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 ДЛЯ ЧИТАННЯ ЕЛЕКТРОННИХ КНИГ МОВОЮ PYTHON»

Виконав: студент 4 курсу, групи ПД–44
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)

Слободянюк О.І.
(прізвище та ініціали)

Керівник Трінтіна Н.А.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Київ – 2022

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.

“____” _____2022 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

СЛОБОДЯНЮКА ОЛЕГА ІГОРОВИЧА

(прізвище, ім‘я, по батькові)

1.Тема роботи: «Розробка мобільного додатку для читання електронних книг мовою Python”_____

Керівник роботи____Трінтіна Н.А.__ к.т.н., доцент_____,

(прізвище, ім‘я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «16» лютого 2022 року №22.

2. Строк подання студентом роботи__ «6» червня 2022 року_____

3. Вхідні дані до роботи:

3.1 Науково-технічна література з питань, пов’язаних з програмним забезпеченням щодо розробки додатків.

3.2 Інтегроване середовище розробки Android Studio для роботи з платформою Android.

3.3 Офіційна документація Kivy.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).

4.1 Теоретичні основи розробки мобільного додатку для читання електронних книг мовою Python

4.2 Розробка мобільного додатка для читання електронних книг

4.3 Техніко–економічне обґрунтування

5. Дата видачі завдання_ «11» квітня 2022 _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	16.02.22 - 12.04.22	Виконано
2	Аналіз концепцій та класифікацій мобільних додатків	12.04.22 - 15.04.22	Виконано
3	Аналіз сучасних засобів проектування та розробки програмного забезпечення мовою Python.	15.04.22 - 18.04.22	Виконано
4	Розробка мобільного додатку для читання електронних книг мовою програмування Python	21.04.22 - 24.04.22	Виконано
5	Тестування мобільного додатку для читання електронних книг.	24.04.22 - 27.04.22	Виконано
6	Техніко–економічне обґрунтування	30.04.22 – 1.05.22	Виконано
7	Вступ, висновки, реферат	01.05.22 – 09.05.22	Виконано
8	Розробка обов‘язкових демонстраційних креслень	10.05.22 – 13.05.22	Виконано
9	Попередній захист роботи	02.06.22	

10	Здача роботи	06.06.22	

Студент _____ _____Слободянюк О.І.

(підпис) (прізвище та ініціали)

Керівник роботи _____ _____ Трінтіна Н.А.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 60 с., 5 табл., 22 рис., 1 дод., 21 джерел.

РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ЧИТАННЯ ЕЛЕКТРОННИХ КНИГ МОВОЮ PYTHON

Об`єкт дослідження – електронні книжки

Предмет дослідження – розробка мобільного додатку для читання електронних книг.

Мета роботи – розробка мобільного додатку для читання електронних книг за допомогою мови програмування Python.

Методи дослідження – аналіз, синтез, индукция, классификация, наблюдение, сравнение, эксперимент..

У роботі створено мобільний додаток для читання електронних книг. Програма підтримує формат EPUB і дозволяє читати книги, переглядати інформацію про них. На відміну від аналочичних програм, програма дозволяє задавати настройки відображення для кожної з книг окремо, а також задавати загальні налаштування.

Програма розроблена мовою Python за допомогою фреймворку Kivy, як середовище розробки використовувалася Android Studio.

Зміст

ВСТУП

Сьогодні мобільний і швидкий доступ до інформації віддають перевагу дедалі більше людей. З кожним роком відсоток користувачів мобільних версій сайтів зростає, і, отже, комп'ютерні версії стають менш популярними.

Безліч програмних продуктів в епоху цифрових технологій, розробляється для мобільних телефонів. Більшість останніх передових моделей електронних пристроїв практично не поступаються за функціоналом та характеристиками комп'ютерів. Однією з галузей, яку повністю змінила епоха сматрфонів, є електронні книги. У міру поширення мобільних телефонів і планшетів читання книг на них стає зручнішим за рахунок мобільності, синхронізації з віддаленими серверами та величезної кількості книг, які можна зберігати в пам'яті пристрою.

На сьогоднішній день, мобільні телефони часто класифікую за застосовуваними операційними системами: Android, iOS та інші. ОС Android є однією з найпростіших і одночасно комплексних платформ.

Метою випускної кваліфікаційної роботи є розробка мобільного додатку для читання електронних книг за допомогою мови програмування Python .

Для реалізації поставленої мети необхідно вирішити наступні завдання:

1. Знайомство з науково-методичною літературою на тему дослідження.

2. Аналіз концепцій та класифікацій мобільних додатків.

3. Аналіз сучасних засобів проектування та розробки програмного забезпечення мовою Python.

4. Розробка мобільного додатка для читання електронних книг мовою програмування Python .

5. Тестування мобільного додатка для читання електронних книг.

Об'єктом дослідження є електронні книжки.

Предметом дослідження є розробка мобільного додатка для читання електронних книг

1. ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ МОБІЛЬНОГО ДОДАТКУ ДЛЯ ЧИТАННЯ ЕЛЕКТРОННИХ КНИГ МОВОЮ PYTHON

1.1 Огляд розвитку та сучасний стан електронних книг

Електронна книга є книжковим виданням, доступним у цифровій формі, що складається з тексту, зображень або того і іншого, яке читається на плоскопанельному дисплеї комп'ютера або іншого електронного пристрою. [1] Хоча деякі електронні книги іноді визначаються як "електронна версія друкованої книги", вони можуть не мати друкованого еквівалента. Електронні книги можна читати на спеціальних пристроях для читання електронних книг, а також на будь-якому комп'ютерному пристрої з керованим екраном перегляду, включаючи настільні комп'ютери, ноутбуки, планшети та смартфони (рис. 1.1).

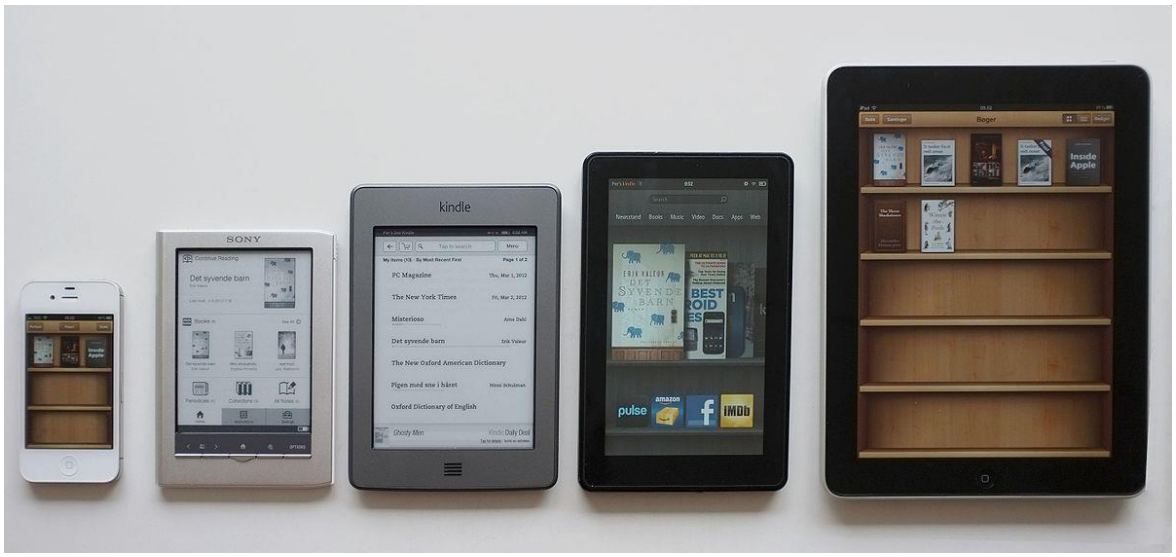


Рисунок 1.1 – Приклади пристроїв для читання електронних книг

У 2000-х роках спостерігалася тенденція до переміщення продажів друкованих та електронних книг в Інтернет, де читачі купують традиційні паперові книги та електронні книги на веб-сайтах із використанням систем електронної

комерції. До початку 2010-х років електронні книги почали обганяти книги у твердій палітурці за загальною кількістю публікацій у США.

Основними причинами, з яких люди купують електронні книги, є нижчі ціни, підвищений комфорт (оскільки вони можуть купувати вдома або в дорозі за допомогою мобільних пристроїв) та ширший вибір найменувань. У разі електронних книг електронні закладки спрощують пошук посилань, а пристрої для читання електронних книг можуть дозволити користувачеві анотувати сторінки[2]. З іншого боку, для книг з програмування можна копіювати приклади коду. Кількість використання електронних книг у США зростає; до 2014р. електронні книги читали 28% дорослих порівняно з 23% у 2013 р.; а до 2014р. 50% дорослих американців мали електронні книги або планшети, порівняно з 30% власників таких пристроїв у 2013 р. [3].

Переваги електронних книг.

- У просторі, який займає фізична книга порівнянного розміру, електронна книга може містити тисячі електронних книг, обмежених лише обсягом пам'яті. Залежно від пристрою електронна книга може читатись при слабкому освітленні або навіть у повній темряві. Багато електронних книг мають вбудований джерело світла, можуть збільшувати або змінювати шрифти, використовувати програмне забезпечення для перетворення тексту в мову для читання тексту вголос для людей з вадами зору, людей похилого віку або людей з дислексією або просто для зручності. Крім того, електронні книги дозволяють читачам шукати слова або відразу знаходити додаткову інформацію на тему за допомогою онлайн-словника. Amazon повідомляє, що 85% читачів електронних книг шукають слово під час читання [4].

- Для виробництва друкованих книжок потрібно втричі більше сировини і в 78 разів більше води проти електронними книгами. Дослідження 2017 року показало, що навіть з урахуванням викидів, що виникають при виробництві пристрою для читання електронних книг, заміна друкованих книг на рік

призводить до менших викидів парникових газів, ніж друк. Хоча електронна книга коштує дорожче, ніж більшість окремих книг, електронні книги можуть мати нижчу вартість, ніж паперові книги. Крім того, чисельні електронні книги доступні в Інтернеті безкоштовно на таких сайтах, як Project Gutenberg.

- Залежно від можливого керування цифровими правами електронні книги (на відміну від фізичних книг) можуть створювати резервні копії та відновлюватися в разі втрати або пошкодження пристрою, на якому вони зберігаються, нова копія може бути завантажена без додаткових витрат з боку споживача. Читачі можуть синхронізувати своє місце читання, виділення та закладки на кількох пристроях.

Недоліки електронних книг.

- при читанні електронних книг може не дотримуватися конфіденційності користувача, наприклад, Amazon знає особистість користувача, що користувач читає, або закінчивши користувач книгу, на якій сторінці знаходиться користувач і т.д. [5]

- Одним із перешкод на шляху широкого впровадження електронних книг є те, що більшість людей цінує друковану книгу як сам предмет, включаючи такі аспекти, як текстура, запах, вага та зовнішній вигляд на поліці. Друковані книги також вважаються цінними культурними цінностями та символами ліберального освіти та гуманітарних наук.

- Існують деякі проблеми з читабельністю та зручністю використання, які необхідно вирішувати видавцям та розробникам програмного забезпечення.

Розглянемо найпопулярніші формати електронних книг.

EPUB

EPUB – це найпоширеніший формат файлів електронних книг. Спочатку розроблений Міжнародним форумом цифрових публікацій (який є частиною консорціуму World Wide Web), він замінив старий формат відкритих електронних книг (OEB) у 2007 році [6].

Оскільки EPUB є безкоштовним для використання, відкритим стандартом та незалежним від постачальників, він став найпоширенішим форматом електронних книг. Хоча це не часто зустрічається, він може навіть підтримувати кольорові зображення, графіку SVG, інтерактивні елементи та відео.

MOBI

Як і EPUB, формат MOBI є спадкоємцем старого формату OEB. Французька компанія Mobipocket відокремила його у 2000 році, і він ліг в основу її програмного забезпечення Mobipocket Reader.

Між EPUB та MOBI є кілька ключових відмінностей. MOBI - це не відкритий стандарт і, отже, не є загальнодоступним. Він також не може підтримувати звук або відео.

AZW та AZW3

Розширення AZW та AZW3 – це два формати електронних книг Amazon. AZW - старший із двох; він дебютував разом із першим Kindle ще у 2007 році. AZW3 з'явився у 2011 році з випуском пристрою для читання Kindle Fire [7].

Оскільки AZW є пропрієтарним, він не так широко підтримується електронними книгами, як EPUB та MOBI. Звичайно, всі продукти Amazon Kindle можуть читати цей формат, але інші популярні пристрої, такі як електронні книги Nook та Kobo, не можуть.

IBA

Іншим поширеним пропрієтарним форматом електронних книг, з яким ви, ймовірно, зіткнетеся, є IBA. Це формат, який використовується для книг, створених у програмі Apple iBooks Author.

Технічно формат дуже подібний до EPUB. Однак для його роботи потрібен код віджету в програмі Apple Books, тому його не можна універсально читати на всіх електронних книгах. Формат підтримує відео, звук, зображення та інтерактивні елементи.

PDF

Найпопулярнішим форматом електронних книг є PDF. Через поширення формату в Інтернеті PDF-файли стали популярним способом передачі електронних книг.

Його великим недоліком є відсутність механізму перекомпонування. Перекомпонування — це термін, який використовується для опису того, коли файл може адаптувати своє подання відповідно до розміру екрана або налаштувань, вибраних користувачем.

Усі спеціалізовані формати електронних книг пропонують перекомпонування на основі послідовності об'єктів у потоці контенту. Формат PDF може обійти відсутність регулярного перекомпонування, використовуючи теги для визначення базової структури документа. Однак PDF-файли з тегами, як і раніше, погано підтримуються програмами для читання електронних книг.

1.2. Аналіз існуючих мобільних програм для читання електронних книг

Далі розглянемо найпопулярніші додатки для читання електронних книг. Серед п'яти найбільш популярних рішень можна виділити: Amazon Kindle, Google Play Books, Apple Books, Kobo Books, FBReader [8].

Додаток Amazon Kindle

Для роботи цієї програми не потрібне Kindle. Програма підтримує безліч різних платформ, включаючи комп'ютери Windows та Mac, а також мобільні пристрої iOS, iPadOS та Android. Воно може працювати з книгами з Amazon, а також іншими форматами через додаток Libby.

Під час читання можна змінити колір, шрифт, розмір тексту, інтервал між рядками та інші атрибути. Є можливість робити позначки та анотувати уривки тексту, додавати поточну сторінку до закладок та шукати певний текст. За необхідності можна виділити незнайоме слово, і з'явиться словарь чи стаття у Вікіпедії, щоб описати його (рис. 1.2).

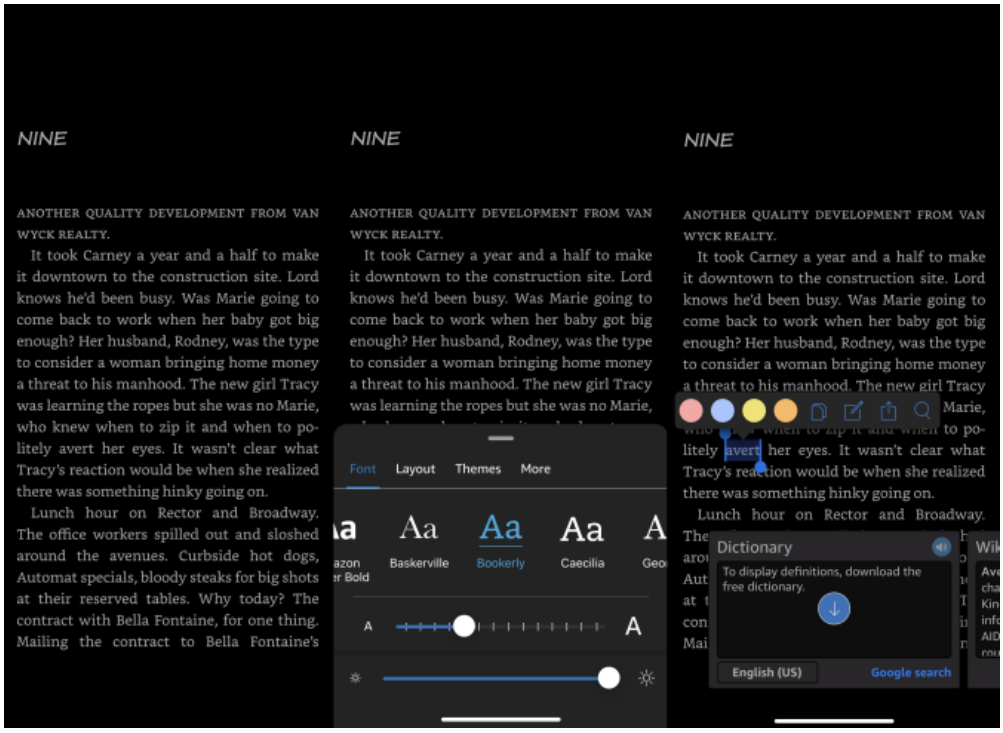


Рисунок 1.2 – Інтерфейс Amazon Kindle

Google Play Books

Програма Google Play Books доступна для пристроїв iPhone та iPad, Android та для браузера Chrome. Ця програма приймає будь-які книги, які завантажені з Google Play, а також PDF-файли та інші формати, завантажені зі сторонніх сайтів.

