україні також спостерігається динамічне зростання популярності мобільних застосунків для читання. Серед провідних платформ, що активно використовуються, можна виділити Yakaboo, PocketBook Reader, Librera та Headway (Top 5 Books Apps Performance in Ukraine, 2024). Ці застосунки набувають особливої популярності серед молоді у закладах вищої освіти та шанувальників електронних книг, що відображає загальносвітову тенденцію до мобільного та доступного читання. Важливою характеристикою сучасних застосунків є також інтеграція з хмарними сервісами для синхронізації бібліотек та прогресу читання, розширення соціальних функцій для взаємодії між користувачами, а також розробка мультиплатформних рішень, що забезпечують єдиний користувацький досвід на різних пристроях. Ці комплексні функціональні можливості, підкріплені надійною архітектурою, забезпечують постійну динаміку зростання ринку електронного читання.

Сучасні мобільні застосунки для читання значно розширили свій функціонал. Вони активно інтегрують соціальні функції, хмарні сервіси та мультиплатформні рішення, що забезпечує синхронізацію бібліотек і прогресу читання між різними пристроями (Meštrović, & Jakominić, 2017).

Багато застосунків використовують штучний інтелект для персоналізованих рекомендацій та адаптації контенту. Для підвищення мотивації читачів застосовуються елементи гейміфікації та відстеження прогресу. Окремі платформи є орієнтованими на освітній контент або на «фрагментоване» читання та пропонують короткі виклади книг, що відповідає сучасному швидкому темпу життя.

Постановка завдання

Основними завданнями дослідження насамперед є визначення ключових тенденцій поширення мобільних застосунків для читання книг у світі, їх класифікація за функціональними ознаками та аналіз основних архітектурних принципів, які забезпечують ефективну побудову та підтримку таких цифрових продуктів. За результатами цих досліджень було досягнуто основну мету – розроблення удосконаленого варіанту архітектури мобільного застосунку для читання книг та його практична реалізація.

Основна частина і результати

З огляду на зростаюче різноманіття функціональних потреб користувачів та еволюцію цифрового контенту, мобільні застосунки для читання книг розвиваються в кількох основних напрямках. Ця диверсифікація дозволяє задовольняти потреби як випадкових читачів, так і академічних користувачів, і пропонує спеціалізовані можливості для кожного сценарію. Розглянемо класифікацію найбільш поширених типів застосунків, що ілюструє їх функціональні відмінності та призначення (Bhati, 2024):

- Мобільні застосунки з підтримкою численних форматів (ePub, FB2, PDF, MOBI). Ці застосунки є універсальними інструментами, призначеними для відтворення електронних книг у найрізноманітніших форматах. Вони забезпечують максимальну гнучкість для користувача, дозволяючи читати файли незалежно від їхнього походження чи платформи, на якій вони були створені. Ключовою особливістю є можливість налаштування шрифтів, розмірів, міжрядкового інтервалу, фону та режимів читання (денний/нічний), що робить процес читання максимально комфортним. Вони часто пропонують функції закладки, виділення тексту та пошуку всередині книги. Приклади: PocketBook Reader, Moon+ Reader, Librera.
- Застосунки-бібліотеки (Електронні бібліотеки). Цей тип застосунків об'єднує функціонал електронного читання з великим інтегрованим каталогом книг. Вони надають користувачам доступ до широкого асортименту літератури, часто з можливістю покупки або передплати на книги безпосередньо в

застосунку. Основними характеристиками є розширені функції пошуку, персоналізовані рекомендації на основі історії читання, створення власних колекцій та синхронізація між пристроями. Це створює цілісну екосистему для споживання контенту. Приклади: Amazon Kindle, Google Play Книги, MyBook, Bookmate.

