КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теорії та технології програмування

Кваліфікаційна робота
на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки
на тему:

БІБЛІОТЕКА ДЛЯ РОБОТИ З ЕЛЕКТРОННИМИ КНИГАМИ ДЛЯ ОС АНДРОЇД

Виконав студент 4-го курсу
Володимир ЧУЧКАНОВ

(підпис)

Науковий керівник:
асистент, кандидат фіз.-мат. наук
Андрій КРИВОЛАП

(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теорії та технології програмування
«____» _____202_ р.,
протокол № ___ Завідувач кафедри
Завідувач кафедри
Микола НІКІТЧЕНКО

(підпис)

Київ — 2023

РЕФЕРАТ

Обсяг роботи 44 сторінки, 29 ілюстрацій, 2 таблиці, 17 джерел посилань.

ANDROID, БІБЛІОТЕКИ, JAVA, FB2, MOBILE, GRADLE, ПЕРЕГЛЯД ФАЙЛІВ, АЛГОРИТМИ ПОШУКУ, ІНВЕРТОВАНИЙ ІНДЕКС, НАВІГАЦІЯ.

Об’єктом роботи є бібліотеки для читання електронних книг для ОС Андроїд. Предметом роботи є програмна реалізація бібліотеки на мові Java.

Метою роботи є створення бібліотеки для забезпечення взаємодії між файлами електронних книг та додатком Андроїд.

Методи розробки: комп’ютерне моделювання, розробка програмного забезпечення, проєктування структури бібліотеки. Інструменти розробки: інтегроване середовище розробки Android Studio, інтегроване середовище розробки Intellij IDEA, мова програмування Java.

Результат роботи: досліджені наявні аналоги, розглянуті їх переваги та недоліки. Розроблено бібліотеку “filereader”, яка надає розробнику можливість переглядати файли електронних книг, систему навігації для цих файлів, а також пошук фрагментів тексту у них. Також бібліотека дає змогу налаштовувати графічне відображення тексту з файлів, змінюючи тип або розмір шрифту, задній фон, розмір вікна з текстом. Також розроблена можливість змінювати мову, якою відображаються усі назви та теги інтерфейсу застосунку.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

DOM — Document Object Model, об’єктна модель документа;

IDE – Integrated Design Environment, інтегроване середовище розробки;

SDK — Software Development Kit, набір із засобів розробки;

EPUB — electronic publication, відкритий стандарт формату електронних книг;

FB2 — FictionBook, формат електронних книг у вигляді XML-документів;

PDF — Portable Document Format, формат файлу для представлення двовимірних документів у незалежному від пристрою виведення та роздільної здатності вигляді;

FXL — Fixed Layout, формат файлу;

CBZ — Comic Book ZIP, формат файлу;

КМП — алгоритм пошуку Кнута-Морріса-Пратта;

БМ — алгоритм пошуку Боєра-Мура;

ВСТУП

Оцінка сучасного стану об’єкта розробки. На цей день ОС Андроїд є найпопулярнішою у світі для мобільних пристроїв [1]. Мобільний пристрій є майже у кожного і доступний для використання без особливих вимог до місця і часу. Тому мобільні додатки користуються популярністю, а серед них і читалки для книг.

На сьогодні бібліотеки для читання електронних книг для ОС Андроїд є досить розвинутими та широко використовуваними. Існує багато відкритих та комерційних бібліотек, які надають розробникам інструменти для роботи з електронними книгами на платформі Андроїд.

Ці бібліотеки зазвичай надають можливості для відображення книг, роботи з різними форматами (наприклад, EPUB, PDF), підтримку налаштувань шрифтів, розмірів, кольорів, можливості пошуку, закладок, зміни розміру тексту, організації змісту та інші функції, що сприяють зручному та зручному читанню електронних книг на пристроях з операційною системою Андроїд.

Бібліотеки для читання електронних книг для ОС Андроїд забезпечують розробникам можливість реалізувати функціонал читання електронних книг у своїх додатках. Вони спрощують завдання з роботи з електронними книгами, забезпечуючи готові компоненти та API для інтеграції з додатками.

Актуальність роботи та підстави для її виконання. Мільйони користувачів [2] завантажують собі додатки для читання електронних, тому вимоги до їх розробників досить високі. Через це задача розробки бібліотеки, яка спростить розробникам задачу написання подібних додатків є вельми актуальною. Попри те, що зараз існують готові аналоги, всі вони мають як свої переваги, так і недоліки. Тому проаналізувавши наявні рішення можна розробити бібліотеку, яка вирішить наявні в аналогах проблеми, та додасть щось нове.

Мета й завдання роботи. Метою кваліфікаційної роботи є розробка бібліотеки “filereader” для забезпечення взаємодії між файлами електронних книг та додатком Андроїд. Для досягнення цієї мети поставлено такі завдання.

- Дослідити наявні рішення для роботи з файлами електронних книг
- Визначити який функціональні вимоги до бібліотеки
- Спроєктувати архітектуру бібліотеки
- Реалізувати бібліотеку

Об'єкт, методи й інструменти розроблення. Об’єктом розроблення є бібліотеки для читання електронних книг для ОС Андроїд. Предметом є розроблення бібліотеки що реалізує базовий інтерфейс для взаємодії додатка Андроїд та файлів електронних книг.

Методами розроблення є комп’ютерне моделювання, розробка програмного забезпечення, проєктування структури бібліотеки.

Інструментами розроблення є інтегроване середовище розробки Android Studio — для основної роботи, Intellij IDEA — для проєктування та налагодження функціонала, який не потребував взаємодії з засобами Android, мова програмування Java, система збірки Gradle.

Можливі сфери застосування. Бібліотеку “filereader” можна використовувати для розробки додатків для ОС Андроїд, що вимагають коректного відображення тексту та зображень зчитаних з файлів електронних книг, а також можливості навігуватися по тексту та змінювати стилістичні параметри.

РОЗДІЛ 1. ОГЛЯД НАЯВНИХ РІШЕНЬ ТА ІНСТРУМЕНТІВ РОЗРОБКИ

1.1 Огляд наявних рішень

На цей момент існує декілька основних бібліотек для мобільних додатків і роботи з файлами. Розглянемо та проаналізуємо їх.

1.1.1 Readium Mobile

Бібліотека “Readium Mobile” [3], надає потужні інструменти та компоненти, які дозволяють розробникам створювати функціональні додатки для читання електронних книжок на мобільних платформах. Головними особливостями “Readium Mobile” є:

- Підтримка різних форматів: “Readium Mobile” надає можливість роботи з файлами EPUB, PDF, FXL, RTL, CBZ. Також присутні можливості навігуватися по інформації з таких файлів, та виконувати пошук у файлах формату EPUB.
- Кросплатформеність: Проєкт "Readium Mobile" підтримує розробку на різних мобільних платформах, включаючи Android та iOS. Це дозволяє розробникам створювати додатки, які можуть працювати на різних пристроях та операційних системах.
- Компонентна архітектура: Бібліотека має модульну структуру, що дозволяє розробникам використовувати лише потрібні компоненти та функції для своїх додатків. Це спрощує розробку та дозволяє гнучко налаштовувати функціонал.
- Налаштування вигляду та стилю: Проєкт "Readium Mobile" надає можливість налаштовувати вигляд та стиль відображення тексту, включаючи розмір шрифту, колір, вирівнювання та інші параметри. Це дозволяє створювати персоналізовані додатки для читання електронних книжок.

Основними перевагами “Readium Mobile” є велика гнучкість налаштувань — розробнику надається інтерфейс за допомогою можна змінювати багато параметрів — таких як тип або розмір тексту тощо. Проте це водночас є і недоліком даного продукту — наявна документація описує можливості бібліотеки неякісно і неповно. Також можливість пошуку лише по EPUB файлам є значним недоліком.

1.1.2 FolioReader-Android

“FolioReader-Android” [4] це Бібліотека для розробки додатків для читання електронних книжок у форматі EPUB на платформі Android. Цей проєкт надає прості та потужні інструменти, які дозволяють розробникам створювати функціональні додатки для читання книжок у форматі EPUB. Головними особливостями даного продукту є:

- Підтримка формату EPUB: Бібліотека дозволяє відкривати та відтворювати електронні книжки у форматі EPUB. Він підтримує основні функції, такі як навігація по сторінках, збільшення та зменшення розміру шрифту, підсвічування тексту та багато інших.
- Пристосовуваність до дизайну: Бібліотека надає можливість налаштовувати вигляд і поведінку читального інтерфейсу. Розробники можуть змінювати кольори, шрифти, фонові зображення та інші елементи для створення унікального дизайну додатка (рисунок 1.2.1).
- Швидкість імплементації: Бібліотека "FolioReader-Android" може бути імплементована у додаток за досить невелику кі-сть кроків. (рисунок 1.2.2)

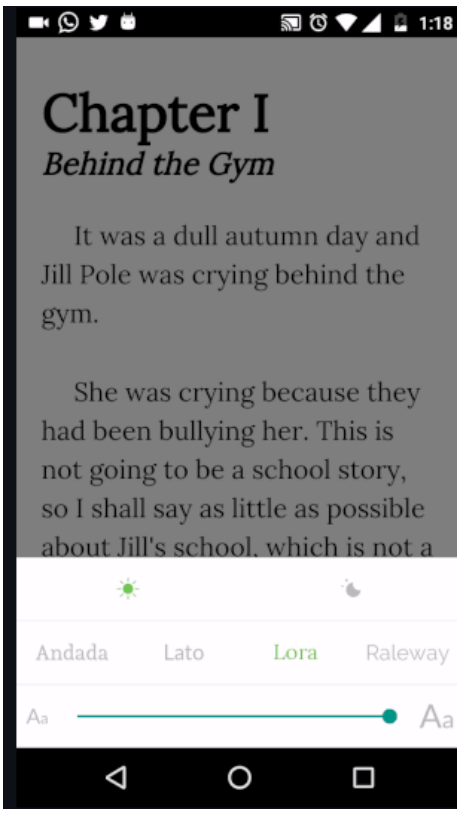

Рисунок 1.1.2.1 — можливості "FolioReader-Android"

Рисунок 1.1.2.2 — підключення бібліотеки “FolioReader-Android"

До переваг даного продукту можна віднести його відносну простоту та ефективність — підключення бібліотеки добре описано на офіційному репозиторії. Проте відсутність документації по майже усім фічам є досить великим недоліком, через який може бути важко розібратися у всіх запропонованих можливостях, або навіть дізнатися про них.

