Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та безпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

ДОСЛІДЖЕННЯ АРХІТЕКТУРНИХ РІШЕНЬ ПРИ СТВОРЕННІ КРОСПЛАТФОРМЕННОГО ДОДАТКУ ЕЛЕКТРОННОЇ БІБЛІОТЕКИ.

RESEARCH INTO ARCHITERTURAL SOLUTIONS WHEN CREATING A CROSS-PLATFORM ELECTRONIC LIBRARY APPLICATION

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІм-21
Годлевський Захар Едуардович

(підпис)

Керівник:
к.т.н., доцент
Пех Петро Антонович

(підпис)

Кваліфікаційну роботу
допущено до захисту
«____» грудня 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Гринюк Сергій Васильович

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *магістр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Т.ТЕРЛЕЦЬКИЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Годлевському Захару Едуардовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Дослідження архітектурних рішень при створенні кросплатформенного додатку електронної бібліотеки*

Керівник роботи *к.т.н., доцент Пех Петро Антонович*

затверджені наказом закладу вищої освіти від «17» червня 2025 року № 290/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 09.12.2025р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області, різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Аналіз предметної області та існуючих рішень
Теоретичні основи кросплатформленої розробки
Дослідження архітектурних рішень для розробки кросплатформеного додатку електронної бібліотеки
Висновки

5. Перелік графічного (ілюстративного) матеріалу:

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області та існуючих рішень	*Пех П.А., доцент*		
Теоретичні основи кросплатформленої розробки	*Пех П.А., доцент*		
Дослідження архітектурних рішень для розробки кросплатформеного додатку електронної бібліотеки	*Пех П.А., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Гринюк С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І., ст.викладач*		

7. Дата видачі завдання 18.06.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми*	До 01.08.2025 р.	
2.	*Аналіз предметної області та існуючих рішень*	До 20.08.2025 р.	
3.	*Теоретичні основи кросплатформленої розробки*	До 25.09.2025 р.	
4.	*Дослідження архітектурних рішень для розробки кросплатформеного додатку електронної бібліотеки*	До 20.10.2025 р.	
5.	*Висновки та пропозиції*	До 25.10.2025 р	
6.	*Формування списку використаних джерел*	До 27.10.2025 р.	
7.	*Формування додатків*	До 30.10.2025 р.	
8.	*Оформлення ілюстративного матеріалу*	До 05.11.2025 р.	
9.	*Представлення остаточного варіанту кваліфікаційної роботи керівникові*	До 11.11.2025 р.	
10.	*Нормоконтроль*	До 29.11.2025 р.	
11.	*Інструментальна перевірка на академічний плагіат*	До 02.12.2025 р.	
12.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	До 09.12.2025 р.	

Здобувач вищої освіти _____ Годлевський З.Е.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Пех П.А.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Годлевський З. Е. Дослідження архітектурних рішень при створенні кросплатформного додатка електронної бібліотеки.

Кваліфікаційна робота магістра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота присвячена вирішенню актуальної задачі вибору та реалізації оптимальних архітектурних рішень для створення кросплатформних програмних систем на прикладі електронної бібліотеки. Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі проведено детальний аналіз предметної області, розглянуто існуючі аналоги (PocketBook, Moon+Reader тощо), сформульовано функціональні та нефункціональні вимоги до системи, а також спроєктовано структуру бази даних та інтерфейсу користувача. У другому розділі здійснено теоретичне дослідження підходів до кросплатформної розробки. Проаналізовано архітектурні патерни (MVC, MVP, MVVM) та стилі побудови додатків (монолітна, мікросервісна, багаторівнева архітектури), а також методи управління станом у сучасних фреймворках. У третьому розділі виконано практичну реалізацію та порівняльний аналіз прототипів додатку електронної бібліотеки з використанням різних технологічних стеків: нативні веб-технології (HTML/JS), Flutter, Xamarin, React Native, .NET MAUI та Kotlin Multiplatform (KMP). На основі розроблених прототипів складено порівняльну характеристику фреймворків за критеріями продуктивності, спільного коду та зручності розробки UI.

Ключові слова: електронна бібліотека, кросплатформна розробка, архітектура ПЗ, MVVM, Flutter, .NET MAUI, React Native, Kotlin Multiplatform, мобільний додаток.

ANNOTATION

Godlevsky Z. Research on architectural solutions for creating a cross-platform electronic library application.

Master's thesis in Computer Engineering, specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2025.

This thesis is devoted to solving the urgent problem of selecting and implementing optimal architectural solutions for creating cross-platform software systems using the example of an electronic library. The thesis consists of an introduction, three chapters, conclusions, a list of references, and appendices.

The first chapter provides a detailed analysis of the subject area, reviews existing analogues (PocketBook, Moon+Reader, etc.), formulates functional and non-functional requirements for the system, and designs the structure of the database and user interface. The second chapter provides a theoretical study of approaches to cross-platform development. Architectural patterns (MVC, MVP, MVVM) and application development styles (monolithic, microservice, multi-tier architectures) are analyzed, as well as state management methods in modern frameworks. The third section presents the practical implementation and comparative analysis of prototypes of an electronic library application using different technology stacks: native web technologies (HTML/JS), Flutter, Xamarin, React Native, .NET MAUI, and Kotlin Multiplatform (KMP). Based on the developed prototypes, a comparative characteristic of frameworks was compiled according to the criteria of performance, shared code, and UI development convenience.

Keywords: electronic library, cross-platform development, software architecture, MVVM, Flutter, .NET MAUI, React Native, Kotlin Multiplatform, mobile application.

ЗМІСТ

ВСТУП

Актуальність дослідження зумовлена стрімким розвитком мобільних технологій та зростаючими вимогами користувачів до доступності інформаційних ресурсів. Сучасні електронні бібліотеки повинні забезпечувати безперебійний доступ до контенту незалежно від операційної системи (iOS, Android, Windows) чи типу пристрою користувача. Традиційна нативна розробка під кожну платформу окремо призводить до значних фінансових та часових витрат, а також ускладнює підтримку коду. У цьому контексті використання кросплатформних фреймворків стає стратегічно важливим рішенням. Однак, різноманіття сучасних інструментів (Flutter, MAUI, React Native, KMP) та архітектурних патернів створює проблему вибору оптимального підходу, який би гарантував високу продуктивність, масштабованість та зручність користування. Таким чином, дослідження спрямоване на аналіз та систематизацію архітектурних рішень для створення ефективного, гнучкого та універсального додатку електронної бібліотеки.

Метою дослідження є дослідження, проєктування та розробка архітектурних рішень для створення кросплатформного додатку електронної бібліотеки, а також проведення порівняльного аналізу сучасних фреймворків для визначення найефективнішого інструментарію реалізації.

Об’єктом дослідження є процеси побудови, функціонування та проєктування кросплатформних програмних систем для доступу до цифрових бібліотечних фондів.

Предмет дослідження – архітектурні підходи (MVC, MVP, MVVM), моделі організації коду, принципи розробки та сучасні інструментальні засоби (фреймворки) створення кросплатформних застосунків.

Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести аналіз предметної області, існуючих систем електронних бібліотек та сформулювати чіткі функціональні та нефункціональні вимоги до розроблюваної системи;

– дослідити теоретичні основи кросплатформної розробки, включаючи порівняння нативного, гібридного та кросплатформного підходів;

– проаналізувати та обґрунтувати вибір архітектурних патернів (зокрема MVVM) для побудови клієнтської частини додатку;

– спроєктувати та розробити серію прототипів додатку електронної бібліотеки з використанням різних технологій: HTML/CSS/JS, Flutter, Xamarin, React Native, .NET MAUI та Kotlin Multiplatform;

– виконати експериментальне дослідження та порівняльний аналіз реалізованих рішень за критеріями архітектурної складності, швидкодії, можливості повторного використання коду та якості інтерфейсу користувача.

Наукова новизна роботи полягає у систематизації та порівняльному аналізі ефективності застосування новітніх кросплатформних фреймворків (.NET MAUI, Kotlin Multiplatform) саме для специфіки інформаційних бібліотечних систем, що дозволило сформувати рекомендації щодо вибору архітектури залежно від бізнес-вимог проєкту.

Практичне значення роботи полягає у створенні функціональних прототипів додатку електронної бібліотеки на базі різних платформ. Отримані результати та програмні реалізації можуть бути використані як основа для розробки повноцінних комерційних або освітніх бібліотечних систем, а наведена порівняльна характеристика фреймворків дозволяє розробникам скоротити час на етап вибору технологічного стеку.

Апробація результатів роботи. Результати дослідження були представлені на IV міжнародній науково-практичній інтернет-конференції [1], а також подані до публікації у журналі [2] (додаток В).

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Електронні бібліотеки: поняття, функції та класифікація

Електронна бібліотека являє собою організовану колекцію цифрових об’єктів, що включає текстові документи, зображення, аудіозаписи, відеоматеріали та інші форми мультимедійного контенту, доступні користувачам через електронні засоби зв’язку та комп’ютерні системи. На відміну від традиційних бібліотек, що оперують фізичними носіями інформації, електронні бібліотеки надають унікальну можливість одночасного доступу необмеженої кількості користувачів до одного й того ж інформаційного ресурсу, при цьому не обмежуючись географічними межами та часом роботи установи [3].

Концепція електронної бібліотеки почала формуватися ще у 1960-х роках з появою перших цифрових архівів наукових публікацій, проте справжній розквіт цього напрямку розпочався у 1990-х роках з широким поширенням інтернету та розвитком технологій оцифрування документів. Історичний аналіз дозволяє зафіксувати ключові моменти еволюції електронних ресурсів, від їх впровадження до сучасного стану, що сприяє збереженню цінного досвіду для майбутніх поколінь [4]. Сьогодні електронні бібліотеки стали невід’ємною частиною глобальної інформаційної інфраструктури, надаючи доступ до мільйонів публікацій, книг, журналів та інших матеріалів людям по всьому світу.

Термінологія електронних бібліотек не є усталеною і продовжує розвиватися разом з технологіями. За визначенням ДСТУ 5034-2008, електронна бібліотека це бібліотека, у якій документи зберігаються і використовуються в машино читаний формі, з якою можна працювати на відстані [3]. Проте існує значна кількість інших визначень, що відображають різні підходи до розуміння цього поняття. Дослідниця Христина Боргман виділяє два основних класи визначень: ті, що походять від дослідників електронних бібліотек, які є

переважно комп'ютерними науковцями та інженерами, і від фахівців бібліотечної та інформаційної сфери [3].

Виходячи з цього, Боргман наводить декілька визначень електронної бібліотеки як сукупності електронних ресурсів і пов'язаних з ними технічних можливостей для створення, пошуку та використання інформації, а також як певного ресурсу, що створюється, збирається та організовується спільнотою користувачів, де функціональні можливості задовольняють інформаційні потреби цієї спільноти [3].

Український дослідник В. А. Резніченко визначає електронну бібліотеку як інтегровану інформаційну систему, яка дозволяє накопичувати, зберігати та ефективно використовувати різноманітні колекції електронних повнотекстових та мультимедійних документів, що доступні в зручному для користувача вигляді [3]. На думку І. Павлуші, електронна бібліотека є видом розподіленої документальної системи, яка зберігає документи у форматі електронних записів незалежно від їх фізичного місця розташування або власника, забезпечуючи програмними засобами доступ через глобальну інформаційну мережу з одного інтерфейсу [3].

М. Яцимірська дала таке визначення електронної бібліотеки як інформаційної системи зберігання та ефективного використання різних форм електронних документів з можливістю передання інформації в зручному для користувачів форматі [3]. Н. С. Прилуцька зазначає, що більшість науковців вважає електронну бібліотеку інформаційною системою, де під цим терміном мається на увазі впорядкований фонд електронних документів, зокрема текстових, який реалізується у вигляді складних бібліотечно-інформаційних систем [3].

Основні функції електронних бібліотек охоплюють широкий спектр операцій з цифровим контентом. Функція зберігання та організації контенту передбачає не лише збереження власне цифрових файлів, але й систематизацію метаданих, що описують ці матеріали за різними критеріями, включаючи категорії, авторство, тематику, ключові слова та хронологічні рамки. Це створює

складну багатовимірну структуру, що дозволяє користувачам ефективно орієнтуватися у великих масивах інформації.

Функція пошуку та навігації є критично важливою для забезпечення практичної корисності електронної бібліотеки. Сучасні системи повинні надавати користувачам потужні інструменти для швидкого знаходження необхідних матеріалів, включаючи повнотекстовий пошук, фасетну фільтрацію, семантичний пошук та інтелектуальні рекомендаційні системи. Ефективна навігація передбачає інтуїтивно зрозумілу структуру каталогів, можливість переходу між пов’язаними матеріалами та візуалізацію зв’язків між різними документами.

Управління доступом та авторизація користувачів становлять важливий аспект функціонування електронних бібліотек, особливо коли йдеться про контент, захищений авторськими правами або доступний лише для певних категорій користувачів. Система повинна забезпечувати гнучке управління правами доступу, підтримувати різні моделі ліцензування та дотримуватися вимог законодавства про захист інтелектуальної власності.

Функціональність читання та перегляду контенту виходить за межі простого відображення тексту на екрані. Сучасні електронні бібліотеки повинні надавати користувачам зручні інструменти для комфортної роботи з контентом, включаючи можливість налаштування параметрів відображення тексту, підтримку різних режимів читання для різних умов освітлення, інструменти для роботи з таблицями та зображеннями, а також спеціалізовані можливості для роботи з науковими публікаціями, такі як навігація по посиланнях та цитатам.

Персоналізація користувацького досвіду стає все більш важливою функцією електронних бібліотек. Система повинна запам’ятовувати персональні налаштування користувача, зберігати закладки та позиції читання, надавати можливість створення власних колекцій та списків для читання, зберігати історію перегляду та пошукових запитів. Деякі системи також впроваджують соціальні функції, що дозволяють користувачам обмінюватися думками про

прочитане, залишати рецензії, створювати обговорення та рекомендувати матеріали іншим користувачам.

Мета створення електронних бібліотек полягає у забезпеченні користувачів доступом до документів з обмеженим доступом, таких як рідкісні книги, рукописи, фотоальбоми, дисертації, архіви, яких немає у більшості бібліотек, доступі до інформації, що існує лише в електронній формі та наданні користувачам більш якісних можливостей роботи з електронними документами великих обсягів [3]. Крім того, електронні бібліотеки забезпечують користувачів повнотекстовими базами даних у режимі теледоступу для задоволення їх інформаційних потреб.

До основних завдань електронної бібліотеки, окрім повних текстів видань, відноситься і можливість роботи з ними. За Н. Самохіною, головною умовою створення електронної бібліотеки є її спрямованість на збереження та ефективне використання інформації згідно з певними вимогами: відповідна системність під час формування електронного фонду документів та забезпечення користувачів необхідними вихідними даними й інструментами для пошуку, навігації, перегляду та експорту інформації [3].

Серед особливостей електронних бібліотек можна виділити здатність до введення, видалення, інтеграції та реструктуризації інформаційних об’єктів, хоча такі операції зазвичай здійснюються переважно з електронними документами, а не з самою інформацією, що міститься в них. Визначена концепція формування інформаційного простору, який є доступним користувачам, та можливість каталогізації об’єктів і їхніх різних об’єднань, що складають цей інформаційний простір, також є характерними ознаками [3].

Класифікація електронних бібліотек може проводитися за різними критеріями. За суб’єктом створення розрізняють електронні бібліотеки, створені фізичними особами, державними установами та приватними установами. За способом доступу виділяють відкриті, закриті, платні та безоплатні бібліотеки [3].