- Соціальні платформи для читачів. Ці застосунки створюють інтерактивне середовище, що перетворює читання з індивідуального досвіду на соціальну активність. Вони дозволяють користувачам не лише читати, а й активно взаємодіяти між собою: обмінюватися рецензіями, оцінками, коментарями, створювати власні історії та спілкуватися з іншими авторами та читачами. Такі платформи часто включають елементи гейміфікації та інтерактивного контенту, залучаючи широку аудиторію, особливо молодь. Приклади: Wattpad, Goodreads, Rork.
- Аудіозастосунки. Фокусуються виключно на аудіокнигах, що дозволяє користувачам «читати» книги на слух. Вони пропонують зручні функції відтворення, такі як зміна швидкості прослуховування, таймери сну, можливість перемотування, створення закладок та синхронізація прогресу прослуховування між різними пристроями. Популярність аудіокниг зростає завдяки можливості споживати контент під час інших занять (наприклад, подорожей, занять спортом). Приклади: Audible, Storytel, Абук.
- Освітні платформи. Призначені для доступу до академічного, наукового та освітнього контенту. Вони часто інтегруються з системами управління навчанням (LMS) вищих навчальних закладів, бібліотеками та науковими базами даних. Такі застосунки підтримують інтерактивні формати, що дозволяє працювати з підручниками, науковими публікаціями, виконувати вправи та тести. Їхня цінність полягає у забезпеченні доступу до знань та сприянні дистанційному навчанню. Приклади: Perlego, Elsevier, Worldreader.

Ефективна розробка мобільних застосунків для читання книг вимагає застосування чітких та раціональних архітектурних принципів, які забезпечують їхню стабільність, масштабованість,

безпеку та зручність використання. Ці принципи виступають як фундаментальні настанови, що скеровують інженерні рішення на всіх етапах життєвого циклу розробки програмного забезпечення. Їх впровадження дозволяє створювати гнучкі та високопродуктивні системи, здатні адаптуватися до постійно зростаючих вимог користувачів та технологічних змін.

Доцільно розглядати не менше десяти основних архітектурних принципів, що відображають важливі особливості й ключові переваги та відповідають за реалізацію конкретних функціональних вимог до мобільних застосунків (табл. 1).

Таблиця 1

Аналіз архітектурних принципів побудови мобільного застосунку

№	Архітектурний принцип	Опис та призначення
1	Розділення відповідальностей (Separation of Concerns)	Логічний поділ застосунку на окремі, незалежні шари або модулі (наприклад, інтерфейс користувача, бізнес-логіка, доступ до даних). Це спрощує розробку, тестування та підтримку.
2	Архітектурні шаблони (Architectural Patterns)	Використання перевірених структурних рішень, таких як MVVM, MVP, MVC або Clean Architecture, для організації коду та потоків даних, що забезпечує структурованість, модульність та легкість масштабування.
3	Офлайн-доступ і кешування	Збереження контенту (книг, прогресу читання, закладок) у локальній базі даних або кеші пристрою, дозволяючи користувачеві отримувати доступ до функцій застосунку без активного підключення до Інтернету.
4	Синхронізація даних	Автоматичне та надійне оновлення користувацьких даних (прогресу читання, закладок, нотаток, колекцій) між різними пристроями через хмарні сервіси, забезпечуючи безперервний досвід.

Продовження таблиці 1

№	Архітектурний принцип	Опис та призначення
5	Модульність	Розбиття функціональності застосунку на невеликі, самодостатні та взаємозамінні компоненти або модулі (наприклад, окремі модулі для читалки, бібліотеки, магазину, профілю користувача).
6	Реактивність	Використання реактивних парадигм програмування (наприклад, бібліотек RxJava, Kotlin Flow, Combine) для ефективної обробки асинхронних подій та змін даних у режимі реального часу, забезпечуючи плавну взаємодію з UI.
7	Безпека даних	Забезпечення конфіденційності та цілісності користувацьких даних шляхом шифрування чутливої інформації, захисту облікових записів, впровадження надійної автентифікації та авторизації.
8	Масштабованість	Здатність архітектури системи ефективно розширюватися та підтримувати зростаючу кількість користувачів, обсяг даних та нові функціональні можливості без суттєвої зміни базової структури.
9	Кросплатформність	Застосування фреймворків та інструментів (наприклад, Flutter, React Native, Xamarin), що дозволяють створювати єдину кодову базу для розгортання застосунку на різних мобільних операційних системах (iOS, Android).
10	Орієнтація на користувача (UX-дизайн)	Фокусування на створенні інтуїтивно зрозумілого, приємного та ефективного користувацького досвіду, що включає ергономічний дизайн, зручну навігацію, адаптивну типографіку, нічний режим та інші функції, що покращують читання.

