Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Катасонова Кирила Сергійовича*
(ПІБ)

академічної групи *122-18ск-1*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка мобільного додатку «Бібліотека» для ОС Андроїд на мові програмування Python з використанням фреймворку Django*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Гуліна І.Г.*			
розділів:				
спеціальний	*доц. Гуліна І.Г.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент	*доц. Шедловський І.А.*			
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18ск-1* *Катасонова Кирила Сергійовича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка мобільного додатку «Бібліотека» для ОС Андроїд на мові програмування Python з використанням фреймворку Django*

затверджена наказом ректора НТУ «ДП» від «07 » 06. 2021 р. № 317-с

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	*13.05.2021 р.*
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2021 р.***

Завдання видав _____ *доц. Гуліна І.Г.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Катасонов К.С.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *11.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *14.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка: 81 с., 19 рис., 3 дод., 26 джерел.

Об'єкт розробки: програмне забезпечення застосунку для мобільних пристроїв для роботи з онлайн бібліотекою.

Мета кваліфікаційної роботи: створення програмного забезпечення, призначеного для роботи з онлайн бібліотекою адаптованої англомовної літератури під ОС Android.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування застосунку, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження застосунку, описана робота програми.

В економічному розділі визначено трудомісткість розробленого програмного продукту, проведений підрахунок вартості роботи по створенню застосунку та розраховано час на його створення.

Практичне значення полягає у створенні застосунку, що надає користувачу простий у використанні і зручний інструмент для отримання можливості читання електроних книг та роботи з віддаленою онлайн бібліотекою на мобільному пристрої.

Актуальність програмного продукту визначається великим попитом на подібні інструменти, що надають користувачам можливість віддаленого доступу до електроної бібліотеки та роботи з адаптованою англомовною літературою, орієнтованою на україномовного читача.

Список ключових слів: БІБЛІОТЕКА, МОБІЛЬНИЙ ПРИСТРІЙ, КЛІЄНТ, СЕРВЕР, ЕКРАН, ЗАПИТ, ANDROID, REST, API.

ABSTRACT

Explanatory note: 81 pages, 19 figs., 3 apps., 26 sources.

The object of development: mobile application for working with the online library.

The purpose of the graduation project: creation of the software designed to working with the online library adapted English literature using an Android device.

In introduction the current state of the problem was examined, the purpose of the qualification work and the field of its application were specified, the relevance of the topic was justified, and the problem statement was specified.

In the first section the status of the problem was reviewed, analytical overview of the methods and tools of application analysis and design was justified, the choice of system architecture and design solutions used was done.

In the second section the features of a software implementation were overviewed, the background information and application functionality were provided, a description of the system user interface was made.

In economic section labour input and duration of software product development was defined, the estimation of expenses for its creation was executed.

The practical significance of the project is to create a software that provides the user with an easy-to-use and easy-to-use tool for reading e-books and working with a remote online library on a mobile device.

The relevance of the software product is determined by the high demand for similar tools that provide users with the possibility of remote access to the e-library and working with adapted English-language literature oriented to the Ukrainian-language reader.

Keywords: LIBRARY, MOBILE DEVICE, CLIENT, SERVER, SCREEN, REQUEST, ANDROID, REST, API.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

API - інтерфейс для програмування застосунків;

БД - база даних;

ЕОМ - електронно-обчислювальна машина;

ІС - інформаційна система;

ООП - об'єктно-орієнтоване програмування;

ОС - операційна система;

ПЗ - програмне забезпечення;

ІТ - інформаційні технології.

ВСТУП

Мобільний сегмент ринку активно розвивається, а поява і розвиток таких платформ, як iOS і Android, змінюють економічні правила і змушують по-новому поглянути на бізнес з позиції цілих галузей. Сучасній людині тепер зручно не тільки використовувати мобільні пристрої для серфінгу по Інтернету, але і оплачувати з їх допомогою покупки, послуги і вирішувати безліч інших повсякденних завдань. Для максимального використання потенціалу мобільної аудиторії важливим завданням є створення мобільного застосунку для сайту інтернет-магазину, інтернет-порталу або сервісу. Подібні застосунки встановлюються безпосередньо на смартфон або планшет користувача, що потенційно забезпечує інтерактивність і більш високий рівень залучення клієнтів.

Зі зростанням числа користувачів комп'ютерів та Інтернету все більша кількість людей починає користуватися електронними книгами. Крім безлічі спеціалізованих пристроїв для читання з технологією E-ink, існує велика кількість застосунків, що перетворюють планшет або смартфон на пристрій для читання. Величезний плюс електронних бібліотек - це доступність. Необхідну книгу можна отримати моментально, а не їхати до книжкового магазину, не маючі оперативної інформації про її наявність. Завдяки мобільним застосункам з'явилася можливість швидкого придбання і завантаження файлу з електронною книгою безпосередньо на смартфон або планшет користувача.

Найчастіше вибір застосунку, з яким буде комфортно проводити час за читанням, є нелегким завданням для користувача, оскільки число зручних електронних бібліотек з розробленим мобільним застосунком, особливо орієнтованих на специфічні жанри або галузі, в даний момент обмежена. Таким чином, актуальною є розробка застосунку для мобільних пристроїв, призначеного для роботи з онлайн бібліотекою адаптованої англомовної літератури.

Завдання даної кваліфікаційної роботи та об'єкт її діяльності безпосередньо пов'язані зі спеціальностю 122 «Комп’ютерні науки» та відповідає узагальненій тематиці кваліфікаційних робіт і переліку зазначених компетенцій.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Повільно, але невпинно популярність ресурсів з легальним контентом підвищується. Цьому чимало сприяє посилення законодавства України в області авторських прав. Однак можна стверджувати, що культура споживання змінюється і завдяки розвитку технологій, мобільного інтернету та інтернет-сервісів. Попри значну кількість безкоштовні ресурсів із нелегальним контентом, існують і популярні мобільні застосунки, доступ до яких надається за підпискою. Нижче поданий стислий огляд найбільш відомих представників цього типу застосунків.

iBooks

ОС: iOS (з версії iOS 8 – є частиною операційної системи).

Вартість: безкоштовно.

Можливості:

- ✓ Підтримує формати pdf та ebook.
- ✓ Можливість налаштування яскравості, розміру шрифта та фону книги.
- ✓ Навігація: прокрутка тексту, гортання сторінок.
- ✓ Нічний режим читання.
- ✓ Можливість пошуку невідомого слова у вбудованому словнику або в Google чи Wikipedia.
- ✓ Можливість придбання книг у магазині застосунку, але на теперішній час доступні тільки англомовні видання.

Переваги: простий та зручний інтерфейс.

Недоліки: невилика кількість підтримуваних форматів, тільки англомовна література у магазині.

Google Play Книги

ОС: Android.

Вартість: безкоштовно.

Можливості:

- ✓ Підтримує більшість популярних форматів електроних книг.
- ✓ Можливість налаштування яскравості, розміру шрифта та фону книги.
- ✓ Навігація: прокрутка тексту, гортання сторінок.
- ✓ Нічний режим читання.
- ✓ Можливість створення нотатків із різним кольоровим виділенням, або закладок.
- ✓ Функція озвучення тексту книги.
- ✓ Можливість завантаження книжок до хмарного сервісу через веб-сайт, з подальшим завантаженням їх на мобільний пристрій.
- ✓ Наявність магазину книжок з доволі великим каталогом багатомовної літератури.

Переваги: велика кількість функцій, зручний інтерфейс, аудіо режим.

Недоліки: відсутність функціоналу з перекладу чи тлумачення слів.

Marvin

ОС: iOS.

Вартість: $3.99, є безкоштовна пробна версія.

Можливості:

- ✓ Підтримує тільки формат epub.
- ✓ Налаштування кольору фона, яскравості, розміру та типу шрифта.
- ✓ Робота з каталогами електроних бібліотек, пошук і придбання книг.
- ✓ Інтеграція з Dropbox – можливість завантажувати книги з хмари.
- ✓ Функція перекладу слів за допомогою Google Translate чи ABBYY Lingvo.
- ✓ Можливість відправки фрагментві тексту до соціальних мереж, отримання додаткової інформації з Google та Wikipedia.
- ✓ Режим Deep View, що дозволяє знаходити у тексті книги власні імена, та проводити по ним навігацію.

- ✓ Можливість отримання довідок про дійові особи книги, місця чи імена (тільки для англомовних текстів).

Переваги: Можливість не тільки читати, але і досліджувати тексти книг.

Недоліки: вартість; мало підтримуваних форматів.

Bookmate

ОС: iOS, Android.

Вартість: за підпискою, є безкоштовна версія.

Можливості:

- ✓ Підтримує формати fb2 та epub.
- ✓ Налаштування виду та розміру шрифта, яскровості екрану.
- ✓ Навігація: прокрутка тексту, гортання сторінок.
- ✓ Можливісить завантаження книг із застосунку на мобільний пристрій користувача.
- ✓ Підтримка синхронізації процесу читання (якщо користувач читає книгу на різних пристроях).
- ✓ Можливість виділяти фрагменти тексту і створювати з ними замітки, ділитися цитатами в соцмережах або відправляти їх по WhatsApp або іншому месенджеру.
- ✓ Можливість створювати книжкові колекції - «книжкові полиці», спостерігати за уподобаннями інших зареєстрованих користувачів застосунку.
- ✓ Книжковий магазин з онлайн-каталогом різноманітної літератури (пріорітетна – російська мова, але доступна також велика кількість іншомовних книжок).
- ✓ Декілька варіантів підписки різної вартості та обсягу отримуваного контенту.

Переваги: доступ до ліцензійного контенту через підписку; можливість читання різного обсягу книг за фіксовану ціну.

Недоліки: у першу чергу, застосунок є книжковим магазином, що обумовлює його обмежений функціонал.

Cool Reader

ОС: Android.

Вартість: безкоштовно.

Можливості:

- ✓ Підтримка великої кількості форматів електроних книг: fb2, txt, rtf, doc, epub, chm, pdb, prc, mobi.
- ✓ Можливість завантаження книг через вбудований диспетчер файлівю
- ✓ Доступ до онлайн-бібліотек, перелік яких користувач може редагувати самостійно.

Переваги: найбільш великий функціонал серед подібних застосунків, підтримувані формати книг.

Недоліки: складний інтерфейс.

Moon+ Reader

ОС: Android.

Вартість: платна та безкоштовна версії.

Можливості:

- ✓ Підтримка форматів електроних книг: fb2, epub, mobi, html, cbz, chm, cbr, umd, txt, rar, zip.
- ✓ Налаштування кольору, інтервалів, відступів, вирівнювання тексту.
- ✓ Декілька варіантів гортання сторінок: гортання, прокручування, по рядкам, по сторінкам.
- ✓ Більш ніж 10 встановлених тем з денним та нічним режимами читання.
- ✓ Можливість створювати закладки.
- ✓ Можливість використання тлумачного словника.
- ✓ Доступ до онлайн-бібліотек.

Переваги: простий та зручний інтерфейс, велика кількість функцій.

Недоліки:реклама у безкоштовній версії.

Але попри велику кількість функцій, всі вищезгадані програми не надають можливості роботи з адаптованої англомовною літературою, орієнтованою на україномовного читача. Таким чином, актуальною видається розробка

застосунку для мобільних пристроїв, призначеного для роботи з онлайн бібліотекою адаптованої англомовної літератури.

1.2. Призначення розробки та галузь застосування

Як об'єкт впровадження розроблюваного застосунку розглядається онлайн бібліотека Readdo, інформаційне наповнення якої скероване на навчання англійській мові за допомогою текстового та аудіо контенту. Інші потенційні споживачі продукту, що розробляється, - електронні бібліотеки, які надають доступ до різножанрової літератури за підпискою.

Розроблений застосунок призначений для:

a) організації послідовного і структурованого сприйняття адаптованих книг англійською мовою;
b) надання доступу до каталогу електронної бібліотеки;
c) придбання та завантаження електронних книг на мобільний пристрій;
d) читання завантажених у застосунок книг у режимі офлайн.

1.3. Підстава для розробки

Отже, підставами для розробки «виконання кваліфікаційної роботи» є:

- освітня програма 122 «Комп’ютерні науки»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» №317-с від 07.06.2021р.;
- завдання на кваліфікаційну роботу на тему «Розробка мобільного додатку «Бібліотека» для ОС Андроїд на мові програмування Python з використанням фреймворку Django».

1.4. Постановка завдання

Завданням кваліфікаційної роботи є розробка застосунку для мобільних пристроїв, призначеного для роботи з онлайн бібліотекою адаптованої англомовної літератури.

Програма, що розробляється, повинна реалізувати наступні функції:

- відображення змісту каталогу онлайн бібліотеки;
- завантаження файлів електронних книг на мобільний пристрій;
- відображення змісту файлів текстових форматів на екран пристрою;
- відтворення файлів аудіо книг;
- налаштування візуальних параметрів відображення тексту у режимі читання.

Для досягнення поставленої мети необхідно:

- вивчити предметну галузь розв'язуваної задачі;

- провести порівняльний аналіз можливостей аналогічних застосунків;

- вибрати раціональну структуру і параметри програми;

- розробити структуру вхідних і вихідних даних для проектованого програмного забезпечення;

- написати програмний код застосунку;

- розробити рекомендації щодо застосування програми.

Дана програма повинна являти собою мобільний застосунок під ОС Android.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для досягнення поставлених цілей програмне забезпечення, що розробляється, повинно підтримувати виконання наступних дій:

- внесення і редагування інформації в базу даних онлайн бібліотеки;

- обмін GET і POST запитами між застосунком і сервером онлайн бібліотеки;
- внесення і редагування інформації на накопичувачі мобільного пристрою;
- авторизація і аутентифікація користувачів;
- робота з файловою системою пристрою;
- здійснення профіля користувача.

Для виконання перерахованих вище функцій у застосунку повинні бути реалізовані:

- можливість інсталяції під ОС Android;
- підтримка мережевих протоколів REST-архітектури;
- наявність типової конфігурації, що забезпечує можливість швидкого введення застосунку в експлуатацію;
- програмно-апаратна переносимість.

1.5.2 Вимоги до інформаційної безпеки

Для уникнення некоректної роботи програми необхідно реалізувати:

- контроль вхідних даних;
- обробку виняткових ситуацій;
- виведення повідомлень про помилки;
- можливість повторного введення даних;
- можливість безперервної роботи протягом не менше 120 годин (5 діб);
- платформну незалежність;
- вірогідність виникнення не більше 2 логічних помилок на 1000 операторів за 1 рік експлуатації;
- забезпечення неушкодженого стану даних, що зберігаються в базі даних, у випадку відмови застосунку.

1.5.3 Вимоги до складу та параметрів технічних засобів

Для нормального функціонування програми необхідно, щоб мобільний пристрій, на якому буде функціонувати застосунок, відповідав наступним вимогам:

- ✓ процесор класу Qualcomm Snapdragon або Intel Atom з кількістю ядер не менше двох;
- ✓ не менше 1 GB оперативної пам'яті;
- ✓ не менше 64 GB вбудованої пам'яті;
- ✓ мережева карта Wi-Fi.

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

1.5.4 Вимоги до інформаційної та програмної сумісності

Для нормального функціонування програми необхідно, щоб програмне забезпечення мобільного пристрою, на якому буде функціонувати застосунок, відповідало наступним вимогам:

- ✓ операційна система Android версії 4.3 та новіше.

Застосунок має бути реалізовано на мові програмування Python з використанням фреймворку Django. У якості СУБД має бути використана MySQL.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

В рамках даної кваліфікаційної роботи була виконана розробка застосунку для мобільних пристроїв для роботи з онлайн бібліотекою адаптованої англомовної літератури.

