Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ЧИТАННЯ КНИГ РІЗНИХ ФОРМАТІВ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-2
Захаров В.В.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Машталір С.В
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Захарову Володимиру Вадимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка мобільного застосунку для читання книг різних форматів

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 31 травня 2025 р.
3. Вихідні дані до роботи статті в інтернеті стосовно розробки під Android, офіційна документація Android API, офіційна документація Android native development kit, документація tinyXML2.

4. Перелік питань, що потрібно опрацювати в роботі
1. Здатність застосунку пошуку файлів підтримуваних форматів на пристрої користувача.
2. Можливість працювати з форматом EPUB.
3. Можливість працювати з форматом FB2.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми читання книг різних форматів, постановка задачі, тестові електронні книги.

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

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-10.04.25	
3	Аналіз літератури з досліджуваної проблеми	11.04.25-14.04.25	
4	Аналіз технічних засобів	15.04.25-20.04.25	
5	Розробка проєкту	21.04.25-27.04.25	
6	Програмна реалізація	28.04.25-11.05.25	
7	Оформлення пояснювальної записки	12.05.25-20.05.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-18.06.25	
12	Занесення роботи в електронний архів	02.06.25-18.06.25	
	Попередній захист кваліфікаційної роботи	02.06.25-18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ _____проф. Машталір С.В_____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 61 с., 52 рис., 30 джерел.

ЕЛЕКТРОННІ КНИГИ, ANDROID, C++, EPUB, FB2, РОЗРОБКА ЗАСТОСУНКІВ, ПАРСИНГ ФОРМАТІВ, РЕНДЕРИНГ ТЕКСТУ, МОБІЛЬНІ ЗАСТОСУНКИ.

Об’єктом роботи є процес розробки Android-застосунку для читання електронних книг різних форматів.

Метою роботи є розробки Android-застосунку для читання електронних книг форматів EPUB та FB2, з використанням Java та C++.

Було досліджено особливості формату EPUB та FB2, можливості мови програмування C++ для обробки даних та Java для відображення даних у Android, також було розроблено архітектуру застосунку, що включає модулі парсингу форматів, відображення даних на Android, та модулі для зв’язку Java та C++.

Результатом роботи є Android-застосунок, що забезпечує читання електронних книг у форматі EPUB та FB2 із парсінгом EPUB та FB2 файлів, відображенням тексту та зображень, можливістю навігації по сторінках.

E-BOOKS, ANDROID, C++, EPUB, FB2, APPLICATION DEVELOPMENT, FORMAT PARSING, TEXT RENDERING, MOBILE APPLICATIONS.

The object of the work is the process of developing an Android application for reading e-books of various formats.

The purpose of the work is to develop an Android application for reading e-books of EPUB and FB2 formats, using Java and C++.

The features of the EPUB and FB2 formats were investigated, the capabilities of the C++ programming language for data processing and Java for displaying data in Android, and the application architecture was also developed, which includes modules for parsing formats, displaying data on Android, and modules for communicating Java and C++.

The result of the work is an Android application that provides reading e-books in EPUB and FB2 formats with parsing of EPUB and FB2 files, displaying of text and images, and the ability to navigate pages were implemented.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

EPUB – Electronic publication (Електронні публікації)

IDPF – (International Digital Publishing Forum) Міжнародний форум цифрових видавців

FB2 – Fictional book 2 (Художня книга 2)

OPF – Open Packaging Format (відкритий формат пакування)

XML – Extensible Markup Language (розширювана мова розмітки)

XHTML – Extensible Hyper Text Markup Language (розширювана мова гіпертекстової розмітки)

HTML – Hyper Text Markup Language (мова гіпертекстової розмітки)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

NCX – Navigation Control file for XML (Файл керування навігацією для XML)

PDF – Portable Document Format (Портативний формат документа)

JNI – Java Native Interface (Рідний інтерфейс Java)

URI – Uniform Resource Identifier (Уніфікований ідентифікатор ресурсу)

MIME – Multipurpose Internet Mail Extensions (Багатоцільові розширення Інтернет-пошти)

ВСТУП

Застосунки для читання електронних книг, без жодного сумніву, є одними з найбільш значущих інструментів у сучасному цифровому світі, де можливість швидко і зручно отримати доступ до різноманітної літератури стала надзвичайно важливою. Завдяки широкому спектру форматів електронних книг, починаючи від звичайного .txt і закінчуючи золотим стандартом EPUB, необхідною стає потреба в розробці та використанні застосунків для їхнього читання.

Окремо потрібно виділити аспект, пов’язаний з обробкою зображень у контексті електронних читалок, що зокрема включає в себе якісне відображення обкладинок електронних книг. Оскільки майже кожна сучасна книга містить хоча б одне зображення, зазвичай це обкладинка, для збереження комфорту при читанні електронних книг, у порівнянні зі звичайними, цей аспект повинен бути врахований, при створенні застосунку для читання електронних книг.

Актуальність роботи над цим питанням полягає у необхідності постійного пошуку найбільш оптимальних рішень, спрямованих на забезпечення високого рівня комфорту при використанні застосунків для читання електронних книг, а також їх універсальності та ефективності, у нашому швидко мінливому сучасному світі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Особливості EPUB-формату

EPUB (electronic publication) – це відкритий стандарт Міжнародного форуму цифрових видавців (IDPF) для створення та розповсюдження цифрових публікацій, наприклад електронних книг. Вміст EPUB є «перекомпонованим», що означає, що до нього можна отримати доступ на будь-якому з численних пристроїв для читання електронних книг, які підтримують стандарт наприклад: Kindle, Sony Reader, Nook, Kobo, тощо, а також на більшості смартфонів і планшетів.

Документ EPUB складається з OPF, XML, XHTML, HTML, CSS, NCX та файлів зображення у єдиному сумісному форматі файлів, скомпресованих за допомогою ZIP, для легкого розповсюдження та публікації. Внутрішню структури EPUB можна легко перевірити, відкривши файл .epub за допомогою файлового архіватору, наприклад WinRAR (рис. 1.1). Де кожний файл index_split_(№).html є окремою главою.

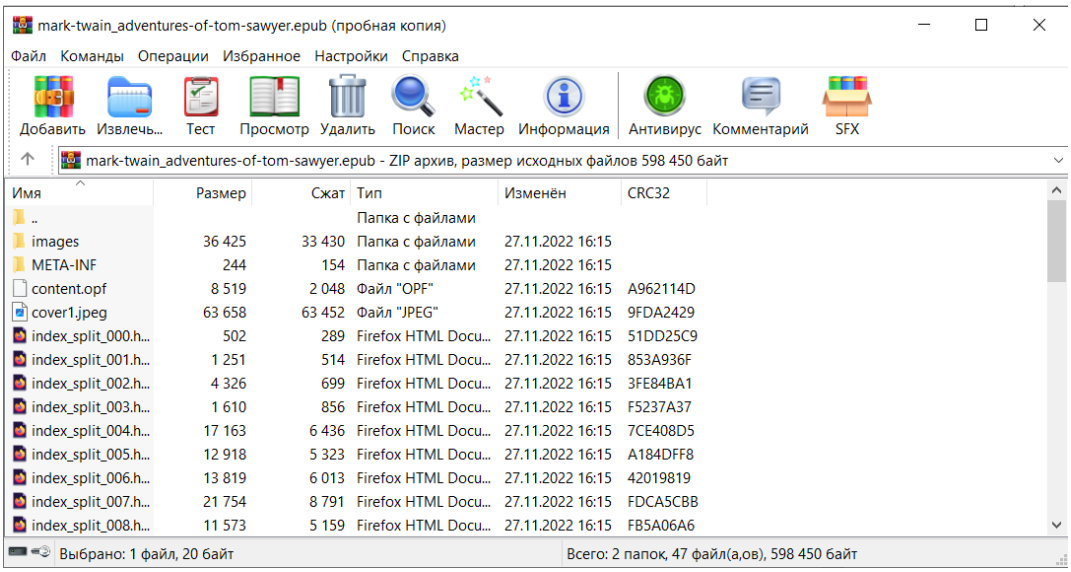

Рисунок 1.1 – Відкриття mark-twain_adventures-of-tom-sawyer.epub у WinRar

Одним з головних файлів є mimetype, у якому зазначено, що цей архів *насправді* є публікацією EPUB. Вміст цього текстового файлу має бути «application/epub+zip» і нічого іншого. Таким чином система читання EPUB гарантує, що він зможе обробити електронну книгу.

Необхідно, щоб файл mimetype був першим файлом, знайденим у zip-архіві. Це може здатися дивним обмеженням, але насправді це обмеження походить від формату файлу Open Document, який був джерелом формату файлу EPUB. Це пояснюється тим, що якщо програма читає перші байти архіву, вона завжди знаходитиме ті самі «магічні числа». Це може замінити визначення формату, якщо розширення файлу .epub є ненадійним. Практична проблема з цією вимогою полягає в тому, що неможливо створити належний файл EPUB за допомогою простого інструменту zip: загальні інструменти не можуть гарантувати, що файл типу MIME буде першим в архіві.

Наступним файлом, який треба відкрити для читання EPUB, є container.xml, що завжди знаходиться у папці META-INF, його єдиною метою є надання шляху до файлу content.opf, він не містить жодної іншої інформації (рис. 1.2).

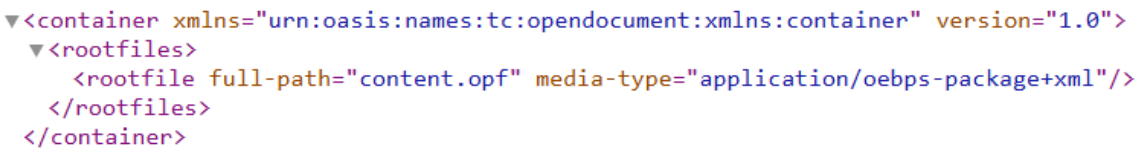

Рисунок 1.2 – Відкриття container.xml у браузері

Далі потрібно перейти за вказаним шляхом та відкрити файл content.opf, який містить усю необхідну інформацію для обкладинки: назву, опис, та зображення обкладинки, окрім цього він також містить структуру того, як потрібно читати EPUB у контейнерах manifest та spine (рис. 1.3).

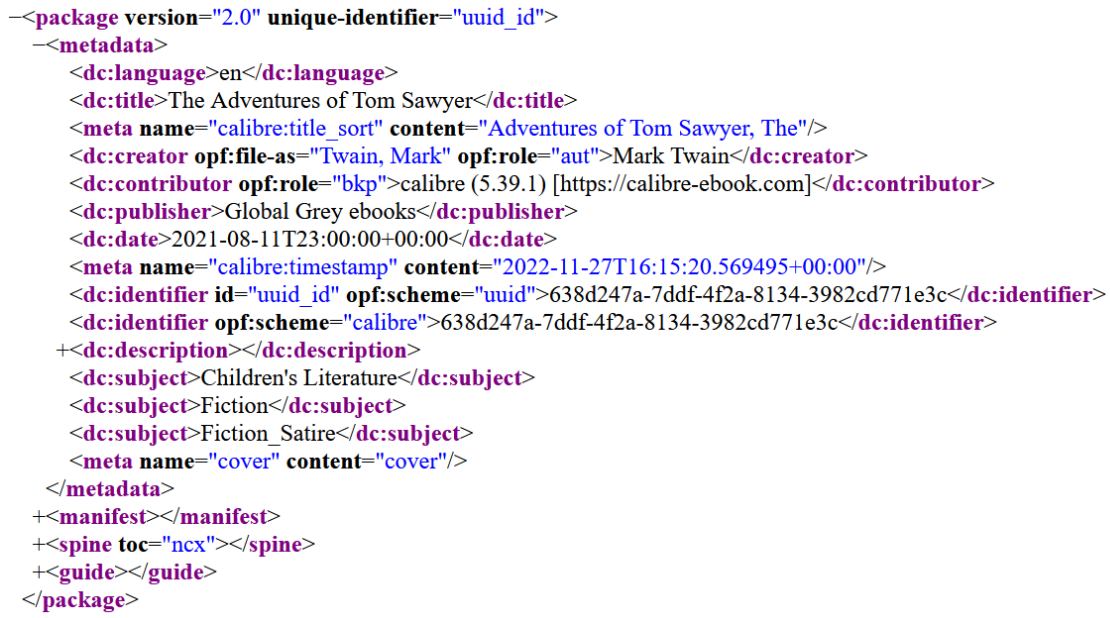

Рисунок 1.3 – Відкриття content.opf у браузері

Контейнер manifest – це вичерпний перелік усіх ресурсів публікації або відтворення, включаючи текстові розділи (x)html, зображення та відео або аудіофайли, шрифти, сценарії, файли css. Система зчитування оброблятиме лише ті файли, які вона знайде в списку, і за їхнім типом носія (так званий тип MIME) знає, що вона може їх обробити. З властивостей, оголошених для кожного елемента, система читання також дізнається його тип, напр. якщо файл відповідає документу навігації, зображенню обкладинки, векторній графіці або скрипту. Якщо система читання не може обробити ресурс, оскільки його формат дещо екзотичний, вона знайде тут резервний ресурс, який він може обробити замість цього. Приклад контейнеру manifest наведено нижче (рис. 1.4).

Контейнер spine як вказує його назва, це «основа», де система читання знаходить порядок читання за замовчуванням усіх «розділів» публікації. Оскільки ці розділи публікації можуть насправді не відображати розділи книги, кожен елемент послідовності називається елементом корінця.

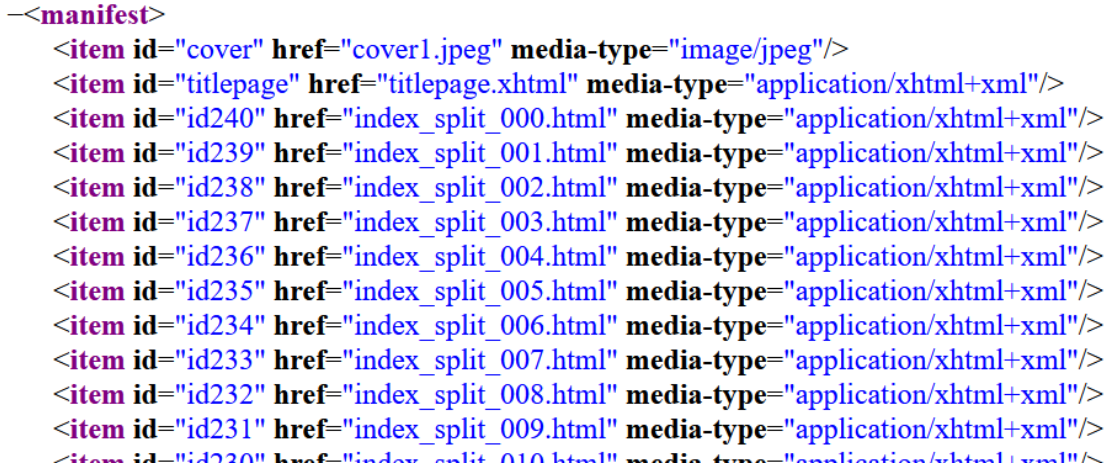

Рисунок 1.4 – Контейнер manifest у content.opf у браузері

Кожен елемент spine містить посилання на елемент manifest. Корінні елементи можна оголошувати як «нелінійні», тобто вони не відображаються у звичайному потоці, але до них можна отримати доступ з іншого елемента як додатковий вміст, наприклад, спливаючий вміст. Приклад контейнеру spine наведено нижче (рис. 1.5).

Рисунок 1.5 – Контейнер spine у content.opf у браузері

Також існує NCX, що іноді зустрічається в контейнерах EPUB 3, але це застарілий спосіб оголошення навігаційного документа EPUB 2. Деякі автори EPUB 3 все ще вважають за краще включати його, щоб системи читання EPUB 2 могли обробити публікацію. Система читання EPUB 3 не матиме до нього доступу, тому його не буде розписано у деталях.