Існує можливість змінювати атрибути тексту та кольору, переглядати вихідні сторінки книги, додавати закладки та навіть прослуховувати читання книги. На екрані налаштувань можна включати темний режим, використовувати автономний словарь та використовувати клавішу гучності для перегортання сторінок (рис. 1.3).

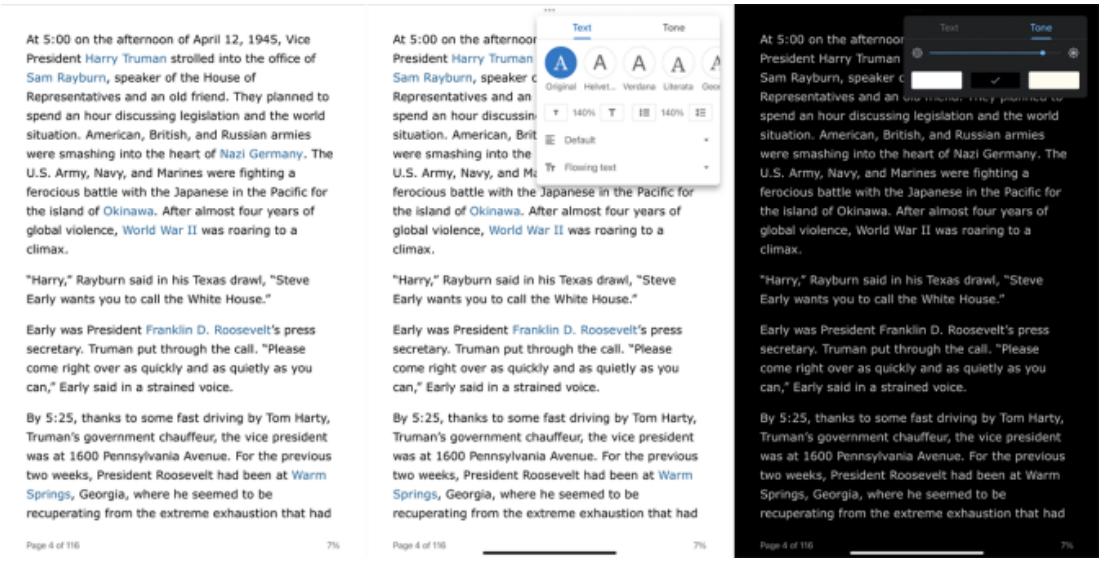


Рисунок 1.3 – Інтерфейс Google Play Books

Apple Books

Програма Apple Books, розроблена для пристроїв Apple, є вбудованою програмою для читання електронних та аудіокниг, завантажених з книгарні Apple. У програмі можна настроїти тип і розмір шрифту, тему та яскравість, а також додавати сторінки до закладок, коментувати текст і робити нотатки.

Ви можете активувати перегляд з прокруткою, щоб прокручувати книгу вертикально, а не скидатися вліво на кожній сторінці. Тривале натискання на слово дозволяє скопіювати його, знайти, виділити, зробити замітку, знайти його в книжці або поділитися ним із кимось ще. Функція пошуку приведе до певного слова або номера сторінки (мал. 1.4).

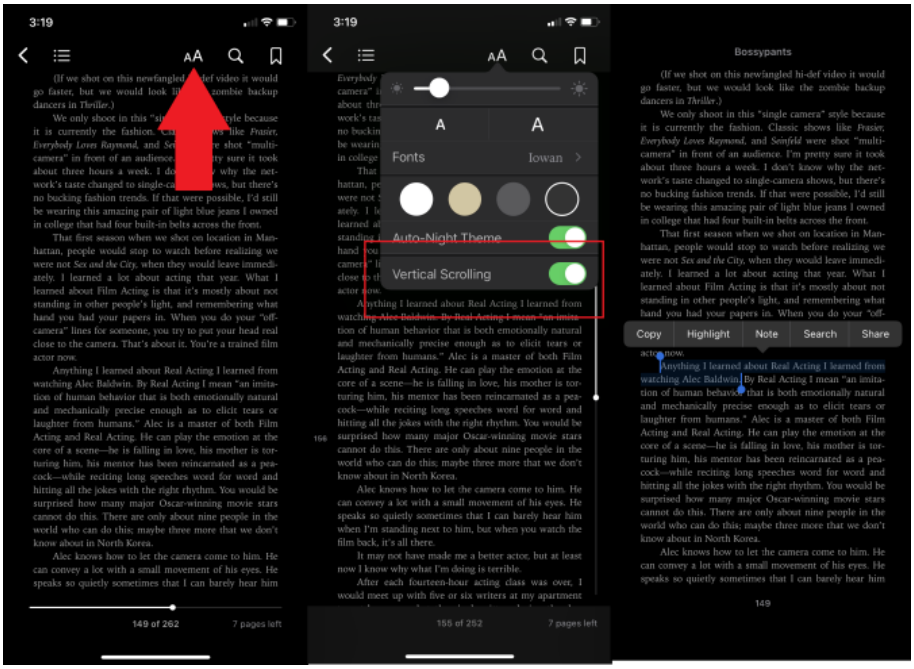


Рисунок 1.4 – Інтерфейс програми Apple Books

Kobo Books

Програма Kobo Books, призначена як для електронних, так і для аудіокниг, доступна для Windows, MacOS, iOS/iPadOS та Android. Це дозволяє читати книги, завантажені з магазину Kobo, а також імпортовані книги, збережені у форматі PDF або EPUB.

При торканні екрана і в нижньому правому куті з'являється ряд значків. Звідси можна керувати стилем та розміром шрифту, макетом та темою. Клавіші регулювання гучності можна використовувати для перегортання кожної сторінки, зміни орієнтації та налаштування переходу між сторінками. При виділенні слова можна отримати його визначення або анотувати його. Також можна додати сторінку до закладок та переглянути список заголовків розділів та інші відомості (рис. 1.5).

Рисунок 1.5 – Інтерфейс програми Kobo Books

FBReader

FBReader дозволяє читати книги, завантажені з власної мережевої бібліотеки або ті, що їх вручну імпортує з інших джерел. Програма підтримує різні формати, включаючи PDF, ePub, mobi, RTF, HTML та звичайний текст. Версії програми доступні для iOS, iPadOS, Android, Windows та Linux.

При читанні електронної книги можна перемикатися між світлими і темними темами, шукати текст, змінювати орієнтацію і збільшувати або зменшувати масштаб. На екрані настроювань можна настроїти стиль та розмір тексту, поля, зовнішній вигляд, колір та перегортання сторінок. Існує можливість розширення можливості FBReader, встановивши та інтегрувавши різні програми та плагіні, такі як автономний словарь та програма для читання PDF (рис. 1.6).

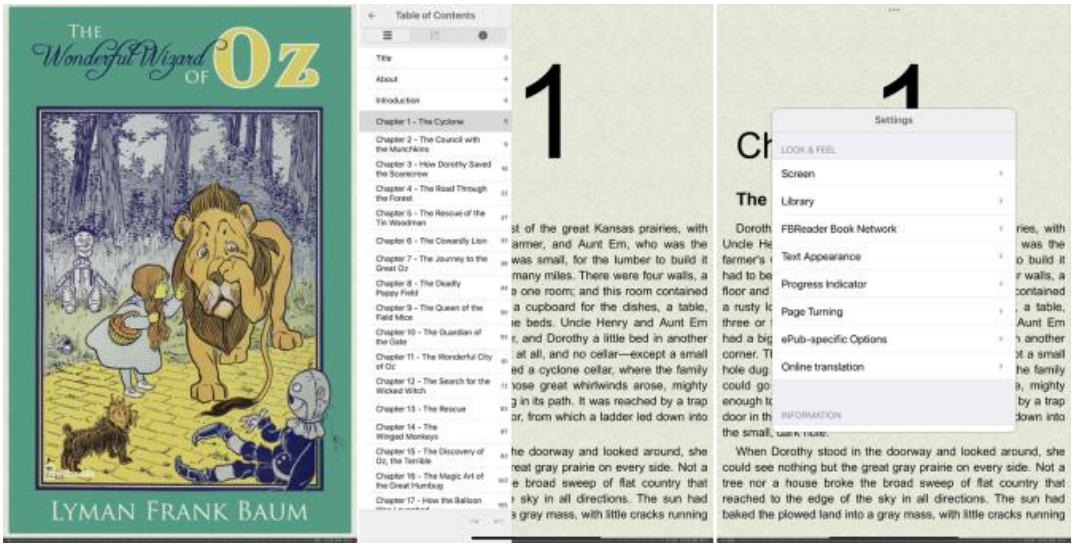


Рисунок 1.6 – Інтерфейс програми FBReader

1.3. Аналіз сучасних бібліотек для створення мобільних додатків мовою програмування Python

Далі буде проведено огляд фреймворків Python, які можна використовувати для розробки програм та ігор для мобільних пристроїв. Деякі з цих платформ також підтримують настільні програми або працюють як окремі інструменти складання для компіляції складання як для настільних комп'ютерів, так і для мобільних пристроїв. Також можна використовувати ту саму кодову базу з невеликими змінами для розгортання програм та ігор як на настільних комп'ютерах, так і на мобільних пристроях [9].

Kivy

Kivy — це безкоштовна среда розробки програм та ігор з відкритим вихідним кодом, яка дозволяє розробляти програми та 2D-ігри, сумісні з настільними та мобільними пристроями. Вона поставляється з досить простою схемою ліцензування, що дозволяє розробляти додатки для комерційних цілей. Основні функції Kivy включають підтримку мультитач-введення, апаратно-прискореної графічної підсистеми, безліч зумовлених віджетів графічного інтерфейсу, віджети, власна унікальна мова дизайну «kv», що підходить для

швидкого прототипування, і можливість створювати пакети для настільних операційних систем і додатків. мобільні пристрої, такі як Android та iOS [10]. Kivy входитиме до репозиторії багатьох дистрибутивів Linux.

BeeWare

BeeWare — безкоштовна среда розробки програм з відкритим вихідним кодом, заснована на Python. За своєю суттю фреймворк працює за ідеєю "напиши один раз – розгорни скрізь". Можна використовувати ту саму кодову базу для розробки та створення програм як для настільних, так і для мобільних операційних систем. Програми, розроблені з використанням BeeWare, добре інтегруються з платформами та зберігають свій зовнішній вигляд, специфічний для платформи. BeeWare також має м'які умови ліцензування, тому його можна використовувати для комерційних проектів. Основні функції BeeWare включають API для доступу до своїх віджетів графічного інтерфейсу, API для доступу до бібліотек для конкретних платформ, можливість розгортання програм на кількох платформах і так далі.

Pyqtdeploy

Pyqtdeploy — надає набір інструментів, який дозволяє запакувати програму PyQt для різних платформ, включаючи настільні та мобільні пристрої (Android та iOS). Pyqtdeploy також може упаковувати програми з графічним інтерфейсом, програми CLI та бібліотеки. Його також можна використовувати для створення пакетів для програм Python, які не використовують бібліотеки PyQt. Pyqtdeploy знаходиться під BSD ліцензією, що дозволяє використовувати його для комерційних проектів [11].

Python для Android

Python-for-android або p4a – це набір інструментів, які можна використовувати для пакування програм Python для платформи Android. Він має надійний набір інструментів для збирання, що дозволяє створювати файли apk для Android, які можна навіть публікувати у Play Store. Python для Android в

основному розроблявся як утиліта для пакування програм Kivy, але тепер він дозволяє також упаковувати інші програми на основі Python. Інші основні функції Python для Android включають підтримку збирання пакетів для декількох архітектур, підтримку упаковки програм, розроблених за допомогою бібліотек SDL2, і так далі.

Ren'Py

Ren'Py – це безкоштовний інструмент з відкритим вихідним кодом, який можна використовувати для розробки візуальних новелів як для настільних комп'ютерів, так і для мобільних пристроїв (Android та iOS). Заснований на Python, має графічний інструмент управління проектами, а також утиліти командної строчки для розробки власних ігор. Крім візуальних новелей, ви також можете використовувати його для створення ігор-симуляторів та рольових ігор, використовуючи систему сценаріїв. Інші основні функції Ren'Py включають API управління компонентами, API сценаріїв діалогів, підтримку мультитач, підтримку автоматичного збереження, підтримку сцен зі швидкою перемоткою вперед, підтримку сцен перемотування назад, підтримку пропуску сцен, підтримку геймпада, попередньо визначену анімацію і переходи. , анімації та переходи і т. д.

1.4 Постановка завдань проектування

З проведеного огляду сформуємо завдання проектування. Для розробки мобільного додатка використовуватимемо фреймворк Kivy. Kivy є досить популярним на даний момент проектом, що забезпечує достатню кількість прикладів та довідкової інформації. Також він дозволяє максимально використовувати ресурси мобільного пристрою та використовує вільну систему ліцензування. Програма повинна працювати з відкритим стандартом EPUB та забезпечувати читання книг та навігацію.

Для вирішення цієї задачі сформуємо наступні завдання проектування:

- - провести встановлення середовища розробки
- - створити інтерфейс користувача
- - створити програмний код програми
- - провести тестування розробленого додатку

2. РОЗРОБКА МОБІЛЬНОГО ДОДАТКА ДЛЯ ЧИТАННЯ ЕЛЕКТРОННИХ КНИГ

2.1. Розробка інтерфейсу користувача

Вікна програми відповідають одному шаблону (рис. 2.1)

Рисунок 2.1 – Шаблон вікон програми

Вікна містять такі елементи:

- кнопка для переходу на головне вікно програми
- кнопка для переміщення вперед-назад
- вікно налаштувань, що викликається шляхом тривалого натискання на екран
- робоча область, містить елементи відображення інформації та управління на конкретних сторінках.

2.2. Розробка вихідного коду

В результаті аналізу опису предметної області виділено такі сутності:

- Жанр (Genre) - описує жанр книги;
- Книга (Book) - описує книгу ;
- Автор (Author) – описує автор книги;
- Видавництво (Publisher) описує видавництво книги.

Концептуальна, логічна та фізична модель (або ERD) – це три різні способи моделювання даних.

Хоча всі вони містять сутності та відносини, вони різняться в цілях, для яких вони створені, та аудиторії, на яку вони націлені. Загальне розуміння трьох моделей полягає в тому, що бізнес-аналітик використовує концептуальну та логічну модель для моделювання даних, необхідних та створюваних системою з точки зору бізнесу, тоді як розробник баз даних уточнює ранній дизайн для створення фізичної моделі з метою представлення фізичної структури бази. даних [19].

Під час першого етапу створення інформаційної системи розроблено концептуальну модель бази даних (рис. 2.2). Концептуальна схема або концептуальна модель даних є картою зрозуміти та їх відносин, що використовуються для баз даних. Вона визначає семантику організації та є ряд тверджень щодо її природи. Зокрема, він описує речі, що мають значення для організації (класи сутностей), про які необхідно збирати інформацію, а також характеристики (атрибути) та асоціації між сутностями (відносини) [20]. Модель створена у програмі PowerDesigner .