Призначення розробленого застосунку:

- організація послідовного і структурованого сприйняття адаптованих книг англійською мовою;
- надання доступу до каталогу електронної бібліотеки;
- придбання та завантаження електронних книг на мобільний пристрій;
- читання завантажених у застосунок книг у режимі офлайн.

Розроблена програма виконує наступні функції:

- відображення змісту каталогу онлайн бібліотеки;
- завантаження файлів електронних книг на мобільний пристрій;
- відображення змісту файлів текстових форматів на екран пристрою;
- відтворення файлів аудіо книг;
- налаштування візуальних параметрів відображення тексту у режимі читання.

Для досягнення поставленої задачі розроблене програмне забезпечення підтримує виконання таких операцій:

- внесення і редагування інформації в базу даних онлайн бібліотеки;
- обмін GET і POST запитами між застосунком і сервером онлайн бібліотеки;
- внесення і редагування інформації на накопичувачі мобільного пристрою;
- авторизація і аутентифікація користувачів;

- робота з файловою системою пристрою;
- здійснення профіля користувача.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної галузі розв'язуваної задачі не передбачають застосування математичних методів, при розробці застосунку для роботи з онлайн бібліотекою математичні методи не використовувалися.

2.3. Опис використаних технологій та мов програмування

Для створення застосунку для роботи з онлайн бібліотекою була використана концепція "Model-View-Template (MVT)". Вона має багато спільного з більш відомої архітектурою Model-View-Controller. Контролер класичної моделі MVC приблизно відповідає рівню, який в MVT називається уявлення (англ. View), а презентаційна логіка уявлення реалізується в MVT рівнем шаблонів (англ. Template).

Архітектура MVT складається з чотирьох основних компонентів (рис. 2.1):

1) модель даних (Model) є серцевиною будь-якого сучасного веб-застосунку. Модель - це найважливіша частина програми, яка постійно звертається до даних будь-який запит з будь-якої сесії. Будь-яка модель є стандартним Python класом. Об'єктно-орієнтована mapper (ORM) забезпечує доступ таких класів безпосередньо до баз даних. Якби не було ORM, довелося б писати запити безпосередньо на SQL. Модель забезпечує полегшений механізм доступу до шару даних, інкапсулює бізнес-логіку. Модель не залежить від конкретного застосунка. Даними можна маніпулювати навіть з командного рядка, не використовуючи при цьому веб-сервер;

2) уявлення (представлення, View) виконують різноманітні функції, в тому числі контролюють запити користувача, видають контекст в залежності від

його ролі. View - це звичайна функція, яка викликається у відповідь на запит якогось адреси (URL) і повертає контекст;

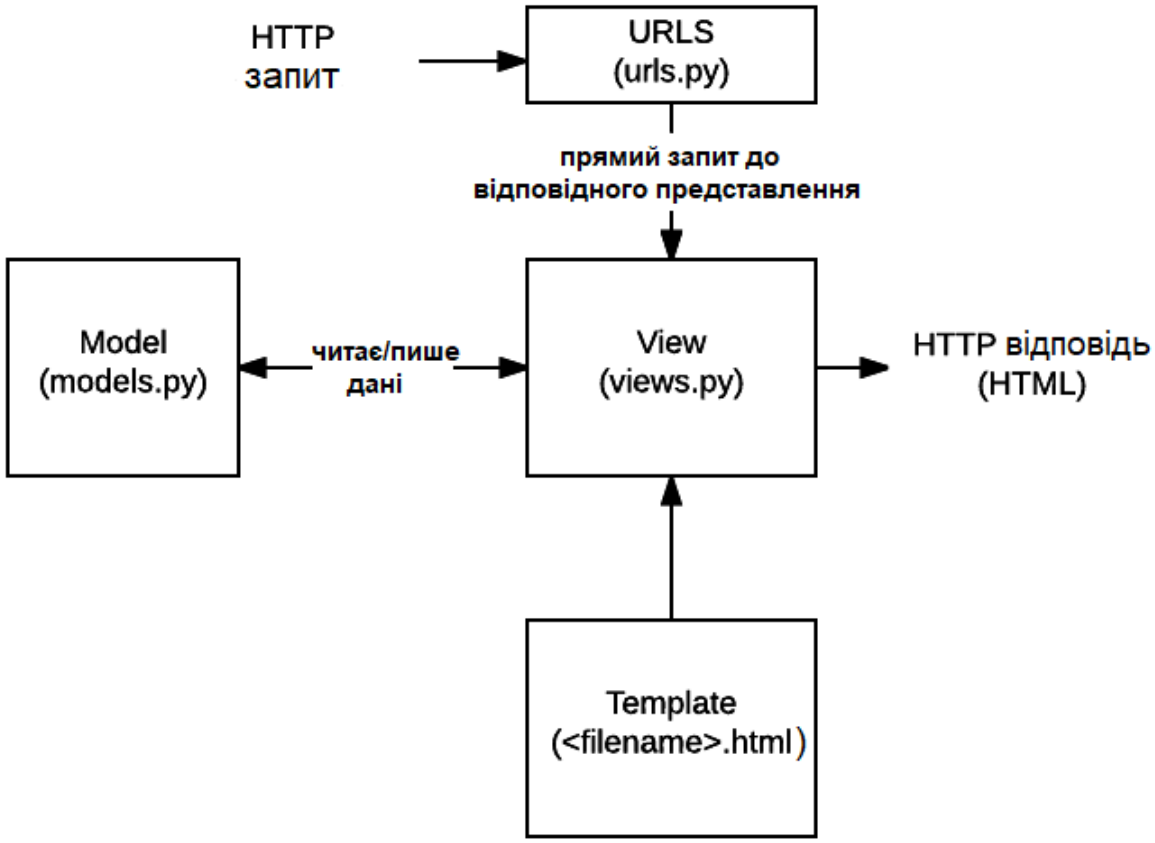


Рис. 2.1. Взаємодія компонентів архітектури MVT

3) шаблони (Template) є формою представлення даних. Шаблони мають свою власну просту метамову і є одним з основних засобів виведення на екран;

4) URL-механізм зовнішнього доступу до уявлень (view). Вбудовані в URL регулярні вирази роблять механізм досить гнучким. При цьому одне представлення може бути налаштоване до кількох URL, надаючи доступ різним застосункам. Тут підтримується філософія закладок: URL стають самодостатніми і починають жити незалежно від уявлення.

Для відображення результатів в розмітку HTML, яка часто є кінцевим результатом роботи програми, використовується мова шаблонів. Шаблони, по суті, є текстовими документами в форматі HTML, зі спеціальним форматуванням там, куди виводяться значення, одержувані динамічно, вони підтримують можливість використовувати прості логічні конструкції, такі як

цикли та інші. Коли від уявлення потрібно повернути документ HTML, воно зазвичай вказує шаблон, передає в нього інформацію для відображення і використовує відображений шаблон в своїй відповіді.

При роботі з СУБД дані завантажуються в пам'ять програми, з ними або над ними здійснюються дії і результат дій, як правило, повинен бути збережений назад. Весь цей час необхідно зберігати інформацію про зміни над даними, щоб знати, що змінилося. Крім цього необхідно зберігати інформацію про створені та віддалених об'єктах.

Застосунок реалізовано на мові програмування Python з використанням фреймворку Django. У якості СУДБ була обрана СУДБ MySQL.

Python — інтерпретована об'єктно-орієнтована мова програмування високого рівня з строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання існуючих компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Серед основних її переваг можна назвати такі:

- чистий синтаксис (для виділення блоків слід використовувати відступи);
- переносність програм (що властиве більшості інтерпретованих мов);
- стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
- можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);

- стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і яке написане на мові Python;
- зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);
- відкритий код (можливість редагувати його іншими користувачами).

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

Інтерпретатор мови Python і багата стандартна бібліотека (як вихідні тексти, так і бінарні дистрибутиви для всіх основних операційних систем) можуть бути отримані з сайту Python www.python.org, і можуть вільно розповсюджуватися. Цей самий сайт має дистрибутиви та посилання на численні модулі, програми, утиліти та додаткову документацію.

Інтерпретатор мови Python може бути розширений функціями та типами даних, розробленими на C чи C++ (або на іншій мові, яку можна викликати із C). Python також зручна як мова розширення для прикладних програм, що потребують подальшого налагодження.

Python портований і працює майже на всіх відомих платформах — від КПК до мейнфреймів. Існують порти під Microsoft Windows, всі варіанти UNIX (включаючи FreeBSD та GNU/Linux), Plan 9, Mac OS та Mac OS X, iPhone OS 2.0 і вище, Palm OS, OS/2, Amiga, AS/400 та навіть OS/390, Symbian та Android.

Дизайн мови Python побудований навколо об'єктно-орієнтованої моделі програмування. Реалізація ООП в Python є елегантною, потужною та добре продуманою, але разом з тим, достатньо специфічною в порівнянні з іншими об'єктно-орієнтованими мовами.

Можливості та особливості Python:

- класи є одночасно об'єктами з усіма нижче наведеними можливостями;
- успадкування, в тому числі множинне;
- поліморфізм (всі функції віртуальні);
- інкапсуляція (два рівні — загальнодоступні та приховані методи і поля). Особливість — приховані члени доступні для використання та помічені як приховані лише особливими іменами;
- спеціальні методи, що керують життєвим циклом об'єкта: конструктори, деструктори, розподільники пам'яті;
- перевантаження операторів (усіх, крім is, '.', '=' і символьних логічних);
- властивості (імітація поля за допомогою функцій);
- управління доступу до полів (емуляція полів і методів, частковий доступ тощо);
- метапрограмування (управління створенням класів, тригери на створення класів, та ін).

Python підтримує парадигму функціонального програмування, зокрема:

- функція є об'єктом;
- функції вищих порядків;
- рекурсія;
- розвинена обробка списків (спискові вирази, операції над послідовностями, ітератори);
- аналог замикань (closures);
- часткове застосування функції;
- можливість реалізації інших засобів на самій мові (наприклад, каррінг).

Програмне забезпечення (застосунок або бібліотека) на Python оформлюється у вигляді модулів, які у свою чергу можуть бути зібрані в пакунки. Модулі можуть розташовуватися як у каталогах, так і в ZIP-архівах.

Модулі можуть бути двох типів за своїм походженням: модулі, написані на «чистому» Python, і модулі розширення (extension modules), написані на інших мовах програмування.

Фреймворк Django. Django - це високорівнева веб-інфраструктура Python, яка дозволяє швидко створювати безпечні і підтримувані веб-застосунки. Django може бути використаний для створення практично будь-якого типу веб-сайту - від систем управління контентом до соціальних мереж і новинних сайтів. Він може працювати з будь-якої клієнтської платформою і може доставляти контент практично в будь-якому форматі (включаючи HTML, RSS-канали, JSON, XML і так далі).

Django забезпечує безпечний спосіб управління обліковими записами користувачів і паролями, уникаючи поширених помилок, таких як включення інформації про сеанс в файли cookie, де вона вразлива (замість цього файли cookie містять тільки ключ, а фактичні дані зберігаються в базі даних), або зберігання паролів у відкритому вигляді, замість їх хеш.

Хеш пароля - це значення фіксованої довжини, створене шляхом обробки пароля через криптографічну хеш-функцію. Django може перевірити правильність введеного пароля, пропустивши його через хеш-функцію і порівнявши висновок зі збереженим значенням хеша. Завдяки «однобічному» характеру функції, навіть якщо збережене хеш-значення скомпрометовано, зловмисникові буде складно витягти вихідний пароль.

Django забезпечує захист від багатьох вразливостей за замовчуванням, включаючи SQL-ін'єкцію, міжсайтовий скриптинг, підробка міжсайтових запитів і клікджекінг.

Django використовує компонентну "shared-nothing" архітектуру (кожна частина архітектури не залежить від інших, і отже, може бути замінена або змінена при необхідності). Чіткий поділ між різними частинами означає, що вона може масштабуватися для збільшення трафіку шляхом додавання обладнання на будь-якому рівні: кешуються сервери, сервери баз даних або сервери застосунків.

Код Django написаний з використанням принципів і шаблонів дизайну, які заохочують створення підтримуваного і багаторазового коду. Зокрема, він використовує принцип Do not Repeat Yourself (DRY), тому немає непотрібного дублювання, що зменшує кількість коду. Django також сприяє угрупованню пов'язаних функцій в багаторазові «застосунки» і на більш низькому рівні групує пов'язаний код модулів відповідно до Model View Controller (MVC) патерном.

Django написаний на Python, який працює на багатьох платформах. Крім того, Django добре підтримується багатьма постачальниками веб-хостингу, які часто надають певну інфраструктуру і документацію для розміщення сайтів Django.

Django продовжує рости і поліпшуватися з моменту його першого релізу (1.0) у вересні 2008 року до недавно випущеної версії 2.0. У кожній версії додані нові функціональні можливості і виправлені помилки, починаючи від підтримки нових типів баз даних, шаблонизатору і кешування, до додавання «загальних» функцій і класів (які зменшують обсяг коду, який розробники повинні писати для ряду задач програмування).

Django зараз є процвітаючим фреймворком з тисячами користувачів і контрибуторів. Немає ніяких доступних і остаточних оцінок популярності серверних фреймворків. Грунтуючись на кількості великих сайтів, які використовують Django, кількості контрибуторів і кількості людей, що надають як безкоштовну, так і платну підтримку, можна вважати, що Django - популярний фреймворк.

Django використовують такі великі сайти, як Disqus, Instagram, Knight Foundation, MacArthur Foundation, Mozilla, National Geographic, Open Knowledge Foundation, Pinterest, і Open Stack (джерело: домашня сторінка Django).

Система управління базами даних MySQL. MySQL - це популярна система управління базами даних. У реляційній базі даних дані зберігаються в окремих таблицях, завдяки чому досягається виграш в швидкості і гнучкості. Таблиці зв'язуються між собою за допомогою відношень, завдяки чому

забезпечується можливість об'єднувати при виконанні запиту дані з декількох таблиць. SQL як частину системи MySQL можна охарактеризувати як мову структурованих запитів плюс найбільш поширену стандартну мову, яка використовується для доступу до баз даних.

MySQL - це ПЗ з відкритим кодом, кожен користувач може вивчити вихідний код і змінити його відповідно до своїх потреб. Використання програмного забезпечення MySQL регламентується ліцензією GPL (GNU General Public License), http://www.gnu.org/licenses/, в якій зазначено, що дозволяється і забороняється робити з цим програмним забезпеченням в різних ситуаціях.

Веб-програмісти часто віддають перевагу СУБД MySQL, оскільки віна є дуже швидкою, надійною і легкою у використанні. Спочатку сервер MySQL розроблявся для управління великими базами даних з метою забезпечити більш високу швидкість роботи в порівнянні з існуючими на той момент аналогами. І ось уже протягом кількох років даний сервер успішно використовується в умовах промислової експлуатації з високими вимогами. Завдяки своїй доступності, швидкості і безпеки MySQL дуже добре підходить для доступу до баз даних по Internet.

MySQL є системою клієнт-сервер, яка містить багатопотоковий SQL-сервер, що забезпечує підтримку різних обчислювальних машин баз даних, а також кілька різних клієнтських програм і бібліотек, засоби адміністрування і широкий спектр програмних інтерфейсів (API). Поставка сервера MySQL у вигляді багатопотокової бібліотеки, яку можна підключити до призначеного для користувача застосунку, дозволяє отримати компактний, більш швидкий і легкий в управлінні продукт. Доступна також велика кількість програмного забезпечення для MySQL, в більшій частині - безкоштовного.