Крім загальних архітектурних принципів, у розробці мобільних застосунків для читання книг активно застосовуються конкретні архітектурні моделі, які надають структурований підхід до організації коду (Lano & Tehrani, 2023). Ці моделі допомагають керувати складністю, підвищувати якість програмного забезпечення та спрощувати його подальший розвиток. Серед найбільш поширених архітектурних підходів часто виділяють чотири типи архітектури мобільного застосунка: MVVM (Model-View-View Model), MVC (Model-View-Controller), MVP (Model-View-Presenter) та Clean Architecture (Vijaywargi & Boddapati, 2024).

Одним з найпопулярніших у мобільній розробці є патерн MVVM (Model-View-ViewModel), що є особливо ефективним для побудови застосунків зі складним та динамічним інтерфейсом (рис. 1).

Рис. 1. *Схема архітектурного патерну MVVM*

Цей патерн чітко розділяє застосунок на Model (дані та бізнес-логіка), View (інтерфейс користувача) та ViewModel (модель представлення), яка виступає посередником, готуючи дані для відображення та обробляючи дії користувача. Головною перевагою MVVM є використання data-прив'язки та реактивного програмування, що дозволяє автоматично оновлювати інтерфейс при зміні даних та ефективно обробляти асинхронні події. Це значно спрощує тестування бізнес-логіки та підвищує модульність коду, що особливо корисно для застосунків, де часто оновлюється стан (наприклад, прогрес читання або синхронізація закладок).

Поряд з MVVM, широко застосовується архітектурні патерни MVC (Model-View-Controller) та MVP (Model-View-Presenter), які також забезпечують розділ відповідальностей (рис. 2).

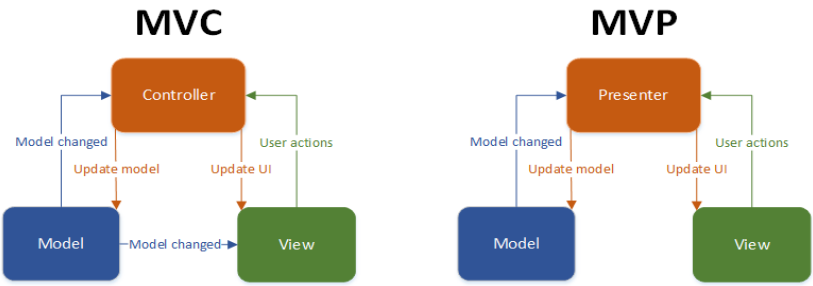

Рис. 2. *Порівняльні схема архітектурних патернів MVC та MVP*

Реалізація архітектури MVC полягає в розбитті програмного коду на три компонента: Model (відповідає за дані, що використовуються програмою), View (описує зовнішній вигляд, що використовуються програмою) і Controller (контролює те, як функціонуватиме програма). Завдяки такому поділу забезпечується зручність проєктування, тестування і супроводу (Basri, 2018). Члени команди проєктувальників можуть працювати над кожним із цих трьох компонентів незалежно один від одного, тим самим, скорочуючи час розробки та спрощуючи налагодження і тестування. Моделі можуть бути повторно використані з різними представленнями, а компоненти (моделі, представлення і контролери) можуть бути використані в різних частинах одного або навіть в різних застосунків. Це дозволяє легко додавати нову функціональність або змінювати існуючу без переробки всієї архітектури забезпечуючи гнучкість і масштабованість.

У MVP Model керує даними, а View є пасивним інтерфейсом. Проте, на відміну від MVVM і MVC, у MVP ключову роль відіграє Presenter (представник) – активний компонент, який отримує події від View, взаємодіє з Model та безпосередньо оновлює View. Така архітектура робить Presenter повністю незалежним від UI, забезпечуючи легкість тестування та підвищуючи якість коду. MVP ідеально підходить для застосунків зі складною логікою представлення, де потрібне жорстке

розділення обов'язків між візуальною частиною та логікою її відображення, забезпечуючи високу масштабованість.