1.1.3 epublib

Проєкт "epublib" [5] є бібліотекою для роботи з електронними книжками у форматі EPUB на платформі Java/Android. Цей проєкт надає розробникам інструменти для створення, читання та модифікації файлів EPUB. Головними особливостями "epublib" є:

- Робота з форматом EPUB: Бібліотека дозволяє відкривати, читати та обробляти електронні книжки у форматі EPUB. Вона підтримує стандартні функції, такі як навігація по сторінках, вибір шрифтів, збільшення та зменшення розміру шрифту, вміст книжки та інші.
- Підтримка метаданих: Бібліотека дозволяє отримувати та змінювати метадані електронних книжок, такі як назва, автор, видавництво, ISBN та інші. Це дозволяє розробникам працювати з інформацією про книжку та використовувати її для відображення та керування даними.
- Робота з ресурсами: Бібліотека дозволяє отримувати доступ до ресурсів електронних книжок, таких як зображення, CSS-стилі, HTML-сторінки та інші. Розробники можуть використовувати ці ресурси для відображення та налаштування вмісту книжки.
- Підтримка розширень: Проєкт "epublib" надає можливість розширення функціонала за допомогою власних розширень. Розробники можуть додавати власні функції та розширювати можливості бібліотеки згідно зі своїми потребами.

Хоча "epublib" надає гнучкий та потужний функціонал роботи з безпосередньо файлами EPUB, проте дане рішення не дає ніяких інструментів для відображення інформації безпосередньо у застосунку Android що може бути як недоліком, так і перевагою, в залежності від потреб розробника.

1.1.4 FBReader SDK

“FBReader SDK” [6] дозволяє розробникам інтегрувати функціонал читання FB2 книг у свої додатки. За допомогою цього SDK розробники можуть створювати зручні та функціональні додатки для читання книг у форматі FB2. SDK надає потужні інструменти та можливості для контролю над процесом читання, налаштування вигляду книги, роботи зі сторінками та багато іншого.

“FBReader SDK” надає розробникам можливість створювати власні застосунки для читання електронних книг у форматі FB2 зі значною гнучкістю та можливостями налаштування. Завдяки цьому, розробники можуть створювати різнопланові додатки для задоволення потреб читачів електронних книг.

Основним недоліком FBReader є те, що базовий безплатний функціонал дуже обмежений (рисунок 1.4.1), а вартість повної версії занадто велика для малих проєктів.

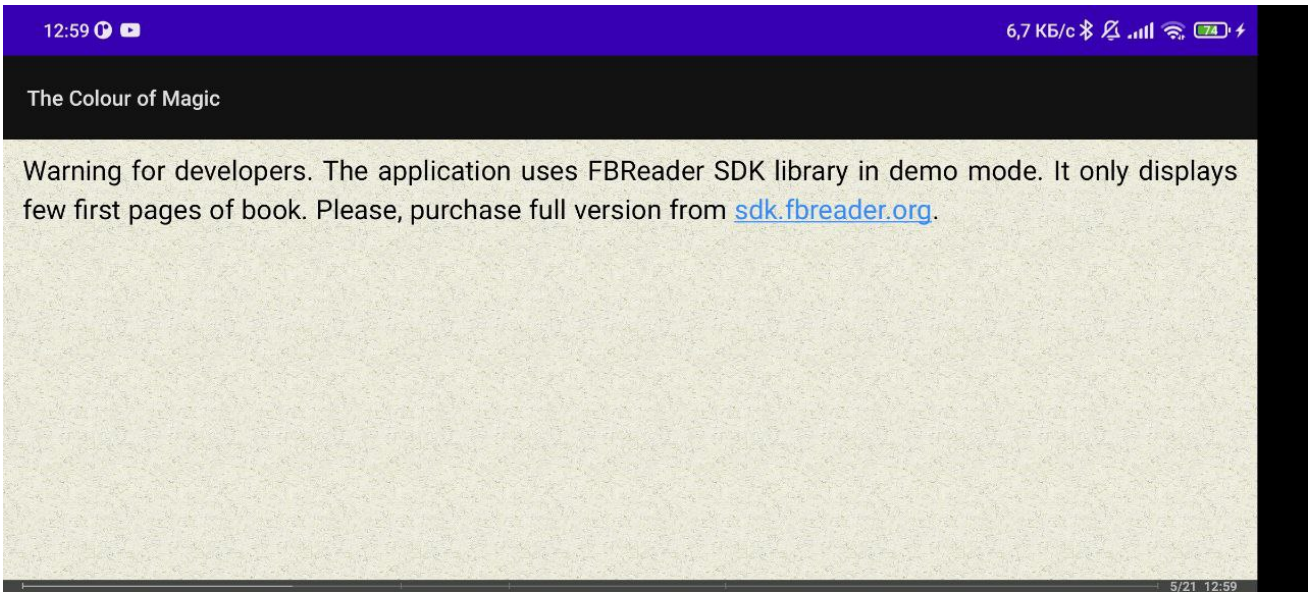

Рисунок 1.1.4.1 — безплатна версія “FBReader SDK”

1.1.5 Існуючі рішення — підсумок

У таблиці 1.1.1 зведено основну інформацію про наявні рішення. Надано порівняння можливостей розглянутих рішень.

Таблиця 1.1.1 — зведена таблиця

Назва	**Формати файлів**	**Пошук у файлі**	**Навігація**
Readium Mobile	EPUB, PDF, CBZ, FXL, RTL	частково	так
FolioReader-Android	EPUB	так	так
epublib	EPUB	ні	частково
FBReader	FB2	так	так

1.2 Огляд інструментів розробки

Вибір набору інструментів обумовлений тим, що Android Studio є популярним вибором для розробки додатків та бібліотек для ОС Андроїд, через наявність великої кі-сті вбудованих інструментів розробки, в тому числі й систему збірки Gradle. Мова Java є одною зі стандартних для розробки під ОС Андроїд. Незважаючи на те, що зараз існує більш сучасна мова Kotlin, Java була обрана через наявність досвіду роботи з нею.

1.2.1 Java

Java — це об'єктно-орієнтована мова програмування [7], яка була розроблена компанією Sun Microsystems у 1995 році. Вона стала однією з найпопулярніших мов програмування завдяки своїй простоті, надійності та широкому спектру застосувань. Особливості мови Java:

- Платформонезалежність: Java використовує віртуальну машину Java (JVM), яка перетворює компільований Java-код у машинний код, зрозумілий для операційної системи. Це означає, що програми Java можуть працювати на будь-якій платформі, яка має відповідну реалізацію JVM.

- Об'єктно-орієнтований підхід: Java побудована на принципах об'єктно-орієнтованого програмування. Вона підтримує класи, об'єкти, спадкування, поліморфізм та інші концепції ООП, що сприяють модульності, повторному використанню коду та зручності розробки програм.
- Безпека: Java має вбудовану систему безпеки, яка дозволяє запускати код у пісочниці (sandbox) і обмежує доступ до ресурсів комп'ютера. Це забезпечує високий рівень захисту від шкідливого коду та можливих вразливостей.
- Збирання сміття: Java використовує автоматичне управління пам'яттю, що означає, що розробникам не потрібно вручну вивільняти пам'ять, використовану об'єктами. Збирання сміття автоматично визначає непотрібні об'єкти та вивільняє пам'ять під час виконання програми.
- Багатопоточність: Java надає вбудовану підтримку для багатопотокового програмування, що дозволяє виконувати одночасно кілька потоків в одній програмі. Це сприяє покращенню продуктивності та швидкодії програми, особливо в сучасних багатопроцесорних системах.
- Велика екосистема: Java має широку підтримку і активну спільноту розробників. Існує велика кількість сторонніх бібліотек, фреймворків та інструментів, які спрощують розробку програм на Java.
- Широкий спектр застосувань: Java використовується для розробки різноманітних програм, від мобільних додатків до веб-серверів, корпоративних систем та великих масштабних проєктів.

1.2.2 Android Studio

Android Studio — це інтегроване середовище розробки (IDE) для платформи Android, яке було розроблено компанією Google [8]. Воно є основним інструментом для розробки мобільних додатків для Android та надає широкі

можливості для створення високоякісних додатків з використанням різних технологій та мов програмування.