Структура MySQL трирівнева: бази даних - таблиці - записи. Бази даних і таблиці MySQL фізично представляються файлами з розширеннями frm, MYD, MYI. Логічно - таблиця являє собою сукупність записів. А записи - це сукупність полів різного типу. Ім'я бази даних MySQL унікально в межах

системи, а таблиці - в межах бази даних, поля - в межах таблиці. Один сервер MySQL може підтримувати відразу кілька баз даних, доступ до яких може розмежовуватися логіном і паролем. Знаючи ці логін і пароль, можна працювати з конкретною базою даних. Наприклад, можна створити або видалити в ній таблицю, додати записи і т. Д. Зазвичай ім'я-ідентифікатор і пароль назначаються хостинг провайдерами, які і забезпечують підтримку MySQL для своїх користувачів.

MySQL є повнофункціональної реляційної СУБД, що відрізняється функціональністю і надійністю, а також великими перспективами за рахунок її розширюваності і вільної ліцензії. З огляду на все вищесказане, дана СУДБ була обрана в якості СУБД для розроблюваного застосунку.

2.4. Опис структури програми та алгоритмів її функціонування

Сервер онлайн бібліотеки Readdo, яку обслуговує розроблений застосунок, має публічно доступне REST API для мобільних клієнтів.

REST - (скор. від англ. Representational State Transfer - «передача стану уявлення») це архітектурний стиль взаємодії компонентів розподіленого застосунку в мережі за моделлю клієнт-сервер. Особливості його архітектурного стилю:

- кожна сутність повинна мати унікальний ідентифікатор – URI;
- сутності повинні бути пов'язані між собою;
- для читання і зміни даних повинні використовуватися стандартні методи;
- повинна бути реалізована підтримка декількох типів ресурсів;
- взаємодія має здійснюватися без стану.

Стандартні REST-методи такі:

- GET - отримання даних без їх зміни. Це найбільш популярний і легкий метод. Він тільки повертає дані, а не змінює їх, тому на

клієнті не потрібно піклуватися про те, що дані можуть пошкодитися;

- POST - метод, що припускає вставку нових записів;
- PUT - метод, що припускає зміну існуючих записів;
- PATCH - метод, що припускає зміну ідентифікатора існуючих записів;
- DELETE - метод, що припускає видалення записів.

REST API - це набір віддалених викликів стандартних методів, які повертають дані в певному форматі.

Розроблений застосунок має архітектура REST-клієнта та використовує таку послідовність Service API: Activity -> Service -> Content Provider (рис. 2.2). Activity працює з API Android Servcie. При необхідності послати REST-запит Activity створює Service, Service асинхронно посилає запити до REST-сервера і зберігає результати в Content Provider (sqlite). Activity отримує повідомлення про готовність даних і зчитує результати з Content Provider (sqlite).

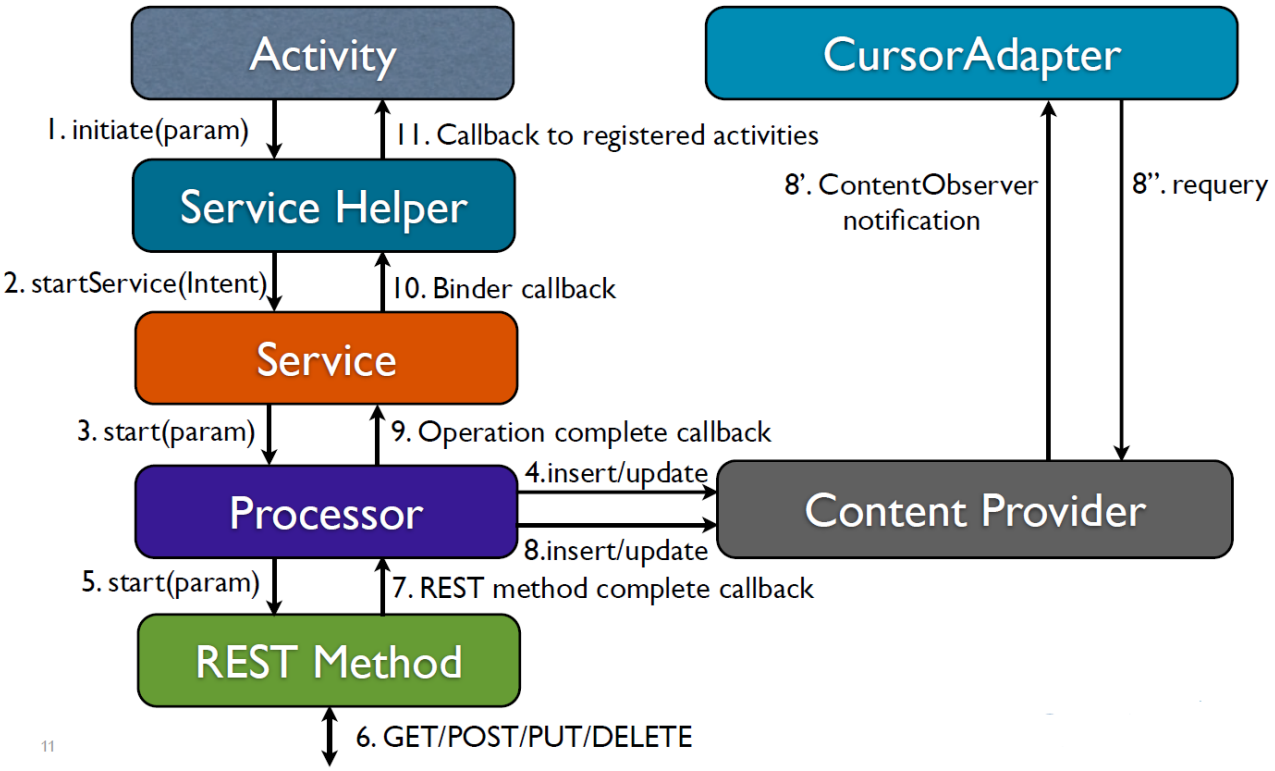


Рис. 2.2. REST-архітектура застосунку

Реалізований у застосунку функціонал REST-клієнта виконує наступні завдання:

- управління сервісом: запуск, зупинка;
- передача результатів з сервісу в Activity;
- кешування результатів в sqlite;
- фіксування статусу даних sqlite перед і після виконання REST-запиту;
- запис інформації про проведені REST-операції в sqlite;
- парсинг отриманих даних;
- конструювання REST-запиту на основі URI і набору параметрів;
- виконання мережевих запитів до REST-сервера (серверу онлайн бібліотеки);
- у разі невдачі REST-запиту, повторення запиту (наприклад, експоненціально збільшуючи час між запитами);
- відкладений запуск REST-запиту через SyncAdapter (використовується для некритичною за часом синхронізації даних між клієнтом і сервером).

Дані, отримані від REST-сервера, завжди зберігаються в sqlite. Безпосередньо в Activity вони ніколи не передаються. Замість цього в Activity передається повідомлення про те, що дані завантажені в sqlite, і їх можна звідти завантажити (варіант - Activity отримує повідомлення про оновлення даних в Content Provider через Content Observer). При виконанні операцій insert, delete, update дані в sqlite оновлюються двічі: перший раз до відправки REST-запиту, другий раз - після отримання результату. Перша операцій виставляє інформаційні прапори, які сигналізують про тип операції, що проводиться над даними, і про статус операції.

REST-методи слід завжди виконувати в окремому потоці.

При виконанні запиту до REST-сервера (серверу онлайн бібліотеки), потрібно виконати ряд операцій:

- сформувати URL;
- задати HTTP-заголовки;
- вибрати тип HTTP-запиту;

- сформувати тіло HTTP-запиту, тобто перетворити Python об'єкт в JSON;
- виконати запит, скориставшись HTTP-клієнтом;
- виконати парсинг результатів запиту - перетворити отриманий JSON в Python об'єкт.

Сервер бібліотеки Readdo відповідає на наступні endpoint-и, або кінцеві точки – спеціальні url для отримання даних або виконання дій:

https://on-the-go.eu/ru/api/booksnew/

https://on-the-go.eu/ru/api/testBooks/

https://on-the-go.eu/ru/api/bookaudio/

https://on-the-go.eu/ru/api/booktext/

https://on-the-go.eu/ru/api/bookaudiosync/

https://on-the-go.eu/ru/api/booklinks/

https://on-the-go.eu/ru/api/bookPreviewLinks/

Endpoint-и 1,2: не приймають ніяких параметрів, реагують тільки на GET запити і повинні повертати json масив такого вигляду:

```
[
 {
      "id": 101,
      "name": "The Canterville ghost",
      "author": "O.Wilde",
      "genre": "Horror",
      "level": "B1",
      "description": "Описание книги",
      "coverLink":
"http://192.168.100.102:8000/Files/readers/101/cover.jpg",
      "languages": [
           "en",
           "ru"
      ],
      "hasDemo": true,
      "pricing": "paid",
      "price": 2.5,
      "isVisible": true,
```

```
        "filesSize": 75777793,
        "filesDuration": 9177792
    }
]
```


де:

name - назва книги латиницею;

level – рівень книги (B1, B2, тощо);

coverLink – посилання на обкладинку книги;

languages- мови, на яких доступні аудіофайли книги (є JSON полем, по факту завжди дорівнює ["en", "ru"]);

hasDemo - булеве поле (true, false), вказує на наявність демо версії книги;

pricing - рядок, що має наступні значення: free, paid, subscription;

price - дійсне число, вартість книги (грає роль в free і paid версії книги. Андроїд клієнт дивиться поле pricing, якщо воно subscription то книга за передплатою, інакше дивиться якщо price = 0 -> вона безкоштовна, інакше -> вартість книги = price);

isVisible - чи є книга видимою;

filesSize - ціле число, сума розмірів англійської та російської чи української аудіодоріжки книги в байтах;

filesDuration - ціле число, сума тривалості аудиодорожек в мілісекундах.

Кінцева точка 3 https://on-the-go.eu/ru/api/bookaudio/

Реагує на POST запит, параметри - form urlencoded:

id - ціле число, id книги,

language - рядок, ru або en

Відповідь – посилання, за яким доступний аудіофайл:


```
{
  "link":
"http://192.168.100.102:8000/Files/readers/102/audio/the_gift_of_the_magi_rus.m4a"
}
```


Endpoint 4 https://on-the-go.eu/ru/api/booktext/

Реагує на POST запит, параметри - form urlencoded:

id - ціле число, id книги,

language - рядок, ru або en.

Відповідь - посилання по якій доступний текст книги

```
{
 "link":
"http://192.168.100.102:8000/Files/readers/102/text/102_text.txt"
}
```

Endpoint 5 https://on-the-go.eu/ru/api/bookaudiosync/

Реагує на POST запит, параметри - form urlencoded:

id - ціле число, id книги,

language - рядок, ru або en.

Відповідь - JSON документ, елемент timings якого складається з цілих чисел, кількості мілісекунд - тимчасових міток, за якими додаток розділяє аудіо на пропозиції.

```
{
 "language": "ru",
 "timings": [
      0,
      2600,
           …...
      934000,
      ]
}
```

Кінцеві точки 6 та 7 https://on-the-go.eu/ru/api/booklinks/ та

https://on-the-go.eu/ru/api/bookPreviewLinks/

Реагує на POST запит, параметри - form urlencoded:

id - ціле число, id книги,

languages - рядок, що містить json масив із затребуваними мовами, наприклад - ["ru", "en"]

```
{
 "audioLinks": {
```

```
        "ru":        "https://on-the-go.eu/Files/readers/101/audio/
the_canterville_ghost_rus.m4a",
        "en":        "https://on-the-go.eu/Files/readers/101/audio/
the_canterville_ghost_eng.m4a"
     },
     "audioSyncs": {
          "ru": [        0,
               5669,
                ……...
               4268180
          ],
          "en": [   0,
               4819,
                ……...
               4902598
          ]
     },
     "textLink":      "https://on-the-go.eu/Files/readers/101/text/
101_text.txt"
    }
```


Для зберігання даних застосунку використовується реляційна база даних, що працює під управлінням СУБД MySQL. На рис.2.3 представлена фізична модель бази даних, що складається з 20 таблиць, наведених до 3-й нормальної форми:

- Auth_user – дані авторизованих користувачів;
- Api_book – каталог книжок;
- Api_author – каталог авторів;
- Api_level – рівні адаптованої літератури;

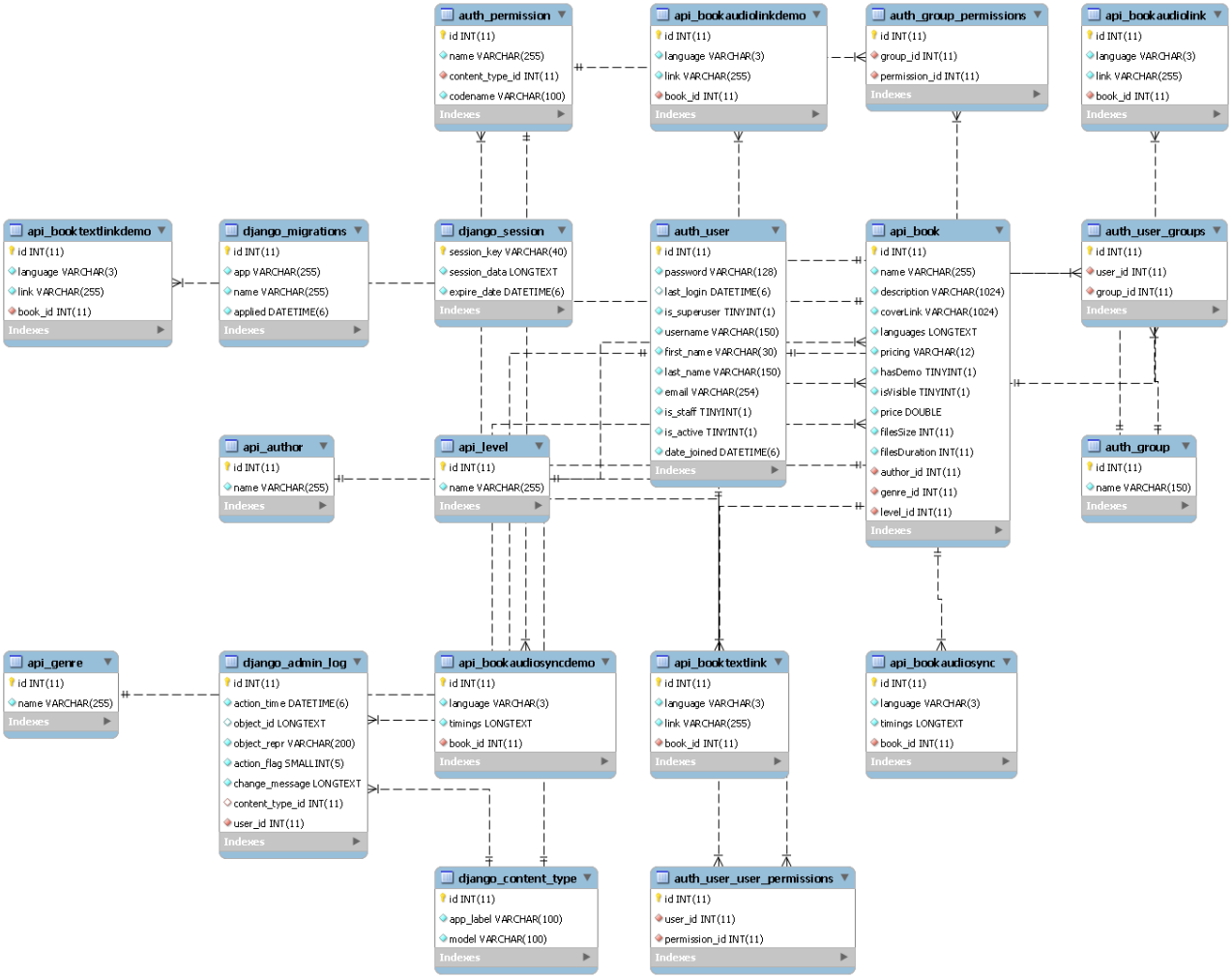


Рис. 2.3. Фізична модель бази даних застосунку

- Api_genre – жанри книг;
- Auth_group – довідник груп;
- Auth_user_groups - групи користувачів;
- Auth_permission – довідник дозволів застосунку (політика розмежування доступу);
- Auth_user_permissions – дозволи користувачів;
- Auth_group_permission – дозволи груп;
- Api_bookaudiolink – каталог посилань на аудиофайли;
- Api_bookaudiolinkdemo – каталог посилань на демо-версії аудиофайлів;
- Api_booktextlink - каталог посилань на тексти електроних книг;

- Api_booktextlinkdemo- каталог посилань на демо-версії текстів електроних книг;
- Api_bookaudiosync – каталог посилань на додаткові аудиодоріжки;
- Api_ bookaudiosyncdemo – каталог посилань на демо-версії додаткових аудиодоріжок;
- Django_session – інформація про сесії застосунку;
- Django_migrations – дані для здійснення міграції даних застосунку;
- Django_content_type – довідник типів контенту;
- Django_admin_log – журнал подій застосунку.

