**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

# Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: **«РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ E READER ДЛЯ ЧИТАННЯ ЕЛЕКТРОННИХ КНИГ ЗА ДОПОМОГОЮ МОВИ ПРОГРАМУВАННЯ JAVA»**

Виконав: студент 4 курсу, групи ПД-41
спеціальності
121 Інженерії програмного забезпечення
(шифр і назва спеціальності)

Волчанський О.С.
(прізвище та ініціали)
Керівник Дібрівний О.А.
(прізвище та ініціали)
Рецензент_____
(прізвище та ініціали)
Нормоконтроль_____
(прізвище та ініціали)

Київ – 2021

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

**Навчально-науковий інститут Телекомунікацій**

Кафедра _Інженерії програмного забезпечення_____
Ступінь вищої освіти - «Бакалавр»_____
Напрям підготовки -121 «Інженерія програмного забезпечення»_____

**ЗАТВЕРДЖУЮ**
Завідувач кафедри
Інженерії програмного забезпечення
_____Негоденко О.В
“____” _____2021 року

**ЗАВДАННЯ**
**НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ**

**Волчанському Олексію Сергійовичу**
(прізвище, ім’я, по батькові)

1.Тема роботи: «Розробка мобільного додатку E READER для читання електронних книг за допомогою мови програмування Java»_____

Керівник роботи__Дібрівний О.А., Старший викладач кафедри ІПЗ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від “12”березня 2021 року №65.

2. Строк подання студентом роботи_____«01» червня 2021 року

3. Вхідні дані до роботи:
3.1.Електронні книги;
3.2.Android;
3.3.Android Studio;
3.4.Науково-технічна література.

4 4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
4.1. Загальна характеристика додатків для читання електронних книг для мобільних пристроїв.
4.2. Базові вимоги до розробки додатків на Android.
4.3. Структура додатку та його робота.

5. Перелік графічного матеріалу
5.1.Мета, Об’єкт та предмет дослідження_____
5.2.Аналоги_____
5.3.Порівняння_____
5.4.Технічні завдання_____
5.5.Програмні засоби реалізаціЇ_____

5.6. Бібліотека MuPDF
5.7. Модулі програми
5.8. Клас BookInfo
5.9. Висновки
6. Дата видачі завдання 19.04.2021

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів бакалаврської роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Підбір науково-технічної літератури | 30.02 — 20.03 | Виконано |
| 2 | Загальна характеристика додатків для читання електронних книг для мобільних пристроїв | 22.03 — 30.03 | Виконано |
| 3 | Базові вимоги до розробки додатків на Android | 02.04 — 05.04 | Виконано |
| 4 | Розробка додатку BookReader | 08.04 — 19.04 | Виконано |
| 5 | Структура додатку та його робота | 20.04 — 30.04 | Виконано |
| 6 | Вступ, висновки, реферат | 02.05 — 10.05 | Виконано |
| 7 | Попередній захист роботи | 11.05 | Виконано |
| 8 | Подання роботи в деканат | 01.06 | Виконано |

Студент _____ Волчанський О.С.
( підпис ) (прізвище та ініціали)

Керівник роботи _____ Дібрівний О.А
( підпис ) (прізвище та ініціали)

# РЕФЕРАТ

Текстова частина бакалаврської роботи: 40 с., 14 рис., 3 дод., 26 джерел.

Об`єкт дослідження – додатки для читання електронних книг.

Предмет дослідження – найбільш важливі функції додатків для читання електронних книг.

Мета роботи – розробити ефективний додаток для читання електронних книг для операційної системи Android із простим та зручним інтерфейсом.

Визначено найбільш важливі функції додатків для читання електронних книг.

Здійснено розробку додатку для читання електронних книг з використанням мови програмування Java та середовища розробки Android Studio.

На основі результатів виконаних досліджень розроблено простий та зручний інтерфейс додатку для читання електронних книг.

Упровадження розробленого додатку дозволяє заповнити нішу між додатками з дуже простим і малофункційним інтерфейсом та додатками із багатофункціональним та складним інтерфейсом.

Ключові слова: ЕЛЕКТРОННІ КНИГИ, ANDROID, JAVA, ANDROID STUDIO, ART, DALVIK, KOTLIN, MUPDF.

**ЗМІСТ**

РЕФЕРАТ ..... 6
ЗМІСТ ..... 7
ВСТУП ..... 8
1. ЗАГАЛЬНА ХАРАКТЕРИСТИКА ДОДАТКІВ ДЛЯ ЧИТАННЯ ЕЛЕКТРОННИХ КНИГ ДЛЯ МОБІЛЬНИХ ПРИСТРОЇВ ..... 10
1.1 Огляд основних функцій та можливостей додатків для читання електронних книг ..... 10
1.2 Інтерфейс і функціонал додатків для читання електронних книг на Android ..... 13
1.3 Інтерфейс і функціонал додатків для читання електронних книг на IOS . 21
1.4 Вибір цільової ОС та вимоги до додатка ..... 23
2. БАЗОВІ ВИМОГИ ДО РОЗРОБКИ ДОДАТКІВ НА ANDROID ..... 25
2.1 Особливості ОС Android ..... 25
2.2 Особливості мови програмування Java ..... 30
2.3 Життєвий цикл та структура додатка на Android ..... 31
3. СТРУКТУРА ДОДАТКУ ТА ЙОГО РОБОТА ..... 41
3.1 Бібліотека MuPDF ..... 41
3.2 Інтерфейс користувача ..... 42
3.3 Програмна структура додатку ..... 44
3.4 Результати тестування додатку ..... 48
ВИСНОВКИ ..... 50
ПЕРЕЛІК ПОСИЛАНЬ ..... 52
ДОДАТОК А ..... 55
ДОДАТОК Б ..... 59
ДОДАТОК В ..... 60
ДОДАТОК Г ..... 61

# ВСТУП

Текстові документи протягом майже всього розвитку західної цивілізації були одним з найважливіших переносників інформації між людьми. Спершу це були сувої з написаним на них друкованим текстом, потім це друковані книги, а зараз все більшої популярності набирають електронні документи в тому числі електронні книги [1]. У сучасному світі дуже важливо мати доступ до інформації написаної саме книгах, оскільки вона хоч трохи є перевіреною та рецензованою на відміну від великої кількості інтернет-статей написаних не завжди кваліфікованими людьми. Зрештою, читання художньої літератури також є корисним для сучасної людини у якої все більше переважає так званий кліповий тип мислення [2]. Художня книга стимулює розвиток уяви та допомагає навчитися концентрації. Оскільки для сучасної людини смартфон є одним із джерел контенту який розвиває кліпове мислення дуже важливо доповнити арсенал додатків для нього програмою яка дозволить людині відволіктися від цього згубного контенту і почитати книгу. Особливо це важливо для школярів.

Метою випускної кваліфікаційної роботи є розробка мобільного Android-додатку для читання електронних книг «BookReader». Додаток повинен мати простий та водночас інтуїтивно зрозумілий інтерфейс який повинен володіти функціями для зручної роботи з книгами.

В процесі роботи вирішувалися наступні основні завдання:

- пошук і аналіз аналогічних додатків, які вирішують поставлену задачу та виявлення найбільш корисних характеристик їх функціоналу;
- аналіз можливостей операційної системи Android з точки зору розробки додатків;
- розробка програми для читання електронних книг «BookReader».
- У процесі розробки додатку використовувалося інтегроване середовище розробки “Android Studio”. Також було використано бібліотеку “MuPDF”.

- Науковою новизною даної роботи є комплексне порівняння додатків для перегляду електронних книг на операційних системах Android та IOS, а також розробка додатку із раніше не запропонованим функціоналом на ринку подібних додатків. Таким функціоналом є можливість перегляду змісту книги та закладок окремо від тексту книги.
- Практична значущість результатів полягає у розробці додатку для перегляду файлів електронних книг на операційній системі Android. Розроблений додаток має високу швидкодію, малі розміри, а також простий та зрозумілий інтерфейс користувача.

# 1. ЗАГАЛЬНА ХАРАКТЕРИСТИКА ДОДАТКІВ ДЛЯ ЧИТАННЯ ЕЛЕКТРОННИХ КНИГ ДЛЯ МОБІЛЬНИХ ПРИСТРОЇВ

## 1.1 Огляд основних функцій та можливостей додатків для читання електронних книг

Смартфони та планшети вже давно зарекомендували себе як універсальні помічники для людини. Кожен такий пристрій обладнаний функціоналом, який ще зовсім недавно був абсолютно фантастичним для девайса такого зовсім невеликого розміру. Такі пристрої можуть з успіхом виконувати не лише роль голосового засобу зв'язку, а й відео- та фотокамери, записника, калькулятора, відеотелефону, засобу доступу до мережі "Інтернет", органайзера, книги та багатьох інших корисних у щоденному використанні пристроїв [3].

Багато користувачів використовують свої планшети та смартфони для читання електронних книг. Це дуже зручно, адже в такому випадку людина у свої власній кишені має дуже велику бібліотеку книг доступну у будь-який момент. Багато додатків, які реалізовують такий функціонал, мають можливість не лише читати книги доступні на внутрішньому носії пристрою, але й завантажувати книги з інтернету. При чому можна завантажити як безкоштовну книгу, так і купити якесь платне видання. В такому випадку компанія-розробник додатку формує базу даних безкоштовних та платних книг, або реалізовує можливість підключення до стандартних бібліотек присутніх в інтернеті.

Також існують додатки які дозволяють "закласти" у книзі потрібну сторінку, або зробити певні замітки на сторінці, або й виділити певним кольором слово, вислів, або речення яке користувачу здалося важливим або цікавим. Непоганою можливістю певних додатків є їх здатність змінити оформлення книги, тобто змінити тип шрифту, його розмір та стиль. Так само досить та потрібною цікавою функцією є пошук певного введеного користувачем слова у тексті книги. Така функція може допомогти при роботі із науковою літературою та словниками. Базовим функціоналом практично кожного додатку є можливість зберегти прогрес

читання, тобто після повторного відкриття додатку немає потреби шукати останню прочитану сторінку — програма автоматично її відкриє. Також багато додатків формують історію перегляду книг що дозволяє одночасно читати декілька книг та не запам'ятовувати сторінку на якій зупинився процес читання. Деякі додатки рахують загальний час потрачений на читання книги, а також середній час читання однієї сторінки. Така функція дозволяє визначити найбільш продуктивний для кожного користувача час доби для прочитання книг.

Практично кожен додаток має продуманий, красивий та зручний інтерфейс користувача. Наприклад, такий інтерфейс формує не просто список книг, які були нещодавно відкриті, а додає до кожного елемента списку зображення обкладинки, або першої сторінки книги. Це дозволяє швидше орієнтуватися у цьому списку, а отже підвищує зручність додатку. Розробник старається зробити інтерфейс додатку максимально “user-friendly” або “usability”, тобто максимально ергономічним та зручним для користувача. Високо цінується інтуїтивно зрозумілий для користувача інтерфейс. Проте не кожен додаток може похвалитися саме таким інтерфейсом користувача, хоча більшість додатків не мають великих проблем з ергономічністю [4].

Екран смартфона чи планшета ніколи не буде настільки ж зручним для очей як екран електронного рідера у вигляді окремого пристрою виготовленого на основі технології електронних чорнил. Тому читання книг з екрана планшета чи смартфона приводить до швидкої втоми очей та може негативно позначитися на здоров’ї органів зору. Особливо помітним це стає при читанні книг в умовах слабкого освітлення, або у повній темноті. Щоб хоч якось компенсувати такий доволі суттєвий недолік програм для читання електронних книг деякі розробники додали режим показ тексту книги при якому фон стає чорним, а шрифт білим. Це так званий “Dark Mode” — темний режим, або “Night Mode” — нічний режим. Такий спосіб зображення тексту на екрані дозволяє зменшити навантаження на зір, а отже збільшити час читання книги без необхідності робити перерву для відпочинку очей [5].

Доволі суттєвою властивістю кожного додатку є кількість підтримуваних форматів файлів книг. Наприклад, є додатки які підтримують лише формат fb2, або ebook та pdf. Під час вибору такого додатку потрібно враховувати, що формат ebook хоч і є сучаснішим і вже багато книг є саме у цьому форматі, проте старіший fb2 все ще не втрачає популярності, тому під час користування додатком прийдеться користуватися конвертерами форматів для деяких книг. Такі конвертери є доступні в мережі інтернет в режимі онлайн, проте користування ними буде займати додатковий час. Проте більшість додатків підтримують дуже широкий формат файлів електронних книг при цьому мають зручний інтерфейс, тому користувачам слід звертати увагу саме на такі програми.

Багато користувачів мають багато пристроїв на яких вони читають книги, тому їм потрібно мати одні й ті ж книги на всіх пристроях та можливість синхронізувати прогрес читання. Велика кількість додатків дає можливість синхронізувати файли та прогрес їх читання на багатьох пристроях на яких встановлено цей додаток.

Зважаючи на сьогоднішню велику популярність аудіокниг багато розробників розширюють свої додатки таким додатковим функціоналом як можливість прослуховування аудіофайлів, або навіть можливістю озвучення тексту. Функція озвучення тексту дозволяє перетворити будь-яку книгу в аудіокнигу та прослуховувати її під час виконання інших занять.

Під час роботи з іноземною літературою інколи необхідно користуватися словником, що є доволі незручним під час читання книги з екрана смартфона. У зв’язку з цим багато розробників вбудовують у свої додатки ще й функцію перекладача та словника. Також ця функція реалізована у багатьох електронних рідерах виконаних у вигляді окремого пристрою.

Деякі додатки мають можливість переглядати файли наявні в архівах. Більшість з них підтримують найбільш популярні формати архівів “ZIP” та “RAR”.

Багато додатків є безкоштовними, деякі навіть мають відкритий код, проте деякі з них мають вбудовану рекламу. Інколи це може перешкоджати ефективному

процесу читання, тому пропонуються платні версії цих додатків, але вже із вимкнутою рекламою. Також платні версії деяких додатків мають ширший набір функцій.

Отже, можна виділити такі основні характеристики додатків для читання електронних книг:

- вартість;
- функціонал;
- зручність інтерфейсу;
- кількість підтримуваних форматів;
- наявність бази даних книг.

У наступних пунктах подамо основну інформацію про деякі додатки для читання електронних книг розроблені для операційної системи “Android” їх переваги та недоліки [6].

## 1.2 Інтерфейс і функціонал додатків для читання електронних книг на Android

Почнемо наш огляд із додатка “ReadEra”. Цей додаток доступний лише для операційної системи “Android”.

Додаток не містить реклами, є абсолютно безкоштовним та дозволяє читати й переглядати книги в форматах PDF, EPUB, Microsoft Word (DOC, DOCX, RTF), Kindle (MOBI, AZW3), DJVU, FB2, TXT, ODT і CHM. Також додаток дозволяє читати книги запаковані в архів формату ZIP без необхідності його розпаковування [7].

На рисунку 1.1 зображено головне меню додатку. Бачимо широкий набір можливостей.