За функціональною спрямованістю електронні бібліотеки поділяються на загальні та спеціалізовані. Загальні містять інформаційні ресурси в різних галузях знань та надають стандартні функції інформаційної системи з використанням мінімальних засобів. Спеціалізовані електронні бібліотеки забезпечують доступ до інформаційних ресурсів певної предметної галузі та є багатофункціональними, забезпечуючи можливість нетрадиційної обробки інформації, такої як зберігання результатів та архівів експериментів, підтримка часових та просторових властивостей даних та обробка спеціальних форм вхідних і вихідних даних [3].

За змістом електронні бібліотеки класифікуються на прості та складні. За типом контенту розрізняють текстові бібліотеки, що зосереджуються переважно на електронних книгах, наукових статтях, журналах, мультимедійні, які включають аудіокниги, відеолекції, інтерактивні матеріали, та гібридні, що поєднують різні типи контенту [3].

За сферою застосування виділяють академічні електронні бібліотеки, що обслуговують потреби університетів та інших навчальних закладів, наукові бібліотеки та репозиторії, що спеціалізуються на збереженні та поширенні наукових публікацій, публічні цифрові бібліотеки, які надають широкий доступ до культурної та освітньої літератури для загального населення, та корпоративні бібліотеки для організації внутрішньої документації організацій [3].

За моделлю доступу електронні бібліотеки можуть бути відкритими, надаючи безкоштовний доступ до всього або більшої частини свого контенту у відповідності з принципами відкритого доступу, комерційними, що функціонують на основі платного доступу або моделі підписки, та змішаними, що поєднують безкоштовний доступ до частини контенту з платним доступом до розширених функцій або ексклюзивних матеріалів [3].

До специфічних ознак електронних бібліотек відносяться можливість містити у своїх фондах різноманітні колекції оцифрованих та цифровізованих творів, функціонування як комунікаційної системи, що забезпечує збирання, пошук, обробку та пересилання інформації, орієнтація на доволі широке коло

користувачів, можливість створення як частини традиційної бібліотеки, так і окремо від неї [3].

Електронні бібліотеки повинні відповідати основним критеріям, таким як системність в організації електронних документів, наявність метаданих та можливість ефективного пошуку. Функціональна структура електронної бібліотеки включає інформаційний фонд, що складається з електронних документів, програмно-технологічний комплекс для обробки інформації, пристрої поширення інформації та користувацькі сервіси, пристрої для зберігання даних та підсистему управління і ведення електронної бібліотеки [3].

До електронних ресурсів сучасних бібліотек відносять електронні каталоги та бази даних, що містять метаінформацію про бібліотечні ресурси і дають змогу шукати матеріали за автором, назвою, темою, ключовими словами чи іншими критеріями. Електронні бібліотеки представляють собою інформаційну систему, яка об’єднує впорядкований фонд цифрових ресурсів, каталог, а також апаратно-програмний комплекс, що забезпечує стабільну роботу пошукової системи, дозволяє оперативно поповнювати колекцію, реєструвати матеріали, забезпечувати їх тривале зберігання та видавати доступ до розповсюдження [4].

Перша електронна бібліотека в Інтернеті була створена в 1971 році й отримала назву «Ґутенберґ» на честь Йоганна Ґутенберга, винахідника друкарського верстата. Сьогодні найбільшою електронною бібліотекою є Всесвітня цифрова бібліотека, урочисте відкриття якої відбулося 21 квітня 2009 року. Ініціатором цього масштабного проекту стала Бібліотека Конгресу США [4].

В епоху розвитку інформаційних технологій найвідомішими цифровими бібліотеками є Google Books, Europeana, Open Library, що включає понад мільйон матеріалів світовими мовами, World Digital Library як міжнародний цифровий ресурс, створений за підтримки ЮНЕСКО та Бібліотеки Конгресу США, Національна бібліотека України імені Вернадського та інші [4].

Репозиторії представляють собою цифрові архіви, де зберігаються наукові праці, дисертації, звіти, дані досліджень, створені співробітниками або

студентами. Бази даних являють собою спеціалізовані платформи для пошуку та доступу до наукових статей, звітів та іншої необхідної користувачеві інформації. Електронні періодичні видання включають цифрові журнали, газети, бюлетені, доступні у форматі PDF, що дуже зручно для перегляду сучасних наукових і популярних публікацій [4].

Мультимедійні ресурси включають аудіо та відео документи, довідково-інформаційні ресурси представляють онлайн-енциклопедії, словники, довідники, буклети, путівники. Віртуальні читальні зали це платформи, які використовують користувачі для отримання доступу до матеріалів без відвідування книгозбірні. Інтегровані бібліотечні системи представляють програмне забезпечення для управління бібліотечними фондами, обліку користувачів і обслуговування [4].

Важливо розрізняти електронні бібліотеки та автоматизовані системи традиційних бібліотек. Хоча широко поширена думка, що електронні бібліотеки є автоматизованою системою традиційних бібліотек, це твердження є дещо неточним. Різниця полягає у різних видах носіїв інформаційної продукції, технологічних засобах комплектування, характері поставлених цілей і завдань з обслуговування користувачів, організаційних формах створення та функціонування, різноманітних формах власності на інформаційну продукцію та різних методах доступу користувачів до ресурсів [3].

Зазначені електронні ресурси мають значні переваги, до яких, в першу чергу, відносимо доступність, оперативність, економічність, зручність пошуку [4]. За думкою багатьох бібліотекарів, електронні видання мають значні переваги перед традиційними носіями інформації, оскільки електронний варіант може бути збережений необмежений час і використовуватися без обмежень при періодичному перезаписувані на нові носії [3].

1.2 Огляд існуючих систем електронних бібліотек

Цифрова ера відкрила нові можливості для книголюбів – тепер цілі бібліотеки можна носити в кишені. Електронні бібліотеки та спеціалізовані застосунки для читання книг дозволяють отримати доступ до величезної кількості літератури в будь-який час та в будь-якому місці. Особливо актуальним це стало для українських читачів, адже за останні роки з’явилося багато якісних ресурсів з україномовною літературою.

PocketBook Reader – універсальний безкоштовний застосунок, який підтримує понад 26 форматів файлів, включаючи популярні FB2, EPUB, PDF, DJVU, MOBI, TXT, DOCX та інші [5]. Крім електронних книг, додаток дозволяє працювати з журналами, коміксами та аудіокнигами. PocketBook Reader має вбудовану озвучку тексту голосом, словники та перекладач, можливість робити закладки та нотатки. Всі матеріали зберігаються у безкоштовній хмарі PocketBook Cloud, а синхронізація доступна через Dropbox, Google Books та Google Drive [6]. Додаток містить великий магазин, де можна купити онлайн майже будь-які книги, навіть найновіші. Користувачі високо оцінюють зручність, інтуїтивно зрозумілий інтерфейс та широкі можливості налаштування для комфортного читання.

Moon+Reader – один із найпопулярніших додатків для читання електронних книг з підтримкою 40 мов та великої кількості форматів (EPUB, DJVU, PDF, FB2, DOCX, ODT, RTF, TXT, HTML та інші) [6]. Програма має зручний інтерфейс з можливістю налаштування візуальних параметрів – форматування тексту, вибір шрифтів, фонових кольорів і текстур, встановлення теми і денного-нічного режиму, налаштування анімації гортання і скроллінгу сторінок. Для безпечного читання застосовується фільтр синього кольору. Позиції читання синхронізуються між пристроями через DropBox та WebDav. Існує також платна версія Moon+Reader Pro з розширеними функціями, без реклами та з додатковими можливостями для роботи з PDF-документами.

Librera Reader – універсальна читалка, яка підтримує понад 12 текстових форматів (PDF, EPUB, FB2, MOBI, DJVU, TXT, RTF, AZW, HTML, XPS, TIFF та інші) [6]. Це простий і мінімалістичний застосунок, який займає мало місця, швидко працює та має сучасний дизайн. Librera Reader автоматично сканує файли на пристрої та формує бібліотеку з сортуванням за назвою, автором, жанром. Додаток має вбудований файловий менеджер та опцію конвертації документів в інший формат. Є інструменти для роботи з PDF-документами – на сторінках можна робити ескізи та переводити файли у текстовий формат TXT. Вбудований інструмент для читання нот з автоматичною прокруткою нотного аркуша. Для тих, хто вивчає іноземні мови, будуть корисні опції озвучки голосом і перекладу слів.

Google Play Книги – зручний і універсальний сервіс від Google, що поєднує магазин і рідер [5]. Дозволяє купувати книги, завантажувати безкоштовні видання або додавати власні файли у форматах EPUB та PDF. У додатку можна налаштовувати розмір шрифту, фон, яскравість екрану та вмикати нічний режим для комфортного читання. Для аудіокниг доступний зручний плеєр із регулюванням швидкості та таймером сну. Всі книги й прогрес синхронізуються між пристроями через Google-акаунт.

Apple Books – для власників iPhone, iPad і Mac застосунок для читання книг Apple Books стане справжнім відкриттям [5]. Вишуканий дизайн і плавна навігація створюють справжню естетику читання. У Apple Books можна купувати книги або завантажувати безкоштовні прямо з вбудованого магазину, де є художня література, нон-фікшн, навчальні посібники та аудіокниги. Застосунок дозволяє налаштовувати вигляд сторінок – змінювати шрифт, колір фону, розмір тексту чи вмикати нічний режим для комфортного читання. Прогрес синхронізується через iCloud.

Amazon Kindle – еталонний додаток для читання книг, який синхронізується з однойменними пристроями та хмарним сервісом Amazon [5]. Це означає, що можна почати читати книгу на смартфоні, продовжити на планшеті чи рідері Kindle – і додаток запам’ятає, на якій сторінці зупинився

читач. У Kindle можна купувати книги, завантажувати безкоштовні видання або додавати власні файли. Є функції налаштування шрифтів, фону, режиму читання та навіть перекладач і словники прямо в тексті.

Wattpad – це не просто додаток для читання, а соціальна платформа, яка об'єднує понад 100 мільйонів користувачів – читачів та авторів зі всього світу [7]. Тут можна знайти твори різних форматів і жанрів більш ніж 50 мовами світу. Читачі формують бібліотеки улюблених книг, читають онлайн чи офлайн, спілкуються з однодумцями та авторами творів, залишають коментарі та відгуки. Застосунок має комфортний інтерфейс, добре структурований каталог, зрозумілу навігацію та всі необхідні функції для зручного читання. Мільйони сучасних творів доступні безкоштовно.

Goodreads – це більше, ніж просто додаток для читання [7]. Тут можна вести власний читацький щоденник, обговорювати книги з друзями та відкривати нові шедеври за рекомендаціями. Кожен користувач створює власну книжкову полицю, додає прочитане, планує майбутні читання і фіксує, що зачепило чи розчарувало. Тут полювання за новими книгами перетворюється на цікаву гру: рекомендації алгоритмів змішуються з порадами друзів, а книжкові клуби знаходять однодумців у різних куточках планети.

Librarius – український мобільний застосунок для онлайн та офлайн читання електронних книг [6]. Окрім світової літератури, на платформі представлено широкий вибір ліцензованих українських книжок. База оновлюється, додаються новинки та бестселери. Книжки доступні на різних умовах: деякі можна читати безкоштовно, деякі – купити за доступною ціною. Є компромісний варіант – недорого орендувати електронну книгу у Librarius на 2 тижні. Перед покупкою чи орендою можна безкоштовно прочитати 10% обсягу твору. Додаток має розумний пошук – знайти бажану книгу можна навіть за неточними ключовими запитами.

Rork – амбітний український стартап, який допомагає організувати процес читання та стежити за власним прогресом [7]. У ньому можна фіксувати, скільки часу читаєш щодня, бачити статистику та планувати, коли завершиш книгу.

Додаток дозволяє зберігати цитати, робити нотатки та створювати власні книжкові списки. Rork більше не дасть книгам загубитися – усе впорядковано в одному місці. Завдяки простому інтерфейсу легко контролювати читання та поступово вибудовувати стабільну звичку повертатися до книжки щодня.

Абук – перша українська аудіокнигарня, яка назбирала великий архів популярних текстів, якісно озвучених професійними дикторами або авторами [7]. Додаток пропонує можливість створювати власні полиці, додавати закладки та робити помітки у прослуханих книгах. Нещодавно платформа розширилась і до електронних книг.

Basmo – застосунок для відстеження читання з особливістю: він спонукає користувача поставити мету й допомагає досягти її [7]. Basmo пропонує відстежування прогресу, збереження бібліотеки та статистики. Частина функцій доступна у преміум-версії за окрему плату.

1.3 Анадіз вимог до системи електронної бібліотеки

Проведений огляд існуючих рішень дозволяє сформулювати комплексний набір вимог до сучасної системи електронної бібліотеки. Ці вимоги можна розділити на функціональні, що описують конкретні можливості, які повинна надавати система, та нефункціональні, що визначають якісні характеристики системи.

1.3.1 Функціональні вимоги до системи

Система управління контентом повинна забезпечувати повний життєвий цикл електронних книг у бібліотеці. Адміністратори та користувачі з відповідними правами повинні мати можливість додавати нові книги до системи через завантаження файлів або імпорт з інших джерел. Під час додавання книги система повинна автоматично витягувати метадані з файлу, якщо вони доступні, або надавати зручний інтерфейс для їх введення вручну. Метадані повинні включати не лише базову інформацію, таку як назва, автор, рік видання та ISBN,

але й розширені дані, включаючи жанр, серію, опис, мови, мітки, рейтинг та рецензії.

Однією з основних функцій електронної бібліотечної системи є надання можливості користувачам реєструватися та входити в систему. Система повинна дозволяти користувачам створювати обліковий запис, вказуючи своє ім’я, адресу електронної пошти та пароль. Після реєстрації користувачі повинні мати можливість увійти в систему, використовуючи свою адресу електронної пошти та пароль [8].

Система повинна підтримувати ієрархічну організацію контенту через категорії та колекції. Категорії можуть бути організовані у вигляді дерева з підкатегоріями, що дозволяє створювати складні таксономії. Колекції надають альтернативний спосіб групування книг за довільними критеріями, незалежно від формальної категоризації. Одна книга може одночасно належати до кількох категорій та колекцій.

Управління обкладинками книг має особливе значення для користувацького досвіду. Система повинна автоматично витягувати обкладинки з файлів книг, якщо вони доступні, або намагатися знайти їх в онлайнджерелах за метаданими книги. Користувачі також повинні мати можливість завантажувати власні зображення обкладинок. Система має зберігати обкладинки у кількох розмірах для оптимізації швидкості завантаження на різних пристроях.

Функціональність пошуку та навігації має бути достатньо потужною для ефективної роботи з великими бібліотеками. Система повинна дозволяти користувачам шукати книги в бібліотеці за назвою, автором або ключовим словом. Функція пошуку повинна бути інтуїтивно зрозумілою і зручною, щоб користувачі могли легко знаходити книги, які вони шукають [8]. Повнотекстовий пошук повинен індексувати не лише метадані книг, але за можливості й сам вміст книг, особливо для текстових форматів.

Система повинна підтримувати складні пошукові запити з булевими операторами, фразовий пошук та пошук з урахуванням морфології для кращої

релевантності результатів. Фасетна навігація дозволяє користувачам поступово звужувати результати пошуку, застосовуючи фільтри за різними критеріями, такими як автор, рік видання, жанр, мова та рейтинг.

Система повинна дозволяти користувачам виписувати книги з бібліотеки. Коли користувач виписує книгу, система повинна відстежувати термін її повернення і надсилати нагадування користувачеві до того, як він її отримає. Коли користувач повертає книгу, система повинна оновити статус книги в інвентарі бібліотеки [8].

Користувачі повинні мати можливість резервувати книги, які в даний момент виписані іншими користувачами. Коли зарезервована книга стає доступною, система повинна повідомити про це користувача, який її зарезервував, і утримувати книгу протягом певного періоду часу.

Компонент читання електронних книг є центральним для користувацького досвіду системи. Система повинна підтримувати найпопулярніші формати електронних книг, включаючи EPUB для рефлоу-контенту, PDF для документів з фіксованим макетом, FB2 як популярний у Східній Європі формат та MOBI для сумісності з пристроями Kindle. Для кожного формату має бути реалізований оптимізований рендерер, що враховує специфічні особливості формату.