2.5. Обгрунтування та організація вхідних та вихідних даних програми

Вхідні дані:

- посилання на текстові та аудиофайли на сервері онлайн бібліотеки;
- JSON документи;
- натискання на візуальні елементи управління застосунку;
- інформація, що вводиться користувачем із клавіатури;
- файли електроних книжок, завантажені на жорсткий диск мобільного пристрою.

Вихідні дані:

- REST запити;
- записи до бази даних застосунку.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Для нормального функціонування програми необхідно, щоб обчислювальна машина, на якій буде функціонувати застосунок, відповідав наступним вимогам:

- ✓ процесор класу Qualcomm Snapdragon або Intel Atom з кількістю ядер не менше двох;
- ✓ не менше 1 GB оперативної пам'яті;
- ✓ не менше 64 GB вбудованої пам'яті;
- ✓ мережева карта Wi-Fi.

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

2.6.2. Використані програмні засоби

Проект реалізований на мові програмування Python у середовищі розробки Django з використанням СУБД MySQL.

Необхідні програмні засоби для мобільного пристрою, на якому буде функціонувати застосунок:

- ✓ операційна система Android версії 4.3 та новіше.

2.6.3. Виклик та завантаження програми

Розроблений застосунок завантажується шляхом одноразового натискання на піктограму застосунку у меню або на робочому столі Android-пристрою, на якому він встановлена. Для встановлення звязку між застосунком та онлайн бібліотекою необхідно, щоб пристрій, на якому функціонує застосунок, був ввімкнений та підключений до мережі Wi-Fi або мобільного інтернету.

2.6.4. Опис інтерфейсу користувача

На початку роботи із застосунком користувачу за замовчуванням надається екран «книжкової полиці» - колекції завантажених книжок. Якщо на даний

момент вона не містить жодної книги, на екрані буде повідомлення з пропозицією відвідати магазин (рис. 2.4). Під магазином розуміється каталог онлайн-бібліотеки, який містить як безкоштовні, так і платні видання, які можна придбати через разову покупку, або на умовах платної підписки.

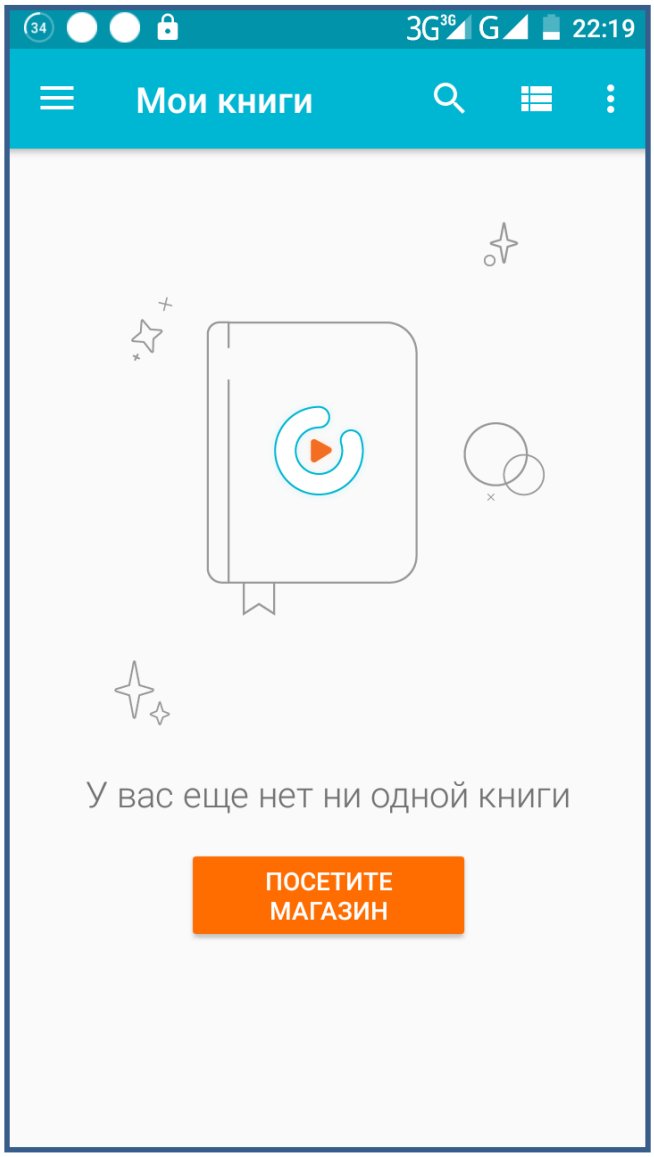


Рис. 2.4. Книжкова полиця

Користувач може перейти до інших екранів застосунку, скориставшись меню (рис. 2.5), яке містить наступні пункти: «Мої книги», «Магазин», «Підписка», «Поділитися», «Налаштування».

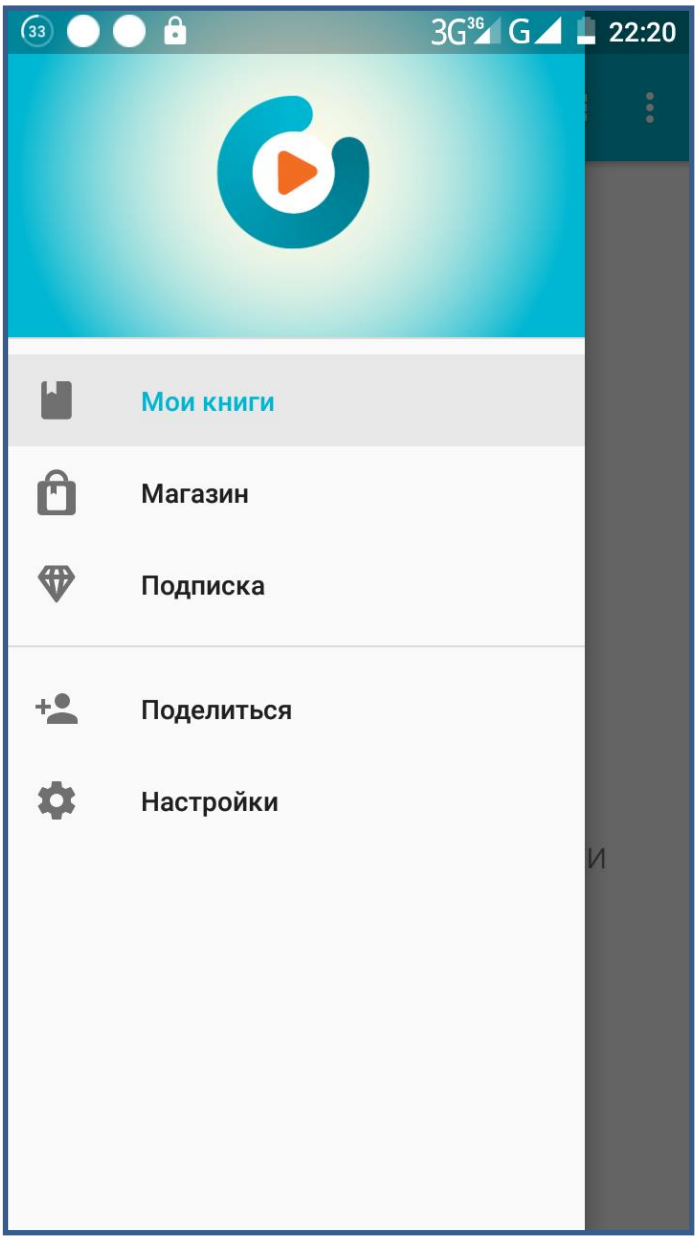
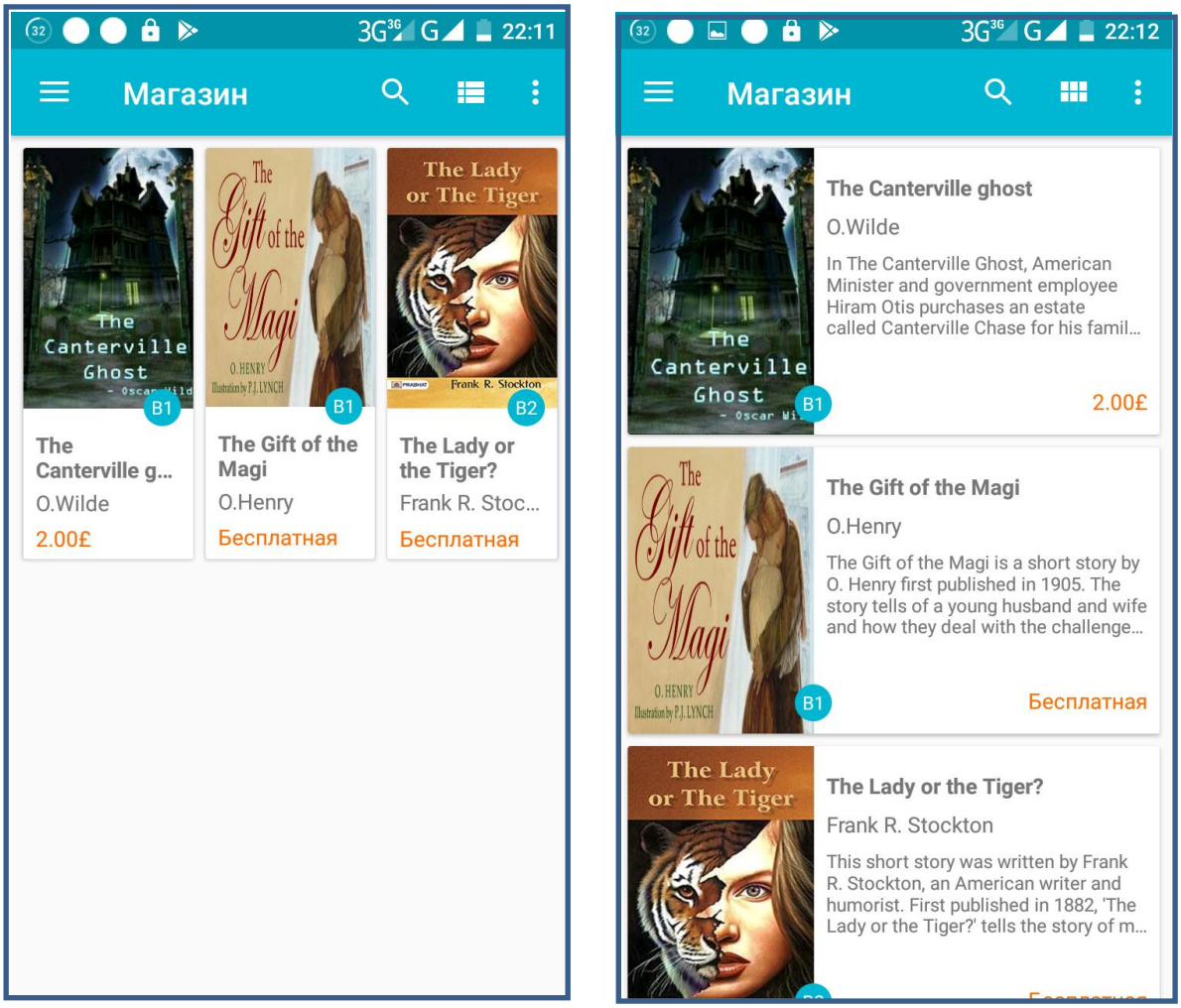


Рис. 2.5 Головне меню

Екран «Магазин» містить плитковий каталог книжок бібліотеки, вид відображення якого – горизонтальне чи вертикальне, користувач може обирати самостійно (рис. 2.6 а-б). Опис книги містить зображення обкладинки, назву, автора, рівень складності та ціну.



а б

Рис. 2.6. Каталог бібліотеки

Також застосунок підтримує функції сортування та фільтрації книг у бібліотеці, що доступні у меню екрану «Магазин» (рис. 2.7), і дозволяють отримувати добірки книжок відповідно до потреб та уподовань читача.

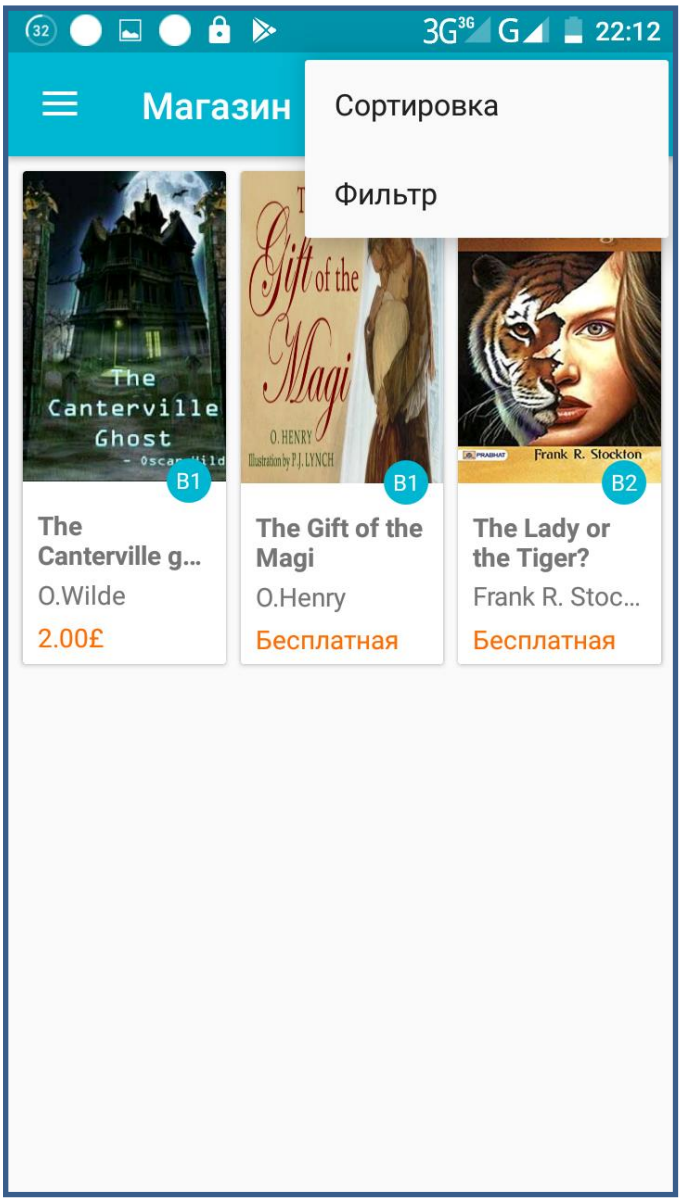


Рис. 2.7. Меню каталогу

Натиснувши на опис книги, користувач може перейти до екрану її детального опису (рис. 2.8), на якому розміщена наступна інформація: зображення обкладинки, назва, автора, списла анотація, рівень складності, жанр, розмір файлу, тривалість аудіозапису, ціну та кнопку покупки\завантаження.