Додаток працює в автономному режимі та автоматично виявляє книги й документи присутні у файловій системі пристрою. Тобто достатньо просто завантажити книгу Epub, журнал PDF, документи Microsoft Word або статтю у форматі PDF з Інтернету, щоб вони з'явилися в бібліотеці додатку для читання.

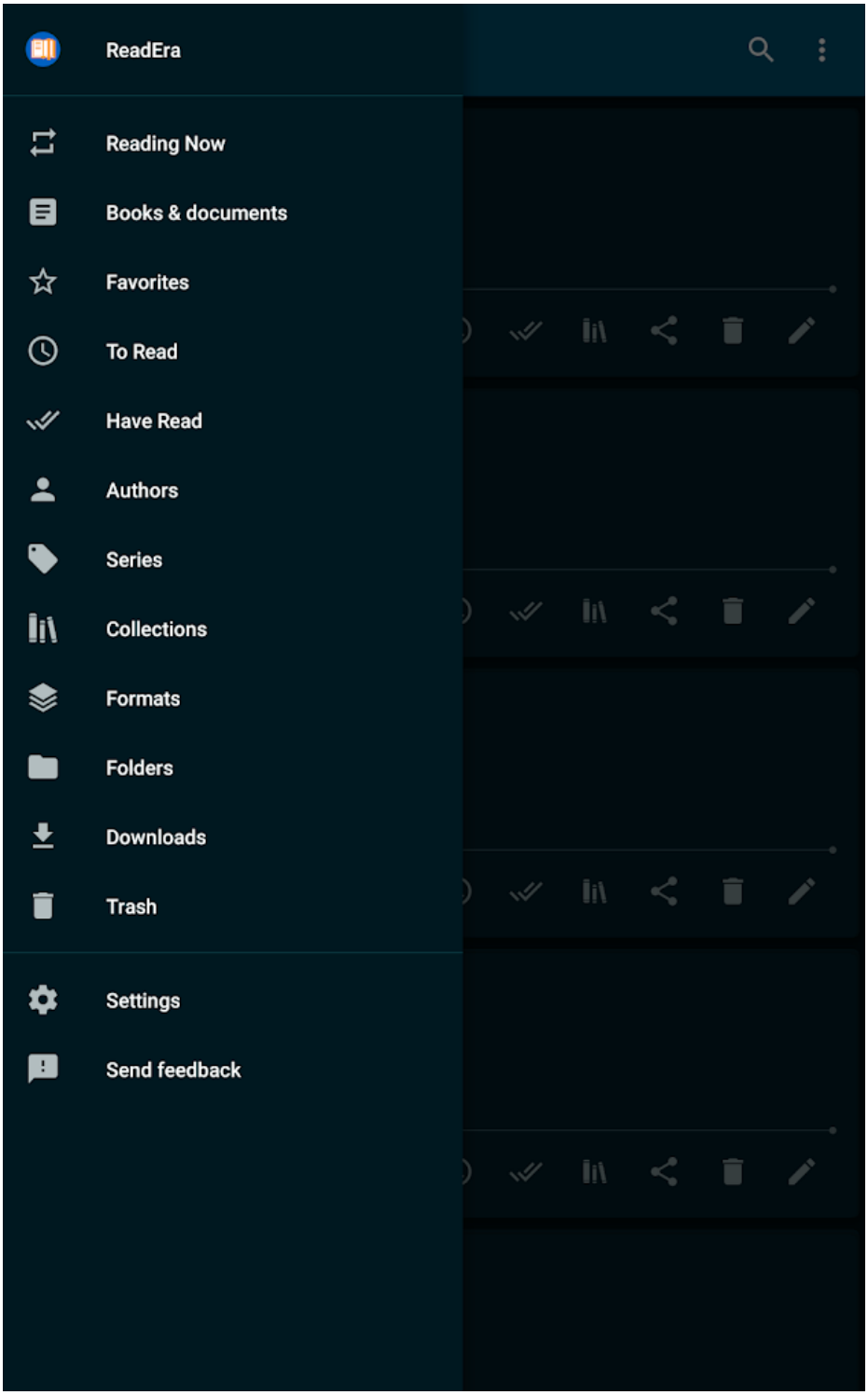


Рисунок 1.1 — Меню додатку “ReadEra”.

Реалізована зручна навігація по директоріях пристрою та самому документу. Також організовано групування книг за авторами та серіями. Так само додаток дозволяє створювати списки читання книг та особисті тематичні колекції. Книги та документи можна додавати до однієї або кількох колекцій одночасно. На

рисунку 1.4 зображено вигляд бібліотеки додатку на екрані планшету. Є організовано можливість збереження закладок певних сторінок книги та нотаток у тексті. Також є можливість швидкого доступу до налаштувань читання, змісту, закладок, виділень тексту, нотаток, історії перегляду сторінок у книзі та інших параметрів електронної книги.

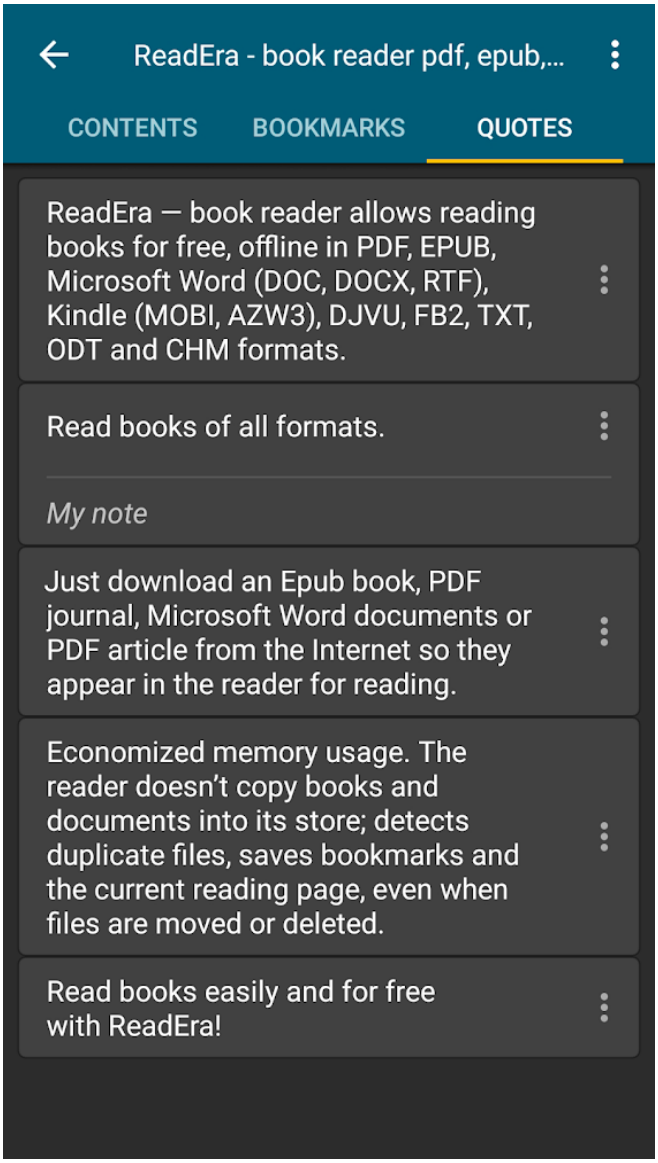


Рисунок 1.2 — Меню зі змістом, нотатками та закладками.

На рисунку 1.2 зображено меню додатку зі змістом книги, нотатками, та закладками. Як бачимо, меню є досить зрозумілим та продуманим. Це дозволяє швидко орієнтуватися у книзі. Навігація по книзі організована за допомогою

вказівника сторінки або лінії прогресу. Тексти виносок у форматах Epub, Mobi, Docx, Fb2 друкуються внизу сторінки, як у паперовій книзі. Показує загальну кількість сторінок книги та окремо сторінок розділу, що читається. Автоматично зберігається поточна сторінка читання [7]. Інтерфейс реалізує такі колірні режими при читанні книг як день, ніч, сепія та консоль. Також є горизонтальний та вертикальний режим віддзеркалення сторінки. Можна налаштувати яскравість сторінки та розмір полів сторінки навіть у файлах форматів PDF та DjVu. Є можливість відрегулювати тип шрифту, його розмір, жирність, міжрядковий інтервал та розстановку переносів для форматів Microsoft Word, Epub, Kindle (Mobi, Azw3), Fb2, TXT і ODT. На рисунку 1.3 зображено меню настройки зовнішнього вигляду відображення тексту книги. Для файлів форматів PDF та Djvu присутня опція масштабування.

Додаток “ReadEra” дозволяє читати кілька книг і документів одночасно та не боятися втратити прогрес читання. Наприклад, можна одночасно читати книги Epub та журнали PDF, розміщуючи їх на екрані пристрою в режимі розділеного екрана (два вікна). Або можна читати файли Microsoft Word, ODT, PDF, Epub/Mobi та Kindle, перемикаючись між ними за допомогою системної кнопки "Активні програми" [7]. Така функція може бути корисна при читанні наукової літератури та дає можливість відкрити до прикладу словник термінів та якусь наукову статтю. Також це може бути корисним при вивченні підручників.

Інтерфейс додатку адекватно відображається як і на екрані смартфона так і на екрані планшету. Вигляд інтерфейсу додатку на планшеті зображено на рисунку 1.4.

Отже, можна виділити основні характеристики цього додатку.

**Переваги:**

- відсутність реклами та нав'язування внутрішніх покупок;
- читання книг з архіву ZIP, без необхідності їх розпаковування у внутрішню пам’ять пристрою;
- настройка полів у файлах PDF та DjVu. Можливість розділення стовпців на окремі сторінки;

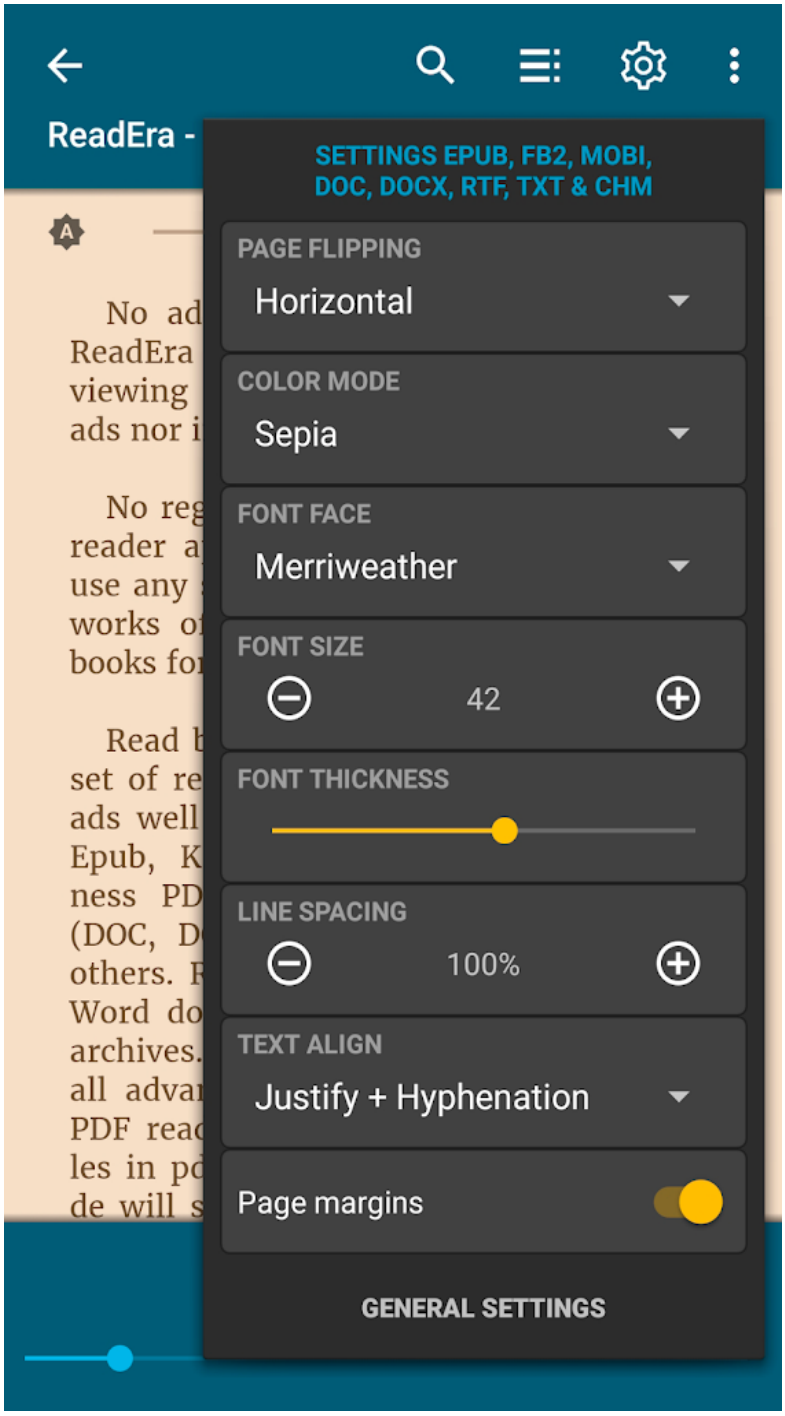


Рисунок 1.3 — Зразок меню настройки відображення тексту книги.

- автоматичне виявлення книг і документів;
- існує багато способів налаштувати відображення тексту;
- продумана навігація по книзі;
- зручні теми при читанні книг;
- пошук повторів файлів;

- мультидокументний режим.