PowerDesigner - це інструмент моделювання корпоративної взаємодії, створений кампанією Sybase, що зараз належить компанії SAP.

PowerDesigner включає підтримку таких функцій:

- Моделювання бізнес-процесів

- Генерація коду (Java, C#, VB .NET, Hibernate, EJB3, NHibernate, JSF, WinForm, PowerBuilder)
- Моделювання даних (працює з більшістю основних систем СУБД)
- Моделювання сховищ даних
- Моделювання об'єктів (діаграми UML 2.0)
- Генерація звітів

У цій роботі Power Designer вибрано за рахунок можливості послідовної розробки концептуальної та фізичних моделей.

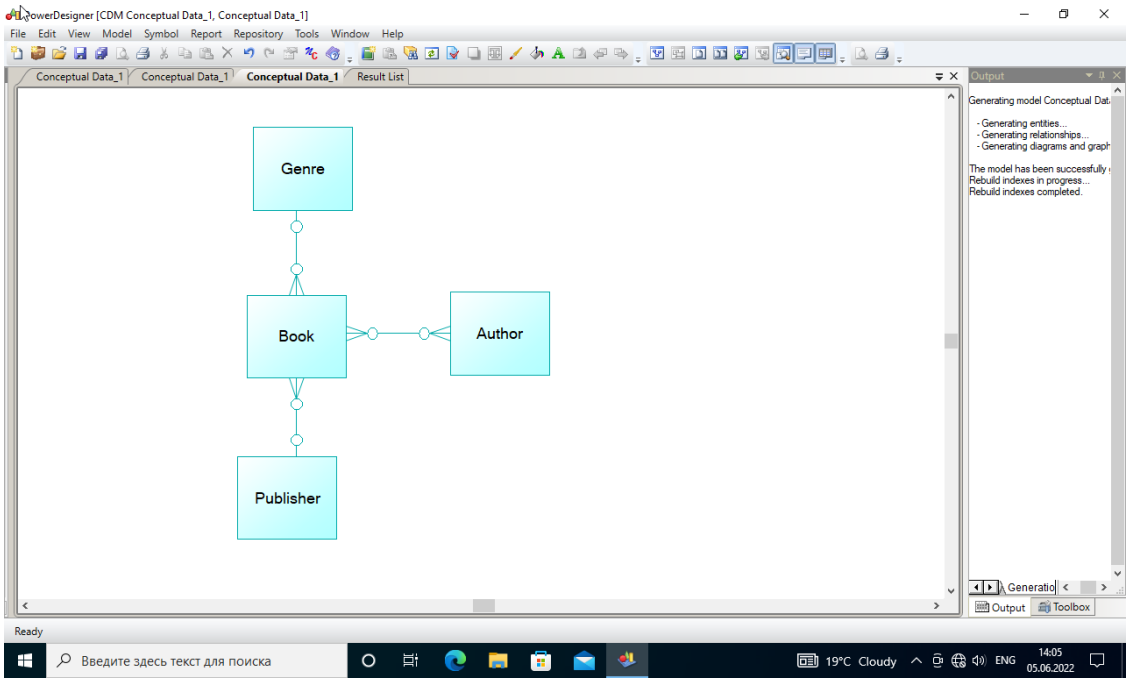


Рисунок 2.2 -Концептуальна модель бази даних

Логічна модель даних сформована з концептуальної моделі та представлена на рисунку 2.3

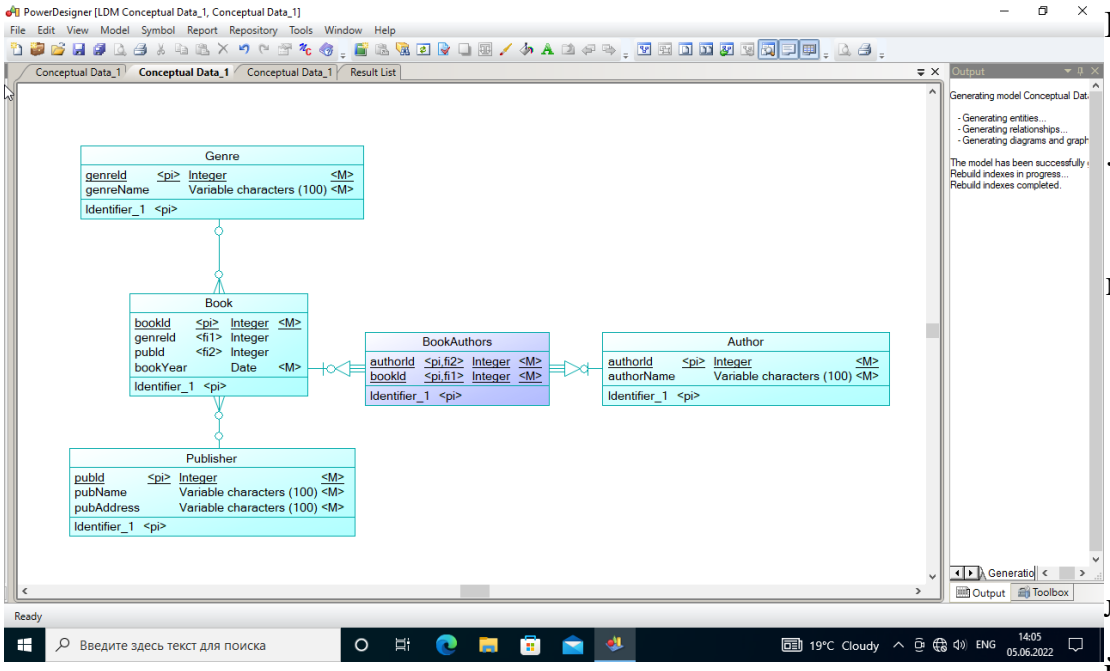


Рисунок 2.3 – Логічна модель даних

У логічній моделі з'явилися обмеження первинних і зовнішніх ключів, також додано відношення BookAuthors, яке реалізує зв'язок «багато-багатьом» між сутностями Book та Author.

У таблицях 1 - 6 показані атрибути кожної сутності логічної моделі

Таблиця 1 – Атрибути та їх характеристики для відношення «Genre»

Атрибут	Опис	Тип даних
genreId	Ідентифікатор жанру	Лічильник
genreName	Ім'я викладача	Текстовий

Таблиця 2 - Атрибути та їх характеристики для відношення "Book"

Атрибут	Опис	Тип даних
bookId	Ідентифікатор книги	Лічильник
genreId	Ідентифікатор жанру	Лічильник
pubId	Ідентифікатор видавництва	Лічильник
bookYear	Рік випуску книги	Дата

Таблиця 3 - Атрибути та їх характеристики для відношення "Publisher"

Атрибут	Опис	Тип даних
pubId	Ідентифікатор видавництва	Лічильник
pubName	Назва видавництва	Текстовий
pubAddress	Адреса видавництва	Текстовий

Таблиця 4 - Атрибути та їх характеристики для відношення "Author"

Атрибут	Опис	Тип даних
authorId	Ідентифікатор автора	Лічильник
authorName	Ім'я автора	Текстовий

Таблиця 5 – Атрибути та їх характеристики щодо “BookAuthors” .

Атрибут	Опис	Тип даних
authorId	Ідентифікатор автора	Лічильник
bookId	Ідентифікатор книги	Лічильник

При перенесенні логічної моделі у фізичну (для конкретної СУБД) виконуються такі дії [21]:

- визначення типів даних для атрибутів;
- визначення допустимості невизначених значень;
- вибір стратегії обробки виняткових ситуацій при спробах порушення цілісності;
- уточнення назва таблиць, атрибутів;
- реалізація обмежень цілісності, процедурне оброблення;
- формування опису моделі даних в межах обраної СУБД;
- формування індексів;
- секціонування та партикування.

Фізична модель бази даних представлена на рисунку 2.4

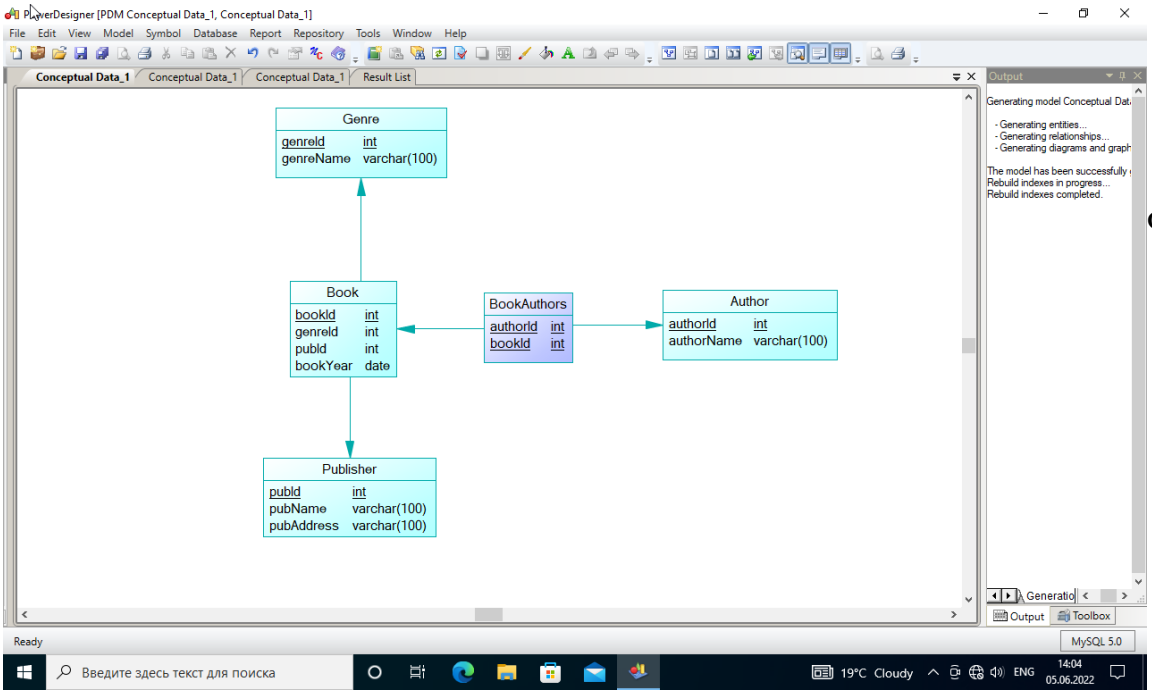


Рисунок 2.4 – Фізична модель бази даних

На рисунку 2.5 показано діаграму класів. Вона містить класи, що відповідають таблицям бази даних, з методами зміни, додавання та видалення інформації. Також створено клас Application, що дозволяє керувати роботою програми.

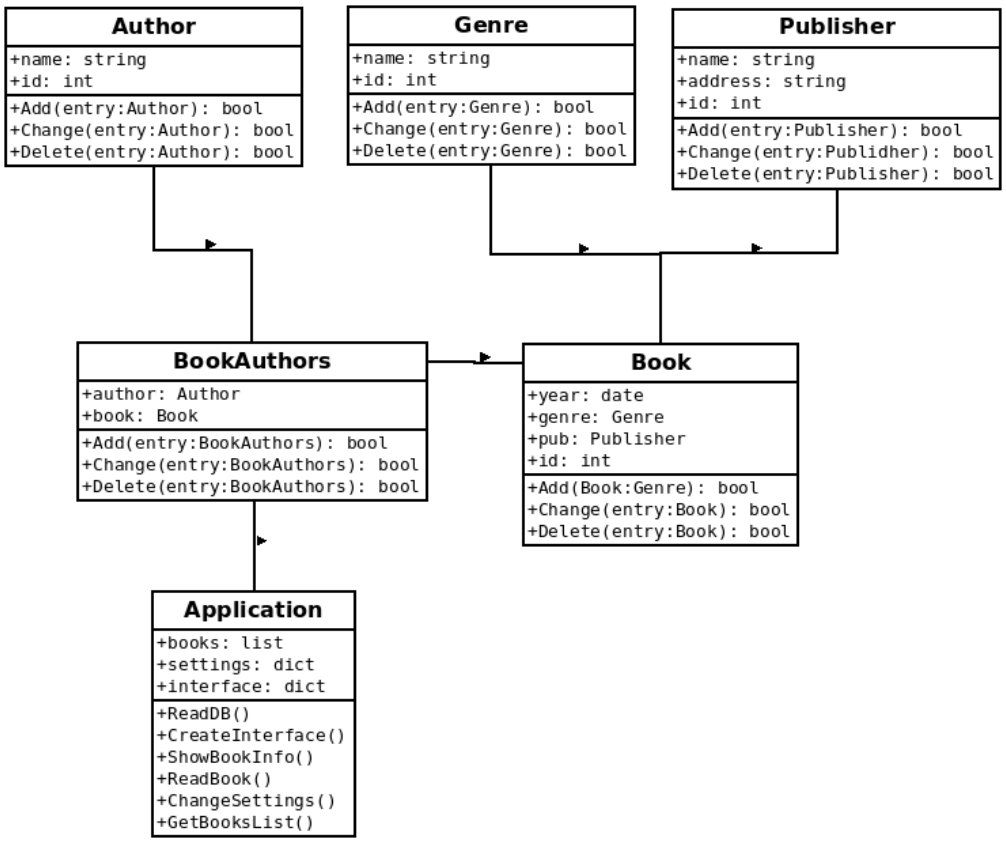


Рисунок 2.5 – Діаграма класів

Алгоритм роботи програми можна описати за допомогою діаграми станів та переходів між ними (рис. 2.6). Перехід може бути ініційований діями користувача або внутрішніми процесами програми.

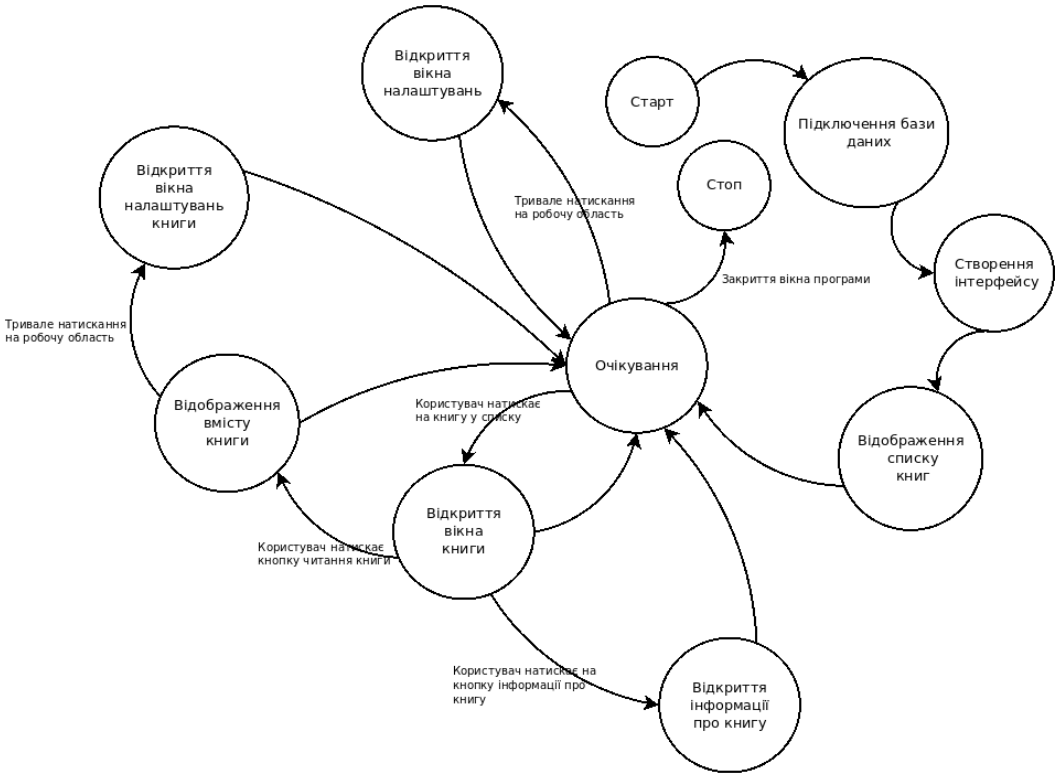


Рисунок 2.6 – Діаграма станів

Інтерфейс клієнтської частини адаптований під використання сенсорним екраном. Розробка клієнтської програми проводилася серед Android Studio .