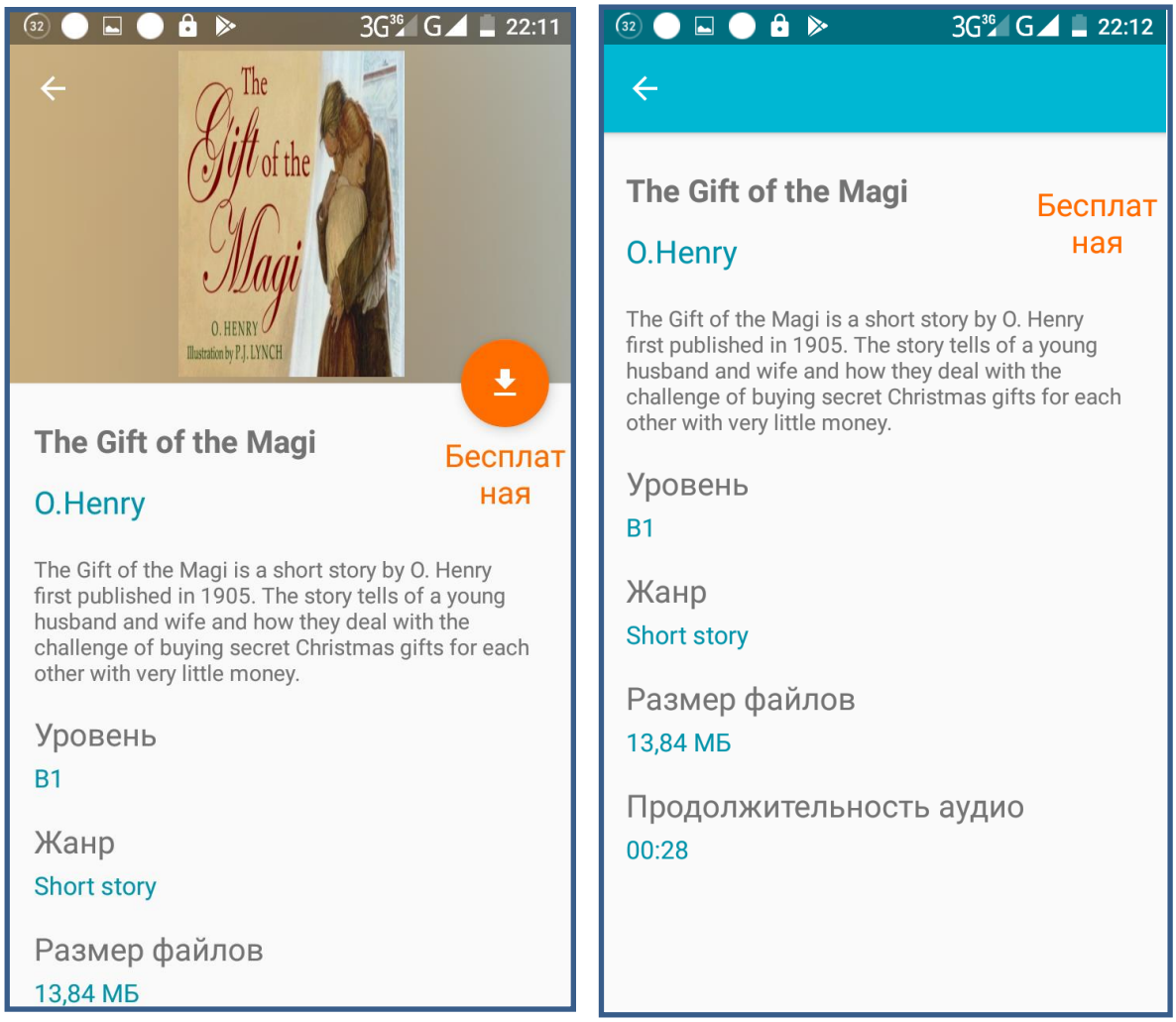


Рис. 2.8. Екран опису книги

Завантажити книгу на мобільний пристрій користувача можно шляхом натискання на кнопку «Завантажити» на екрані опису книги. Процес завантаження, що розпочався після натискання на кнопку, відображається за допомогою прогрес-бару у верхній частині екрану (рис. 2.9).

Рис. 2.9. Завантаження книги

Після того, як завантаження завершене, книгу можна відкрити для читання. На початку файлу знаходится перелік ключових слів (рис. 2.10), далі йде текст самою книги (рис. 2.11). У нижній частині екрану переглядача знаходиться елементи управління аудіоплеєром, що відтворює аудиофайл з відповідними звуковии доріжками.

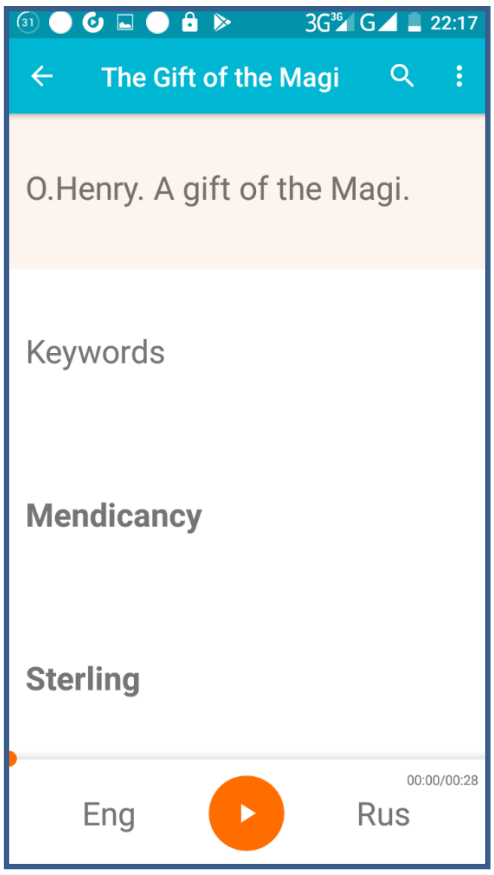


Рис. 2.10. Ключові слова

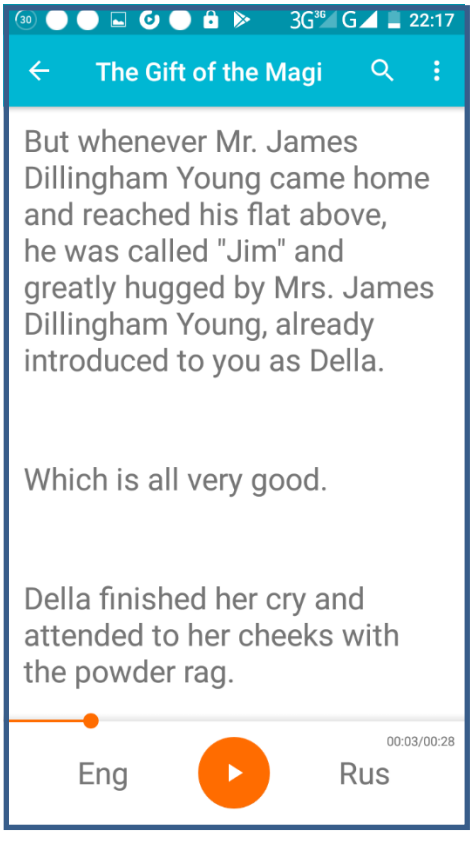


Рис. 2.11. Відтворення аудіо

Екран переглядача також має меню, доступ до якого надається через піктограму у верхньому правому кутку екрану (рис. 2.12). Меню містить такі пункти: «Вибрати мову аудиодоріжки», «Змінити параметри відображення», «Ключові слова» і «Про книгу».

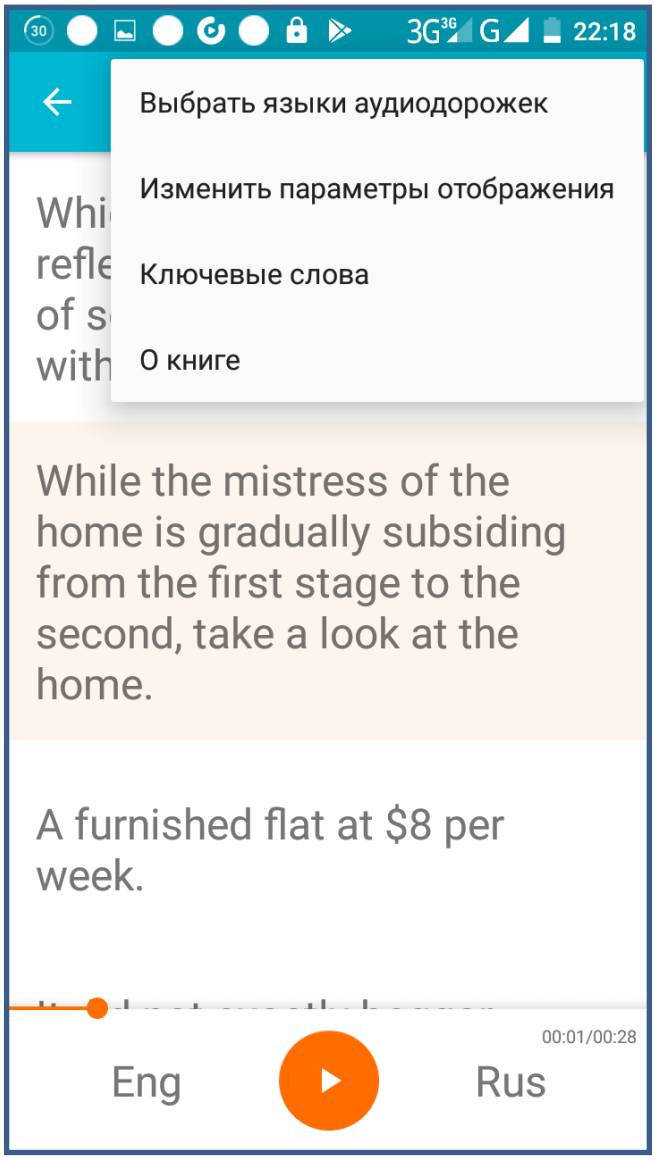


Рис. 2.12. Меню переглядача книг

«Ключові слова» дозволє увімкнути або вимкнути показ ключових слів книги на екрані (рис. 2.13).

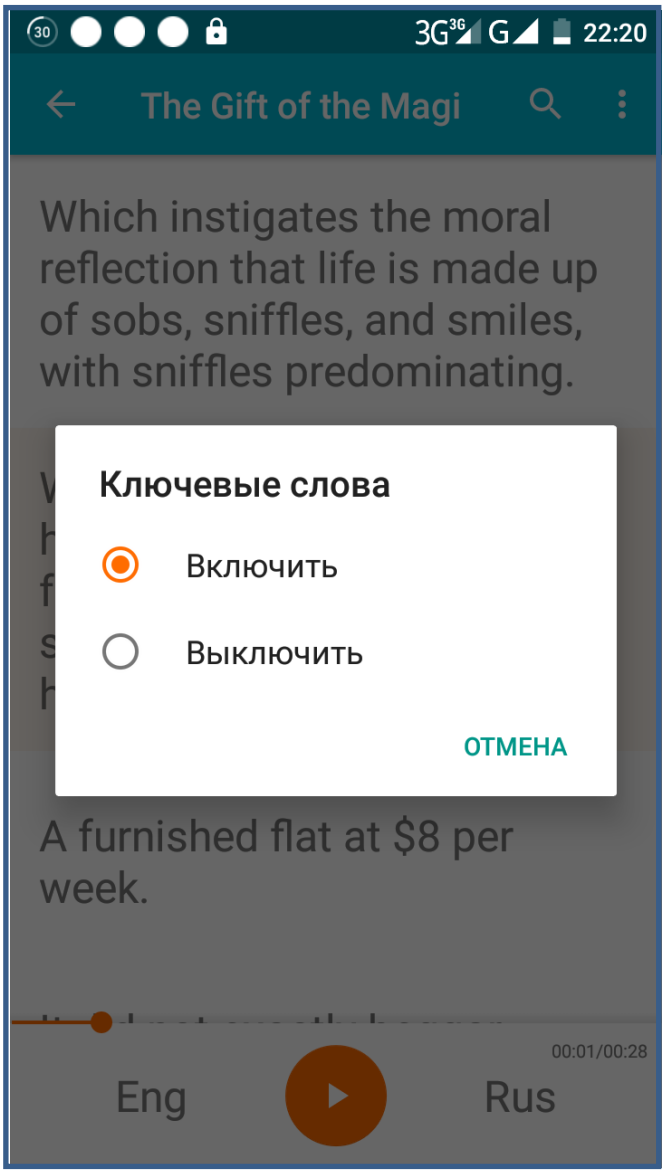


Рис. 2.13. Управління ключовими словами

Пункт меню «Вибрати мову аудиодоріжки» дозволяє читачеві перемекати наявні аудиодоріжки файлу книги, встановлюючи їх на першу чи другу позицію (рис. 2.14).

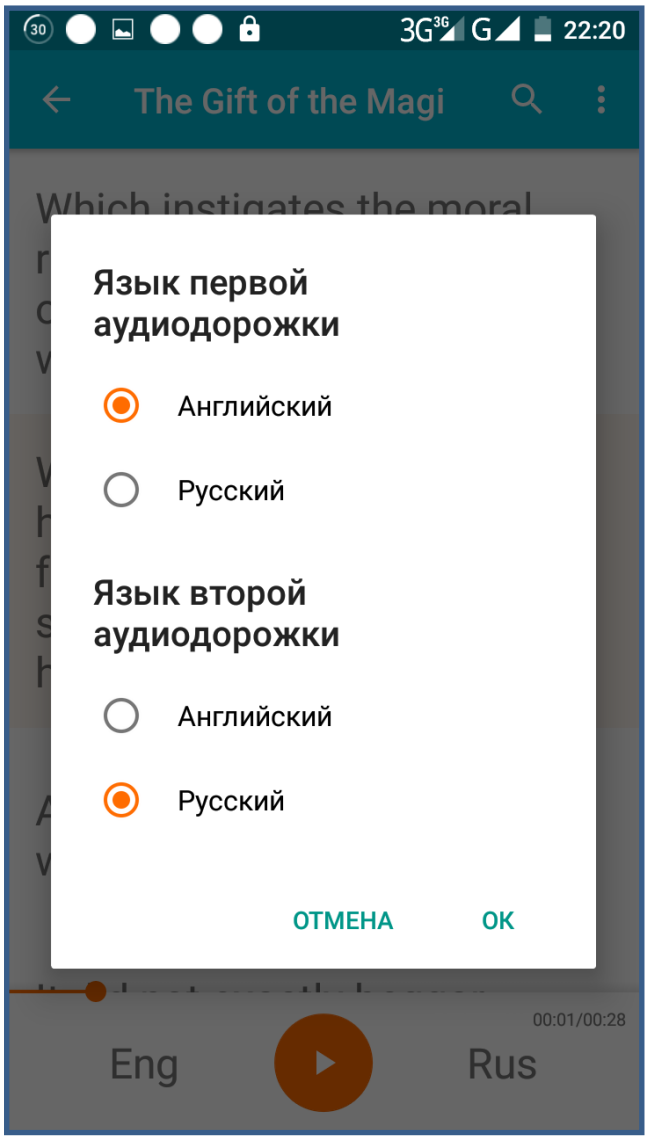


Рис. 2.14. Перемикання аудиодоріжок

Пункт «Змінити параметри відображення» містить елементи керування кольором фону, розміром шрифта та яскравістю екрану (рис. 2.15). На екрані «Про книгу» розміщена анотація видання.