**Основні недоліки:**

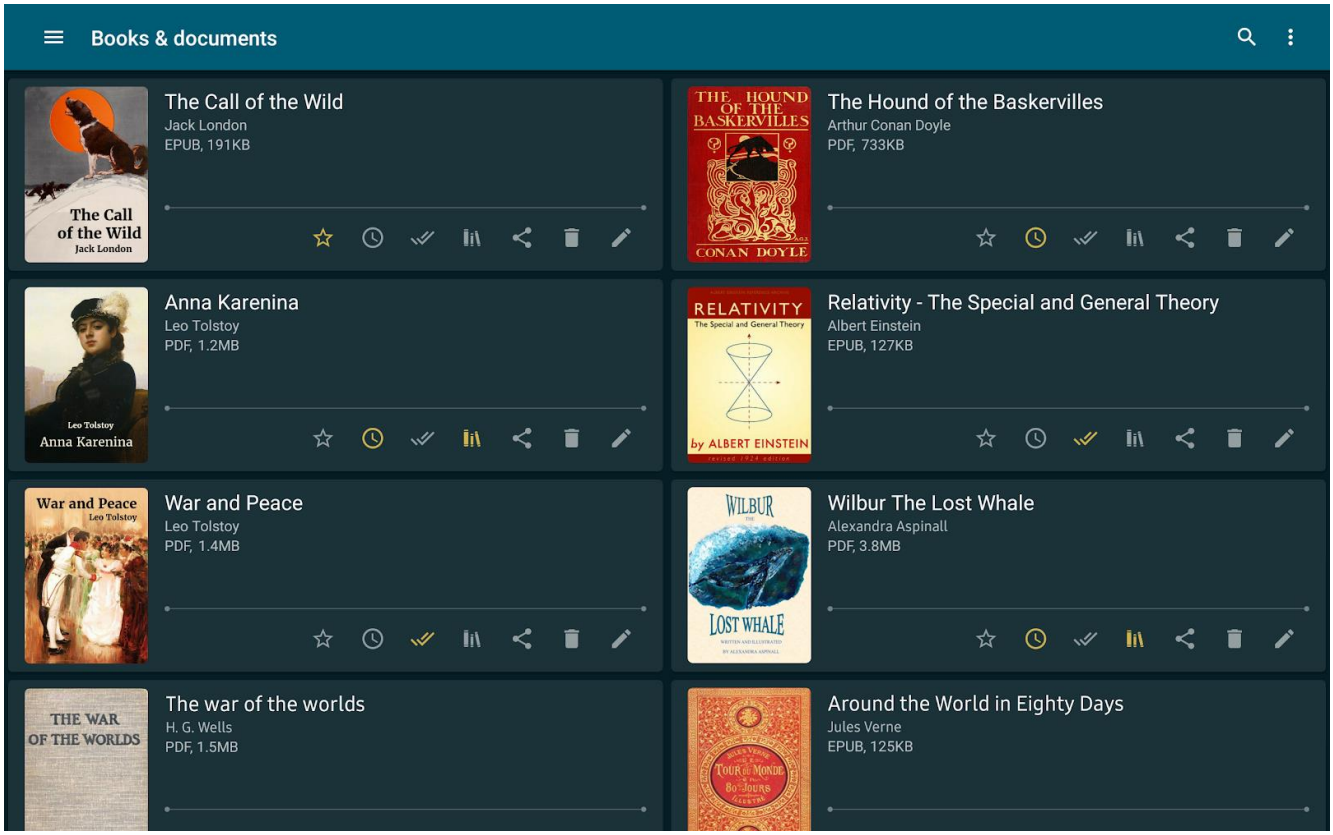


Рисунок 1.4 — Бібліотека у додатку “ReadEra”

- немає голосового озвучення тексту книги;
- синхронізація потребує окремого резервного виклику;
- немає мережевих бібліотек та можливості до них підключитися.

Наступним розглянемо додаток “Reasily”, який також доступний лише для операційної системи “Android”.

Цей додаток має можливість відкривати лише один формат файлу EPUB при цьому повністю зберігаючи авторське форматування. Також він дає змогу переглядати елементи книги створені з допомогою мови математичної розмітки MathML (*Mathematical Markup Language*) [8].

Додаток дозволяє відкривати декілька книг немов різні програми. Можна перемикатися між відкритими книгами та списком книг за допомогою кнопки

"Останні програми" пристрою. Приклад роботи з даною функцією зображено на рисунку 1.5.

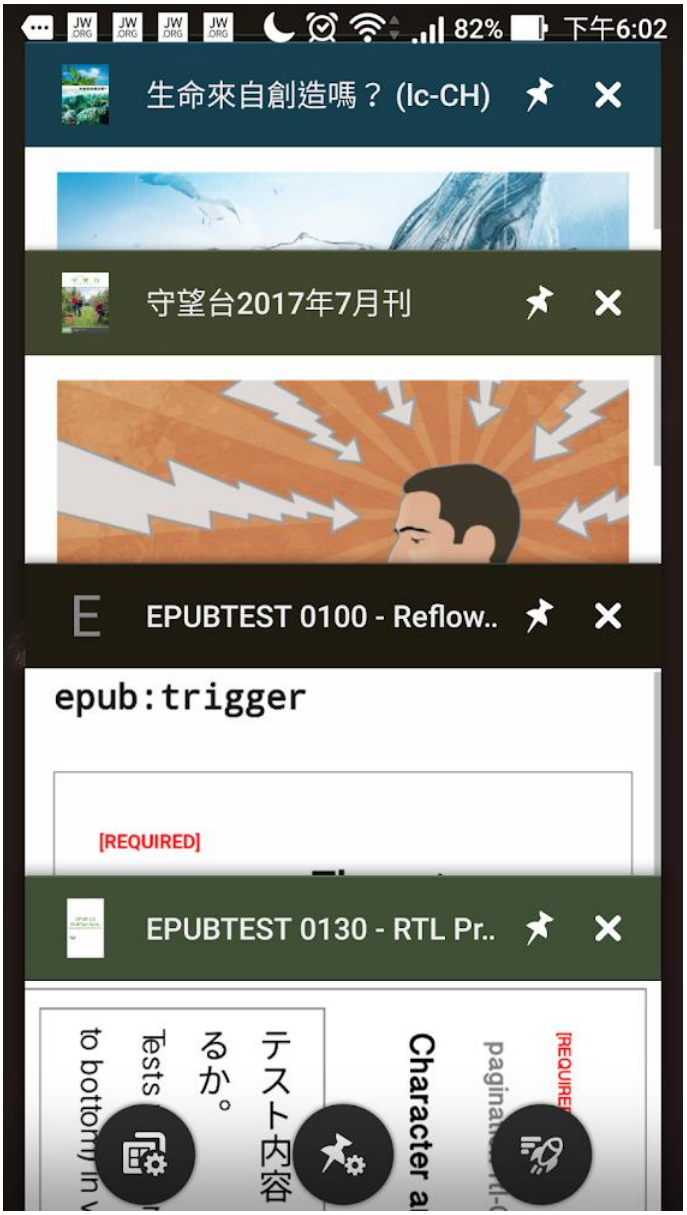


Рисунок 1.5 — Мультидокументний режим у додатку “Reasily”.

Програма дозволяє створювати нотатки у тексті книги. Також є організована синхронізація між додатками користувача на різних пристроях. Організована можливість створити ярлик книги на робочому столі пристрою [8].

Великим плюсом даного додатку є наявність вбудованого словника та перекладача. Також є можливість пошуку слова у тексті відкритої книги. Приклад роботи з перекладачем зображено на рисунку 1.6.

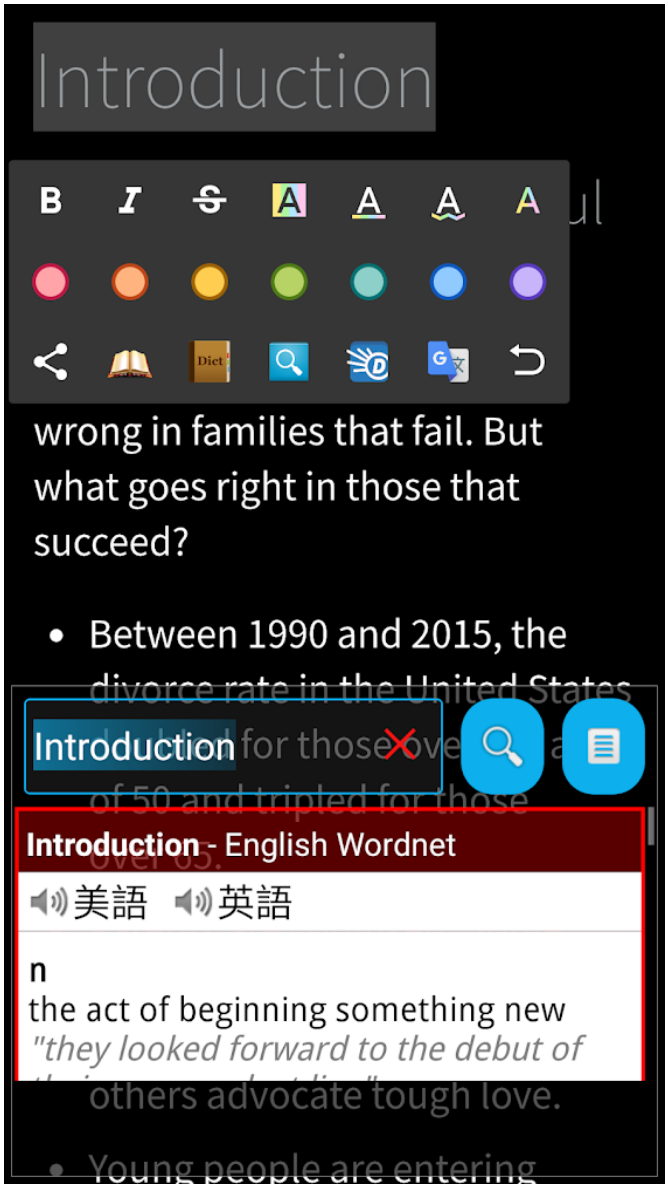


Рисунок 1.6 — Робота з перекладачем у додатку “Reasily”.

Отже, до основних переваг додатку можна віднести “Reasily”**:**

- зберігається авторське форматування тексту (що особливо важливо для підручників з програмування, адже потрібно читати код складений автором);
- є можливість одночасного запуску декількох файлів в окремих вікнах “Android”;
- мінімалізм у всьому;
- немає реклами;
- легка синхронізація нотаток між пристроями;

- оптимізований, а отже швидкий код який дозволяє швидко відкривати великі книжкові файли;
- підтримка розмітки MathML.
- До основних недоліків можна віднести:
- можливість роботи лише з одним типом файлу “EPUB”;
- деякі користувачі вказують на проблеми із синхронізацією.
- Загалом додатки мають більш-менш однаковий набір можливостей та можуть забезпечити потреби більшості користувачів.

## 1.3 Інтерфейс і функціонал додатків для читання електронних книг на IOS

У даній частині огляду зупинимося на додатку “eBoox”. Цей додаток доступний як для “Android”, так і для “IOS”. Перевагою цього додатку є можливість завантажувати книги з інтернету. Більше того програма підтримує роботу з декількома базами даних книг серед яких надзвичайно популярні “Amazon” та “ЛитРес”. Також реалізовано можливість синхронізації закладок, нотаток та прогресу читання на декількох пристроях [9].

Додаток працює з широким набором форматів файлів - “FB2”, “EPUB”, “DOC”, “DOCX”, “MOBI”, “PRC”, “TXT”, “RTF”, “ODT”, “HTML”, “CBR”, “CBZ”. Також програма дозволяє відкривати архіви з форматами “ZIP” та “RAR” [9].

Суттєвою перевагою даного додатку є гнучке налаштування режимів відображення тексту — є можливість настроїти колір фону та шрифту та його розмір тип та розмір.

Загалом додаток має красивий та зрозумілий інтерфейс користувача. На рисунку 1.7 зображено бібліотеку книг відображену в інтерфейсі додатку.

Отже, до основних переваг додатку можна віднести:

- просте завантаження книг із внутрішньої пам’яті пристрою, а також із баз даних присутніх в мережі Інтернет;
- синхронізація прогресу читання, закладок і нотаток;

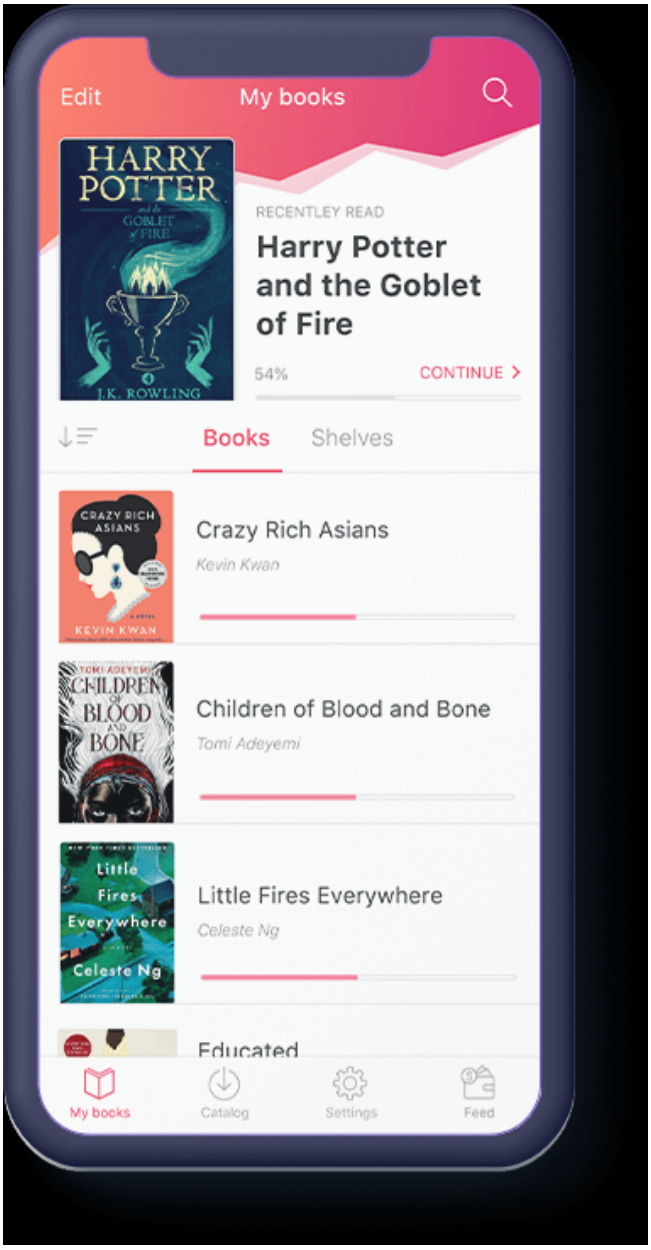


Рисунок 1.7 — Інтерфейс додатку “eBoox”.

- вибір кольору фону;
- регулювання яскравості;
- відкриття найбільш популярних форматів архівів;

До недоліків можна віднести:

- не завжди коректно працює синхронізація;
- в останніх версіях додаток почав частіше зависати.

Додатки, розроблені для операційної системи “IOS”, по функціоналу та побудові інтерфейсу користувача загалом мало чим відрізняються від додатків розроблених для операційної системи “Android” і також можуть задовольнити потреби більшості користувачів.

Можна зробити висновок про те, що всі розглянуті додатки володіють досить складним інтерфейсом, який не завжди є доступним для більшості користувачів.

## 1.4 Вибір цільової ОС та вимоги до додатка

Як було сказано в попередньому розділі масова частка пристроїв під управлінням операційної системи Android майже у два рази перевищує масову частку пристроїв під управлінням операційної системи IOS. Тому для того, щоб охопити якомога більшу аудиторію користувачів будемо розробляти додаток для операційної системи Android.

Наступним питанням для розгляду повинна бути цільова версія операційної системи Android. На рисунку 1.8 зображено розподіл кількості пристроїв під управлінням різних версій Android. Можемо бачити що кількість пристроїв під управлінням нових версій ОС є значно меншою ніж старших. Тому можна зробити висновок що найбільш оптимально встановити мінімальну версію Android для нашого додатку на рівні Jelly Bean, або ж API 16. Такий вибір дозволить охопити велику частку пристроїв [10].

Під час розгляду основних функцій існуючих додатків для читання електронних книг вдалося виділити серед них найбільш основні та затребувані:

- відкриття файлів з файлової системи пристрою;
- зміна розміру шрифта;
- створення та відкриття закладок;

- пошук по книзі;
- перегляду змісту книги;
- створення списку улюблених книг;
- зручне меню зі списком востаннє відкритих документів.