Особливості Android Studio:

- Інтуїтивний інтерфейс: Android Studio має зручний інтерфейс користувача, що дозволяє розробникам зручно працювати з проєкт. Він має багатофункціональні панелі, редактори коду та інші інструменти, що сприяють зручності та продуктивності розробки.
- Підтримка мов програмування: Android Studio підтримує різні мови програмування, такі як Java, Kotlin та C++. Це дозволяє розробникам вибирати мову, з якою вони найкраще працюють або яка найкраще підходить для конкретного проєкт.
- Засоби розробки інтерфейсу користувача: Android Studio надає вбудовані засоби для розробки інтерфейсу користувача, такі як редактор макетів інтерфейсу, палітра компонентів та переглядач макетів. Це допомагає розробникам швидко створювати та налаштовувати елементи користувача.
- Налагодження та профілювання: Android Studio надає потужні засоби для налагодження та профілювання додатків. Розробники можуть використовувати вбудовані засоби налагодження, переглядати логи, аналізувати продуктивність додатка та знаходити помилки.
- Інтеграція з Android SDK: Android Studio поставляється з комплектом Android SDK, який містить всі необхідні бібліотеки, інструменти та документацію для розробки на платформі Android. Це дозволяє розробникам швидко створювати, компілювати та випробовувати свої додатки.
- Підтримка інструментів сторонніх розробників: Android Studio дозволяє легко інтегрувати сторонні інструменти та плагіни для поліпшення розробки. Це дозволяє розробникам використовувати додаткові функціональність та розширювати можливості IDE.

1.2.3 Intellij IDEA

IntelliJ IDEA [9] — це інтегроване середовище розробки для програмістів, розроблене компанією JetBrains. IntelliJ IDEA надає широкий спектр функцій та інструментів для розробки програмного забезпечення на різних платформах, включаючи Java, Kotlin, Groovy, Scala, Android, JavaScript, TypeScript, HTML, CSS та багато інших.

Основні особливості IntelliJ IDEA включають:

- Розширений редактор коду: IntelliJ IDEA надає потужний та інтуїтивно зрозумілий редактор коду з функціями підсвічування синтаксису, автоматичного завершення коду, перехоплення помилок та багато інших корисних функцій.
- Аналіз коду: IDE включає різноманітні інструменти для аналізу коду, такі як автоматична перевірка синтаксису, виявлення помилок, використання неоптимальних конструкцій, підказки з документації та рекомендації щодо поліпшення коду.
- Підтримка мов програмування: IntelliJ IDEA підтримує багато мов програмування та фреймворків. Вона надає інструменти для розробки на Java, Kotlin, Groovy, Scala та інших мовах. Також IDE має інтеграцію з популярними фреймворками, такими як Spring, Hibernate, Android SDK та багато інших.
- Управління проектами: IntelliJ IDEA дозволяє зручно створювати та керувати проектами. Вона надає інструменти для імпорту проектів, підтримує системи контролю версій, такі як Git, SVN та Mercurial, та має інтегровану підтримку інструментів для автоматизованої збірки та тестування, таких як Maven, Gradle та інші.

1.2.4 Gradle

Gradle — це система збірки проєктів, що використовується в розробці програмного забезпечення, зокрема в середовищі розробки Android Studio. Вона

дозволяє ефективно керувати залежностями, компілювати та збирати програмний код, налаштовувати процес збирання і розгортання додатків.

Основні особливості Gradle:

- Декларативна синтаксична модель: Gradle використовує декларативну синтаксичну модель, що дозволяє описувати проєкт і його залежності в гнучких скриптах на мові Groovy або Kotlin. Це дає розробникам можливість визначати власні завдання, плагіни та правила збирання проєкт.
- Управління залежностями: Gradle має потужний механізм управління залежностями, що дозволяє автоматично завантажувати та інтегрувати зовнішні бібліотеки, фреймворки та інші компоненти в проєкт. Він підтримує різні сховища, такі як Maven Central або JCenter, і дозволяє легко встановлювати та оновлювати залежності.
- Розширюваність: Gradle має велику кількість плагінів, що дозволяють розширити його функціональність. Це дозволяє розробникам налаштувати збирання проєкт під свої потреби, додавати спеціалізовані завдання і функціональність.
- Швидкість і ефективність: Gradle має оптимізований механізм кешування та інкрементального збирання, що дозволяє збирати проєкт швидко та ефективно. Він підтримує паралельне виконання завдань, що дозволяє збільшити продуктивність і зменшити час збирання.
- Мультиплатформність: Gradle підтримує розробку для різних платформ, включаючи Android, Java, Kotlin, C++, JavaScript тощо. Це дозволяє розробникам використовувати Gradle для управління проєкт на різних платформах з використанням спільних засобів та налаштувань.

РОЗДІЛ 2. ФОРМУВАННЯ ВИМОГ ТА ОГЛЯД АЛГОРИТМІВ

2.1 Призначення та вимоги бібліотеки

На основі розглянутих наявних рішень було сформулювано вимоги до бібліотеки “filereader”.

2.1.1 Призначення бібліотеки “filereader”

Призначенням бібліотеки “filereader” є надати розробнику мобільних додатків під Андроїд можливості інтегрувати читач файлів електронних книг у додаток. Бібліотека має надавати базові можливості для налаштування відображення і перегляду файлів. Також бібліотека має змогу розширюватися у необхідних аспектах.

2.1.2 Функціональні вимоги до бібліотеки “filereader”

Згідно з проведеним аналізом наявних рішень було сформовано наступні функціональні вимоги:

- Тип файлів: система має працювати з різними типами файлів електронних книг (для початкової версії розробити для FB2 та EPUB, як найбільш популярних).
- Пошук: бібліотека має надавати інструменти для швидкого пошуку фрагментів у тексті в незалежності від типу файлів електронних книг.
- Навігація: бібліотека має надавати можливість переходити по сторінках за номером та використовувати наявний в книзі зміст для навігації.
- Шрифти: повинна бути можливість змінювати тип та розмір шрифтів.
- Відображення: бібліотека має надавати інтерфейс для взаємодії з книгою шляхом застосування стандартних компонент Android SDK.
- Мова інтерфейсу: розробник має мати можливість обирати мову для графічного інтерфейсу бібліотеки (заголовки меню тощо).

2.1.3 Нефункціональні вимоги

Згідно з проведеним аналізом наявних рішень було сформовано наступні нефункціональні вимоги:

- Масштабованість: бібліотека має мати змогу розширюватися у наступних аспектах: додавання нових парсерів для нових типів файлів електронних книг, додавання нових мов інтерфейсу, додавання нових типів шрифту, додавання нових пунктів меню.
- Швидкість: функції бібліотеки, в особистості пошук, мають працювати якомога швидше.
- Невеликий обсяг: бібліотека повинна бути якомога меншою за розміром і не використовувати зайвих інструментів там, де без них можна обійтися.

2.1.4 Технічні вимоги

Бібліотека має деякі обмеження щодо цільових додатків, у які він може бути інтегрований.

- Версія Android не менше 5.0 (що покриває близько 98% усіх існуючих пристроїв) (рисунок 2.1.4.1).
- Дозвіл на читання файлів (рисунок 2.1.4.2)
- Дозвіл на створення діалогових вікон (рисунок 2.1.4.2)