Android Studio - це офіційне інтегроване середовище розробки (IDE) для операційної системи Android від Google, побудоване на програмному забезпеченні JetBrains IntelliJ IDEA та розроблене спеціально для розробки під Android. Вона доступна для завантаження в операційних системах на базі Windows, macOS і Linux або як послуга на основі підписки в 2022 році.

Зовнішній вигляд IDE представлений рисунку 2.7.

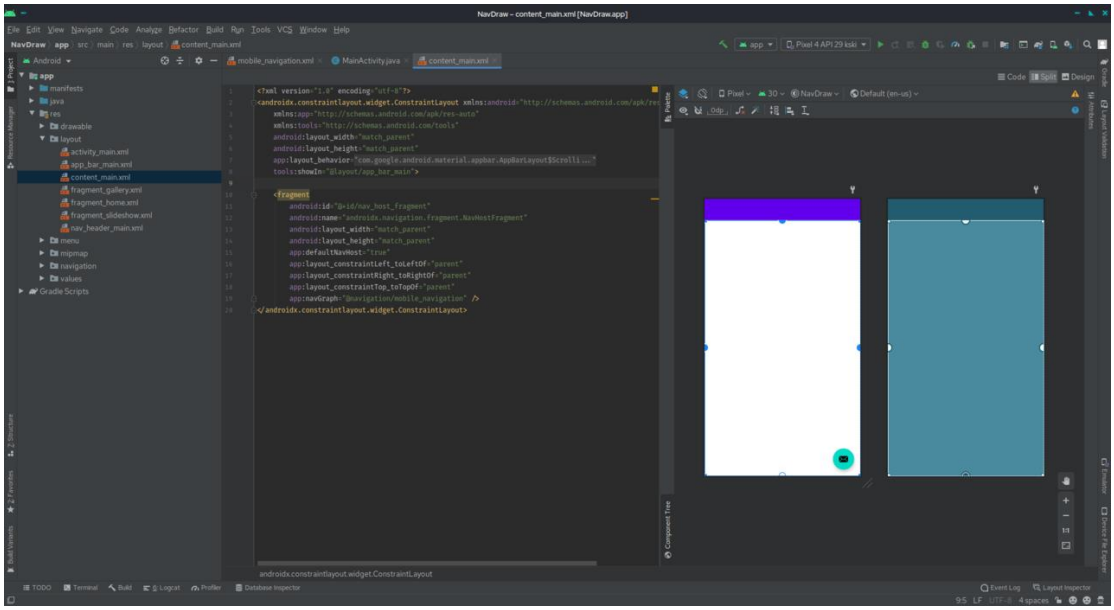


Рисунок 2.7 – Інтерфейс IDE Android Studio

У поточній стабільній версії передбачено такі функції:

- Підтримка збирання на основі Gradle
- Рефакторинг для Android та швидкі виправлення
- Інструменти Lint для визначення продуктивності, зручності використання, сумісності версій та інших проблем
- Можливості інтеграції ProGuard та підпису додатків
- Майстри на основі шаблонів для створення загальних дизайнів та компонентів Android
- Редактор макетів, який дозволяє користувачам перетягувати компоненти інтерфейсу користувача, можливість попереднього перегляду макетів на декількох конфігураціях екрану
- Підтримка створення програм Android WearBіртуальний пристрій Android (емулятор) для запуску та налагодження програм у студії Android.

З точки зору програміста, Android – платформа, яка абстрагує розробника від ядра, і дозволяє йому створити код у Java, також іншими мовами програмування за допомогою відповідного фреймворку, який потім здійснюють трансляцію в Java

код. Android має кілька корисних функцій. По-перше, ця основа пропонує великий набір API для створення різних типів додатків, а також забезпечує можливість повторного використання та заміни компонентів, що пропонуються програмами платформи та сторонніх виробників. По-друге, наявність віртуальної машини Dalvik відповідає за запуск додатків на Android. Крім того, послуги безлічі розробників графічних бібліотек для 2D та 3D додатків, підтримують мультимедійні формати (Ogg Vorbis, MP3, MPEG-4, H.264, PNG), API для доступу до камери, GPS, компас, акселерометр, сенсорний екран, джойстик або клавіатури. Існує навіть спеціальний API для відтворення звукових ефектів фону. Не всі Android-пристрої мають усі ці можливості – є апаратний підрозділ. Звичайно, Android не вичерпується список можливостей згаданих тут. Android-архітектура формується з багатьох компонентів. Кожен компонент ґрунтується на елементах нижнього рівня. На рисунку 2.8 наведено огляд основних компонентів Android.

У нижній частині малюнка 3 показано, що ядро Linux забезпечує основні драйвери апаратних компонентів системи. Крім того, ядро відповідає за пам'ять, управління процесами, мережеву підтримку тощо.

Середовище виконання Android, побудоване навколо ядра, відповідає за розробку та здійснення додатків Android. Кожна програма працює у власному процесі зі своєю власною віртуальною машиною Dalvik.

Dalvik запускає програми у форматі байткодом DEX. Java-файли з класом перетворені на формат DEX за допомогою спеціального інструменту DX, доступного в SDK. Формат DEX займає набагато менше місця в пам'яті, ніж класичний тип файлів CLASS, що досягається за рахунок високого ступеня стиснення, поділу на таблиці та об'єднання кількох CLASS-файлів.

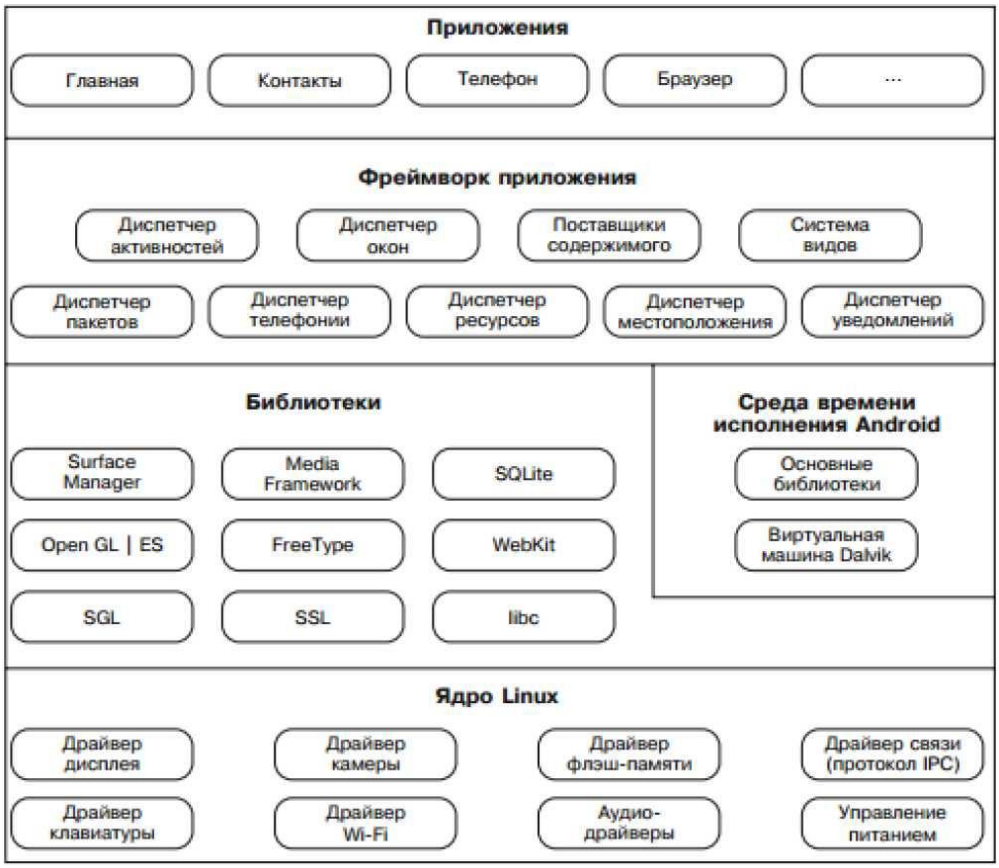


Рисунок 2.8 – Огляд архітектури

Dalvik віртуальна машина взаємодіє з бібліотеками ядра, які пропонують базові функціональні можливості Java-програм. Ці бібліотеки мають великий, але не повний список класів, доступних через Java SE.

До Android 2.2 (Froyo) весь код було інтерпретовано. У Froyo було введено відстеження JIT-компілятора, який може компілювати частини байткоду в машинний код на льоту. Це значно підвищує продуктивність програм, які вимагають більшої кількості обчислень. JIT компілятор може використовувати можливості процесорів, які спеціально розроблені для складних обчислень, таких як операції з плаваючою комою. Крім того, він включав свій власний збирач сміття Dalvik (Garbage Collector, GC). Він працює за принципом "повідомити і забрати", що іноді ставить розробників у безвихідь. Проте, якщо його ретельно вивчити, він може бути ефективно використаний у створенні ігор. Кожна програма, що працює

в екземплярі віртуальної машини Dalvik, має у своєму розпорядженні від 16 до 24Mb RAM. Це необхідно мати на увазі, використовуючи картинки та звукові ресурси.

Фреймворк пов'язує системні бібліотеки та середовища виконання, створюючи тим самим прив'язку до Android. Структура управляє програмами та пропонує складне середовище, в якому вони працюють. Розробники створюють додатки для цієї структури з набором API-інтерфейсів, що охоплюють такі області, як розробка інтерфейсу користувача, фонові служби, повідомлення, управління ресурсами, доступ до периферійних пристроїв і т.д. Всі види ключових програм, що постачаються з операційною системою Android (наприклад, клієнт електронної пошти), написані за допомогою API. Програми, інтерфейс або фонові служби можуть взаємодіяти з іншими програмами. Це з'єднання дозволяє застосунку використовувати інші компоненти. Простий приклад – це програма, яка робить фото-образ, а потім може його обробити. Програма запитує у системи інший компонент програми, який забезпечує цю дію. Далі, перша програма може повторно використовувати цей компонент (наприклад, від вбудованої камери або з фотогалереї). Цей алгоритм видаляє значну частину тягаря від програміста, і дозволяє настроїти поведінку різноманіття аспектів Android.

Для розробки власної програми необхідно вивчити інструментарій комплекту для розробки програм Android Software Development Kit (SDK).

Для розробки програм використовується високорівневий прикладний інтерфейс програмування для Android, за допомогою якого можна створювати програми для кінцевих користувачів Android. Розглянемо особливості емулятора Android, фундаментальні компоненти Android та пакети, що входять до складу SDK. Також буде наведено кілька фрагментів коду.

Комплект для розробки програмного забезпечення (SDK) для Android постачається з Android Studio, плагіном, який називається набором інструментальних засобів для розробки на Android (ADT). Android SDK може

використовуватись без ADT; замість інструментів можна використовувати інструменти командного рядка. Емулятор підтримує використання обох підходів, і за його допомогою можна запустити, виправити та перевірити програми. 90% розробки програм може бути завершено, навіть без використання реального пристрою. Повністю функціональний емулятор для Android відтворює найбільш вивчені характеристики пристрою. Серед функцій, які не можуть бути імітовані в емуляторі, є USB-підключення, робота камери та відео, імітація роботи навушників, батареї та технології Bluetooth.

Android Emulator базується на технології з відкритим вихідним кодом імітація процесора під назвою QEMU, який був розроблений Беллар. Та сама технологія може емулювати одну операційну систему на інший, незалежно від того, яка використовується процесором. QEMU забезпечує емуляцію рівня процесора.

При використанні емулятора Android імітує процесор, який функціонує на базі ARM (Advanced RISC Machine, Advanced RISC-машина). ARM - 32-розрядна архітектура мікропроцесорів на базі RISC (Reduced Instruction Set Computer, комп'ютер зі скороченим набором команд), яка за рахунок скорочення кількості команд досягає простоти конструкції та підвищення продуктивності. Емулятор використовує процесор в такій версії Linux, яка використовується в Android.

ARM широко використовується в мобільних і вбудованих електронних пристроях, де важливо розподілити невелику кількість енергії. Багато мобільних пристроїв, що є у продажу, мають процесори з цією архітектурою. Наприклад, Apple Newton з урахуванням процесора ARM6.

Інтерфейс користувача, що використовується в рамках ОС Android, можна порівняти з іншими повнофункціональними UI-структурами, що застосовуються на локальних комп'ютерах. Це більш сучасний та асинхронний характер. Насправді, UI-фреймворк Android належить до четвертого покоління, якщо вважати перше покоління традиційного прикладного програмування інтерфейсу

Microsoft Windows, що базується на C та MFC (класів Microsoft Foundation, Microsoft бібліотеки базових класів, заснованих на C++) – другий. У цьому випадку Swing UI-фреймворк, заснований на Java, буде третім поколінням, і, як було запропоновано в його конструктивних можливостях, набагато перевершує MFC за гнучкістю. Android UI, JavaFX, Microsoft Silverlight та мова Mozilla XML (XUL) інтерфейси користувача включають новий тип UI-фреймворк четвертого покоління, який є декларативним UI і підтримує незалежну тематизацію.

При програмуванні в інтерфейсі користувача Android використовується оголошення інтерфейсу у файлах XML. Тоді визначення подання XML завантажуються в додаток з інтерфейсом користувача, як у Windows. Навіть меню програми завантажуються з файлу XML. Екрани (вікна) Android часто називають діяльністю, які включають кілька типів і користувачі повинні виконати логічний елемент процесу. Види (уявлення) є основними елементами, які знаходяться в інтерфейсі Android. Форми можуть бути об'єднані у групи (групи видів). Для внутрішньої організації цього виду вони використовують протягом тривалого часу відому в програмуванні концепцію полотна і взаємодію користувача з системою.

Такі композитні уявлення, які включають види і групи видів, працюють на основі спеціальної логіки компонента інтерфейсу Android.

Одним із ключових понять бази є управління Android-Power життєвим циклом вікон подій (вікна діяльності). Протоколи системи використовуються, так що Android може керувати ситуацією поки користувач, виконує дії, щоб приховати, відновити, зупинити та закрити вікно явища.

Фреймворк для інтерфейсу Android, разом з іншими компонентами Android заснований на новій сутності, називається намір. Намір – це складний феномен, який поєднує у собі ідеї, такі як повідомлення, що відображаються у вікні, дії модель “публікації та підписки”, обмін міжпроцесорною інформацією та регістрами додатків. Нижче наведено приклад використання класу Intent для активації або запуску веб-браузера:

public static void invokeWebBrowser (Activity activity) {

Intent intent = new Intent (Intent. ACTION_VIEW): intent. setData (Uri. parse ("http://www.google.com")); Діяльність. startActivity (intent);}

У цьому прикладі, використовуючи намір, ми говоримо Android відкрити вікно, що підходить для відображення вмісту веб-сайту. Залежно від того, який браузер встановлений на мобільному пристрої, Android вибиратиме найбільш підходяще для відображення сайту.

Крім того, широко підтримується ресурси Android, яким знайомі рядки та растрові зображення, та принаймні деякі відомі елементи, такі як визначення уявлення на основі XML. Використання ресурсів у цих рамках здійснюється по-новому, за допомогою чого робота ресурсу стає більш простою, зрозумілою та зручною. Нижче наведено приклад, в якому автоматично генерується ID ресурсу та визначається у файлах XML:

```
public final class R {
public static final class attr { }
public static final class drawable {
public static final int myanimation=0x7f020001;
public static final int numbersl9=0x7f02000e:
}
public static final class id {
public static final int textViewIdl=Ox7f080003;
}
public static final class layout {
public static final int frame_animations_layout=0x7f030001;
public static int main=0x7f030002;
}
public static final class string {public static final int hello=0x7f070000; }
}
```

Ці класи відповідають автоматично згенерованому ID або XML-файлу, елементу або безлічі. Якщо потрібно використовувати такі визначення XML, замість того, щоб використовувати їх ідентифікаційні дані. Таке посередництво дуже корисне у локалізації. Ще одна інноваційна концепція Android – це контент-провайдер (постачальник контенту). Постачальник контенту відноситься до абстракції джерела даних, який може бути представлений як емітер та REST обслуговування (репрезентативна State Transfer). В основі цієї бази даних лежить абстракція SQLite і робить цю властивість контент-провайдерів потужним інструментом для розробки програм. Було зазначено, що XML відіграє важливу роль в описі інтерфейсів для Android.

