МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «**РОЗРОБКА ЗАСТОСУНКУ ЧИТАННЯ ЕЛЕКТРОННИХ КНИГ ФОРМАТУ FB2**»

Виконав: студент 4 курсу, групи 6.1222
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
М. В. Мірошин
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н., Борю С.Ю.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри програмної інженерії, доцент, к.ф.-м.н., Лісняк А.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри комп'ютерних наук, к.пед.н., доцент

_____ Пшенична О.С.
(підпис)

" 08 " грудня 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Мірошину Микиті Віталійовичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка застосунку читання електронних книг формату FB2

керівник роботи Борю Сергій Юрійович, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 02 » грудня 2025 року № 1741-с

2. Строк подання студентом роботи 04.06.2026

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка застосунку читання електронних книг формату FB2.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 05.12.2025

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	15.12.2025	
2.	Збір вихідних даних.	05.01.2026	
3.	Обробка методичних та теоретичних	07.02.2026	
	джерел.		
4.	Розробка першого та другого розділу.	18.03.2026	
5.	Розробка третього розділу.	28.04.2026	
6.	Оформлення та нормоконтроль	16.05.2026	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	19.06.2026	

Студент _____ М.В. Мірошин
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.Ю. Борю
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка застосунку читання електронних книг формату FB2»: 60 с., 15 рис., 2 табл., 20 джерел, 1 додаток.

АРХІТЕКТУРА ЗАСТОСУНКУ, ЕЛЕКТРОННІ КНИГИ, ФОРМАТ FB2, JAVA, JAVAFX, МЕТОДИ ЧИТАННЯ, РОЗРОБКА ЗАСТОСУНКУ.

Об’єкт дослідження – процес обробки, структурування та візуалізації цифрового текстового контенту в застосунках для читання електронних книг.

Предмет дослідження – методи парсингу XML-структури файлів, архітектурні патерни та інструменти розробки кросплатформеного інтерфейсу користувача для читання книг формату FB2 мовою Java.

Мета роботи: розробити прототип застосунку для зчитування, парсингу та комфортного читання електронних книг у форматі FB2 з можливістю персоналізації інтерфейсу.

Методи дослідження – аналіз, порівняння, моделювання, проєктування, тестування.

У представленій кваліфікаційній роботі було розглянуто особливості роботи застосунків для читання електронних книг у форматі FB2.

У практичній частині розроблено прототип застосунку мовою програмування Java із використанням бібліотеки JavaFX та рушія WebEngine. Реалізовано рекурсивний алгоритм парсингу XML-структури, динамічну навігацію та механізм персоналізації відображення тексту. Застосунок забезпечує відкриття, перегляд та навігацію електронними книгами формату FB2, а також налаштування параметрів відображення тексту.

Результати роботи демонструють архітектурну спроможність обраного підходу до організації ПЗ, зокрема ефективність розділення рівнів бізнес-логіки та представлення даних. Це забезпечує стабільну роботу застосунку та створює надійну базу для його подальшої експлуатації як персонального засобу читання електронних книг.

SUMMARY

Bachelor's Qualifying Thesis «Development of an application for reading FB2 e-books»: 60 pages, 15 figures, 2 tables, 20 references, 1 supplement.

APPLICATION DEVELOPMENT, E-BOOKS, FB2 FORMAT, JAVA, JAVAFX, READING METHODS, SOFTWARE ARCHITECTURE.

Object of the study – the process of processing, structuring, and visualizing digital text content in e-book reading applications.

Subject of the study – methods of XML structure parsing, architectural patterns, and tools for developing a cross-platform user interface for reading FB2 books in Java.

Aim of the study: to develop an application prototype for loading, parsing, and comfortable reading of FB2 electronic books with user interface personalization capabilities.

Methods of research – analysis, comparison, modeling, design, testing.

The presented qualification work examines the features of applications for reading electronic books in the FB2 format.

In the practical part, an application was developed in Java using the JavaFX library and the WebEngine engine. A recursive algorithm for XML structure parsing, dynamic navigation, and a mechanism for personalizing text display have been implemented.

The results demonstrate the architectural viability of the chosen approach, in particular, the effectiveness of separating the business logic and data representation levels. This ensures stable operation of the application and creates a reliable basis for its further use as a personal e-book reader.

ЗМІСТ

Реферат 4
Summary 5
Вступ 7
1 Особливості використання електронних книг 9
1.1 Існуючі формати електронних книг 9
1.2 Формат FB2 та його особливості 12
1.3 Існуючі рішення для читання формату FB2 13
1.4 Постановка задачі 16
2 Проєктування застосунку читання електронних книг формату FB2 18
2.1 Формування вимог до програмного продукту 18
2.2 Обґрунтування вибору технологій розробки 20
2.3 Проєктування архітектури застосунку 22
2.4 Проєктування інтерфейсу взаємодії користувача 25
3 Опис процесу реалізації програмного застосунку 27
3.1 Структура проєкту та архітектурні рішення застосунку 27
3.2 Алгоритми парсингу та перетворення даних FB2 у формат HTML 29
3.3 Демонстрація результатів роботи та взаємодія з інтерфейсом користувача 35
3.4 Шляхи модернізації та перспективи розвитку системи 40
Висновки 42
Перелік посилань 44
Додаток А Програмний код 46

ВСТУП

В останні роки електронні книги стають все більш популярними, оскільки вони забезпечують зручність зберігання та доступу до великої кількості літератури на різних пристроях. Зростання попиту на електронні книги спричинило розвиток численних програмних рішень для їх читання, проте не всі існуючі застосунки забезпечують зручний інтерфейс, гнучку навігацію та підтримку популярних форматів, таких як FB2.

Формат FB2 (FictionBook 2.0), розроблений на основі мови розмітки XML, є одним із найбільш затребуваних у сегменті художньої літератури. Його семантична структура дозволяє зберігати не лише текст, а й детальні метадані, ілюстрації та чітку ієрархію розділів. Розробка застосунку для читання FB2 вимагає не просто відображення символів, а реалізації складних алгоритмів парсингу, трансформації даних та ефективного керування об’єктною моделлю документа (DOM).

Метою кваліфікаційної роботи є розробка прототипу застосунку читання електронних книг у форматі FB2 мовою Java, що забезпечує автоматизоване вилучення метаданих, інтерактивну навігацію по розділах та персоналізацію відображення тексту.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- проаналізувати існуючі формати електронних книг та програмні рішення для їх читання;
- визначити функціональні та технічні вимоги до застосунку;
- спроєктувати архітектуру системи на основі об’єктно-орієнтованого підходу;
- розробити алгоритми безпечного парсингу XML-структури та конвертації семантичних тегів у формат HTML;
- реалізувати графічний інтерфейс користувача з підтримкою динамічного керування стилями;

– провести тестування прототипу на книгах різного обсягу та оцінити його продуктивність.

Об’єктом кваліфікаційної роботи є процес програмної обробки та візуалізації структурованих електронних документів.

Предметом кваліфікаційної роботи є методи парсингу XML-даних, алгоритми трансформації формату FB2 та засоби побудови графічних інтерфейсів на платформі Java.

Методами дослідження в роботі слугують аналіз технічної документації, порівняння програмних бібліотек, об’єктно-орієнтоване моделювання, проектування програмних інтерфейсів (API) та експериментальне тестування.

1 ОСОБЛИВОСТІ ВИКОРИСТАННЯ ЕЛЕКТРОННИХ КНИГ

1.1 Існуючі формати електронних книг

Електронна книга – це цифровий документ, призначений для читання на електронних пристроях: комп’ютерах, планшетах, смартфонах та спеціалізованих рідерах. На сьогодні існує велика кількість форматів електронних книг, які відрізняються за структурою, способом зберігання даних та можливостями відображення [17].

Формат **PDF (Portable Document Format)** є одним із найпоширеніших форматів електронних документів. Він зберігає точне форматування сторінок незалежно від пристрою або операційної системи. Формат підтримує різні методи стиснення та рівні захисту даних (DRM), а також дозволяє вбудовувати шрифти безпосередньо у файл. Це забезпечує автономність документа, але призводить до значного збільшення обсягу файлу порівняно з XML-файлами. Серед його переваг виділяють високу якість відображення, підтримка графіки та шрифтів. Серед його недоліків виділяють незручність читання на малих екранах та відсутність адаптивного форматування тексту [19].

Формат **EPUB (Electronic Publication)** є відкритим стандартом електронних книг, який широко використовується на більшості сучасних пристроїв для читання. Він базується на технологіях HTML та CSS, що забезпечує адаптивне відображення тексту залежно від розміру екрана [12]. Формат підтримує два типи макетів: адаптивний (Reflowable Layout) та фіксований (Fixed Layout). Фіксована верстка дозволяє використовувати EPUB для дитячих книг з ілюстраціями та фотоальбомів, зберігаючи точне розташування тексту поверх зображень, подібно до формату PDF [13]. До переваг формату EPUB належать можливість зміни шрифту, розміру тексту, міжрядкового інтервалу, а також підтримка мультимедійних елементів.

Недоліком є складніша внутрішня структура файлу, що ускладнює його обробку без спеціалізованих бібліотек.

Формат **FB2 (FictionBook)** є відкритим XML-орієнтованим форматом, який призначений переважно для художньої літератури [19]. Основною особливістю формату є збереження логічної та семантичної структури тексту, а не його візуального оформлення, що дозволяє відокремити зміст від механізмів його відображення [3]. На відміну від контейнерних форматів, формат FB2 є монолітним документом, де весь контент, включаючи метадані та ілюстрації, зберігається в одному файлі. Зображення інтегруються безпосередньо в структуру XML у вигляді бінарних даних, закодованих за допомогою Base64 [2].

Такий підхід забезпечує високу адаптивність контенту до різних пристроїв та гарантує цілісність документа при копіюванні. До переваг формату належать компактність, простота парсингу завдяки чіткій ієрархії тегів та зручність обробки стандартними програмними засобами обробки XML [3]. Однак семантичний підхід накладає певні обмеження: можливості складного візуального форматування є значно вужчими порівняно з HTML-орієнтованими форматами, а також відсутня вбудована підтримка захисту авторських прав (DRM).

Формат **MOBI** використовувався переважно в електронних рідерах компанії Amazon. Він підтримує навігацію, закладки та коментарі, що робить його зручним для читання [9]. Водночас формат є частково закритим і має обмежену підтримку на сторонніх платформах, що зменшує його універсальність.

Формати **AZW** та **AZW3** є розвитком формату MOBI та застосовуються в екосистемі Amazon Kindle [9]. Вони підтримують розширені можливості форматування та захисту контенту. Основним недоліком є жорстка прив’язаність до пристроїв та програмного забезпечення Amazon.

Формат **DJVU** призначений для зберігання відсканованих документів і книг. Він забезпечує високий рівень стиснення зображень при збереженні прийнятної якості [4]. Недоліком цього формату є відсутність адаптивного тексту, що ускладнює читання на мобільних пристроях.

Формат **TXT** є найпростішим текстовим форматом без підтримки форматування. Його перевагами є мінімальний розмір файлу та сумісність з будь-якими пристроями. Проте відсутність структури, стилів і метаданих значно обмежує можливості використання цього формату для повноцінних електронних книг [10].

Далі наведено порівняльну таблицю основних форматів електронних книг, у якій подано їхні ключові характеристики та особливості (табл. 1.1).