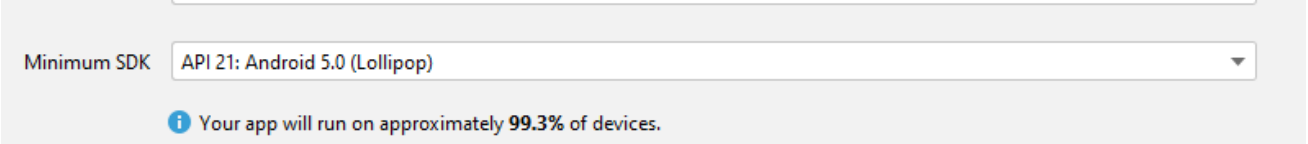

Рисунок 2.1.4.1 — вимоги до версії Android

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
</manifest>
```

Рисунок 2.1.4.2 — вимоги наявності дозволу

2.2 Огляд алгоритмів пошуку

Пошук фрагментів у тексті електронної книги є задачею з найбільшими витратами часових ресурсів (серед усіх поставлених задач), тому було прийняте рішення проаналізувати існуючі алгоритми та обрати найкращий варіант для реалізації.

Для створення ефективної системи пошуку оглянемо наявні алгоритми, які призначені для знаходження фрагмента у тексті [10].

2.2.1 Прямий пошук

Алгоритм прямого пошуку, також відомий як алгоритм "brute force" або "наївний алгоритм", є простим методом пошуку підрядка (фрагмента) в рядку. Він полягає в послідовній перевірці кожної позиції рядка на збіг зі шуканим підрядком [11].

Основний принцип алгоритму прямого пошуку полягає у встановленні початкової позиції входження підрядка на початку рядка і послідовному порівнянні кожного символу підрядка з відповідним символом рядка. Якщо всі символи збігаються, пошук вважається успішним, і повертається позиція входження підрядка. Якщо зустрічається невідповідність, пошук продовжується з наступної позиції рядка.

Основною перевагою алгоритму прямого пошуку є його простота реалізації. Він підходить для пошуку невеликих підрядків у коротких рядках. Однак, його часова складність становить O(n*m), де n — довжина рядка, а m — довжина підрядка. Це означає, що час виконання алгоритму зростає лінійно зі збільшенням розміру рядка і підрядка, що робить його неефективним для великих обсягів даних.

2.2.2 Алгоритм Кнута-Морріса-Пратта

Алгоритм КМП є ефективним алгоритмом пошуку підрядка в рядку [12, с.323]. Він дозволяє знайти всі входження підрядка, а не лише перше, і має часову складність O(n + m), де n — довжина рядка, а m — довжина підрядка. Цей алгоритм є покращенням алгоритму прямого пошуку та обсновується на використанні префіксної функції.

Основна ідея алгоритму КМП полягає в побудові префіксної функції, яка вказує на найбільшу можливу довжину правильного суфікса підрядка, який одночасно є префіксом. Ця інформація допомагає здійснювати більш ефективний зсув підрядка при невідповідності.

Алгоритм КМП складається з двох основних кроків: побудови префіксної функції та виконання пошуку. Під час побудови префіксної функції визначається для кожної позиції підрядка, яка є правильним суфіксом, його довжина. Потім, при виконанні пошуку, використовується ця інформація для здійснення ефективного зсуву підрядка при невідповідності символів.

Алгоритм КМП є ефективним у випадках, коли рядок і підрядок мають значну довжину або коли потрібно знайти всі входження підрядка в рядок. Він широко використовується в областях, де потрібно виконувати пошук із заданими обмеженнями часу, таких як обробка текстових даних, аналіз логів, компіляція програм і багато інших варіантів застосування.

2.2.3 Алгоритм Боєра-Мура

Алгоритм Боєра-Мура, часто відомий як БМ-пошук, є одним з найшвидших алгоритмів пошуку підрядка в рядку [13]. Вперше запропонований Робертом Боєром та Даніелем Муром в 1977 році, цей алгоритм знаходить застосування в багатьох сферах, таких як обробка тексту, пошук у базах даних, редактори коду та багато іншого.

Основна ідея БМ-пошуку полягає в тому, що пошук починається з кінця підрядка, а не з початку. Під час пошуку алгоритм здійснює зсуви на основі

певних правил, що дозволяють пропустити велику кількість символів, що не збігаються з шуканим підрядком.

Один з основних елементів БМ-пошуку — це таблиця зсувів, яка містить інформацію про кожен символ алфавіту і вказує, на скільки позицій можна зсунути підрядок, якщо відповідний символ не збігається зі знайденим символом у тексті.

Завдяки цій стратегії зсувів, алгоритм БМ-пошуку може бути дуже ефективним для пошуку великих підрядків в довгих рядках. Він має часову складність O(n/m) в найкращому випадку, де n — довжина тексту, m — довжина підрядка, і може здійснити значні економії в порівнянні з іншими алгоритмами.

2.2.4 Інвертований індекс

Пошук за допомогою інвертованого індексу є потужним методом пошуку, який використовується для швидкого та ефективного знаходження документів, що містять конкретні слова або терміни [14] [15] [16, с.559].

У традиційному індексі, деякі документи містять посилання на слова, які вони містять. З іншого боку, в інвертованому індексі слова або терміни зберігаються з посиланнями на документи, в яких вони з'являються. Це дозволяє швидко знаходити документи, що містять певні слова або терміни без необхідності проглядати всі документи.

Процес побудови інвертованого індексу включає такі кроки:

- Токенізація: текстові дані (документи) розбиваються на окремі слова або терміни.
- Створення термінів: унікальні терміни або слова зберігаються разом з посиланнями на документи, де вони зустрічаються.
- Індексування: інформація про терміни та їх посилання зберігається у структурі даних, яка називається інвертованим індексом.

Після побудови інвертованого індексу, пошук відбувається за наступними кроками:

- Токенізація запиту: запит розбивається на окремі слова або терміни.
- Пошук термінів: для кожного терміна в запиті, знаходяться документи, що містять цей термін за допомогою інвертованого індексу.
- Ранжування: документи, що містять запитові терміни, ранжуються за релевантністю для наданого запиту.

Переваги використання інвертованого індексу включають швидкий доступ до документів, ефективну обробку запитів і можливість розширення для великих обсягів текстових даних. Він широко використовується в системах пошуку, базах даних, пошукових двигунах та інших додатках, де важливим є швидке та ефективне знаходження відповідних документів.

2.2.5 Використання в роботі

Хоча БМ та КМП є досить ефективними алгоритмами, у роботі був використаний пошук по інвертованому індексу. Для збереження індексів використана структура HashMap, через те, що операції put() та get() працюють в середньому за константний час, тобто часова складність O(1) (у гіршому випадку O(log(n))) [17]. І хоча формування такої HashMap має часову складність O(n) (у гіршому випадку O(n*log(n)), проте швидкість пошуку по самім індексам O(1). Через це операції пошуку будуть працювати значно швидше ніж з використанням алгоритмів БМ та КМП.

Швидкість пошуку є ключовою характеристикою ефективності для даної задачі, через це був обраний саме такий варіант. І хоча витрати пам’яті для збереження структури HashMap у найгіршому випадку подвоїть витрати пам’яті на збереження інформації про книгу, це не є значним недоліком, через загалом незначний обсяг електронних книг.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ БІБЛІОТЕКИ

3.1 Опис структури бібліотеки “filereader”

Бібліотека “filereader” не використовує ніякі сторонні бібліотеки, крім вбудованих в середовище. Тому розробнику, що буде інтегрувати бібліотеку, не доведеться перейматися через залежності. Також перевагою є те, що вся взаємодія з користувачем додатка відбувається через стандартні View елементи. Цей факт дозволяє розробнику працювати з наданим йому функціоналом за допомогою звичних конструкції, що значно спрощує процес розробки.

3.1.1 Загальний опис структури

Для реалізації функціонала бібліотеки було розроблено 9 класів, опис яких наведено у таблиці 3.1.1.1.

Таблиця 3.1.1.1 — короткий опис класів.

Назва	**Опис**
Book	Структура для збереження та оперування інформацією з файлу. Надає методи для навігації, збереження, форматування.
EditableMenuDialog	Клас, що наслідує вбудований клас DialogFragment. Створений для надання зручної обробки деяких дії, як навігація, пошук тощо.
FileViewer	Основний клас, що надає інтерфейс користувача. За його допомогою відбувається відображення тексту, зображень, а також меню.
InvertedIndex	Клас що надає функціонал для пошуку тексту у файлі.
ReaderFb2	Реалізація абстрактного класу ReaderFormat. Виконує зчитування файлу типу FB2 та формує з нього структуру для подальшого опрацювання
ReaderEpub	Реалізація абстрактного класу ReaderFormat. Виконує зчитування файлу типу EPUB та формує з нього структуру для подальшого опрацювання

ReaderFormat	Абстрактний клас, створений для надання розробнику шаблону, за яким той може створити власний парсер файлу будь-якого типу.
Section	Клас, що являє собою представлення глави\секції файлу. Може зберігати як текст, так і зображення.
StyleSet	Клас, що зберігає усі налаштування стилю, та надає методи по їх впровадженню.

3.1.2 Клас Book

Клас Book (рисунок 3.1.2.1) являє собою структуру, яка зберігає інформацію з файлу у вигляді списку елементів класу Section, та має допоміжні поля для надання зручної навігації та зображення даних. Клас Book має два конструктори, що дозволяють створити книгу з файлу, що знаходиться по вказаному шляху. Різниця полягає у тому, що один з конструкторів дозволяє створити книгу за допомогою користувацької реалізації класу ReaderFormat. Інші методи дозволяють отримувати необхідну інформацію зі секції книги, а також здійснювати переходи по сторінках різними методами:

- Переходити з поточної сторінки на вказану кі-сть сторінок
- Переходити на сторінку по її абсолютному номеру.
- Переходити на сторінку відповідно заданому індексу слова.

Також клас Book має метод для оновлення розбиття секції на сторінки, в залежності від розміру цільової TextView. Розділення атрибутів класу на приватні та публічні відбувалася наступним чином: уся інформація, що формується автоматично і використовується для внутрішніх потреб класу — приватна. Уся інформація, що може бути використана зовні — публічна.

```
public class Book {


    private List<String> title = new ArrayList<>();
    private String bookAndAuthorLine = "";
    private String pathToFile = "";
    private int currentPage = 0;
    private int currentSection = 0;
    private ReaderFormat readerFormat;
    private int absolutePageNumber = 0;
    private int absoluteCurrentPage = 0;
    private String language = "";
    public String searchText = "";
    public int searchIndex = -1;
    public List<Section> sections = new ArrayList<>();

    public Book(String path, ReaderFormat readerFormat1){...}
    public Book(String path){...}
    public InvertedIndex createSearchIndex(){...}
    private static String getFileExtension(String filePath) {...}
    public List<String> getTitles(){...}
    public int getSectionType(int page){...}
    public int getCurrentPage() { return currentPage; }
    public String getLanguage() { return language; }
    public int getCurrentSection() { return currentSection; }
    public String getTitle(int section){...}
    public List<String> getTableOfContents(){...}
    public String moveToSection(int section){...}
    public String movePage(int direction){...}
    public String goToIndex(int section, int index, String searchText1){...}
    public String goToPage(int moveTo){...}
    public void updatePaging(int charsPerLine, int maxLines, int page, int section){...}
    public int getAbsoluteCurrentPage() { return absoluteCurrentPage; }
    public int getAbsolutePageNumber() { return absolutePageNumber; }
}
```

Рисунок 3.1.2.1 — структура класу Book

3.1.3 Клас EditableMenuDialog

Задача класу EditableMenuDialog (рисунок 3.1.3.1) — надавати можливість для створення діалогових вікон під різні задачі. Поточна реалізація має функціонал для створення діалогу навігації по номеру сторінки, зміни розміру шрифту, а також пошукового діалогу. Також клас має набір констант, який визначає різні типи діалогу. На цей момент клас покриває увесь потрібний функціонал, проте розробник може легко додати нові типи діалогів.