Саме ці функції вважаємо доцільним реалізувати у нашому додатку. Також під час створення інтерфейсу потрібно звернути особливу увагу на його простоту та зручність, а також забезпечити широкий формат підтримуваних документів та книг. Наш додаток отримає змогу відкривати такі найбільш розповсюджені формати документів електронних книг: “TXT”, “PDF”, “XPS”, “OXPS”, “CBZ”, “EPUB”, “FB2”, “XML”.

| ANDROID PLATFORM VERSION | API LEVEL | CUMULATIVE DISTRIBUTION |
|---|---|---|
| 4.0 Ice Cream Sandwich | 15 | |
| 4.1 Jelly Bean | 16 | 99,8% |
| 4.2 Jelly Bean | 17 | 99,2% |
| 4.3 Jelly Bean | 18 | 98,4% |
| 4.4 KitKat | 19 | 98,1% |
| 5.0 Lollipop | 21 | 94,1% |
| 5.1 Lollipop | 22 | 92,3% |
| 6.0 Marshmallow | 23 | 84,9% |
| 7.0 Nougat | 24 | 73,7% |
| 7.1 Nougat | 25 | 66,2% |
| 8.0 Oreo | 26 | 60,8% |
| 8.1 Oreo | 27 | 53,5% |
| 9.0 Pie | 28 | 39,5% |
| 10. Android 10 | 29 | 8,2% |

Рисунок 1.8 — Розподіл версій ОС Android.

# 2. БАЗОВІ ВИМОГИ ДО РОЗРОБКИ ДОДАТКІВ НА ANDROID

## 2.1 Особливості ОС Android

Android - операційна система для смартфонів і безлічі інших пристроїв, наприклад розумних годинників, планшетів, SmartTV і т.д. Спочатку ця операційна система розроблялася каліфорнійською компанією Android Inc.. Згодом цю компанію викупив американський пошуковий гігант Google.

Частка Android на ринку ОС складає приблизно 82%, Кількість додатків для Андроїд в магазині додатків Google Play перевищує 2.8 мільйона проти 2.2 мільйона в магазині додатків AppStore. Акаунт розробника для публікації додатка в Play Маркет платний, проте коштує набагато менше ніж в App Store до того ж платіж здійснюється одноразово. Для розміщення додатку в App Store необхідно здійснювати щорічний платіж [11].

Для розробки додатків під операційну систему Android в більшості випадків використовується мова програмування Java. Хоча зараз набирає популярності і мова програмування Kotlin.

Kotlin (Котлін) — це об'єктно-орієнтована мова програмування що є статично типізованою та працює поверх Java Virtual Machine. Її розробкою займається компанія JetBrains. Особливістю цієї мови є можливість компіляції в JavaScript та у виконуваний код ряду платформ через інфраструктуру LLVM. Мова отримала назву на честь острова Котлін у Фінській затоці, на якому розташоване місто Кронштадт [12].

Розробники цієї мови ставили за мету створити більш лаконічну та типобезпечнішу мову ніж Java та простішу, ніж Scala. Наслідком спрощення в порівнянні зі Scala стали також більш швидка компіляція і краща підтримка мови в IDE. Kotlin повністю сумісний з Java, що дозволяє java-розробникам поступово перейти до його використання. Наприклад, в Android мова вбудовується за допомогою Gradle, що дозволяє для існуючого android-додатку додавати нові функції написані на Kotlin без цілковитого переписування програми.

Так само як і при розробці програм на Java написаний код програми для Android за допомогою наданого Google API Java компілюється в файли класів. Проте Android не використовує віртуальну машину Java (JVM) для виконання файлів класів, натомість, в ньому використовується віртуальна машина Dalvik, яка не є справжньою JVM і тому не працює з Java-байткодами. Проте у сучасних версіях Android Dalvik замінена на Android Runtime. Для виконання на віртуальних машинах Dalvik, файли класів компілюються в формат DEX (Dalvik EXecutable - виконувані файли Dalvik). Після перетворення в формат DEX, файли класів разом з іншими ресурсами об'єднуються в пакети Android (APK) для їх подальшого поширення та інсталяції на різних Android-пристроях.
В основі базової бібліотеки класів віртуальної машини Dalvik лежить підмножина проєкту Apache Harmony, внаслідок чого вона не підтримує весь J2SE API. Проте якщо для написання коду додатків під Android використовується IDE типу Eclipse чи Android Studio, можна особливо не хвилюватися, оскільки там є автодоповнення коду [13].

Віртуальна машина Dalvik - це регістрова віртуальна машина, вона працює в операційній системі Android і виконує програми, написані для Android. Формат байтового коду Dalvik досі використовується як формат розповсюдження, але більше не виконується в нових версіях Android. Dalvik був невід’ємною частиною стеку програмного забезпечення Android до версії Android 4.4 «KitKat».

Програмне забезпечення Dalvik має відкритий код та було написане Даном Борнштейном й назване ним на честь рибальського села Дальвік в Ейяфьордурі, Ісландія звідки походили його предки. Компактний формат файлів програм Dalvik Executable призначений для систем, обмежених у таких ресурсах як швидкодія процесора та пам’ять [14].

Наступником Dalvik є Android Runtime (ART), який використовує однакові файли байт-коду та .dex, але не файли .odex. Метою введення ART було збільшення продуктивності відчутне для кінцевих користувачів. Нове середовище виконання було вперше включено у версію Android 4.4 "KitKat" як попередній перегляд технології та повністю замінило Dalvik у пізніших версіях. Android 5.0

"Lollipop" є першою версією, в якій єдиним включеним середовищем виконання є ART [14].

Android Runtime (ART) - це середовище виконання програми, що виконує переклад байт-коду програми у власні інструкції, які пізніше виконуються середовищем виконання пристрою [15]. Android 2.2 "Froyo" вніс у Dalvik компіляцію перед виконанням (JIT — just-in-time), оптимізуючи виконання додатків шляхом постійного профілювання програм при кожному її запуску та динамічної компіляції часто виконуваних коротких сегментів свого байт-коду у власний машинний код. Ці короткі сегменти байт-коду називаються “слідами”, або “traces” англійською. Поки Dalvik інтерпретує решту байт-коду програми, власне виконання цих коротких сегментів байт-коду, або “слідів”, забезпечує значне покращення продуктивності. На відміну від Dalvik, ART вводить використання компіляції перед виконанням (AOT — ahead-of-time) шляхом компіляції цілих програм у власний машинний код після їх встановлення. Усуваючи інтерпретацію Dalvik та компіляцію JIT на основі “слідів”, ART покращує загальну ефективність виконання програми тим самим зменшуючи споживання енергії, що призводить до поліпшення автономності мобільних пристроях, які живляться від батареї. У той же час ART забезпечує швидше виконання додатків, покращений механізм виділення пам'яті та збирання сміття (GC — garbage collect), нові функції налагодження програм та більш точне профілювання програм на високому рівні. Для підтримки зворотної сумісності ART використовує той самий вхідний байт-код, що і Dalvik, що постачається через стандартні файли .dex як частина файлів APK, тоді як файли .odex замінюються виконуваними файлами та виконуваним форматом (ELF). Після того, як додаток скомпільовано на пристрої за допомогою утиліти під назвою “dex2oat”, яка входить до складу ART, він запускається виключно зі скомпільованого виконуваного файлу ELF. Такий підхід усуває різні затратні витрати на виконання додатків, пов'язані з інтерпретацією Dalvik та компіляцією JIT на основі трасування, тобто “слідів”.

Отже, до мінусів ART можна віднести наступне:

- збільшений час установки програми;

- додаток потребує більше місця у внутрішній пам'яті пристрою;
- завантаження операційної системи потребує більше часу.

До плюсів ART можна віднести наступне:

- додатки запускаються та виконуються швидше;
- перемикання між додатками займає більше часу;
- ефективніше використання оперативної пам’яті пристрою.

Рівнем вище від ART знаходиться Application Framework, який також прийнято називати рівнем каркасу додатків. Саме через такі каркаси додатку розробники отримують доступ до API, який надано компонентами системи, що лежать на рівень нижче. Крім того, завдяки архітектурі фреймворка, будь-якому додатку можуть бути надані вже реалізовані можливості інших додатків, до яких дозволено отримувати доступ.

До базового набіру сервісів та систем, що є частинами згаданого вище фреймворка та лежать в основі кожного додатку, входять:

- велика колекція представлень, або ж Views, які застосовуються для створення візуальної частини додатку, таких як, списки, текстові поля, таблиці, кнопки або ж навіть вбудовані web-браузери;
- контент-провайдери, або ж Content Providers, основною функцією яких є керування даними, які одні додатки відкривають один для другого, для використання у своїй роботі;
- менеджер ресурсів, або ж Resource Manager, що забезпечує доступ до ресурсів які не мають функціональності, тобто таких що не несуть програмного коду. До прикладу стрічкові дані, графіка, файли та інше;
- менеджер оповіщень, або ж Notification Manager, що забезпечує можливість всім додаткам показувати власні повідомлення для користувача пристрою в рядку стану;
- менеджер дій, або ж Activity Manager, що забезпечує управління життєвими циклами додатків та організовує роботу системи навігації по історії дій та надає інформацію про неї;

- менеджер розташування, або ж Location Manager, який забезпечує додатки інформацією про актуальні дані поточного географічного положення пристрою.

Таким чином, завдяки Application Framework, організовано багаторазове використання компонентів додатків та операційної системи. І додатки в ОС Android мають змогу отримувати в своє розпорядження допоміжний функціонал. Звичайно, всі ці дії відбуваються в рамках політики безпеки закладеної в операційну систему [16].

Також слід звернути увагу на те що, фреймворк лише виконує код, який написаний для нього, на відміну від бібліотек, які можуть виконуватися самі. Тобто фреймворк складається з великої кількості бібліотек які мають різний функціонал та призначення, а самі бібліотеки містять в собі набори функцій, які є близькими за логікою.

На вершині програмного стека Android лежить рівень додатків — Applications Level. Аналогічно з тим як набір базових додатків, встановлений на ОС Android за замовчуванням може змінюватися, наприклад, в нього можуть входити браузер, поштовий клієнт, SMS-клієнт, карти, календар, телефон, менеджер контактів і багато інших так само список інтегрованих програм може змінюватися в залежності від версії Android та моделі пристрою. До цього базового набору програм до рівня додатків також відносяться всі програми під платформу Android. До них також відносяться і ті що були інстальовані користувачем.

Найбільш розповсюдженою мовою програмування для розробки додатків є Java, проте слід зазначити , що існує можливість розробки і з допомогою C/C ++ з використанням Native Development Kit, або ж NDK. Також можна створювати власні програми за допомогою конструкторів додатків, таких як App Inventor. Як бачимо, можливостей для розробки додатків існує досить багато [17].

Оскільки мова програмування Java є найбільш розповсюдженою та найкраще документованою розробка додатку буде проводитися саме з використанням цієї мови.

## 2.2 Особливості мови програмування Java

Java — це об'єктно-орієнтована мова програмування, що побачила світ у 23 травні 1995 року. Вона розробляється компанією Sun Microsystems починаючи з 1991 року. Початковою назвою мови була “Oak” і її розробка велася для використання у побутовій електроніці, але згодом її перейменували у Java і почали використовувати для написання аплетів, додатків і серверного програмного забезпечення [18].

Мова Java зародилася як частина проєкту створення передового програмного забезпечення для різних побутових приладів. Спочатку реалізація цього програмного забезпечення була розпочата на мові C ++, але незабаром виник ряд проблем тому було вирішено, що найкращим засобом боротьби з ними є зміна самого інструменту - мови програмування [19]. Стало очевидним, що необхідна мова програмування, яка буде незалежна від платформи. Це рішення дозволяє створювати програми, які не потрібно компілювати окремо для кожної архітектури і можна використовувати на різних процесорах під різними операційними системами. Мова Java потрібна для створення інтерактивних продуктів для мережі Internet.

Можна зробити висновок про те що, більшість архітектурних рішень, прийнятих при створенні Java, були продиктованими намаганням надати їй подібний до C/C ++ синтаксис. В Java використовуються практично ідентичні вирази для оголошення змінних, передачі параметрів, операторів і управління потоком виконанням коду. Мова Java включає в себе всі позитивні сторони C ++.

Наступні три сильні сторони об'єдналися в технології мови Java:

- Java надає для широкого використання так звані аплети (Applets) - невеликі, надійні, динамічні активні мережеві додатки, що вбудовуються в сторінки Web та не залежать від платформи. Аплети Java можуть налаштовуватися і поширюватися користувачам з такою ж легкістю, як будь-які документи HTML;

- Java демонструє потужність об'єктно-орієнтованого підходу до розробки додатків, завдяки поєднанню простого та знайомого синтаксису із надійним та зручним в роботі середовищем розробки. Такий підхід дозволяє широкому колу програмістів швидко створювати нові програми та аплети;
- Java надає програмісту багатий набір класів для простого та зрозумілого абстрагування багатьох системних функцій, використовуваних при роботі з вікнами, мережею та функціями вводу-виводу. Ключова риса цих класів полягає в тому, що вони забезпечують створення незалежних від використовуваної платформи абстракцій для широкого спектра системних інтерфейсів.

## 2.3 Життєвий цикл та структура додатка на Android

Ключовим компонентом для створення візуального інтерфейсу в додатку Android є activity (активність). Нерідко activity асоціюється з окремим екраном або вікном додатка, а перемикання між вікнами відбувається як переміщення від однієї activity до іншої. Додаток може мати одну або декілька activity. Наприклад, при створенні проєкту з порожньою Activity в проєкт за замовчуванням додається один клас Activity — MainActivity, з якого і починається робота програми:

```
public class MainActivity extends Activity {
    //вміст класу
}
```

Всі об'єкти activity є об'єктами класу android.app.Activity, який надає функціональність для всіх activity. Проте активність не обов’язково повинна наслідувати саме клас Activity. Наприклад, вона може наслідувати AppCompatActivity або FragmentActivity. Проте ці класи, хоч і не прямо, наслідують базовий клас Activity [20].

Всі додатки у операційній системі Android мають певний, строго визначений системою життєвий цикл. При запуску додатка користувачем система надає цьому додатку високий пріоритет. Кожна програма запускається у вигляді окремого

процесу, що дозволяє системі давати одним процесам вищий пріоритет, а іншим нижчий. Завдяки цьому, наприклад, є змога при відкритому одному додатку приймати вхідні дзвінки. Після припинення роботи з додатком, система звільняє всі пов'язані ресурси, переводить додаток у розряд низькопріоритетного та закриває його [21].

Всі об'єкти активностей, які є в додатку, управляються системою у вигляді стека activity, який називається back stack. При запуску нової активності вона поміщається поверх стека і виводиться на екран пристрою, поки не з'явиться нова activity. Коли поточна активність закінчує свою роботу (наприклад, користувач виходить з програми), вона видаляється з стека, і відновлює роботу та activity, яка раніше була другою в стеці.

Після запуску, activity проходить через ряд подій, які обробляються системою і для обробки яких існує ряд зворотних викликів:

```
protected void onCreate(Bundle saveInstanceState);
protected void onStart();
protected void onRestart();
protected void onResume();
protected void onPause();
protected void onStop();
protected void onDestroy();
```

Схематично взаємозв'язок між усіма цими зворотними викликами представлено на рисунку 2.1.