Налаштування відображення тексту повинні надавати користувачам максимальну гнучкість для створення комфортного читацького досвіду. Це включає вибір шрифту з набору попередньо встановлених шрифтів та

можливість завантаження власних шрифтів, налаштування розміру шрифту з підтримкою динамічного масштабування, регулювання інтерліньяжу для покращення читабельності, налаштування відступів та вирівнювання тексту, вибір кольорової схеми включаючи світлу, темну та сепію теми.

Система повинна дозволяти користувачам керувати своїми профілями, в тому числі оновлювати особисту інформацію, переглядати історію позик та керувати бронюванням книг [8]. Функціонал закладок та виділення тексту дозволяє користувачам позначати важливі місця в тексті для майбутнього референсу.

Користувачі повинні мати можливість залишати відгуки та оцінки прочитаних книг. Система повинна відображати ці відгуки та оцінки на сторінці книги, дозволяючи іншим користувачам бачити, що інші думають про книгу.

Система повинна включати адміністративну панель, яка дозволяє адміністратору керувати інвентарем бібліотеки, переглядати історію користування та керувати обліковими записами користувачів. Панель також повинна надавати аналітику про використання бібліотеки, включаючи найпопулярніші книги та кількість книг, взятих за певний період часу.

Синхронізація між пристроями є критично важливою функцією для забезпечення безшовного досвіду при використанні системи на різних платформах. Позиція читання повинна автоматично синхронізуватися в реальному часі або з мінімальною затримкою, щоб користувач міг продовжити читання з того самого місця на будь-якому пристрої.

Система повинна мати можливість інтегруватися із зовнішніми системами, такими як онлайн-платіжні шлюзи та постачальники книг, щоб забезпечити безперебійну роботу користувачів та ефективно управляти бібліотечним інвентарем [8].

1.3.2 Нефункціональні вимоги до системи

Продуктивність системи безпосередньо впливає на користувацький досвід і є одним з найважливіших нефункціональних критеріїв. Час завантаження сторінок інтерфейсу не повинен перевищувати двох секунд при нормальних умовах мережі. Це включає як початкове завантаження додатку, так і навігацію між різними розділами системи.

Прокрутка та перегортання сторінок повинні бути абсолютно плавними, без затримок або пропусків кадрів. Це вимагає ретельної оптимізації рендерингу та використання апаратного прискорення де це можливо. Частота кадрів під час анімацій має підтримуватися на рівні не менше шістдесяти кадрів за секунду для забезпечення природного відчуття від взаємодії з інтерфейсом.

Масштабованість системи визначає її здатність ефективно функціонувати при зростанні обсягів даних та кількості користувачів. Підтримка бібліотек з

тисячами книг вимагає ефективної індексації та пошуку. База даних повинна бути спроєктована з урахуванням можливості швидкого виконання запитів навіть при великих обсягах метаданих.

Надійність системи визначає її здатність стабільно функціонувати в різних умовах та відновлюватися після збоїв. Цільовий показник доступності системи на рівні дев'яносто дев'ять цілих дев'ять десятих відсотка часу означає, що допустимий час простою не повинен перевищувати приблизно вісім годин на рік.

Безпека системи охоплює захист даних користувачів, контенту бібліотеки та самої інфраструктури від несанкціонованого доступу та зловмисних дій. Захист персональних даних користувачів вимагає відповідності сучасним стандартам. Безпечна передача даних між клієнтом та сервером повинна обов'язково використовувати протокол HTTPS з сучасними криптографічними алгоритмами.

Юзабіліті або зручність використання визначає наскільки легко та приємно користувачам працювати з системою. Інтуїтивний інтерфейс користувача означає, що навіть нові користувачі можуть швидко зрозуміти як працювати з системою без необхідності читання довгих інструкцій.

Кросплатформеність системи означає її здатність функціонувати на різних операційних системах та пристроях з єдиною кодовою базою або мінімальними відмінностями. Підтримка сучасних веб-браузерів повинна охоплювати всі популярні браузери, включаючи Google Chrome, Mozilla Firefox, Safari, Microsoft Edge та їх мобільні версії.

1.3.3 Вимоги до архітектури бази даних

Проектування архітектури бази даних є важливим кроком у створенні електронної бібліотеки. База даних – це центральний компонент, який зберігає всю інформацію про книги, авторів, користувачів та їхні взаємовідносини. База даних повинна бути спроектована таким чином, щоб бути ефективною, масштабованою і простою в обслуговуванні [8].

Є чотири основні сутності: книги, автори, користувачі та транзакції. Сутність книги має зв'язок багато-добагатьох з авторами, що означає, що книга

може мати кількох авторів, а автор може написати кілька книг. Сутність транзакції має зв’язок багато-до-одного як з книгами, так і з користувачами, що означає, що користувач може позичити кілька книг, а книга може бути позичена кількома користувачами.

Після визначення сутностей та їх зв’язків, наступним кроком є визначення атрибутів та типів даних для кожної сутності. Атрибут – це характеристика сутності, наприклад, назва книги або ім’я автора. Тип даних визначає тип даних, які можуть зберігатися в атрибуті, наприклад, рядок, ціле число або дата.

Сутність books може мати такі атрибути: ідентифікатор, назва, опис, дата публікації, ISBN, зображення обкладинки, жанр. Аналогічно, сутність author може мати атрибути: ID, ім’я, біографія, зображення профілю. Типи даних для кожного атрибуту залежатимуть від вимог програми та системи управління базами даних, що використовується [8].

Нормалізація – це процес організації структури бази даних таким чином, щоб зменшити залежність. Це допомагає поліпшити цілісність даних, зменшити вимоги до зберігання даних і підвищити продуктивність запитів. Існують різні рівні нормалізації, починаючи від першої нормальної форми до п’ятої нормальної форми. Кожен рівень нормалізації має свій власний набір правил і рекомендацій.

Для електронної бібліотеки база даних повинна бути нормалізована принаймні до третьої нормальної форми. Це означає, що кожна таблиця повинна мати первинний ключ, а кожен неключовий атрибут повинен залежати тільки від первинного ключа.

Індексування та розбиття на розділи – це дві стратегії оптимізації продуктивності бази даних. Індексування передбачає створення індексів на одному або декількох стовпцях таблиці, що дозволяє базі даних швидко шукати певні дані. Розбиття на розділи передбачає поділ великої таблиці на менші розділи на основі певного критерію, наприклад, дати або географічного розташування [8].

1.3.4 Вимоги до інтерфейсу користувача

Інтерфейс користувача є частиною програмної системи, з якою взаємодіють користувачі. Метою дизайну інтерфейсу користувача є створення зручного та інтуїтивно зрозумілого інтерфейсу, який дозволяє користувачам легко та ефективно орієнтуватися в системі. У випадку електронної бібліотеки інтерфейс користувача відіграє вирішальну роль у забезпеченні користувачам легкого доступу до ресурсів і функцій бібліотеки.

Розробка інтерфейсу користувача для електронної бібліотеки включає в себе кілька етапів, включаючи визначення вимог користувача, створення каркасів і макетів і реалізацію остаточного дизайну за допомогою HTML, CSS і JavaScript [8].

Першим кроком у розробці інтерфейсу користувача є визначення вимог користувача. Це передбачає визначення цільової аудиторії, її потреб і переваг, а також завдань, які вони виконуватимуть у системі. Цільова аудиторія електронної бібліотеки може включати студентів, дослідників та інших осіб, яким потрібен доступ до академічних ресурсів.

Щоб визначити вимоги користувачів, дизайнери можуть проводити дослідження користувачів, такі як опитування та інтерв’ю, щоб зібрати відгуки та ідеї від потенційних користувачів. Потім цю інформацію можна використовувати для створення персонажів користувачів, які є вигаданими представленнями різних типів користувачів та їхніх характеристик [8].

Після визначення вимог користувача дизайнери можуть створювати каркаси та макети для візуалізації дизайну інтерфейсу користувача. Каркаси – це базові ескізи, які окреслюють макет і функціональність інтерфейсу, тоді як макети – це більш детальні представлення, які включають візуальні елементи, такі як кольори, типографіка та зображення.

Каркаси та макети дозволяють дизайнерам тестувати різні концепції дизайну та збирати відгуки від зацікавлених сторін, перш ніж вкладати час і ресурси в розробку остаточного дизайну. Вони також допомагають

переконатися, що інтерфейс користувача відповідає вимогам користувача, є інтуїтивно зрозумілим і простим у використанні.

Після затвердження каркасів і макетів остаточний дизайн можна реалізувати за допомогою HTML, CSS і JavaScript. HTML використовується для структурування вмісту інтерфейсу користувача, тоді як CSS використовується для стилізації інтерфейсу користувача та надання йому візуальної привабливості. JavaScript використовується для додавання інтерактивності та динамічних функцій інтерфейсу користувача, таких як спадні меню та поля для пошуку [8].

На етапі впровадження дизайнери також повинні переконатися, що інтерфейс користувача адаптований до різних розмірів екрана та пристроїв. Це особливо важливо для електронної бібліотеки, оскільки користувачі можуть отримати доступ до системи з різних пристроїв.

1.3.5 Вимоги до серверної частини системи

Архітектура клієнт-сервер є концепцією інформаційної мережі, в якій основна частина її ресурсів зосереджена в серверах, що обслуговують своїх клієнтів. Ця архітектура визначає два типи компонентів: сервери і клієнти [8].

Сервер – це об’єкт, що надає сервіс іншим об’єктам мережі за їх запитами. Сервіс – це процес обслуговування клієнтів. Сервер працює за завданнями клієнтів і управляє виконанням їх завдань. Після виконання кожного завдання сервер посилає отримані результати клієнту, який послав це завдання.

Процес, який викликає сервісну функцію за допомогою певних операцій, називається клієнтом. Ним може бути програма або користувач. Клієнти – це робочі станції, які використовують ресурси сервера і надають зручні інтерфейси користувача.

У сучасній клієнт-серверній архітектурі виділяється чотири групи об’єктів: клієнти, сервери, дані і мережеві служби. Клієнти розташовуються в системах на робочих місцях користувачів. Дані в основному зберігаються в серверах. Мережеві служби є спільно використовуваними серверами і даними. Крім того служби керують процедурами обробки даних [8].

Мережі клієнт-серверної архітектури мають наступні переваги: дозволяють правильно організовувати мережі з великою кількістю робочих станцій; забезпечують централізоване управління обліковими записами користувачів, безпекою та доступом, що спрощує мережне адміністрування; ефективний доступ до мережевих ресурсів; користувачеві потрібен один пароль для входу в мережу і для отримання доступу до всіх ресурсів, на які поширюються права користувача.

Поряд з перевагами мережі клієнт-серверної архітектури мають і ряд недоліків: несправність сервера може зробити мережу непрацездатною або призвести до втрати мережевих ресурсів; вимагають кваліфікованого персоналу для адміністрування; мають вищу вартість мереж і мережевого обладнання [8].

РОЗДІЛ 2
ТЕОРЕТИЧНІ ОСНОВИ КРОСПЛАТФОРМЕНОЇ РОЗРОБКИ

2.1 Підходи до кросплатформеної розробки мобільних та веб-додатків

Проблема кросплатформеної розробки існувала з самого початку епохи персональних комп'ютерів, коли різні виробники створювали несумісні між собою системи. З появою смартфонів та планшетів ця проблема набула ще більшої актуальності, оскільки розробникам потрібно підтримувати не лише різні операційні системи, але й величезне різноманіття розмірів екранів, роздільних здатностей та апаратних можливостей пристроїв.

Вибір правильної архітектури є ключовим у створенні успішного програмного забезпечення. Архітектура додатка визначає план його структури та організації, визначає як різні компоненти додатка взаємодіятимуть один з одним, і відіграє ключову роль у його продуктивності, масштабованості та зручності обслуговування [9]. Важливо не плутати архітектуру програми з дизайном програмного коду, оскільки архітектура є вищим рівнем проєктування, який фокусується на загальних компонентах та їхній взаємодії.

Нативний підхід до розробки мобільних додатків передбачає створення окремих версій програми для кожної цільової платформи з використанням специфічних для платформи інструментів, мов програмування та фреймворків. Для платформи iOS компанія Apple з самого початку запропонувала мову програмування ObjectiveC, а у 2014 році представила нову мову Swift. Для платформи Android основною мовою програмування спочатку була Java, а у 2017 році компанія Google офіційно анонсувала підтримку мови Kotlin як офіційної мови для розробки Android-додатків.

Нативна розробка має суттєві переваги, які роблять її привабливою для певних типів додатків. Максимальна продуктивність досягається завдяки прямому доступу до апаратних ресурсів пристрою та відсутності додаткових шарів абстракції між кодом додатку та операційною системою. Повний доступ

до платформо-специфічних API дозволяє нативним додаткам використовувати всі можливості операційної системи та апаратного забезпечення.

Проте нативна розробка має й суттєві недоліки, які стимулюють пошук альтернативних підходів. Необхідність підтримки кількох окремих кодових баз є найбільшим викликом нативної розробки. Кожна платформа вимагає не лише окремого коду, написаного різними мовами програмування, але й різних підходів до архітектури, організації проекту та тестування. Високі витрати на розробку є прямим наслідком необхідності підтримки кількох кодових баз.

Гібридний підхід до розробки мобільних додатків виник як спроба вирішити проблеми нативної розробки, дозволяючи розробникам використовувати знайомі веб-технології для створення додатків, що працюють на різних платформах. Основна ідея полягає в тому, що додаток створюється за допомогою HTML, CSS та JavaScript, а потім упаковується в нативний контейнер, який надає доступ до можливостей пристрою через JavaScript API.

Кросплатформені нативні фреймворки представляють еволюційний крок у розвитку підходів до мобільної розробки, намагаючись поєднати переваги нативної та гібридної розробки [9]. Ці фреймворки дозволяють розробникам писати код один раз, який потім або компілюється в нативний код, або виконується таким чином, що використовує справжні нативні компоненти інтерфейсу.

2.2 Архітектурні патерни для побудови складних клієнтських додатків

Архітектурні патерни у розробці програмного забезпечення являють собою перевірені часом рішення типових проблем проектування. Для клієнтських додатків, особливо складних систем як електронні бібліотеки, вибір правильного архітектурного патерну критично важливий для довгострокової підтримки та розвитку системи.

При розробці клієнт-серверного додатку, який претендує на довготривале існування, дуже важливо вибрати архітектурний патерн, який повністю

задовільнить всі потреби команди та самого продукта. Шаблони проектування програмного забезпечення є ефективними способами вирішення задач проектування. Шаблон не є закінченим зразком, який можна безпосередньо транслювати в програмний код, а об’єктно-орієнтований шаблон найчастіше є зразком вирішення проблеми і відображає відношення між класами та об’єктами [8].

2.2.1 Model-View-Controller та його місце в історії

Model-View-Controller є одним з найстаріших та найвпливовіших архітектурних патернів у розробці програмного забезпечення. Патерн MVC розділяє застосунок на три компоненти: модель, яка відповідає за зберігання даних, представлення, що забезпечує відображення даних користувачеві, і контролер, який управляє взаємодією між моделлю та поданням [9].

Найпростіший паттерн це MVC, який компанія Apple рекомендує використовувати для архітектурно простих додатків. Model – це місце, де перебувають дані програми, включаючи об’єкти моделей, парсери, менеджери та мережевий код. View являє собою обличчя програми, її класи часто використовуються для багаторазового використання, оскільки не містять певної логіки [10].

Controller опосередковує погляд і модель за допомогою шаблону делегування [10]. В ідеальному сценарії суб’єкт контролера не знатиме конкретного погляду, з яким він має справу, натомість він буде спілкуватися з абстракцією за допомогою протоколу.