Таблиця 1.1 – Порівняльна таблиця існуючих форматів електронних книг

Характеристика	**PDF, DJVU**	**EPUB, FB2, MOBI, TXT**
Принцип побудови контенту	Базуються на фіксованій сторінці або растрових зображеннях. Орієнтовані на точне збереження графічного та візуального вигляду друкованого оригіналу	Використовують текстову семантичну розмітку (XML, HTML) або спеціалізовані структури даних. Орієнтовані на логічну будову тексту
Адаптивність	Текст не підлаштовується під геометричні розміри екрана, що обмежує рівень ергономіки при читанні на мобільних пристроях	Текст динамічно змінює своє розташування на дисплеї залежно від його роздільної здатності, обраного шрифту та інтервалу
Можливість роботи з метаданими	Обмежена технічними специфікаціями файлового контейнера. Пошук, індексація та вилучення метаданих ускладнені у растрових копіях документів	Завдяки структурованій розмітці (крім формату TXT) дозволяють зберігати детальну ієрархію розділів, анотацій та авторів
Тип поширення та ліцензія	Переважно відкриті стандарти, проте формат PDF може містити складні вбудовані DRM-системи захисту авторських прав	Поєднують відкриті кросплатформові стандарти (FB2, EPUB) та закриті пропрієтарні формати (MOBI), які обмежені екосистемами виробників
Ефективність зберігання даних	Фіксується значний обсяг файлів через необхідність інтеграції шрифтових гарнітур або збереження графічних шарів високої якості	Текстові дані займають мінімальний обсяг дискового простору, що дозволяє оптимізувати збереження великих масивів літератури

1.2 Формат FB2 та його особливості

Формат FB2 (FictionBook) є відкритим форматом електронних книг, який був розроблений для зберігання та поширення художньої літератури в електронному вигляді. Основна ідея формату полягає в збереженні логічної та семантичної структури тексту, а не його зовнішнього візуального оформлення. Формат базується на мові розмітки XML, що забезпечує зручність обробки файлів програмними засобами [2].

Файл формату FB2 містить текст книги, структурований у вигляді вкладених елементів, таких як розділи, абзаци, заголовки та примітки. Основна текстова частина книги розміщується в тегу <body>, який може містити декілька секцій <section>. Кожна секція, у свою чергу, складається з абзаців <p> та заголовків <title>. Такий підхід дозволяє чітко відокремлювати логічні частини книги [3].

Важливою складовою формату FB2 є блок метаданих <description>, у якому зберігається інформація про автора, назву книги, жанр, мову, дату публікації та інші відомості. Наявність валідованих метаданих значно полегшує автоматизовану організацію електронної бібліотеки, сортування та швидкий пошук книг у розроблюваному програмному забезпеченні [19].

Формат FB2 підтримує інтеграцію графічних зображень, які зберігаються безпосередньо всередині файлу у вигляді двійкових даних, закодованих за допомогою Base64. Це дозволяє використовувати ілюстрації без необхідності підключення зовнішніх ресурсів. Однак така особливість може збільшувати розмір файлу [4].

До основних переваг формату FB2 належать:

- відкритість і відсутність ліцензійних обмежень;
- компактність та ефективне зберігання текстових даних [4];
- зручність парсингу завдяки XML-структурі [3];
- адаптивність відображення на різних пристроях;
- наявність чітко визначеної логічної структури тексту.

Серед недоліків формату FB2 можна виокремити:

– обмежені можливості візуального форматування;
– відсутність вбудованих механізмів захисту авторських прав (DRM);
– нижчу популярність у порівнянні з форматом EPUB у західних країнах.

Завдяки своїй простоті та орієнтації на зміст, формат FB2 широко використовується в програмних засобах для читання електронних книг і є зручним об’єктом для розробки власних застосунків-рідерів.

1.3 Існуючі рішення для читання формату FB2

На сьогодні існує значна кількість програмних рішень для читання електронних книг формату FB2, які реалізовані для різних операційних систем. Ці застосунки забезпечують базовий функціонал перегляду тексту, навігації між розділами, а також додаткові можливості персоналізації процесу читання.

Одним із найпоширеніших застосунків є **FBReader**, який підтримує формат FB2 та працює на платформах Windows, Linux, Android і iOS. Застосунок надає можливість налаштування шрифтів, кольорових тем, міжрядкового інтервалу, а також підтримує закладки та збереження прогресу читання. Водночас інтерфейс застосунку є досить мінімалістичним і не завжди зручним для початкових користувачів.

Застосунок **Calibre** є багатофункціональним інструментом для керування електронною бібліотекою. Окрім читання файлів формату FB2, застосунок надає можливість конвертації книг у різні формати та редагування метаданих. Головним недоліком застосунку є значна кількість функцій, які перевантажують інтерфейс, що не завжди доцільно для простого процесу читання.

Застосунок **Cool Reader** також широко використовується для читання файлів FB2, особливо на мобільних пристроях та електронних рідерах. Серед його головних переваг можна виділити високу швидкодію, підтримку різних форматів електронних книг і розширені можливості налаштування зовнішнього

вигляду тексту. Головним недоліком застосунку є застарілий дизайн інтерфейсу та обмежена підтримка сучасних операційних систем [18].

Застосунок **AlReader** орієнтований переважно на мобільні пристрої та підтримує широкий набір функцій, серед яких автопрокрутка тексту, робота з закладками та нічний режим. Проте складність інтерфейсу може ускладнювати користування для нових користувачів.

Далі наведено порівняльну таблицю існуючих програмних рішень для читання електронних книг формату FB2 (табл. 1.2).

Таблиця 1.2 – Порівняльна таблиця існуючих програмних рішень для читання електронних книг формату FB2

Характеристика	**FBReader, Cool Reader**	**Calibre**	**AlReader**
Візуалізація та рендеринг	Використовують власні легковагові процедурні рушії відображення тексту	Базується на потужних вебрушіях (зокрема Qt WebEngine) з повною підтримкою стандартів	Орієнтований на максимально швидкий рендеринг текстових рядків на екрані
Інтерфейс користувача та зручність (UX)	Мінімалістичний, іноді застарілий дизайн вікон керування	Має суттєво ускладнений інтерфейс користувача, зумовлений наявністю великої кількості допоміжних підсистем для перезапису та конвертування файлових структур	Має сучасний вигляд, але часто складний через велику кількість жестів
Робота з метаданими та змістом	Базова обробка основних тегів. Зміст відображається у вигляді простого текстового списку	Розширена робота з реляційною базою даних та повне редагування метаданих	Автоматична індексація та сортування за метаданими з усієї пам’яті пристрою
Навігація по тексту	Лінійна прокрутка або гортання сторінок. Базова підтримка гіперпосилань у тексті	Складна система пошуку, закладок та перехресних посилань між різними книгами бібліотеки	Інтерактивна навігація з використанням автопрокрутки та швидкого переходу за прогрес-баром
Системні вимоги	Працюють стабільно на застарілому обладнанні та слабких операційних системах	Потребує значного обсягу оперативної пам’яті для обробки бази даних та рендерингу	Оптимізована під енергоефективність мобільних процесорів та обмежену пам’ять

Аналіз існуючих програмних рішень для читання електронних книг формату FB2 дозволяє зробити наступні висновки:

Більшість розглянутих аналогів (зокрема Calibre та AlReader) мають широкий набір додаткових функцій, таких як редагування метаданих, конвертація форматів або складне керування бібліотеками. Це призводить до перевантаження інтерфейсу елементами керування, що ускладнює взаємодію з програмою для користувачів, чиєю основною метою є виключно читання тексту [16].

Значна частина застосунків (наприклад, Cool Reader) використовує застарілі графічні бібліотеки, що негативно позначається на якості рендерингу шрифтів та загальній естетиці інтерфейсу. Обмежені можливості візуалізації, відсутність оптимізації під графічні процесори та ігнорування сучасних стандартів CSS-стилізації у базових рушіях конкурентів суттєво знижують якість типографіки та загальну ергономіку читання.

Хоча існують універсальні рішення (FBReader), вони часто жертвують нативним виглядом та зручністю керування на конкретній операційній системі заради підтримки мобільних платформ.

Отже, проведений огляд підтверджує доцільність розробки спеціалізованого застосунку мовою Java з використанням бібліотеки JavaFX. Такий підхід дозволить:

- спростити інтерфейс взаємодії, залишивши лише необхідний функціонал для читання електронних книг [16];
- забезпечити високу якість відображення тексту завдяки використанню рушія WebEngine;
- створити легкий та швидкий продукт, оптимізований для десктопних операційних систем.

Таким чином, наведені аргументи підтверджують інженерну необхідність розробки власного програмного забезпечення, головною метою якого є реалізація ефективного балансу між технічним потенціалом програми та зручністю її повсякденного використання.

1.4 Постановка задачі

На підставі аналізу існуючих форматів електронних книг та доступних рішень для їх читання визначено, що на ринку програмного забезпечення існує потреба в розробці спеціалізованого десктопного застосунку. Такий продукт має поєднувати простий, інтуїтивно зрозумілий інтерфейс користувача з оптимізованим набором функцій для комфортного читання електронних книг формату FB2, без перевантаження системи зайвими модулями адміністрування баз даних чи конвертації.

Мета роботи полягає в розробці прототипу застосунку для читання електронних книг у форматі FB2 мовою Java, що забезпечує автоматизоване вилучення метаданих, інтерактивну навігацію розділами та персоналізацію відображення тексту.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- проаналізувати існуючі формати електронних книг та визначити технічні особливості формату FB2;
- дослідити наявні програмні рішення для читання книг у форматі FB2, проаналізувавши їхні функціональні можливості, переваги та недоліки;
- визначити функціональні та технічні (нефункціональні) вимоги до розроблюваного застосунку;
- спроєктувати архітектуру системи та графічний інтерфейс користувача на основі об’єктно-орієнтованого підходу;
- реалізувати програмний продукт із підтримкою рекурсивного алгоритму парсингу XML-структури, конвертації семантичних тегів у формат HTML та динамічного керування стилями відображення тексту;
- провести експериментальне тестування прототипу застосунку на електронних книгах різного обсягу та оцінити його загальну продуктивність.

Очікуваним результатом роботи є функціональний програмний застосунок для читання електронних книг формату FB2, що демонструє архітектурну спроможність обраного підходу (розділення бізнес-логіки та представлення даних) і створює надійну базу для його подальшої експлуатації як персонального засобу читання.

2 ПРОЄКТУВАННЯ ЗАСТОСУНКУ ЧИТАННЯ ЕЛЕКТРОННИХ КНИГ ФОРМАТУ FB2

2.1 Формування вимог до програмного продукту

Основою для успішної розробки будь-якого програмного забезпечення є чітке визначення функціональних та нефункціональних вимог до нього. На базі проведеного у першому розділі аналізу аналогів та сформульованої постановки задачі було визначено ключові вимоги до застосунку для читання електронних книг формату FB2. Стандартно такі вимоги поділяються на функціональні, що описують безпосередню поведінку та можливості системи, та нефункціональні, які визначають атрибути якості, критерії надійності та технічні обмеження середовища виконання [7].

До основних функціональних вимог (Functional Requirements) застосунку належать:

- завантаження та відкриття файлів із розширенням .fb2 з локальної файлової системи пристрою користувача;
- лексичний та синтаксичний аналіз (парсинг) монолітної XML-структури документа для коректного вилучення метаданих, основного тексту, заголовків, епіграфів та інтегрованих бінарних зображень;
- забезпечення динамічної навігації електронною книгою за допомогою автоматично згенерованого інтерактивного змісту (Table of Contents), який формується на основі парсингу тегів <title>;
- функціонування механізму персоналізації відображення тексту (можливість динамічної зміни розміру шрифту користувачем через елементи керування графічного інтерфейсу).