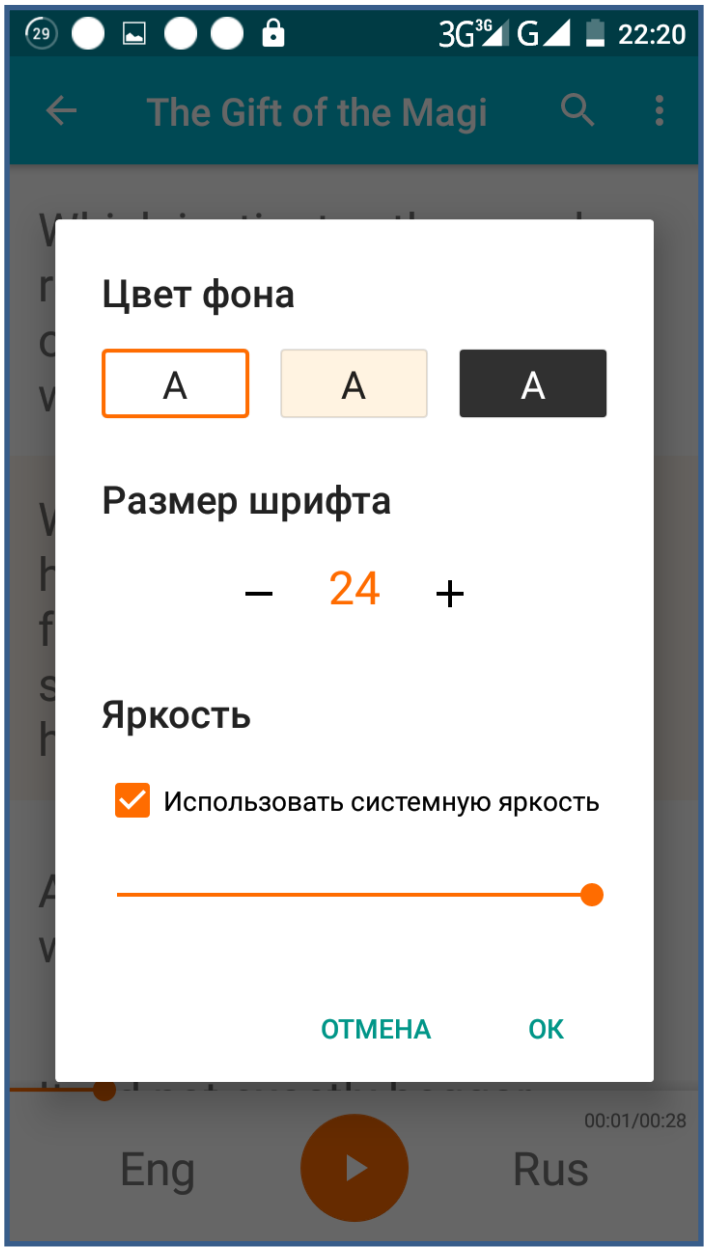


Рис. 2.15. Налаштування відображення книги

Книги, які користувач завантажив із каталогу бібліотеки, відображуються на екрані «Мої книги» (рис. 2.16). У нижньому правому куті цього екрану знаходиться кнопка переходу до режиму читання обраної книги.

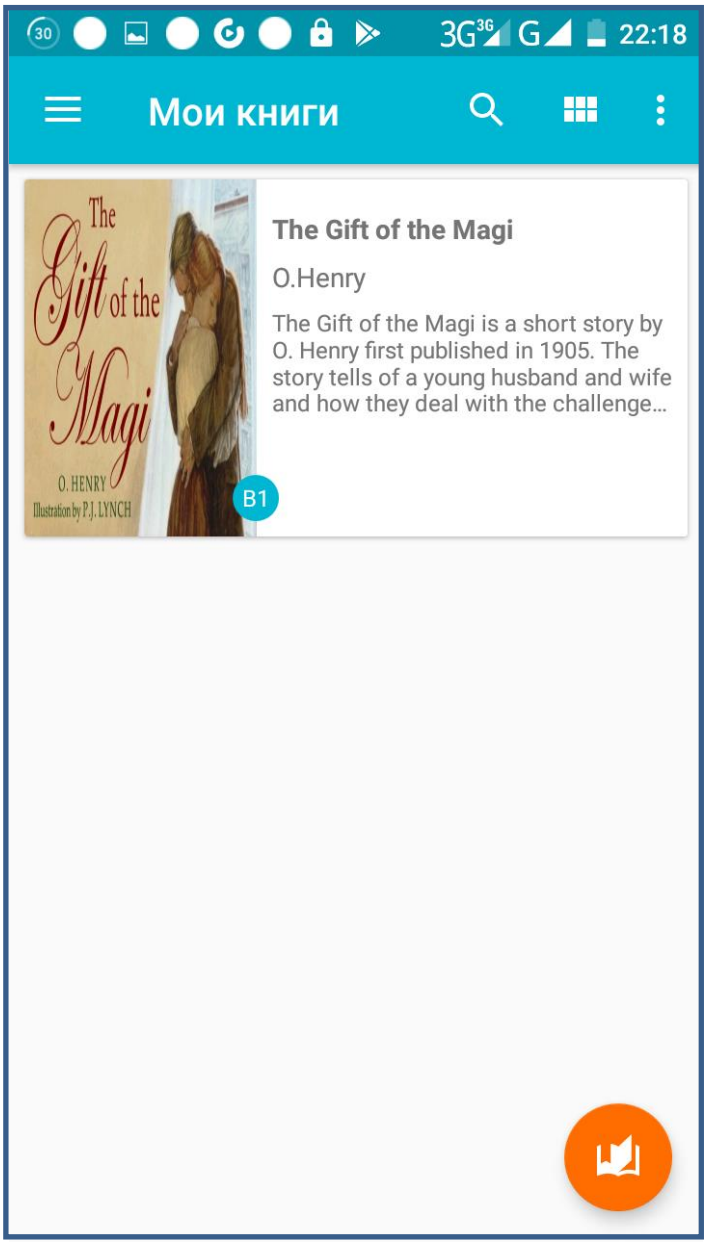


Рис. 2.16. Список завантажених книг

Також завантажені книги відповідним чином помічаються у загальному каталозі на екрані «Магазин» - на це вказує зміна фону книги, та заміни ціни на символ «√» (рис. 2.17).

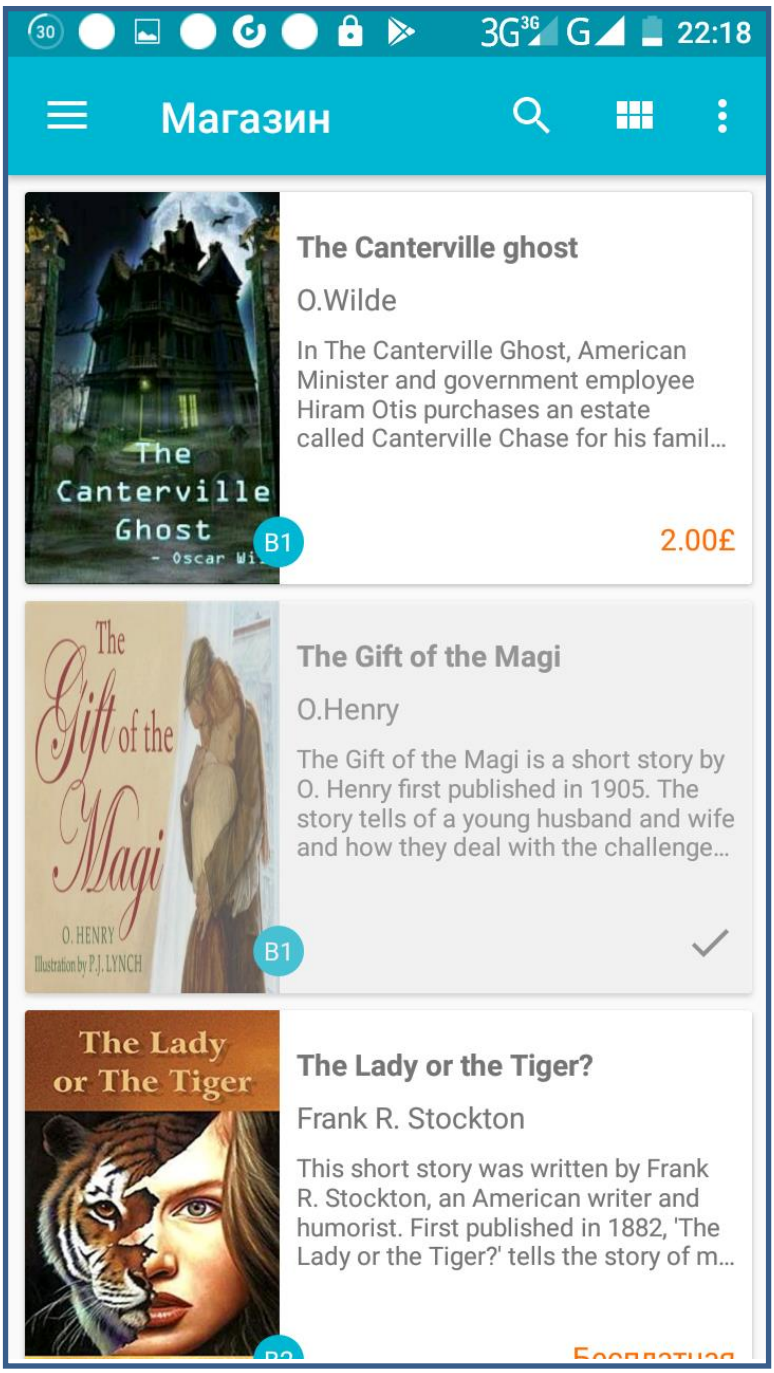


Рис. 2.17. Відображення завантажених книг у загальному каталозі

На екрані «Підписка» розміщена інформація про доступні у бібліотеці платні абоніменти, та кнопки для їх замовлення, помічені значенням відповідної ціни (рис. 2.18).

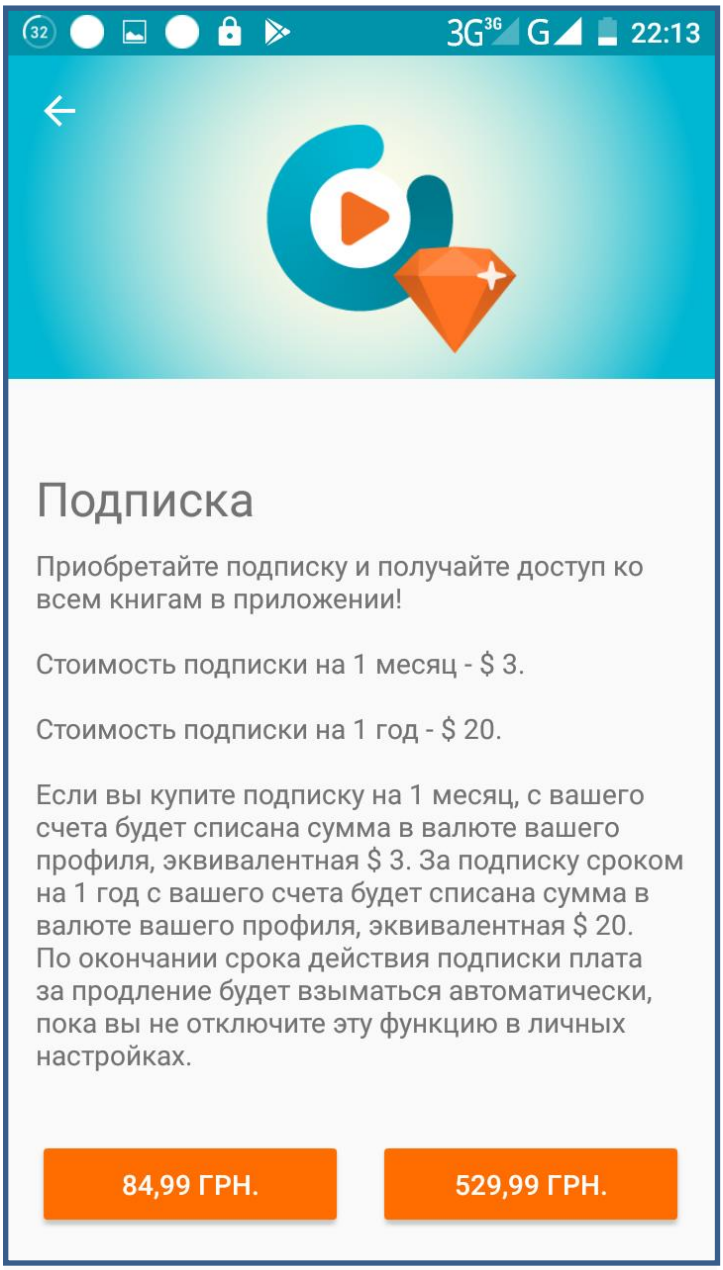


Рис. 2.18. Підписка

Екран «Налаштування» (рис.2.19) надає інструменти для увімкнення/вимкнення Push-повідомлень, вибору мови інтерфейсу та теми оформлення екранів застосунку, отримання довідки про програму.

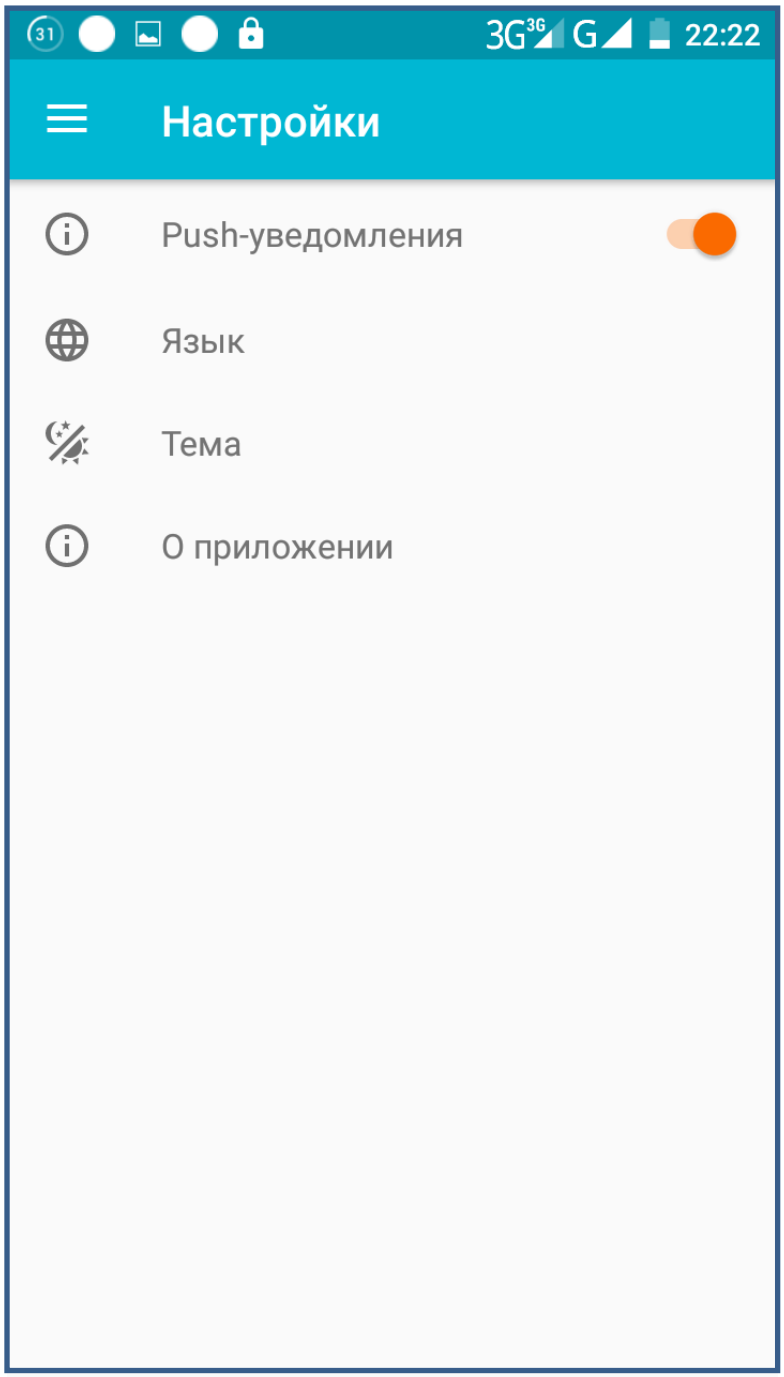


Рис. 2.19. Налаштування

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 1000;
2. коефіцієнт складності програми – 1,6;
3. коефіцієнт корекції програми в ході її розробки – 0,05;
4. годинна заробітна плата програміста – 65 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,2;
7. вартість машино-години ЕОМ – 14 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмногму забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p),$$

де q - передбачуване число операторів (1200);

C - коефіцієнт складності програми (1,6);

p - коефіцієнт корекції програми в ході її розробки (0,05).