У класичному патерні MVC компонент Model відповідає за дані додатку та бізнес-логіку. Model не знає нічого про те, як дані відображаються користувачу, він просто надає інтерфейс для роботи з даними та повідомляє про зміни через систему подій або спостерігачів. View відповідає за відображення даних користувачу, отримуючи дані від Model і рендеруючи їх у формі, зрозумілій для користувача.

Проте класичний MVC має проблеми при адаптації до сучасної веб та мобільної розробки. Найбільша проблема полягає в тому, що Controller часто

стає надто великим і відповідає за забагато різних речей. Тестування також стає складним, оскільки Controller часто має залежності і від View, і від Model, що робить його важким для ізольованого тестування.

2.2.2 Model-View-ViewModel як еволюція для сучасних додатків

MVVM заснований на розділенні інтерфейсу користувача та бізнес-логіки з використанням проміжного шару ViewModel, що надає дані та методи для їх обробки [9]. Основна інновація MVVM полягає у введенні проміжного шару ViewModel між Model та View.

З розвитком реактивного програмування все більше набувають популярності патерни, в основу яких лежить патерн Спостерігач, одним з таких є MVVM. Суть цього патерна полягає в тому, що компоненти не взаємодіють між собою напряму, а реагують на зміну ViewModel. Модель уявлення пов’язує модель і уявлення через механізм прив’язки даних [10].

ViewModel це абстракція View, яка надає дані та команди у формі, зручній для відображення. На відміну від Controller у MVC, ViewModel не має прямих посилань на View і взаємодіє з ним виключно через механізм зв’язування даних. Якщо в моделі змінюються значення властивостей, при реалізації моделлю інтерфейсу автоматично йде зміна відображуваних даних в інших компонентах [10].

Двостороннє зв’язування даних є ключовою особливістю, що робить MVVM потужним. Коли користувач вводить текст у поле пошуку, це автоматично оновлює відповідну властивість у ViewModel. Коли ViewModel оновлює список результатів пошуку, View автоматично перерендерює список.

Переваги MVVM особливо помітні у складних додатках. Відмінна тестованість досягається завдяки повному відокремленню бізнес-логіки від UI. ViewModel може бути протестований без необхідності рендерингу реального інтерфейсу, що робить тести швидкими та надійними.

2.2.3 Model-View-Presenter як альтернативний підхід

MVP є схемою розробки, що складається з таких основних компонентів як Model, View та Presenter. Патерн MVP схожий на MVC, але з активнішою роллю презентера, який керує взаємодією між моделлю та представленням [10].

При реалізації цього патерну View взагалі немає ніякої логіки і відповідає лише за відображення. Різниця полягає в тому, що Presenter буде відповідати за всю логіку, включаючи реагування на дії користувача та оновлення інтерфейсу користувача через делегата. Найголовніше полягає в тому, що ведучий не буде залежати від View, це означає, що він добре ізольований і тому легко тестується [10].

Важливо зазначити, що MVP використовує пасивний режим перегляду, це означає, що всі дії будуть передані Presenter, який запустить оновлення інтерфейсу користувача за допомогою делегатів [10]. Таким чином, перегляд буде лише проходити дії та слухати оновлення ведучого.

MVP надає чіткіше розділення відповідальності порівняно з MVC, де Controller часто стає надто великим. Presenter інкапсулює презентаційну логіку, роблячи її легко тестованою незалежно від View. Це особливо корисно для складних інтерфейсів з багатьма елементами управління та складною логікою відображення.

2.3 Архітектурні підходи до організації додатків

Архітектура застосунку складається з декількох ключових компонентів, які взаємодіють один з одним і тим самим забезпечують його роботу як єдиного цілого [9]. Користувацький інтерфейс визначає те, як користувач взаємодіє із застосунком через елементи інтерфейсу. Бізнес-логіка визначає правила та процеси, що керують роботою додатка та обробкою даних.

Модель даних визначає структуру даних, що використовуються в застосунку, включно із сутностями, атрибутами та їхніми взаємозв’язками [9].

Інфраструктурні аспекти включають сервери, бази даних, інструменти розроблення та інші ресурси, необхідні для роботи й підтримки програми.

2.3.1 Мікросервісна архітектура

Мікросервісна архітектура є підходом до розроблення застосунків, за якого застосунок розбивається на невеликі та незалежні сервіси [9]. Кожен із них відповідає за певну функціональність або бізнес-завдання і може бути розроблений, розгорнутий і масштабований окремо від інших сервісів.

Переваги мікросервісної архітектури включають поліпшену масштабованість, гнучкість розгортання та оновлення, а також легкість у розумінні та підтримці коду [9]. Однак також є недоліки, зокрема складнощі в управлінні розподіленими системами, що вимагає додаткових інструментів для моніторингу та забезпечення надійності, мережеві затримки, а також складність розробки й тестування.

Платформи електронної комерції можуть використовувати мікросервіси для розділення функціональності онлайн-магазинів, такої як управління товарами, обробка замовлень і оплати, для підвищення ефективності та гнучкості. Соціальні мережі застосовують мікросервіси для управління даними користувачів, контентом, автентифікацією та опрацюванням повідомлень, забезпечуючи швидку та масштабовану роботу платформи [9].

Фінансові додатки використовують мікросервіси для забезпечення обробки транзакцій, управління рахунками, надання звітності та аналітики, що гарантує безпеку та надійність операцій. Кожен мікросервіс може бути розроблений різними командами з використанням найбільш підходящих технологій для конкретної задачі.

2.3.2 Монолітна архітектура

Монолітна архітектура є класичним підходом до розроблення застосунків, за якого весь його функціонал розміщується в єдиному кодовому сховищі. Характеристики включають єдиний кодовий базис, який містить у собі всю функціональність програми, інтеграцію всіх компонентів застосунку в одну

систему, вертикальне масштабування через збільшення ресурсів сервера, одну базу даних для всієї програми [9].

Моноліти можуть бути розгорнуті на одному сервері або віртуальній машині. Сценарії застосування включають маленькі та середні проєкти, де відсутня необхідність у складній масштабованості, проєкти з обмеженими ресурсами, де вартість розроблення та підтримки мікросервісної архітектури занадто висока, а також прототипування і початкову стадію розробки програми для швидкого запуску.

Мікросервісна архітектура має більшу гнучкість і масштабованість, але вимагає складнішого управління залежностями між сервісами [9]. У той час як монолітна архітектура простіша в розробці та управлінні, але обмежена у своїй здатності масштабуватися й адаптуватися до мінливих вимог.

2.3.3 Багаторівнева архітектура

Багаторівнева архітектура, також відома як n-рівнева архітектура, є структурою застосунку, в якій функціональність поділяється на кілька рівнів або шарів [9]. Кожен рівень виконує певні завдання і має свою сферу відповідальності.

Поділ відповідальності між різними шарами додатка дає змогу досягти чіткості і модульності, оскільки кожен рівень виконує певні функції, що робить код більш структурованим і легким для розуміння та підтримки. Гнучкість і масштабованість досягаються завдяки можливості зміни або заміни кожного рівня незалежно від інших, що робить простішим внесення змін та адаптацію додатків під нові вимоги.

Окремі шари можна повторно використати в різних частинах додатка або в інших проєктах, що економить час і зусилля під час розроблення. Добре визначені межі між шарами дають змогу контролювати доступ до даних і функцій, що підвищує безпеку застосунку [9].

2.4 Управління станом у кросплатформених додатках

Управління станом є критично важливим аспектом розробки сучасних додатків, особливо у контексті кросплатформеної розробки. Стан додатку включає всі дані, що визначають поточний стан інтерфейсу користувача, дані користувача, результати мережевих запитів та інші змінні параметри.

Централізоване сховище стану з однонаправленим потоком даних забезпечує передбачуваність поведінки додатку. Концепції включають єдине джерело істини, описи подій через дії, чисті функції для обробки дій та зміни стану, а також обчислені значення на основі стану через селектори.

Реактивне управління станом на основі спостережуваних об’єктів передбачає, що стан є спостережуваним, будьякі зміни автоматично викликають оновлення UI, що вимагає мінімум шаблонного коду. Принципи включають природність та простоту реалізації, хоча це може призводити до меншої передбачуваності та складнішого відлагодження.

2.5 Розробка архітектури застосунку з нуля

При створенні архітектури додатка з нуля важливо дотримуватися систематичного підходу [11]. Необхідно вивчити і визначити основні цілі, функції та потреби користувачів застосунку, дослідити різні технології та інструменти, враховуючи вимоги до продуктивності, масштабованості та безпеки проєкту.

Розділення функціональності на невеликі, незалежні модулі збільшує гнучкість та підтримуваність системи. Визначення структури та відношень в базі даних має враховувати вимоги до зберігання та обробки даних. Проведення дослідження ринку і вивчення деталей проєкту допомагає краще зрозуміти потреби користувачів.

Використання принципів SOLID сприяє створенню гнучкої та розширюваної архітектури [9]. Проведення функціонального, модульного та

інтеграційного тестування переконує в коректності роботи всіх компонентів. Розробка архітектури застосунку і додавання нових функцій з використанням ітеративного підходу дозволяє поступове поліпшення системи.

Аналіз архітектури та документування застосунку відіграють важливу роль у поліпшенні його продуктивності та забезпеченні зручності підтримки. Методи аналізу включають огляд коду для виявлення проблем і поліпшень в архітектурі програми, статичний аналіз коду за допомогою спеціальних інструментів, профілювання для аналізу швидкості й ефективності роботи застосунку.

Документування повинно включати опис основних концепцій та принципів архітектури, створення архітектурних діаграм для наочного представлення взаємозв’язків компонентів, опис важливих компонентів та інтерфейсів. Регулярне оновлення документації відповідно до змін у коді та вимог проєкту забезпечує актуальність інформації [9].

РОЗДІЛ 3
ДОСЛІДЖЕННЯ АРХІТЕКТУРНИХ РІШЕНЬ ДЛЯ РОЗРОБКИ КРОСПЛАТФОМЕНОГО ДОДАТКУ ЕЛЕКТРОННОЇ БІБЛІОТЕКИ

3.1 Базова архітектура сайту «Електронна бібліотека» засобами HTML, CSS та JavaScript

3.1.1 Використання новітніх технологій

Для того, щоб у подальшому порівнювати різні архітектурні рішення щодо розробки косплатформенних додатків або сайтів, виберемо як базову архітектуру простого сайту для невеликої електронної бібліотеки. Якщо у майбутньому виникне потреба розширити його функціональність (реєстрація, збереження книг, рейтинг, пошук, API), до нього можна буде підключити серверну частину або базу даних. Тому початкова простота сайту у даному випадку повністю себе виправдовує. Для нас головне оцінити у кожному окремому випадку трудомісткість чи, навпаки, простоту технології створення сайту або додатку, ступінь використання новітніх технологій та якість створеного програмного продукту. Саме з цією метою нами розроблений сайт «Електронна бібліотека», який детально аналізується нижче. У процесі розробки сайту були використані такі технології:

– HTML5. Мова розмітки гіпертекстових документів, яка використовується для структурування вмісту сторінок;

– CSS3. Каскадні таблиці стилів для оформлення сторінок;

– JavaScript (Vanilla JS). Мова програмування для забезпечення необхідного функціоналу сторінок;

– DOM API. Забезпечує взаємодію HTML з JavaScript;

– Responsive Web Design. Сайт адаптується під різні екрани за допомогою CSS Media Queries;

– семантична верстка. Використано сучасні семантичні теги: <header>, <nav>, <main>, <section>, <footer>, що покращує зручність користування сайтом та можливість SEO.

3.1.2 Структура проєкту

На рисунку 3.1 наведена структура сайту «Електронна бібліотека» з базовою архітектурою.

```
/ (коренева директорія)
│
├── index.html        Головна сторінка
├── catalog.html      Каталог книг
├── book.html         Сторінка перегляду окремої книги
├── about.html        Сторінка "Про нас"
├── contacts.html     Сторінка "Контакти"
├── admin.html        Панель додавання книг
│
├── /css/
│   └── styles.css    Основні стилі сайту
│
├── /js/
│   └── script.js     Логіка додавання книг на admin.html
```

Рисунок 3.1 – Структура сайту «Електронна бібліотека»

3.1.3 Головна сторінка – файл index.html

Головна сторінка сайту, код якої наведено на лістингу 3.1, містить вітання користувача, ознайомлення з місією ресурсу, і відіграє роль навігаційного центру сайту: через меню можна перейти до інших розділів. Особливості: містить заголовок і опис, не має динамічного вмісту і по суті є вступною сторінкою, «обкладинкою» сайту.

Лістинг 3.1 – Код головної сторінки сайту – файлу index.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <title>Електронна бібліотека</title>
  <link rel="stylesheet" href="css/styles.css">
</head>
```

```
<body>
  <header>
    <h1>Електронна бібліотека</h1>
    <nav>
      <a href="index.html">Головна</a>
      <a href="catalog.html">Каталог</a>
      <a href="about.html">Про нас</a>
      <a href="contacts.html">Контакти</a>
      <a href="admin.html">Адмін</a>
    </nav>
  </header>
  <div class="container">
    <h2>Ласкаво просимо!</h2>
    <p>Тут ви знайдете велику колекцію книг українською мовою.</p>
  </div>
  <footer>
    <p>© 2025 Електронна бібліотека</p>
  </footer>
</body>
</html>
```

кінець лістингу 3.1

3.1.4 Стилі сайту – файл style.css

Файл style.css (додаток А), містить стилі CSS що використовуються для оформлення всіх сторінок сайту. Вони задають кольори, відступи, шрифти та адаптивну поведінку елементів. Стилі забезпечують єдину візуальну форму та покращують зручність користування.

3.1.5 Скрипт сайту – файл script.js

Файл script.js, код якого наведено на лістингу 3.2, забезпечує додавання даних в текстові поля сторінки admin.html.

Лістинг 3.2 – Код файлу script.js

```
function addBook(title, author) {
  const bookList = document.getElementById("bookList");
  const card = document.createElement("div");
  card.className = "book-card";
  card.innerHTML = `<h3>${title}</h3><p>Автор: ${author}</p>`;
  bookList.appendChild(card);
}

// Використання в admin.html:
// addBook("Назва книги", "Автор");
```

кінець лістингу 3.2

3.1.6 Сторінка «Каталог книг» – файл catalog.html

Сторінка «Каталог книг» сайту, код якої наведено на лістингу 3.3, відображає перелік доступних книг. Особливості: вручну додані книжки (статичний HTML), немає пошуку чи фільтрації – тільки візуальний перегляд.

Лістинг 3.3 – Код сторінки «Каталог книг» сайту – файлу catalog.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Каталог — Електронна бібліотека</title>
  <link rel="stylesheet" href="css/styles.css">
</head>
<body>
  <header>
    <div class="container header-container">
      <h1>📚 Електронна бібліотека</h1>
      <nav>
        <a href="index.html">Головна</a>
        <a href="catalog.html">Каталог</a>
        <a href="about.html">Про нас</a>
        <a href="contacts.html">Контакти</a>
        <a href="admin.html">Адмін</a>
      </nav>
    </div>
  </header>

  <main class="container">
    <h2>Каталог книг</h2>

    <div class="book-list">

      <div class="book-card">
        <h3><a href="book.html">Кобзар</a></h3>
        <p>Автор: Тарас Шевченко</p>
      </div>

      <div class="book-card">
        <h3><a href="books/tini_zabutykh_predkiv.html">Тіні забутих
предків</a></h3>
        <p>Автор: Михайло Коцюбинський</p>
      </div>

      <div class="book-card">
        <h3><a href="books/zacharovanaya_desna.html">Зачарована
Десна</a></h3>
```

```
        <p>Автор: Олександр Довженко</p>
      </div>

      <div class="book-card">
        <h3><a href="books/intermezzo.html">Intermezzo</a></h3>
        <p>Автор: Михайло Коцюбинський</p>
      </div>

      <div class="book-card">
        <h3><a href="books/lyudyna.html">Людина</a></h3>
        <p>Автор: Ольга Кобилянська</p>
      </div>


    </div>
  </main>

  <footer>
    <div class="container">
      <p>© 2025 Електронна бібліотека</p>
    </div>
  </footer>
</body>
</html>
```

кінець лістингу 3.3

3.1.7. Сторінка «Книга» сайту – файл book.html

Сторінка «Книга» сайту (додаток Б), динамічно завантажує інформацію про вибрану книгу на основі параметра book в URL (?book=lyudyna). Показує: назву, автора, опис. Особливості: дані зберігаються у JavaScript-об’єкті books, якщо книга не знайдена – показується повідомлення «Книгу не знайдено».