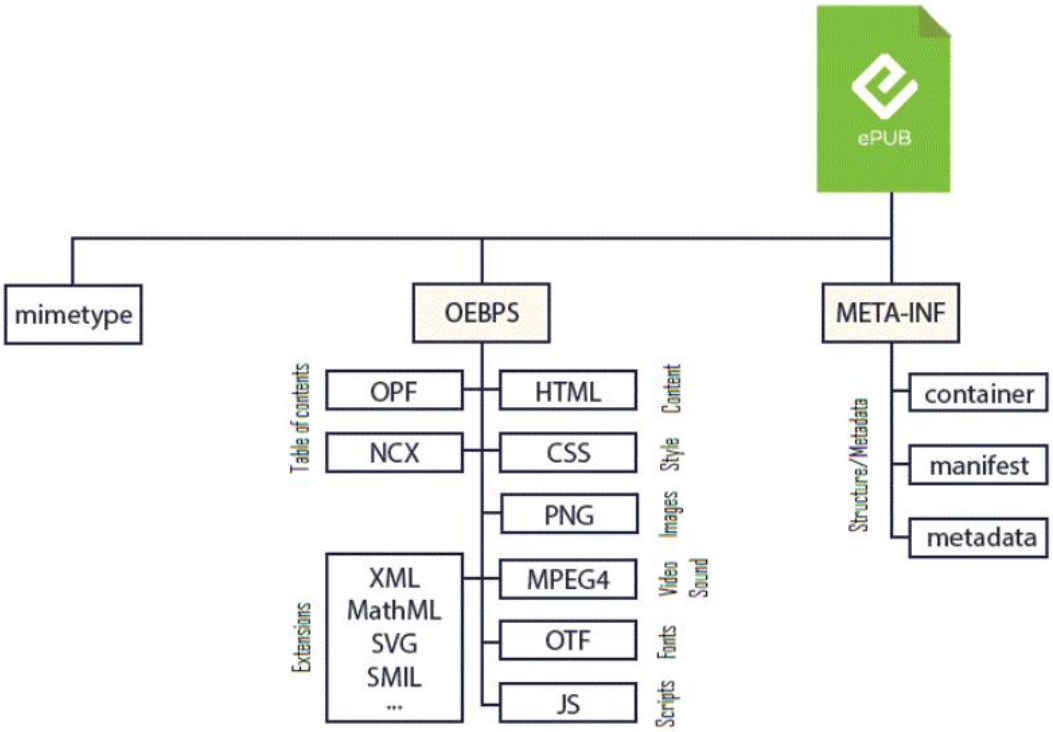

Рисунок 1.6 – Візуалізація структури EPUB

1.2 Особливості FB2-формату

FB2 (Fiction book 2) – це відкритий формат електронних книг на основі XML, що відноситься до більшого формату FictionBook, який виник в Росії і набув популярності в Росії та Україні, в першу чергу завдяки піратським сайтам. Файли FictionBook мають розширення .fb2 або .fb3 залежно від версії. FictionBook2 і FictionBook3 відрізняються двома аспектами: FB2 складається з одного файлу XML, із зображеннями, вбудованими як «бінарні» блоки в кодуванні base64, таким чином усуваючи потребу в додаткових файлах. FB3 має структуру каталогів, подібну до структури формату EPUB, і включає макет і вбудовування мультимедійного вмісту.

Формат FictionBook2 описує структуру електронної книги, а не визначає її макет. Наприклад, для епіграфів, віршів і цитат є спеціальні теги. Усі метадані електронної книги, як-от ім’я автора, назва та видавець, також

присутні у файлі електронної книги. Це робить формат зручним для автоматичної обробки, індексування, керування колекцією електронних книг і дозволяє легко автоматизовано конвертувати в інші формати. FB3 додає можливості компонування та усуває потребу вставляти зображення як блоки, закодовані Base64, але здебільшого залишається вірним концепції, встановленій у FB2.

Ось загальний опис елементів, дещо змінений з сайту FB2 та доповнений аналізом файлу. Це лише приклади, а не повний список підтримуваних тегів. Файл взагалі не стиснутий, це звичайна розмітка .txt, але деякі програми для читання підтримують його архівування для зменшення розміру. Всі дані знаходяться в одному файлі. Існує стандартний XML-заголовок, що складається з (рис. 1.7).

```
<?xml version="1.0" encoding="UTF-8"?>
<FictionBook xmlns="http://www.gribuser.ru/xml/fictionbook/2.0" xmlns:l="http://www.w3.org/1999/xlink">
```

Рисунок 1.7 – Стандартний XML-заголовок у FB2

Елементи таблиці стилів безпосередньо під кореневим каталогом містять таблиці стилів; їхній атрибут type містить тип MIME таблиці стилів. Рекомендується включити одну таблицю стилів text/css, щоб полегшити конвертацію в інші формати, якщо в документі використовуються будь-які стилі.

Розділ опису містить метадані для електронної книги. Він поділений на чотири основні категорії: title-info, document-info, publish-info та custom-info. Зазвичай присутні лише перші дві. Коментарі дозволені за допомогою <! -comment -! >. Зразок title-info виглядає так (рис. 1.8).

```
    <title-info>
        <genre>antique</genre>
        <author><first-name>J.</first-name><middle-name>R. R.</middle-name><last-name>Tolkien</last-name></author>
        <book-title>The Lord of the Rings</book-title>
        <coverpage><image l:href="#img_0"/></coverpage>
        <lang>en</lang>
        <keywords>classics, adventure, fantasy, mystery, fairytale, friendship, children-literature</keywords>
        <sequence name="The Lord of the Rings" number="2"/>
        <annotation><p>### Amazon.com Review
A Christian can almost be forgiven for not reading the Bible, but there's no salvation for a fantasy fan who hasn't read the gospel of
the genre, J.R.R. Tolkien's definitive three-book epic, the Lord of the Rings (encompassing *The Fellowship of the Ring* , *The Two
Towers* , and *The Return of the King* ), and its charming precursor, *The Hobbit*. That many (if not most) fantasy works are in some
way derivative of Tolkien is understood, but the influence of the Lord of the Rings is so universal that everybody from George Lucas to
Led Zeppelin has appropriated it for one purpose or another.
Not just revolutionary because it was groundbreaking, the Lord of the Rings is timeless because it's the product of a truly top-shelf
mind. Tolkien was a distinguished linguist and Oxford scholar of dead languages, with strong ideas about the importance of myth and
story and a deep appreciation of nature. His epic, 10 years in the making, recounts the Great War of the Ring and the closing of
Middle-Earth's Third Age, a time when magic begins to fade from the world and men rise to dominance. Tolkien carefully details this
transition with tremendous skill and love, creating in the Lord of the Rings a universal and all-embracing tale, a justly celebrated
classic. *\--Paul Hughes*</p></annotation>
    </title-info>
```

Рисунок 1.8 – Title-info у FB2

Можна ввести кілька розділів жанру. Можна ввести кілька розділів автора. Також можна використовувати <middle-name> та <nickname>. Анотація (опис) може складатися з кількох абзаців. Поле дати може мати необов’язкове поле значення, яке призначене для машинного зчитування. Другий розділ – це інформація про документ, яка може виглядати так (рис. 1.9).

```
<document-info>
    <author><first-name>J.</first-name><middle-name>R. R.</middle-name><last-name>Tolkien</last-name></author>
    <program-used>calibre 4.11.2</program-used>
    <date>6.1.2021</date>
    <id>d8c12bfe-6c68-41bc-a0ac-e5f46fe742b8</id>
    <version>1.0</version>
</document-info>
<publish-info>
    <publisher>http://adapted-english-books.site/</publisher>
    <year>2009</year>
</publish-info>
```

Рисунок 1.9 – Document-info у FB2

Блок Publish-info використовується, коли документ створюється з друкованого видання. Він має кілька бібліографічних полів, таких як назва книги, видавець, місто, рік та ISBN.

Наступною важливою частиною документа FictionBook є тіло (body), яке містить власне текст книги. Перший елемент тіла завжди є основним піддокументом у книзі. Наступні тіла можна використовувати для зберігання виносок, коментарів та інших матеріалів, які не належать до звичайного

текстового потоку в книзі. Кожен елемент тіла може містити необов'язкову назву, необов’язковий епіграф та принаймні один розділ.

Існує два різних типи розділів. Один містить лише інші підрозділи, а інший містить фактичні текстові абзаци. У поточній версії стандарту FictionBook неможливо змішувати підрозділи та абзаци в одному контейнері. Кожен розділ може мати кілька необов’язкових полів заголовка: назву, епіграф, зображення та анотацію. Після них має бути принаймні один елемент типу абзацу для текстових розділів або принаймні один підрозділ для інших.

Весь текст у документі зберігається в розділі тіла. Наступні елементи типу абзацу: p, v та subtitle. Елемент <empty-line/>, який не має вмісту, використовується для вставки одного рядка вертикального пробілу. Кілька складніших контейнерів створюються з цих основних елементів: title (містить будь-яку кількість p та порожніх рядків), annotation, poem, cite, epigraph. Тег <section> використовується для позначення розділу, а <title> відображає назву та використовується для створення змісту. Зразок може виглядати так (рис. 1.10).

```
<body>
<section>
<empty-line/><empty-line/>
<p><image l:href="#img_1"/></p>
<empty-line/>
<p><image l:href="#img_2"/></p>
<empty-line/>
<p>THE LORD</p>
<empty-line/>
<p>
  OF THE RINGS</p>
<empty-line/>
<p><strong>BY</strong></p>
<p><strong>J.R.R. TOLKIEN</strong></p>
<empty-line/><empty-line/>
<p><strong>ILLUSTRATED BY</strong></p>
<p><strong>Alan Lee</strong></p>
<empty-line/><empty-line/>
<p><image l:href="#img_3"/></p>
<empty-line/>
<p><image l:href="#img_4"/></p>
<empty-line/>
</section>
<section>
```

Рисунок 1.10 – Body у FB2

1.3 Інші популярні в Україні формати

PDF (портативний формат документів), стандартизований як ISO 32000, – це формат файлів, розроблений Adobe у 1992 році для представлення документів, включаючи форматування тексту та зображень, у спосіб, не залежний від прикладного програмного забезпечення, апаратного забезпечення та операційних систем. На основі мови PostScript кожен PDF-файл містить повний опис плоского документа з фіксованим макетом, включаючи текст, шрифти, векторну графіку, растрові зображення та іншу інформацію, необхідну для його відображення. PDF бере свій початок у «Проєкті Камелот», ініційованому співзасновником Adobe Джоном Ворноком у 1991 році. PDF був стандартизований як ISO 32000 у 2008 році. Останнє видання як ISO 32000-2:2020 було опубліковано в грудні 2020 року.

PDF-файли можуть містити різноманітний вміст, окрім простого тексту та графіки, включаючи елементи логічної структури, інтерактивні елементи, такі як анотації та поля форм, шари, мультимедіа (включно з відеовмістом), тривимірні об’єкти, що використовують U3D або PRC, і різні інші формати даних. Специфікація PDF також передбачає шифрування та цифрові підписи, вкладення файлів і метадані для забезпечення робочих процесів, які потребують цих функцій.

MOBI, що спочатку був розширенням формату PalmDOC шляхом додавання певних HTML-тегів до даних (див. EBook HTML). Багато документів у форматі MOBI все ще використовують цю форму. Однак існує також версія цього формату файлу з високим рівнем стиснення, яка стискає дані більшою мірою запатентованим способом. Режим вищого стиснення використовує схему кодування Хаффмана, яку називають Huff /cdic алгоритм.

PalmDOC використовує методи стиснення LZ77, реалізацію для PalmDOC можна знайти на Github. Файли DOC можуть містити лише

стиснутий текст. Формат не допускає жодного форматування тексту. Завдяки цьому файли залишаються невеликими відповідно до філософії Palm.

Алгоритми LZ77 досягають стиснення шляхом заміни частин даних посиланнями на відповідні дані, які вже пройшли через кодер і декодер. Збіг кодується парою чисел, яка називається парою довжина-відстань, що еквівалентно вислову «кожен із наступних символів довжини дорівнює символу, що знаходиться на точній відстані від нього в нестисненому потоці».

У форматі PalmDoc пара довжина-відстань завжди кодується двобайтовою послідовністю. З 16 біт, які складають ці два байти, 11 біт йдуть на кодування відстані, 3 йдуть на кодування довжини, а решта два використовуються, щоб переконатися, що декодер може визначити перший байт як початок такої двобайтової послідовності. Дані PalmDOC завжди поділяються на блоки по 4096 байт (розмір без стиснення), і блоки обробляються незалежно; під час стиснення або розпакування блоку інформація з попередніх чи наступних блоків не потрібна.

TXT файли – це звичайні текстові документи ASCII, які також називають «простим текстом». За межами англомовних країн TXT також може означати розширений текст.

У сучасних комп’ютерних системах основною одиницею інформації є байт, який складається з восьми бітів. Біт може бути увімкнено або вимкнено. Існує 256 можливих комбінацій увімкнення та вимкнення в рядку з восьми бітів, отже, є 256 можливих представлень у байті. Файли TXT складаються з цих основних символів, які називаються символами; перш за все ті, що визначені в наборі символів ASCII.

ASCII є підмножиною стандартних наборів символів ISO-8859-1 і UTF-8, а також Windows-1252 і багатьох інших. Він використовує перші 128 символів у цих стандартах.

Текстові файли часто мають розширення .txt, особливо якщо вони містять досить довгий документ. Зазвичай передбачається, що вони використовують шрифти фіксованої ширини, такі як Courier. В інших випадках використовується більш семантичне розширення, наприклад .log для файлу журналу, створеного комп'ютером, або .eml для електронного листа. Текстові файли також можна називати etext. Fanfiction використовує чисті текстові файли.

1.4 Постановка задачі

Ця робота зосереджена на розробці мобільного застосунку для операційної системи Android, який дозволяє читати електронні книги у форматах EPUB та FB2.

Об'єктом роботи є процес розробки Android-застосунку для читання електронних книг різних форматів.

Метою роботи є розробки Android-застосунку для читання електронних книг форматів EPUB та FB2, з використанням Java та C++.

Для досягнення цієї мети необхідно виконати:

– аналіз існуючих форматів електронних книг (EPUB, FB2) та вибір способів їх обробки;

– ознайомлення з інструментами розробки Android-застосунків (Java, Android SDK, NDK, JNI);

– реалізація модулів парсингу EPUB та FB2-файлів;

– розробка графічного інтерфейсу користувача з можливістю навігації по книзі, зміни розміру шрифтів, кольору фону тощо;

– проведення тестування застосунку на різних пристроях з Android.

2 ПРОЄКТУВАННЯ ЗАСТОСУНКУ

2.1 Загальне проєктування проєкту

Проєкт мобільного застосунку для читання електронних книг форматів EPUB та FB2 було розроблено з урахуванням вимог до сучасних Android-застосунків, тобто зручності, продуктивності, підтримки основних форматів та гнучкості архітектури. Основною метою цього проєкту є створення застосунку, з можливостю відображення електронних книг у форматах EPUB та FB2, включно з текстом, зображеннями та реалізованою логікою перегортання сторінок.

Для досягнення поставленої мети була обрана гібридна архітектура з використанням двох мов програмування: Java для інтеграції з Android-системою та C++ для усього іншого, тобто функцій обробки кни та перегортання сторінок. Компоненти C++ було інтегровано з Java за допомогою технології JNI (діаграму JNI наведено на рис 2.1).

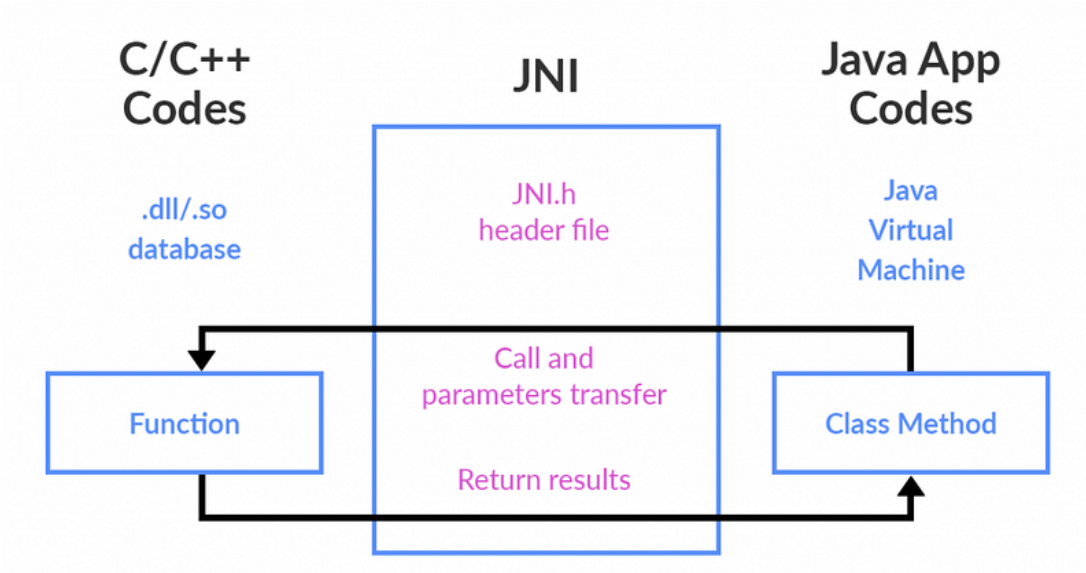

Рисунок 2.1 – Діаграма загального проєктування

У цій архітектурі Java-частина виконує функції інтерфейсу користувача (UI – user interface), взаємодії з файловою системою Android, обробки дій

користувача та керування життєвим циклом застосунку. У той час як C++ частина відповідає за обробку електронних книг форматів EPUB та FB2, включно з розпакуванням архівів, обробкою XML файлів, витягуванням з них тексту та зображень, формуванням сторінок для відображення на пристрої користувача, підтримання роботи з Java класами та передачу аргументів у Java частину.