Нижче показано, як оголосити меню в XML-файлі:

<menu xmlns: android="http://schemes. android.com/apk/res/android">

<! – У цій групі використовується стандартна категорія. - ->

<group android: id="@+id/menuGroup_Main">

<item android: id="@+id/menu_clear"

android: orderInCategory="10"

android: title="clear" />

<item android: id="@+id/menu_show_browser"

android: orderInCategory="5"

android: title="show browser" />

</group>

</menu>

Хоча Android підтримує і діалогові вікна, вони є асинхронними. Робота з такими асинхронними діалогами особливо важка для розробників, які звикли працювати із синхронними модальними діалоговими вікнами, що використовуються в деяких віконних фреймворках.

Android також підтримує анімацію - ця функція включена разом зі стеком інтерфейсу користувача, який заснований на видах об'єктів. В Android

використовується анімація двох видів: з побудовою проміжних кадрів (tweening animation) та покадрової (frame-by-frame animation). Проміжною анімацією називають картинки, що відображаються між основними. На комп'ютері ці зображення створюють зміни середніх значень у певні проміжки часу та перемальовки фону. Кадрова анімація складається із послідовності кадрів, які намальовані один за одним через рівні проміжки часу. На Android використовуються обидві версії анімації, використовуючи функцію зворотного виклику анімації, інтерполяторів та матрицю перетворення. Крім того, Android може ідентифікувати ці типи анімації у файлі XML-ресурсів.

Графічна бібліотека, яка заснована на такому процесі, підтримує стандартні матричні перетворення, що дозволяє масштабувати, переміщувати та повертати зображення. Об'єкт камери, які присутні у графічній бібліотеці, забезпечує підтримку глибини та проекції, так що на двовимірному інтерфейсі вдається імітувати тривимірні ефекти.

Android доступна і тривимірна графіка. Це пов'язано з тим, що запроваджено стандарт OpenGL ES 1.0. Як і OpenGL, він заснований плоскою мовою API C. Так як інтерфейс прикладного програмування Android SDK заснований на Java, OpenGL для доступу до нього повинен використовувати Java-зв'язування. Java ME є обов'язковим для OpenGL ES і вже визначено за допомогою запиту на специфікації Java QSR 239 і для Android OpenGL ES використовує той же тип Java-зв'язування.

На Android використовуються нові підходи, пов'язані з ідеєю інформації на кінчиках пальців (information at your fingertips), доступ до якої здійснюється через домашню сторінку. Перша з цих ідей називається живі каталоги (живі папки). Використовуючи живі каталоги, можна опублікувати колекцію елементів у папці, розташованій на головній сторінці сайту. Зміст цієї колекції змінюється від базової зміни у його базі даних. Нові дані можуть бути надіслані на пристрій або зі знімних носіїв або з Інтернету.

Друга ідея пов'язана із домашньою сторінкою – це домашній віджет (home screen widget). Головні віджети використовуються для малювання за допомогою інтерфейсу користувача віджету інформації на домашній сторінці. Ця інформація може змінюватися через регулярні проміжки часу. Приклад таких даних може бути кілька повідомлень електронної пошти, які зберігаються на пристрої.

Вбудований пошук Android (Integrated Android Search) – це третя ідея, пов'язана із використанням домашньої сторінки. Цей тип пошуку дозволяє шукати інформацію як у пристрої, так і в Інтернеті. В Android ця функція не обмежується лише одним пошуком і може давати команди за допомогою елемента керування пошуку.

Крім того, так звані підтримувані жести Android, тобто інтерпретація рухів пальців користувача роботи з пристроєм. Android дозволяє записувати будь-яку послідовність рухів пальців на екрані та зберігати їх у вигляді жестів. Потім такі жести можуть використовуватись програмами для визначення конкретних дій.

Окрім інструментів Android SDK, є й інші незалежні інновації, які роблять процес розробки цікавим та простим. Деякі приклади таких явищ – XML/VM, PhoneGap та титану. Titanium дозволяє використовувати HTML технологію у програмуванні для браузера на основі WebKit Android.

Ще одним цікавим компонентом Android SDK є послуги на основі розташування. У цьому розділі SDK надає розробникам інтерфейсів прикладного програмування, програму, призначену для роботи з картами, а також для отримання інформації в режимі реального часу, пов'язаної з розташуванням пристрою.

Компоненти дисплея та компоненти, пов'язані з телефонією. Інтерфейси Android API призначені для роботи з аудіо-, відео- та телефонними компонентами.

До версії 1.5 на Android можна було лише записувати аудіо, а відео – ні. Версія 1.5 дала можливість записувати аудіо та відео. Це було зроблено за допомогою MediaRecorder. У версіях Android 2.0 та вище застосовується Pico

двигун для перетворення тексту на мову (TTS). Інтерфейс, що використовується в роботі Android при перетворенні тексту в мову, дуже проста, а також має відповідний код:

TextToSpeech mTTS;

mTTS. speak (sometextString, TextToSpeech. QUEUE_ADD); mTTS. setOnUtteranceCompletedListener (this); mTTS. stop();

mTTS. shutdown();

mTTS. synthesizeToFile (...);

Ще кілька методів, що відносяться до цієї сфери: playSilence

setLanguage

setPitch

setSpeechRate

isSpeaking

Важливо відзначити, що у всіх цих концепціях Android, об'єднаних у єдиний файл XML, який визначає пакет додатків. Цей файл є файлом опису (файл маніфесту) (AndroidManifest. Xml). Приклад цього файлу:

<? xml version="1.0" encoding="utf-8"? >

<manifest xmlns: android=http://schemas. android.com/apk/res/android package="com. ai. android. HelloWorid"> android: versionCode="l"

android: versionName="1.0.0">

<application android: icon="@drawab1e/icon"

android: label="@string/app_name">

<activity android: name="HelloWorid"

android: label="@string/app_name">

<intent-filter>

<action android: name="android. intent. action. MAIN"

<category

android: name="android. intent. category. LAUNCHER"

</intent-filter>

</activity

</application>

<-/manifest>

У файлі опису Android містяться визначення дій, реєструються постачальниками вмісту та постачальниками служб, і позначаються права доступу.

Щоб побачити платформу Android, необхідно розглянути структуру пакетів Java. Оскільки Android SDK відрізняється від стандартного розподілу, важливо знати, які пакети підтримуються, а які – ні. Нижче наведено короткий опис важливих пакетів, що складають Android SDK:

• Android.app – реалізує модель програми Android. Серед основних класів – додаток, який описує початкову та кінцеву семантику, а також цілу низку класів, пов'язаних з явищами, елементи управління, діалогових вікон, вікон попереджень та повідомлень;

• Android.Bluetooth – містить класи для роботи з технологією Bluetooth. Основні класи VluetoothAdapter, BluetoothDevise, BluetoothSocket, BluetoothServerSocket та BluetoothClass. BluetoothAdapter клас може бути використаний для керування Bluetooth-адаптером, встановленим на локальному комп'ютері. Цей адаптер може увімкнути, вимкнути або запустити процес виявлення. Клас BluetoothDevise — це віддалений пристрій Bluetooth, до якого можна підключитися. Bluetooth має два сокети, що використовуються між пристроями. Bluetooth Class є типом пристрою Bluetooth, до якого ви підключені;

• Android.content – реалізує концепції, пов'язані із постачальниками контенту. Постачальник контенту дозволяє сумувати обмін та зберігання даних. Крім того, цей пакет реалізує основні ідеї щодо намірів та рівномірного ідентифікаторів ресурсів (URI) в Android;

• Android. content. pm. – надає класи для роботи, пов'язані з диспетчером пакетів. Він містить інформацію про дозволи, встановлені пакети, встановлені

постачальниками, послуги та компоненти, такі як акції, а також встановлені програми;

• Android.content.res – надає доступ до файлів ресурсів, як структурованих неструктурованих. Основні класи AssetManager (для неструктурованих ресурсів) та матеріальних ресурсів;

• Android.database – реалізує ідею абстрагування бази даних. Основний інтерфейс називається Cursor;

• Android.database.sqlite – реалізує концепцію пакета бази даних Android, використовуючи як фізичну базу даних SQLite. Основні класи SQLiteCursor, SQLiteDatabase, SQLiteQuery, SQLiteQueryBuilder та SQLiteStatement. Проте, переважно доводиться працювати з класами абстрактного пакету бази даних Android;

• Android.gesture – у цьому пакеті є всі класи та інтерфейси, необхідні для роботи з певними користувачем жестами. Основні класи Gesture, GestureLibrary, GestureOverlayView, GestureStore, GestureStroke, GesturePoint. Клас Gesture є добіркою GestureStrokes та GesturePoints. Жести зібрані у бібліотеці GestureLibrary. Бібліотеки жестів зберігаються у GestureStore. Імена жест такі, що система може ідентифікувати їх як дії;

• Android.graphics – містить клас Canvas, Camera, Color, Matrix, Movie, Paint, Path, Rasterizer, Shader, SweepGradient nTypeFace;

• Android.graphics.drawable – призначений для роботи з протоколами малювання та фонових зображень, забезпечує ефекти анімації під час роботи з мальованими об'єктами;

• Android.graphics.drawable.shapes – призначені для роботи з контурами, у тому числі ArcShape, OvalShape, PathShape, RectShape та RoundRectShape;

• Android.hardware – дозволяє використовувати звані природні класи, призначені для роботи з камерою. Клас камери є поширеним пристроєм –

камерою, а клас android.graphics.Camera – графічна концепція, яка не має жодного відношення до реальної фізичної камери;

• Android.location – містить класи Address, GeoCoder, Location, LocationManager і LocationProvider. Клас Address є спрощенням-заміщенням Language XAL (мова адреси, що розширюється). Geocoder дозволяє дізнатися адресу координат об'єкта (широта та довгота), і навпаки. У Location представлена інформація про широту та довготу;

• Android.media – містить класи MediaPlayer, MediaRecorder, Ringtone, AudioManager та FaceDetector. Клас MediaPlayer призначений для потокової підтримки аудіо та відео. Клас Ringtone використовується для відтворення коротких аудіо-відліків, які можуть бути використані в мелодії або повідомлень. AudioManager відповідає за регулювання гучності. FaceDetector може бути використаний для виявлення людських осіб на точці (растрові) малюнки;

android.net – реалізує базову мережу на рівні сокетів API. Основні класи включають Uri, ConnectivityManager, локальний сокет та місцевий ServerSocket. Слід зазначити, що Android підтримує рівень HTTPS-браузера і мережевий рівень. Крім того, Android підтримує JavaScript у браузері;

• android.net.WiFi – керує підключенням Wi-Fi. Основні класи WifiManager та WifiConfiguration. Клас WifiManager відповідає за складання списку налаштованих мереж та працює з поточною активною мережею Wi-Fi;

• Android.OpenGL – містить допоміжні класи, які використовуються під час операцій OpenGL ES. Класи Basic OpenGL ES є частиною іншого набору пакетів, взятих із JSR 239.

• Android.os – служба операційної системи, доступ до якої здійснюється за допомогою Java. Деякі важливі класи – BatteryManager, Біндер, FileObserver, Хендлер, Looper та PowerManager. Binder клас забезпечує обмін інформацією між процесами. FileObserver веде облік змін у файлах. Клас Handler використовується

для виконання завдань у потоці повідомлень, і Looper починається саме повідомлення потоку;

• Android.preference – дозволяє програмам надати користувачам можливість керувати своїми налаштуваннями для цієї програми в єдиній формі. Основні класи PreferenceActivity, PreferenceScreen;

• Android.provider – включає набір готових контент-провайдерів, пов'язаних з android. content. ContentProvider. Серед постачальників контенту – Контакти, MediaStore, браузер та налаштування. Цей набір інтерфейсів та класів містить метадані для опису базової структури даних;

• Android.sax – забезпечує ефективний набір простих API для XML (SAX), допоміжні класи для синтаксичного аналізу. Основні класи Element, RootElement, деякі ElementListener інтерфейси;

• Android.speech – містить константи для розпізнавання мовлення. Цей пакет входить лише у версії 1.6 та вище;

• Android.speech.tts – забезпечує підтримку для перетворення тексту на мову. Основний клас – TextToSpeech. Android має механізм PICO TTS (перетворення тексту в мову, синтезатор мови) виробництва SVOX;

• andmid.tekphony – містить класи CellLocation, PhoneNumberUtils та TelephonyManager. TelephonyManager клас для визначення місцезнаходження, з якого було здійснено виклик, номер телефону, ім'я постачальника послуг, тип мережі, тип телефону та серійний номер модуля ідентифікації абонента (Subscriber Identity Module, SIM);

• Android.телефонії.GSM – дозволяє збирати інформацію про адреси осередків, заснованих на розташування даних вишок стільникового зв'язку, але також містить класи, відповідальні за роботу з SMS-повідомленнями. визначає глобальну систему мобільного зв'язку в ім'я цього пакета називається GSM, як оригінальні короткі стандарти обміну повідомленнями (SMS) (Глобальна система мобільного зв'язку);

• Android.telephony.CDMA – підтримує CDMA-телефонію;

• Android.text – містить класи для обробки тексту;

• Android.text.method – надає класи для введення тексту до різних елементів управління;

• Android.text.style – забезпечує різноманітність методів обробки тексту;

• Android.Utils – містить класи, DebugUtils, TimeUtils та Xml;

• Android.view – містить класи, меню View, ViewGroup, а також деякі з процесів слухачів та зворотних викликів;

• Android.view. animation – забезпечує підтримку анімації у структурі проміжних кадрів;

• Android.view. InputMethod – реалізує введення-виведення системної архітектури. Цей пакет міститься лише у версіях 1.5 і вище;

• Android.WebKit – містить класи, пов'язані з веб-браузером. Серед основних класів WebView, CacheManager та CookieManager;

• Android.widget – містить всі класи елементів керування інтерфейсу користувача, які отримані головним чином з точки зору класу. Основні віджети – Кнопка, Галочка, хронометр, AnalogClock, DatePicker, EditText, ListView, FrameLayout, GridView, ImageButton, MediaController, ProgressBar, RadioButton, RadioGroup, RatingButton, скроллер, Scrollview, Spinner, TabWidget,

• com.google. android. maps – містить клас MapView, MapController та MapActivity, необхідних для роботи з Google Maps.

Вищезгадані пакети дуже важливі при роботі з Android. Виходячи з цього списку, можна отримати уявлення про глибинну будову платформи Android.

Загалом, Android Java API включає більше 40 пакетів і більше 700 класів. Тим не менш, всі ці численні пакети складають багату обчислювальну платформу, призначену для написання програм для мобільних пристроїв.

Мобільний додаток для читання електронних книг розроблений за шаблоном проектування модель-представлення-контролер (Model - View - Controller), який

дає переваги більш динамічного програмного забезпечення, окремі компоненти якого можуть бути легко змінені, не порушуючи інші частини. Схема взаємодії компонентів MVC показана рисунку 2.9.