До основних нефункціональних вимог (Non-functional Requirements) застосунку належать:

- продуктивність: Застосунок повинен забезпечувати стабільну обробку та завантажувати файли великого розміру (понад 5-10 МБ) без значних затримок у роботі інтерфейсу;
- ергономічність: Інтерфейс користувача (UI) має бути інтуїтивно зрозумілим, мінімалістичним, не перевантаженим зайвими функціями та відповідати сучасним стандартам проєктування (UX-дизайну) [1];
- надійність: Програма повинна коректно обробляти виняткові ситуації (наприклад, спробу відкриття пошкодженого XML-файлу або файлу некоректного формату) та виводити інформаційні повідомлення користувачу без аварійного завершення процесу;
- масштабованість: Архітектура застосунку має базуватися на принципах розділення рівнів бізнес-логіки та представлення даних, що забезпечить у майбутньому можливість легкого розширення функціоналу.

Для наочного представлення взаємодії користувача з функціональними модулями системи було розроблено діаграму прецедентів (рис. 2.1).

Як зображено на діаграмі, головним актором є користувач, який взаємодіє з набором прецедентів. Основний сценарій починається з вибору файлу. Цей прецедент пов’язаний із виведенням повідомлення про помилку, що забезпечує надійність системи у разі спроби відкриття некоректного файлу. Інші прецеденти, такі як перегляд змісту, читання тексту та налаштування шрифту повністю охоплюють задекларовані функціональні вимоги до програмного забезпечення.

Таким чином, сформований перелік вимог до програмного продукту повністю відповідає потребам цільової аудиторії та є достатнім для переходу до етапу вибору технологічного стека та проєктування архітектури програмного продукту [11].

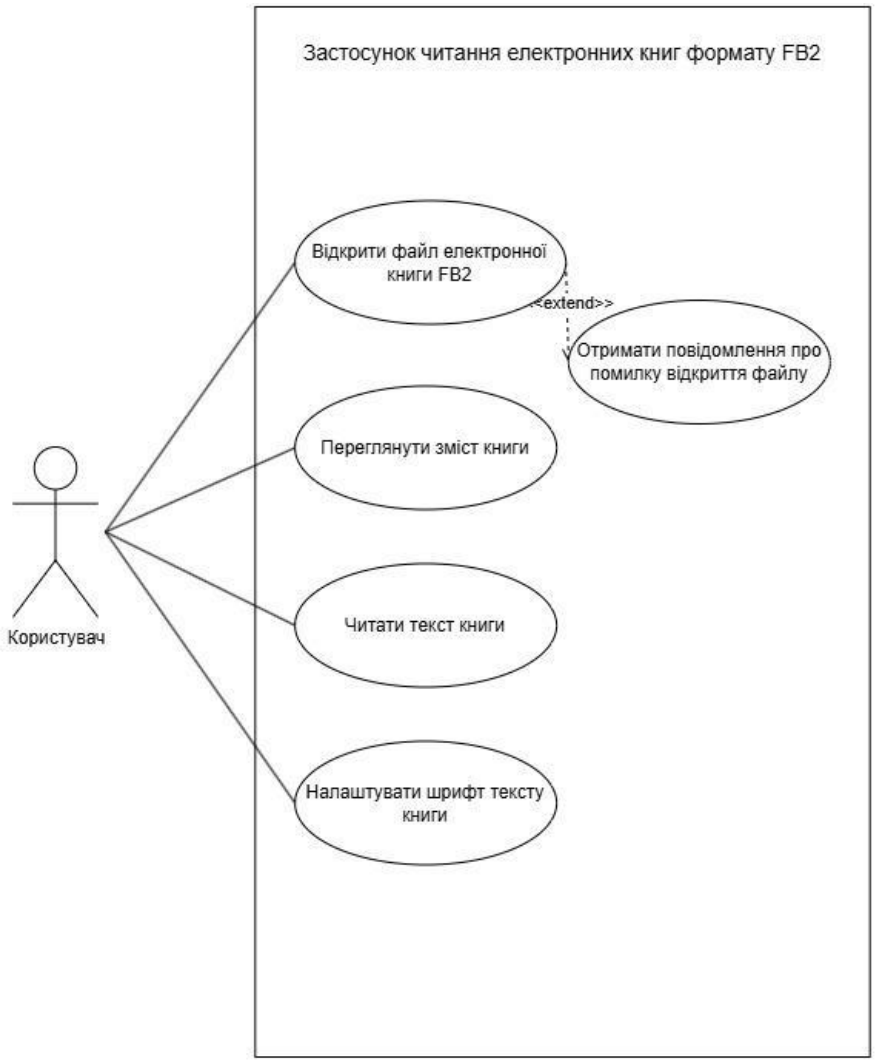


Рисунок 2.1 – Діаграма прецедентів

2.2 Обґрунтування вибору технологій розробки

Вибір технологічного стека є фундаментальним етапом проєктування, оскільки він визначає продуктивність, масштабованість та кросплатформність майбутнього застосунку [7].

Для реалізації логіки застосунку було обрано об’єктно-орієнтовану мову Java. Вибір обґрунтовується її надійністю та архітектурною відповідністю завданням проєкту:

– кросплатформність: Завдяки віртуальній машині JVM, застосунок може функціонувати на Windows, macOS та Linux без модифікації вихідного коду [6];

- ефективна обробка XML: Оскільки формат FB2 базується на мові розмітки XML, використання вбудованого пакету javax.xml.parsers дозволяє обробляти дані без сторонніх бібліотек. Обрана модель DOM (Document Object Model) забезпечує побудову повного ієрархічного дерева документа в пам’яті, що необхідно для швидкої навігації та формування змісту книги [2];
- управління ресурсами: автоматичне керування пам’яттю (Garbage Collection) мінімізує ризики витоків пам’яті при роботі з великими текстовими масивами та ілюстраціями.

Для побудови графічного інтерфейсу (GUI) було обрано JavaFX, яка на відміну від застарілих бібліотек (Swing, AWT), надає інструменти для створення сучасних реактивних інтерфейсів [6]:

- компонент WebView: Інтегрує браузерний рушій WebEngine (на базі WebKit), що дозволяє використовувати HTML5 та CSS3 для рендерингу тексту. Це забезпечує високу якість типографіки та підтримку складного форматування (епіграфи, вірші, виноски);
- міст Java-JavaScript: Технологія дозволяє виконувати скрипти безпосередньо з Java-контролера. Це використано для реалізації динамічної навігації та зміни стилів відображення в реальному часі без перезавантаження документа;
- апаратне прискорення: Використання графічного процесора (GPU) забезпечує плавне прокручування тексту та візуалізацію графічних елементів, що є критичним для комфортного читання.

Для автоматизації життєвого циклу розробки було обрано Maven, що дозволило стандартизувати структуру проєкту та спростити управління зовнішніми залежностями. Вибір Maven обґрунтований наступними факторами:

- декларативне управління залежностями: Автоматичне завантаження та оновлення необхідних бібліотек JavaFX та модулів XML-парсингу через файл конфігурації pom.xml;

- стандартизація збірки: забезпечення ідентичного процесу компіляції та пакування застосунку на різних робочих станціях розробників.

Для основного середовища розробки (IDE) було обрано Visual Studio Code. Ця платформа надає розширені інструменти для написання та рефакторингу коду, зручний вбудований термінал для тестування та інтегровані засоби налагодження. Серед переваг Visual Studio Code можна виділити:

- продуктивність: Низьке споживання системних ресурсів порівняно з важкими IDE дозволяє швидко працювати з великими файлами вихідного коду та XML-документами;
- екосистему розширень: Використання пакета Extension Pack for Java забезпечує повноцінну інтелектуальну підтримку коду (IntelliSense), автоматичне форматування та засоби статичного аналізу;
- інструменти профайлінгу: Наявність інтегрованих засобів для аналізу стеку викликів та споживання оперативної пам'яті дозволила провести оптимізацію рекурсивних алгоритмів парсингу, забезпечивши їх стабільну роботу на файлах великого обсягу.

2.3 Проєктування архітектури застосунку

Архітектура застосунку була розроблена з урахуванням принципу розділення відповідальності (Separation of Concerns). Основна ідея полягає у відокремленні логіки парсингу та обробки даних формату FB2 від логіки графічного інтерфейсу користувача (GUI). Такий підхід забезпечує модульність системи, що спрощує подальше тестування та супровід коду [14].

Для опису фізичної структури системи, визначення зв’язків між незалежними модулями та специфікації інтерфейсів взаємодії було розроблено діаграму компонентів (рис. 2.2).

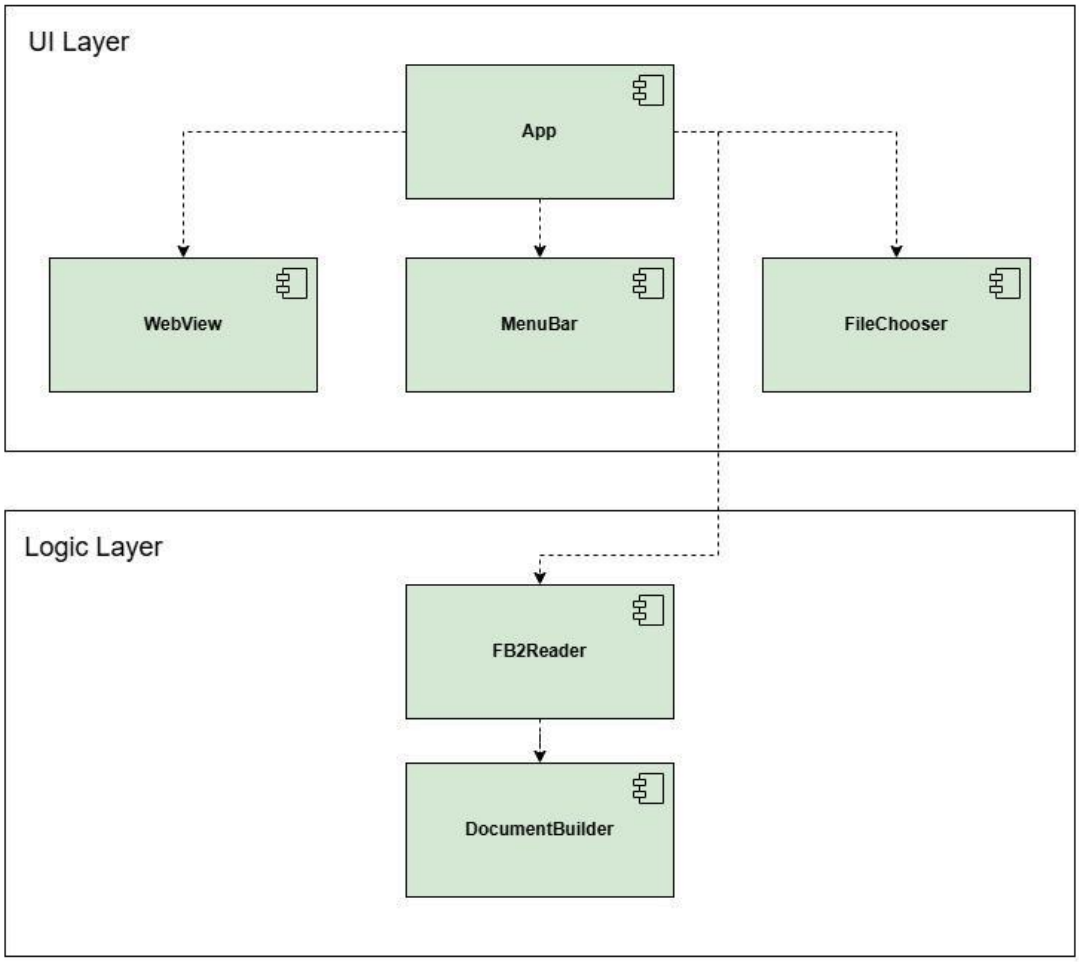


Рисунок 2.2 – Діаграма компонентів

Відповідно до розробленої діаграми компонентів, систему розділено на два ключові рівні:

- рівень представлення (UI Layer): Центральним компонентом є клас App, який координує роботу інтерфейсу та успадковує функціональність Application (JavaFX). Він керує MenuBar для обробки подій, використовує FileChooser для вибору джерела даних та передає сформований HTML-контент у WebView для рендерингу;
- рівень бізнес-логіки (Logic Layer): Містить компонент FB2Reader, який відповідає за інтелектуальну обробку файлу. Компонент ініціює роботу низькорівневого аналізатора DocumentBuilder, що будує DOM-модель документа для подальшого вилучення метаданих (title, author) та основного тексту.