3.1.8 Сторінка «Контакти» – файл contacts.html

Сторінка «Контакти» сайту, код якої наведено на лістингу 3.4, містить контактну інформацію для налагодження зв’язків між користувачами та адміністрацією сайту.

Лістинг 3.4 – Код сторінки «Контакти» сайту – файлу contacts.html

```
<div class="container">
  <h2>Про нас</h2>
  <p>Наша місія — зробити українську літературу доступною кожному.</p>
</<!DOCTYPE html>
<html lang="uk">
<head>
```

```
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Контакти — Електронна бібліотека</title>
  <link rel="stylesheet" href="css/styles.css">
</head>
<body>
  <header>
    <div class="container header-container">
      <h1>📚 Електронна бібліотека</h1>
      <nav>
        <a href="index.html">Головна</a>
        <a href="catalog.html">Каталог</a>
        <a href="about.html">Про нас</a>
        <a href="contacts.html">Контакти</a>
        <a href="admin.html">Адмін</a>
      </nav>
    </div>
  </header>

  <main class="container">
    <section>
      <h2>Контактна інформація</h2>
      <p>Якщо у вас є запитання, пропозиції або ви бажаєте співпрацювати
— звертайтесь до нас:</p>

      <ul>
        <li><strong>Електронна пошта:</strong> <a
href="mailto:info@elib.ua">info@elib.ua</a></li>
        <li><strong>Телефон:</strong> <a href="tel:+380441234567">+38
(044) 123-45-67</a></li>
        <li><strong>Facebook:</strong> <a
href="https://facebook.com/elibua"
target="_blank">facebook.com/elibua</a></li>
        <li><strong>Адреса:</strong> вул. Книжкова, 12, м. Київ, Україна,
01001</li>
      </ul>
    </section>

    <section>
      <h3>Зв'язатися через форму</h3>
      <form action="#" method="post">
        <label for="name">Ім’я:</label><br>
        <input type="text" id="name" name="name" required><br><br>

        <label for="email">Електронна пошта:</label><br>
        <input type="email" id="email" name="email" required><br><br>

        <label for="message">Повідомлення:</label><br>
        <textarea id="message" name="message" rows="5"
required></textarea><br><br>
```

```
        <button type="submit" class="btn">Надіслати</button>
      </form>
    </section>
  </main>

  <footer>
    <div class="container">
      <p>© 2025 Електронна бібліотека</p>
    </div>
  </footer>
</body>
</html> div>
```

кінець лістингу 3.4

3.1.9 Сторінка «Додавання книги» – файл admin.html

Сторінка «Додавання книги» сайту, код якої наведено на лістингу 3.5, додає нові книги у список на поточній сторінці (але не зберігає їх постійно, лише в DOM під час сесії). Імітує адмін-панель для тестування або локального додавання книг.

Лістинг 3.5 – Код сторінки «Додавання книги» – файлу admin.html

```
<div class="container">
  <h2>Додати нову книгу</h2>
  <form id="bookForm">
    <input type="text" id="title" placeholder="Назва книги" required>
    <input type="text" id="author" placeholder="Автор" required>
    <button type="submit">Додати</button>
  </form>
  <div id="bookList"></div>
</div>

<script src="js/script.js"></script>
<script>
  document.getElementById("bookForm").addEventListener("submit",
function(e) {
    e.preventDefault();
    const title = document.getElementById("title").value;
    const author = document.getElementById("author").value;
    addBook(title, author);
  });
</script>
```

кінець лістингу 3.5

3.1.10 Сторінка «Про нас» – файл about.html

Сторінка «Про нас» сайту, код якої наведено на лістингу 3.6, інформує користувача про місію проєкту та його цілі. Описує: ідею проєкту, його користь, що пропонується користувачам.

Лістинг 3.6 – Код сторінки «Про нас» сайту – файлу about.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Про нас — Електронна бібліотека</title>
  <link rel="stylesheet" href="css/styles.css">
</head>
<body>
  <header>
    <div class="container header-container">
      <h1>📚 Електронна бібліотека</h1>
      <nav>
        <a href="index.html">Головна</a>
        <a href="catalog.html">Каталог</a>
        <a href="about.html">Про нас</a>
        <a href="contacts.html">Контакти</a>
        <a href="admin.html">Адмін</a>
      </nav>
    </div>
  </header>

  <main class="container">
    <section class="about-section">
      <h2>Про нас</h2>
      <p>Наша місія — зробити українську літературу доступною кожному. Ми віримо, що книги мають силу формувати свідомість, розвивати критичне мислення та зберігати національну ідентичність.</p>

      <p>Електронна бібліотека — це некомерційний проєкт, створений для збереження та поширення української літературної спадщини. Ми постійно оновлюємо нашу колекцію, додаючи нові твори класичних та сучасних авторів.</p>

      <h3>Що ми пропонуємо:</h3>
      <ul>
        <li>Доступ до книг онлайн і офлайн (PDF)</li>
        <li>Класична та сучасна українська література</li>
        <li>Зручний каталог та пошук</li>
        <li>Можливість завантаження у зручному форматі</li>
      </ul>
```

```
      <p>Приєднуйтесь до спільноти любителів української книги!</p>
    </section>
  </main>

  <footer>
    <div class="container">
      <p>© 2025 Електронна бібліотека</p>
    </div>
  </footer>
</body>
</html>
```

кінець лістингу 3.6

3.1.11 Аналіз результатів функцінування сайту з базовою архітектурою

На рисунках 3.2 – 3.5 наведені результати відображення сторінок сайту «Електронна бібліотека» в браузері. Критичний аналіз цих результатів дозволяє розробити шляхи подальшого удосконалення сайту.

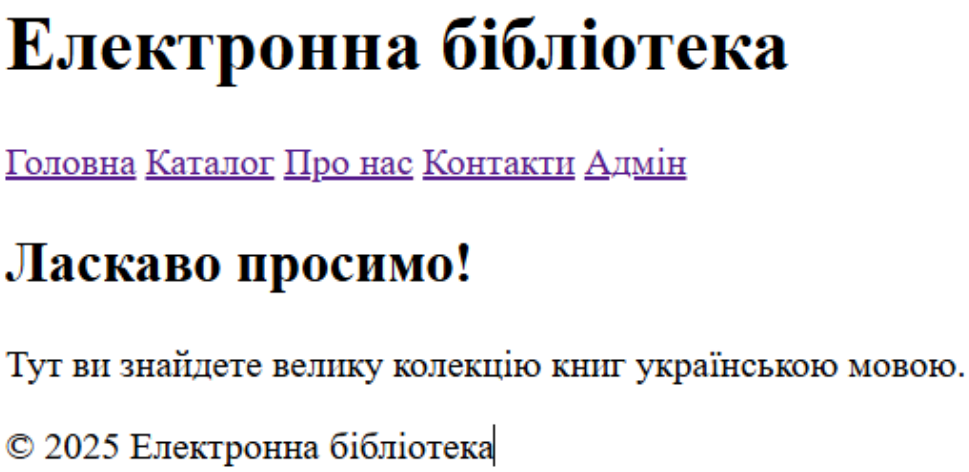

Рисунок 3.2 – Відображення головної сторінки index.html сайту в браузері

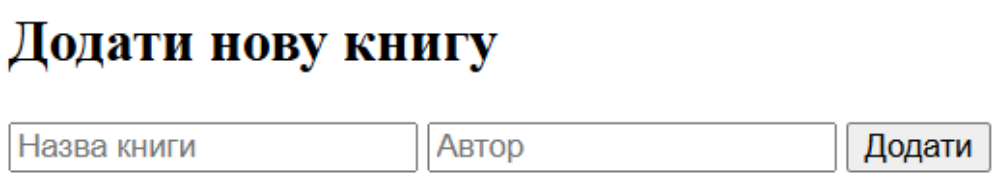

Рисунок 3.3 – Відображення сторінки admin.html сайту в браузері

Рисунок 3.4 – Відображення сторінки catalog.html сайту в браузері

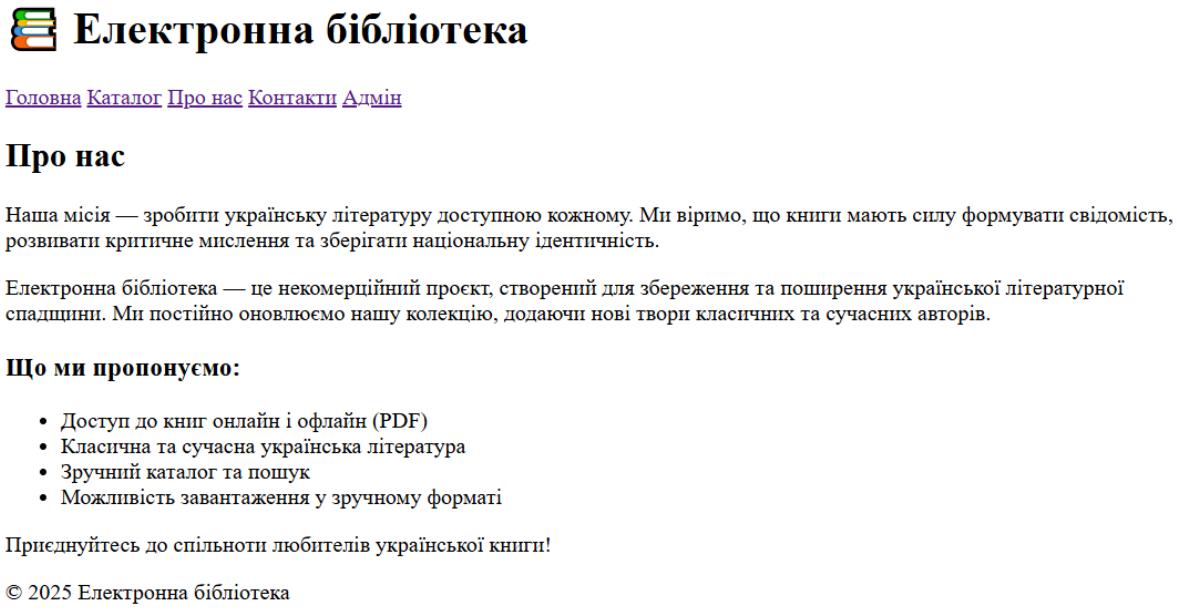

Рисунок 3.5 – Відображення сторінки about.html сайту в браузері

На рисунку 3.6 наведено відображення сторінки з контактною інформацією сайту «Електронна бібліотека» в браузері.

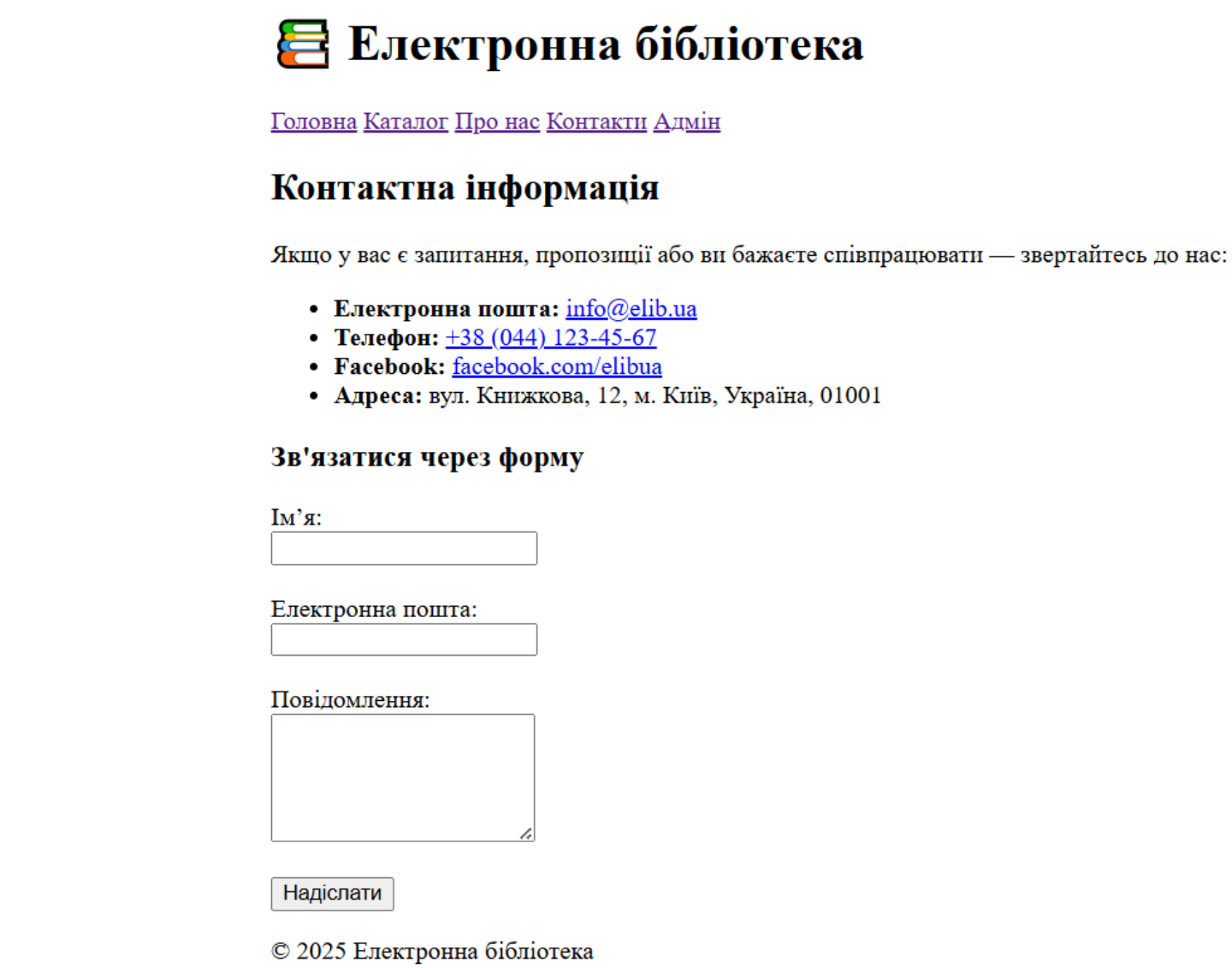

Рисунок 3.6 – Відображення сторінки contacts.html сайту в браузері

3.1.12 Шляхи удосконалення сайту з базовою архітектурою

Виходячи з проведеного аналізу створення сайту «Електронна бібліотека» з базовою архітектурою розроблені заходи щодо подальшого його удосконалення.

На нашу думку, варто розробити динамічний каталог книг (замість статичного). Зараз книги в каталозі (catalog.html) додаються вручну в HTML-код. Пропонується таке рішення: зберігати всі книги в окремому JSON-файлі або на сервері; завантажувати список через JavaScript та автоматично рендерити картки. Можна підключити Firebase або простий backend (Node.js/Express).

Запропонована сторінка admin.html додає книги тільки тимчасово в DOM (без збереження). Рішення: Зробити реальну адмін-сторінку**:** книги зберігаються

в базі даних (наприклад, JSON файл, MongoDB, Firebase). Використати API для додавання, редагування, видалення книг. Передбачити авторизацію адміністратора (логін/пароль).

Передбачити мобільну адаптацію. Зараз сайт частково адаптований (@media (max-width: 768px)), але потребує покращення. Рішення: зробити: меню-гамбургер; покращити верстку форм на мобільних пристроях. Тестувати на різних екранах (телефон, планшет).

Передбачити пошук по каталогу. Проблема: користувач не може знайти книгу за назвою чи автором. Рішення: додати рядок пошуку (JS-фільтрація по назві або автору); додати автозаповнення (autocomplete) за першими літерами.

Додати SEO та мета-теги. Рішення: додати meta-теги для опису, ключових слів, Open Graph (для пошуковиків і соцмереж; додати динамічні заголовки: щоб кожна книга мала свою мета-інформацію.

Передбачити відгуки або рейтинг книг. Рішення: додати можливість залишати відгуки/оцінки для кожної книги; зберігати у backend або у Firebase Realtime DB.

Передбачити форму контакту з обробкою. Проблема: форма в contacts.html не надсилає дані. Рішення: підключити сервіси: Formspree.io; EmailJS. Або реалізувати обробку на сервері.

Вирішити питання безпеки**.** Рішення: додати авторизацію з захищеними паролями, Організувати передачу даних по HTTPS. Передбачити перевірку введених даних (Sanitization).

3.2 Аналіз архітектури додатку на базі фреймворку Flutter

3.2.1 Використані новітні технології:

– Flutter (Dart);

– Stateless / Stateful Widgets;

– вбудована навігація Navigator.push.

3.2.2 Структура проєкту на базі фреймворку Flutter

На рисунку 3.7 наведена структура додатку «Електронна бібліотека» на базі фреймворку Flutter. В лістингу 3.7 навелений конфігураційний файл проекту, який визначає метадані (назву, опис), версію мови та список усіх зовнішніх бібліотек, необхідних для роботи додатку. Модель даних (ліст. 3.8), яка виступає шаблоном для створення об'єктів книг із фіксованими властивостями: назвою, автором та описом.

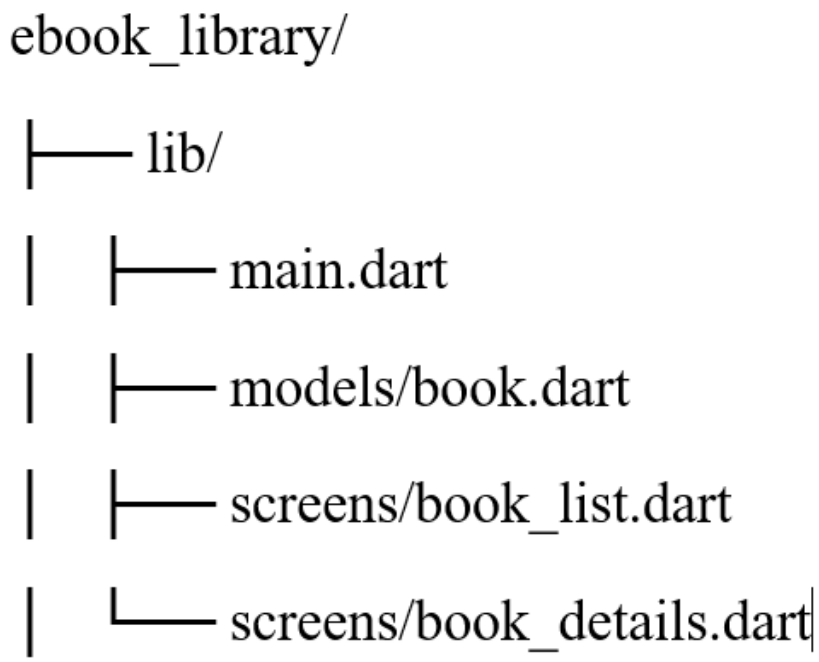

Рисунок 3.7 – Структура додатку «Електронна бібліотека» на базі фреймворку Flutter

Лістинг 3.7 – Код файлу pubspec.yaml