Розглянемо детальніше роль цих функцій зворотніх викликів.

**onCreate** - перший метод, з якого починається виконання activity. У цьому методі activity переходить в стан Created. Цей метод обов'язково повинен бути визначений в класі activity. У ньому проводиться первісна настройка activity. Зокрема, створюються об'єкти візуального інтерфейсу. Цей метод отримує об'єкт Bundle, який містить попередній стан activity, якщо він був збережений. Якщо activity заново створюється, то даний об'єкт має значення null.

Якщо ж activity вже раніше була створена, але перебувала в загальмованому стані, то bundle містить пов'язану з activity інформацію.

Після того, як метод **onCreate**() завершив виконання, activity переходить в стан Started, і система викликає метод onStart().

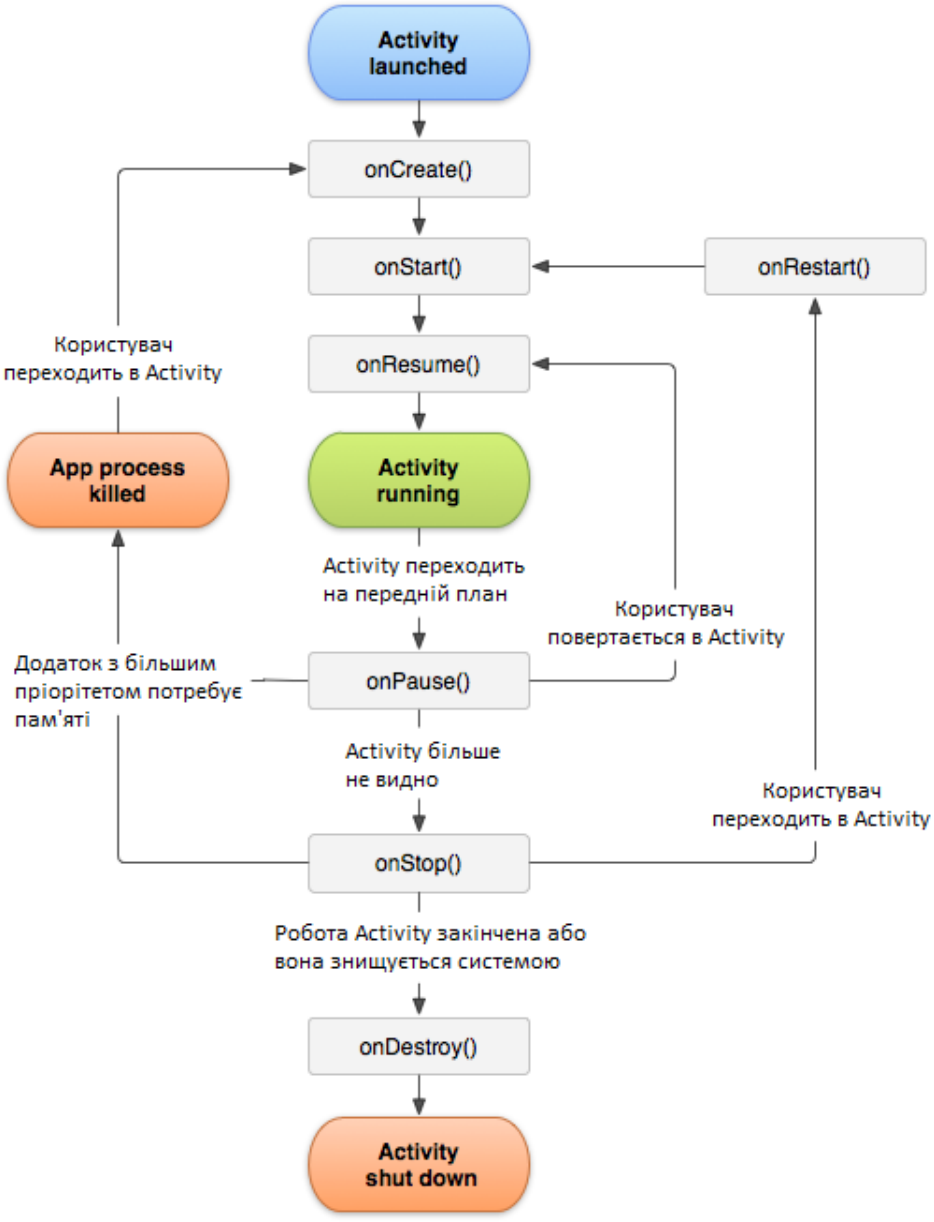


Рисунок 2.1 — Життєвий цикл Activity.

У методі **onStart()** здійснюється підготовка до відображення активності на екрані пристрою. Як правило, цей метод не вимагає перевизначення, а всю роботу робить вбудований код. Після завершення роботи методу activity відображається

на екрані, викликається **onResume()**, а активіті переходить у стан Resumed. У цьому стані користувач дістає змогу взаємодіяти з activity.

Активність залишається в цьому стані, поки вона не втратить фокус, наприклад, внаслідок перемикання на іншу activity або просто через вимкнення екрану пристрою.Якщо користувач вирішить перейти до іншої activity, то система викликає метод **onPause**, а activity переходить в стан Paused. У цьому методі можна звільняти використовувані ресурси, припиняти процеси, наприклад, відтворення аудіо, анімацій, зупиняти роботу камери (якщо вона використовується) і т.д.

Проте, слід мати на увазі, що в цей стан activity як і раніше залишається видимою на екрані, і на роботу даного методу відводиться дуже мало часу, тому не варто тут зберігати якісь дані, особливо якщо при цьому потрібно звернення до мережі, наприклад, відправка даних по інтернету, або звернення до бази даних - подібні дії краще виконувати в методі onStop().

Після виконання цього методу activity стає невидимою, не відображається на екрані, але вона все ще активна. І якщо користувач вирішить повернутися до цієї activity, то система викличе знову метод onResume, і activity знову з'явиться на екрані.

Інший ситуація виникає, якщо система бачить, що для роботи активних додатків необхідно більше пам'яті. В такому випадку система може сама завершити роботу activity, яка невидима і знаходиться у фоні. Або користувач може натиснути на кнопку Back, або ж Назад. У цих випадках викликається метод onStop.

У методі **onStop** активність переходить в стан Stopped. У цьому стані activity повністю невидима. У методі onStop слід зівльнити використовувані ресурси, які не потрібні користувачеві, коли він не взаємодіє з activity. Тут також можна зберігати дані, наприклад, у базу даних.

При цьому під час стану Stopped activity залишається в пам'яті пристрою та зберігає стан всіх елементів інтерфейсу. Наприклад, якщо в текстове поле був

введений якийсь текст, то після відновлення роботи activity і переходу її в стан Resumed ми знову побачимо в текстовому полі раніше введений текст.

Якщо після виклику методу **onStop** користувач вирішить повернутися до колишньої activity, тоді система викличе метод onRestart. Якщо ж activity зовсім завершила свою роботу, наприклад, через закриття програми, то викликається метод onDestroy().

Робота activity завершується викликом методу **onDestroy**. Він може бути викликаний якщо система вирішить знищити активність в силу конфігураційних причин (наприклад, поворот екрану) або при виклику методу finish() з коду додатку. Слід зазначити, що при зміні орієнтації екрану система завершує activity і потім створює її заново, викликаючи метод onCreate [22].

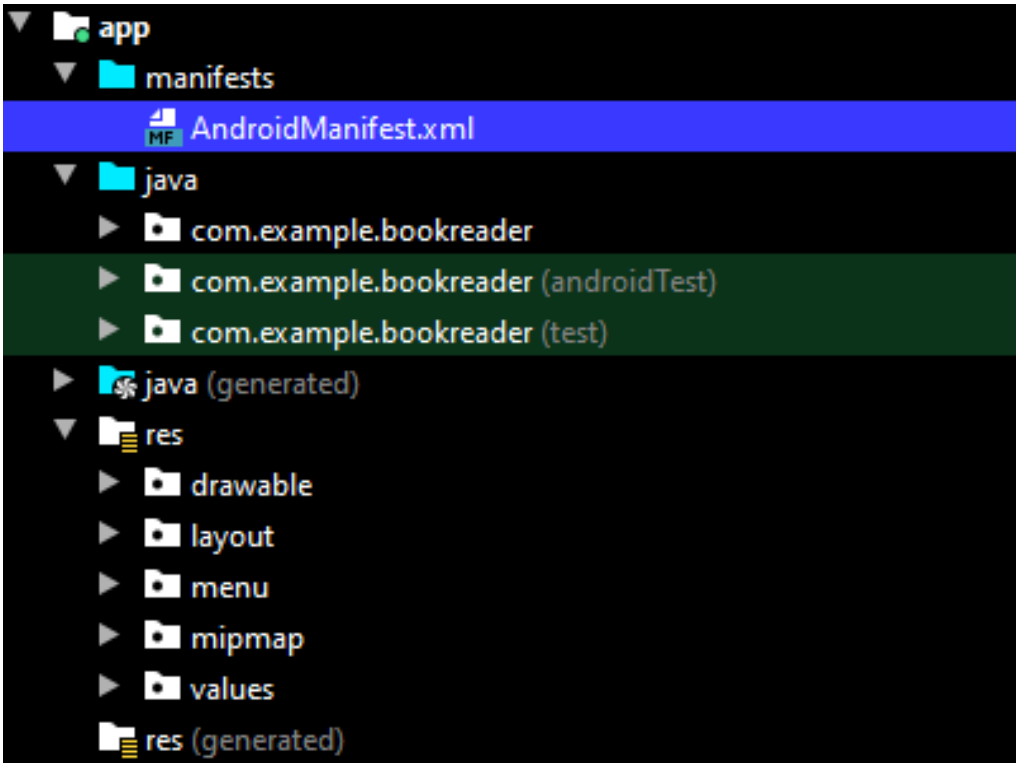


Рисунок 2.2 — Розташування файлу маніфесту у дереві проєкту.

Кожна програма містить файл AndroidManifest.xml, що визначає важливу інформацію про програму. Також цей файл можна називати файлом маніфесту. Даний файл визначає таку інформацію про програму як назва, версія, іконка, дозволи, які використовує додаток, а також реєструє всі використовувані класи

activity, сервіси і т.д. Цей файл знаходиться у файлах проєкту в папці manifests. На рисунку 2.2 показано розташування даного файлу у дереві проєкту. Фіолетовим кольором виділено файл маніфесту.

Розглянемо файл маніфесту на прикладі такого файлу з нашого проєкту BookReader:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.bookreader">
    <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.MANAGE_DOCUMENTS"
        tools:ignore="ProtectedPermissions" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.Light">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
        <activity android:name=".BookActivity" />
    </application>
</manifest>
```

Елементом кореневого рівня є вузол manifest. В даному випадку визначається тільки пакет додатку — package = "com.example.bookreader". Власне це визначення файлу маніфесту є таким за замовчуванням, якщо не враховувати трьох елементів uses-permision, які надають дозволи програмі на роботу із внутрішньою пам’яттю пристрою. У кожному конкретному випадку може лише відрізнятися пакет додатку, решту вмісту при створенні проєкту з порожньою activity буде аналогічним.

Більшість налаштувань рівня додатку визначаються елементом application. Ряд налаштувань задаються за допомогою атрибутів. За замовчуванням застосовуються наступні атрибути:

- android: allowBackup вказує, чи буде для додатка створюватися резервна копія. Значення android: allowBackup = "true" дозволяє створення резервної копії;
- android: icon встановлює іконку програми. При значенні android: icon = "@ mipmap / ic_launcher" іконка програми береться з каталогу res / mipmap;
- android: roundIcon встановлює круглу іконку програми. Також береться з каталогу res / mipmap;
- android: label задає назву додатка, яка буде відображатися на мобільному пристрої в списку додатків і в заголовку при його запуску. В даному випадку вона зберігається в ресурсах стрічкового типу — android: label = "@ string / app_name";
- android: supportsRtl вказує, чи можуть використовуватися різні види RTL API — спеціальні API для роботи з правобічної орієнтацією тексту (наприклад, для таких мов як арабська або фарсі);
- android: theme встановлює тему програми. Тема визначає загальний стиль програми. Значення @ style / Theme.ViewApp застосовує Theme.ViewApp " з каталогу res/values/themes.

Вкладені елементи activity визначають всі використовувані в додатку activity. В даному випадку видно, що в додатку є тільки одна activity — MainActivity.

Елемент intent-filter в MainActivity вказує, яким чином ця активність буде використовуватися. Наприклад, за допомогою вузла action android: name = "android.intent.action.MAIN", показано що дана activity буде вхідною точкою в додаток.

Елемент category android: name = "android.intent.category.LAUNCHER" вказує, що MainActivity відображає стартовий екран, який відображається під час запуску програми.

Файл маніфесту може містити безліч елементів, які мають безліч атрибутів. Всі ці можливі елементи та їх атрибути можна знайти в документації на операційну систему Android. Далі розглянемо деякі приклади використання найважливіших елементів та атрибутів.

За допомогою атрибутів елемента manifest можна визначити версію програми і її код:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.bookreader"
     android:versionName="1.0"
     android:versionCode="1">
     .
     .
     .
</manifest>
```

Атрибут android: versionName вказує на номер версії, який буде відображатися користувачеві і на яку будуть орієнтуватися користувачі при роботі з додатком.

Тоді як атрибут android: versionCode представляє номер версії для внутрішнього використання. Цей номер лише визначає, що одна версія додатка новіша, ніж якась інша з меншим номером номером версії. Цей номер не відображається користувачам.

Також можливо визначити версію в ресурсах, а тут у самому файлі маніфесту посилатися на ресурс.

Для управління версією android sdk в файлі маніфесту визначається елемент <uses-sdk>. Він може використовувати наступні атрибути:

- minSdkVersion: мінімальна підтримувана версія SDK;
- targetSdkVersion: оптимальна версія;
- maxSdkVersion: максимальна версія.

Версія визначається номером API, наприклад, Android 4.1 під назвою Jelly Beans має версію 16, а Android 11 має версію 30.