На вищому рівні абстракції знаходиться патерн Clean Architecture (чиста архітектура) – філософія побудови програмного забезпечення, що створює гнучкі та незалежні системи (рис. 3).

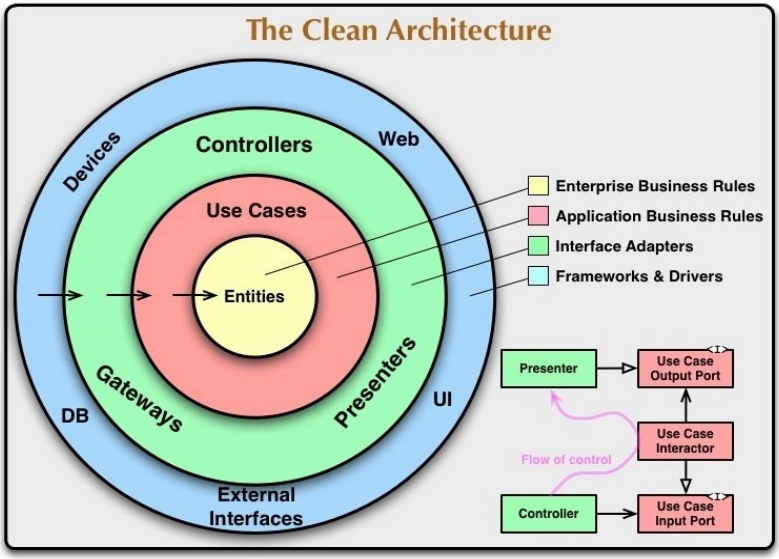

Рис. 3. *Схема Clean Architecture*

Центральна ідея Clean Architecture полягає в організації коду у вигляді концентричних шарів, де зовнішні компоненти залежать від внутрішніх, але не навпаки. Ядром є доменний шар з бізнес-правилами, що повністю незалежний від технологій. Зовнішні шари (шар застосування, інфраструктурний шар) містять лише реалізації для взаємодії з UI, базами даних тощо. Головний принцип – «Правило залежностей» – гарантує, що зміни у зовнішніх компонентах не впливають на основну бізнес-логіку. Це робить Clean Architecture ідеальним вибором для великих та складних застосунків, що потребують тривалої підтримки, частих

змін та високої стійкості до зовнішніх впливів, забезпечуючи при цьому легкість тестування та модульність на найвищому рівні.

Виходячи з особливостей мобільних застосунків для читання книг, що характеризуються інтенсивною взаємодією з користувачем, частими оновленнями даних (прогрес, закладки), потребою в гнучкості інтерфейсу та можливості легко інтегрувати нові функції, архітектурна модель MVVM (Model-View-ViewModel) у поєднанні з принципами Clean Architecture є найбільш раціональним вибором. MVVM забезпечує ідеальне розділення UI та бізнес-логіки за допомогою реактивних прив'язок даних, що значно спрощує розробку та тестування інтерактивних елементів, таких як сторінки читання, налаштування та синхронізація. Доповнення MVVM елементами Clean Architecture дозволяє зберегти ядро застосунку (бізнес-логіку та правила) незалежним від конкретних UI-технологій, баз даних або зовнішніх API. Це забезпечує максимальну гнучкість, масштабованість та довговічність системи, дозволяючи легко адаптуватися до майбутніх змін у вимогах або технологіях без переписування основної логіки, що критично для програмних продуктів з тривалим циклом життя та постійним оновленням контенту. Clean Architecture фокусує увагу на чіткому поділі рівнів – дані, домен, інтерфейс та сприяє високій тестованості, що особливо корисно у масштабних проєктах із потенціалом розвитку. При цьому, обидві архітектури мають одну й ту ж мету: розділити завдання та логіку на шари або компоненти. MVVM добре інтегрується з сучасними фреймворками, зокрема Flutter або Jetpack Compose у середовищі Android (Bizzotto, 2023).