```
public class EditableMenuDialog extends DialogFragment {
    private Context appContext;
    private FileViewer fileViewer;
    private AlertDialog.Builder builderAdd;

    public String dialogResultString ="";
    public int dialogPurpose = 1;
    public String additionalInfo;
    public final int DIALOG_PURPOSE_GO_TO_PAGE=1;
    public final int DIALOG_PURPOSE_CHANGE_FONT_SIZE = 2;
    public final int DIALOG_PURPOSE_SEARCH = 3;
    public EditableMenuDialog(Context context, FileViewer fv, int purpose, String addInfo){...}
    public EditableMenuDialog(Context context, FileViewer fv, int purpose, String addInfo, AlertDialog.Builder build){...}
    @NonNull
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {...}
    private void createGoToPageDialog(AlertDialog.Builder builder){...}
    private void createFontSizeDialog(AlertDialog.Builder builder){...}
    private void createSearchDialog(AlertDialog.Builder builder){...}
    public boolean isNumeric(String str){...}

}
```

Рисунок 3.1.3.1 — структура класу EditableMenuDialog

3.1.4 Клас FileViewer

Клас FileViewe (рисунок 3.1.41) є основним інструментом розробника. Створення цього класу прив'язано до цільового елементу типу LinearLayout, у якому і буде відображено текст, та до якого буде прив'язано базовий функціонал бібліотеки. За рахунок цього розробник має повний контроль над зображенням і може змінювати у ньому все, що побажає як ззовні, так і редагуючи код модуля. Проте базовий функціонал також надає досить повний набір інструментів. Клас надає користувачу робочий варіант для перегляду, навігації тощо. Також у класі зберігаються екземпляри усіх інших класів, що надає можливість розробнику вносити зміни, працюючи лише зі створеним екземпляром класу FileViewer.

```
public class FileViewer{
    private String currentText;
    private int currentPage = 0;
    private boolean isClickAllowed = true;
    private int charsPerLine = 0;
    private int maxLines = 0;
    private int currentSection = 0;

    public int absolutePageNumber = 0;
    public int absoluteCurrentPage = 1;
    public InvertedIndex invertedIndex = new InvertedIndex();
    public TextView mainTextView;
    public LinearLayout mainLinearLayout;
    public Context appContext;
    public StyleSet styleSet;
    public String defaultLanguage = "eng";
    public Book book;

    public int getAbsoluteCurrentPage() { return absoluteCurrentPage; }
    public int getAbsolutePageNumber() { return absolutePageNumber; }
    public FileViewer(LinearLayout lr){...}
    public void setBook(String path, ReaderFormat rf){...}
    public void setBook(String path){...}
    public void updateFont(Context context, String name){...}
    @SuppressLint({"ClickableViewAccessibility","RestrictedApi"})
    private void setMainView() {...}
    public void addFontToFontTypes(SubMenu sb){...}
    public void addTitleToSubMenu(SubMenu sb){...}
    public void updateFontSize(int size){...}
    public void checkPaging(){...}
    public void displayText(String text){...}
    public String goToPage(int page){...}
    public String goToIndex(int section, int index, String text){...}
    public String movePage(int direction){...}
    public String goToSection(int section){...}


}
```

Рисунок 3.1.4.1 — структура класу FileViewer

3.1.5 Клас InvertedIndex

Клас InvertedIndex (рисунок 3.1.5.1) надає функціонал для додання та отримання позиції слова у тексті. Для зберігання побудованих інвертованих індексів було використано структуру HashMap, через те, що час пошуку значень у середньому фіксований. Індекси будуються один раз, при зчитуванні файлу, та зберігаються впродовж використання. Через це пошук працює швидко та ефективно див. 2.2.5. У випадку, коли треба знайти фрагмент з більше ніж одного слова — пошук працює найбільш ефективно. Тоді виконується операція get() структури HashMap і перевіряються збіги для підрядків довжини фрагмента для кожного вказаного посилання на місце в тексті. У випадку коли треба знайти

лише одне слово — перевіряються усі інвертовані індекси, щоб знайти усі слова, які містять фрагмент. Навіть у цьому випадку пошук працює швидко — часова складність O(m), де m — кі-сть ключів, що не більше ніж слів у тексті (рисунок 3.1.5.2).

```
public class InvertedIndex {
    public Map<String, String> invIndex = new HashMap<>();

    public void addText(String Text, int section){...}
    public String getIndByName(String name){...}

}
```

Рисунок 3.1.5.1 — структура класу InvertedIndex

```
public String getIndByName(String name, boolean isSingleWord){
    if(name.equals("")){
        return "";
    }
    if(invIndex.containsKey(name)&&!isSingleWord){
        return invIndex.get(name);
    }
    String res = "";
    List<String> keys = new ArrayList<>(invIndex.keySet());
    for(String k : keys){
        if(k.contains(name)){
            res+=invIndex.get(k);
        }
    }
    return res;

}
```

Рисунок 3.1.5.2 — реалізація пошуку слів

3.1.6 Клас ReaderFb2

Клас ReaderFb2 (рисунок 3.1.6.1) є реалізацією класу ReaderFormat і призначений для зчитування файлів типу FB2. У процесі зчитування інформація розбивається на секції наступним чином: кожен тег “<section>” містить у собі окрему секцію. Окремо виділяються назви секцій, та зберігаються. Зображення

зберігаються в окремі секції без назви. Секції без назви не беруть участь у формуванні змісту.

```
public class ReaderFb2 extends ReaderFormat {
    private List<String> images = new ArrayList<>();
    private List<Section> sections = new ArrayList<>();
    private List<String> titles = new ArrayList<>();
    private String authorLine = "";
    private String language = "eng";
    private int img_id = 0;
    private Section s= new Section( type0: 0, text0: "", title0: "");
    @Override
    public List<String> getTitles(){...}
    @Override
    public  List<Section> getSections() { return sections; }
    @Override
    public String getAuthorLine() { return authorLine; }
    @Override
    public String getLanguage() { return language; }
    @Override
    public void parseFile( File file){...}
    public void getChildElements(NodeList list,int tit){...}
}
```

Рисунок 3.1.6.1 — структура класу ReaderFb2

Варто зазначити, що клас ReaderFormat має усі ті ж самі атрибути та методи, окрім getChildElements(), тому, що є абстрактним. Аналогічно виглядає і файл ReaderEpub, хіба що наявно більше методів для коректного опрацювання формату файлу (рисунок 3.1.6.2).

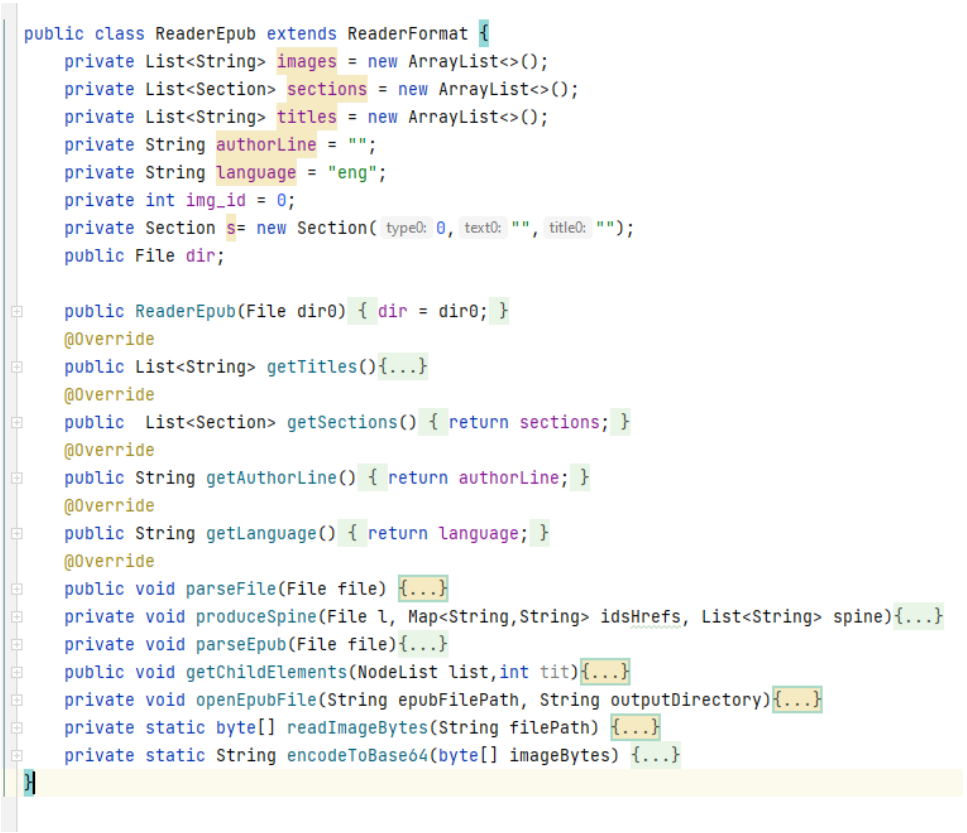

Рисунок 3.1.6.2 — структура класу ReaderEpub

3.1.7 Клас Section

Клас Section (рисунок 3.1.7.1) являє собою структуру, що представляє секцію книги. Окрім методів, що видають потрібну інформацію та конструктора, наявний метод для розбиття тексту секції на сторінки в залежності від переданих параметрів TextView.

```
public class Section {
    private final int type;
    private String text;
    private String title;
    private int charNumber;
    private int charsPerLine = 0;
    private int maxLines = 0;

    public int SECTION_TYPE_TEXT = 1;
    public int SECTION_TYPE_IMAGE = 2;
    public List<String> pages = new ArrayList<>();

    public Section( int type0, String text0, String title0){...}
    public String getTitle() { return title; }
    public void appendSection(String text0){...}
    public String getText() { return text; }
    public int getType() { return type; }
    public int getCharsNumber() { return charNumber; }
    public int getPagesNumber() { return pages.size(); }
    public String getImageString() { return text; }
    public void updatePaging(int currentCharsPerRow, int currentMaxLines){...}
    public String getPage(int i){...}


}
```

Рисунок 3.1.7.1 — структура класу Section

3.1.8 Клас StyleSet

Клас StyleSet (рисунок 3.1.8.1) зберігає у собі налаштування цільової TextView, у якій відображено дані з файлу. Також зберігаються структури Map, у яких зберігаються набори налаштувань, які можуть бути змінені користувачем та розробником. Також наявний набір за замовчуванням — усі назви меню, що бачить користувач, зберігаються у відповідному атрибуті та можуть бути дуже легко змінені. При бажанні розробник може додавати нові стилі шрифту, мови, кольори, або ж додати щось своє.