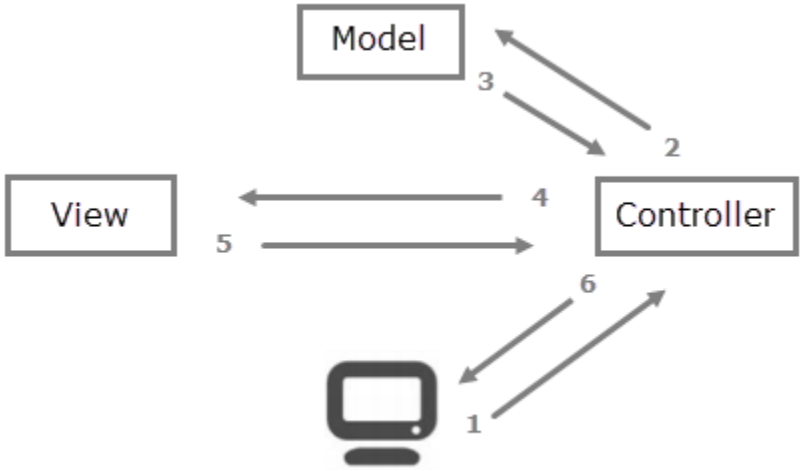


Рисунок 2.9 - Схема передачі у шаблоні MVC

Модель MVC визначає програми з трьома логічними шарами:

- Модель (бізнес-рівень);
- Перегляд (рівень відображення);
- Контролер (управління введенням).

Модель представляє стан конкретного аспекту програми. Модель працює із базою даних.

Контролер обробляє взаємодії та оновлює модель, щоб відобразити зміну стану програми, а потім передає інформацію у виставу.

Подання приймає необхідну інформацію від контролера і відображає інтерфейс користувача для відображення цієї інформації.

Наступним кроком є створення моделей даних, тобто перенесення розробленої раніше фізичної моделі бази даних в її опис засобами Python. Для цього кожну таблицю фізичної моделі представляють клас Python, параметри якого визначають його зв'язки з іншими класами (рис. 2.10).

```
 4 class Genre(models.Model):
 5     genreName = models.CharField(max_length=100)
 6
 7 class Book(models.Model):
 8     bookYear = models.DateField()
 9     genreId = models.ForeignKey(Genre, on_delete=models.CASCADE)
10     pubId = models.ForeignKey(Publisher, on_delete=models.CASCADE))
11
12 class Publisher(models.Model):
13     pubName = models.CharField(max_length=100)
14     pubAddress = models.CharField(max_length=100)
15
16 class Author(models.Model):
17     authorName = models.CharField(max_length=100)
18
19 class BookAuthors(models.Model):
20     authorId = models.ForeignKey(Author, on_delete=models.CASCADE)
21     bookId = models.ForeignKey(Book, on_delete=models.CASCADE)
22
```

Рисунок 2.10 – Створення бази даних

2.3. Тестування програми

Після запуску програми з'являється головне вікно (рис. 2.11). Воно містить такі елементи:

- кнопку для переходу до головного вікна
- кнопки навігації вперед-назад
- список книг

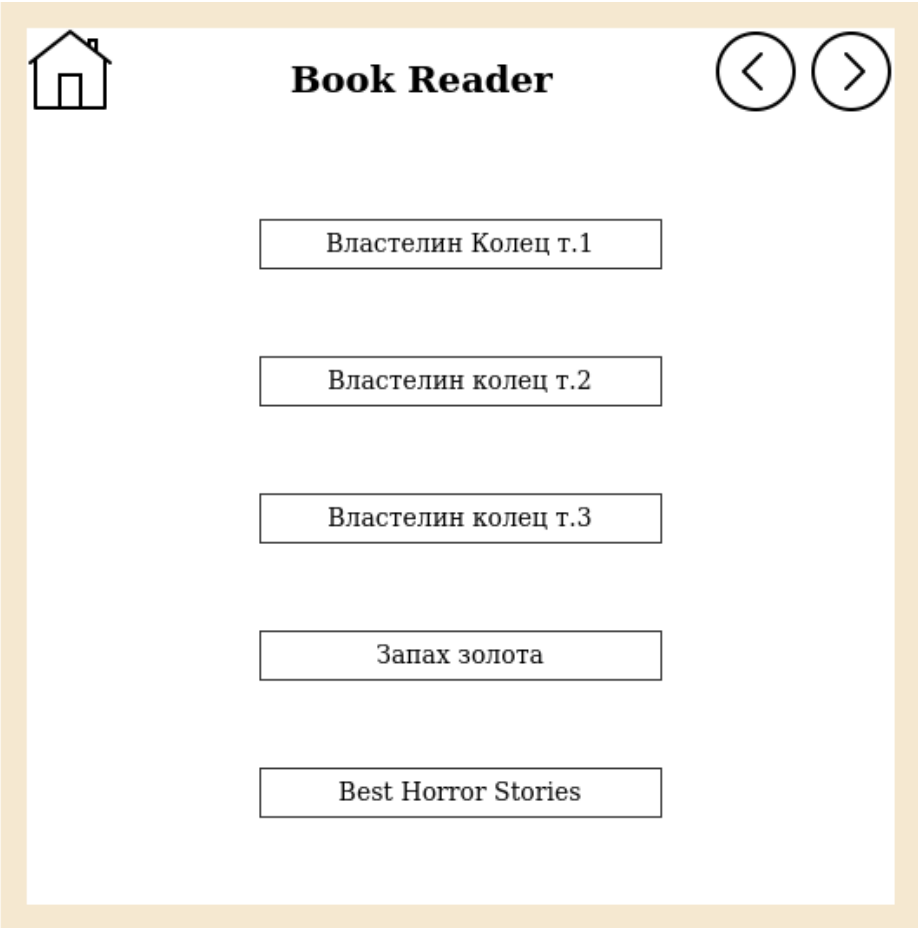


Рисунок 2.11 – Головне вікно програми

При тривалому натисканні на робочому полі головного вікна з'являєтеся вікно налаштувань (рис. 2.12). Тут можна встановити порядок сортування, колір фону та колір шрифту. Книжки можна сортувати за такими параметрами:

- Жанр
- Автор
- видавництво
- час додавання

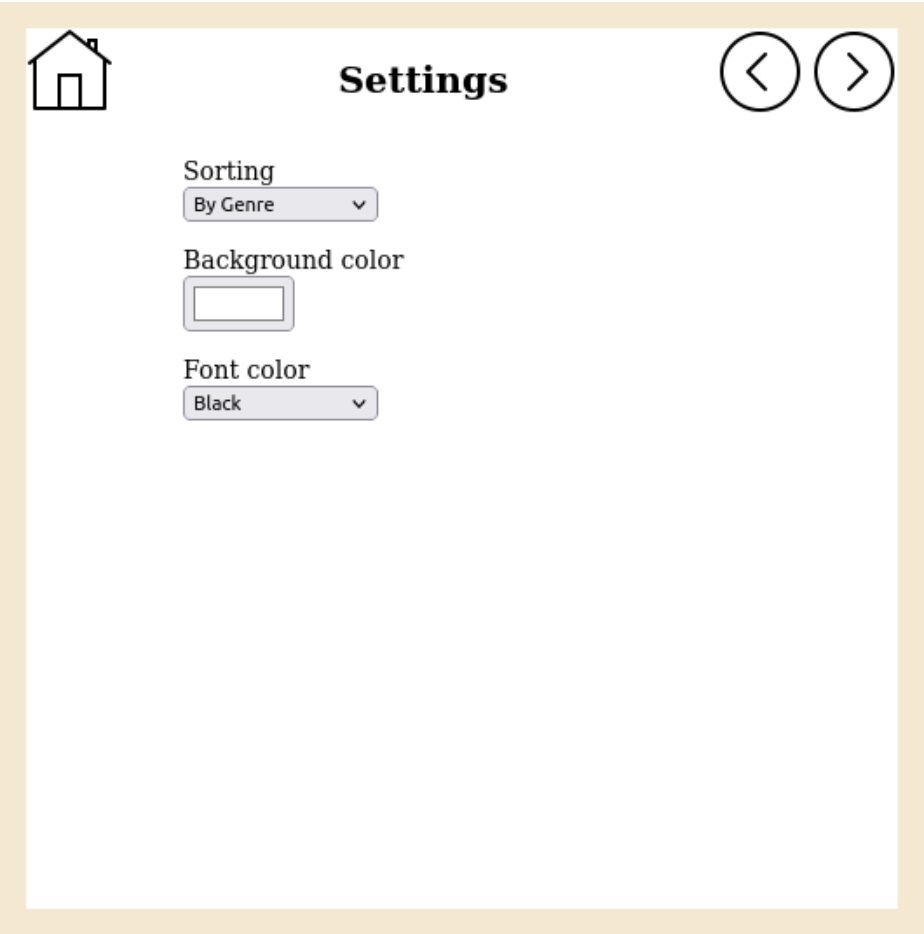


Рисунок 2.12 – Вікно налаштувань програми

При натисканні на книгу відкривається меню книги (рис. 2.13). Меню дозволяє почати читання або переглянути інформацію про книгу.

Рисунок 2.13 – Меню книги

Перегляд інформації про книгу включає відображення таких даних, як

- назва
- Жанр
- авторів
- видавництво
- Рік випуску (рис. 2.14)

Рисунок 2.14 – Перегляд інформації про книгу

Читання книги здійснюється шляхом натискання на кнопку Read меню. Перегортання сторінок здійснюється за допомогою кнопок навігації (рис. 2.15)

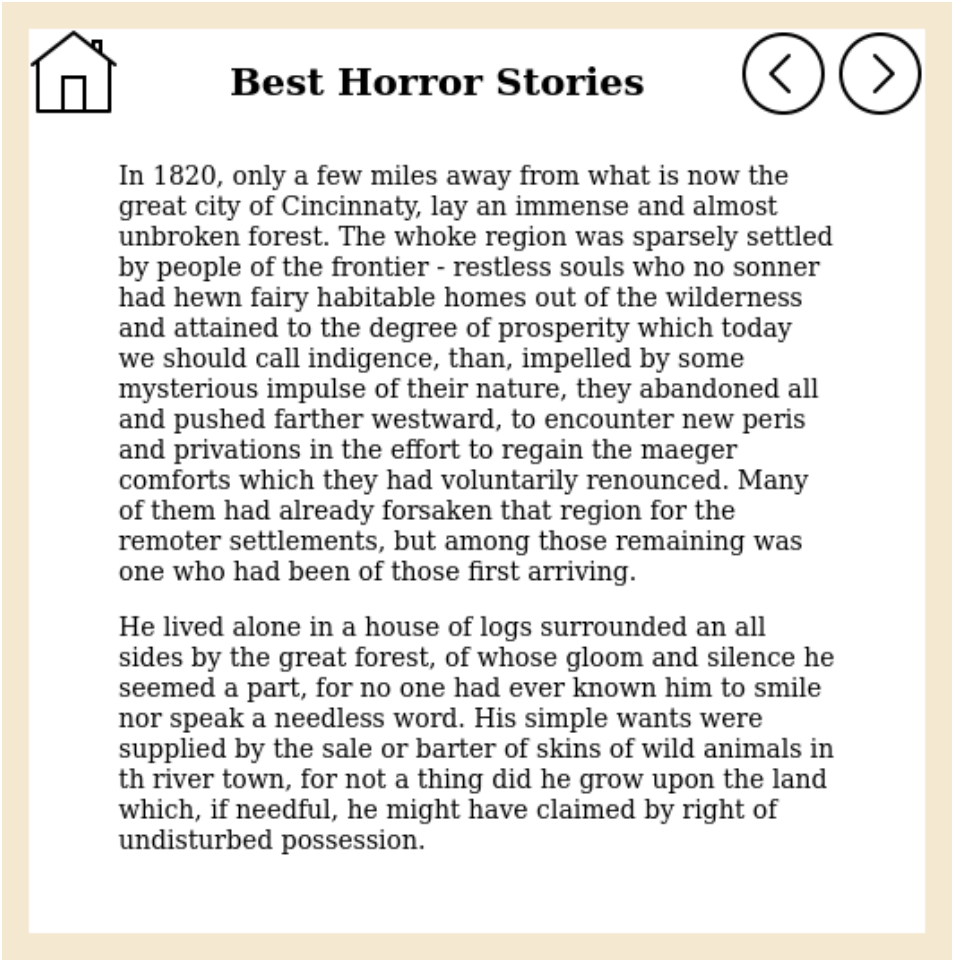


Рисунок 2.15 – Читання книги

При тривалому натисканні в робочому полі під час читання відкриваються налаштування книги (рис. 2.16). Вони дозволяють встановити для конкретної книги колір фону, колір шрифту і розмір шрифту. Для кожної книги ці установки зберігаються.

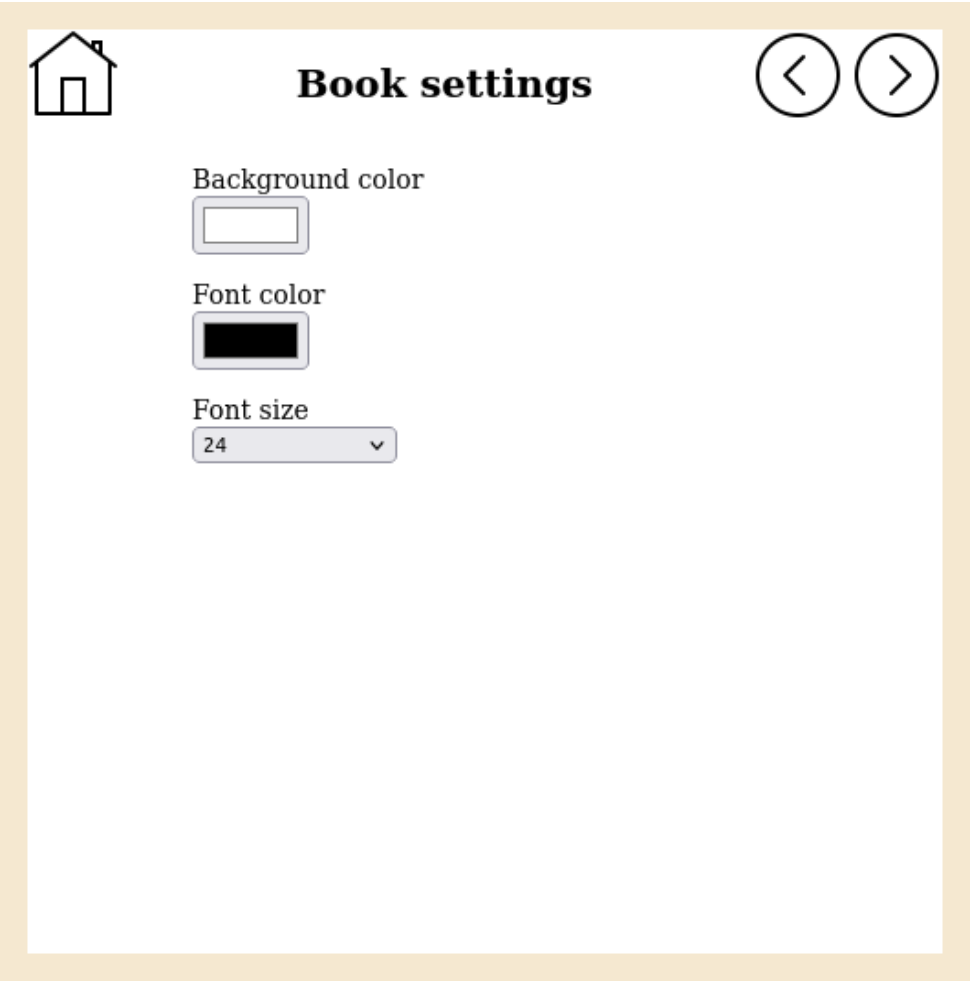


Рисунок 2.16 – Налаштування відображення книги

Таким чином, у цій роботі створено мобільний додаток для читання електронних книг. Програма підтримує формат EPUB і дозволяє читати книги, переглядати інформацію про них. На відміну від аналочичних програм, програма дозволяє задавати настройки відображення для кожної з книг окремо, а також задавати загальні налаштування.

Подальший розвиток програми може включати додавання функції пошуку книг та синхронізацію з веб-сервісами.

3. ТЕХНІКО–ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

Витрати розробку програмного продукту розраховуються за такою формуле:

$$З_{п} = З_{1} + З_{2} + З_{3} + З_{4} + Р_{Н} (1)$$

де $З_{1}$ - загальний фонд оплати праці розробників програмного продукту;

$З_{2}$ - нарахування на заробітну плату розробників у позабюджетні фонди;

$З_{3}$ - Витрати, пов'язані з експлуатацією техніки;

$З_{4}$ – витрати на спеціальні програмні продукти, необхідних розробки ПП;

$Р_{Н}$ - накладні витрати (P_{H} = 30% від $З_{1}$).