Звідси умовне число операторів в програмі:

$$Q = 1{,}6 \cdot 1000 \cdot (1 + 0{,}05) = 1680$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин,}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3 до 5 років він складає 1,2.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1.2). З урахуванням коефіцієнта кваліфікації k = 1,2, отримуємо витрати праці на вивчення опису завдання:

$$t_u = (1680 \cdot 1{,}2) / (75 \cdot 1{,}2) = 22{,}4 \text{ людино-година}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин,}\quad (3.2)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.2), отримаємо:

$$t_a = 1680 / (20 \cdot 1{,}2) = 70 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин.}$$

$$t_n = 1680 / (25 \cdot 1{,}2) = 56 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}\ \text{, людино-годин.}$$

$$t_{отл} = 1680 / (5 \cdot 1{,}2) = 280 \text{ чел.-ч.}$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{k} = 1{,}5 \cdot t_{отл}\ \text{, людино-годин.}$$

$$t^{k}_{отл} = 1{,}5 \cdot 280 = 420 \text{ людино-годин}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_{д} = t_{др} + t_{до} \text{ , людино-годин,}$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20) \cdot k} \text{ , людино-годин,}$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 \cdot t_{др} \text{ , людино-годин.}$$

Підставляючи відповідні значення, отримаємо:

$$t_{др} = 1680 / (18 \cdot 1{,}2) = 77{,}78 \text{ людино-годин.}$$

$$t_{до} = 0{,}75 \cdot 77{,}78 = 58{,}33 \text{ людино-годин.}$$

$$t_{д} = 77{,}78 + 58{,}33 = 136{,}11 \text{ людино-годин.}$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 50 + 22{,}4 + 70 + 56 + 280 + 136{,}11 = 614{,}51 \text{ людино-годин.}$$

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.}$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,}$$

де: t - загальна трудомісткість, людино-годин;

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 65 грн / год, отримуємо:

$$З_{ЗП} = 614{,}51 \cdot 65 = 36\,943{,}15 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot С_{мч}, \text{ грн,} \quad (3.3)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$С_{мч}$ - вартість машино-години ЕОМ, грн/год (14 грн/год).

Підставивши в формулу (3.3) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 280 \cdot 14 = 3920 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 36943{,}15 + 3920 = 40\,863{,}15 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.}$$

де B_k - число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Звідси витрати на створення програмного продукту:

$$T = 614{,}51 / 1 \cdot 176 \approx 3{,}5 \text{ міс.}$$

Висновок: програмне забезпечення призначене для роботи з онлайн бібліотекою адаптованої англомовної літератури під ОС Android. Вартість даного програмного забезпечення становить майже 40,8 тис. грн. і не вимагає додаткових витрат як при розробці, так і при впровадженні програми. Очікуваний час розробки становить 3,5 місяців. Цей термін пов'язаний зі значним числом операторів, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми і підготовку документації.

ВИСНОВКИ

В данній кваліфікаційній роботі був розроблений застосунок для мобільних пристроїв для роботи з онлайн бібліотекою.

Це програмне забезпечення призначене для підключення до серверу онлайн бібліотеки за допомогою Android-пристороюю через мережі Wi-Fi та Інтернет та завантаження і читання електроних книжок.

Практичне призначення даного застосунку полягає у наданні користувачу простого у використанні і зручного інструменту для отримання можливості читання електроних книг та роботи з віддаленою онлайн бібліотекою на мобільному пристрої.

Під час виконання даної кваліфікаційної роботи були виконані наступні задачі:

- вивчино предметну галузь розв'язуваної задачі;
- проведено порівняльний аналіз можливостей аналогічних застосунків;
- обрано раціональну структуру і параметри програми;
- розроблено структуру вхідних і вихідних даних для проектованого програмного забезпечення;
- написано програмний код застосунку;
- розроблено рекомендації щодо застосування програми.

Розроблене програмне забезпечення дозволяє:

- відображення змісту каталогу онлайн бібліотеки;
- завантаження файлів електронних книг на мобільний пристрій;
- відображення змісту файлів текстових форматів на екран пристрою;
- відтворення файлів аудіо книг;
- налаштування візуальних параметрів відображення тексту у режимі читання.

Програма реалізована на мові програмування Python з використанням фреймворку Django та СУБД MySQL.

Впровадження даного програмного продукту є економічно вигідним, оскільки його повна вартість є помірною за рахунок використання некомерційних інструментів для розробки, відсутності необхідності конфігурації і обслуговування робочих місць користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К. : Держспоживстандарт України, 2007. – 47 с. – (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).
2. Васильев А. Н. Python на примерах. Практический курс по программированию. 2-е изд. /А.Н. Васильев. — Наука и Техника, 2017. — 432 с.
3. Гарсиа-Молина Г. Системы баз данных. Полный курс / Г. Гарсиа-Молина, Дж. Ульман, Дж. Уидом. — М.: Вильямс, 2003. – 1088 с.
4. Грабер М. Введение в SQL. — М.: Лори, 1996. — 479 с.
5. Гриффитс Д. Head First. Программирование для Android / Д. Гриффитс, Д. Гриффитс. – СПб.: Питер, 2016. – 704 с.
6. Дейт К. Дж. Введение в системы баз данных. 6-е изд. – СПб.: Вильямс, 2000. – 848с.
7. Дейтел П. Android для разработчиков. / П. Дейтел, Х. Дейтел, А. Уолд. – СПб.: Питер, 2016. – 512 с.
8. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.
9. Дронов В. Django. Практика создания Web-сайтов на Python. / В. Дронов. — СПб.: БХВ-Петербург, 2016. — 528 с.
10. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. — Київ : Держстандарт України, 1994. – 88 с.
11. Камер Д.Э. Сети TCP/IP. Том 1. Принципы, протоколы и структура – 4-е изд. / Пер. с англ. под ред. С.Г. Тригуб – М.: Издат. дом «Вильямс», 2003

12. Коннолли Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание: Пер. с англ. / Т. Коннолли, К. Бегг - М.: Вильямс, 2003. — 1440 с.

13. Кошик А. Веб-аналитика 2.0 на практике. Тонкости и лучшие методики. / А. Кошик. – М.: Диалектика-Вильямс, 2018. – 528 с.

14. Куроуз Дж., Росс К. Компьютерные сети. Многоуровневая архитектура Интернета – 2-е изд. – СПб.: Питер, 2004

15. Лутц М. Python. Карманный справочник, 5-е издание. / М. Лутц. — М.: Вильямс, 2017. — 320 с.

16. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності «Програмне забезпечення» / О.Г. Вагонова, Н.Н. Романюк; М-во освіти і науки України, ДВНЗ «Нац. гірн. ун- т». – Д.: НГУ, 2016. – 11 с.

17. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 6.050103 «Програмна інженерія» / І.М. Удовик, Л.М. Коротенко, О.С. Шевцова; Нац. гірн. ун-т. – Д: НТУ «Дніпровська політехніка», 2018. – 65 с.

18. Мэтиз Э. Изучаем Python. Программирование игр, визуализация данных, веб-приложения. 2-е изд. / Э. Мэтиз. — СПб.: Питер Пресс, 2018. —496 с.

19. Пьюривал С. Основы разработки веб-приложений / С. Пьюривал. — СПб.: Питер, 2015. — 272 с.

20. Олифер В.Г. Олифер Н.А. Компьютерные сети. Принципы технологии протоколы (4-е изд.) / В.Г. Олифер, Н.А. Олифер - СПб.: - Питер, 2010. - 916 с.

21. Семенов Ю.А. Протоколы Internet – 2-е изд. – М.: Горячая линия-Телеком, 2005.

22. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

23. Скотт Б. Проектирование веб-интерфейсов / Б. Скотт, Т. Нейл. — Спб.: Символ-Плюс, 2010. — 352 с.

24. Столлингс В. Передача данных/ В. Столлингс. - СПб.: Питер, 2004. – 750 с.

25. Филлипс Б. Android. Программирование для профессионалов. / Б. Филлипс, Б. Харди. – СПб.: Питер, 2016. – 640 с.

26. Форсье Д. Django. Разработка веб-приложений на Python. / Д. Форсье, П. Биссекс, У. Чан — СПб.: Символ-Плюс, 2016. — 456 с.

ДОДАТОК А

КОД ПРОГРАМИ

Models.py // моделі застосунку

```
from django.db import models
from jsonfield import JSONField
from model_utils import Choices


class Author(models.Model):
    name = models.CharField(max_length=255)

    def __eq__(self, other):
        values = [(k, v) for k, v in self.__dict__.items() if k != '_state' and k != 'id']
        other_values = [(k, v) for k, v in other.__dict__.items() if k != '_state' and k !=
'id']
        return values == other_values


class Genre(models.Model):
    name = models.CharField(max_length=255)

    def __eq__(self, other):
        values = [(k, v) for k, v in self.__dict__.items() if k != '_state' and k != 'id']
        other_values = [(k, v) for k, v in other.__dict__.items() if k != '_state' and k !=
'id']
        return values == other_values


class Level(models.Model):
    name = models.CharField(max_length=255)

    def __eq__(self, other):
        values = [(k, v) for k, v in self.__dict__.items() if k != '_state' and k != 'id']
        other_values = [(k, v) for k, v in other.__dict__.items() if k != '_state' and k !=
'id']
        return values == other_values


class Book(models.Model):
    PRICING = Choices(
        ("free", "free"),
        ("paid", "paid"),
```

```
        ("subscription", "subscription"),
    )

    name = models.CharField(max_length=255)
    author = models.ForeignKey(Author, related_name='author',
on_delete=models.CASCADE)
    genre = models.ForeignKey(Genre, related_name='genre',
on_delete=models.CASCADE)
    level = models.ForeignKey(Level, related_name='level',
on_delete=models.CASCADE)
    description = models.CharField(max_length=1024)
    coverLink  = models.CharField(max_length=1024)
    languages = JSONField()
    # P° PIPsC, PrP°P»CЊC€Pµ PICЃPµ PSPµ C,PsC‡PSPs
    pricing = models.CharField(choices=PRICING,max_length=12)
    hasDemo = models.BooleanField()
    isVisible = models.BooleanField()
    # P»PsPiPëC‡PSPs PrPsP±P°PIPëC,CЊ price Float, PSP° P°PSPrCЂPsPëPrPµ
PїPs PrPµC„PsP»C,Cѓ CЂP°PIPµPS 0
    price = models.FloatField(default=0)
    filesSize = models.IntegerField(default=0)
    filesDuration = models.IntegerField(default=0)
    # PїPs P»PsPiPëPєPµ PIPµC‰PµP№, product id
PIPsP·PICЂP°C‰P°PµC,CЃCЏ CЃ CЃPµCЂPIPµCЂP° PїPsPєCѓPїPsPє GPlay Pë
Pє PSP°P°C€PµPjCѓ CЃPµCЂPIPµCЂCѓ PsC,PSPsC€PµPëPSCЏ PSPµ
PëPjPµPµC,
    # productId =

    def __eq__(self, other):
        values = [(k, v) for k, v in self.__dict__.items() if k != '_state' and k != 'id']
        other_values = [(k, v) for k, v in other.__dict__.items() if k != '_state' and k !=
'id']
        return values == other_values


class BookAudioLink(models.Model):
    book = models.ForeignKey(Book, related_name='book_audio',
on_delete=models.CASCADE)
    language = models.CharField(max_length=3)
    link = models.CharField(max_length=255)

    def __eq__(self, other):
        values = [(k, v) for k, v in self.__dict__.items() if k != '_state' and k != 'id']
        other_values = [(k, v) for k, v in other.__dict__.items() if k != '_state' and k !=
'id']
```

```
        return values == other_values


class BookTextLink(models.Model):
    book = models.ForeignKey(Book, related_name='book_text',
on_delete=models.CASCADE)
    language = models.CharField(max_length=3)
    link = models.CharField(max_length=255)

    def __eq__(self, other):
        values = [(k, v) for k, v in self.__dict__.items() if k != '_state' and k != 'id']
        other_values = [(k, v) for k, v in other.__dict__.items() if k != '_state' and k !=
'id']
        return values == other_values


class BookAudioSync(models.Model):
    book = models.ForeignKey(Book, related_name='book_audiosyncs',
on_delete=models.CASCADE)
    language = models.CharField(max_length=3)
    timings = JSONField()

    def __eq__(self, other):
        values = [(k, v) for k, v in self.__dict__.items() if k != '_state' and k != 'id']
        other_values = [(k, v) for k, v in other.__dict__.items() if k != '_state' and k !=
'id']
        return values == other_values


class BookAudioLinkDemo(models.Model):
    book = models.ForeignKey(Book, related_name='book_audio_demo',
on_delete=models.CASCADE)
    language = models.CharField(max_length=3)
    link = models.CharField(max_length=255)

    def __eq__(self, other):
        values = [(k, v) for k, v in self.__dict__.items() if k != '_state' and k != 'id']
        other_values = [(k, v) for k, v in other.__dict__.items() if k != '_state' and k !=
'id']
        return values == other_values


class BookTextLinkDemo(models.Model):
    book = models.ForeignKey(Book, related_name='book_text_demo',
on_delete=models.CASCADE)
```

```
    language = models.CharField(max_length=3)
    link = models.CharField(max_length=255)

    def __eq__(self, other):
        values = [(k, v) for k, v in self.__dict__.items() if k != '_state' and k != 'id']
        other_values = [(k, v) for k, v in other.__dict__.items() if k != '_state' and k !=
'id']
        return values == other_values


class BookAudioSyncDemo(models.Model):
    book = models.ForeignKey(Book, related_name='book_audiosyncs_demo',
on_delete=models.CASCADE)
    language = models.CharField(max_length=3)
    timings = JSONField()

    def __eq__(self, other):
        values = [(k, v) for k, v in self.__dict__.items() if k != '_state' and k != 'id']
        other_values = [(k, v) for k, v in other.__dict__.items() if k != '_state' and k !=
'id']
        return values == other_values
```