Однією з ключових переваг такого підходу є високий рівень масштабованості проєкту у майбутньому. Це забезпечує можливість в майбутньому додати підтримку нових форматів електронних книг, наприклад MOBI або PDF, або поліпшити навігацію у електронних книгах обраних користувачем, наприклад можливість переходу на сторінку за вказаним номером.

Застосунок має реалізувати наступні основні функції: відкриття файлів форматів EPUB та FB2 із внутрішньої пам’яті пристрою, обробка структури електронної книги та розбиття тексту та зображень на сторінки, відображення тексту та зображень, отриманих після обробки електронної книги, на екрані пристрою користувача, можливість перегортання сторінок, з метою навігації по електронній книзі, кешування зображень та оптимізація ресурсів.

Таким чином, загальне проєктування застосунку (рис. 2.2) забезпечує розбиття необхідних для реалізації функцій між компонентами застосунку, що підвищує рівень масштабованості і легкості тестування проєкту. Що безпосередньо впливає на необхідні витрати часу на додавання нових функцій та можливостей до застосунку.

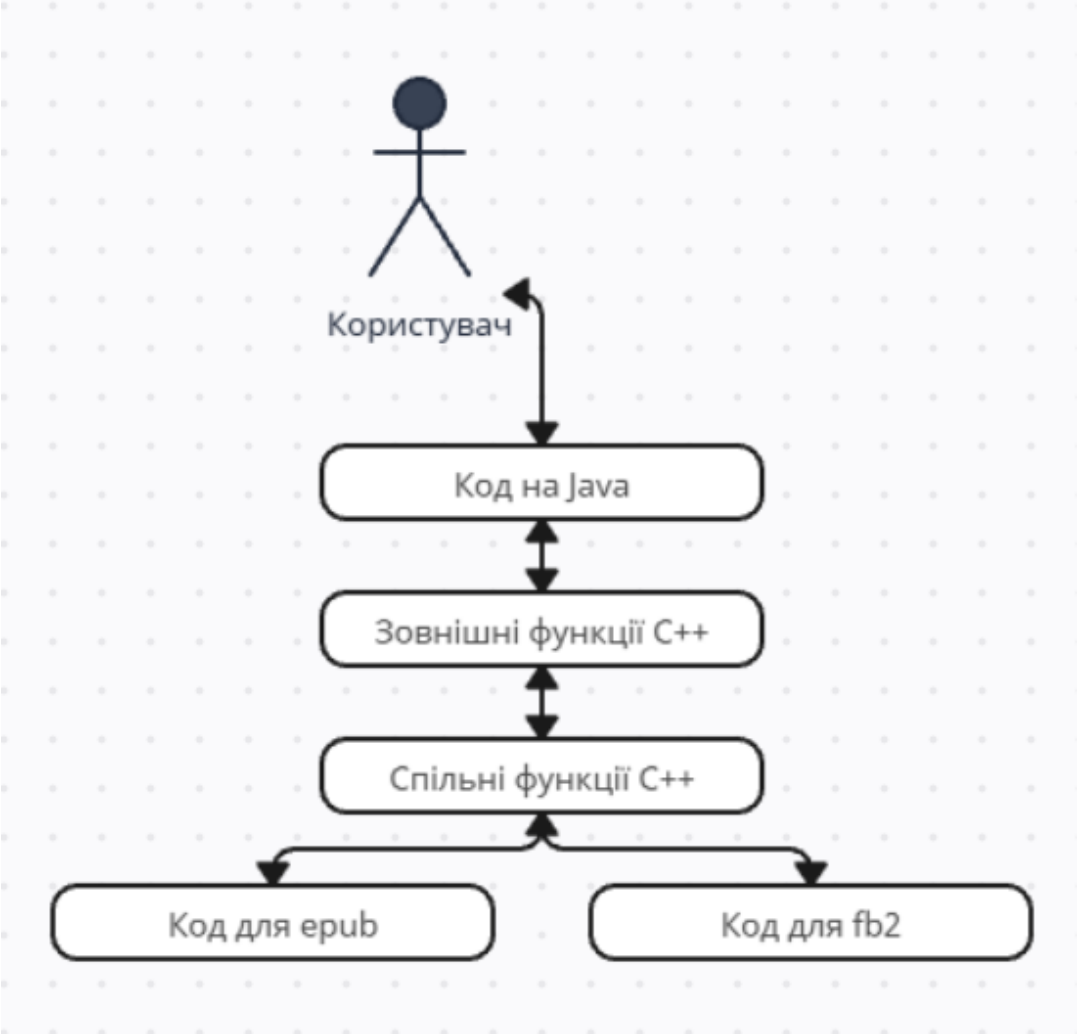

Рисунок 2.2 – Діаграма загального проєктування

2.2 Проєктування Java частини

Java-частина застосунку існує у якості точки входу для користувача. Саме тут було реалізовано інтерфейс користувача, логіку обробки подій та виклики функція C++ частини. Головні елементи Java частини – це активність для відкриття електронної книги форматів EPUB та FB2 та відображення її вмісту, система обробки натискань на екран, взаємодія з файловою системою за допомогою стандартного провідника Android (рис 2.3)

Запуск застосунку запускає стандартний інтерфейс вибору файлу, реалізований за допомогою стандартного провідника Android. Після вибору

файл копіюється у кеш пристрою користувача, з метою полегшення обробки файлу.

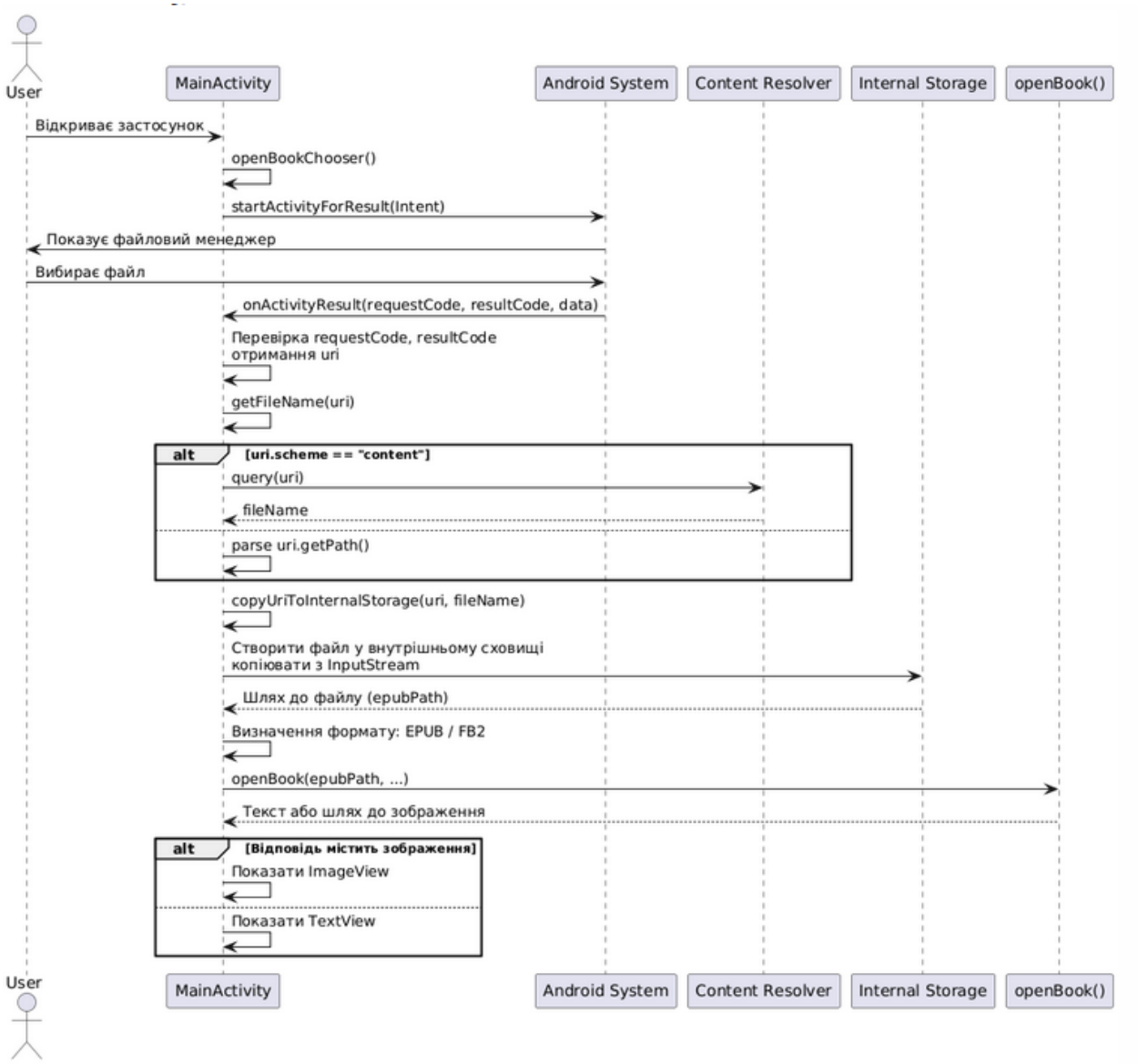

Рисунок 2.3 – Діаграма відкриття електронної книги

У Java-частині було реалізовано методи, що через JNI викликають необхідні функції з C++ частини. Спочатку Java-код отримує результати роботи функцій з C++ частини у вигляді рядків, які відображаються на екран користувача за допомогою стандартних компонентів Android UI: TextView та ImageView. Перегортання сторінок електронної книги здійснюється за допомогою натискання на ліву та праву половини екрану пристрою

користувача, які змінюють індекс сторінки та отримують новий текст або зображення для відображення з C++ частини.

Отже, Java-частина проєкту забезпечує створення зручного та зрозумілого інтерфейсу користувача, реалізує логіку взаємодії зі стандартною файловою системою Android та виклик функцій з C++ частини.

2.3 Проєктування C++ частини

2.3.1 Проєктування зовнішніх функцій

Зовнішні функції в C++ частині було реалізовано згідно до специфікації JNI, яка дозволяє Java коду взаємодіяти з бібліотеками на мовах C. Для використання JNI кожна зовнішня функція C++ частини має мати чітку сигнатуру, яка включає в себе вказівник на середовище виконання JNIEnv*, для правильного перетворення аргументів з Java до C++, об’єкт класу, а також аргументи, передані з Java. Назва зовнішніх функцій має починатися з "Java_", за яким іде повна назва класу рівня Java (з крапками, заміненими підкресленням), і, нарешті, сама назва методу.

Усі зовнішні функції C++ частини є лише "мостами" між Java та C++ частинами і не містять будь-якої логіки обробки електронних книг (рис 2.4). Вони лише отримують параметри з Java, перетворюють їх на відповідні параметри у C++ (наприклад з jstring у std::string), викликають внутрішні функції парсингу, отримують результат і перетворюють його до вигляду Java, повертають його назад у Java-код у зручному для сприйняття вигляді.

Використання цього підходу дозволяє зберегти чистоту архітекрури та полегшити можливість майбутнього переносу на інші платформи, наприклад Apple або Windows.

Рисунок 2.4 – Діаграма JNI функції changePage

2.3.2 Проєктування функцій для передавання аргументів між Java та C++

У процесі розробки застосунку для читання електронних книг форматів EPUB та FB2 на Android виникає потреба у взаємодії між Java частиною, що відповідає за інтерфейс користувача та обробку подій, і C++ частиною, яка реалізує обробку електронних книг форматів EPUB та FB2 та навігацію у них. Щоб забезпечити таку взаємодію, було використано Java Native Interface (JNI), який дозволяє викликати функції на мові C++ з Java і обмінюватися аргументами та посиланнями на аргументи між цими мовами.

Для проєктування функцій передачі аргументів між Java частиною та C++ частиною було створено допоміжні механізми, які виконують перетворення даних між типами, згідно до потреб середовищ. У C++ частині застосовуються стандартні типи C++, такі як std::string, int, double, тоді як у Java використовуються jstring, об’єкти-обгортки (Integer, Double), для передачі посилань на аргументи, та користувацькі класи.

Було створено функції для перетворення рядків. Оскільки передача даних у текстовому форматі, таких як шляхи до файлів або тексти глав,

вимагає конвертації між jstring та std::string для можливості обробки їх середовищем. У результаті це забезпечує коректну обробку кодувань, при переході між різними мовами.

Також було створено функції для роботи з числовими обгортками. Оскільки у Java об’єкти типу Integer, Double вимагають спеціально створені класи, для реалізації можливості передання аргументів за посиланням, яка необхідна для надання можливості C++ частині змінювати ці аргументи. У C++ для роботи з цими класами проєктуються функції, які отримують об’єкт, витягують або змінюють значення, відповідно від вимоги, необхідні поля через JNI.

Було створено функції для зворотної передачі значень. Коли значення повинно бути змінене у C++ частині та передане назад у Java частину, застосунок використовує спеціальні функції для оновлення полів створених класів. Наприклад, оновлення аргументу класу IntegerHolder після перегортання сторінок у електронній книзі.

У результаті чіткого розділення цих функцій, основна логіка роботи програми є простою для розширення та легкою для підтримки. Отже коли виникне потреба додати нові типи параметрів або підтримку нових об’єктів Java, буде достатньо просто додати нову функцію-конвертер не змінюючи вже існуючи механізми. Такий підхід забезпечує надійність і масштабованість взаємодії між цими мовами програмування в межах Android застосунку.

2.3.3 Проєктування частини для EPUB

Формат EPUB зберігає усі дані: текст, зображення та форматування, у вигляді ZIP-архіву, який містить файли HTML або XHTML, метадані, зображення та інші ресурси. З метою обробки EPUB-файлів у C++ частині

програми було реалізовано модуль, який відповідає за відкриття EPUB файлів та їх обробку

Відкриття EPUB як ZIP-архіву виконується з використанням бібліотеки libzip, далі виконується витягнення шляху до основного файлу опису content.opf з container.xml, який обробляється для отримання порядку глав (spine) та відповідних їм файлів у маніфесті (manifest).

Читання вмісту глави в HTML або XHTML форматі та подальший розбір відбувається за допомогою бібліотеки tinyxml2, через побудову дерева елементів HTML або XHTML документа та рекурсивний пошук тегів та зображень у главі для формування тексту глави, та, у випадку зображень, формування шляху до відповідного файлу для подальшого витягнення.

Проєктування реалізації застосунку передбачає обробку можливих варіацій структури документа, наприклад наявність вкладених елементів div, br, або тегів img. Важливим компонентом є рекурсивна функція, тобто функція яка викликає саму себе під час її виконання, яка виконує парсинг HTML або XHTML та обробляє дані, відокремлюючи текст від інших елементів, повертаючи текст у вигляді суцільного рядка. Якщо під час обробки даних зустрічається зображення, функція повертає тег <img> для того, щоб застосунок міг відобразити його окремо.

Для забезпечення можливості комфортного читання тексту було реалізовано функцію для додавання переносів через певну кількість символів, а також функції для перевірки, чи містить рядок літери, для уникання вставки зайвих службових символів, здебільшого існуючих у електронній книзі переносів.

Отже, у частині для роботи з EPUB (рис. 2.5 та 2.6) було реалізовано повний цикл: від відкриття архіву до передачі структурованого тексту та зображень до Android частини.