Під час розробки програмного продукту загальний час розробки становив 3,5 місяці. З них машинний час (безпосередня робота з обчислювальною та оргтехнікою) становить 2,5 міс.

Фонд оплати праці під час роботи над програмним продуктом:

$$З_{1} = \sum_{j=1}^{m} Opj * Трпрj * (1 + kд)(1 + ky)(2)$$

де Opj- оклад j -го розробника. У розробці брала участь 1 особа, її оклад становить 4744 грн.;

Трпрj– загальний час роботи над ПР у місяцях,

$kд$- Коефіцієнт додаткової зарплати, ($kд = 0{,}1$);

ky- Районний коефіцієнт, ($ky = 1{,}5$) .

Таким чином,

$$З_{1} = 4744 * 3{,}5 * (1 + 0{,}1) * (1 + 1{,}5) = 45665грн$$

$$З_2 = 0{,}302 * З_1 = 0{,}302 * 45665 = 13790 \text{ грн}$$

Витрати, пов'язані з амортизацією для комп'ютера вартістю 25000 грн. та терміном амортизації рівним 25 місяців, становитимуть 1000 грн. на рік або 83 грн. на місяць.

Витрати, пов'язані з використанням обчислювальної та оргтехніки:

$$З_З = Т_{МРПР} * k_Г * n * С_{МЧ}$$

де $k_Г$ - Коефіцієнт готовності ЕОМ ($k_Г = 0{,}095$);

n – кількість одиниць техніки, що дорівнює 1;

$С_{МЧ}$ - собівартість машино-години, ($С_{МЧ} = 4$ грн);

$Т_{МРПР}$ - машинний час роботи над програмним продуктом, що дорівнює 2,5 міс.

Таким чином,

$$З_З = 440 * 0{,}95 * 1 * 4 = 1672 \text{ грн}$$

З урахуванням амортизації за 2,5 місяці витрати становитимуть 1672+418 = 2090 грн.

Витрати на спеціальні програмні продукти, необхідні для розробки ПП, включають витрати на програмне середовище, таким чином

$$З_4 = 25000 \text{ грн}$$

Накладні витрати:

$$Р_Н = З_1 * k_{НР} = 45\,665 * 0{,}3 = 13\,699 \text{ грн.}$$

Таким чином, витрати на розробку програмного продукту становитимуть:

$$Зп = 45665 + 13790 + 1672 + 25000 + 13699 = 99\ 826 \text{ грн}$$

ВИСНОВОК

У цій випускній роботі було розроблено мобільний додаток для читання електронних книг. Програма розроблена мовою Python за допомогою фреймворку Kivy, як середовище розробки використовувалася Android Studio.

Робота складається із трьох розділів, у яких послідовно вирішуються поставлені завдання.

У першому розділі були розглянуті концепції розробки мобільних додатків за допомогою мови Python та класифікації типів електронних книг. Проведено огляд існуючих рішень програм для читання електронних книг. Прикладами може бути Amazon Kindle, Google Play Books, Kobo Books. Вони підтримують різні формати електронних книг, функції пошуку інформації та інтеграцію із популярними онлайн-бібліотеками. Їх загальним недоліком є неможливість настроювання параметрів, таких як шрифт, колір фону та тексту для кожної книги окремо. З проведеного огляду сформульовані завдання проектування. Для розробки мобільного додатка використовуватимемо фреймворк Kivy. Kivy є досить популярним на даний момент проектом, що забезпечує достатню кількість прикладів та довідкової інформації. Також він дозволяє максимально використовувати ресурси мобільного пристрою та використовує вільну систему ліцензування. Програма повинна працювати з відкритим стандартом EPUB та забезпечувати читання книг та навігацію.

У другому розділі описано особливості розробки програми, включаючи опис інтерфейсу, розробку бази даних, особливості розробки програми операційної системи Android.

На підставі проведеного аналізу предметної області сформовано інфологічну модель бази даних, з якої надалі були створені логічна та фізична моделі. Показано реалізацію фізичної моделі в Python і Kivy.

Інтерфейс програми складається з набору вікон, які забезпечують виконання наступних функцій:

- Відображення списку книг
- Сортування книг
- відображення інформації про книгу
- Читання книги
- налаштування зовнішнього вигляду програми
- Налаштування зовнішнього вигляду інтерфейсу для кожної книги.

У третьому розділі наведено підрахунок економічної ефективності проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Абрамов, Г.В. Проектування інформаційних систем/Г.В. Абрамов, І.Є. Медведкова, Л.А. Коробова – Воронеж: ВДУІТ, 2018 – 172 с.

2. Дарвін, Я.Ф. Android. Збірник рецептів. Завдання та рішення для розробників додатків/Я.Ф. Дарвін. - СпБ.: ТОВ Альфа-книга, 2018. - 768 с.

3. Альошин, Л.І. Інформаційні технології: Навчальний посібник/Л.І. Альошин - М.: Маркет ДС, 2016. - 384 с.

4. Вейл, Е. HTML5. Розробка програм для мобільних пристроїв / Е. Вейл – СПБ.: Пітер, 2015. – 480 с.

5. Воронкова, Ю.Б. Інформаційні технології в освіті/Ю.Б. Воронкова. - РНД: Фенікс, 2010. - 314 с.

6. Філіпс, Б. Android. Програмування для професіоналів 3-тє видання/ Б. Філліпс, К. Стюарт, К. Марсікано. - СПБ.: Пітер, 2017. - 688 с.

7. Голіцина, О.Л. Інформаційні технології: Підручник/О.Л. Голіцина, Н.В. Максимов, Т.Л. Партика, І.І. Попов. - М.: Форум, ІНФРА-М, 2017. - 608 с.

8. Грекул, В.І. Проектування інформаційних систем. Навчальний курс/В.І. Грекул

9. Абельсон, Х.; Сассман, Дж. Структура и интерпретация компьютерных программ; М.: Добросвет - Москва, 2012. - 608 с.

10. Котляров, В. П. Основы тестирования программного обеспечения / В.П. Котляров, Т.В. Коликова. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2013. - 288 с.

11. Програмування числових методів мовою Python / А. В.Анісімов, А. Ю.

12. Дорошенко, С. Д. Погорілий, Я. Ю. Дорогий. – Київ: Видавничо-поліграфічний центр "Київський університет", 2014. – 640 с.

13. Пишем игру на Python [Електронний ресурс] // Режим доступу:https://thecode.media/pygames/

14. Гультяєв, А.К. Проектування та дизайн інтерфейсу / А.К. Гультяєв – СПб: Корона-принт, 2019 – 349 с.

15. Дубейківський, В.І. Практика функціонального моделювання з AllFusion Process Modeler 7. Де? Навіщо? Як? / В.І. Дубейковський - М.: Діалог - МІФІ, 2014. - 464 с.

16. Захарова, І.Г. Інформаційні технології в освіті: / І.Г. Захарова. - М.: Academia, 2016. - 543 с.

17. Хорстман, К. Java. Бібліотека професіонала/К. Хорстманн. - М.: ТОВ І. Д. Вільямс, 2016. - 864 с.

18. Гріффідс, Д. Head First. Програмування для Android. 2-е вид / Д. Гріффідс, Д. Гріффідс. - СПБ.: Пітер, 2018. - 912 с.

19. Федотова, Є.Л. Інформаційні технології в науці та освіті: Навчальний посібник/Є.Л. Федотова, А.А. Федотов. - М.: Форум, 2018. - 256 с.

20. Федотова, Є.Л. Інформаційні технології та системи: Навчальний посібник/Є.Л. Федотова. - М: ІД ФОРУМ, 2017. - 352 с

21. Хлєбніков, А.А. Інформаційні технології: Підручник/О.О. Хлібніков. - М.: КноРус, 2019. - 472 с .

ДОДАТКИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка мобільного додатку для читання електронних книг мовою Python

Виконав студент 4 курсу
групи ПД-44
Слободянюк Олег Ігорович
Керівник роботи
к.т.н., доцент
Трінтіна Наталія Альбертівна

Київ – 2022

Таблиця порівнянь аналогів

Назва	Переваги	Недоліки
Amazon Kindle	1. Можливість змінювати мову словника 2. Зручна навігація 3. Функція пояснення важких для розуміння слів	1. Словник треба завантажувати окремо
Google Play Books	1. Підтримує аудіокниги 2. Доступ як до онлайн так і оффлайн словника 3. Функція адаптивного тону дисплею, в залежності від навколишнього освітлення 4. Синхронізація заміток та бібліотеки з Google Drive	1. Не підтримує формат MOBI 2. Відсутність можливості листати сторінки

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: Підвищення зручності читання електронних книг у мобільному додатку

Об’єкт дослідження: Процес читання електронних книжок

Предмет дослідження: Програмне забезпечення для читання електронних книг

3

Технічне завдання

1. Реалізація навігації по книгам.
2. Реалізація підтримки відкритого стандарту EPUB.
3. Реалізація налаштувань стилю окремо для кожної книги.
4. Реалізація сортування книжок.
5. Реалізація перегляду інформації про книгу.

4

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

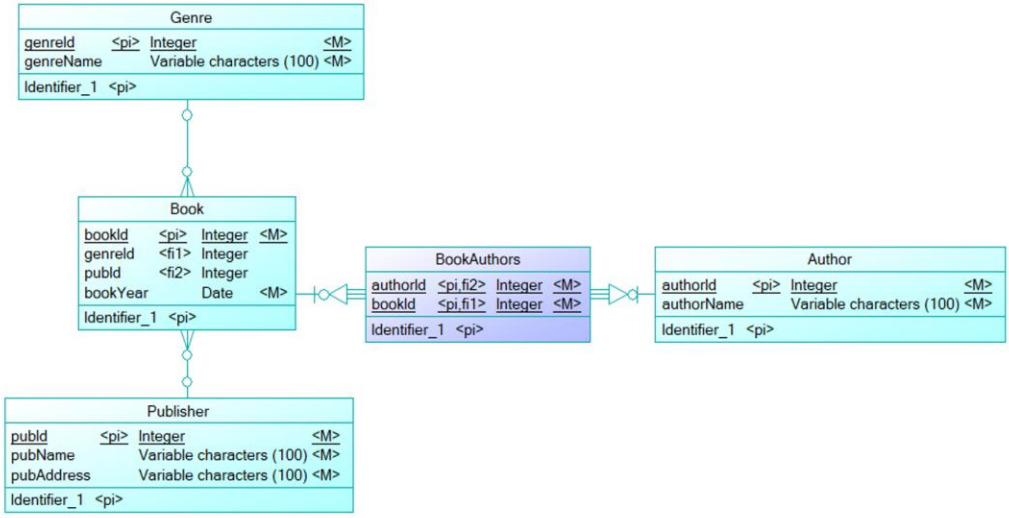
Python

Android Studio

Kivy

5

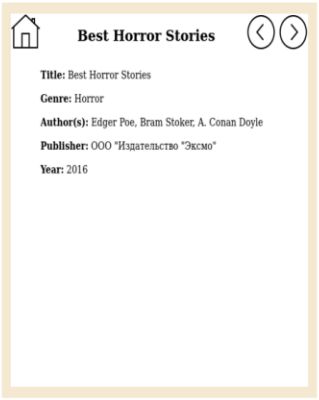
Логічна модель бази даних



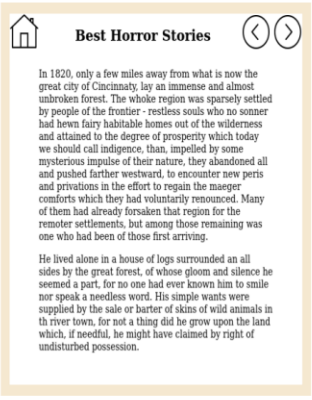
6

Екранні форми додатку

Екран – головне меню



Екран – інформація про книгу



Екран – читання книжки

7

ВИСНОВКИ

1. Проаналізовано існуючі додатки, з метою виявлення недоліків та досліджено існуючі варіанти розробки за цим напрямком. З проведеного огляду сформульовані завдання проектування.
2. Розроблено архітектуру електронної бібліотеки.
3. Розроблено інтерфейс програми.
4. Розроблено функцій процесу читання (навігація, сортування, налаштування стилю).