Для деталізації внутрішньої структури програмного продукту та визначення об’єктно-орієнтованої моделі було розроблено діаграму класів (рис. 2.3).

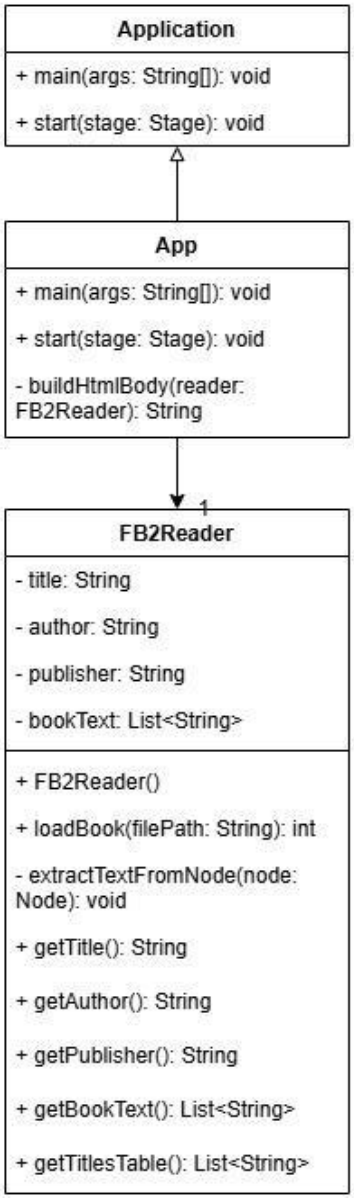


Рисунок 2.3 – Діаграма класів

У даній моделі виділено два основні класи, які забезпечують роботу системи: App та FB2Reader. Клас App відповідає за ініціалізацію графічного інтерфейсу, управління життєвим циклом застосунку та візуалізацію даних у компоненті WebView. Клас FB2Reader містить бізнес-логіку обробки файлів, включаючи зчитування метаданих книги (назва, автор, видавництво) та формування текстового контенту [20].

Зв’язки між класами вказують на ієрархічну взаємодію: клас App наслідує базовий функціонал Application для роботи з віконною системою JavaFX та використовує екземпляр класу FB2Reader для отримання структурованої інформації з документів. Така модель забезпечує чітке розділення відповідальності між візуальним представленням та механізмом обробки XML-даних, що дозволяє ефективно реалізувати функціональність системи відповідно до принципів об’єктно-орієнтованого програмування [5].

В основі архітектури лежить принцип ієрархічної залежності: рівень представлення ініціює запити до логічного рівня, отримуючи структуровані дані, але рівень логіки залишається повністю автономним від візуальної реалізації. Використання вбудованого браузерного рушія всередині WebView дозволяє перекласти складне завдання рендерингу тексту з підтримкою CSS-стилів на готові системні рішення, що робить архітектуру самого застосунку максимально стабільною та продуктивною.

2.4 Проєктування інтерфейсу взаємодії користувача

Проєктування інтерфейсу користувача (UI) для системи читання електронних книг базується на принципах чистоти контенту та логічного розділення функцій керування і перегляду. У розробленому застосунку використано бібліотеку JavaFX, яка забезпечує високу швидкість рендерингу та гнучкість налаштувань за допомогою дерева візуальних об’єктів (Scene Graph).

Для організації візуального простору обрано контейнер верхнього рівня BorderPane, що дозволяє чітко зонувати інтерфейс відповідно до функціонального призначення елементів:

- верхня область (Top): Розміщує компонент MenuBar, який забезпечує доступ до глобальних функцій керування файлами та налаштуваннями;
- центральна область (Center): Займає основну площу вікна та містить компонент SplitPane, що реалізує адаптивний патерн «Master-Detail».

Це дозволяє користувачеві динамічно змінювати ширину навігаційної панелі або приховувати її для максимального фокусування на тексті;

– компонент WebView: Інтегрований у праву частину SplitPane, він використовує рушій WebEngine (на базі WebKit) для прецизійного відображення тексту з підтримкою стандартів HTML5 та CSS3.

Для зручного орієнтування у книгах великого обсягу спроєктовано навігаційну панель на базі ListView, яка розташована у лівій частині вікна. Таке рішення забезпечує швидкий доступ до ієрархічної структури документа, сформованої під час парсингу.

Інтерфейс спроєктовано як подійно-орієнтовану систему, що миттєво відгукується на дії користувача. Для забезпечення взаємодії з файловою системою реалізовано механізм вибору документів на основі класу FileChooser. При ініціалізації діалогового вікна застосовується декларативний фільтр розширень. Після парсингу об’єкт FB2Reader передає структуру даних у WebView, де через ін’єкцію JavaScript реалізовано безшовну навігацію між розділами без перезавантаження всього документа.

Для реалізації параметрів персоналізації (зміна розміру шрифту) обрано модель модальних діалогових вікон (Stage з параметром Modality.WINDOW_MODAL). Це дозволяє:

– зберегти фокус користувача на конкретній операції налаштування;

– запобігти випадковим конфліктам взаємодії з основним вікном під час введення параметрів;

– реалізувати валідацію введених значень у реальному часі перед застосуванням нових стилів [16].

При проєктуванні особливу увагу приділено ергономіці та зоровому навантаженню. Використано палітру з нейтральними кольорами фону та високим контрастом шрифтів для забезпечення максимальної читабельності. Завдяки властивостям прив’язок JavaFX, елементи інтерфейсу автоматично масштабуються відповідно до розмірів вікна, зберігаючи пропорції та доступність елементів керування.

3 ОПИС ПРОЦЕСУ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАСТОСУНКУ

3.1 Структура проєкту та архітектурні рішення застосунку

Для реалізації застосунку для читання електронних книг формату FB2 було розроблено чітко структуровану архітектуру, яка забезпечує логічну організацію всіх компонентів програми та суворе розділення графічного інтерфейсу від бізнес-логіки.

Зокрема, у проєкті містяться два основні файли (класи), які керують роботою застосунку: App.java є центральним файлом, що охоплює ініціалізацію графічного інтерфейсу користувача (GUI), налаштування головного вікна (Stage), обробку подій (натискання кнопок меню, вибір розділів) та логіку переходу між станами програми (від стартового екрана до режиму читання). Опис усіх елементів інтерфейсу, таких як головне меню, панель налаштувань, інтерактивний зміст (ListView) та безпосередньо область читання тексту, також міститься у цьому файлі. Натомість для глибокої обробки даних використовується файл FB2Reader.java, який задає правила парсингу XML-структури книги, вилучає метадані (назва, автор, видавництво) та відповідає за конвертацію сирого тексту формату FB2 у структурований HTML-код (рис 3.1).

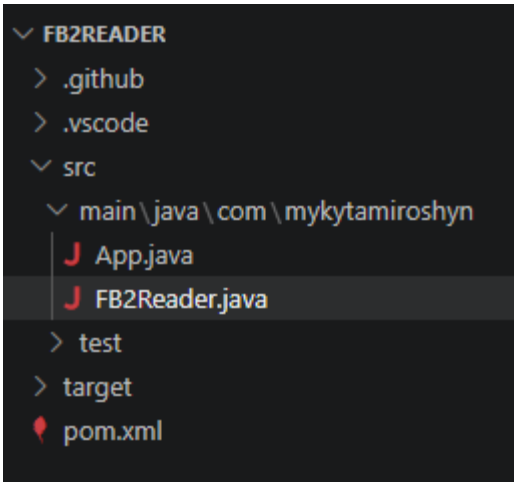


Рисунок 3.1 – Дерево файлів та структура пакетів розробленого застосунку

У процесі розробки було реалізовано багато алгоритмів та елементів, що вирішують специфічні задачі та оптимізують роботу застосунку: це рекурсивний алгоритм трансформації специфічних тегів FB2 (таких як <section>, <emphasis>, <image>) у відповідні HTML-еквіваленти, кастомізована система навігації, що автоматично формує зміст на основі тегів <title>, авторський механізм зміни розміру шрифту тексту «на льоту» за допомогою ін’єкції JavaScript-коду (без необхідності повного перезавантаження документа), а також власна логіка управління станами пам’яті при закритті чи зміні файлу книги.

Для запобігання витокам оперативної пам’яті (Memory Leaks) було імплементовано механізм деструктуризації об’єктів при закритті файлу. Система ініціює обнулення посилань на Document та примусове очищення навігаційних колекцій ArrayList. Такий підхід у поєднанні з особливостями роботи Garbage Collector віртуальної машини Java дозволяє ефективно звільняти ресурси, що раніше займали масиви текстових даних та структури DOM-дерева, забезпечуючи стабільну роботу застосунку на пристроях з обмеженим обсягом оперативної пам’яті.

Водночас, було ефективно використано стандартний функціонал мови Java та платформи JavaFX, що дозволило значно пришвидшити та оптимізувати процес розробки. До цього функціоналу належать базова система обробки XML-документів за допомогою вбудованого DOM-парсера (javax.xml.parsers), стандартні механізми роботи з файловою системою операційної системи (FileChooser), стандартні структури даних (ArrayList для кешування тексту та заголовків), а також потужний компонент WebView із рушієм WebEngine, який виконує апаратний рендеринг тексту, обробку CSS-стилів та забезпечення плавного прокручування контенту.

3.2 Алгоритми парсингу та перетворення даних FB2 у формат HTML

Для підвищення модульності та читабельності коду основний алгоритм обробки книги було розділено на спеціалізовані підпрограми. Це дозволило ізолювати логіку налаштування XML-парсера від бізнес-логіки вилучення тексту та метаданих.