Такий підхід дозволяє забезпечити двосторонній зв'язок між інтерфейсом і даними, що є критично важливим для функцій на кшталт закладок, персоналізації або фільтрації книг. У випадку з Flutter-застосунком для читання книг таке поєднання сприяє масштабованості, зручності тестування та швидкому впровадженню нових функцій. Запропонований варіант можна вважати удосконаленою і адаптованою формою Clean Architecture, яка добре поєднується з GetX, Provider та хмарними сервісами Firebase.

Для побудови архітектури застосунку було використано модульний підхід, що структурує проєкт на незалежні частини – окремі вікна, сервіси, моделі, контролери та репозиторії. Таке компонування забезпечує гнучкість, розширення функціоналу та легкість у масштабуванні проєкту (AlexCodeX, 2025).

Управління станом у застосунку реалізується за допомогою Provider та GetX (рис. 4).

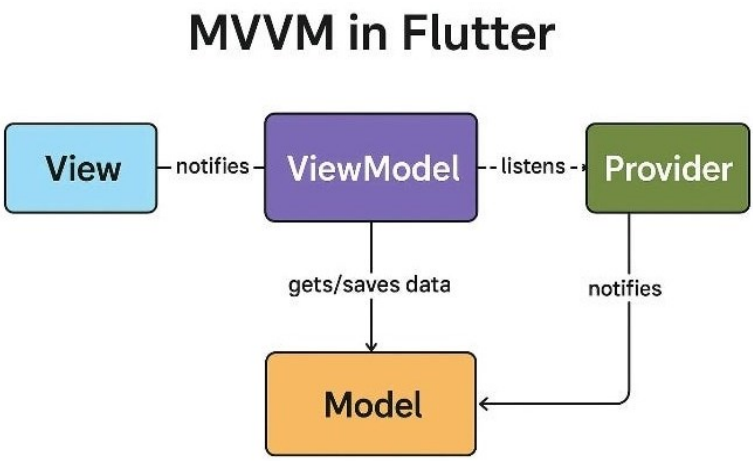

Рис. 4. *Схема взаємодії компонентів архітектури MVVM у середовищі Flutter*

Provider використовується переважно для глобального доступу до даних користувачів, списку книг, обраних налаштувань, тощо. У мобільному застосунку для читання книг він використовується для авторизації обраного користувача та отримуванні інформації про нього: логін, дата реєстрації, електронна пошта, стать, особиста колекція книг і так далі.

Натомість GetX відповідає за маршрутизацію між вікнами, реактивну взаємодію елементів інтерфейсу, обробку подій та локальний стан елементів в системі. В першу чергу, він використовується для ефективного отримання даних про книги та їх пошук через записи в хмарній базі даних.

Бекенд-застосунок повністю реалізований на базі хмарної платформи Firebase, яка інтегрується з фреймворком Flutter завдяки офіційній підтримці SDK. Firebase надає комплексні сервіси для аутентифікації, зберігання даних у режимі реального часу, обробки файлів, аналітики та хостингу (Google Firebase, 2025).

Основні компоненти платформи Firebase, що були задіяні в реалізації застосунку:

– Firebase Authentication використовується для безпечної аутентифікації користувачів. Мобільний застосунок підтримує вхід за допомогою електронної пошти та паролю, а також передбачає можливість реєстрації нових користувачів та виходу з облікового запису. Завдяки цій системі забезпечується індивідуальний доступ до користувацьких даних і персональних бібліотек;

– Cloud Firestore – це гнучка, масштабована NoSQL база даних, яка використовується для зберігання основного вмісту застосунку: профілів користувачів, списків книг, розділів кожної книги, коментарів, уподобань тощо. Дані зберігаються у вигляді колекцій і документів, що дозволяє реалізувати складну ієрархію даних з фільтрацією та пошуком за категоріями, тегами або назвами;

– Firebase Storage застосовується для збереження мультимедійного контенту – обкладинок книг, зображень для категорій, а також PDF- або EPUB-файлів, якщо застосунок реалізує можливість завантаження та читання документів локально;

– Firebase Analytics і Crashlytics інтегровані для відстеження взаємодії користувача із застосунком, а також для швидкого виявлення можливих помилок чи збоїв у роботі системи.

Відповідно до розробленої архітектури представлено діаграму взаємодії основних компонентів застосунку (рис. 5).