Іноді додатку для роботи потрібні дозволи на доступ до певних ресурсів, наприклад, таких як список контактів, камера, доступ до місцезнаходження пристрою і т.д. У нашому випадку це доступ до внутрішньої пам'яті пристрою. Тобто для того щоб додаток міг працювати із внутрішньою пам'яттю пристрою, в файлі маніфесту необхідно встановити відповідні дозволи. Для установки дозволів застосовується елемент <uses-permission>:

```
 <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

 <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

Атрибут android: name встановлює назву дозволу: в даному випадку на читання внутрішнього сховища — `READ_EXTERNAL_STORAGE` та запис у внутрішнє сховище `WRITE_EXTERNAL_STORAGE`.

Враховуючи, що світ Android-пристроїв дуже сильно фрагментований, оскільки тут зустрічаються як гаджети з невеликим екраном, так і великі широкоформатні телевізори, бувають випадки, коли необхідно обмежити використання програми для певних розширень екранів. Для цього в файлі маніфесту визначається елемент <supports-screens>:

Даний елемент приймає чотири наступні атрибути:

- android: largeScreens - екрани з діагоналлю від 4.5 до 10 дюймів;
- android: normalScreens - екрани з діагоналлю від 3 до 4.5 дюймів;
- android: smallScreens - екрани з діагоналлю менше 3 дюймів;
- android: xlargeScreens - екрани з діагоналлю більше 10 дюймів.

Якщо атрибут має значення true, то додаток буде підтримуватися відповідним розміром екрану.

Також існує можливість заборонити зміну орієнтації екрану в додатку. Додаток в залежності від положення пристрою може перебувати в альбомній або портретній орієнтації. Для деяких особливостей відображення користувацького інтерфейсу це буває не завжди зручно. За допомогою певних атрибутів можна

зробити так, щоб додаток незалежно від фізичного повороту екрану пристрою використовув тільки одну вибрану орієнтацію. Для цього в файлі маніфесту у атрибутах необхідної активності слід встановити атрибут android: screenOrientation. Наприклад, наступна конфігурація забороняє альбомну орієнтацію:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.bookreader"

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.Light">
        <activity android:name=".MainActivity">

            android:screenOrientation="portrait">

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
        <activity android:name=".BookActivity" />
    </application>
</manifest>
```

Значення android: screenOrientation = "portrait" вказує, що дана activity буде знаходитися тільки в портретній орієнтації. У випадку якщо потрібно встановити лише альбомну орієнтацію, тоді треба використовувати значення android: screenOrientation = "landscape" [23].

Слід зазначити, що кожен модуль проєкту має свій файл маніфесту.

# 3. СТРУКТУРА ДОДАТКУ ТА ЙОГО РОБОТА

## 3.1 Бібліотека MuPDF

Пакет MuPDF складається з таких компонентів як бібліотека програмного забезпечення, інструменти командного рядка та переглядачі для різних платформ.

Бібліотека MuPDF - це легка бібліотека для перегляду PDF, XPS та інших найбільш розповсюджених форматів електронних книг.

Візуалізатор тексту у MuPDF відображає високоякісну згладжену графіку. Він робить текст з метриками та інтервалами точними до часток пікселя таким чином забезпечуючи високу точність відтворення вигляду друкованої сторінки на екрані.

Переглядач не займає багато місця, працює швидко і при цьому є повноцінним оскільки дозволяє відображати багато форматів документів, таких як PDF, XPS, OpenXPS, CBZ, EPUB та FictionBook 2. Інструменти написані для використання з допомогою командного рядка дають можливість редагувати та конвертувати документи в інші формати, такі як HTML, SVG, PDF та CBZ. Також є можливість писати сценарії для роботи з документами за допомогою мови Javascript.

Бібліотека складається з модулів та написана на портативному C, тому існує можливість додавати та видаляти функції які завгодно функції. Також існує Java-бібліотека яка працює з використанням JNI. Вона працює як на Java, так і на Android Oracle. Для роботи бібліотека надає API під назвою MuPDF API, який дозволяє виконувати широкий спектр дій над документами [24].

MuPDF випускається за двома ліцензіями — комерційною та GNU. GNU - General Public License це загальна публічна ліцензія GNU або Загальна громадська ліцензія GNU — одна з найпопулярніших ліцензій на вільне програмне забезпечення, створена Річардом Столменом для проєкту GNU. Часто її скорочено називають GNU GPL чи просто GPL, якщо з контексту зрозуміло, про яку

ліцензію йдеться оскільки існує чимало інших ліцензій зі словами «general public license» у назві.

Бібліотека MuPDF доступна за ліцензією загального призначення GNU Afferro (тепер більш відома як GNU AGPL). Саме цю ліцензію ми використаємо для користування цією бібліотекою в нашому додатку. Така ліцензія вимагатиме від нас використовувати при розповсюдженні нашого додатку таку ж ліцензію - GNU AGPL. Також ця ліцензія вимагатиме у нас готовності надати повний вихідний код додатка користувачу який його встановив [25].

### 3.2 Інтерфейс користувача

При запуску програми відкривається екран головної активності і користувач бачит перед собою список книг, які були востаннє відкриті з допомогою додатку. Якщо ж список пустий, тобто користувач ще не відкривав жодного файлу, на його місці появиться повідомлення про те що список книг порожній - “Booklist is empty”. Повідомлення про пустий список книг можна бачити на рисунку 3.2. На рисунку 3.1 показано знімок екрану додатку одразу після його запуску. Можна бачити відкритий список останніх файлів.

Для кожного файлу зліва відображається вигляд титульної сторінки тексту. Також відображається назва книги та її автор. Знизу кожного елементу списку можна бачити дві кнопки — одну у формі зірочки, а другу у формі смітника. Перша дозволяє додати книгу, або видалити книгу із списку улюблених книг (для прикладу книга Б. Лепкого “Мишка” є додана в улюблені книги), а друга видалити дані про книгу із пам’яті додатку. При натисканні цієї кнопки додаток запитає чи дійсно користувач хоче видалити книгу. Для відображення списку улюблених книг достатньо вибрати “Favorite” замість “Recent” у правому верхньому куті екрана. Для відкриття нового файлу книги потрібно натиснути на значок папки, який знаходить у правому нижньому куті екрана. Користувачу буде показано переглядач файлової системи пристрою.

При першому запуску щойно інстальованого додатку на пристрої програма відкриє діалог запиту від користувача дозволу на доступ до внутрішньої пам’яті пристрою. Скріншот цього діалогу показано на рисунку 3.2.

При натисканні на один з елементів списку відкривається активність із детальнішим оглядом файлу електронної книги. Приклад такого екрану зображено на рисунку 3.3. Бачимо вкладки із збереженими користувачем закладками та змістом книги. Довге натискання по закладці викликає меню для її видалення. Коротке — відкриває книгу на закладеній сторінці. Для відкриття книги на останній переглянутій сторінці достатньо натиснути по її назві, або ж по зображенню титульної сторінки.

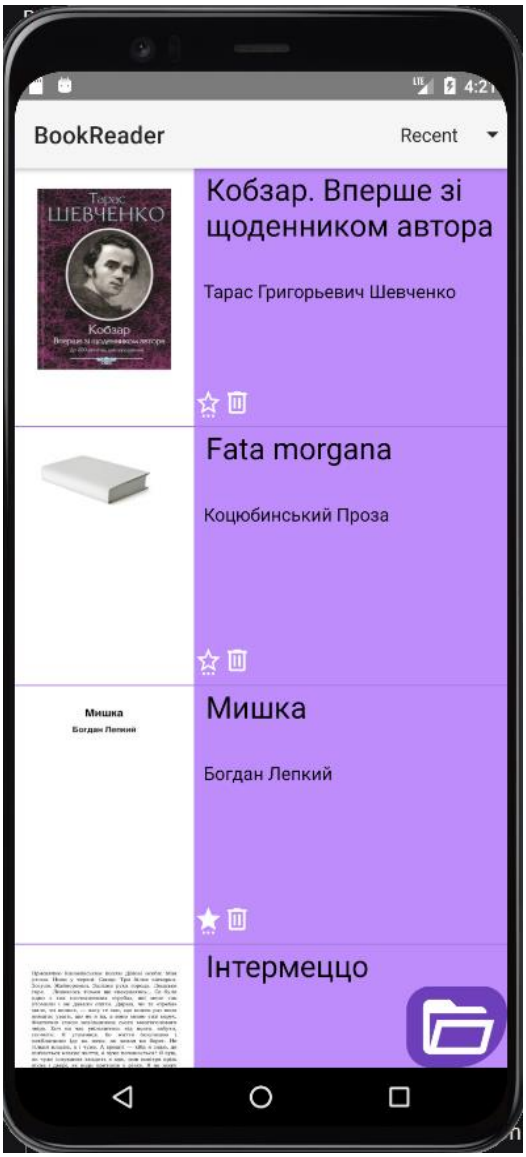


Рисунок 3.1 — Скріншот програми після запуску.

Кнопки зірочка та смітник здійснюють аналогічні операції як і у випадку зі списком книг.

На рисунку 3.4 зображено скріншот екрану актівіту із відображенням тексту книги. У лівому верхньому куті відображається назва книги. Далі зліва направо: кнопка активація якої включає гіперпосилання у тексті, кнопка пошуку по тексту, кнопка вибору розміру шрифта, кнопка відкриття змісту книги, кнопка створення та відкриття закладок.  Активація кнопки зміни розміру шрифта відкриває меню зі списком всіх доступних розмірів кегля шрифта. Натискання кнопки змісту приводить до відкриття активність зі змістом книги, якщо такий наявний. Однократне натискання по кнопці закладок відкриває діалог із введенням назви закладки. Довге натискання по цій кнопці відкриває меню зі списком всіх закладок.

Знизу екрану бачимо слайдер для гортання сторінок документу. Також є можливість гортати сторінки свайпами вліво та вправо. Для збільшення сторінки можна застосувати стандартний жест для зуму.

Загалом розроблений інтерфейс є простим, інтуїтивно зрозумілим та водночас достатньо функціональним.

### 3.3 Програмна структура додатку

Додаток для операційної системи Android написаний з використанням IDE Android Studio на мові програмування Java. У свої роботі для перегляду документів електронних книг додаток використовує бібліотеку MuPDF, яка в загальних рисах описана у першому пункті цього розділу.

Додаток складається з двох модулів — “app” та “mupdf-lib”. Перший модуль відповідає за відкриття файлів книг із файлової системи пристрою, формування списку востаннє переглянутих та улюблених книг. У ньому знаходиться головна активність “MainActivity.java” з якої починається робота програми. Код цієї активності приведений в додатку А. Другий модуль безпосередньо відповідає за

візуалізацію книг для перегляду на екрані пристрою. Головна активність цього модуля називається “DocumentActivity.java”.

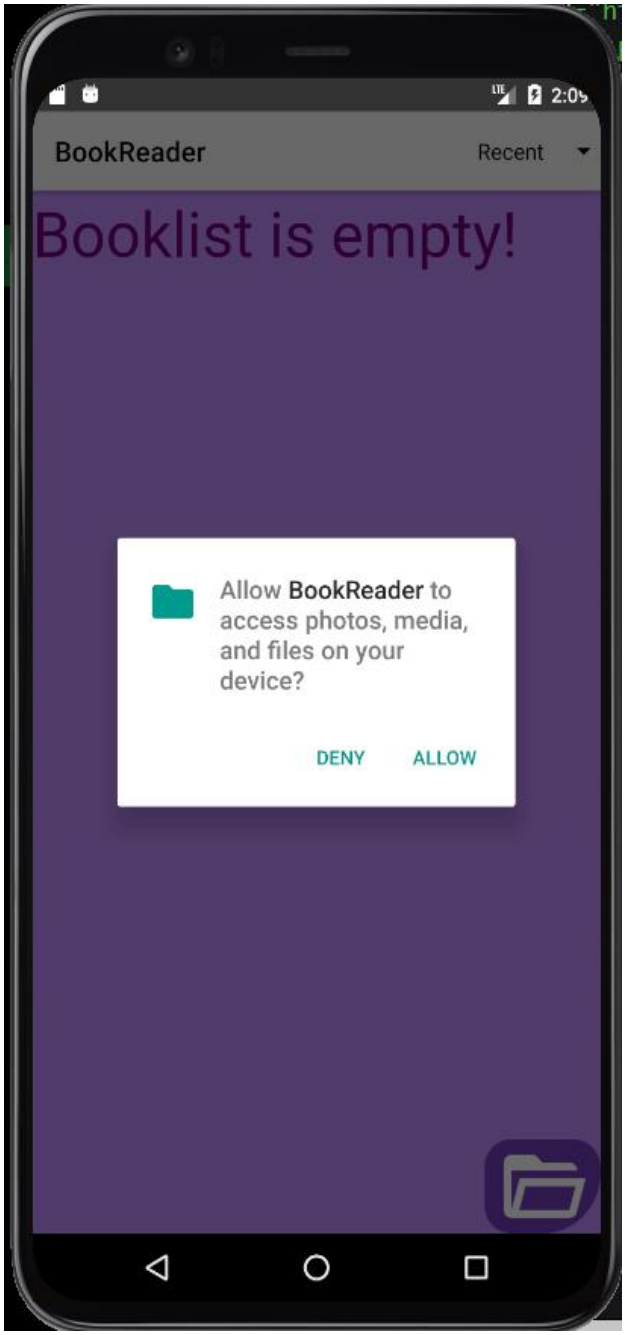


Рисунок 3.2 — Запит дозволу на доступ до пам’яті пристрою.

Для збереження інформації про відкриті книги, їх настройки та закладки у них використовується клас “BookInfo.java”. Код цього класу приведений у додатку Б. Цей клас є частиною модуля “mupdf-lib”. Цей клас надзвичайно важливий для

роботи додатку оскільки він використовується у кожній активності. У ньому реалізовані наступні функції:

- public static BookInfo get(String uriString, SharedPreferences prefsAll) — повертає об’єкт класу “BookInfo” прочитавши його у настройках додатку за посиланням SharedPreferences prefsAll по стрічковому значенню посилання на файл у файловій системі пристрою String uriString;

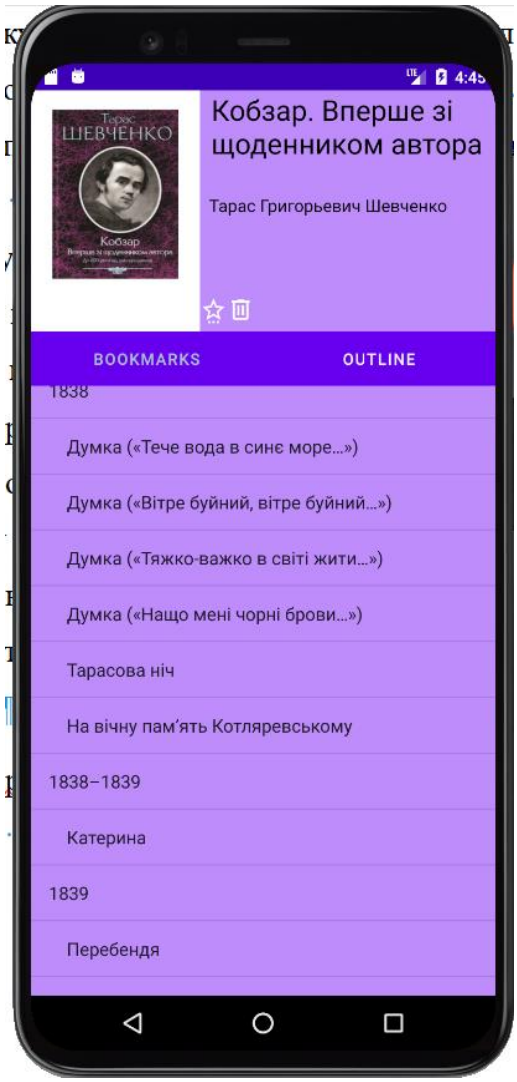


Рисунок 3.3 — Перегляд детальної інформації про книгу.

- public static void save(BookInfo bookInfo**,** SharedPreferences prefsAll) – зберігає об’єкт класу “BookInfo” у настройках додатку за посиланням SharedPreferences prefsAll;

- public void setOpenTime() - записує час відкривання книги;

- public void openBook(Context context) — відкриває поточний файл книги;
- public static void openBook(Context context**,** Uri documentUri) – відкриває файл із заданим посиланням типу Uri documentUri.