Центральним вузлом цієї структури є метод parseBook, який виконує роль головного керуючого компонента (диспетчера), що координує послідовність виклику допоміжних методів (рис. 3.2).

```
public int parseBook(String filePath)
{
    File file = new File(filePath);
    if (!file.exists() || !file.isFile())
    {
        return 0;
    }

    try
    {
        Document document = buildDocument(file);

        parseTitlesTable(document);

        parseBookBody(document);

        getTagsContent(document, "book-title", "author",
"publisher");

        return 1;
    }
    catch (Exception e)
    {
        e.printStackTrace();
        return 0;
    }
}
```

Рисунок 3.2 – Реалізація головного керуючого методу parseBook

Метод buildDocument виконує низькорівневе конфігурування DocumentBuilderFactory. Важливою особливістю реалізації є активація функції disallow-doctype-decl, що блокує можливість проведення атак типу XML External Entity (XXE). Це забезпечує високий рівень безпеки користувацьких даних, унеможливлюючи доступ шкідливого коду до файлової системи через механізм зовнішніх сутностей XML. Додатково у методі реалізовано етап нормалізації отриманого документа (normalize()), що приводить ієрархію вузлів до єдиного стандарту, видаляючи надлишкову сегментацію тексту. Такий підхід гарантує стабільність подальшої обробки та перетворює сирий файл на структуровану об'єктну модель, готову до вилучення метаданих та контенту (рис. 3.3).

```
private Document buildDocument(File file) throws
Exception
{
    DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();

factory.setFeature("http://apache.org/xml/features/disal
low-doctype-decl", true);

    DocumentBuilder builder =
factory.newDocumentBuilder();
    Document document = builder.parse(file);
    document.getDocumentElement().normalize();

    return document;
}
```

Рисунок 3.3 – Реалізація методу buildDocument для конфігурації парсера

Метод parseTitlesTable відповідає за автоматичне формування змісту книги. Алгоритм ітерує по всіх елементах <section> і вилучає перший знайдений тег <title>, що дозволяє побудувати точну навігаційну карту документа. Вибір рекурсивного перебору нащадків кожної секції замість прямого виклику всіх тегів <title> обумовлений специфікою формату FB2. Тег <title> може використовуватися не лише для назв розділів, а й у блоках анотацій (<annotation>)

або епіграфах. Обмеження пошуку контекстом елемента <section> дозволяє відфільтрувати зайві дані та сформувати чистий навігаційний список, що складається виключно з назв глав книги. Алгоритм включає етап нормалізації текстового контенту за допомогою методу trim(). Це необхідно для видалення зайвих пробілів та символів перенесення рядка, які часто виникають при форматуванні XML-файлів. Такий підхід гарантує естетичний вигляд елементів у графічному списку ListView (рис. 3.4).

```
private void parseTitlesTable(Document document)
{
    titlesTable.clear();
    NodeList sections =
document.getElementsByTagName("section");
    for (int i = 0; i < sections.getLength(); i++)
    {
        Node section = sections.item(i);
        NodeList childNodes = section.getChildNodes();
        for (int j = 0; j < childNodes.getLength(); j++)
        {
            Node child = childNodes.item(j);
            if (child.getNodeName().equals("title"))
            {
                String titleText =
child.getTextContent().trim();
                if (!titleText.isEmpty())
                {
                    titlesTable.add(titleText);
                }
                break;
            }
        }
    }
}
```

Рисунок 3.4 – Реалізація методу parseTitlesTable для формування змісту книги

Метод parseBookBody виступає в ролі координатора процесу трансформації семантичної розмітки у візуальну. Він здійснює пошук тегів <body> у структурі DOM-дерева документа. Оскільки стандарт FictionBook 2.0 передбачає можливість існування кількох текстових блоків (наприклад,

основного тексту та приміток), у поточній реалізації фокус зосереджено на першому знайденому вузлі (item(0)), що містить головний літературний контент. Для оптимізації використання системних ресурсів метод ініціалізує StringBuilder, який накопичує результати роботи рекурсивного обробника. Після завершення обходу всього дерева вузлів, результуючий HTML-код зберігається в колекції bookText. Такий підхід забезпечує передбачуваність рендерингу, мінімізує навантаження на Garbage Collector та дозволяє миттєво передавати готовий контент до компонента WebView (рис. 3.5).

```
private void parseBookBody(Document document)
{
    NodeList bodyNodes =
document.getElementsByTagName("body");
    if (bodyNodes.getLength() > 0)
    {
        Node bodyNode = bodyNodes.item(0);
        StringBuilder htmlBuilder = new StringBuilder();
        extractTextFromNode(bodyNode, htmlBuilder);
        this.bookText.clear();
        this.bookText.add(htmlBuilder.toString());
    }
}
```

Рисунок 3.5 – Реалізація методу parseBookBody для ініціалізації обробки тіла документа

Метод getTagsContent реалізує логіку пакетного вилучення метаданих книги, використовуючи механізм аргументів змінної довжини (varargs). Таке архітектурне рішення дозволяє гнучко передавати довільний перелік імен тегів у межах одного виклику, що значно спрощує масштабування застосунку при необхідності обробки додаткових полів (наприклад, жанру, дати написання чи мови видання). Процес ініціалізується циклічним обходом переданого масиву рядків, де для кожного імені тегу викликається допоміжний метод getTagContent. Отримані значення централізовано присвоюються відповідним полям класу (title, author, publisher) через систему умовних перевірок. Використання валідації назви

тегу безпосередньо під час ітерації забезпечує сувору відповідність між структурою XML-документа та внутрішніми атрибутами об’єкта книги. Метод викликається на завершальному етапі парсингу, коли структура DOM-дерева вже повністю нормалізована. Це дозволяє гарантувати, що дані, які потрапляють у поля класу, є актуальними та синхронізованими з поточним відкритим файлом (рис. 3.6).

```
private void getTagsContent(Document document, String...
tagNames)
{
    for (String tagName : tagNames)
    {
        String tagContent = getTagContent(document,
tagName);
        if (tagName.equals("book-title"))
        {
            this.title = tagContent;
        }
        else if (tagName.equals("author"))
        {
            this.author = tagContent;
        }
        else if (tagName.equals("publisher"))
        {
            this.publisher = tagContent;
        }
    }
}
```

Рисунок 3.6 – Реалізація методу getTagsContent для пакетної обробки метатегів

Завершальною ланкою є допоміжний метод getTagContent, який виконує безпосереднє звернення до вузлів DOM-дерева. Окрім перевірки наявності вузла, метод виконує обов’язкову нормалізацію отриманого тексту за допомогою функції trim(). Це дозволяє видалити пробіли та символи перенесення рядків, які часто з’являються в XML-документах через особливості їхнього форматування. У разі відсутності тегу в структурі FB2-файлу, метод реалізує логіку динамічного формування заглушки: префікс Unknown комбінується з назвою тегу, де перша

літера автоматично переводиться у верхній регістр. Така реалізація виступає захисним бар'єром, який гарантує повернення валідного рядка у будь-якій ситуації, що повністю виключає виникнення помилок NullPointerException та забезпечує візуальну стабільність і передбачуваність інтерфейсу застосунку (рис. 3.7).

```
private String getTagContent(Document document, String
tagName)
{
    NodeList nodes =
document.getElementsByTagName(tagName);
    if (nodes.getLength() > 0)
    {
        return nodes.item(0).getTextContent().trim();
    }

    return "Unknown " + tagName.substring(0,
1).toUpperCase() + tagName.substring(1);
}
```

Рисунок 3.7 – Реалізація методу getTagContent для вилучення текстового вмісту тегу

Така декомпозиція алгоритму парсингу на окремі методи забезпечує високу підтримуваність коду та дозволяє легко розширювати функціонал застосунку, наприклад, додаючи підтримку нових метаданих або специфічних розділів книги. Відповідно до принципів чистої архітектури, це дозволяє локалізувати зміни: наприклад, впровадження модуля підтримки ілюстрацій потребує внесення змін лише у відповідний метод обробки бінарних даних. Це створює надійний фундамент для подальшої модернізації системи. Завдяки чіткому розділенню рівнів обробки даних та візуалізації, застосунок демонструє високий рівень гнучкості (flexibility). Це дозволяє не лише підтримувати поточний стан системи, а й забезпечує можливість інтеграції нових форматів та функціональних модулів без необхідності докорінної зміни архітектурного каркаса програми. Таким чином, розроблений інструментарій парсингу та

рендерингу файлів формату FB2 є завершеним, стабільним рішенням, яке готове до подальшого розширення до рівня багатофункціональної екосистеми читання.

3.3 Демонстрація результатів роботи та взаємодія з інтерфейсом користувача

Для перевірки та наочної демонстрації результатів роботи розробленого застосунку для читання електронних книг формату FB2 доцільно розглянути процес взаємодії користувача з його графічним інтерфейсом на всіх етапах експлуатації: від запуску програми до персоналізації налаштувань відображення тексту. Проєктування інтерфейсу базувалося на принципах ергономіки та самоочевидності функцій, що дозволяє мінімізувати час на освоєння програми користувачем.

Після запуску застосунку клас App за допомогою методу start ініціалізує головне вікно застосунку. Користувач бачить інтерфейс, розроблений із дотриманням принципів мінімалізму. Візуальна структура реалізована за допомогою декларативної розмітки FXML, що дозволило відокремити логіку обробки подій у контролері від графічного представлення. У центральній робочій області відображається початкова сторінка, сформована компонентом WebView. Її вміст генерується методом buildHtmlBody, який використовує об’єкт FB2Reader для підготовки даних. У верхній частині вікна розташована панель головного меню (MenuBar), що надає доступ до операцій із файлами та параметрів програми (рис. 3.8).

Для початку читання користувачеві необхідно завантажити електронну книгу. Для цього у меню «Файл» необхідно обрати «Відкрити файл». Програма викликає стандартне системне діалогове вікно вибору файлів (FileChooser). З метою захисту від помилкових дій користувача, у цьому вікні попередньо встановлено фільтр розширень (*.fb2), що дозволяє системі відображати та завантажувати виключно файли формату FB2.



Рис. 3.8 – Головне вікно застосунку після запуску

Після вибору файлу та натискання кнопки «Відкрити» система ініціалізує потік читання даних за допомогою засобів Java IO. Процес завантаження супроводжується миттєвою перевіркою доступу до файлу: у разі, якщо документ заблокований іншим процесом або пошкоджений на рівні файлової системи, програма коректно обробляє відповідну виняткову ситуацію, не допускаючи зависання графічного інтерфейсу (рис. 3.9).