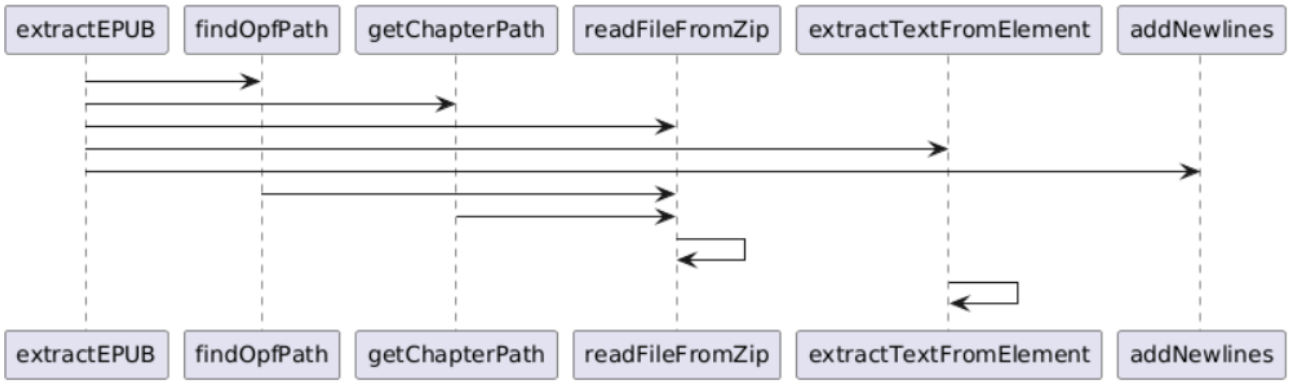

Рисунок 2.5 – Діаграма частини для отримання тексту з EPUB

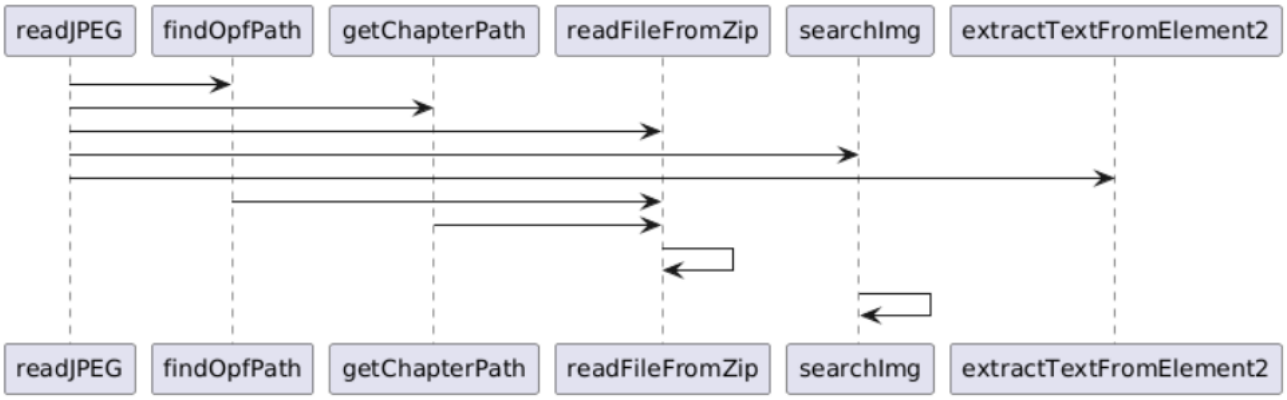

Рисунок 2.6 – Діаграма частини для отримання зображень з EPUB

2.3.4 Проєктування частини для FB2

Формат FB2 тобто FictionBook є XML-документом, що зберігає структуру тексту електронної книги, включно з главами, параграфами та вбудованими графічними елементами у вигляді елементів <binary>. Для обробки FB2-файлів у C++ було реалізовано окремий модуль, який відповідає за відкриття файлу, зчитування даних з нього та обробки цих файлів

Зчитуванні дані з файлу формату FB2 виводяться у вигляді одного рядка за допомогою стандартного потоку вводу/виводу – ifstream. Розбір XML-документа виконується за допомогою бібліотеки tinyxml2. Розбір включає обробку вкладених елементів, рекурсивний обхід усіх вузлів XML для отримання тексту з параграфів (<p>) та глав (<section>), а також

видалення службових тегів окрім та <section>, які залишаються для формування логічної структури. Також розбір включає обробку елементів <image>, з метою знаходження посилань l:href на відповідні id у тегах <binary>. Там містяться зображення закодовані у форматі base64. Отже було розроблено декодування base64-зображень та збереження їх у кеш застосунку за допомогою JNI, використовуючи jCacheDir. Останнім кроком у розборі XML-документа є розбиття вмісту книги на глави відповідно до елементів <section>.

Процес обробки FB2 було поділено на два основні етапи. Спочатку виконується обхід XML-дерева з метою формування списку слів і службових тегів, зокрема <section>, </section>, , . Відокремлення цих тегів дозволяє потім поділити текст на глави та абзаци. Під час обходу застосунок також виявляє вбудовані зображення, для витягування яких було зроблено функцію, що шукає відповідний id у блоці <binary> та зберігає зображення у кеш застосунку.

Наступним кроком у обробці файлу FB2 є побудова списку глав на основі розмітки, та форматування кожної глави, а саме: додавання табуляцій, нових рядків, а також вставка тегів (<img>), для повідомлення Java-частині про місця, де потрібно підставити зображення.

Отже результатом обробки електронної книги формату FB2 є текст окремої глави, який був сформований з урахуванням логічної структури, що відповідає стандарту електронних книг формату FB2, з зображеннями, які було збережено окремо, та маркерами для їхнього відображення. Ця частина (рис. 2.7) дозволяє забезпечити підтримку електронних книг формату FB2 у застосунку з рівнем якості, що дорівнює рівню якості підтримки електронних книг EPUB.

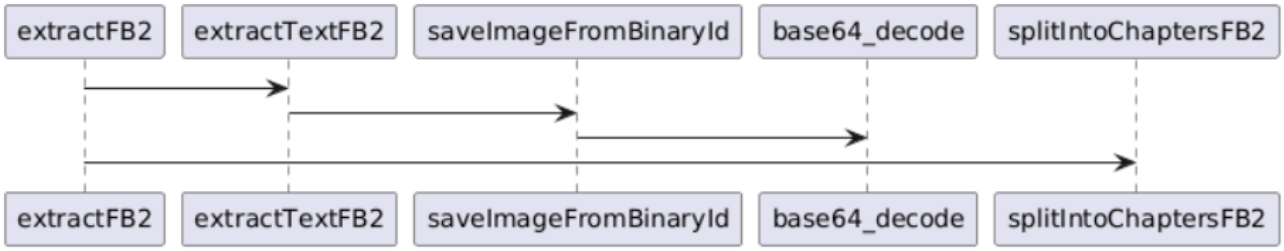

Рисунок 2.7 – Діаграма частини для отримання інформації з FB2

2.3.5 Проєктування спільної для форматів частини

Спільна для форматів частина є центральним логічним вузлом, який пов’язує роботу з конкретними форматами електронних книг, тобто EPUB та FB2, у єдину систему. Наявність цієї частини дозволяє забезпечити необхідний, для підтримки можливості подальшого розширення застосунку, рівень абстракції над форматами, й забезпечує єдиний інтерфейс для зовнішніх функцій C++, незалежно від формату електронної книги. Основна мета – це дати змогу Android застосунку можливість однаково працювати з текстом та зображеннями для усіх форматів електронних книг.

Усі електронні книги незалежно від формату, застосунок обробляє єдиним механізмом навігації та поділу на сторінки. Основний принцип роботи цього механізму полягає в тому, щоб брати окрему главу електронної книги, розбивати її на сторінки за фіксованою висотою по кількості рядків, і здійснювання навігації між сторінками або главами електронної книги.

Було створено функцію returnPages (рис. 2.8), що відповідає за розбиття всієї глави по сторінкам, згідно до обмеження на кількість рядків. Також якщо зустрічається тег <img>, який вказує на наявність зображення, то сторінка одразу закінчується та для зображення відводиться окрема сторінка. Кожна сторінка зберігається у векторі std::wstring pages.

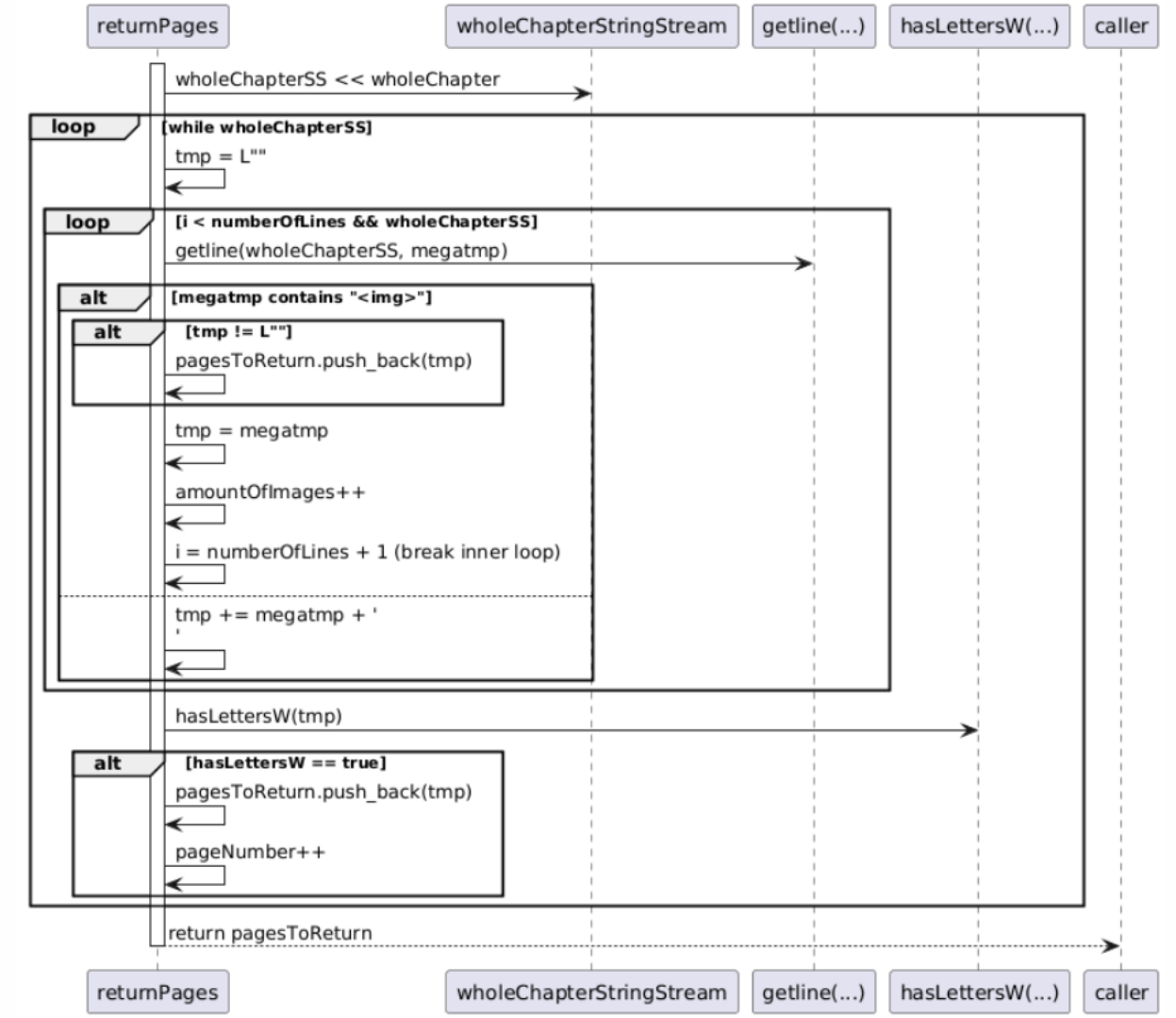

Рисунок 2.8 – Діаграма функції returnPages

Було створено функцію returnTextOrImage(), що визначає, чи поточна сторінка містить текст чи зображення. У випадку якщо сторінка не містить тегу зображення, який повідомляє про його наявність, текст сторінки повертається без змін. Інакше передається повідомлення про необхідність запуску механізму відображення зображення.

Було створено головну функцію changePage (рис. 2.9 та 2.10), що забезпечує навігацію між сторінками та главами. Якщо, при перегортанні сторінки, було досягнуто межу глави, то ця функція завантажує наступну або попередню главу, в залежності від напрямку перегортання.

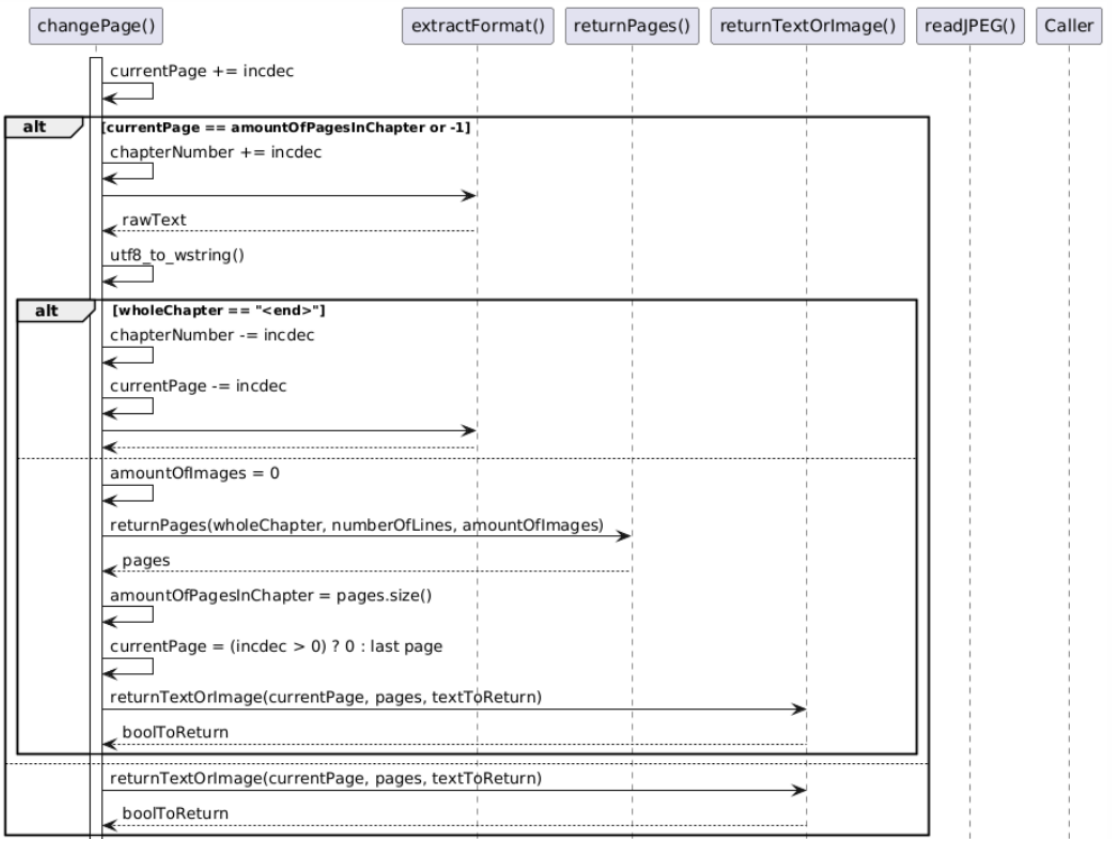

Рисунок 2.9 – Діаграма першої половини функції changePage

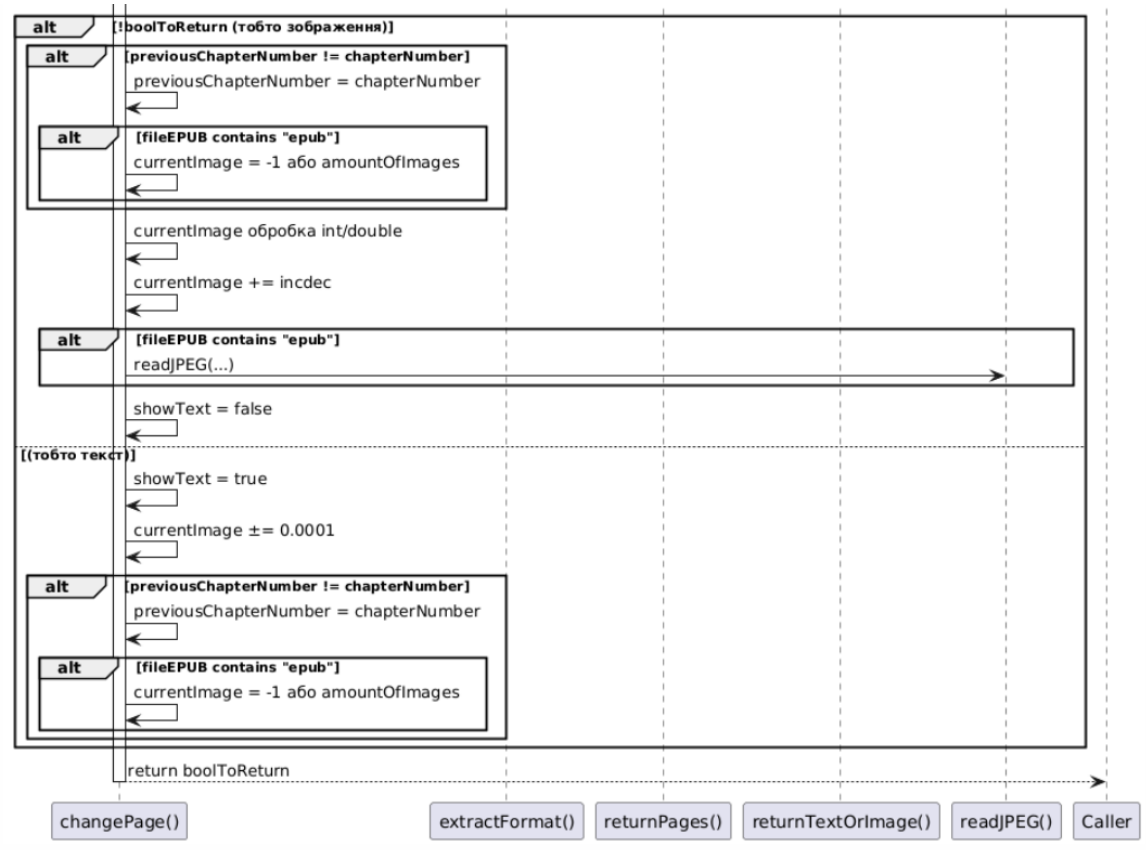

Рисунок 2.10 – Діаграма другої половини функції changePage

Ця функція також визначає потрібність відображення тексту чи зображення. Для виконання перегортання сторінок функція оновлює відповідні аргументи: номер поточної сторінки, загальну кількість сторінок у поточній главі, кількість зображень, тощо.