За відображення списку книг відповідає клас “BookListAdapter.java”. Цей клас успадковує “ArrayAdapter” із типом даних “BookInfo”. Код цього класу приведений у додатку В.

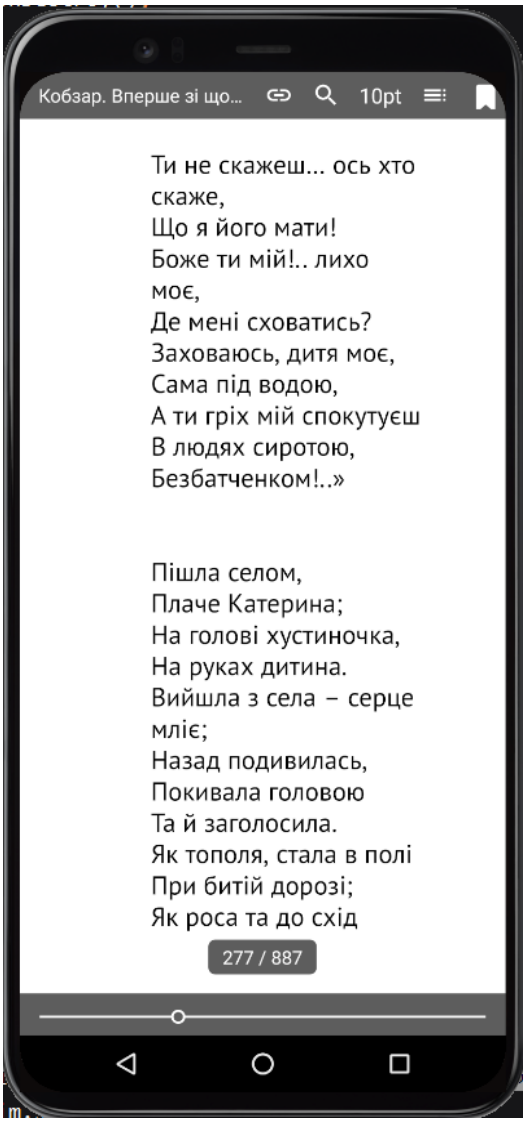


Рисунок 3.4 — Відображення тексту книги.

Для відображення детальної інформації про книгу, а саме записаних у ній закладок та змісту використовується активність під назвою “BookActivity”. Активність запускається методом startActivityForResult із “MainActivity”.

Результатом роботи цієї активності може бути потреба видалити книгу із списку відкритих книг. В разі вимоги користувача відкрити книгу “BookActivity” використовує методи класу “BookInfo” для запуску “DocumentActivity”.

Як було сказано вище, цей клас відповідає за видалення книги з пам’яті додатку, проте саме видалення здійснюється класом “MainActivity”. Також цей клас може додати книгу до списку обраних, або улюблених книг при натисканні користувачем кнопки у вигляді зірочки.

Для побудови перегляду змісту і закладок за допомогою вкладок використовується клас “PageFragment” що успадковує “Fragment”. Цей клас також відповідає за видалення обраних закладок. Також він відкриває книгу на сторінці обраної закладки чи пункту змісту [26].

Під час кожного завершення роботи “DocumentActivity” у настройки програми записується інформація про востаннє відкриту сторінку, а також про зроблені закладки у книзі.

При першому відкритті книги додатком вона одразу починає відображатися за допомогою “DocumentActivity”. Ця активність зберігає у кеш додатку файл зображення з унікальною назвою у якому записано титульну сторінку книги. Також у настройки записується зміст книги, якщо він наявний у файлі. При кожній зміні розміру кегля шрифта сторінки змісту та закладок переписуються для коректного відображення.

### 3.4 Результати тестування додатку

Додаток було протестовано на наступних пристроях:

- Asus Zenfone Max Pro (M1), Android 10;
- Lenovo K6 Note, Android 8;
- Assistant AP-108 Cetus, Android 8;
-One Plus 5, Android 10.

Під час тестування додатку було досліджено відкриття ним заявлених форматів документів електронних книг, а саме “TXT”, “PDF”, “XPS”, “OXPS”,

“CBZ”, “EPUB”, “FB2”, “XML” — проблем під час їх відображення виявлено не було. У всіх випадках книга відкривалася швидко та без затримок.

Наступним етапом було протестовано додавання закладок та їх відкриття. Проблем виявлено не було — закладки справно додавалися та відкривалися як із активності книги так із активності детального перегляду книги. Також адекватною була робота зі змістом книги.

Також було протестовано збереження прогресу читання книги. Кожного разу після закриття документу книга відкривалася на востаннє відкритій сторінці.

Під час тестування списки востаннє відкритих та обраних, або ж улюблених книг адекватно формувалися та відображалися.

Загалом додаток отримав схвальні відгуки від користувачів. Їм сподобався простий та водночас інтуїтивно зрозумілий інтерфейс додатку. Проте було висловлено такі зауваження:

- не зовсім продумані та правильно підібрані кольори інтерфейсу;
- наявність лише англійської мови в інтерфейсі;
- немає можливості створити декілька списків улюблених книг.

Отже, додаток витримав тестування та загалом є готовим до завантаження у магазин додатків Android – Play Market. Проте, у наступних версіях ще є над чим попрацювати.

# ВИСНОВКИ

На ринку додатків для читання електронних книг існує досить багато додатків. Проте вони мають або дуже наповнений і через це складний для користувача інтерфейс, або ж дуже простий і відповідно не надто наповнений потрібним функціоналом інтерфейс. Тобто існує потреба у заповнені прірви між цими двома крайностями.

У ході виконання роботи було проаналізовано низку додатків для операційних систем Android та IOS та виділено найважливіші для користувача їх функції та виділено такі найважливіші функції:

- відкриття файлів з файлової системи пристрою;
- зміна розміру шрифта;
- створення та відкриття закладок;
- пошук по книзі;
- перегляду змісту книги;
- створення списку улюблених книг;
- зручне меню зі списком востаннє відкритих документів.

Ці функції були реалізовані у додатку “BookReader”.

Виходячи із порівняння популярності операційних систем Android та IOS було зроблено висновок про доцільність розробки додатку спершу для операційної системи Android.

Було проаналізовано структуру операційної системи Android з точки зору розробки додатків, а також найпопулярніші мови для цього. Виходячи з цього було зроблено висновок про доцільність використання мови Java та IDE Android Studio.

В результаті було розроблено додаток для читання електронних книг “BookReader”. Додаток отримав простий, інтуїтивно зрозумілий та достатньо функціональний інтерфейс.

В результаті тестування додатку було виявлено що додаток готовий до завантаження у магазин додатків Android Play Market, проте у майбутніх версіях слід виконати такі доробки:

- додати ще хоча б одну мову інтерфейсу;
- надати можливість користувачам створювати декілька списків улюблених книг.

В майбутньому, якщо популярність додатку буде досить високою, буде розглядатися питання про розробку версії для операційної системи IOS, а також додавання можливості завантаження книг з інтернету прямо з додатку. Також доцільним може бути реалізація можливості відкриття файлів типу “DJVU”.

## ПЕРЕЛІК ПОСИЛАНЬ


1. Bazerman, Charles. Literacy and the Organization of Society. A Theory of Literate Action. Anderson, SC, 2013. №2. — С.193. — [Електронний ресурс] – Режим доступу: https://wac.colostate.edu/docs/books/literateaction/v2/theory.pdf.

2. С. М. Соболєва. КЛІПОВЕ МИСЛЕННЯ ЯК СОЦІАЛЬНО-ПСИХОЛОГІЧНИЙ ФЕНОМЕНТА ЙОГО РОЛЬ У НАВЧАЛЬНО-ПІЗНАВАЛЬНІЙ ДІЯЛЬНОСТІСТУДЕНТІВ. Теорія і практика сучасної психології. Запоріжжя, 2019. №3. — С.22-27. — [Електронний ресурс] – Режим доступу: http://tpsp-journal.kpu.zp.ua/archive/3_2019/part_2/18.pdf.

3. Класифікація мобільних додатків : стаття. [Електронний ресурс] – Режим доступу: http://voroninstudio.eu/uk/service/razrabotka-mobilnih-prilozheniy

4. Н.М. Матросова. ПРОФІЛЬ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА ІНТЕРФЕЙС КОРИСТУВАЧА ЯКСКЛАДОВІ МЕТОДИЧНОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМ. Інформаційні технології і засоби навчання. Київ, 2010. №. — С.10.03.2021. — [Електронний ресурс] – Режим доступу: https://core.ac.uk/download/pdf/11084034.pdf.

5. ЯК УБЕРЕГТИ ОЧІ ПРИ РОБОТІ ЗА КОМПЬЮТЕРОМ : стаття. - [Електронний ресурс] – Режим доступу: https://healthclub.com.ua/ua/kak-uberech-glaza-pri-rabote-za-kompyuterom.html

6. 15 best eBook reader apps for Android : стаття. [Електронний ресурс] – Режим доступу: https://www.androidauthority.com/best-ebook-ereader-apps-for-android-170696/

7. ReadEra - book reader pdf, epub, word : стаття. [Електронний ресурс] – Режим доступу: https://play.google.com/store/apps/details?id=org.readera&hl=en&gl=US

8. Reasily - EPUB Reader : стаття. [Електронний ресурс] – Режим доступу: https://play.google.com/store/apps/details?id=com.gmail.jxlab.app.reasily&hl=en&gl=US

9. eBoox - Читалка книг fb2 ePub : стаття. [Електронний ресурс] – Режим доступу: https://apps.apple.com/ru/app/eboox-

%D1%87%D0%B8%D1%82%D0%B0%D0%BB%D0%BA%D0%B0-%D0%BA%D0%BD%D0%B8%D0%B3-fb2-epub/id1489172068
10. Android Version Distribution statistics will now only be available in Android Studio : стаття. [Електронний ресурс] – Режим доступу: https://www.xda-developers.com/android-version-distribution-statistics-android-studio/
11. Android vs iOS Development: Which One is Best for Your App : стаття. [Електронний ресурс] – Режим доступу: https://ncube.com/blog/android-vs-ios-development
12. Introduction to Kotlin : сайт. [Електронний ресурс] – Режим доступу: https://medium.com/@rohinideshmane.21/introduction-to-kotlin-5f39b31610e0
13. Android Studio : стаття. [Електронний ресурс] – Режим доступу: https://developer.android.com/studio/features
14. О. В. Шматко, А. О. Поляков, В. М. Федорченко. АНАЛІЗ МЕТОДІВ І ТЕХНОЛОГІЙ РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ ДЛЯ ПЛАТФОРМИ ANDROID. ЧАСТИНА 2 : навч. посіб.. Харків : НТУ «ХПІ», 2018. 284 с.
15. ARTist: The Android Runtime Instrumentation and Security Toolkit : стаття. [Електронний ресурс] – Режим доступу: https://core.ac.uk/download/pdf/249324963.pdf
16. Guide to app architecture : [Електронний ресурс] – Режим доступу: https://developer.android.com/jetpack/guide
17. AndroidNative Development Kit (NDK) : стаття. [Електронний ресурс] – Режим доступу:
https://www3.ntu.edu.sg/home/ehchua/programming/android/Android_NDK.html
18. Jeff Friesen. Learn Java for Android Development : . New York : Apress Media, 2013. с.
19. Java. Progopedia : стаття. [Електронний ресурс] – Режим доступу: http://progopedia.com/language/java/
20. Busy Coder’s Guide to Android Development : стаття. [Електронний ресурс] – Режим доступу: https://commonsware.com/Android/Android_3-6-CC.pdf

21. Architecture of Android Apps : стаття. [Електронний ресурс] – Режим доступу: https://guides.codepath.com/android/Architecture-of-Android-Apps
22. Introduction to Activities : стаття. [Електронний ресурс] – Режим доступу: https://developer.android.com/guide/components/activities/intro-activities
23. AndroidManifest.xml : стаття. [Електронний ресурс] – Режим доступу: https://medium.com/android-news/androidmanifest-xml-4d5a3e821f91
24. MuPDF Overview : стаття. [Електронний ресурс] – Режим доступу: https://mupdf.com/index.html
25. MuPDF Explored : стаття. [Електронний ресурс] – Режим доступу: http://priede.bf.lu.lv/ftp/pub/RakstuDarbi/pdf/MuPDF/mupdf_explored.pdf
26. Re-using layouts with <include/> : стаття. [Електронний ресурс] – Режим доступу: https://developer.android.com/training/improving-layouts/reusing-layouts