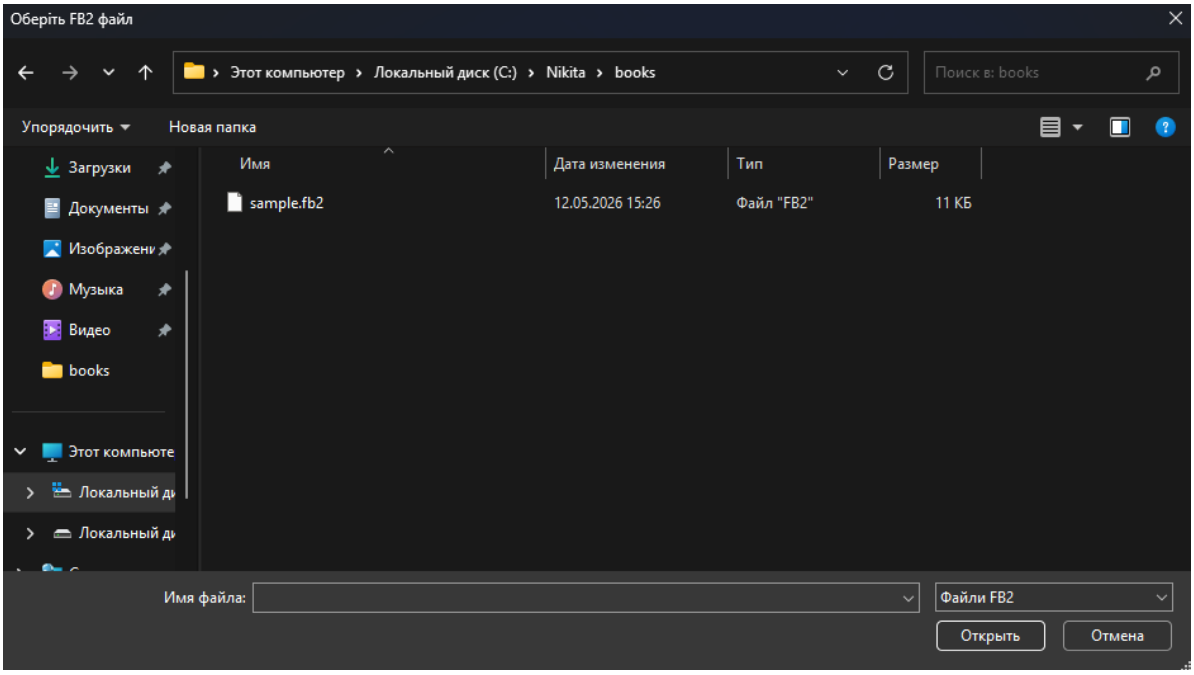


Рисунок 3.9 – Вікно вибору файлу FB2

Після успішного завершення семантичного аналізу документа модулем FB2Reader, застосунок автоматично ініціалізує подію оновлення графічного інтерфейсу для переходу в режим читання. Для ідентифікації поточного контенту заголовок головного вікна динамічно змінюється на назву книги, вилучену з метаданих файлу. Робочий простір вікна трансформується за допомогою компонента SplitPane, який розділяє область на дві функціональні зони з можливістю ручного регулювання їх меж:

- ліва панель (навігаційна): Реалізована на базі компонента ListView. Вона займає 25% ширини вікна та містить інтерактивний список розділів, сформований шляхом рекурсивного обходу тегів <section> внутрішньої структури документа;
- права панель (контентна): Базується на рушії WebEngine. Вона відображає згенерований HTML-документ, де перед основним текстом автоматично формується блок титульної інформації (назва, автор, видавництво), що забезпечує відповідність стандартам візуалізації електронних видань.

Важливою функцією реалізованого інтерфейсу є двостороння інтерактивність: при виборі елемента у списку розділів на лівій панелі, контролер ініціює виконання JavaScript-команди window.scrollTo() у середовищі WebEngine. Це дозволяє здійснювати миттєву навігацію до відповідного фрагмента тексту без перезавантаження сторінки. Окрім цього, використання компонента SplitPane дозволяє користувачеві адаптувати робочий простір під власні потреби, змінюючи співвідношення сторін панелей без втрати фокусу читання (рис 3.10).

Важливою функцією застосунку є зручна навігація по об’ємних творах. При натисканні на будь-який розділ у лівій панелі змісту, текст у правій робочій області миттєво прокручується до відповідного заголовка. Це досягається завдяки виконанню вбудованих JavaScript-скриптів безпосередньо у середовищі WebEngine, що забезпечує швидкий перехід без затримок і мерехтіння екрана.

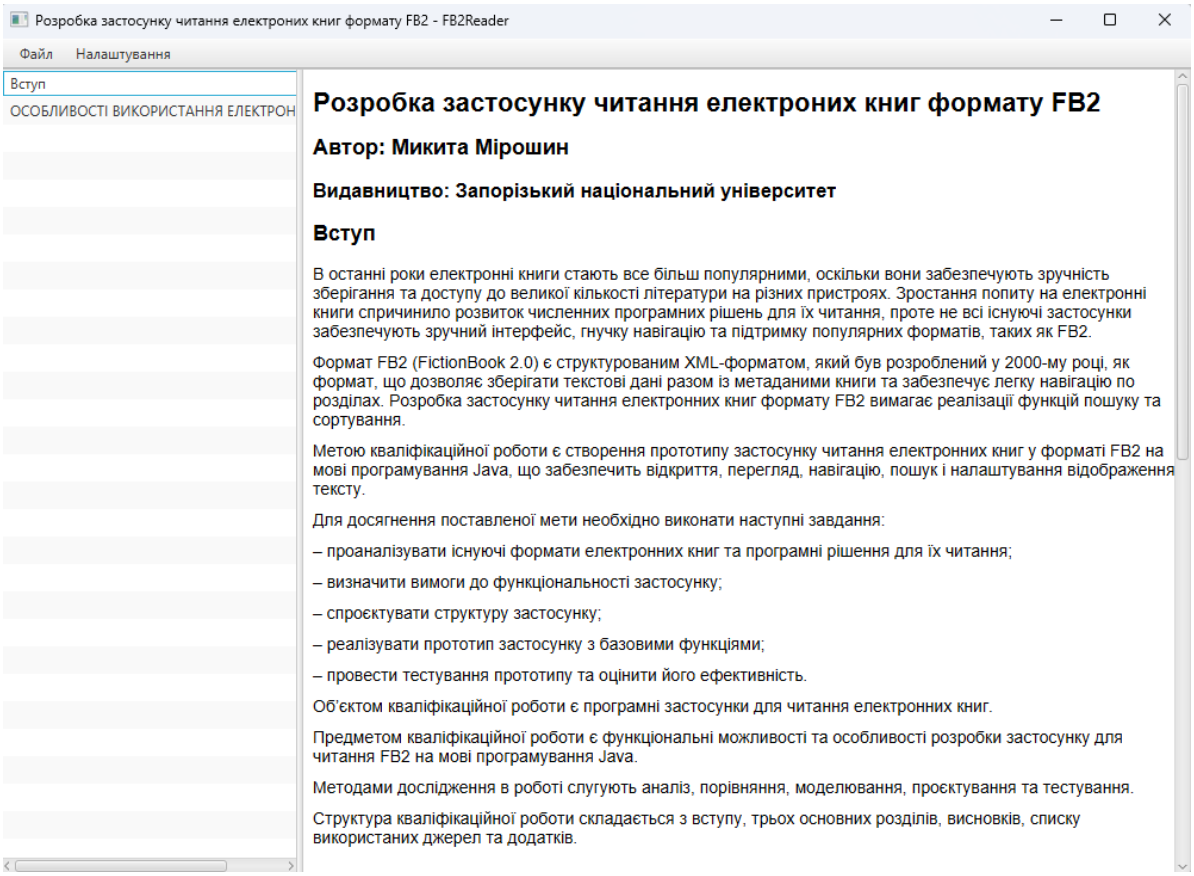


Рисунок 3.10 – Режим читання та навігації по книзі

Якщо користувачу потрібно адаптувати текст для комфортнішого читання, він може скористатися меню «Налаштування» та обрати «Налаштування шрифту». При виборі цього пункту з’являється додаткове вікно, де розташовано поле введення для вказання бажаного розміру шрифту в пікселях. Програма містить вбудовану валідацію: введене значення має бути цілим числом у діапазоні від 1 до 32. Після натискання кнопки «Зберегти» розмір тексту у відкритій книзі одразу змінюється (рис 3.11).

Після завершення сеансу читання, користувач може закрити поточний документ через меню «Файл» та обрати «Закрити файл». У результаті система коректно вивільняє ресурси: очищує навігаційну панель, прибирає текст книги з оперативної пам’яті та повертає інтерфейс до початкового екрана привітання.

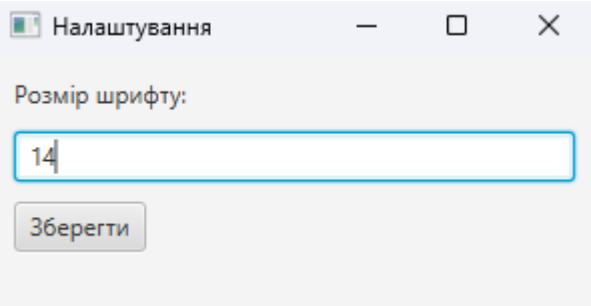


Рисунок 3.11 – Вікно налаштувань відображення шрифту

У випадку, якщо обраний користувачем файл формату FB2 виявиться пошкодженим або матиме невалідну XML-структуру, програма перехопить виняткову ситуацію та не припинить свою роботу екстрено. Замість цього в центральній області відобразиться інформаційна HTML-сторінка з повідомленням «Помилка завантаження книги», що свідчить про високий рівень надійності та відмовостійкості розробленого програмного забезпечення.

Механізм перехоплення виняткових ситуацій реалізований за допомогою блоків try-catch. У разі виявлення критичної помилки в XML-структурі, програма не лише виводить повідомлення, а й автоматично звільняє стек пам’яті від залишків некоректних даних, що запобігає витоку ресурсів (рис 3.12).

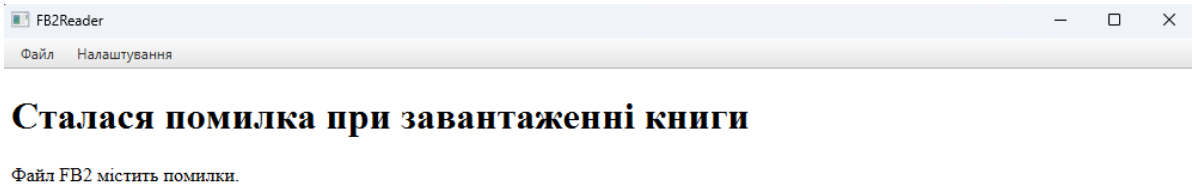


Рисунок 3.12 – Відображення інформаційного повідомлення про помилку завантаження книги

3.4 Шляхи модернізації та перспективи розвитку системи

Розроблений застосунок читання електронних книг формату FB2 є повністю функціональним продуктом, який успішно вирішує поставлену задачу базового завантаження та перегляду текстових документів. Проте закладена під час проєктування гнучка дворівнева архітектура (з чітким розділенням бізнес-логіки та графічного інтерфейсу) створює міцний фундамент для подальшого масштабування та вдосконалення системи.

Основними напрямками модернізації та перспективами розвитку розробленого програмного застосунку є:

- розширення підтримки форматів електронних книг: Завдяки принципу інверсії залежностей (Dependency Inversion), поточна архітектура дозволяє легко інтегрувати нові модулі парсингу без зміни логіки інтерфейсу. У найближчій перспективі доцільно реалізувати підтримку найпоширеніших форматів, таких як EPUB та MOBI, шляхом створення відповідних класів-наслідників, які реалізовуватимуть спільний з FB2Reader інтерфейс доступу до даних;
- поглиблення підтримки специфікації FB2: На поточному етапі алгоритм парсингу розпізнає структурні теги зображень. Перспективним кроком є реалізація повноцінного модуля для декодування бінарних зображень, закодованих у форматі Base64 всередині тегів <binary>, та їхньої динамічної ін'єкції в HTML-структуру книги під час рендерингу;
- інтеграція локальної бази даних: Для переходу від концепції звичайного «засобу перегляду файлів» до повноцінного «менеджера бібліотеки» (Library Manager) планується впровадження легкої вбудованої реляційної бази даних, такої як SQLite. Це дозволить застосунку зберігати історію відкритих книг, фіксувати позицію останнього прочитаного абзацу (збереження прогресу), а також надасть

користувачеві можливість створювати та керувати власними закладками і цитатами;

- вдосконалення користувацького інтерфейсу (UI/UX): Серед візуальних покращень пріоритетним є додавання підтримки «темної теми» (Dark Mode) та режиму сепії для комфортного читання в умовах слабкого освітлення. Також передбачається розширення налаштувань типографіки, що дозволить користувачам змінювати не лише розмір, але й гарнітуру шрифту, міжрядковий інтервал та колір фону;
- покращення навігації: Розширення функціоналу навігаційної панелі шляхом додавання повнотекстового пошуку по відкритій книзі з підсвічуванням знайдених збігів (Highlighting) безпосередньо у компоненті WebView;
- портування на мобільні платформи: Оскільки застосунок побудовано на технологіях Java та JavaFX, існує перспектива портування продукту на мобільні операційні системи (Android, iOS) за допомогою інструментарію GraalVM та фреймворку Gluon Mobile, що значно розширить цільову аудиторію та зробить процес читання ще зручнішим.