Для початку відкриття книги було створено функцію openingBook(), що починає з переходу уперед і назад по сторінках, для забезпечення коректного відкриття книги та перевірки, чи перша сторінка містить текст, чи одразу потрібно відображати зображення. Це забезпечує універсальний механізм відкриття електронної книги.

Отже, використання цього підходу дозволяє не лише спростити логіку виводу тексту чи зображень, але й забезпечує можливість розширення для підтримки інших форматів електронних книг у майбутньому. Функції спільної для форматів частини можуть бути легко, з мінімальними змінами, використані для роботи з частиною обробки нових форматів.

3 РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

3.1 Реалізація Java частини

Java-частина застосунку виконує функції графічного інтерфейсу користувача (GUI), обробки подій, що пов’язані з життєвим циклом Android-застосунку, та забезпечення взаємодії з C++-частиною, що пов’язана з Java-частиною за допомогою механізму JNI (Java Native Interface). Основне завдання Java-компонента – створення зручного інтерфейсу для відкриття електронних книг, перегортання сторінок і передачі параметрів до C++ модуля.

Оскільки у C++ немає прямого доступу до змінних Java за посиланням, для створення можливості передачі аргументів між Java і C++ було розроблено спеціальні класи-обгортки (рис. 3.1)

```
12 usages
class IntegerHolder {
    3 usages
    private int value;

    3 usages
    public IntegerHolder(int value) { this.value = value; }

    public int getValue() { return value; }

    public void setValue(int value) { this.value = value; }
}

4 usages
class DoubleHolder {
    3 usages
    private double value;

    1 usage
    public DoubleHolder(double value) { this.value = value; }

    public double getValue() { return value; }

    public void setValue(double value) { this.value = value; }
}
```

Рисунок 3.1 – Класи-обгортки

Ці класи відіграють роль контейнерів для значень змінних, які мають бути змінені, зі збереженням змін, на C++-стороні, для подальшого використання в Java-частині. Це є стандартною практикою для обходу обмеження Java щодо передачі примітивних типів за посиланням при використанні JNI.

Всі об’єкти, що треба передавати через JNI, оголошуються у класі MainActivity, який є основною діяльністю Android-застосунку (рис. 3.2). Оскільки об’єкти оголошені у цьому класі зберігаються в пам’яті протягом усього життєвого циклу роботи програми, таке рішення дозволяє зберігати дані між викликами різних методів.

```
private ActivityMainBinding binding;

1 usage
int increse = 1;
1 usage
int decrese = -1;

3 usages
IntegerHolder chapterNumber = new IntegerHolder( value: 0);
3 usages
IntegerHolder currentPage = new IntegerHolder( value: 0);
3 usages
DoubleHolder currentImage = new DoubleHolder( value: 0);
3 usages
IntegerHolder previousChapterNumber = new IntegerHolder( value: -1);
4 usages
String epubPath;
```

Рисунок 3.2 – Класи-обгортки

Після запуску застосунку одразу викликається метод onCreate (рис. 3.3). Цей метод є необхідним для правильної роботи Android-застосунку, оскільки це основна точка входу для логіки ініціалізації. Метод onCreate() виконує базову логіку запуску програми, що відбувається лише один раз за весь час дії.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    clearAppCache();

    super.onCreate(savedInstanceState);

    binding = ActivityMainBinding.inflate(getLayoutInflater());
    setContentView(binding.getRoot());

    openBookChooser();
}
```

Рисунок 3.3 – Функція onCreate

Цей метод є точкою входу до логіки застосунку після його запуску. У першу чергу він викликає функцію, що виконує очищення кешу програми. Очищення кешу потрібно не лише для ефективного використання простору в пам’яті пристрою, але й для запобігання конфліктів, викликаних залишковими файлами від попередніх сесій. Оскільки, після того як попередня електронна книга була оброблена та збережена у тимчасових файлах, присутність попередніх тимчасових файлів може призвести до помилок при її повторному відкритті або завантаженні. Для вирішення цієї проблеми, застосунок викликає метод очищення кешу одразу після запуску.

Функція clearAppCache() (рис. 3.4) відповідає за очищення кешу, тобто видалення всіх тимчасових файлів, розміщених у стандартному каталозі кешу застосунку. Ця функція була реалізована за допомогою API Android, а саме за допомогою отримання посилання на директорію кешу, використовуючи метод getCacheDir().

Рекурсивне видалення вмісту директорії кешу, було реалізовано за допомогою функції deleteDir(File dir) (рис. 3.5). Якщо об’єкт File – це існуюча папка, функція викликає саму себе для всіх файлів і папок, що містяться у цій директорії. Після очищення вмісту отриманої папки, функція видаляє саму отриману папку. Саме завдяки цій рекурсії, тобто самовиклику функції,

застосунок забезпечує повне очищення усієї структури кешу, незалежно від рівня глибини.

```
1 usage
private void clearAppCache() {
    try {
        File cacheDir = getCacheDir();
        if (cacheDir != null && cacheDir.isDirectory()) {
            deleteDir(cacheDir);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Рисунок 3.4 – Функція clearAppCache

```
2 usages
private boolean deleteDir(File dir) {
    if (dir != null && dir.isDirectory()) {
        String[] children = dir.list();
        if (children != null) {
            for (String child : children) {
                boolean success = deleteDir(new File(dir, child));
                if (!success) {
                    return false;
                }
            }
        }
    }
    return dir.delete();
}
```

Рисунок 3.5 – Функція deleteDir

На наступному етапі роботи застосунку, необхідно надати користувачу можливість обрати електронну книгу для читання. Для цього було реалізовано функцію для виклику провідника Android – openBookChooser() (рис. 3.6). Ця функція формує Intent з аргументом Intent.ACTION_GET_CONTENT, що дозволяє обрати потрібний користувачу файл із файлової системи пристрою. Використання стандартного файлового

провідника Android дозволяє мати чіткий та зрозумілий інтерфейс вибору книги, бо відповідний інтерфейс вже встановлений на пристрої.

```
125
126     private static final int PICK_BOOK_FILE = 1;
127
128     public void openBookChooser() {
129         Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
130         intent.setType("*/*");
131         intent.addCategory(Intent.CATEGORY_OPENABLE);
132         startActivityForResult(Intent.createChooser(intent, title: "Виберіть книгу"), PICK_BOOK_FILE);
133     }
```

Рисунок 3.6 – Функція openBookChooser

Після обирання користувачем файлу, застосунок повертає результат вибору як об’єкт Intent у метод onActivityResult() (рис. 3.7). Цей метод обробляє дані, повернуті стандартним файловим провідником Android, зокрема отримує Uri обраного файлу. Саме цей URI, застосунок використовує для виконання обробки та підготовки файлу до завантаження у C++ частину.

Оскільки для подальшої роботи з обраною електронною книгою необхідно знати її точну назву та розширення, було реалізовано допоміжну функцію getFileName(Uri uri) (рис. 3.8), яка визначає повне ім’я обраного користувачем файлу. Ця функція запитує системний ContentResolver, для отримання метаданих про файл, включаючи його назву, MIME-тип, розмір тощо. У випадку, якщо у цьому підході виникає помилка, застосунок використовує резервний варіант – визначення імені з URI, за допомогою парсингу останнього сегмента шляху.

З метою підвищення безпеки, стабільності, та полегшення роботи C++ частини, обрану користувачем електронну книгу копіюють у внутрішнє сховище застосунку, викликаючи функцію copyUriToInternalStorage(Uri uri, String fileName) (рис. 3.9). Ця функція зчитує байти з відкритого потоку InputStream, відкритого за допомогою URI, та записує їх у локальний файл у директорії getCacheDir(), тобто у кеш.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == PICK_BOOK_FILE && resultCode == RESULT_OK && data != null) {
        Uri uri = data.getData();

        String filename = getFileName(uri);
        epubPath = copyUriToInternalStorage(uri, filename);

        String lowerCaseName = filename.toLowerCase();
        boolean isEpub = lowerCaseName.endsWith(".epub");
        boolean isFb2 = lowerCaseName.endsWith(".fb2");

        if (isEpub || isFb2) {
            String str = openBook(epubPath, chapterNumber, currentPage, currentImage, previousChapterNumber, getCacheDir().getAbsolutePath());

            ImageView imageView = findViewById(R.id.imageView);
            TextView tv = findViewById(R.id.sampleText);

            if (str.contains("temp_image.png") || (str.contains(".png") && str.contains("cache"))) {
                Bitmap bitmap = BitmapFactory.decodeFile(str);
                imageView.setImageBitmap(bitmap);
                imageView.setVisibility(View.VISIBLE);
                tv.setVisibility(View.GONE);
            } else {
                tv.setText(str);
                imageView.setVisibility(View.GONE);
                tv.setVisibility(View.VISIBLE);
            }
        }
    }
}
```

Рисунок 3.7 – Функція onActivityResult

```
1 usage
private String getFileName(Uri uri) {
    String result = null;
    if (uri.getScheme().equals("content")) {
        try (Cursor cursor = getContentResolver().query(uri, projection: null, selection: null, selectionArgs: null, sortOrder: null)) {
            if (cursor != null && cursor.moveToFirst()) {
                result = cursor.getString(cursor.getColumnIndexOrThrow(OpenableColumns.DISPLAY_NAME));
            }
        }
    }
    if (result == null) {
        result = uri.getPath();
        int cut = result.lastIndexOf( ch: '/');
        if (cut != -1) {
            result = result.substring( beginIndex: cut + 1);
        }
    }
    return result;
}
```

Рисунок 3.8 – Функція getFileName

За допомогою цього підходу C++ частина може уникнути багатьох обмежень, викликаних через відсутність прав доступу, оскільки для роботи із кешом, тобто із папкою внутрішнього сховища не потрібно жодних спеціальних дозволів.

```
1 usage
public String copyUriToInternalStorage(Uri uri, String fileName) {
    try {
        InputStream in = getContentResolver().openInputStream(uri);
        File outFile = new File(getFilesDir(), fileName);
        OutputStream out = new FileOutputStream(outFile);

        byte[] buffer = new byte[1024];
        int length;
        while ((length = in.read(buffer)) > 0) {
            out.write(buffer,  off: 0, length);
        }

        in.close();
        out.close();

        return outFile.getAbsolutePath();
    } catch (IOException e) {
        e.printStackTrace();
        return null;
    }
}
```

Рисунок 3.9 – Функція copyUriToInternalStorage

Оскільки жодна електронна книга не обмежується лише однією сторінкою, було реалізовано функції для створення можливості гортати сторінки. Для цього було використано обробку дотиків у функції onTouchEvent(MotionEvent event) (рис. 3.10). Ця функція фіксує координати дотику та обчислює їх щодо центру екрану, що далі інтерпретується як команди "наступна сторінка", якщо було натиснуто на праву половину екрану, або "попередня сторінка" якщо було натиснуто на ліву половину екрану. Після розпізнавання команди викликається нативна функція, тобто функція що оголошена у C++, changePage, яка формує текстове представлення або посилання на зображення наступної або попередньої сторінки.

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    ImageView imageView = findViewById(R.id.imageView);
    TextView tv = findViewById(R.id.sampleText);

    if (event.getAction() == MotionEvent.ACTION_DOWN) {
        float x = event.getX(); // координата X касания
        float screenWidth = getResources().getDisplayMetrics().widthPixels;

        if (x > screenWidth / 2) {
            // Правая половина экрана
            String str = changePage(getAssets(), epubPath , chapterNumber, currentPage, currentImage, previousChapterNumber, increse, getCacheDir().getAbsolutePath());
            if (str.contains("temp_image.png") || (str.contains(".png") && str.contains("cache"))) {
                Bitmap bitmap = BitmapFactory.decodeFile(str);  // /data/user/0/com.example.myapplication/cache/temp_image.png
                imageView.setImageBitmap(bitmap);
                imageView.setVisibility(View.VISIBLE);          // /data/user/0/com.example.myapplication/cache/1.png
                tv.setVisibility(View.GONE);
            } else {
                tv.setText(str);
                imageView.setVisibility(View.GONE);
                tv.setVisibility(View.VISIBLE);
            }
        }
        if (x < screenWidth / 2) {
            String str = changePage(getAssets(), epubPath, chapterNumber, currentPage, currentImage, previousChapterNumber, decrese, getCacheDir().getAbsolutePath());
            if (str.contains("temp_image.png") || (str.contains(".png") && str.contains("cache"))) {
                Bitmap bitmap = BitmapFactory.decodeFile(str);
                imageView.setImageBitmap(bitmap);
                imageView.setVisibility(View.VISIBLE);
                tv.setVisibility(View.GONE);
            } else {
                tv.setText(str);
                imageView.setVisibility(View.GONE);
                tv.setVisibility(View.VISIBLE);
            }
        }
    }
    return super.onTouchEvent(event);
}
```

Рисунок 3.10 – Функція onTouchEvent

Для взаємодії Java-частини з C++-частиною було реалізовано за допомогою механізму JNI дві нативні функції:

Функція для відкриття книги (рис. 3.11).

```
1 usage
public native String openBook(String path, IntegerHolder chapterNumber,
    IntegerHolder currentPage, DoubleHolder currentImage, IntegerHolder previousChapterNumber, String cacheDir);
```

Рисунок 3.11 – Функція openBook

Ця функція передає шлях до електронної книги обраної користувачем (path), а також посилання на аргументи, що зберігають поточний стан читання: номер глави (chapterNumber), номер сторінки (currentPage), номер зображення (currentImage) та номер попередньої глави (previousChapterNumber). C++-частина використовує ці аргументи для розпакування, відкриття і парсингу книги, а також для повернення першої сторінки в текстовому форматі чи шляху до зображення у текстовому форматі.

Функція для зміни сторінки (рис. 3.12).

```
2 usages
public native String changePage(AssetManager javaAssetManager, String jCacheDir,
    IntegerHolder chapterNumber, IntegerHolder currentPage, DoubleHolder currentImage,
    IntegerHolder previousChapterNumber, Integer incdec, String cacheDir);
```

Рисунок 3.12 – Функція changePage

Цей метод передає до C++-частини аргументи поточного стану читання, а також об’єкт AssetManager, що дозволяє коду на C++, отримувати доступ до ресурсів Java та Android, застосунок використовує цей доступ для зміни переданих із Java аргументів, та отримання доступу до кешу. Параметр incdec вказує напрямок перегортання сторінки: вперед (increase, що дорівнює одиниці) або назад (decrease, що дорівнює мінус одиниці). Результатом виконання цієї функції є текст або зображення з наступної або попередньої сторінки.