Таким чином, взаємодія фронтенду з бекендом мобільного застосунку організована за допомогою відповідних контролерів і сервісів, що спрощують логіку підключення до Firebase.

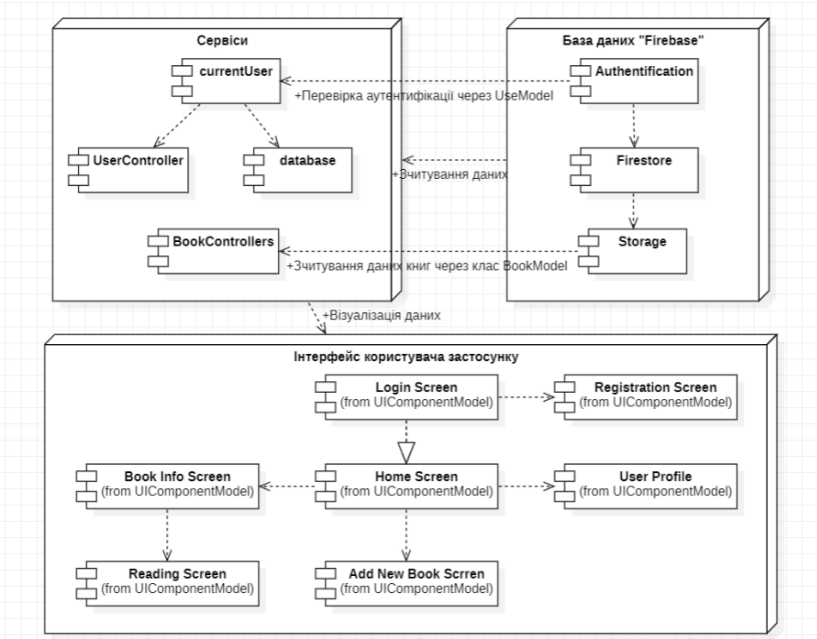

Рис. 5. *Діаграма взаємодії компонентів застосунку для читання книг*

Для кожного функціонального модулю застосунку створений окремий сервіс, які реалізують логіку запису, оновлення, вилучення або читання даних із Firestore.

Сутності у програмній реалізації представлені класами мовою Dart, що реалізовані за принципом технології об’єктно-орієнтованого програмування. Кожна сутність відповідає окремому елементу структури бази даних або логіки застосунку та має власний набір властивостей і методів.

У табл. 2 наведено основні сутності, які застосовуються в логіці мобільного застосунку, їх тип і призначення.

Такі моделі як BookModel, UserModel і BookChapte використовуються для інкапсуляції даних, отриманих із Firebase, а контролери керують їхнім станом та обробкою.

Таблиця 2

Опис сутностей застосунку

Назва сутності	Тип (клас / контролер)	Призначення
BookModel	Клас-модель	Представляє дані книги, зчитані з Firestore; зберігає назву, автора, рейтинг, список розділів тощо.
UserModel	Клас-модель	Відповідає за дані користувача: UID, ім'я, регіон тощо.
BookController	Контролер	Отримує дані з колекції books, виконує фільтрацію та пошук книг.
UserController	Контролер	Завантажує поточного користувача та дозволяє доступ до його книг/метаданих.
UserController	Контролер	Завантажує поточного користувача та дозволяє доступ до його книг/метаданих.
BookChapter	Клас-модель	Представляє окремий розділ книги з полями: назва, вміст, номер.
CurrentUser	Модуль (сервіс)	Реєстрація, аутентифікація користувачів.
Database	Модуль (клас)	Реалізує основну логіку взаємодії з Firebase Firestore, включаючи створення записів користувачів при реєстрації.