Окремим пріоритетом модернізації залишається забезпечення безпеки обробки вхідних даних шляхом посилення фільтрації XML-сутностей, що гарантуватиме захист користувача від потенційних вразливостей при відкритті файлів з неперевірених джерел.

Таким чином, створений застосунок має високий потенціал для модернізації. Впровадження запропонованих рішень дозволить органічно перетворити базовий застосунок на конкурентоспроможний, багатофункціональний продукт для комфортного читання та зручного керування цифровою бібліотекою.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було успішно досягнуто поставленої мети, яка полягала у проєктній реалізації та програмному впровадженні застосунку для читання електронних книг у форматі FB2 мовою Java.

Аналітичний огляд підтвердив, що XML-орієнтована структура формату FB2 є оптимальною для збереження семантичної побудови художніх творів, оскільки дозволяє відокремити контент від механізмів його візуалізації. Дослідження ринку програмних рішень виявило тенденцію до надмірної функціональної перевантаженості сучасних рідерів, що обґрунтовує доцільність створення лаконічного програмного продукту з пріоритетом на якість цифрової типографіки. Сформований перелік функціональних (XML-парсинг, ієрархічна навігація) та нефункціональних вимог (продуктивність, стабільність UI, адаптивність UX) став базисом для подальшого проєктування системи.

Архітектура програмного продукту була спроєктована з використанням засобів UML-моделювання, що дозволило чітко розмежувати рівень бізнес-логіки обробки даних (клас FB2Reader) та рівень графічного представлення (клас App). Такий підхід забезпечив високу модульність системи та створив умови для її подальшої масштабованості.

Тестування прототипу на файлах різного обсягу підтвердило стабільність алгоритмів парсингу та відсутність значних затримок при рендерингу. Перевірка відмовостійкості засвідчила коректну обробку некоректних або пошкоджених XML-структур із виведенням відповідних повідомлень користувачу.

Інтеграція бібліотеки JavaFX та рушія WebEngine дозволила задіяти механізми апаратного прискорення рендерингу, що гарантує високу якість типографіки та безперебійну роботу інтерфейсу навіть при опрацюванні об'ємних текстових масивів.

Розроблений програмний продукт має високий потенціал для впровадження на десктопних платформах як легковаговий та продуктивний інструмент. Перспективним напрямом модернізації є масштабування архітектури для підтримки формату EPUB шляхом розширення модулів парсингу. Впровадження локальної бази даних дозволить реалізувати систему збереження стану сесії та персоналізованих закладок. Подальша оптимізація графічного інтерфейсу та розширення налаштувань типографіки дозволить підвищити ергономічність застосунку та його конкурентоспроможність серед сучасних засобів візуалізації контенту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Alan Cooper, Robert Reimann, David Cronin, Christopher Noessel. About Face : The Essentials of Interaction Design. 4th ed. John Wiley & Sons, 2014. 720 p.

2. Brett McLaughlin, Justin Edelson. Java & XML : Data Binding, Processing, and Presentation. O'Reilly Media, 2006. 718 p.

3. Chris Buckley. XML Data Processing with Java : A Practical Guide. Academic Press, 2014. 288 p.

4. David Salomon. Data Compression : The Complete Reference. Springer, 2006. 1092 p.

5. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns : Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1994. 395 p.

6. Herbert Schildt. Java : The Complete Reference, Eleventh Edition. McGraw-Hill Education, 2018. 1344 p.

7. Ian Sommerville. Software Engineering : Tenth Edition. Pearson, 2015. 816 p.

8. John Whitington. PDF Explained : The ISO Standard for Document Exchange. O'Reilly Media, 2011. 144 p.

9. Joshua Tallent. Kindle Formatting : The Complete Guide to Formatting Books for the Amazon Kindle. Booksurge Publishing, 2009. 256 p.

10. Jukka K. Korpela. Unicode Explained : Internationalize Documents, Programs, and Web Sites. O'Reilly Media, 2006. 688 p.

11. Karl Wiegers, Joy Beatty. Software Requirements : Practical Techniques for Gathering and Managing Requirements Throughout the Product Development Cycle. Microsoft Press, 2013. 672 p.

12. Matt Garrish. What is EPUB 3? : An Introduction to the EPUB Specification. O'Reilly Media, 2011. 62 p.

13. Paul Salvette. The eBook Design and Development Guide : Professional eBook Formatting. BB eBooks, 2012. 300 p.

14. Robert C. Martin. Clean Architecture : A Craftsman’s Guide to Software Structure and Design. Prentice Hall, 2017. 432 p.

15. Robert C. Martin. Clean Code : A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008. 464 p.

16. Steve Krug. Don’t Make Me Think, Revisited : A Common Sense Approach to Web Usability. New Riders, 2014. 200 p.

17. Гавенко С. Ф., Мельников О. В. Електронні видання : навчальний посібник. Львів : Українська академія друкарства, 2015. 320 с.

18. Женченко М. І. Цифрові трансформації видавничої галузі : монографія. Київ : Жнець, 2018. 440 с.

19. Ткаченко В. П., Чеботарьова І. Б. Електронні видання та специфіка їх форматів : навчальний посібник. Харків : ХНУРЕ, 2019. 144 с.

20. Яковина В. С. Об’єктно-орієнтоване проєктування та архітектура програмного забезпечення : навчальний посібник. Львів : Видавництво Львівської політехніки, 2022. 216 с.

ДОДАТОК А

Програмний код

А.1 Код головного класу керування інтерфейсом App.java

```
package com.mykytamiroshyn;

import java.io.File;

import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.ListView;
import javafx.scene.control.Menu;
import javafx.scene.control.MenuBar;
import javafx.scene.control.MenuItem;
import javafx.scene.control.SplitPane;
import javafx.scene.control.TextField;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.VBox;
import javafx.scene.web.WebView;
import javafx.scene.web.WebEngine;
import javafx.stage.FileChooser;
import javafx.stage.Stage;

// Головний клас додатку, який відповідає за інтерфейс та
взаємодію з користувачем
public class App extends Application
{
    int fontSize = 14;

    public static void main(String[] args) throws
Exception
    {
        launch(args);
    }

    @Override
```

```
    public void start(Stage stage) throws Exception
    {
        WebView webView = new WebView();
        WebEngine webEngine = webView.getEngine();

        initStartupWindow(stage, webEngine);

        BorderPane root = new BorderPane();
        root.setCenter(webView);

        ListView<String> titlesListView = new
ListView<>();
        titlesListView.setPrefWidth(200);

        initMenuAndHandlers(stage, root, webView,
webEngine, titlesListView);

        Scene scene = new Scene(root, 1000, 700);
        stage.setScene(scene);
        stage.show();
    }

    // Метод для ініціалізації головного вікна додатку та
завантаження стартової сторінки
    private void initStartupWindow(Stage stage, WebEngine
webEngine)
    {
        stage.setTitle("FB2Reader");
        loadStartupPage(webEngine);
    }

    // Метод для ініціалізації меню та обробників подій
для пунктів меню та списку змісту
    private void initMenuAndHandlers(Stage stage,
BorderPane root, WebView webView, WebEngine webEngine,
ListView<String> titlesListView)
    {
        MenuBar menuBar = new MenuBar();
        Menu fileMenu = new Menu("Файл");
        Menu settingsMenu = new Menu("Налаштування");
        MenuItem openItem = new MenuItem("Відкрити
файл");
        MenuItem closeItem = new MenuItem("Закрити
файл");
```

```
        MenuItem settingsItem = new
MenuItem("Налаштування шрифту");

        fileMenu.getItems().add(openItem);
        fileMenu.getItems().add(closeItem);
        settingsMenu.getItems().add(settingsItem);
        menuBar.getMenus().add(fileMenu);
        menuBar.getMenus().add(settingsMenu);
        initHandlers(stage, root, webView, webEngine,
titlesListView, openItem, settingsItem, closeItem);

        root.setTop(menuBar);
    }

    // Метод для ініціалізації обробників подій для
пунктів меню та списку змісту
    private void initHandlers(Stage stage, BorderPane
root, WebView webView, WebEngine webEngine,
ListView<String> titlesListView, MenuItem openItem,
MenuItem settingsItem, MenuItem closeItem)
    {
        openItem.setOnAction(e -> {
            openFile(webEngine, webView, root, stage,
titlesListView);
        });

        settingsItem.setOnAction(e -> {
            showSettingsWindow(webEngine);
        });

        closeItem.setOnAction(e -> {
            closeCurrentBookAndLoadStartupPage(webEngine,
webView, root, stage, titlesListView);
        });


titlesListView.getSelectionModel().selectedItemProperty()
.addListener((observable, oldValue, newValue) -> {
            if (newValue != null)
            {
                webEngine.executeScript("var elements =
document.getElementsByTagName('title');" +
                "for (var i = 0; i < elements.length;
i++) {" +
```

```
                "if (elements[i].textContent === '" +
newValue + "') {" +
                "elements[i].scrollIntoView();" +
                "break;" +
                "}" +
                "}");
            }
        });
    }

    // Метод для завантаження стартової сторінки з
привітанням
    private void loadStartupPage(WebEngine webEngine)
    {
        String startupHtmlContent = "<!DOCTYPE
html><html><head><meta charset='UTF-
8'><title>FB2Reader</title></head><body>" +
        "<h1>Ласкаво просимо до FB2Reader!</h1>" +
        "</body></html>";
        webEngine.loadContent(startupHtmlContent);
    }

    // Метод для завантаження сторінки з помилкою
    private void loadErrorPage(WebEngine webEngine,
String errorMessage)
    {
        String errorHtmlContent = "<!DOCTYPE
html><html><head><meta charset='UTF-
8'><title>Помилка</title></head><body>" +
        "<h1>Сталася помилка при завантаженні книги</h1>"
+
        "<p>" + errorMessage + "</p>" +
        "</body></html>";
        webEngine.loadContent(errorHtmlContent);
    }


    // Метод для завантаження книги з FB2 файлу та
відображення її вмісту у WebView
    private void loadBook(WebEngine webEngine, WebView
webView, Stage stage, ListView<String> titlesListView,
BorderPane root, String filePath)
    {
        FB2Reader reader = new FB2Reader();
        try
```

```
        {
            int result = reader.parseBook(filePath);
            if (result == 1 && reader.getTitle() != null
&& reader.getAuthor() != null && reader.getPublisher() !=
null)
            {
                stage.setTitle(reader.getTitle() + " -
FB2Reader");
                titlesListView.getItems().clear();
                if (!reader.getTitlesTable().isEmpty())
                {

titlesListView.getItems().addAll(reader.getTitlesTable())
;
                }

                SplitPane splitPane = new SplitPane();

splitPane.getItems().addAll(titlesListView, webView);
                splitPane.setDividerPositions(0.25);

                root.setCenter(splitPane);

                String bookHtml = "<!DOCTYPE
html><html><head><meta charset='UTF-8'><title>" +
reader.getTitle() +
                        "</title><style>body { font-size:
" + fontSize + "px; }</style></head><body>" +
                        "<h1>" + reader.getTitle() +
"</h1>" +
                        "<h2>Автор: " +
reader.getAuthor() + "</h2>" +
                        "<h3>Видавництво: " +
reader.getPublisher() + "</h3>" +
                        buildHtmlBody(reader) +
                        "</body></html>";

                webEngine.loadContent(bookHtml);
            }
            else
            {
                loadErrorPage(webEngine, "Файл FB2
містить помилки.");
                root.setCenter(webView);
            }
```

```
        }
        catch (Exception e)
        {
            loadErrorPage(webEngine, "Не вдалося
завантажити файл");
            root.setCenter(webView);
        }
    }

    // Метод для відкриття діалогового вікна вибору файлу
та завантаження вибраного FB2 файлу
    private void openFile(WebEngine webEngine, WebView
webView, BorderPane root, Stage stage, ListView<String>
titlesListView)
    {
        FileChooser fileChooser = new FileChooser();
        fileChooser.setTitle("Оберіть FB2 файл");
        fileChooser.getExtensionFilters().addAll(new
FileChooser.ExtensionFilter("Файли FB2", "*.fb2"));
        File selectedFile =
fileChooser.showOpenDialog(stage);

        if (selectedFile != null)
        {
            loadBook(webEngine, webView, stage,
titlesListView, root, selectedFile.getAbsolutePath());
        }
    }

    // Метод для відображення вікна налаштувань
    private void showSettingsWindow(WebEngine webEngine)
    {
        Stage settingsStage = new Stage();
        settingsStage.setTitle("Налаштування");

        Label fontSizeLabel = new Label("Розмір
шрифту:");
        TextField fontSizeField = new
TextField(String.valueOf(fontSize));
        Button saveButton = new Button("Зберегти");

        saveButton.setOnAction(event -> {
            saveSettings(webEngine, fontSizeField);
        });
```

```
        VBox layout = new VBox(10);
        layout.setPadding(new Insets(10));
        layout.getChildren().addAll(fontSizeLabel,
fontSizeField, saveButton);

        settingsStage.setScene(new Scene(layout, 300,
150));
        settingsStage.show();
    }

    // Метод для збереження налаштувань
    private void saveSettings(WebEngine webEngine,
TextField fontSizeField)
    {
        try
            {
                int newFontSize =
Integer.parseInt(fontSizeField.getText());
                if (newFontSize > 0 && newFontSize <= 32)
                {
                    fontSize = newFontSize;

webEngine.executeScript("document.body.style.fontSize =
'" + fontSize + "px';");
                }
                else
                {

fontSizeField.setText(String.valueOf(fontSize));
                }
            }
            catch (NumberFormatException e
            )
            {

fontSizeField.setText(String.valueOf(fontSize));
            }
    }

    // Метод для закриття поточної книги та завантаження
стартової сторінки
    private void
closeCurrentBookAndLoadStartupPage(WebEngine webEngine,
WebView webView, BorderPane root, Stage stage,
ListView<String> titlesListView)
```

```
    {
        closeCurrentBook(webEngine, webView, root, stage,
titlesListView);
        loadStartupPage(webEngine);
    }

    // Метод для закриття поточної книги та очищення
інтерфейсу
    private void closeCurrentBook(WebEngine webEngine,
WebView webView, BorderPane root, Stage stage,
ListView<String> titlesListView)
    {
        titlesListView.getItems().clear();
        stage.setTitle("FB2Reader");
        root.setCenter(webView);
    }

    // Метод для побудови HTML вмісту книги з урахуванням
стилів та форматування
    private String buildHtmlBody(FB2Reader reader)
    {
        StringBuilder stringBuilder = new
StringBuilder();
        stringBuilder.append("<style>");
        stringBuilder.append("body { font-family: Arial,
sans-serif; font-size: ");
        stringBuilder.append(fontSize).append("px; }");
        stringBuilder.append("strong { font-weight:
bold; }");
        stringBuilder.append("title { font-size:
").append(fontSize + 6).append("px; margin: 20px 0
10px; }");
        stringBuilder.append("h1 { font-size:
").append(fontSize + 8).append("px; }");
        stringBuilder.append("h2 { font-size:
").append(fontSize + 4).append("px; }");
        stringBuilder.append("p { margin: 10px 0; }");
        stringBuilder.append("p strong { margin: 0; }");
        stringBuilder.append("p:empty, p br { margin:
0; }");
        stringBuilder.append(".section { margin-bottom:
20px; }");
        stringBuilder.append("</style>");

        for (String paragraph : reader.getBookText())
```

```
        {
            stringBuilder.append(paragraph);
        }
        return stringBuilder.toString();
    }
}
```