3.2 Реалізація C++ частини

C++-частина забезпечує основну обробку електронних книг форматів EPUB та FB2, а також передання тексту та зображень, у Java для відображення у інтерфейсі. Комунікація C++ частини з Java частиною відбувається через JNI, де аргументи приймаються у вигляді Java-об’єктів.

3.2.1 Реалізація зовнішніх функцій

Зовнішні функції, що викликаються з Java через JNI, мають наступну сигнатуру (рис. 3.13). Зовнішні функції отримують, переробляють аргументи та викликають функції зі спільної для форматів частини, переробляють та відправляють інформацію назад до Java.

```
JNIEXPORT jstring JNICALL MainActivity.openBook(JNIEnv* env,jobject MainActivity thiz,
   jstring jpath, jobject IntegerHolder jChapterNumber, jobject IntegerHolder jCurrentPage,
   jobject DoubleHolder jCurrentImage, jobject IntegerHolder jPreviousChapterNumber, jstring jCacheDir)
JNIEXPORT jstring JNICALL MainActivity.changePage(JNIEnv* env,jobject MainActivity thiz,
   jobject AssetManager javaAssetManager, jstring jpath, jobject IntegerHolder jChapterNumber,
   jobject IntegerHolder jCurrentPage, jobject DoubleHolder jCurrentImage,
   jobject IntegerHolder jPreviousChapterNumber,jobject Integer jIncdec, jstring jCacheDir)
```

Рисунок 3.13 – Зовнішні функції

3.2.2 Реалізація функцій для передавання аргументів між Java та C++

Для правильної інтеграції між частинами із різними мовами було реалізовано низку допоміжних функцій (рис. 3.14 – 3.17).

```
std::string jstringToString(JNIEnv* env, jstring jStr) {
    if (!jStr)
        return "";

    const jclass stringClass = env->GetObjectClass(jStr);
    const jmethodID getBytes = env->GetMethodID(stringClass, "getBytes", "(Ljava/lang/String;)[B");
    const jbyteArray stringJbytes = (jbyteArray)env->CallObjectMethod(jStr, getBytes, env->NewStringUTF("UTF-8"));

    size_t length = (size_t)env->GetArrayLength(stringJbytes);
    jbyte* pBytes = env->GetByteArrayElements(stringJbytes, NULL);

    std::string ret = std::string((char*)pBytes, length);
    env->ReleaseByteArrayElements(stringJbytes, pBytes, JNI_ABORT);

    env->DeleteLocalRef(stringJbytes);
    env->DeleteLocalRef(stringClass);
    return ret;
}
```

Рисунок 3.14 – Функція для перетворення Java-рядка у std::string

```
jstring stdStringToJstring(JNIEnv* env, const std::string& str) {
    jstring jStr = env->NewStringUTF(str.c_str());
    return jStr;
}
```

Рисунок 3.15 – Функція для зворотнього перетворення

```
int integerHolderToInt(JNIEnv* env, jobject chapter)
{
    jclass integerHolderClass = env->GetObjectClass(chapter);
    jfieldID valueFieldID = env->GetFieldID(integerHolderClass, "value", "I");
    jint currentValue = env->GetIntField(chapter, valueFieldID);
    int valueToReturn = currentValue;
    return valueToReturn;
}
```

Рисунок 3.16 – Функція для витягу значення з IntegerHolder

```
void changeJavaIntLinkTo(JNIEnv* env, jobject obj, int FinalValue) {
    jclass integerHolderClass = env->GetObjectClass(obj);
    if (integerHolderClass == nullptr) {
        return;
    }
    jfieldID valueFieldID = env->GetFieldID( clazz: integerHolderClass,  name: "value",  sig: "I");
    if (valueFieldID == nullptr) {
        return;
    }
    jint currentValue = env->GetIntField(obj,  fieldID: valueFieldID);
    jint newValue = FinalValue;
    env->SetIntField(obj,  fieldID: valueFieldID,  value: newValue);
}
```

Рисунок 3.17 – Функція для зміни значення в IntegerHolder

Так само (рис. 3.17) зроблено витяг double із DoubleHolder та зміну значення в DoubleHolder.

3.2.3 Реалізація частини для EPUB

Для реалізації можливості зчитування вмісту EPUB-файлів з боку C++ частини застосунку було використано бібліотеки libzip та tinyxml2. Основною метою частини для обробки EPUB є відкриття архіву EPUB, знаходження потрібної глави, зчитування тексту та зображень з XHTML або HTML файлу та конвертація отриманих даних у текст, для відображення у Java-інтерфейсі.

Основна функція для роботи з EPUB, – це extractEPUB, яка на вхід приймає шлях до EPUB-файлу та номер глави. Ця функція (рис. 3.18) виконує послідовність кроків:

- відкриття EPUB-файлу як архіву;
- знаходження шляху до content.opf;
- визначення потрібної глави (XHTML або HTML файлу);
- зчитування вмісту потрібної глави;
- рекурсивне вилучення тексту з потрібної глави;

– форматування та повернення фінального результату у вигляді std::string;

– відкриття EPUB-файлу здійснюється за допомогою zip_open від libzip, що відкриває EPUB як звичайний ZIP-архів;

```
std::string extractEPUB(std::string epub_path, int chapterNumber) {

    const char* epubFile = epub_path.c_str();
    int err = 0;
    zip_t* archive = zip_open(epubFile, ZIP_RDONLY, &err);
    if (!archive) {
        return "error1";
    }
```

Рисунок 3.18 – Функція extractEPUB

Наступним кроком відбувається пошук файлу content.opf, який визначає структуру книги. Це реалізується за допомогою функції findOpfPath (рис. 3.19), яка зчитує шлях до content.opf із файлу META-INF/container.xml.

```
string findOpfPath(zip_t* archive) {
    string containerXML = readFileFromZip( epub: archive, outputDir: "META-INF/container.xml");
    if (containerXML.empty()) return "";

    XMLDocument doc;
    if (doc.Parse( xml: containerXML.c_str()) != XML_SUCCESS) {
        return "";
    }

    XMLElement* rootfile = doc.FirstChildElement( name: "container")
            ->FirstChildElement( name: "rootfiles")
            ->FirstChildElement( name: "rootfile");

    if (!rootfile) {
        return "";
    }

    return rootfile->Attribute( name: "full-path");
}
```

Рисунок 3.19 – Функція findOpfPath

Після цього функція getChapterPath визначає шлях до необхідної глави книги, використовуючи відповідні секції OPF-документа: manifest та spine. Порядок глав визначається за допомогою списку елементів <itemref> у <spine>.

Розділ книги, представлений у вигляді XHTML або HTML документа, зчитується функцією readFileFromZip, яка зчитує цей файл прямо з EPUB-архіву та повертає його вміст у вигляді рядка.

Після зчитування вмісту глави, її обробляє функція extractTextFromElement (рис. 3.20), яка рекурсивно, тобто викладаючи саму себе під час виконання, проходить по всіх дочірніх елементах вузлів <p>, замінюючи
 на пробіл, бо
 використовується для звичайних переносів, положення яких відрізняється в залежності від ширини екрану. Для віршів у цих форматах використовується тег <p>.

```
string extractTextFromElement(XMLElement* element, string& text) {
    if (!element) return "error";
    for (XMLNode* node = element->FirstChild(); node != nullptr; node = node->NextSibling()) {
        if (node->ToText()) {
            text += node->Value();
            //text += "\n";
        }
        {
            string tmpStr( s: node->Value());
            if (tmpStr.find( s: "img") != std::wstring::npos || tmpStr.find( s: "image") != std::wstring::npos)
                return "img";
        }
        if (node->ToText());
        else if (node->ToElement())
        {
            if (strcmp(node->Value(), "br") == 0) {
                text += " ";
            }
            else {
                if (extractTextFromElement( element: node->ToElement(), &: text) == "img")
                    return "img";
            }
        }
    }
    return "error";
}
```

Рисунок 3.20 – Функція extractTextFromElement

Після завершення аналізу XHTML або HTML документа, зібраний текст форматується за допомогою функції addNewlines, яка вставляє переводи рядків через кожні n символів. Це необхідно для виведення тексту оскільки фізичні пристрої мають обмежений розмір екрану.

Окремо було реалізовано обробку зображень електронної книги. З метою виведення ілюстрацій була реалізована функція readJPEG, яка витягує зображення у кеш та повертає шлях до витягнутого зображення.

Пошук зображення було реалізовано за допомогою функції searchImg (рис. 3.21), яка рекурсивно, тобто викладаючи саму себе під час виконання, обходить XML-файл в пошуках тегів <img> або <image> та повертає шлях до зображення, що знаходиться у значенні атрибутів src або xlink:href.

```
string searchImg(XMLElement* element, int& imageNumber) {
    if (!element) return "error";

    for (XMLNode* node = element->FirstChild(); node != nullptr; node = node->NextSibling()) {
        if (node->ToElement())
        {
            string tmpStr( s: node->ToElement()->Value());
            bool wereImgFound = false;

            if (tmpStr.find( s: "img") != std::wstring::npos || tmpStr.find( s: "image") != std::wstring::npos)
            {
                const char* src = node->ToElement()->Attribute( name: "src");
                const char* xlink = node->ToElement()->Attribute( name: "xlink:href");
                if (src || xlink) {
                    if (imageNumber == 0)
                    {
                        if (src)
                            return src;
                        if(xlink)
                            return xlink;
                        wereImgFound = true;
                    }
                    else
                        imageNumber--;
                }
            }
            if (!wereImgFound) {
                string tmp = searchImg( element: node->ToElement(), &: imageNumber);
                if (tmp != "error")
                    return tmp;
            }
        }
    }
    return "error";
}
```

Рисунок 3.21 – Функція searchImg

Після знаходження зображення функцією searchImg викликається функція extractFileFromZip2, яка відкриває EPUB як архів, шукає відповідний файл у ньому за допомогою readJPEG (рис. 3.22) та витягує цей файл у кеш, шлях до якого застосунок отримує від Java частини.

```
string tmpForFullImagePath = basePath + searchImg( element: p, &: tmpImageNumber);
std::string tempImagePath = jstringToString(env, jStr: jCacheDir) + "/temp_image.png";
if(extractFileFromZip2( zipPath: epub_path.c_str(), filePathInZip: tmpForFullImagePath, outputPath: tempImagePath))
    return tempImagePath;
```

Рисунок 3.22 – Частина функції readJPEG

Таким чином, усі графічні елементи електронної книги стають доступними для перегляду у межах застосунку, забезпечуючи повноцінну підтримку контенту, що міститься в EPUB-файлах.

3.2.4 Реалізація частини для FB2

Ще однією важливою частиною реалізації застосунку є підтримка обробки електронних книг представлених у форматі FictionBook 2.0 (FB2), який, незважаючи на непопулярність у світі, залишається популярним в Україні. Важливою особливістю цього формату, при роботі з ним, є зберігання всієї інформації, включно з ілюстраціями, що зберігаються у бінарному вигляді, в одному XML-файлі. Для коректної обробки таких електронних книг було реалізовано окремі для FB2 функції парсингу, декодування та розбиття вмісту на глави.

Для обробки FB2 було створено функцію extractFB2 (рис. 3.23), яка відкриває FB2 файл, зчитує його XML-вміст в рядковий потік під назвою «buffer», виконує парсинг XML-вмісту, використовуючи зовнішню бібліотеку tinyxml2, витягує текст й зображення у кеш застосунку, для полегшення робити з зображеннями у Java частині. Після успішної обробки XML-документу рекурсивно, тобто із викликанням самої себе під час виконання, викликається функція extractTextFB2, яка проходить усі елементи дерева документа, знаходячи та виділяючи текстові вузли та додаючи їх у вектор std::string під назвою «chapters». Під час обробки тегів, ці теги видаляються, окрім тегів <section> та <p> що зберігаються, оскільки це згодом дозволяє точно відновити структуру електронної книги, з главами та абзацами.

```
std::string extractFB2(std::string fb2_path, int chapterNumber, jstring jCacheDir, JNIEnv* env)
{
    std::ifstream fopen( s: fb2_path);
    if (!fopen.is_open()) {
        std::cerr << "Error: failed to open " << fb2_path << std::endl;
        return "error";
    }
    std::stringstream buffer;
    std::string line;
    while (std::getline( &: fopen,  &: line)) {
        buffer << line << "\n";
    }
    std::string xmlContent = buffer.str();
    fopen.close();
    if (xmlContent.empty()) {
        std::cerr << "Error: failed to read FB2 content" << std::endl;
        return "error";
    }
    tinyxml2::XMLDocument doc;
    if (doc.Parse( xml: xmlContent.c_str()) != tinyxml2::XML_SUCCESS) {
        std::cerr << "Error: failed to parse FB2 content: " << doc.ErrorStr() << std::endl;
        return "error";
    }
    std::vector<std::string> words;
    const tinyxml2::XMLElement* root = doc.RootElement();
    extractTextFB2( element: root,  &: words,  &: doc, jCacheDir, env);
    std::vector<std::string> chapters;
    splitIntoChaptersFB2(words,  &: chapters);
    std::string finalText = addNewlines( &: chapters[chapterNumber],  maxLineLength: 120);
    return finalText;
}
```

Рисунок 3.23 – Функція extractFB2

Особливу увагу було приділено обробці ілюстрацій. У форматі FB2 зображення за стандартом зберігаються в окремих тегах <binary> у вигляді base64-кодування, а вміщуються в текст, використовуючи посилання у вигляді тегів <image l:href="#id">. Для обробки таких елементів було створено функцію saveImageFromBinaryId (рис. 3.24), яка декодує рядки у форматі base64, використовуючи функцію base64_decode, та зберегає зображення у тимчасовий кеш застосунку, а також вставляю спеціальний маркер <img> у текст книги. Це дозволяє спільній частині на мові C++ коректно передати шлях до необхідного зображення, для його відображення.

```
void saveImageFromBinaryId(tinyxml2::XMLDocument& doc, const std::wstring& id, jstring jCacheDir, JNIEnv* env)
{
    static int imageCounter = 1;
    std::string id_utf8( first: id.begin(), last: id.end());
    const tinyxml2::XMLElement* binaryElement = doc.FirstChildElement( name: "FictionBook")->FirstChildElement( name: "binary");
    while (binaryElement)
    {
        const char* attrId = binaryElement->Attribute( name: "id");
        if (attrId && id_utf8 == attrId) {
            const char* base64Text = binaryElement->GetText();
            if (base64Text) {
                std::string base64data( s: base64Text);
                std::vector<unsigned char> decodedData = base64_decode( encoded_string: base64data);
                std::string filename = jstringToString(env, jStr: jCacheDir) + "/" + std::to_string( val: imageCounter) + ".png";
                std::ofstream outFile( s: filename, mode: std::ios::binary);
                outFile.write( s: reinterpret_cast<const char*>(decodedData.data()), n: decodedData.size());
                outFile.close();
                imageCounter++;
            }
            return;
        }
        binaryElement = binaryElement->NextSiblingElement( name: "binary");
    }
}
```

Рисунок 3.24 – Функція saveImageFromBinaryId

Необхідною для правильної обробки тексту є функція – splitIntoChaptersFB2, яка поділяє текст на окремі глави, використовуючи збережені теги <section> та <p>. Зокрема, кожна глава починається при виявленні тега <section>. Це дозволяє забезпечити використання спільного з EPUB-парсером коду, в якому кожна глава обробляється окремо, оскільки є окремим файлом у архіві. Далі функція addNewlines обробляє отриманий текст, ця функція вставляє переведення рядків для покращення читабельності, за допомогою обмеження ширини рядка до певної кількості символів.

Таким чином, C++ частина реалізує забезпечення повноцінної підтримки формату FB2, тобто не лише тексту але й графічних зображень. Всі витягнуті дані уніфіковано передаються у Java-частину застосунку через JNI, що дозволяє створення єдиного інтерфейсу користувача для роботи з різними форматами електронних книг.

3.2.5 Реалізація спільної для форматів частини

Для забезпечення спільного для різних форматів механізму навігації між сторінками, відображення тексу та зображень, що мають бути оброблені та відображені як у форматі EPUB, так і у форматі FB2, було реалізовано спільну логіку, що відповідальна за коректне перегортання користувачем сторінок в книзі, поділ глав на сторінки, відображення контенту сторінок, тобто тексту або зображень.