```
public class StyleSet {
    private Typeface tf;
    private double elipsCoef = 1.2;

    public int TITLE_SELECTED_COLOR = Color.GREEN;
    public int TITLE_NOT_SELECTED_COLOR = Color.WHITE;
    public Map<String, Map<String, String>> languageSettings = new HashMap<>();
    public Map<String,String> currentLanguage = new HashMap<>();
    public String defaultLang = "eng";
    public Map<String, Typeface> typefaceMap = new HashMap<>();
    public final int DIALOG_PURPOSE_GO_TO_PAGE=1;
    public final int DIALOG_PURPOSE_CHANGE_FONT_SIZE = 2;
    public final int DIALOG_PURPOSE_SEARCH = 3;
    public double currentFontSize = 0;
    public int currentTextColor;
    public Map<String, Integer> textColorsMap = new HashMap<>();

    public StyleSet() { setDefaultLanguageSettings(); }
    public void setLanguage(String language){...}
    public void updateEllipsize(double upd) { elipsCoef += upd; }
    public int getMaxLines(TextView textView) {...}
    public int getMaxCharactersPerLine(TextView textView) {...}
    public void doAfterTextViewCreation(TextView textView) {...}
    public void addFont(Typeface tf, String publicName){...}
    public void updateFontByName(String name, TextView tv){...}
    public void removeFontByName(String name, TextView tv){...}
    public void addTextColor(int val, String publicName){...}
    public void updateTextColorByName(String name, TextView tv){...}
    public void removeTextColorByName(String name, TextView tv){...}
    public void updateFont(Context context, String name, TextView tv){...}
    public void updateFontSize(TextView tv, int size){...}
    public void setDefaultLanguageSettings(){...}
}
```

Рисунок 3.1.8.1 — структура класу StyleSet

3.2 Візуальне відображення

Бібліотека “filereader” може бути використана у двох варіантах:

- Використовувати лише клас Book для роботи з даними електронної книги.
- Використати клас FileViewer, який надає можливість прикріпити реалізований за замовчуванням функціонал до користувацького LinearLayout.

FileViewer створює усі потрібні графічні елементи (такі як TextView для відображення тексту/зображень або меню та його пунктів) для надання базового функціонала. Також налаштовуються усі необхідні слухачі для натисків на елементи. Головна TextView у якій відображується текст розділена на три

частини (ліві 30% - перегорнути сторінку вліво, центральні 40% - відкрити меню, праві 30% - перегорнути сторінку вправо) котрі оброблюють натискання на них. Таке розділення було обране як найпростіше для реалізації та інтуїтивно зрозуміле. При бажанні розробник може змінювати ці параметри або замінити їх на власні. Меню було сформоване з чотирьох головних пунктів: Зміст, Навігація, Шрифти, Пошук. Ці пункти надають доступ до усіх реалізованих на цей час функцій бібліотеки. Звісно, розробник має доступ до сформованого меню та може з легкістю змінювати порядок пунктів, редагувати їх, додавати нові, або ж видаляти наявні.

3.3 Інструкція користувача

3.3.1 Підключення бібліотеки

Для того, щоб під'єднати бібліотеку “filereader” у проєкт достатньо лише розмістити папку з бібліотекою у кореневій директорії проєкту (рисунок 3.3.1.1).

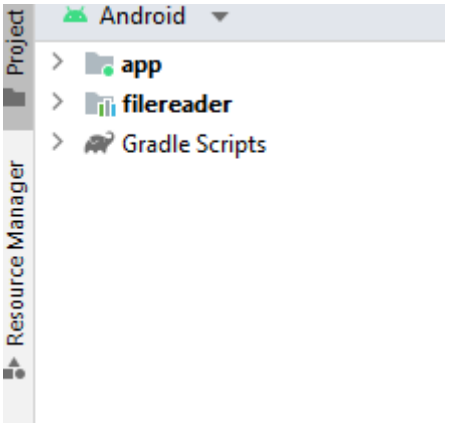

Рисунок 3.3.1.1 — розміщення бібліотеки

Наступний крок — додати залежність у файл “build.gradle” додатку (рисунок 3.3.1.2).

```
dependencies {

    implementation project(path: ':filereader')
}
```

Рисунок 3.3.1.2 — додання залежності

3.3.2 Використання

Для того, щоб почати використовувати функціонал бібліотеки розробнику потрібно створити LinearLayout у якому буде відображено текст, та передати його у конструктор класу FileViewer. Для отримання доступу до функціонала за замовчуванням, розробник має передати шлях до файлу у відповідний метод (рисунок 3.3.2.1).