# ДОДАТОК А

```
public class MainActivity extends AppCompatActivity {
    private static final int REQUEST_PERMISSION = 101;
    private static Uri selectedfile = null;
    private BookListAdapter bookListAdapter;
    private List<Uri> urifiles;
    private List<BookInfo> booksToOpen;
    private boolean favorite_view = false;
    SharedPreferences prefsAll;
    private final int BOOK_DETAILED_REQUEST = 20;
    public void startBookViewActivity(BookInfo bookInfo) {
        Intent intent = new Intent(this, BookActivity.class);
        intent.putExtra("uriString", bookInfo.uriString);
        startActivityForResult(intent, BOOK_DETAILED_REQUEST);
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        prefsAll = PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
        makeBookList(favorite_view);

        ImageButton mOpenFileButton = (ImageButton) findViewById(R.id.openFileButton);

        mOpenFileButton.setOnClickListener(new View.OnClickListener() {
            @RequiresApi(api = Build.VERSION_CODES.KITKAT)
            @Override
            public void onClick(View v) {
                if (ContextCompat.checkSelfPermission(
                        MainActivity.this, Manifest.permission.READ_EXTERNAL_STORAGE) ==
                        PackageManager.PERMISSION_GRANTED) {
                    // You can use the API that requires the permission.
                    openFileExplorer();                } else {
                    ActivityCompat.requestPermissions(MainActivity.this,
                            new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},
REQUEST_PERMISSION);
                }
            }
        });
    }
    private void openFileExplorer() {
        Intent intent = new Intent()
                .setType("*/*")
                .setAction(Intent.ACTION_OPEN_DOCUMENT);
        startActivityForResult(Intent.createChooser(intent, "Select a file")
                .addFlags(Intent.FLAG_GRANT_PERSISTABLE_URI_PERMISSION)
                .addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION)
                .addFlags(Intent.FLAG_GRANT_WRITE_URI_PERMISSION)
                .addFlags(Intent.FLAG_GRANT_PREFIX_URI_PERMISSION), 123);
    }
    private void makeBookList(boolean favorite) {
        Gson gson = new Gson();
        String tableJSON = prefsAll.getString("booksInfo", "");
        Hashtable<String, String> booksTable = gson.fromJson(tableJSON, Hashtable.class);
        if (booksTable == null) {
            booksTable = new Hashtable<>();
        }
        ListView listView = (ListView) findViewById(R.id.bookList);
        listView.setEmptyView(findViewById(R.id.empty_list_item));
```

```
        ArrayList<BookInfo> books = getBookArrayFromHashTable(booksTable, favorite_view);
        if (books == null || books.size() == 0) {
            listView.setVisibility(View.INVISIBLE);
        }
        bookListAdapter = new BookListAdapter(MainActivity.this, books);
        listView.setAdapter(bookListAdapter);
        listView.setOnItemClickListener((adapterView, view, position, l) -> {
            startBookViewActivity(booksToOpen.get(position));
        });

    }
    private ArrayList<BookInfo> getBookArrayFromHashTable(Hashtable<String, String>
booksTable, boolean favorite) {
        ArrayList<BookInfo> books = new ArrayList<>();
        if (booksTable != null) {
            if (urifiles == null) {
                urifiles = new ArrayList<>();
            } else {
                urifiles.clear();
            }
            Set<String> keys = booksTable.keySet();
            Gson gson = new Gson();
            for (String key : keys) {
                String bookInfoString = String.valueOf(booksTable.get(key));
                BookInfo bookInfo = gson.fromJson(bookInfoString, BookInfo.class);
                if (favorite) if (!bookInfo.favorite) continue;
                urifiles.add(Uri.parse(bookInfo.uriString));
                books.add(bookInfo);
            }
            Collections.sort(books, (m1, m2) -> m2.openTime.compareTo(m1.openTime));
            booksToOpen = books;
        }
        return books;
    }
    public void favoriteCheckBoxClick(View v) {
        LinearLayout parentRow = (LinearLayout) v.getParent();
        CheckBox mFavoriteCheckBoxCurrent = (CheckBox)
parentRow.findViewById(R.id.favoriteCheckBox);
        ConstraintLayout parentRowConstraint = (ConstraintLayout) (parentRow).getParent();
        TextView mTitleTextView = (TextView)
parentRowConstraint.findViewById(R.id.titleTextView);
        BookInfo bookInfo = BookInfo.get(String.valueOf(mTitleTextView.getTag()),
prefsAll);
        bookInfo.favorite = !mFavoriteCheckBoxCurrent.isChecked();
        BookInfo.save(bookInfo, prefsAll);
    }

    public void trashButtonClick(View v) {
        ConstraintLayout parentRow = (ConstraintLayout) ((LinearLayout)
v.getParent()).getParent();
        TextView mTitleTextView = (TextView) parentRow.findViewById(R.id.titleTextView);
        new AlertDialog.Builder(this)
                .setTitle("Delete item")
                .setMessage("Are you sure to delete " + mTitleTextView.getText() + "?")
                .setIcon(android.R.drawable.ic_dialog_alert)
                .setPositiveButton(android.R.string.yes, (DialogInterface.OnClickListener)
(dialog, whichButton) -> {
                    Toast.makeText(MainActivity.this, "Deleted!",
Toast.LENGTH_SHORT).show();
                    Hashtable<String, String> booksTable = getBooksTableFromSet();
                    deleteBookItemFromList(booksTable,
String.valueOf(mTitleTextView.getTag()));
```

```
                    saveBooksTableToSet(booksTable);
                })
                .setNegativeButton(android.R.string.no, null).show();
    }
    private Hashtable<String, String> getBooksTableFromSet() {
        SharedPreferences prefs =
PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
        Gson gson = new Gson();
        String tableJSON = prefs.getString("booksInfo", "");
        Hashtable<String, String> booksTable = gson.fromJson(tableJSON, Hashtable.class);

        if (booksTable == null) booksTable = new Hashtable<>();

        return booksTable;
    }
    private void saveBooksTableToSet(Hashtable<String, String> booksTable) {
        SharedPreferences prefs =
PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
        Gson gson = new Gson();

        SharedPreferences.Editor prefsEditor = prefs.edit();
        String json = gson.toJson(booksTable);
        prefsEditor.putString("booksInfo", json);
        prefsEditor.commit();
    }
    private void deleteBookItemFromList(Hashtable<String, String> booksTable, String key)
{
        booksTable.remove(key);
        refreshBookItemList(booksTable);
    }
    private void refreshBookItemList(Hashtable<String, String> booksTable) {
        ArrayList<BookInfo> books = getBookArrayFromHashTable(booksTable, favorite_view);
        if (bookListAdapter == null) bookListAdapter = new BookListAdapter(this, books);
        bookListAdapter.clear();
        bookListAdapter.addAll(books);
        bookListAdapter.notifyDataSetChanged();
    }
    public void onRequestPermissionsResult(int requestCode,
                                           String permissions[], int[] grantResults) {
        if (grantResults.length > 0
                && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            openFileExplorer();
        } else {
            Toast.makeText(MainActivity.this, "Permission denied to read your External
storage", Toast.LENGTH_SHORT).show();
        }
        return;
    }
    @RequiresApi(api = Build.VERSION_CODES.KITKAT)
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode == RESULT_OK) {
            switch (requestCode) {
                case BOOK_DETAILED_REQUEST: {
                    if (Boolean.valueOf(data.getBooleanExtra("delete", false))) {
                        String uriString = data.getStringExtra("uriStringBook");
                        Toast.makeText(MainActivity.this, "Deleted!",
Toast.LENGTH_SHORT).show();
                        Hashtable<String, String> booksTable = getBooksTableFromSet();

                        deleteBookItemFromList(booksTable, String.valueOf(uriString));
```

```
                        saveBooksTableToSet(booksTable);
                    }
                }
                break;
                case 123: {
                    selectedfile = data.getData(); //The uri with the location of the file
                    Uri uri = Uri.parse(selectedfile.toString());
                    ContentResolver contentResolver = getContentResolver();
                 contentResolver.takePersistableUriPermission(uri, 0);
BookInfo.openBook(this, uri);
                }
                break;
            }
        }
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
getMenuInflater().inflate(R.menu.android_action_bar_spinner_menu, menu);
        MenuItem item = menu.findItem(R.id.spinner);
        Spinner spinner = (Spinner) MenuItemCompat.getActionView(item);
        final List<String> items = new ArrayList<>();
        items.add("Recent");
        items.add("Favorite");
        ArrayAdapter<CharSequence> adapter = new ArrayAdapter(this,
android.R.layout.simple_spinner_item, items);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position,
long id) {
                favorite_view = position == 1 ? true : false;
                refreshBookItemList(getBooksTableFromSet());
            }
            @Override
            public void onNothingSelected(AdapterView<?> parent) {
            }
        });
        return true;
    }
    @Override
    protected void onResume() {
        super.onResume();
        refreshBookItemList(getBooksTableFromSet());
    }
    @Override
    protected void onPause() {
        super.onPause();
    }
}
```

# ДОДАТОК Б

```
public class BookInfo {
    public String title;
    public String format;
    public String author;
    public File imgCoverFile;
    public String uriString;
    public int mLayoutEM;
    public boolean favorite;
    public Date openTime;
    public List<Bookmark> bookmarks;
    public ArrayList<OutlineActivity.Item> mFlatOutline;
    public int pageToOpen;
    public void setOpenTime() {
        openTime.setTime(System.currentTimeMillis());
    }
    public BookInfo(String title, String format, String author, File imgCoverFile, String uriString,
int mLayoutEM, int pageToOpen) {
        this.title = title;
        this.format = format;
        this.author = author;
        this.imgCoverFile = imgCoverFile;
        this.uriString = uriString;
        this.mLayoutEM = mLayoutEM;
        openTime = new Date(System.currentTimeMillis());
        bookmarks = new ArrayList<>();
        this.pageToOpen = pageToOpen;
    }
    public static BookInfo get(String uriString, SharedPreferences prefsAll) {
        Gson gson = new Gson();
        Hashtable<String, String> booksInfoTable = gson.fromJson(prefsAll.getString("booksInfo",
""), Hashtable.class);
        if (booksInfoTable == null) {
            booksInfoTable = new Hashtable<String, String>();
        }
        BookInfo bookInfo = gson.fromJson(booksInfoTable.get(uriString), BookInfo.class);
        return bookInfo;
    }
    public static void save(BookInfo bookInfo, SharedPreferences prefsAll) {
        Gson gson = new Gson();
        Hashtable<String, String> booksInfoTable = gson.fromJson(prefsAll.getString("booksInfo",
""), Hashtable.class);
        if (booksInfoTable == null) {
            booksInfoTable = new Hashtable<String, String>();
        }
        booksInfoTable.remove(bookInfo.uriString);
        booksInfoTable.put(bookInfo.uriString, gson.toJson(bookInfo));
        SharedPreferences.Editor prefsEditor = prefsAll.edit();
        String json = gson.toJson(booksInfoTable);
        prefsEditor.putString("booksInfo", json);
        prefsEditor.commit();
    }
    public void openBook(Context context) {startMuPDFActivity(context, Uri.parse(uriString));
    }
    public static void openBook(Context context, Uri documentUri) {startMuPDFActivity(context,
documentUri);
    }
    private static void startMuPDFActivity(Context context, Uri documentUri) {
        Intent intent = new Intent(context, DocumentActivity.class);
        intent.setAction(Intent.ACTION_VIEW);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        intent.setData(documentUri);
        context.startActivity(intent);
    }
}
```

# ДОДАТОК В

```
public class BookListAdapter extends ArrayAdapter<BookInfo> {
    private static final String LOG_TAG = BookListAdapter.class.getSimpleName();

    public BookListAdapter(Activity context, ArrayList<BookInfo> books) {
        super(context, 0, books);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View listItemView = convertView;
        if (listItemView == null) {
            listItemView = LayoutInflater.from(getContext()).inflate(
                    R.layout.book_element, parent, false);
        }

        BookInfo book = getItem(position);

        ImageView mCoverImageView = (ImageView)
listItemView.findViewById(R.id.coverImageView);
        //Bitmap myBitmap =
BitmapFactory.decodeFile(books.get(position).imgCoverFile.getAbsolutePath());

mCoverImageView.setImageBitmap(BitmapFactory.decodeFile(book.imgCoverFile.getAbsolutePath(
)));

        TextView mTitleTextView = (TextView)
listItemView.findViewById(R.id.titleTextView);
        mTitleTextView.setText(book.title);
        mTitleTextView.setTag(book.uriString);

        TextView mAuthorTextView = (TextView)
listItemView.findViewById(R.id.authorTextView);
        mAuthorTextView.setText(book.author);

        CheckBox mFavoriteCheckBox = listItemView.findViewById(R.id.favoriteCheckBox);
        mFavoriteCheckBox.setChecked(!book.favorite);

        return listItemView;
    }
}
```

**ДОДАТОК Г**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**


Розробка мобільного(Android) додатку для читання електронних книг на мові програмування Java

Виконав студент 4 курсу
групи ПД-41
Волчанський О.С.
Керівник роботи
Дібрівний О.А.

Київ – 2021

# МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи**

Розробити мобільний Android-додаток для читання електронних книг.

- **Об’єкт дослідження**
- Існуючі додатки для читання електронних книг.
- **Предмет дослідження**
- Найбільш важливі функції мобільних додатків для читання електронних книг.

# АНАЛОГИ

## ReadEra - book reader pdf, epub, word

Додаток не містить реклами, є абсолютно безкоштовним та дозволяє читати й переглядати книги в форматах PDF, EPUB, Microsoft Word (DOC, DOCX, RTF), Kindle (MOBI, AZW3), DJVU, FB2, TXT, ODT і CHM. Також додаток дозволяє читати книги запаковані в архів формату ZIP без необхідності розпаковування.

## Reasily - EPUB Reader

Додаток має можливість відкривати лише один формат файлу EPUB при цьому повністю зберігаючи авторське форматування. Також він дає змогу переглядати елементи книги створені з допомогою мови математичної розмітки MathML (*Mathematical Markup Language*)



# Порівняння

| | ReadEra | Reasily |
|---|---|---|
| Підтримка розмітки MathML | ✗ | ✓ |
| Немає реклами | ✓ | ✓ |
| Мульти-документний режим | ✗ | ✓ |
| Перекладач | ✗ | ✓ |
| Кількість форматів | Більше 5 | Один - EPUB |
| Читання «ZIP» | ✓ | ✗ |

# ТЕХНІЧНІ ЗАВДАННЯ

*1. Розробити концепцію мобільного додатку для читання електронних книг;*

*2. Забезпечити підтримку форматів: “TXT”, “PDF”, “XPS”, “OXPS”, “CBZ”, “EPUB”, “FB2”, “XML”;*

*3. Забезпечити додаток наступним функціоналом:*

- *а) створення закладок у тексті;*
- *б) перегляд змісту книги;*
- *в) зміна розміру шрифту;*
- *г) створення списку улюблених книг;*
- *д) пошук по тексту книги;*
- *е) формування списку востаннє відкритих книг;*



# ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ


Android Studio

інтегроване середовище розробки (IDE) для платформи Android, представлене 16 травня 2013 року на конференції Google I/O менеджером по продукції корпорації Google — Еллі Паверс.
8 грудня 2014 року компанія Google випустила перший стабільний реліз Android Studio 1.0

# БІБЛІОТЕКА MuPDF


Це легка бібліотека для перегляду PDF, XPS та інших найбільш розповсюджених форматів електронних книг.

General Public License це загальна публічна ліцензія GNU або Загальна громадська ліцензія GNU — одна з найпопулярніших ліцензій на вільне програмне забезпечення, створена Річардом Столменом для проєкту GNU.



# МОДУЛІ ПРОГРАМИ

Модуль «app» відповідає за відкриття файлів книг із файлової системи пристрою, формування списку востаннє переглянутих та улюблених книг. У ньому знаходиться головна активність “MainActivity.java” з якої починається робота програми.

Модуль «mupdf-lib» безпосередньо відповідає за візуалізацію книг для перегляду на екрані пристрою. Головна активність цього модуля називається “DocumentActivity.java”.

## КЛАС *BookInfo*

*Клас "BookInfo.java" відповідає за збереження інформації про відкриті книги, їх настройки та закладки у них. Код цього класу приведений у додатку Б. Цей клас є частиною модуля "mupdf-lib". Цей клас надзвичайно важливий для роботи додатку оскільки він використовується у кожній активності. У ньому реалізовані наступні функції:*

*- public static BookInfo get(String uriString, SharedPreferences prefsAll) — повертає об'єкт класу "BookInfo" прочитавши його у настройках додатку за посиланням SharedPreferences prefsAll по стрічковому значенню посилання на файл у файловій системі пристрою String uriString;*

*- public static void save(BookInfo bookInfo, SharedPreferences prefsAll) – зберігає об'єкт класу "BookInfo" у настройках додатку за посиланням SharedPreferences prefsAll;*

*- public void setOpenTime() - записує час відкривання книги;*

*- public void openBook(Context context) — відкриває поточний файл книги;*

*- public static void openBook(Context context, Uri documentUri) – відкриває файл із заданим посиланням типу Uri documentUri.*



# ВИСНОВКИ

Найважливішими функціями додатків для читання електронних книг є:

- відкриття файлів з файлової системи пристрою;
- зміна розміру шрифта;
- створення та відкриття закладок;
- пошук по книзі;
- перегляду змісту книги;
- створення списку улюблених книг;
- зручне меню зі списком востаннє відкритих документів;

Наступними етапами роботи над додатком є:

- додати ще хоча б одну мову інтерфейсу;
- надати можливість користувачам створювати декілька списків улюблених книг;
- створення версії для операційної системи IOS;