Метод returnPages є додатковим методом, що виконує розбиття вмісту глави книги на окремі сторінки. Глава подається як єдиний рядок std::string wholeChapter, який розбивається на сторінки за кількістю рядків numberOfLines. Водночас відбувається пошук у тексті тегів <img> та розширення .png, які є ознакою того, що сторінка має бути відведена під відображення зображення. Отже якщо така сторінка знаходиться, вона додається до вектору std::wstring pagesToReturn як окрема сторінка. Цей метод також веде підрахунок кількості зображень у главі за допомогою змінної amountOfImages, що в подальшому використовується для коректного відображення зображень.

Метод returnTextOrImage є додатковим методом, що відповідає за визначення типу контенту поточної сторінки, тобто чи потрібно відображати текст, чи зображення. Якщо сторінка не містить тегу <img> або посилання на зображення .png, вона вважається текстовою, і текст цієї сторінки повертається через посилання на аргумент textToReturn. Цей метод повертає булеве значення булеве значення – значення яке має лише два можливих стани.

Метод changePage є основним методом, що реалізує логіку перегортання сторінок електронної книги. В залежності від параметра incdec, який вказує напрямок переходу (вперед або назад), та отримав свою назву як поєднання слів increase та decrease, змінюється індекс поточної сторінки currentPage. Якщо поточна сторінка виявляється за межами кількості

сторінок поточної глави, здійснюється перехід до наступної або попередньої глави. Після завантаження нової глави, викликається функція returnPages, яка розписана вище у цьому підрозділі (рис. 3.25).

```
bool changePage(int& chapterNumber, std::wstring& wholeChapter, int& numberOfLines, std::vector<std::wstring>& pages, int& currentPage,
        int& amountOfPagesInChapter, wstring& textToReturn, int incdec, int& amountOfImages, string fileEPUB,
        int& previousChapterNumber, double& currentImage, bool& showText, int Width, int Height, jstring jCacheDir, JNIEnv* env)
{
    bool boolToReturn;
    currentPage += incdec;
    if (currentPage == amountOfPagesInChapter || currentPage == -1)
    {
        chapterNumber += incdec;
        if (fileEPUB.find( s: "epub") != std::wstring::npos)
            wholeChapter = utf8_to_wstring( str: extractEPUB( epub_path: fileEPUB, chapterNumber));
        if (fileEPUB.find( s: "fb2") != std::wstring::npos)
            wholeChapter = utf8_to_wstring( str: extractFB2( fb2_path: fileEPUB, chapterNumber, jCacheDir, env));

        if (wholeChapter == L"<end>")
        {
            chapterNumber -= incdec;
            currentPage -= incdec;

            if (fileEPUB.find( s: "epub") != std::wstring::npos)
                wholeChapter = utf8_to_wstring( str: extractEPUB( epub_path: fileEPUB, chapterNumber));
            if (fileEPUB.find( s: "fb2") != std::wstring::npos)
                wholeChapter = utf8_to_wstring( str: extractFB2( fb2_path: fileEPUB, chapterNumber, jCacheDir, env));
        }
        else
        {
            amountOfImages = 0;
            pages = returnPages(wholeChapter, numberOfLines,  &: amountOfImages);
            amountOfPagesInChapter = pages.size();

            (incdec > 0) ? currentPage = 0 : currentPage = amountOfPagesInChapter - 1;//-1 because amountOfPagesInChapter is amount of pages but it starts counting from 0

            boolToReturn = returnTextOrImage( &: currentPage,  &: pages,  &: textToReturn);
        }
    }
    else
    {
        boolToReturn = returnTextOrImage( &: currentPage,  &: pages,  &: textToReturn);
    }
```

Рисунок 3.25 – Частина функції changePage

Після цього метод визначає, чи нова сторінка містить текст чи зображення, і повертає відповідне булеве значення, булеве значення – значення яке має лише два можливих стани. Якщо це зображення, викликається функція readJPEG, яка відповідає за підготовку зображення до відображення. Для коректного рахування зображень, при переході на сторінку з текстом лічильник змінюється на 0,0001, а при переході на сторінку з зображенням лічильник округляється у необхідну сторону. Для EPUB лічильник є відносним до глави, у той час як для FB2 лічильник є абсолютним, це пов’язано з тим, що FB2 є одним файлом, коли EPUB є архівом з багатьох файлів (рис. 3.26).

```
if (!boolToReturn)
{
    if (previousChapterNumber != chapterNumber)
    {
        previousChapterNumber = chapterNumber;
        if (fileEPUB.find( s: "epub") != std::wstring::npos)
            (incdec > 0) ? currentImage = -1 : currentImage = amountOfImages;
    }
    int tmpInt = static_cast<int>(currentImage);
    if (incdec > 0)
    {
        currentImage = tmpInt;
    }
    else if (currentImage > tmpInt)
    {
        currentImage = tmpInt + 1;
    }
    currentImage += incdec;
    if (fileEPUB.find( s: "epub") != std::wstring::npos)
        readJPEG( epub_path: fileEPUB, chapterNumber, imageNumber: currentImage, jCacheDir, env);

    showText = false;
}
else
{
    showText = true;
    if(incdec > 0)
        currentImage += 0.0001;
    else
        currentImage -= 0.0001;
    if (previousChapterNumber != chapterNumber)
    {
        previousChapterNumber = chapterNumber;
        if (fileEPUB.find( s: "epub") != std::wstring::npos)
            (incdec > 0) ? currentImage = -1 : currentImage = amountOfImages;
    }
}
return boolToReturn;
```

Рисунок 3.26 – Частина функції changePage

Застосунок викликає метод openingBook (рис. 3.27) при першому відкритті книги. Цей метод виконує ініціалізацію, тобто перегортає сторінку вперед і назад для забезпечення коректного стану навігації, встановлює коректне зображення або текст на першій сторінці. У випадку зображення для FB2 та EPUB виконується різний код з метою побудови шляху до файлу .png на основі локального каталогу.

```
string openingBook(int& chapterNumber, std::wstring& wholeChapter, int& numberOfLines, std::vector<std::wstring>& pages,
    int& currentPage,int& amountOfPagesInChapter, wstring& textToReturn, int& amountOfImages, string fileEPUB,
    int& previousChapterNumber, double& currentImage, bool& showText, int Width, int Height, jstring jCacheDir, JNIEnv* env)
{
    if (!changePage( & chapterNumber,  & wholeChapter,  & numberOfLines,  & pages,  & currentPage,  & amountOfPagesInChapter,  & textToReturn,  incdec: 1,  & amountOfImages, fileEPUB, jCacheDir, env));
    if (!changePage( & chapterNumber,  & wholeChapter,  & numberOfLines,  & pages,  & currentPage,  & amountOfPagesInChapter,  & textToReturn,  incdec: -1,  & amountOfImages, fileEPUB, jCacheDir, env))
    {
        currentImage = 0;
        showText = false;
        if (fileEPUB.find( s: "epub") != std::wstring::npos)
            return readJPEG( epub_path: fileEPUB, chapterNumber,  imageNumber: currentImage, jCacheDir, env);
        if (fileEPUB.find( s: "fb2") != std::wstring::npos)
        {
            int tmpN = currentImage + 1;
            string tempImagePath = jstringToString(env,  jStr: jCacheDir) + "/" + to_string( val: tmpN) + ".png";
            return tempImagePath;
        }
    }
    else
    {
        showText = true;
        return "";
    }
}
```

Рисунок 3.27 – Функція openingBook

Найбільшою користю від використання такого підходу є можливість створити спільний алгоритм для роботи з обома форматами, незважаючи на відмінності у структурі EPUB та FB2. Таким чином, повна логіка перегляду книги, зокрема зміна сторінок, визначення типу контенту, була реалізована у вигляді спільних функцій, що є дуже важливим для майбутнього розширення застосунку.

3.3 Скріншоти роботи програми

На рисунку 3.28 можна побачити панель вибору електронних книг за допомогою стандартного провідника Android. Ця панель відкривається одразу після запуску застосунку на вкладці downloads (завантаження), з метою швидкого доступу до завантажених з інтернету книг. Використання стандартного провідника Android дозволяє забезпечити швидкий та зрозуміли інтерфейс користувача, оскільки користувачі Android вже ознайомлені з інтерфейсом стандартного провідника Android.

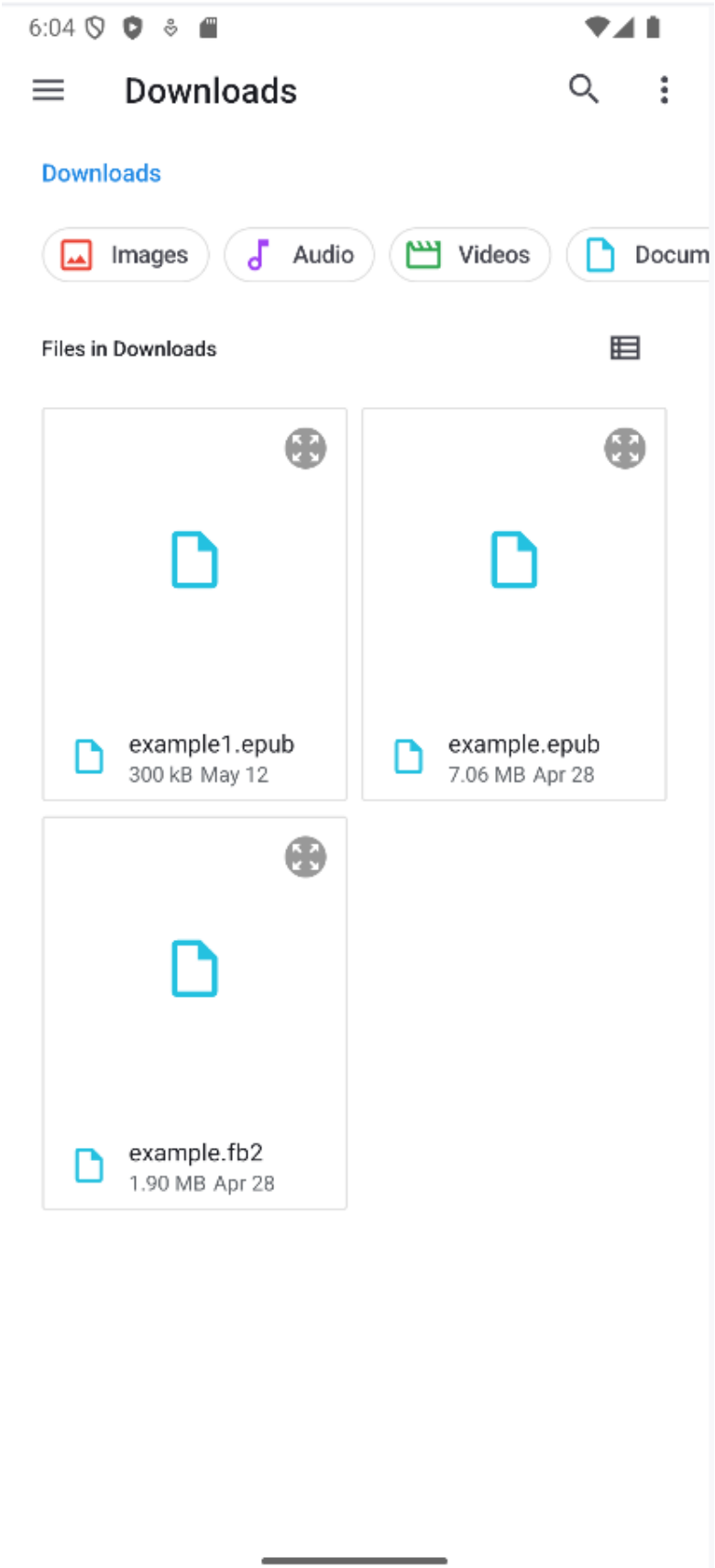

Рисунок 3.28 – Панель вибору книги.

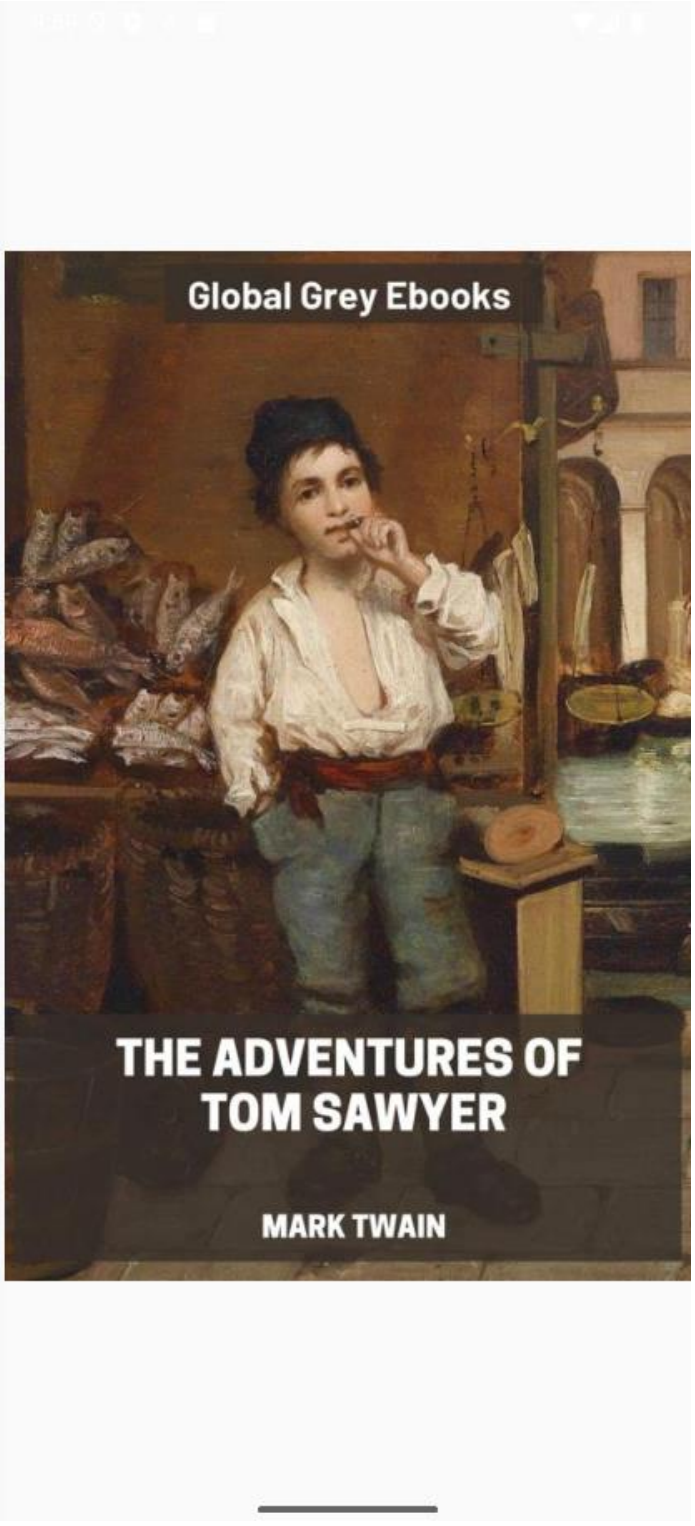

Рисунок 3.29 – відображення зображення в книгах EPUB.

На рисунку 3.29 можна побачити відображення обкладинок електронних книг формату EPUB. Отримані з електронних книг зображення одразу масштабуються відповідно до розміру екрану користувача, завдяки використанню стандартного Android класу – ImageView.