```
FileViewer fv = new FileViewer(findViewById(R.id.reader));
fv.setBook("/storage/emulated/0/Documents/jerome_three_men.fb2");
```

Рисунок 3.3.2.1 — базова ініціалізація

У разі, якщо усі кроки виконані правильно, буде отримано наступний результат (рисунок 3.3.2.2). На екран буде виведена перша сторінка файлу.

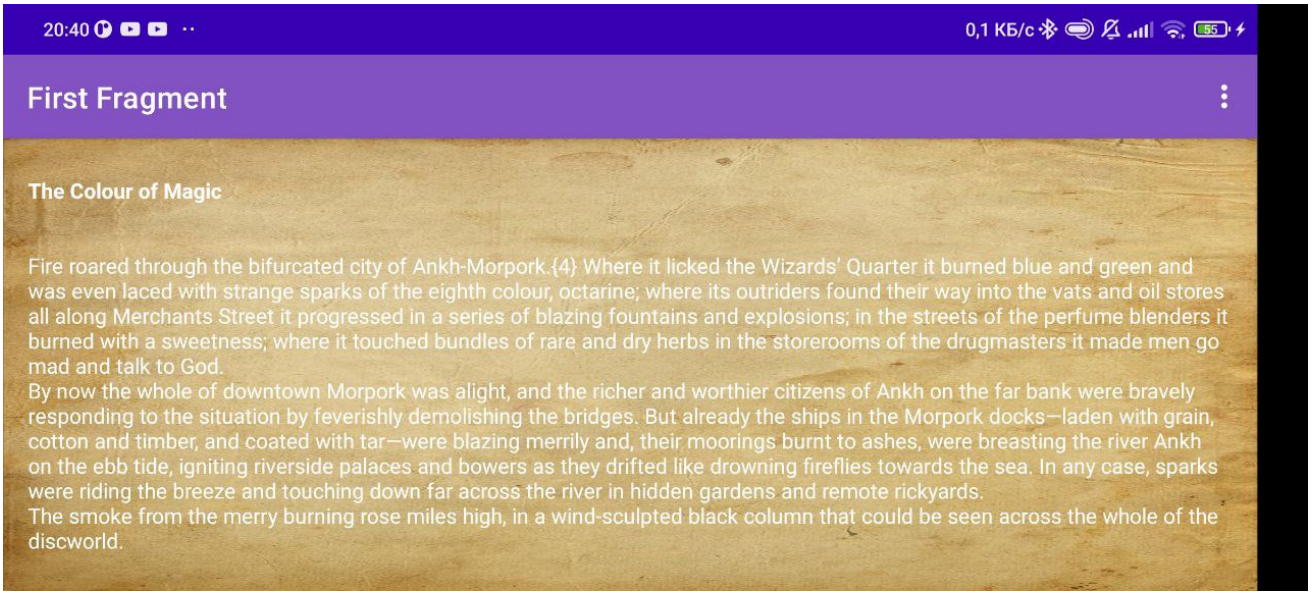

Рисунок 3.3.2.2 — базовий вид

Базовий функціонал надає прослуховувачі подій, тому за замовчуванням натиснення на різні зони екрана будуть виконувати дії за замовчуванням (рисунок 3.3.2.3):

- Ліві 30% екрана — перегорнути сторінку вліво
- Праві 30% екрана — перегорнути сторінку вправо
- Центральна частина — відкрити меню застосунку.

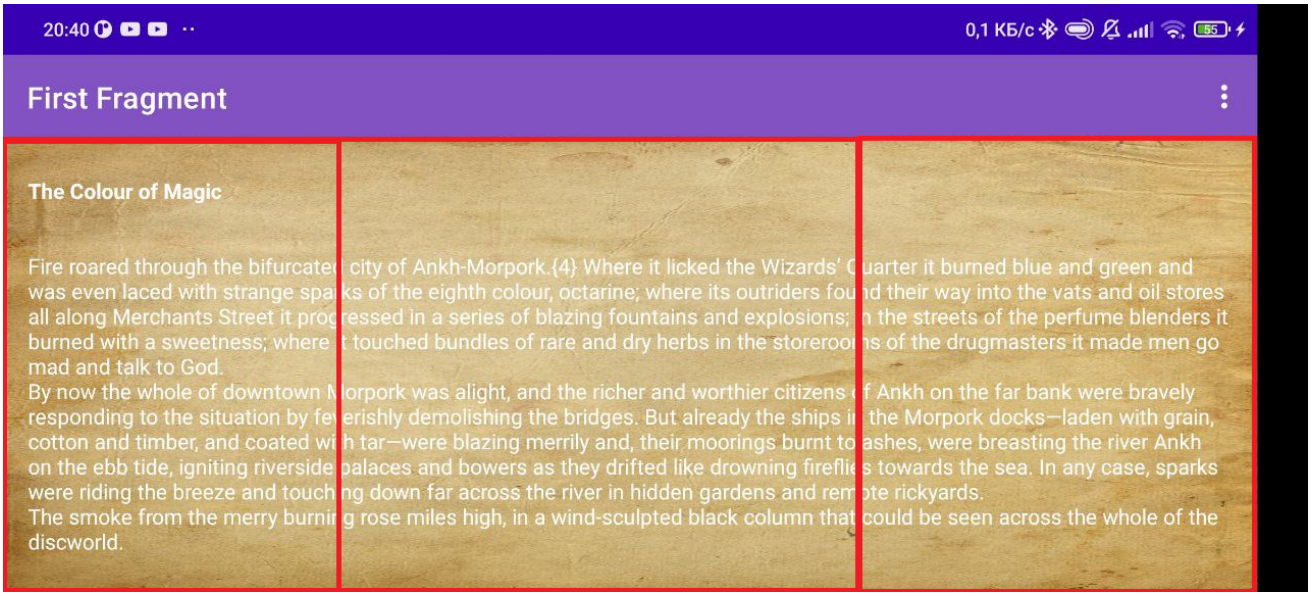

Рисунок 3.3.2.3 — базові зони екрана.

Зображення, наявні у файлі відображаються, займаючи повну сторінку (рисунок 3.3.2.4). Зображення автоматично підлаштовується під розмір екрана.

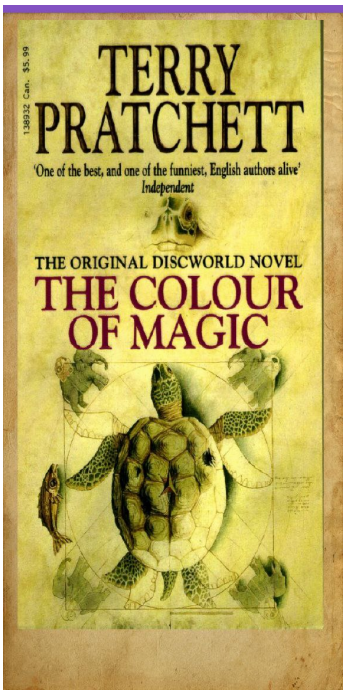

Рисунок 3.3.2.4 — відображення зображень.

При натисненні на центр екрана відкривається меню. В базовому варіанті меню має чотири пункти: Зміст, Навігація, Шрифти, Пошук (рисунок 3.3.2.5). Приклад зміни мови застосунку(рисунок 3.3.2.6).

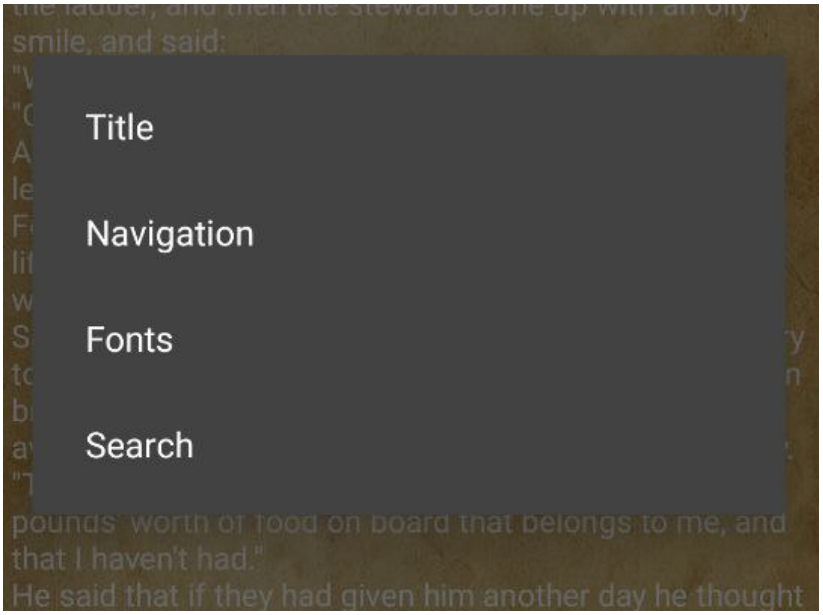

Рисунок 3.3.2.5 — меню

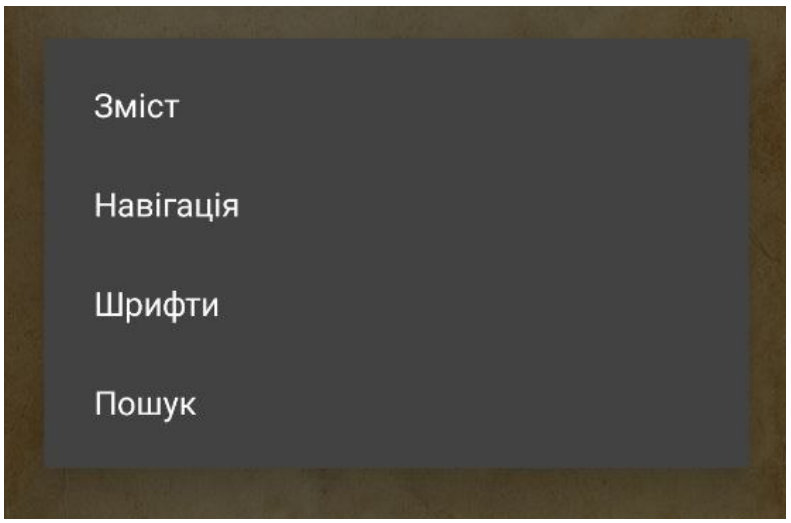

Рисунок 3.3.2.6 — меню іншою мовою

Зміст відкриває підменю з переліком усіх назв секцій, натиснувши на які можна потрапити на першу сторінку обраної секції. Варто зазначити, що поточна секція (якщо є) помічається кольором (рисунок 3.3.2.7).

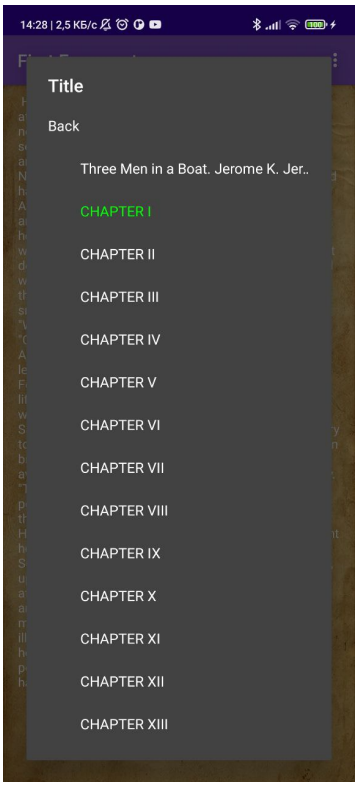

Рисунок 3.3.2.7 — Зміст

Пункт Навігація дає можливість перейти на сторінку за номером (рисунок 3.3.2.8). При введенні некоректного номера перехід не відбудеться. Якщо номер

більший або менший за граничні значення — відбудеться на останню та першу сторінки відповідно.

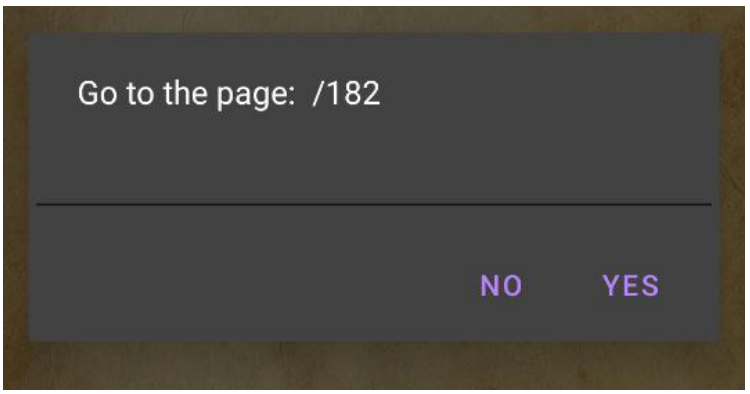

Рисунок 3.3.2.8 — перехід на сторінку за номером

Пункт Шрифти відкриває підменю, у якому можна змінити тип шрифтів (рисунок 3.3.2.9). Або змінити розмір шрифту (рисунок 3.3.2.10).

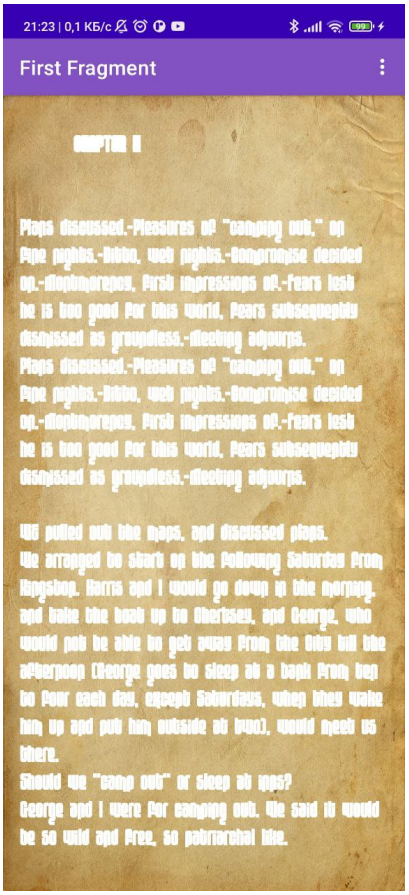

Рисунок 3.3.2.9 — інший шрифт

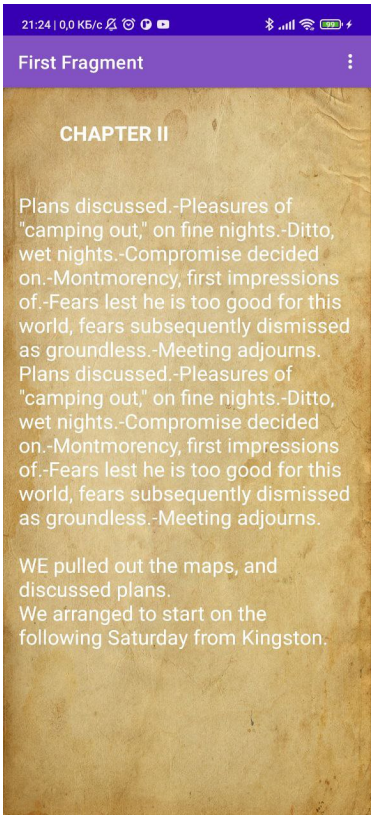

Рисунок 3.3.2.10 — розмір шрифту

Пункт Пошук дає можливість знайти слово або фразу у тексті (рисунок 3.3.2.11), та перейти в будь-яке місце, де був знайдений фрагмент (рисунок 3.3.2.12) (рисунок 3.3.2.13).

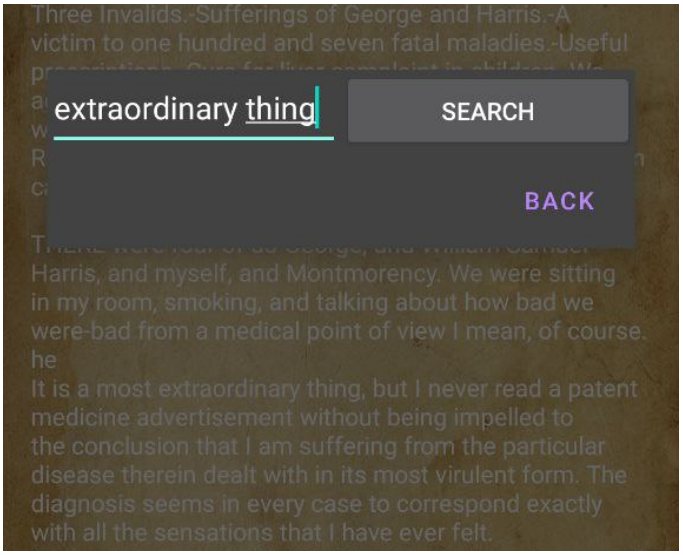

Рисунок 3.3.2.11 — пошук слів

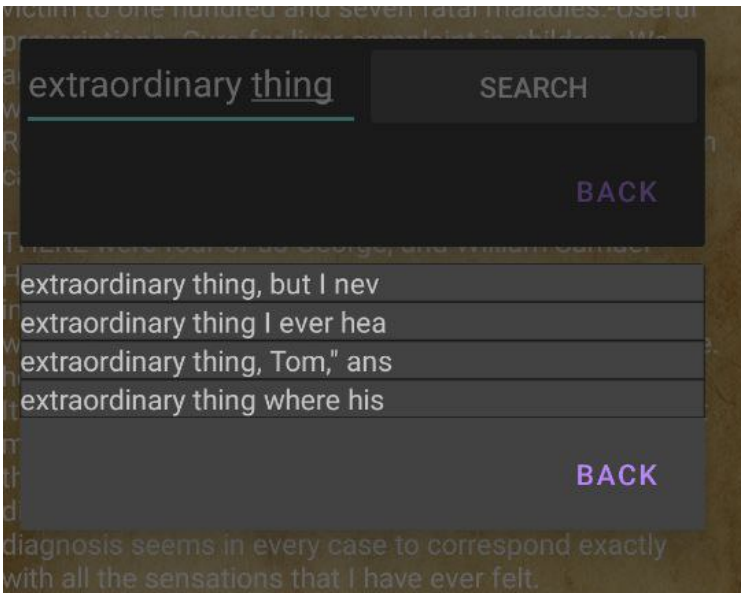

Рисунок 3.3.2.12 — результати пошуку

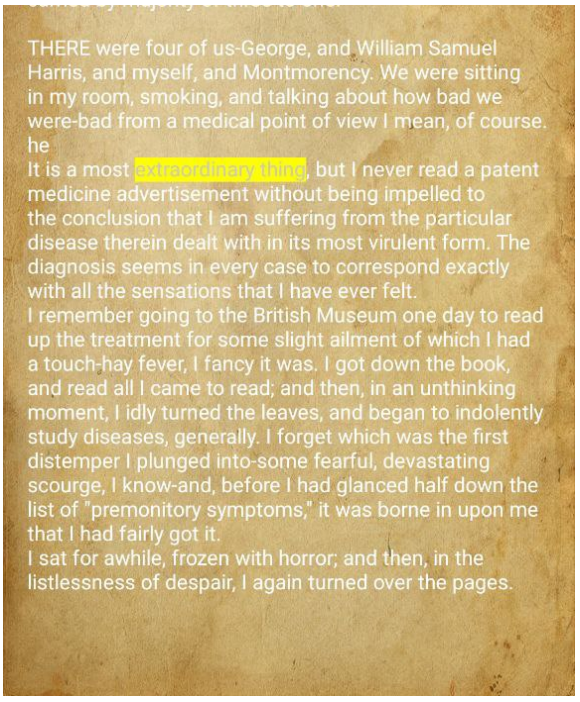

Рисунок 3.3.2.13 — перехід до результату пошуку

ВИСНОВКИ

У результаті роботи було розглянуто існуючі бібліотеки. На основі огляду наявних рішень було сформовано вимоги до бібліотеки. Найбільш ресурсоємною задачею було визначено пошук фрагментів у тексті, через це було розглянуто алгоритми пошуку, та обрано оптимальний варіант, який і було імплементовано.

В результаті була розроблена бібліотека “filereader” яка реалізує наступні функції:

- Читання файлів електронних книг форматів FB2 та EPUB.
- Пошук у файлі будь-якого типу (система пошуку не прив’язана до формату файлу, який був зчитаний).
- Навігація у книзі: перехід на сторінку за номером та перехід по змісту книги.
- Зміна величини та типу шрифту, збереження різних типів у вигляді списку.
- Підтримка різних мов для інтерфейсу, який представляє бібліотека.