```
name: ebook_library
description: A simple electronic library app.

environment:
  sdk: '>=2.17.0 <3.0.0'

dependencies:
  flutter:
    sdk: flutter

  cupertino_icons: ^1.0.2
```

кінець лістингу 3.7

Лістинг 3.8 – Код файлу lib/models/book.dart

```
class Book {
  final String title;
  final String author;
  final String description;

  Book({
    required this.title,
    required this.author,
    required this.description,
  });
}
```

кінець лістингу 3.8

У лістингу 3.9 представлено код екрану списку книг у Flutter. Використовується ListView.builder для генерації списку з об’єктів Book. Натиснення на елемент відкриває екран із деталями книги, що демонструє базову навігацію фреймворку.

Лістинг 3.9 – Код файлу lib/screens/book_list.dart

```
import 'package:flutter/material.dart';
import '../models/book.dart';
import 'book_details.dart';

class BookListScreen extends StatelessWidget {
  final List<Book> books = [
    Book(
      title: 'Кобзар',
      author: 'Тарас Шевченко',
      description: 'Збірка поезій українського класика.',
    ),
    Book(
      title: 'Майстер і Маргарита',
      author: 'Михайло Булгаков',
      description: 'Містичний роман про добро і зло.',
    ),
    Book(
      title: '451° за Фаренгейтом',
      author: 'Рей Бредбері',
      description: 'Антиутопія про заборону книг.',
    ),
  ];

  @override
  Widget build(BuildContext context) {
```

```
    return Scaffold(
      appBar: AppBar(
        title: const Text('Електронна бібліотека'),
      ),
      body: ListView.builder(
        itemCount: books.length,
        itemBuilder: (context, index) {
          final book = books[index];
          return ListTile(
            title: Text(book.title),
            subtitle: Text(book.author),
            onTap: () {
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (_) => BookDetailsScreen(book: book),
                ),
              );
            },
          );
        },
      ),
    );
  }
}
```

кінець лістингу 3.9

Лістинг 3.10 містить код екрану деталей книги у Flutter. Тут відображаються заголовок, автор і опис вибраної книги. Структура реалізована через Widgets і підтримує просту та інформативну навігацію.

Лістинг 3.10 – Код файлу lib/screens/book_details.dart

```
import 'package:flutter/material.dart';
import '../models/book.dart';
class BookDetailsScreen extends StatelessWidget {
  final Book book;
  const BookDetailsScreen({super.key, required this.book});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Деталі книги'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
```

```
          children: [
            Text(book.title,
                style: const TextStyle(
                    fontSize: 24, fontWeight: FontWeight.bold)),
            const SizedBox(height: 8),
            Text(book.author, style: const TextStyle(fontSize: 18, color:
Colors.grey)),
            const Divider(height: 24),
            Text(book.description, style: const TextStyle(fontSize: 16)),
          ],
        ),
      ),
    );
  }
}
```

кінець лістингу 3.10

Лістинг 3.11 демонструє точку входу до Flutter-додатку. У функції main() запускається MaterialApp, у якому визначено головний екран. Це стандартна стартова конфігурація Flutter-проєкту.

Лістинг 3.11 – Код файлу lib/main.dart

```
import 'package:flutter/material.dart';
import 'screens/book_list.dart';

void main() {
  runApp(const EBookLibraryApp());
}

class EBookLibraryApp extends StatelessWidget {
  const EBookLibraryApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Електронна бібліотека',
      theme: ThemeData(
        primarySwatch: Colors.indigo,
        useMaterial3: true,
      ),
      home: BookListScreen(),
    );
  }
}
```

кінець лістингу 3.11

3.2.2 Основні отримані результати:

– список книг зі скролом;

– перехід на екран деталей при кліку на елемент;

– назад – стандартна навігація.

3.2.3 Шляхи удосконалення додатку:

– додати кнопку «Додати книгу» (з формою);

– зберігати книги в SQLite (sqflite) або Hive;

– інтеграція з Firebase Firestore для хмарного зберігання;

– авторизація користувачів.

3.3 Дослідження архітектури додатку на базі фреймворку Xamarin

3.3.1 Створення проєкту на базі фреймворку Xamarin

В Visual Studio виконати послідовність команд:

File > New > Project > Xamarin.Forms App (Blank). В результаті буде створено проект з такими характеристиками:

– назва: EBookLibrary;

– шаблон: Blank;

– спільний код: .NET Standard.

3.3.2 Структура рішення

На рисунку 3.8 наведена структура додатку «Електронна бібліотека» на базі фреймворку Xamarin. Лістинг 3.12 описує модель даних мовою C#, створюючи клас із властивостями для збереження назви, автора та опису книги.

Лістинг 3.12 – Код файлу Book.cs

```
namespace EBookLibrary.Models
{
    public class Book
    {
        public string Title { get; set; }
        public string Author { get; set; }
```

```
        public string Description { get; set; }
    }
}
```

кінець лістингу 3.12

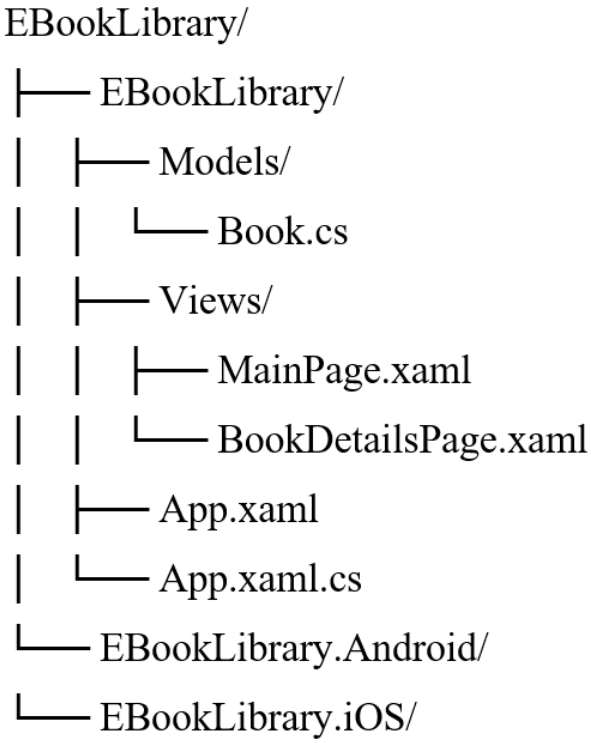

Рисунок 3.8 – Структура додатку «Електронна бібліотека» на базі фреймворку Xamarin

XAML-розмітку головної сторінки Xamarin-додатку (ліст. 3.13). Вона містить заголовок і список доступних книг, реалізований через ListView. Сторінка забезпечує базову навігацію до детального перегляду книги.

Лістинг 3.13 – Код файлу MainPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             x:Class="EBookLibrary.Views.MainPage"
             Title="Електронна бібліотека">

    <StackLayout Padding="10">
        <Label Text="Список книг:"
               FontSize="20"
               HorizontalOptions="Center" />
        <ListView x:Name="BooksListView"
                  ItemSelected="BooksListView_ItemSelected">
```

```
            <ListView.ItemTemplate>
                <DataTemplate>
                    <TextCell Text="{Binding Title}"
                              Detail="{Binding Author}" />
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
</ContentPage>
```

кінець лістингу 3.13

Лістинг 3.14 містить C#-код для роботи головної сторінки в Xamarin. Формується список об’єктів Book та обробляється перехід до сторінки з деталями книги. Код демонструє типовий спосіб роботи з подіями у Xamarin.Forms.

Лістинг 3.14 – Код файлу MainPage.xaml.cs

```
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using EBookLibrary.Models;
using System.Collections.Generic;

namespace EBookLibrary.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class MainPage : ContentPage
    {
        List<Book> books;

        public MainPage()
        {
            InitializeComponent();

            books = new List<Book>
            {
                new Book { Title = "Кобзар", Author = "Тарас Шевченко",
Description = "Збірка поезій українського класика." },
                new Book { Title = "Майстер і Маргарита", Author =
"Михайло Булгаков", Description = "Містичний роман про добро і зло." },
                new Book { Title = "451° за Фаренгейтом", Author = "Рей
Бредбері", Description = "Антиутопія про заборону книг." }
            };

            BooksListView.ItemsSource = books;
        }
```

```
        private async void BooksListView_ItemSelected(object sender,
SelectedItemChangedEventArgs e)
        {
            var selectedBook = e.SelectedItem as Book;
            if (selectedBook != null)
            {
                await Navigation.PushAsync(new
BookDetailsPage(selectedBook));
                BooksListView.SelectedItem = null;
            }
        }
    }
}
```

кінець лістингу 3.14

У лістингу 3.15 наведено XAML-код сторінки деталей книги в Xamarin. Виводяться заголовок, автор і опис, оформлені у вертикальний стек. Це проста структура для відображення інформації у мобільному додатку.

Лістинг 3.15 – Код файлу BookDetailsPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             x:Class="EBookLibrary.Views.BookDetailsPage"
             Title="Деталі книги">

    <StackLayout Padding="10">
        <Label Text="{Binding Title}" FontSize="24" FontAttributes="Bold"
/>
        <Label Text="{Binding Author}" FontSize="18" TextColor="Gray" />
        <BoxView HeightRequest="1" Color="LightGray" Margin="0,10" />
        <Label Text="{Binding Description}" FontSize="16" />
    </StackLayout>
</ContentPage>
```

кінець лістингу 3.15

Лістинг 3.16 ініціалізує сторінку деталей книги у додатку Xamarin, встановлюючи переданий об'єкт як контекст прив'язки для відображення його даних на екрані. Точкою входу в додаток Xamarin (ліст. 3.17), де ініціалізується головна сторінка та обгортається в навігаційний контейнер, щоб користувач міг переміщатися між сторінками.

Лістинг 3.16 – Код файлу BookDetailsPage.xaml.cs

```
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using EBookLibrary.Models;

namespace EBookLibrary.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class BookDetailsPage : ContentPage
    {
        public BookDetailsPage(Book book)
        {
            InitializeComponent();
            BindingContext = book;
        }
    }
}
```

кінець лістингу 3.16

Лістинг 3.17 – Код файлу App.xaml.cs

```
using Xamarin.Forms;
using EBookLibrary.Views;

namespace EBookLibrary
{
    public partial class App : Application
    {
        public App()
        {
            InitializeComponent();
            MainPage = new NavigationPage(new MainPage());
        }
    }
}
```

кінець лістингу 3.17

3.3.9 Результат тестування додатку

В результаті запуску додатку:

– відображається список книг;

– клік по книзі відкриває сторінку з детальним описом;

– все працює локально, без зовнішніх джерел.

3.3.10 Шляхи подальшого удосконалення:

– додати SQLite (через SQLite-net-pcl) для збереження книг;

– додати можливість додавання/редагування книг;

– синхронізація з сервером через API;

– пошук і фільтрація;

– авторизація користувачів.

3.4 Аналіз архітектури додатку на базі фреймворку React Native

3.4.1 Створення та ініціалізація проєкту на базі фреймворку React Native

Якщо потрібно швидко створити та запустити проєкт, то найпростіше це виконати за допомогою Expo:

```
npx create-expo-app EbookLibrary
cd EbookLibrary
npx expo install react-native-screens react-native-safe-area-context react-native-gesture-handler react-native-reanimated react-navigation/native react-navigation/native-stack
```

3.4.2 Структура проєкту на базі фреймворку React Native

На рисунку 3.9 наведена структура додатку «Електронна бібліотека» на базі фреймворку React Native.

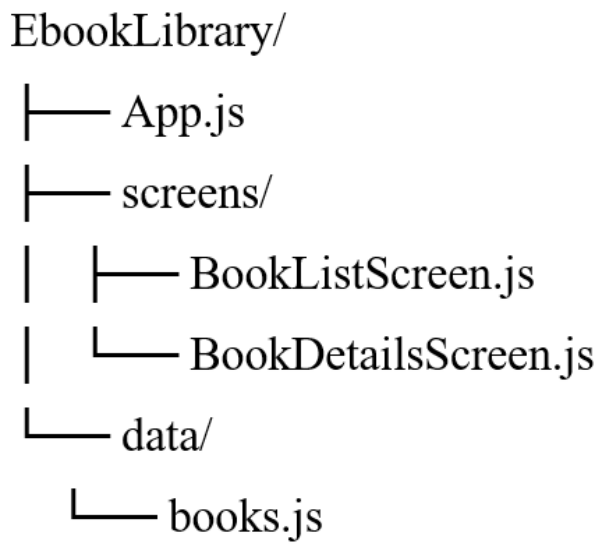

Рисунок 3.9 – Структура додатку «Електронна бібліотека» на базі фреймворку React Native

Лістинг 3.18 містить файл даних React Native із масивом книг. Дані використовуються для наповнення списку та навігації між екранами. Це аналог локальної JSON-бази для невеликих застосунків.

Лістинг 3.18 – Код файлу data/books.js