9

```
import './fonts/Montserrat/stylesheet-montserrat.css'
import './fonts/Caveat/caveat.css'
import './fonts/Forum/forum.css'
//Импорт компонентов библиотеки реакт-роутера
import {BrowserRouter as Router, Switch, Route} from "react-router-dom";
```

```
//Основная функция приложения. Данный компонент рендерится в файле index.js
function App() {
   /*Функция возвращает JSX-код и рендерит в HTML*/
 return (
    <div className="App">
       {/*Блок прелоадера*/}
       <div id="loading">
          <PulseLoader color="#FFFFFF" loading/>
       </div>
       {/*Хэдэр*/}
       <Header/>
       {/*Блок смены и отображения подгружаемых компонентов*/}
       <Router>
          <Switch>
            {/*В path записывается тот путь, при котором будет отображаться
 компонент из атрибута component. Данный путь равен значению атрибута
 href в пункте 2 в файле "shablon-group.js" или "shablon-games.js"*/}
              <Route exact path="/" component={Home}></Route>
              <Route exact path="/notes" component={Notes}></Route>
              <Route exact path="/sources" component={Sources}></Route>
              <Route exact path="/group_1" component={Group_1}></Route>
              <Route exact path="/group_1/1" component={Group_1_1_Games}></Route>
            <Route exact path="/group_1/1/1" component={Group_1_1_Games_1}></Route>
            <Route exact path="/group_1/1/2" component={Group_1_1_Games_2}></Route>
            <Route exact path="/group_1/1/3" component={Group_1_1_Games_3}></Route>
            <Route exact path="/group_1/1/4" component={Group_1_1_Games_4}></Route>
            <Route exact path="/group_1/1/5" component={Group_1_1_Games_5}></Route>
            <Route exact path="/group_1/2" component={Group_1_2_Games}></Route>
            <Route exact path="/group_1/2/1" component={Group_1_2_Games_1}></Route>
            <Route exact path="/group_1/2/2" component={Group_1_2_Games_2}></Route>
            <Route exact path="/group_1/2/3" component={Group_1_2_Games_3}></Route>
            <Route exact path="/group_1/2/4" component={Group_1_2_Games_4}></Route>
            <Route exact path="/group_1/2/5" component={Group_1_2_Games_5}></Route>
```

```
<Route exact path="/group_1/3" component={Group_1_3_Games}></Route>
<Route exact path="/group_1/3/1" component={Group_1_3_Games_1}></Route>
<Route exact path="/group_1/3/2" component={Group_1_3_Games_2}></Route>
<Route exact path="/group_1/3/3" component={Group_1_3_Games_3}></Route>
<Route exact path="/group_1/3/4" component={Group_1_3_Games_4}></Route>
<Route exact path="/group_1/3/5" component={Group_1_3_Games_5}></Route>
<Route exact path="/group_1/4" component={Group_1_4_Games}></Route>
<Route exact path="/group_1/4/1" component={Group_1_4_Games_1}></Route>
<Route exact path="/group_1/4/2" component={Group_1_4_Games_2}></Route>
<Route exact path="/group_1/4/3" component={Group_1_4_Games_3}></Route>
<Route exact path="/group_1/4/4" component={Group_1_4_Games_4}></Route>
<Route exact path="/group_1/4/5" component={Group_1_4_Games_5}></Route>
<Route exact path="/group_1/4/6" component={Group_1_4_Games_6}></Route>
<Route exact path="/group_1/5" component={Group_1_5_Games}></Route>
<Route exact path="/group_1/5/1" component={Group_1_5_Games_1}></Route>
<Route exact path="/group_1/5/2" component={Group_1_5_Games_2}></Route>
<Route exact path="/group_1/5/3" component={Group_1_5_Games_3}></Route>
<Route exact path="/group_1/5/4" component={Group_1_5_Games_4}></Route>
<Route exact path="/group_1/5/5" component={Group_1_5_Games_5}></Route>
<Route exact path="/group_1/6" component={Group_1_6_Games}></Route>
<Route exact path="/group_1/6/1" component={Group_1_6_Games_1}></Route>
<Route exact path="/group_1/6/2" component={Group_1_6_Games_2}></Route>
<Route exact path="/group_1/6/3" component={Group_1_6_Games_3}></Route>
<Route exact path="/group_1/6/4" component={Group_1_6_Games_4}></Route>
<Route exact path="/group_1/6/5" component={Group_1_6_Games_5}></Route>
<Route exact path="/group_1/7" component={Group_1_7_Games}></Route>
<Route exact path="/group_1/7/1" component={Group_1_7_Games_1}></Route>
<Route exact path="/group_1/7/2" component={Group_1_7_Games_2}></Route>
<Route exact path="/group_1/7/3" component={Group_1_7_Games_3}></Route>
<Route exact path="/group_1/7/4" component={Group_1_7_Games_4}></Route>
<Route exact path="/group_1/7/5" component={Group_1_7_Games_5}></Route>
<Route exact path="/group_1/8" component={Group_1_8_Games}></Route>
<Route exact path="/group_1/8/1" component={Group_1_8_Games_1}></Route>
```

```
<Route exact path="/group_1/8/2" component={Group_1_8_Games_2}></Route>
<Route exact path="/group_1/8/3" component={Group_1_8_Games_3}></Route>
<Route exact path="/group_1/8/4" component={Group_1_8_Games_4}></Route>
<Route exact path="/group_1/8/5" component={Group_1_8_Games_5}></Route>
<Route exact path="/group_1/9" component={Group_1_9_Games}></Route>
<Route exact path="/group_1/9/1" component={Group_1_9_Games_1}></Route>
<Route exact path="/group_1/9/2" component={Group_1_9_Games_2}></Route>
<Route exact path="/group_1/9/3" component={Group_1_9_Games_3}></Route>
<Route exact path="/group_1/9/4" component={Group_1_9_Games_4}></Route>
<Route exact path="/group_1/9/5" component={Group_1_9_Games_5}></Route>
<Route exact path="/group_2" component={Group_2}></Route>
<Route exact path="/group_2/1" component={Group_2_1_Games}></Route>
<Route exact path="/group_2/1/1" component={Group_2_1_Games_1}></Route>
<Route exact path="/group_2/1/2" component={Group_2_1_Games_2}></Route>
<Route exact path="/group_2/1/3" component={Group_2_1_Games_3}></Route>
<Route exact path="/group_2/1/4" component={Group_2_1_Games_4}></Route>
<Route exact path="/group_2/1/5" component={Group_2_1_Games_5}></Route>
<Route exact path="/group_2/2" component={Group_2_2_Games}></Route>
<Route exact path="/group_2/2/1" component={Group_2_2_Games_1}></Route>
<Route exact path="/group_2/2/2" component={Group_2_2_Games_2}></Route>
<Route exact path="/group_2/2/3" component={Group_2_2_Games_3}></Route>
<Route exact path="/group_2/2/4" component={Group_2_2_Games_4}></Route>
<Route exact path="/group_2/2/5" component={Group_2_2_Games_5}></Route>
<Route exact path="/group_2/3" component={Group_2_3_Games}></Route>
<Route exact path="/group_2/3/1" component={Group_2_3_Games_1}></Route>
<Route exact path="/group_2/3/2" component={Group_2_3_Games_2}></Route>
<Route exact path="/group_2/3/3" component={Group_2_3_Games_3}></Route>
<Route exact path="/group_2/3/4" component={Group_2_3_Games_4}></Route>
<Route exact path="/group_2/3/5" component={Group_2_3_Games_5}></Route>
<Route exact path="/group_2/4" component={Group_2_4_Games}></Route>
<Route exact path="/group_2/4/1" component={Group_2_4_Games_1}></Route>
<Route exact path="/group_2/4/2" component={Group_2_4_Games_2}></Route>
<Route exact path="/group_2/4/3" component={Group_2_4_Games_3}></Route>
```

```
<Route exact path="/group_2/4/4" component={Group_2_4_Games_4}></Route>
<Route exact path="/group_2/4/5" component={Group_2_4_Games_5}></Route>
<Route exact path="/group_2/5" component={Group_2_5_Games}></Route>
<Route exact path="/group_2/5/1" component={Group_2_5_Games_1}></Route>
<Route exact path="/group_2/5/2" component={Group_2_5_Games_2}></Route>
<Route exact path="/group_2/5/3" component={Group_2_5_Games_3}></Route>
<Route exact path="/group_2/5/4" component={Group_2_5_Games_4}></Route>
<Route exact path="/group_2/5/5" component={Group_2_5_Games_5}></Route>
<Route exact path="/group_2/6" component={Group_2_6_Games}></Route>
<Route exact path="/group_2/6/1" component={Group_2_6_Games_1}></Route>
<Route exact path="/group_2/6/2" component={Group_2_6_Games_2}></Route>
<Route exact path="/group_2/6/3" component={Group_2_6_Games_3}></Route>
<Route exact path="/group_2/6/4" component={Group_2_6_Games_4}></Route>
<Route exact path="/group_2/6/5" component={Group_2_6_Games_5}></Route>
<Route exact path="/group_2/7" component={Group_2_7_Games}></Route>
<Route exact path="/group_2/7/1" component={Group_2_7_Games_1}></Route>
<Route exact path="/group_2/7/2" component={Group_2_7_Games_2}></Route>
<Route exact path="/group_2/7/3" component={Group_2_7_Games_3}></Route>
<Route exact path="/group_2/7/4" component={Group_2_7_Games_4}></Route>
<Route exact path="/group_2/7/5" component={Group_2_7_Games_5}></Route>
<Route exact path="/group_2/8" component={Group_2_8_Games}></Route>
<Route exact path="/group_2/8/1" component={Group_2_8_Games_1}></Route>
<Route exact path="/group_2/8/2" component={Group_2_8_Games_2}></Route>
<Route exact path="/group_2/8/3" component={Group_2_8_Games_3}></Route>
<Route exact path="/group_2/8/4" component={Group_2_8_Games_4}></Route>
<Route exact path="/group_2/8/5" component={Group_2_8_Games_5}></Route>
<Route exact path="/group_2/9" component={Group_2_9_Games}></Route>
<Route exact path="/group_2/9/1" component={Group_2_9_Games_1}></Route>
<Route exact path="/group_2/9/2" component={Group_2_9_Games_2}></Route>
<Route exact path="/group_2/9/3" component={Group_2_9_Games_3}></Route>
<Route exact path="/group_2/9/4" component={Group_2_9_Games_4}></Route>
<Route exact path="/group_2/9/5" component={Group_2_9_Games_5}></Route>
<Route exact path="/group_3" component={Group_3}></Route>
```

```
<Route exact path="/group_3/1" component={Group_3_1_Games}></Route>
<Route exact path="/group_3/1/1" component={Group_3_1_Games_1}></Route>
<Route exact path="/group_3/1/2" component={Group_3_1_Games_2}></Route>
<Route exact path="/group_3/1/3" component={Group_3_1_Games_3}></Route>
<Route exact path="/group_3/1/4" component={Group_3_1_Games_4}></Route>
<Route exact path="/group_3/1/5" component={Group_3_1_Games_5}></Route>
<Route exact path="/group_3/2" component={Group_3_2_Games}></Route>
<Route exact path="/group_3/2/1" component={Group_3_2_Games_1}></Route>
<Route exact path="/group_3/2/2" component={Group_3_2_Games_2}></Route>
<Route exact path="/group_3/2/3" component={Group_3_2_Games_3}></Route>
<Route exact path="/group_3/2/4" component={Group_3_2_Games_4}></Route>
<Route exact path="/group_3/2/5" component={Group_3_2_Games_5}></Route>
  <Route exact path="/group_3/3" component={Group_3_3_Games}></Route>
<Route exact path="/group_3/3/1" component={Group_3_3_Games_1}></Route>
<Route exact path="/group_3/3/2" component={Group_3_3_Games_2}></Route>
<Route exact path="/group_3/3/3" component={Group_3_3_Games_3}></Route>
<Route exact path="/group_3/3/4" component={Group_3_3_Games_4}></Route>
<Route exact path="/group_3/3/5" component={Group_3_3_Games_5}></Route>
<Route exact path="/group_3/4" component={Group_3_4_Games}></Route>
<Route exact path="/group_3/4/1" component={Group_3_4_Games_1}></Route>
<Route exact path="/group_3/4/2" component={Group_3_4_Games_2}></Route>
<Route exact path="/group_3/4/3" component={Group_3_4_Games_3}></Route>
<Route exact path="/group_3/4/4" component={Group_3_4_Games_4}></Route>
<Route exact path="/group_3/4/5" component={Group_3_4_Games_5}></Route>
  <Route exact path="/group_3/5" component={Group_3_5_Games}></Route>
<Route exact path="/group_3/5/1" component={Group_3_5_Games_1}></Route>
<Route exact path="/group_3/5/2" component={Group_3_5_Games_2}></Route>
<Route exact path="/group_3/5/3" component={Group_3_5_Games_3}></Route>
<Route exact path="/group_3/5/4" component={Group_3_5_Games_4}></Route>
<Route exact path="/group_3/5/5" component={Group_3_5_Games_5}></Route>
<Route exact path="/group_3/6" component={Group_3_6_Games}></Route>
<Route exact path="/group_3/6/1" component={Group_3_6_Games_1}></Route>
<Route exact path="/group_3/6/2" component={Group_3_6_Games_2}></Route>
```

```
<Route exact path="/group_3/6/3" component={Group_3_6_Games_3}></Route>
<Route exact path="/group_3/6/4" component={Group_3_6_Games_4}></Route>
<Route exact path="/group_3/6/5" component={Group_3_6_Games_5}></Route>
 <Route exact path="/group_3/7" component={Group_3_7_Games}></Route>
<Route exact path="/group_3/7/1" component={Group_3_7_Games_1}></Route>
<Route exact path="/group_3/7/2" component={Group_3_7_Games_2}></Route>
<Route exact path="/group_3/7/3" component={Group_3_7_Games_3}></Route>
<Route exact path="/group_3/7/4" component={Group_3_7_Games_4}></Route>
<Route exact path="/group_3/7/5" component={Group_3_7_Games_5}></Route>
 <Route exact path="/group_3/8" component={Group_3_8_Games}></Route>
<Route exact path="/group_3/8/1" component={Group_3_8_Games_1}></Route>
<Route exact path="/group_3/8/2" component={Group_3_8_Games_2}></Route>
<Route exact path="/group_3/8/3" component={Group_3_8_Games_3}></Route>
<Route exact path="/group_3/8/4" component={Group_3_8_Games_4}></Route>
<Route exact path="/group_3/8/5" component={Group_3_8_Games_5}></Route>
  <Route exact path="/group_4" component={Group_4}></Route>
  <Route exact path="/group_4/1" component={Group_4_1_Games}></Route>
<Route exact path="/group_4/1/1" component={Group_4_1_Games_1}></Route>
<Route exact path="/group_4/1/2" component={Group_4_1_Games_2}></Route>
<Route exact path="/group_4/1/3" component={Group_4_1_Games_3}></Route>
<Route exact path="/group_4/1/4" component={Group_4_1_Games_4}></Route>
  <Route exact path="/group_4/2" component={Group_4_2_Games}></Route>
<Route exact path="/group_4/2/1" component={Group_4_2_Games_1}></Route>
<Route exact path="/group_4/2/2" component={Group_4_2_Games_2}></Route>
<Route exact path="/group_4/2/3" component={Group_4_2_Games_3}></Route>
<Route exact path="/group_4/2/4" component={Group_4_2_Games_4}></Route>
<Route exact path="/group_4/3" component={Group_4_3_Games}></Route>
<Route exact path="/group_4/3/1" component={Group_4_3_Games_1}></Route>
<Route exact path="/group_4/3/2" component={Group_4_3_Games_2}></Route>
<Route exact path="/group_4/3/3" component={Group_4_3_Games_3}></Route>
<Route exact path="/group_4/3/4" component={Group_4_3_Games_4}></Route>
<Route exact path="/group_4/4" component={Group_4_4_Games}></Route>
<Route exact path="/group_4/4/1" component={Group_4_4_Games_1}></Route>
```

```
<Route exact path="/group_4/4/2" component={Group_4_4_Games_2}></Route>
<Route exact path="/group_4/4/3" component={Group_4_4_Games_3}></Route>
<Route exact path="/group_4/4/4" component={Group_4_4_Games_4}></Route>
<Route exact path="/group_4/4/5" component={Group_4_4_Games_5}></Route>
<Route exact path="/group_4/5" component={Group_4_5_Games}></Route>
<Route exact path="/group_4/5/1" component={Group_4_5_Games_1}></Route>
<Route exact path="/group_4/5/2" component={Group_4_5_Games_2}></Route>
<Route exact path="/group_4/5/3" component={Group_4_5_Games_3}></Route>
<Route exact path="/group_4/5/4" component={Group_4_5_Games_4}></Route>
<Route exact path="/group_4/5/5" component={Group_4_5_Games_5}></Route>
<Route exact path="/group_4/5/6" component={Group_4_5_Games_6}></Route>
<Route exact path="/group_4/6" component={Group_4_6_Games}></Route>
<Route exact path="/group_4/6/1" component={Group_4_6_Games_1}></Route>
<Route exact path="/group_4/6/2" component={Group_4_6_Games_2}></Route>
<Route exact path="/group_4/6/3" component={Group_4_6_Games_3}></Route>
<Route exact path="/group_4/6/4" component={Group_4_6_Games_4}></Route>
<Route exact path="/group_4/6/5" component={Group_4_6_Games_5}></Route>
<Route exact path="/group_4/7" component={Group_4_7_Games}></Route>
<Route exact path="/group_4/7/1" component={Group_4_7_Games_1}></Route>
<Route exact path="/group_4/7/2" component={Group_4_7_Games_2}></Route>
<Route exact path="/group_4/7/3" component={Group_4_7_Games_3}></Route>
<Route exact path="/group_4/7/4" component={Group_4_7_Games_4}></Route>
<Route exact path="/group_4/7/5" component={Group_4_7_Games_5}></Route>
<Route exact path="/group_4/8" component={Group_4_8_Games}></Route>
<Route exact path="/group_4/8/1" component={Group_4_8_Games_1}></Route>
<Route exact path="/group_4/8/2" component={Group_4_8_Games_2}></Route>
<Route exact path="/group_4/8/3" component={Group_4_8_Games_3}></Route>
<Route exact path="/group_4/8/4" component={Group_4_8_Games_4}></Route>
<Route exact path="/group_4/8/5" component={Group_4_8_Games_5}></Route>
{/*Сюда добавлять новые страницы,
где path="предполагаемый путь к странице",
а component="название компонента"*/}
</Switch>
```

```
        </Router>
      </div>
  );}
function onReady(callback) {
  var intervalId = window.setInterval(function() {
    if (document.getElementsByTagName('body')[0] !== undefined) {
      window.clearInterval(intervalId);
      callback.call(this);
    }
  }, 1000);}
function setVisible(selector, visible) {
  document.querySelector(selector).style.opacity = visible ? '1' : '0';
  document.querySelector(selector).style.zIndex = visible ? '100' : '-1';
}
onReady(function() {
  setVisible('.App', true);
  setVisible('#loading', false);
});
export default App;
```

app.scss

```
a{
  text-decoration: none;
}
#loading {
  z-index: 100;
  width: 100vw;
  position: absolute;
  height: 100vh;
  background-color: #57D3FF;
  transition: 0.5s;
  background-repeat: no-repeat;
  background-position: center;
  display: flex;
```

```
  justify-content: center;
  align-items: center;
}
*{
  margin: 0;
  padding: 0
}
.header{
  position: fixed;
  top: 0;
  left: 0;
  width: 100vw;
  height: 3.6875rem;
  background: #414141;
  display: flex;
  align-items: center;
}
```