Приклад Dart-коду атрибутів класу BookModel наведений на рис. 6.

```dart
// Модель книги
class BookModel {
  String? id; // ідентифікаційний номер книги
  String? bookID; // номер книги
  String? name; // назва книги
  String? author; // автор
  String? description; // опис
  String? category; // категорія(жанр)
  String? status; // статус
  String? bookImage; // зображення книги
  String? language; // мова книги
  int? numberofRating; // кількість оцінок
  double? rating; // оцінка книги
  Timestamp? dateCreated; // дата коли була створена або додана книга
  List<BookChapter>? chapterList; // список розділів
  int? chaptersCount; // кількість розділів
  Map<String, double>? userRatings; // Нове поле: оцінки користувачів
```

Рис. 6. *Dart-код атрибутів класу BookModel*

Загалом використання таких окремих модулів та моделей не лише покращує загальну структуру коду, але й дозволяє забезпечити високу швидкість доступу до даних, надійність у роботі з Firebase, а також сприяє гнучкому масштабуванню архітектури у випадку подальшого розширення функціональності мобільного застосунку. Це відповідає сучасним архітектурним принципам мобільної розробки, заснованим на модульності, розділуі відповідальностей і використанні реактивного підходу до оновлення інтерфейсу.

Дискусія і висновки

Поширення мобільних застосунків для читання книг у світі свідчить про значний інтерес до мобільного і доступного всюди читання. На фоні цифрової трансформації освіти та видавничої справи, архітектура таких застосунків повинна враховувати масштабованість, безпеку, UX-дизайн та кросплатформність.

У результаті проведених досліджень сучасних архітектурних принципів та моделей було розроблено на основі патернів MVVM і Clean Architecture адаптований варіант архітектури мобільного застосунку для читання книг, який забезпечує гнучкість та можливості подальшого масштабування і розвитку системи, що відповідає ключовим принципам створення довговічних та конкурентоспроможних продуктів на ринку застосунків для електронного читання.

Список використаних джерел

AlexCodeX. (2025). *Mastering MVVM in Flutter with Provider (2025 edition)*. Medium. https://medium.com/@AlexCodeX/mastering-mvvm-in-flutter-with-provider-2025-edition-7d325395d6b8.

Basri, U. (2018). *Penerapan pattern MVP di Java*. Medium. https://medium.com/@basriumar/penerapan-pattern-konsep-mvp-di-java-94ad56846cca.

Bizzotto, A. (2023). *A comparison of popular Flutter app architectures*. Code With Andrea. https://codewithandrea.com/articles/comparison-flutter-app-architectures/.

Bhati, N. (2024). *Complete guide to build ebook reader mobile app. Emizentech Tech Blog: Mobile App, eCommerce, Salesforce Insights*. EmizenTech. https://emizentech.com/blog/develop-ebook-reading-app.html.

Chavez-Helaconde E., et al. (2020). Mobile application to improve reading habits using virtual reality. In *Communications in Computer and Information Science.* 160–170. Springer. https://doi.org/10.1007/978-3-030-66919-5_17.

Lano, K., & Yassipour Tehrani, S. (2023). *Introduction to software architecture*. Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-44143-1.

Meštrović, D., & Jakominić Marot, N. (2017). E-book reading applications – Innovative technology as a response to growing consumer preferences and its implications for tourism. In *Tourism in Southern and Eastern Europe 2017: Tourism and Creative Industries: Trends and Challenges*. https://doi.org/10.20867/tosee.04.12.

Firebase. (2025). *Realtime database Firebase docs*. Google. https://firebase.google.com/docs/database/web/read-and-write.

Revankar, S. (2025). *eBooks statistics by revenue, user, country, sales, genre and facts*. Coolest Gadgets. https://coolest-gadgets.com/ebooks-statistics/.

Statista. (2025). *eBooks – Worldwide*. https://www.statista.com/outlook/amo/media/books/ebooks/worldwide?currency=usd.

Statista. (2025). *Global: eBooks number of users*. https://www.statista.com/forecasts/1294239/number-of-ebook-users-global.

Sensor Tower. (2024). *Top 5 books apps performance in Ukraine*. Digital Intelligence & App Data Analysis. https://sensortower.com/blog/2024-q1-unified-top-5-books-units-ua-6042071f241bc16eb8c33b77.

Vijaywargi, A., & Boddapati, U. K. (2024). Architectural patterns in Android development: Comparing MVP, MVVM, and MVI. *International Journal for Research in Applied Science and Engineering Technology, 12*(4), 4611–4616. https://doi.org/10.22214/ijraset.2024.60762.