```
export const books = [
  {
    id: '1',
    title: 'Кобзар',
    author: 'Тарас Шевченко',
    description: 'Збірка поезій українського класика.',
  },
  {
    id: '2',
    title: 'Майстер і Маргарита',
    author: 'Михайло Булгаков',
    description: 'Містичний роман про добро і зло.',
  },
  {
    id: '3',
    title: '451° за Фаренгейтом',
    author: 'Рей Бредбері',
    description: 'Антиутопія про заборону книг.',
  },
];
```

кінець лістингу 3.18

У лістингу 3.19 наведено код екрана списку книг у React Native. FlatList використовується для виведення даних, а TouchableOpacity забезпечує натискання на елемент. При виборі книги виконується навігація до екрану деталей.

Лістинг 3.19 – Код файлу screens/BookListScreen.js

```
import React from 'react';
import { View, Text, FlatList, TouchableOpacity, StyleSheet } from
'react-native';
import { books } from '../data/books';

export default function BookListScreen({ navigation }) {
  const renderItem = ({ item }) => (
    <TouchableOpacity
      style={styles.item}
```

```
      onPress={() => navigation.navigate('BookDetails', { book: item })}
    >
      <Text style={styles.title}>{item.title}</Text>
      <Text style={styles.author}>{item.author}</Text>
    </TouchableOpacity>
  );

  return (
    <View style={styles.container}>
      <Text style={styles.header}>Електронна бібліотека</Text>
      <FlatList
        data={books}
        renderItem={renderItem}
        keyExtractor={(item) => item.id}
      />
    </View>
  );
}

const styles = StyleSheet.create({
  container: { flex: 1, padding: 16 },
  header: { fontSize: 24, marginBottom: 16, fontWeight: 'bold',
textAlign: 'center' },
  item: { padding: 16, borderBottomWidth: 1, borderBottomColor: '#ccc' },
  title: { fontSize: 18 },
  author: { fontSize: 14, color: 'gray' },
});
```

кінець лістингу 3.19

Екран деталей книги у React Native (ліст. 3.20). Відображаються заголовок, автор та опис, структуровані за допомогою View і Text. Це проста реалізація другого екрану у стек-навігації.

Лістинг 3.20 – Код файлу screens/BookDetailsScreen.js

```
import React from 'react';
import { View, Text, StyleSheet } from 'react-native';

export default function BookDetailsScreen({ route }) {
  const { book } = route.params;

  return (
    <View style={styles.container}>
      <Text style={styles.title}>{book.title}</Text>
      <Text style={styles.author}>{book.author}</Text>
      <Text style={styles.description}>{book.description}</Text>
    </View>
  );
```

```
}

const styles = StyleSheet.create({
  container: { flex: 1, padding: 16 },
  title: { fontSize: 24, fontWeight: 'bold', marginBottom: 8 },
  author: { fontSize: 18, color: 'gray', marginBottom: 16 },
  description: { fontSize: 16 },
});
```

кінець лістингу 3.20

У лістингу 3.21 подано головний файл App.js, який містить конфігурацію навігації. Використовується NavigationContainer та стек-навігатор для переходів між екранами. Це основа маршрутизації React Native-додатка.

Лістинг 3.21 – Код файлу App.js

```
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-
stack';
import BookListScreen from './screens/BookListScreen';
import BookDetailsScreen from './screens/BookDetailsScreen';

const Stack = createNativeStackNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="BookList" component={BookListScreen}
options={{ title: 'Бібліотека' }} />
        <Stack.Screen name="BookDetails" component={BookDetailsScreen}
options={{ title: 'Деталі книги' }} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

кінець лістингу 3.21

3.4.3 Отримані результати:

– список книг – заголовок, автор;

– натиск по елементу – відкриває деталі книги;

– навігація через react-navigation.

3.4.4 Шляхи удосконалення додатку:

– додати форму додавання нової книги;

– підключити AsyncStorage або SQLite для збереження;

– інтегрувати з Node.js API / Firebase / Supabase;

– додати авторизацію користувачів.

3.5 Дослідження архітектури додатку на базі фреймворку .NET MAUI

3.5.1 Структура проєкту на базі фреймворку .NET MAUI

На рисунку 3.10 наведена структура додатку «Електронна бібліотека» на базі фреймворку .NET MAUI.

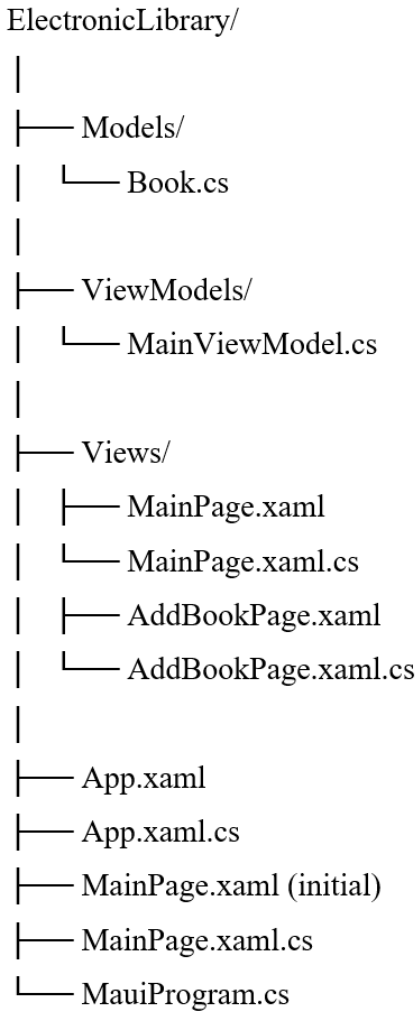

Рисунок 3.10 – Структура додатку «Електронна бібліотека» на базі фреймворку .NET MAUI

C#-клас Book для .NET MAUI (ліст. 3.21). Це модель даних із трьома властивостями: Title, Author і Description. Модель використовується у ViewModel та інтерфейсі користувача.

Лістинг 3.21 – Код файлу Book.cs

```
namespace ElectronicLibrary.Models
{
    public class Book
    {
        public string Title { get; set; }
        public string Author { get; set; }
        public string Description { get; set; }
    }
}
```

кінець лістингу 3.21

У лістингу 3.22 наведено код ViewModel для MAUI-додатку. Тут реалізовано список книг і метод для додавання нових записів. ViewModel працює за принципами MVVM і забезпечує розділення логіки та інтерфейсу.

Лістинг 3.22 – Код файлу MainViewModel.cs

```
using System.Collections.ObjectModel;
using System.Windows.Input;
using ElectronicLibrary.Models;

namespace ElectronicLibrary.ViewModels
{
    public class MainViewModel : BaseViewModel
    {
        public ObservableCollection<Book> Books { get; set; } = new();

        public ICommand AddBookCommand { get; }

        public MainViewModel()
        {
            // Початкові дані
            Books.Add(new Book { Title = "1984", Author = "George
Orwell", Description = "Dystopian novel" });
            Books.Add(new Book { Title = "Brave New World", Author =
"Aldous Huxley", Description = "Science fiction" });
        }

        public void AddBook(Book newBook)
        {
```

```
            if (newBook != null
&& !string.IsNullOrWhiteSpace(newBook.Title))
                Books.Add(newBook);
        }
    }
}
```

кінець лістингу 3.22

XAML-розмітку головної сторінки MAUI-додатку (ліст. 3.23). Відображається список книг через CollectionView та кнопка для додавання нової книги. Інтерфейс дотримується патерну MVVM.

Лістинг 3.23 – Код файлу MainPage.xaml

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:viewmodels="clr-
namespace:ElectronicLibrary.ViewModels"
             x:Class="ElectronicLibrary.Views.MainPage">

    <ContentPage.BindingContext>
        <viewmodels:MainViewModel />
    </ContentPage.BindingContext>

    <StackLayout Padding="20">
        <Label Text="Електронна бібліотека" FontSize="24"
HorizontalOptions="Center" />

        <CollectionView ItemsSource="{Binding Books}">
            <CollectionView.ItemTemplate>
                <DataTemplate>
                    <Frame BorderColor="Gray" Padding="10" Margin="5">
                        <StackLayout>
                            <Label Text="{Binding Title}"
FontAttributes="Bold" />
                            <Label Text="{Binding Author}" />
                            <Label Text="{Binding Description}"
FontSize="12" />
                        </StackLayout>
                    </Frame>
                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>

        <Button Text="Додати книгу" Clicked="OnAddBookClicked" />
    </StackLayout>
</ContentPage>
```

кінець лістингу 3.23

У лістингу 3.24 подано C#-код для головної сторінки MAUI, що реалізує обробку події додавання книги. При натисканні здійснюється перехід на форму створення нової книги. Код пов’язує логіку з ViewModel.

Лістинг 3.24 – Код файлу MainPage.xaml.cs

```
using ElectronicLibrary.ViewModels;

namespace ElectronicLibrary.Views;

public partial class MainPage : ContentPage
{
    private MainViewModel viewModel;

    public MainPage()
    {
        InitializeComponent();
        viewModel = BindingContext as MainViewModel;
    }

    private async void OnAddBookClicked(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new AddBookPage(viewModel));
    }
}
```

кінець лістингу 3.24

XAML-розмітку сторінки додавання книги (ліст. 3.25). Вона складається з полів введення для назви, автора та опису, а також кнопки збереження.

Лістинг 3.25 – Код файлу AddBookPage.xaml

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             x:Class="ElectronicLibrary.Views.AddBookPage"
             Title="Додати книгу">

    <StackLayout Padding="20">
        <Entry x:Name="titleEntry" Placeholder="Назва книги" />
        <Entry x:Name="authorEntry" Placeholder="Автор" />
        <Editor x:Name="descEditor" Placeholder="Опис"
AutoSize="TextChanges" />

        <Button Text="Зберегти" Clicked="OnSaveClicked" />
    </StackLayout>
</ContentPage>
```

кінець лістингу 3.25

Код, описаний в лістингу 3.26, відповідає за логіку сторінки додавання книги: він збирає дані користувача, створює об'єкт Book, зберігає його через ViewModel та керує навігацією на головний екран. Безпосередньо роботу програми запускає код (ліст. 3.27), який ініціалізує компоненти, встановлює головну сторінку та забезпечує основну навігаційну структуру додатку. Увесь цей функціонал будується на основі лістингу 3.28, який є основним конфігураційним файлом .NET MAUI, що створює та налаштовує екземпляр застосунку, реєструючи необхідні ресурси та сервіси.

Лістинг 3.26 – Код файлу AddBookPage.xaml.cs

```
using ElectronicLibrary.Models;
using ElectronicLibrary.ViewModels;

namespace ElectronicLibrary.Views;

public partial class AddBookPage : ContentPage
{
    private MainViewModel viewModel;

    public AddBookPage(MainViewModel vm)
    {
        InitializeComponent();
        viewModel = vm;
    }

    private async void OnSaveClicked(object sender, EventArgs e)
    {
        var book = new Book
        {
            Title = titleEntry.Text,
            Author = authorEntry.Text,
            Description = descEditor.Text
        };

        viewModel.AddBook(book);
        await Navigation.PopAsync(); // Повертаємось на головну
    }
}
```

кінець лістингу 3.26

Лістинг 3.27 – Код файлу App.xaml.cs

```
public partial class App : Application
{
```

```
    public App()
    {
        InitializeComponent();

        MainPage = new NavigationPage(new Views.MainPage());
    }
}
```

кінець лістингу 3.27

Лістинг 3.28 – Код файлу MauiProgram.cs

```
public static class MauiProgram
{
    public static MauiApp CreateMauiApp()
    {
        var builder = MauiApp.CreateBuilder();
        builder
            .UseMauiApp<App>()
            .ConfigureFonts(fonts =>
            {
                fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");
            });

        // builder.Services.AddSingleton<MainViewModel>();

        return builder.Build();
    }
}
```

кінець лістингу 3.28

Шляхи удосконалення додатку на базі фреймворку .NET MAUI (табл. 3.1)

Таблиця 3.1 – Шляхи удосконалення додатку

Напрямок	Опис
Збереження даних	Підключити SQLite через SQLite-net-pcl, або використовувати Preferences/File IO для збереження книжок
Редагування/Видалення	Додати можливість редагування та видалення книг зі списку
Пошук та фільтрація	Додати пошук по назві або автору
Синхронізація з хмарою	Додати інтеграцію з Firebase або REST API
Стилізація UI	Підключити кастомні теми, зробити адаптивний дизайн
Юзер-менеджмент	Авторизація/реєстрація користувачів
Unit/UI тести	Підключити MSTest або xUnit, написати тести для ViewModel-ів

3.6 Аналіз архітектури додатку на базі фреймворку Kotlin Multiplatform

3.6.1 Архітектура проєкту на базі фреймворку Kotlin Multiplatform

На рисунку 3.11 наведена структура додатку «Електронна бібліотека» на базі фреймворку Kotlin Multiplatform.

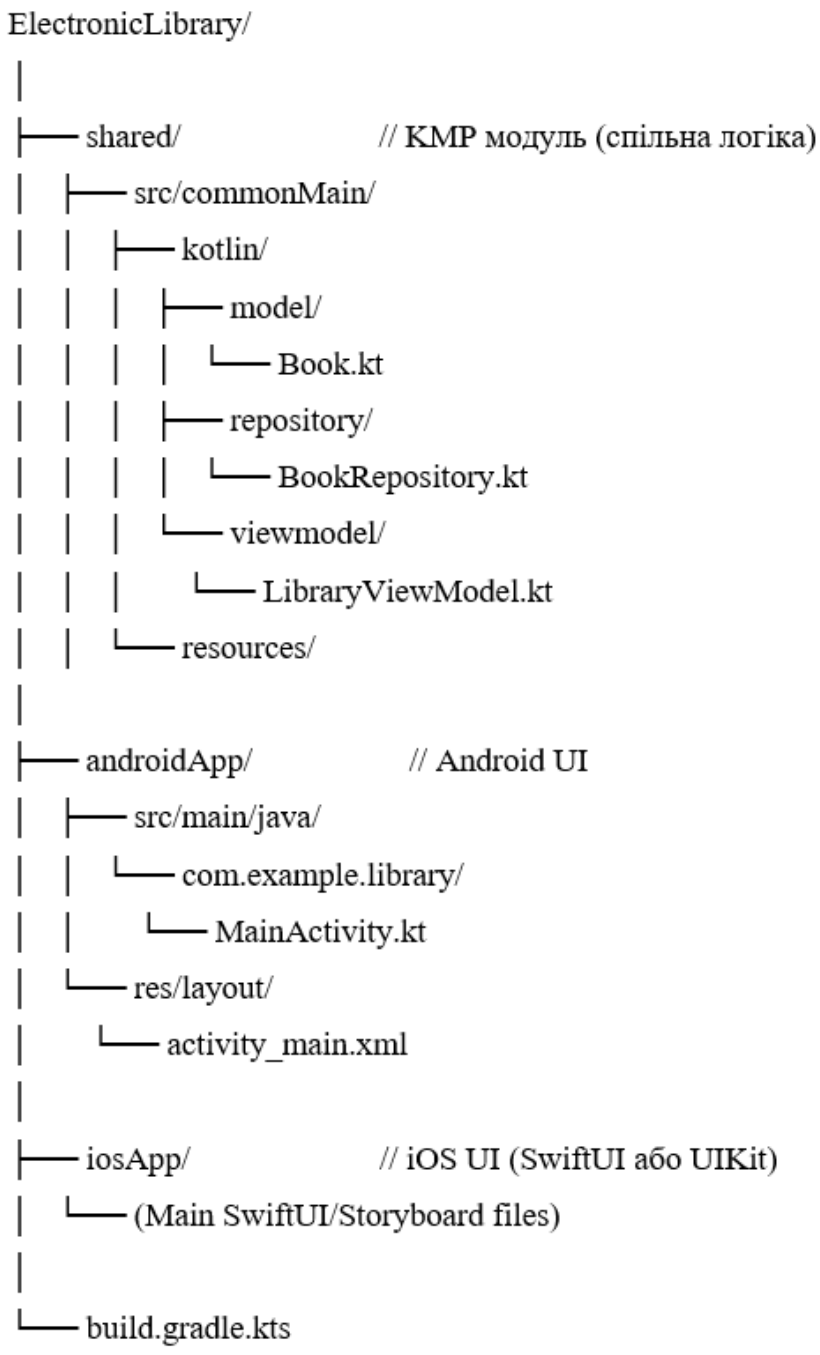

Рисунок 3.11 – Структура додатку «Електронна бібліотека» на базі фреймворку Kotlin Multiplatform

В лістингу 3.29 файл налаштовує спільний KMP модуль, визначаючи цільові платформи (Android та iOS (x64, arm64, симулятор)) і підключаючи ключову залежність для асинхронної роботи.