A.2 Код класу обробки та парсингу файлів FB2Reader.java

```
package com.mykytamiroshyn;

import java.io.File;
import java.util.ArrayList;
import java.util.List;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.*;

// Клас для читання та парсингу FB2 файлів
public class FB2Reader
{
    private String title;
    private String author;
    private String publisher;
    private List<String> titlesTable;
    private List<String> bookText;

    public FB2Reader()
    {
        this.title = "Unknown Title";
        this.author = "Unknown Author";
        this.publisher = "Unknown Publisher";
        this.titlesTable = new ArrayList<>();
        this.bookText = new ArrayList<>();
    }

    // Метод для парсингу FB2 файлу та отримання його
вмісту
    public int parseBook(String filePath)
    {
```

```
        File file = new File(filePath);
        if (!file.exists() || !file.isFile())
        {
            return 0;
        }

        try
        {
            Document document = buildDocument(file);

            parseTitlesTable(document);

            parseBookBody(document);

            getTagsContent(document, "book-title",
"author", "publisher");

            return 1;
        }
        catch (Exception e)
        {
            e.printStackTrace();
            return 0;
        }
    }

    // Метод для побудови Document з FB2 файлу
    private Document buildDocument(File file) throws
Exception
    {
        DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();

factory.setFeature("http://apache.org/xml/features/disall
ow-doctype-decl", true);

        DocumentBuilder builder =
factory.newDocumentBuilder();
        Document document = builder.parse(file);
        document.getDocumentElement().normalize();

        return document;
    }

    // Метод для парсингу таблиці змісту з FB2 файлу
```

```
    private void parseTitlesTable(Document document)
    {
        titlesTable.clear();
        NodeList sections =
document.getElementsByTagName("section");
        for (int i = 0; i < sections.getLength(); i++)
        {
            Node section = sections.item(i);
            NodeList childNodes =
section.getChildNodes();
            for (int j = 0; j < childNodes.getLength();
j++)
            {
                Node child = childNodes.item(j);
                if (child.getNodeName().equals("title"))
                {
                    String titleText =
child.getTextContent().trim();
                    if (!titleText.isEmpty())
                    {
                        titlesTable.add(titleText);
                    }
                    break;
                }
            }
        }
    }

    // Метод для парсингу основного тексту книги з FB2
файлу
    private void parseBookBody(Document document)
    {
        NodeList bodyNodes =
document.getElementsByTagName("body");
        if (bodyNodes.getLength() > 0)
        {
            Node bodyNode = bodyNodes.item(0);
            StringBuilder htmlBuilder = new
StringBuilder();
            extractTextFromNode(bodyNode, htmlBuilder);
            this.bookText.clear();
            this.bookText.add(htmlBuilder.toString());
        }
    }
```

```
    // Метод для отримання вмісту тегів з FB2 файлу
    private void getTagsContent(Document document,
String... tagNames)
    {
        for (String tagName : tagNames)
        {
            String tagContent = getTagContent(document,
tagName);
            if (tagName.equals("book-title"))
            {
                this.title = tagContent;
            }
            else if (tagName.equals("author"))
            {
                this.author = tagContent;
            }
            else if (tagName.equals("publisher"))
            {
                this.publisher = tagContent;
            }
        }
    }

    // Метод для отримання вмісту певного тегу з FB2
файлу
    private String getTagContent(Document document,
String tagName)
    {
        NodeList nodes =
document.getElementsByTagName(tagName);
        if (nodes.getLength() > 0)
        {
            return nodes.item(0).getTextContent().trim();
        }

        return "Unknown " + tagName.substring(0,
1).toUpperCase() + tagName.substring(1);
    }

    // Рекурсивний метод для отримання тексту з вузла та
його нащадків
    private void extractTextFromNode(Node node,
StringBuilder htmlBuilder)
    {
        String nodeName = node.getNodeName();
```

```
        int nodeType = node.getNodeType();

        if (nodeType == Node.TEXT_NODE)
        {
            String text = node.getTextContent();
            if (!text.replace("\n", "").replace("\r",
"").isEmpty())
            {
                htmlBuilder.append(text);
            }

            return;
        }

        boolean isElement = true;

        if (nodeName.equals("section"))
        {
            htmlBuilder.append("<div class='section'>");
        }
        else if (nodeName.equals("image"))
        {
            Element imgElement = (Element) node;
            String href =
imgElement.getAttribute("l:href");
            if (href.isEmpty())
            {
                href = imgElement.getAttribute("href");
            }

            String imageId = href.replace("#", "");

            htmlBuilder.append("<img
src='").append(imageId).append("' style='max-width:100%;
display:block; margin: 10px auto;'>");

            isElement = false;
        }
        else if (nodeName.equals("title"))
        {
            htmlBuilder.append("<h2>");
        }
        else if (nodeName.equals("p"))
        {
            htmlBuilder.append("<p>");
```

```
        }
        else if (nodeName.equals("strong"))
        {
            htmlBuilder.append("<strong>");
        }
        else if (nodeName.equals("emphasis"))
        {
            htmlBuilder.append("<i>");
        }
        else
        {
            isElement = false;
        }

        NodeList childNodes = node.getChildNodes();
        if (childNodes.getLength() == 0 &&
nodeName.equals("p"))
        {
                htmlBuilder.append("<br>");
        }
        else
        {
            for (int i = 0; i < childNodes.getLength();
i++)
            {
                extractTextFromNode(childNodes.item(i),
htmlBuilder);
            }
        }

        if (isElement)
        {
            if (nodeName.equals("section"))
            {
                htmlBuilder.append("</div>");
            }
            else if (nodeName.equals("title"))
            {
                htmlBuilder.append("</h2>");
            }
            else if (nodeName.equals("p"))
            {
                htmlBuilder.append("</p>");
            }
            else if (nodeName.equals("strong"))
```

```
                {
                    htmlBuilder.append("</strong>");
                }
                else if (nodeName.equals("emphasis"))
                {
                    htmlBuilder.append("</i>");
                }
            }
        }

        // Геттер для отримання назви книги
        public String getTitle()
        {
            return title;
        }

        // Геттер для отримання автора книги
        public String getAuthor()
        {
            return author;
        }

        // Геттер для отримання видавця книги
        public String getPublisher()
        {
            return publisher;
        }

        // Геттер для отримання тексту книги
        public List<String> getBookText()
        {
            return bookText;
        }

        // Геттер для отримання таблиці змісту книги
        public List<String> getTitlesTable()
        {
            return titlesTable;
        }
    }
```