References

AlexCodeX. (2025). *Mastering MVVM in Flutter with Provider (2025 edition)*. Medium. https://medium.com/@AlexCodeX/mastering-mvvm-in-flutter-with-provider-2025-edition-7d325395d6b8.

Basri, U. (2018). *Penerapan pattern MVP di Java*. Medium. https://medium.com/@basriumar/penerapan-pattern-konsep-mvp-di-java-94ad56846cca.

Bizzotto, A. (2023). *A comparison of popular Flutter app architectures*. Code With Andrea. https://codewithandrea.com/articles/comparison-flutter-app-architectures/.

Bhati, N. (2024). *Complete guide to build ebook reader mobile app. Emizentech Tech Blog: Mobile App, eCommerce, Salesforce Insights*. EmizenTech. https://emizentech.com/blog/develop-ebook-reading-app.html.

Chavez-Helaconde E., et al. (2020). Mobile application to improve reading habits using virtual reality. In *Communications in Computer and Information Science.* 160–170. Springer. https://doi.org/10.1007/978-3-030-66919-5_17.

Lano, K., & Yassipour Tehrani, S. (2023). *Introduction to software architecture*. Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-44143-1.

Meštrović, D., & Jakominić Marot, N. (2017). E-book reading applications – Innovative technology as a response to growing consumer preferences and its implications for tourism. In *Tourism in Southern and Eastern Europe 2017: Tourism and Creative Industries: Trends and Challenges*. https://doi.org/10.20867/tosee.04.12.

Firebase. (2025). *Realtime database Firebase docs*. Google. https://firebase.google.com/docs/database/web/read-and-write.

Revankar, S. (2025). *eBooks statistics by revenue, user, country, sales, genre and facts*. Coolest Gadgets. https://coolest-gadgets.com/ebooks-statistics/

Statista. (2025). *eBooks – Worldwide*. https://www.statista.com/outlook/amo/media/books/ebooks/worldwide?currency=usd.

Statista. (2025). *Global: eBooks number of users*. https://www.statista.com/forecasts/1294239/number-of-ebook-users-global.

Sensor Tower. (2024). *Top 5 books apps performance in Ukraine*. Digital Intelligence & App Data Analysis. https://sensortower.com/blog/2024-q1-unified-top-5-books-units-ua-6042071f241bc16eb8c33b77.

Vijaywargi, A., & Boddapati, U. K. (2024). Architectural patterns in Android development: Comparing MVP, MVVM, and MVI. *International Journal for Research in Applied Science and Engineering Technology, 12*(4), 4611–4616. https://doi.org/10.22214/ijraset.2024.60762.

Отримано редакцією журналу / Received: 25.09.25
Прорецензовано / Revised: 27.09.25
Схвалено до друку / Accepted: 01.10.25

Yevhenii TOPOLSKOV, PhD(Engin.), Assoc. Prof.
ORCID ID: 0000-0001-5587-3069
e-mail: y.topolskov@knu.ua
Taras Shevchenko National University of Kyiv, Kyiv, Ukraine

Vadym RYBCHYNCHUK, Student
ORCID ID: 0009-0002-5868-9608
e-mail: afecn777@gmail.com
State University "Kyiv Aviation Institute", Kyiv, Ukraine

ENHANCED ARCHITECTURE OF A MOBILE APPLICATION FOR E-BOOK READING

This paper presents an analysis of current trends in the development of mobile applications for e-book reading, including their global distribution. It offers a classification of the main types of such applications and provides an overview of modern architectural principles used in their design. Particular attention is given to architectural approaches that support personalization, scalability, offline access, modularity, user behavior analytics, interface adaptability, and integration with cloud and social services. An enhanced MVVM-based architecture is proposed, incorporating elements of Clean Architecture to separate the UI and business logic through reactive data bindings and ensure the application's independence from specific UI technologies, databases, or external APIs. The findings of this study are valuable for professionals and developers in the fields of digital technology, software engineering, electronic publishing, and education.

Keywords: ***mobile application, e-book, book reading, digital library, software architecture.***

Автори заявляють про відсутність конфлікту інтересів. Спонсори не брали участі в розробленні дослідження; у зборі, аналізі чи інтерпретації даних; у написанні рукопису; в рішенні про публікацію результатів.

The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.