Лістинг 3.29 – Код KMP модуля (shared)

```
build.gradle.kts (в shared):
kotlin {
    android()
    iosX64()
    iosArm64()
    iosSimulatorArm64()

    sourceSets {
        val commonMain by getting {
            dependencies {
                implementation("org.jetbrains.kotlinx:kotlinx-coroutines-
core:1.7.3")
            }
        }
        val androidMain by getting
        val iosMain by creating {
            dependsOn(commonMain)
        }
        iosX64Main.dependsOn(iosMain)
        iosArm64Main.dependsOn(iosMain)
        iosSimulatorArm64Main.dependsOn(iosMain)
    }
}
```

кінець лістингу 3.29

Клас-дата-модель (ліст. 3.30), написаний на Kotlin, визначає структуру даних для сутності «Книга» (назва, автор, опис), і є спільним для всіх платформ.

Лістинг 3.30 – Код файлу Book.kt

```
package model

data class Book(
    val title: String,
    val author: String,
    val description: String
)
```

кінець лістингу 3.30

Клас в лістингу 3.31 реалізує логіку зберігання даних. Він використовує StateFlow для реактивного відстеження списку книг і містить функцію для додавання нових книг до цього списку.

Лістинг 3.31 – Код файлу BookRepository.kt

```
package repository

import model.Book
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow

class BookRepository {
    private val _books = MutableStateFlow<List<Book>>(emptyList())
    val books: StateFlow<List<Book>> = _books

    init {
        // Початкові книги
        _books.value = listOf(
            Book("1984", "George Orwell", "Dystopian novel"),
            Book("Brave New World", "Aldous Huxley", "Science fiction")
        )
    }

    fun addBook(book: Book) {
        _books.value = _books.value + book
    }
}
```

кінець лістингу 3.31

ViewModel містить бізнес-логіку (ліст. 3.32), отримуючи дані про книги з репозиторію та надаючи функції для їх модифікації, при цьому він є спільним і доступним як для Android, так і для iOS UI.

Лістинг 3.32 – Код файлу LibraryViewModel.kt

```
package viewmodel

import kotlinx.coroutines.flow.StateFlow
import model.Book
import repository.BookRepository

class LibraryViewModel(
    private val repository: BookRepository = BookRepository()
) {
    val books: StateFlow<List<Book>> = repository.books
```

```
    fun addBook(title: String, author: String, description: String) {
        repository.addBook(Book(title, author, description))
    }
}
```

кінець лістингу 3.32

Код в лістингу 3.33 створює Android-екран за допомогою Jetpack Compose, отримує реактивні дані про книги з LibraryViewModel і дозволяє користувачеві відображати та додавати нові книги.

Лістинг 3.33 – Код файлу налаштування Android UI

```
(androidApp/src/main/java/com.example.library/MainActivity.kt)
package com.example.library

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import viewmodel.LibraryViewModel

class MainActivity : ComponentActivity() {
    private val viewModel = LibraryViewModel()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            val books by viewModel.books.collectAsState()

            var title by remember { mutableStateOf("") }
            var author by remember { mutableStateOf("") }
            var description by remember { mutableStateOf("") }

            Scaffold(
                topBar = {
                    TopAppBar(title = { Text("Електронна бібліотека") })
                }
            ) {
                Column(Modifier.padding(16.dp)) {
                    LazyColumn(Modifier.weight(1f)) {
                        items(books) { book ->
```

```
                            Card(Modifier.fillMaxWidth().padding(8.dp),
elevation = 4.dp) {
                                Column(Modifier.padding(8.dp)) {
                                    Text(book.title, style =
MaterialTheme.typography.h6)
                                    Text("Автор: ${book.author}")
                                    Text(book.description)
                                }
                            }
                        }
                    }

                    OutlinedTextField(value = title, onValueChange = {
title = it }, label = { Text("Назва") })
                    OutlinedTextField(value = author, onValueChange = {
author = it }, label = { Text("Автор") })
                    OutlinedTextField(value = description, onValueChange
= { description = it }, label = { Text("Опис") })

                    Button(onClick = {
                        viewModel.addBook(title, author, description)
                        title = ""
                        author = ""
                        description = ""
                    }, modifier = Modifier.padding(top = 8.dp)) {
                        Text("Додати книгу")
                    }
                }
            }
        }
    }
}
```

кінець лістингу 3.33

Код в лістингу 3.34 створює iOS-екран за допомогою SwiftUI, використовує клас-обгортку (LibraryViewModelWrapper) для адаптації LibraryViewModel під вимоги SwiftUI (ObservableObject) та реалізує відображення списку і функціонал додавання нових книг.

Лістинг 3.34 – Код файлу налаштування iOS UI

```
import SwiftUI
import shared

struct ContentView: View {
    @ObservedObject var viewModel = LibraryViewModelWrapper()

    var body: some View {
```

```
        NavigationView {
            VStack {
                List(viewModel.books, id: \.title) { book in
                    VStack(alignment: .leading) {
                        Text(book.title).font(.headline)
                        Text("Автор: \(book.author)")
                        Text(book.description).font(.subheadline)
                    }
                }

                TextField("Назва", text: $viewModel.title)
                TextField("Автор", text: $viewModel.author)
                TextField("Опис", text: $viewModel.description)

                Button("Додати книгу") {
                    viewModel.addBook()
                }
            }
            .padding()
            .navigationTitle("Електронна бібліотека")
        }
    }
}

class LibraryViewModelWrapper: ObservableObject {
    private let viewModel = LibraryViewModel()

    @Published var books: [Book] = []
    @Published var title = ""
    @Published var author = ""
    @Published var description = ""

    init() {
        viewModel.books.watch { newBooks in
            self.books = newBooks ?? []
        }
    }

    func addBook() {
        viewModel.addBook(title: title, author: author, description:
description)
        title = ""
        author = ""
        description = ""
    }
}
```

кінець лістингу 3.34

Шляхи удосконалення додатку на базі фреймворку KMP (табл. 3.2).

Таблиця 3.2 – Шляхи удосконалення додатку

Напрямок	Опис
Збереження даних	Використання SQLDelight або Realm для збереження книжок
Синхронізація з сервером	Підключення Ktor клієнта для роботи з API
Редагування та видалення	Додати підтримку редагування / видалення книг
UI покращення	Додавання анімацій, адаптивного дизайну, тіней, тем
Тестування	Написання unit-тестів для спільної логіки (commonTest)
Авторизація	Додати логін через OAuth або email
DI	Впровадження Koin або Kodein у модуль shared

3.7 Підсумкова порівняльна характеристика різних фреймворків

На підставі аналізу, виконаного нами в процесі дослідження архітектурних рішень при створенні кросплатформенного додатку електронної бібліотеки, запропонована підсумкова порівняльна характеристика різних фреймворків (табл. 3.3).

Таблиця 3.3 – Підсумкова порівняльна характеристика різних фреймворків для розробки кросплатформенних додатків

Критерій	Flutter	React Native	.NET MAUI	Xamarin	Kotlin Multiplatform (KMP)
Мови програмування	Dart	JavaScript / TypeScript	C#	C#	Kotlin
User Interface (UI)	Власний рендеринг (Canvas)	Нативні компоненти + JS Bridge	Нативні компоненти	Нативні компоненти	Нативний UI (iOS/Android окремо)
Продуктивність	Висока	Середня – висока (залежить від UI)	Висока	Висока	Висока (особливо логіка)
Розробка UI	Кастомний у Flutter	JSX, схожий на HTML	XAML	XAML / C#	Нативно (Jetpack Compose / SwiftUI)

Продовження таблиці 3.3

Критерій	Flutter	React Native	.NET MAUI	Xamarin	Kotlin Multiplatform (KMP)
Засоби навігації	Flutter Navigator	React Navigation	Shell / NavigationPage	Shell / NavigationPage	Залежить від реалізації UI
Hot Reload / Live Reload	Так	Так	Так	Так	Частково (залежить від платформи)
Web підтримка	Так (Flutter Web)	Частково (через ReactDOM / адаптації)	Частково (через Blazor/Hybrid)	Ні (лише через MAUI)	Розвивається (KMP Web – ще експериментально)
Спільний код (логіка)	Так, повністю	Так	Так	Так	Так (спільна логіка, UI – окремо)
Спільний UI	Так	Так (хоча нативний)	Так	Так (Xamarin.Forms)	(тільки логіка –UI пишеться окремо)
Платформи	Android, iOS, Web, Desktop	Android, iOS	Android, iOS, Windows, Mac (Blazor)	Android, iOS, Windows	Android, iOS, (Web – частково)
Підтримка корпорацій	Google	Meta (Facebook)	Microsoft	Microsoft	JetBrains + Google (підтримка Kotlin)

Кожен кросплатформний фреймворк пропонує свій унікальний компроміс: Flutter забезпечує максимальну спільність коду (логіка та UI) через власний рендеринг; React Native, .NET MAUI та Xamarin балансують між спільною логікою та використанням нативних UI-компонентів; тоді як Kotlin Multiplatform (KMP) фокусується виключно на спільній логіці, залишаючи розробку нативного UI окремою для кожної платформи для досягнення найкращої нативної якості.

ВИСНОВКИ

У даній кваліфікаційній роботі було вирішено науково-практичне завдання, що полягало в аналізі, проєктуванні та дослідженні архітектурних рішень для створення кросплатформного додатку електронної бібліотеки з метою забезпечення універсального доступу до інформаційних ресурсів. У процесі виконання роботи було отримано вагомі теоретичні та практичні результати.

Насамперед проведено комплексний аналіз предметної області та існуючих рішень на ринку, таких як PocketBook, Moon+Reader та Google Play Книги, що дозволило виявити обмеження існуючих систем щодо кросплатформності та відкритості. На основі цього аналізу сформульовано детальні функціональні вимоги до системи, зокрема щодо каталогізації, пошуку та підтримки форматів, а також нефункціональні вимоги, що стосуються швидкодії, надійності та адаптивності інтерфейсу. Разом із цим розроблено та обґрунтовано архітектуру бази даних системи, яка відповідає третій нормальній формі, де визначено основні сутності та зв’язки між ними, що забезпечує цілісність даних та ефективність виконання запитів.

Важливим етапом роботи стало глибоке теоретичне дослідження патернів проєктування клієнтських додатків, у ході якого проведено порівняльний аналіз підходів MVC, MVP та MVVM. Встановлено, що патерн Model-View-ViewModel є найбільш доцільним для сучасних кросплатформних рішень завдяки ефективному розділенню логіки та інтерфейсу. Для перевірки базової логіки взаємодії користувача з контентом було створено прототип веб-сайту засобами HTML5, CSS3 та JavaScript, що дозволило виявити обмеження нативних веб-технологій для мобільних сценаріїв використанн.

Практична складова роботи полягала у розробці та дослідженні серії прототипів мобільного додатку з використанням провідних фреймворків, таких як Flutter, Xamarin, React Native, .NET MAUI та Kotlin Multiplatform. Результати порівняльного аналізу показали, що Flutter є оптимальним вибором для

швидкого старту та створення єдиного інтерфейсу, тоді як Kotlin Multiplatform найкраще підходить для проєктів зі складною спільною бізнес-логікою.

Робота має значну практичну цінність, оскільки розроблені архітектурні моделі та програмні прототипи є готовою основою для впровадження повноцінної системи електронної бібліотеки. Отримані результати дослідження допомагають оптимізувати процес вибору технологічного стеку для подібних проєктів, а визначені шляхи подальшого вдосконалення, такі як хмарна синхронізація та локальне збереження даних, окреслюють перспективи розвитку системи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Пех П.А., Годлевський З. Е. Вибір фреймворку для кросплатформенної розробки сайту чи додатку. Інновації та перспективні шляхи розвитку інформаційних технологій: матеріали IV міжнародної науково-практичної Інтернет-конференції (ІПШРІТ 2025), м. Черкаси, 25 листопада 2025 р. Черкаси: ЧДТУ, 2025. С.208-210.

2. Годлевський З.Е., наук.кер. Пех П.А. Порівняльний аналіз фреймворків для кросплатформенної розробки сайтів та додатків. Студентський науковий вісник. Випуск 54. Одеса: Видавництво «Гельветика», 2025. С.38-43.

3. Пашковська І.О. Становлення та розвиток електронних бібліотек: історія, сучасний стан таперспективи: кваліфікаційна робота на здобуття освітнього ступеня бакалавр. Київ: Національнаакадемія керівних кадрів культури і мистецтв, 2023. 75 с.

4. Барабаш В., Чернявська Я. Історія розвитку електронних ресурсів бібліотечних установ. Збірникпраць молодих науковців ЦНТУ. 2024. Вип. 14. С. 76-79.

5. Топ застосунків для читання книг: зручні рішення для книголюбів. Galera News. URL: https://galera.news/top-zastosunkiv-dlya-chytannya-knyg-zruchni-rishennya-dlya-knygolyubiv-7505/ (дата звернення: 21.08.2025).

6. 7 кращих мобільних застосунків для читання книг. KITAPP. URL: https://kitapp.pro/uk/7-krashhih-mobilnih-zastosunkiv-dlya-chitannya-knig/ (дата звернення: 29.08.2025).

7. Застосунки для книголюбів. Litosvita. URL: https://litosvita.com/zastosunky-dlya-knyholiubiv/ (дата звернення: 06.09.2025).

8. Баланська Р.Р. Електронна бібліотека: кваліфікаційна робота. Тернопіль, 2024. 32 с.

9. Архітектура застосунку: компоненти та популярні патерни. FoxmindEd. URL: https://foxminded.ua/arkhitektura-zastosunku/ (дата звернення: 20.09.2025).

10. Резанова В.Г., Марченко В.І. Архітектурні патерни при розробці клієнт-серверних додатків. Інформаційнітехнології в науці, виробництві та підприємництві. Київ: КНУТД, 2024. С. 185-188.

11. Розкриття потенціалу кросплатформової розробки: інструменти та бібліотеки, що заслуговують на увагу. PNN. URL: https://pnn.com.ua/ua/blog/detail/unlocking-the-potential-of-cross-platform-development-tools-and-libraries-you-need-to-know (дата звернення: 03.10.2025).