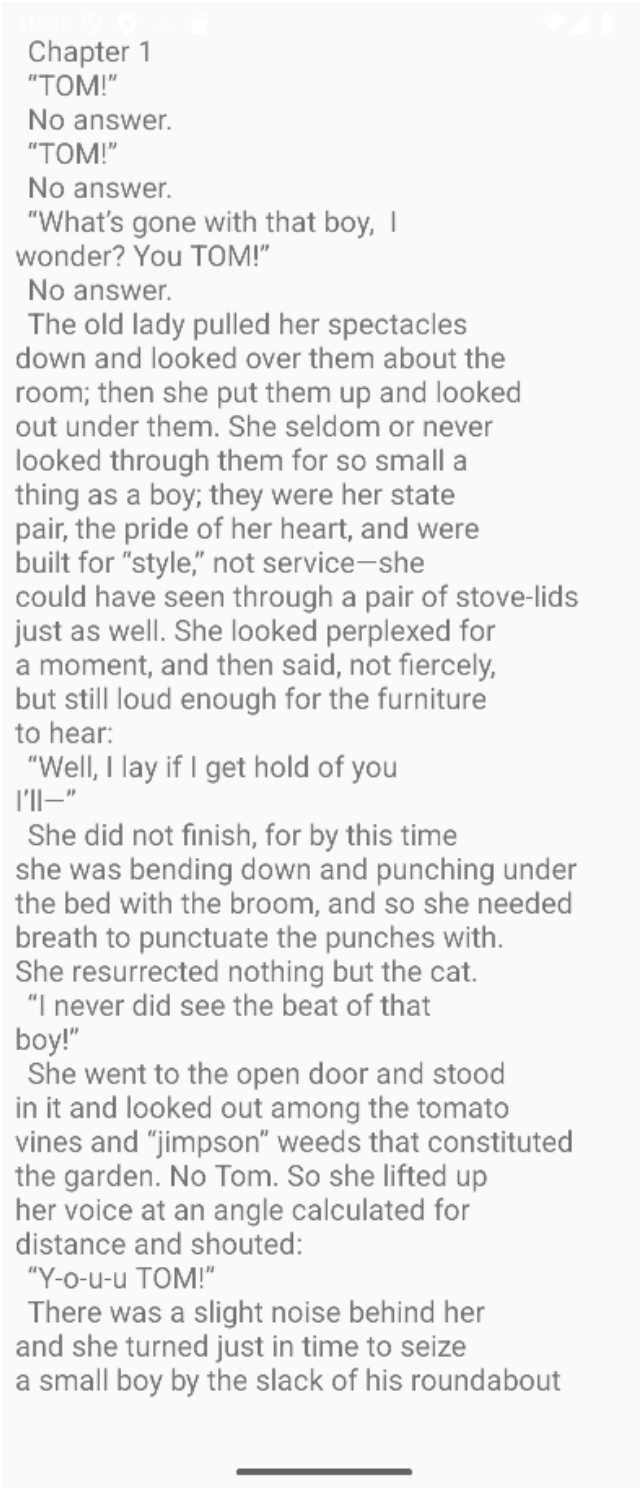

Рисунок 3.30 – Текст в книгах EPUB

На рисунку 3.30 можна побачити відображення текстових сторінок електронних книг формату EPUB. Обмеження довжини строки виконується за допомогою функції addNewlines, що додає перенос якщо довжина строки перебільшує максисальну кількість символів. Схожим чином виконується робота з кількістю строк у сторінці у функції returnPages.

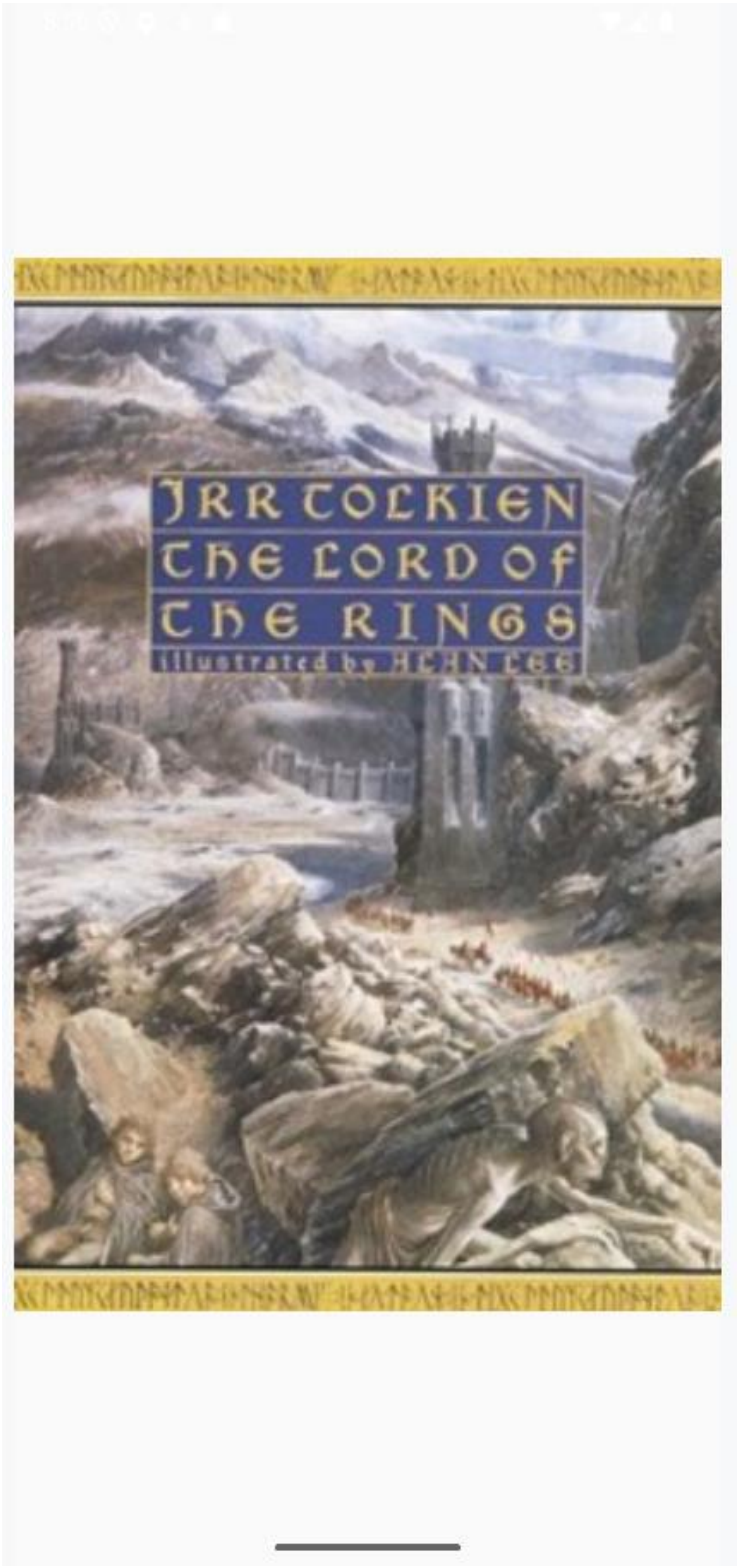

Рисунок 3.31 – Відображення зображення в книгах FB2

На рисунку 3.31 можна побачити відображення обкладинок електронних книг формату FB2, хоча передача посилання на зображення для FB2 відбувається відмінно від процедури для EPUB, відображення зображення відбувається таким самим чином як і з форматом EPUB.

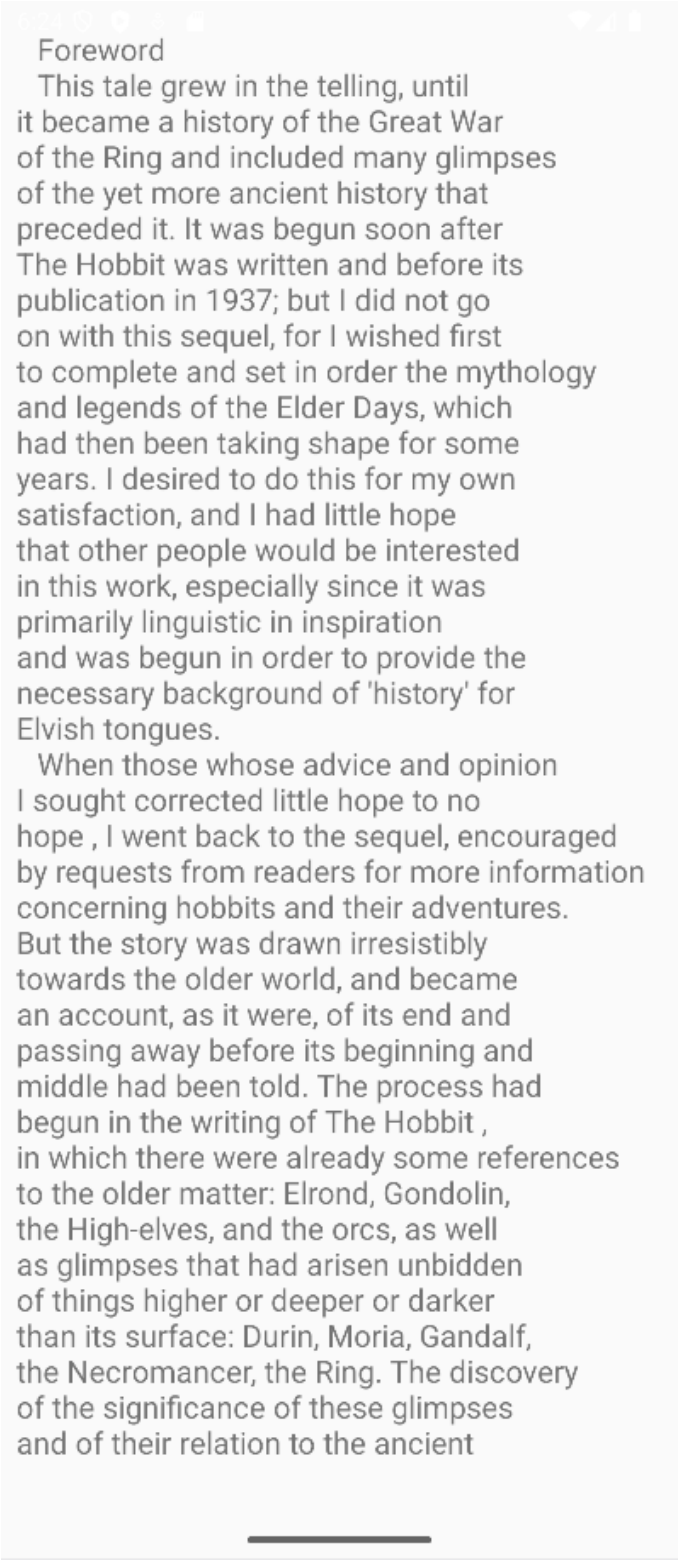

Рисунок 3.32 – Відображення тексту в книгах FB2

На рисунку 3.32 можна побачити відображення текстових сторінок електронних книг формату FB2. Обмеження довжини строки, та кількості строк у сторінці у функції виконується так само як і в форматі EPUB, за допомогою функція addNewlines та returnPages.

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено мобільний застосунок для платформи Android, з метою забезпечення зручного перегляду електронних книг у форматах EPUB та FB2. Оскільки, згідно до IDPF, формат EPUB є найкращим форматом електронних книг, а формат FB2 користується великою популярністю в Україні. Розроблений мобільний застосунок для платформи Android забезпечує можливість перегортання сторінок електронної книги, відображення як текстового вмісту електронних книг, так і вбудованих графічних елементів, з дотриманням адаптивного дизайну розробленого під екрани мобільних телефонів.

Архітектура застосунку була реалізована з використанням мови програмування Java для Android інтерфейсу та мови програмування C++ для обробки й розбору структур файлів у форматах EPUB та FB2. Завдяки використанню технології JNI (Java Native Interface), у застосунку було досягнуто ефективної взаємодії між Java та C++ частиною. Саме за допомогою використання такого підходу було забезпечено можливість досягти високого рівня масштабованості застосунку, дозволяючи перенести проєкт на інші платформи з мінімальними змінами.

Результати роботи апробовано у вигляді тези доповіді під час 29-го Міжнародного молодіжного форуму «радіоелектроніка і молодь у xxi століттІ» [42].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Mashtalir, S.V., Nikolenko, O.V. (2025). Tree-Based Classification of the Technical Ukrainian Texts. In: Hu, Z., Yanovsky, F., Dychka, I., He, M. (eds) Advances in Computer Science for Engineering and Education VII. ICCSEEA 2024 2024. Lecture Notes on Data Engineering and Communications Technologies, vol 242. Springer, Cham

2. Mashtalir S., Nikolenko O. (2024) Hierarchical classification of ukrainian technical texts using tree-based models: applications in the automotive industry, *Advances in information control systems and technologies*, pp. 288-314.

3. Mashtalir S., Nikolenko O. (2023) Data preprocessing and tokenization techniques for technical Ukrainian texts, *Прикладні аспекти інформаційних технологій, 6(3)*, pp. 318–326.

4. Соколова Л.В., Путятін Є.П., Кобилін А.М., Степанов В.П., Сердюченко В.Я., Кузьомін О.Я., Васильєв С.В. (1999) Збірник задач зі спеціальності “Інформатика”: навч. посібник. *Харків: ХНУРЕ*.

5. Kuzminska O., Mazorchuk M., Morze N., Kobylin O. (2020). Digital learning environment of ukrainian universities: The main components to influence the competence of students and teachers, *Information and Communication Technologies in Education, Research, and Industrial Applications: 15th International Conference, ICTERI 2019, Kherson, Ukraine, June 12–15, 2019, Revised Selected Papers*, 1175, pp. 210-230

6. Kobylin O. (2020). Digital Learning Environment of Ukrainian Universities: The Main Components, *Information and Communication Technologies in Education, Research, and Industrial Applications: 15th International Conference, ICTERI 2019, Kherson, Ukraine, June 12–15, 2019, Revised Selected Papers*, 1175, pp. 210-230

7. Tvoroshenko I., Andrieieva A. (2021). Development of web applications for remote learning of English: навч. посібник.

8. Tvoroshenko I., Kuznetsov M. (2021). About the role of testing in process of mobile application development, *Problems of modern science and practice*, 1, pp. 416

9. Путятін Є.П., Гороховатський В.О., Матат О.О. (2006) Методи та алгоритми комп'ютерного зору: навч. посіб.

10. Гороховатський В.О., Творошенко І.С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник. –Харків: ХНУРЕ, с. 7-15.

11. Захаров В.В. (2025) Порівняння ефективності стиснення електронних книг у різних форматах. *Радіоелектроніка і молодь у XXI столітті: тези доповідей 29-го Міжнародного молодіжного форуму (Харків, 16–19 квітня 2025 р.)*. Харків: ХНУРЕ, 2025. Т. 7. С. 49-51.

12. A C Library for Reading, Creating, and Modifying Zip Archives. URL: https://github.com/nih-at/libzip (дата звернення 25.04.2025).

13. A simple, small, efficient, C++ XML parser that can be easily integrated into other programs. URL: https://github.com/leethomason/tinyxml2 (дата звернення 25.04.2025).

14. Object-oriented programming. URL: https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP (дата звернення 27.04.2025).

15. Building with Gradle. URL: https://www.baeldung.com/gradle-fat-jar (дата звернення 01.05.2025).

16. Configure your build. URL: https://developer.android.com/build (дата звернення 01.05.2025).

17. Android Custom intents. URL: https://developer.android.com/guide/app/actions/custom-intents (дата звернення 01.05.2025).

18. Boosting User Engagement. URL: https://medium.com/mset/boosting-user-engagement-improved-ranking-with-sharesheet-custom-actions-android-14-e97f79407f5b (дата звернення 01.05.2025).

19. Особливості «permission handling» під конкретні версії Android. URL: https://developer.android.com/about/versions/13/behavior-changes-13 (дата звернення 01.05.2025).

20. Патерни проєктування. URL: https://refactoring.guru/design-patterns (дата звернення 01.05.2025).

21. Chupikov A., Kinoshenko D., Mashtalir V., Konstantin Shcherbinin (2006). Image retrieval with segmentation-based query, pp. 1-2.

22. Kinoshenko D., Mashtalir V., Orlov A., Yegorova E. (2003). Method of creating of functional invariants under one-parameter geometric image transformations, pp.33-42.

23. Anatomy of an EPUB 3 file https://www.edrlab.org/open-standards/anatomy-of-an-epub-3-file/ (дата звернення 10.04.2025).

24. FB2 description https://wiki.mobileread.com/wiki/FB2 (дата звернення 10.04.2025).

25. MOBI description https://wiki.mobileread.com/wiki/Mobi (дата звернення 10.04.2025).

26. PDF description https://en.wikipedia.org/wiki/PDF (дата звернення 10.04.2025).

27. Chupikov A., Mashtalir S., Yegorova E. (2006) Morphological normalization of image binary cuts, *Computer Vision and Graphics: International Conference, ICCVG 2004, Warsaw, Poland, September 2004, Proceedings*, pp. 558-564

28. Mashtalir S., Putyatin E. (2009) Image normalization under projective transforms, GraVisMa 2009, p. 27

29. Guide to JNI (Java Native Interface) https://www.baeldung.com/jni (дата звернення 10.05.2025).

30. CMake Tutorial https://cmake.org/cmake/help/latest/guide/tutorial/index.html (дата звернення 10.05.2025).