- Графічне відображення тексту/зображень на екрані, а також обробка дій користувача.

У майбутньому бібліотеку можна розширювати у наступних напрямках:

- Додати нові парсери файлів електронних книг (наприклад PDF) за замовчуванням.
- Додати можливість зберігати теми: колір шрифту, задній фон тощо. Розробити структуру для збереження теми.
- Додати нові мови, на яких може бути відображений інтерфейс наданий бібліотекою.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Global mobile OS market share 2022 | Statista [Електронний ресурс] // Statista. – Режим доступу: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/.
2. Llc R. Додатки в Google Play – ReadEra – читалка книг fb2 pdf [Електронний ресурс] / Readera Llc // Android Apps on Google Play. – Режим доступу: https://play.google.com/store/apps/details?id=org.readera&hl=ru&gl=US.
3. GitHub — readium/mobile: 📱 readium mobile is a toolkit for ebooks, audiobooks and comics written in swift & kotlin. [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/readium/mobile
4. GitHub — psiegman/epublib: a java library for reading and writing epub files [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/psiegman/epublib
5. GitHub — folioreader/folioreader-android: a java epub reader and parser framework for android. [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/FolioReader/FolioReader-Android.
6. FBReader SDK [Електронний ресурс] // FBReader SDK. – Режим доступу: https://sdk.fbreader.org.
7. Getting started with java — dev.java [Електронний ресурс] // Dev.java: The Destination for Java Developers. – Режим доступу: https://dev.java/learn/getting-started/.
8. Meet android studio | android developers [Електронний ресурс] // Android Developers. – Режим доступу: https://developer.android.com/studio/intro.
9. IntelliJ IDEA – the leading java and kotlin IDE [Електронний ресурс] // JetBrains. – Режим доступу: https://www.jetbrains.com/idea/.

10. Brute force algorithm [Електронний ресурс] // IGM. – Режим доступу: http://www-igm.univ-mlv.fr/~lecroq/string/node3.html.
11. Юрий. Алгоритмы поиска в строке [Електронний ресурс] / Юрий // Хабр. – Режим доступу: https://habr.com/ru/articles/111449/.
12. Knuth D. E. Fast Pattern Matching in Strings [Електронний ресурс] / Donald E. Knuth, James H. Morris, Jr., Vaughan R. Pratt // SIAM Journal on Computing. – 1977. – Т. 6, № 2. – С. 323–350. – Режим доступу: https://doi.org/10.1137/0206024
13. Boyer R. S. A fast string searching algorithm [Електронний ресурс] / Robert S. Boyer, J. Strother Moore // Communications of the ACM. – 1977. – Т. 20, № 10. – С. 762–772. – Режим доступу: https://doi.org/10.1145/359842.359859
14. Salton G. Extended Boolean information retrieval [Електронний ресурс] / Gerard Salton, Edward A. Fox, Harry Wu // Communications of the ACM. – 1983. – Т. 26, № 11. – С. 1022–1036. – Режим доступу: https://doi.org/10.1145/182.358466
15. Zobel J. Inverted files versus signature files for text indexing [Електронний ресурс] / Justin Zobel, Alistair Moffat, Kotagiri Ramamohanarao // ACM transactions on database systems. – 1998. – Т. 23, № 4. – С. 453–490. – Режим доступу: https://doi.org/10.1145/296854.277632
16. Knuth D. E. Art of computer programming, volume 3: sorting and searching (2nd edition) / Donald Ervin Knuth. – [Б. м.] : Addison-Wesley Professional, 1998. – 800 с.
17. Internal Working of HashMap in Java — GeeksforGeeks [Електронний ресурс] // GeeksforGeeks. – Режим доступу: https://www.geeksforgeeks.org/internal-working-of-hashmap-java/.