Views.py //представлення застосунку

```
from django.http.response import HttpResponse, JsonResponse

from rest_framework.parsers import FileUploadParser, MultiPartParser
from rest_framework.views import APIView
import json
import base64

from api import csvparser
#from api.importData import DataImport
from api.models import Book as BookModel

from api.models import BookAudioLink as BookAudioLinkModel
from api.models import BookTextLink as BookTextLinkModel
from api.models import BookAudioSync as BookAudioSyncModel

from api.models import BookAudioLinkDemo as BookAudioLinkDemoModel
from api.models import BookTextLinkDemo as BookTextLinkDemoModel
from api.models import BookAudioSyncDemo as BookAudioSyncDemoModel

from api.serializers import BookSerializer, BookAudioLinkSerializer,
BookTextLinkSerializer, BookAudioSyncSerializer, \
```

```
    BookTextLinkDemoSerializer, BookAudioLinkDemoSerializer,
BookAudioSyncDemoSerializer

from api.importData import DataImport
from Readdo.settings import DATAPATH

class Book(APIView):

    def get(self, request):
        books = BookModel.objects.filter(isVisible=True).all()
        serializer = BookSerializer(books, many=True)
        return JsonResponse(serializer.data, safe=False, status=200)


class BookTest(APIView):

    def get(self, request):
        books = BookModel.objects.filter(isVisible=True).filter(hasDemo=True).all()
        serializer = BookSerializer(books, many=True)
        return JsonResponse(serializer.data, safe=False, status=200)

class BookAudioLink(APIView):
    def post(self, request):
        # get audio book link
        #id = request.GET.get('id')
        #language = request.GET.get('language')

        id = request.data.get('id')
        language = request.data.get('language')


        audiolink = BookAudioLinkModel.objects.filter(book_id=id,
language=language).first()
        if audiolink is None:
            return JsonResponse({'error': 'Book audio link not found'}, safe=False,
status=403)
        else:
            request_data = audiolink.__dict__
            request_data.pop('_state')
            book_id = request_data.pop('book_id')
            request_data['book'] = {'id': book_id}
            #print(request_data)
            serializer = BookAudioLinkSerializer(data=request_data)
            if serializer.is_valid():
                final_response = serializer.data
```

```
            final_response.pop('book')
            final_response.pop('language')
            return JsonResponse(final_response, safe=False, status=200)
        else:
            print(serializer.errors)
            return JsonResponse(serializer.errors, safe=False, status=403)


class BookTextLink(APIView):
    def post(self, request):
        # get audio book link
        # id = request.GET.get('id')
        # language = request.GET.get('language')

        id = request.data.get('id')
        language = request.data.get('language')

        textlink = BookTextLinkModel.objects.filter(book_id=id,
language=language).first()
        if textlink is None:
            return JsonResponse({'error': 'Book text link not found'}, safe=False,
status=403)
        else:
            request_data = textlink.__dict__
            request_data.pop('_state')
            book_id = request_data.pop('book_id')
            request_data['book'] = {'id': book_id}
            #print(request_data)
            serializer = BookTextLinkSerializer(data=request_data)
            if serializer.is_valid():
                final_response = serializer.data
                final_response.pop('book')
                final_response.pop('language')
                return JsonResponse(final_response, safe=False, status=200)
            else:
                print(serializer.errors)
                return JsonResponse(serializer.errors, safe=False, status=403)


class BookAudioSync(APIView):
    def post(self, request):
        # get audio book link
        print("Book audio sync called")
        #id = request.GET.get('id')
        #language = request.GET.get('language')
```

```
        id = request.data.get('id')
        language = request.data.get('language')

        audiosync = BookAudioSyncModel.objects.filter(book_id=id,
language=language).first()
        if audiosync is None:
            return JsonResponse({'error': 'Book audio sync not found'}, safe=False,
status=403)
        else:
            request_data = audiosync.__dict__
            request_data.pop('_state')
            book_id = request_data.pop('book_id')
            request_data['book'] = {'id': book_id}
            print(request_data)
            serializer = BookAudioSyncSerializer(data=request_data)
            if serializer.is_valid():
                final_response = serializer.data
                final_response.pop('book')
                # final_response.pop('language')
                return JsonResponse(final_response, safe=False, status=200)
            else:
                print(serializer.errors)
                return JsonResponse(serializer.errors, safe=False, status=403)


class BookLinks(APIView):
    def post(self, request):
        id = request.data.get('id')
        languages = request.data.get('languages')

        languages = languages.replace("[","")
        languages = languages.replace("]","")
        languages = languages.replace("\"", "")

        lang = []

        if "," in languages:
            splitted = languages.split(",")
            for item in splitted:
                lang.append(item)
        else:
            lang.append(languages)

        book = BookModel.objects.filter(id=id).first()
```

```
        if book is None:
            return JsonResponse({'error': 'Book not found'}, safe=False, status=403)
        else:
            final_response = {}
            final_response["audioLinks"] = {}

            audioLinks = BookAudioLinkModel.objects.filter(book=book)
            if audioLinks is not None:
                for link in audioLinks:
                    if link.language in lang:
                        final_response["audioLinks"][link.language] = link.link
            else:
                return JsonResponse({'error': 'Book audio links not found'}, safe=False,
status=403)

            final_response["audioSyncs"] = {}
            audioSyncs = BookAudioSyncModel.objects.filter(book=book)
            if audioSyncs is not None:
                for sync in audioSyncs:
                    if sync.language in lang:
                        final_response["audioSyncs"][sync.language] = sync.timings
            else:
                return JsonResponse({'error': 'Book audiosync not found'}, safe=False,
status=403)

            textLink = BookTextLinkModel.objects.filter(book=book).first()
            if (textLink is not None):
                final_response["textLink"] = textLink.link
                print(final_response)
                return JsonResponse(data=final_response)

            else:
                return JsonResponse({'error': 'Book text not found'}, safe=False,
status=403)


class BookLinksPreview(APIView):
    def post(self, request):
        id = request.data.get('id')
        languages = request.data.get('languages')

        languages = languages.replace("[","")
        languages = languages.replace("]","")
        languages = languages.replace("\"", "")
```

```
        lang = []

        if "," in languages:
            splitted = languages.split(",")
            for item in splitted:
                lang.append(item)
        else:
            lang.append(languages)

        book = BookModel.objects.filter(id=id).filter(hasDemo=True).first()
        if book is None:
            return JsonResponse({'error': 'Book not found'}, safe=False, status=403)
        else:
            final_response = {}
            final_response["audioLinks"] = {}

            audioLinks = BookAudioLinkDemoModel.objects.filter(book=book)
            if audioLinks is not None:
                for link in audioLinks:
                    if link.language in lang:
                        final_response["audioLinks"][link.language] = link.link
            else:
                return JsonResponse({'error': 'Book audio links not found'}, safe=False,
status=403)

            final_response["audioSyncs"] = {}
            audioSyncs = BookAudioSyncDemoModel.objects.filter(book=book)
            if audioSyncs is not None:
                for sync in audioSyncs:
                    if sync.language in lang:
                        final_response["audioSyncs"][sync.language] = sync.timings
            else:
                return JsonResponse({'error': 'Book audiosync not found'}, safe=False,
status=403)

            textLink = BookTextLinkDemoModel.objects.filter(book=book).first()
            if (textLink is not None):
                final_response["textLink"] = textLink.link
                #print(final_response)
                return JsonResponse(data=final_response)

            else:
                return JsonResponse({'error': 'Book text not found'}, safe=False,
status=403)
```

```
class ImportData(APIView):
    def get(self,request):
        data_importer = DataImport()
        result = data_importer.parseFinalData()
        return JsonResponse(result,status=200,safe=False)
```

Serializers.py // функції серіалізації бази даних

```
from drf_writable_nested import WritableNestedModelSerializer,
UniqueFieldsMixin, NestedUpdateMixin
from rest_framework import serializers

from api.models import Book, Author, Genre, Level, BookAudioLink,
BookTextLink, BookAudioSync, BookAudioLinkDemo, \
    BookTextLinkDemo, BookAudioSyncDemo


class ChoicesField(serializers.Field):
    def __init__(self, choices, **kwargs):
        self._choices = choices
        super(ChoicesField, self).__init__(**kwargs)

    def to_representation(self, obj):
        return self._choices[obj]

    def to_internal_value(self, data):
        return getattr(self._choices, data)


class AuthorSerializer(serializers.ModelSerializer):

    class Meta:
        model = Author
        fields = ('name',)

    def to_representation(self, instance):
        return instance.name



class GenreSerializer(serializers.ModelSerializer):

    class Meta:
        model = Genre
        fields = ('name',)
```

```
    def to_representation(self, instance):
        return instance.name

class LevelSerializer(serializers.ModelSerializer):

    class Meta:
        model = Level
        fields = ('name',)

    def to_representation(self, instance):
        return instance.name


class BookSerializer(serializers.ModelSerializer):
#class BookSerializer(WritableNestedModelSerializer):
    id = serializers.IntegerField(required=False)
    name = serializers.CharField(max_length=255,required = True)
    author = AuthorSerializer(many=False, required = True)
    genre = GenreSerializer(many=False, read_only=False,required = True)
    level = LevelSerializer(many=False, read_only=False,required = True)
    description = serializers.CharField(max_length=255,required = True)
    coverLink = serializers.CharField(max_length=255,required = False)
    languages = serializers.JSONField(required = True)
    hasDemo = serializers.BooleanField(required = True)
    pricing = ChoicesField(choices=Book.PRICING,required = True)
    price = serializers.FloatField(required=True)
    isVisible = serializers.BooleanField(required = True)
    filesSize = serializers.IntegerField(required=False)
    filesDuration = serializers.IntegerField(required=False)

    class Meta:
        model = Book
        fields =
('id','name','author','genre','level','description','coverLink','languages','hasDemo','pricin
g','price','isVisible','filesSize','filesDuration')

    def create(self, validated_data):
        author = validated_data.pop('author')
        level = validated_data.pop('level')
        genre = validated_data.pop('genre')

        authorObj = Author.objects.get_or_create(name=author['name'])[0]
        levelObj = Level.objects.get_or_create(name=level['name'])[0]
        genreObj = Genre.objects.get_or_create(name=genre['name'])[0]
```

```
        book = Book.objects.create(**validated_data, author=authorObj,
level=levelObj, genre=genreObj)
        return book

    def update(self, instance, validated_data):
        author = validated_data.pop('author')
        level = validated_data.pop('level')
        genre = validated_data.pop('genre')

        authorObj = Author.objects.get_or_create(name=author['name'])[0]
        levelObj = Level.objects.get_or_create(name=level['name'])[0]
        genreObj = Genre.objects.get_or_create(name=genre['name'])[0]

        instance.author = authorObj
        instance.level = levelObj
        instance.genre = genreObj

        super(BookSerializer, self).update(instance,validated_data)
        return instance

class BookSerializerWithID(serializers.ModelSerializer):
    class Meta:
        model= Book
        fields = ('id',)

class BookAudioLinkSerializer(WritableNestedModelSerializer):
    book = BookSerializerWithID(many=False,required=True)
    language = serializers.CharField(max_length=3,required = True)
    link = serializers.CharField(max_length=255,required = True)

    class Meta:
        model = BookAudioLink
        fields = ('id','language','link','book',)

class BookTextLinkSerializer(WritableNestedModelSerializer):
    book = BookSerializerWithID(many=False, required=True)
    language = serializers.CharField(max_length=3, required=True)
    link = serializers.CharField(max_length=255, required=True)

    class Meta:
        model = BookTextLink
        fields = ('id', 'language', 'link', 'book',)
```

```
class BookAudioSyncSerializer(WritableNestedModelSerializer):
    book = BookSerializerWithID(many=False, required=True)
    language = serializers.CharField(max_length=3, required=True)
    timings = serializers.JSONField(required=True)

    class Meta:
        model = BookAudioSync
        fields = ('id', 'language', 'timings', 'book',)

#-------------
class BookAudioLinkDemoSerializer(WritableNestedModelSerializer):
    book = BookSerializerWithID(many=False,required=True)
    language = serializers.CharField(max_length=3,required = True)
    link = serializers.CharField(max_length=255,required = True)

    class Meta:
        model = BookAudioLinkDemo
        fields = ('id','language','link','book',)

class BookTextLinkDemoSerializer(WritableNestedModelSerializer):
    book = BookSerializerWithID(many=False, required=True)
    language = serializers.CharField(max_length=3, required=True)
    link = serializers.CharField(max_length=255, required=True)

    class Meta:
        model = BookTextLinkDemo
        fields = ('id', 'language', 'link', 'book',)


class BookAudioSyncDemoSerializer(WritableNestedModelSerializer):
    book = BookSerializerWithID(many=False, required=True)
    language = serializers.CharField(max_length=3, required=True)
    timings = serializers.JSONField(required=True)

    class Meta:
        model = BookAudioSyncDemo
        fields = ('id', 'language', 'timings', 'book',)
```

Settings.py //управління налаштуваннями застосунку

```
"""
Django settings for Readdo project.

Generated by 'django-admin startproject' using Django 2.1.3.
```

```
For more information on this file, see
https://docs.djangoproject.com/en/2.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/2.1/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'tg@zt)k8r&-wy7rh3%xqt*#3sdehsz)3a$jhr!xnss_nhh_c0q'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True



# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'api.apps.ApiConfig',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
```

```
]

ROOT_URLCONF = 'Readdo.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')]
        ,
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'Readdo.wsgi.application'


# Database
# https://docs.djangoproject.com/en/2.1/ref/settings/#databases

# Password validation
# https://docs.djangoproject.com/en/2.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
```

```
# Internationalization
# https://docs.djangoproject.com/en/2.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

APPEND_SLASH = False
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.1/howto/static-files/

STATIC_URL = '/Files/readers/'

#   Dev test
# DATAPATH = '/home/alex/Readdo/test_env/'
# HOST = "http://192.168.100.102:8000"
# IMPORT_CSV_PATH = "/home/alex/Readdo/test_env/books_final.csv"
# ALLOWED_HOSTS = ['192.168.100.102', 'localhost']
# DATABASES = {
#     'default': {
#         'ENGINE': 'django.db.backends.sqlite3',
#         'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
#     }
# }

#   PRODUCTION !!
DATAPATH = '/var/www/html/on-the-go.eu/public/Files/readers/'
HOST = "https://on-the-go.eu"
IMPORT_CSV_PATH = "/var/www/html/on-the-go.eu/books_final.csv"
ALLOWED_HOSTS = ['on-the-go.eu', 'localhost', '94.23.204.156']

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'readdo',
        'USER': 'readdouser',
        'PASSWORD': 'vH7jnBUg49UtBvpu',
```

```
        'HOST': 'localhost',
    }
}

STATIC_ROOT = DATAPATH
#STATICFILES_DIRS = [
#    os.path.join(BASE_DIR, ""),
#    DATAPATH
#]
```

Urls.py //посилання на контент бібліотеки

```
# #from .views import BookViewSet
from api.views import BookAudioLink, BookTextLink, BookAudioSync,
BookLinks,  BookLinksPreview, \
    ImportData, BookTest
from .views import Book
from django.urls import path
# from rest_framework.routers import DefaultRouter
#
# router = DefaultRouter(trailing_slash=False)
# router.register('book', Book.as_view(),base_name='book')
urlpatterns = [
    path('booksnew/',Book.as_view()),
    path('testBooks/', BookTest.as_view()),
    path('bookaudio/',BookAudioLink.as_view()),
    path('booktext/', BookTextLink.as_view()),
    path('bookaudiosync/', BookAudioSync.as_view()),
    path('booklinks/',BookLinks.as_view()),
    path('bookPreviewLinks/', BookLinksPreview.as_view()),

    path('importData/', ImportData.as_view()),

 ]
```

ДОДАТОК Б

Відгук керівника економічного розділу
на кваліфікаційну роботу
на тему:
«Розробка мобільного додатку «Бібліотека» для ОС Андроїд на мові програмування Python з використанням фреймворку Django»
студента групи 122-18ск-1 Катасонова Кирила Сергійовича

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файла	Опис
Пояснювальні документи	
Диплом Катасонов.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом Катасонов.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
Diplom Катасонов.zip	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація Катасонов.ppt	Презентація кваліфікаційної роботи.