

**СОДЕРЖАНИЕ**

[1 ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ 9](#_Toc200057069)

[1.1 Описание предметной области 9](#_Toc200057070)

[1.2 Анализ существующих аналогов 10](#_Toc200057071)

[1.2.1 Libib 10](#_Toc200057072)

[1.2.2 CLZ Books 13](#_Toc200057073)

[1.2.3 ReadEra 15](#_Toc200057074)

[1.2.4 Сравнение аналогов 17](#_Toc200057075)

[1.3 Техническое задание 17](#_Toc200057076)

[1.3.1 Определение пользовательских требований 17](#_Toc200057077)

[1.3.2 Определение функциональных требований 18](#_Toc200057078)

[2 ПРОЕКТНАЯ ЧАСТЬ 21](#_Toc200057079)

[2.1 Описание процесса в нотации IDEF0 21](#_Toc200057080)

[2.2 Описание процесса в нотации DFD 28](#_Toc200057081)

[2.3 Описание и обоснование выбора программного обеспечения 30](#_Toc200057082)

[2.4 Архитектура программной среды 31](#_Toc200057083)

[3 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 34](#_Toc200057084)

[3.1 Описание моделей и структур данных 34](#_Toc200057085)

[3.2 Тестирование программного продукта 40](#_Toc200057086)

# ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящем отчете о НИР применяют следующие термины с соответствующими определениями:

Авторизация — процесс предоставления пользователю или группе пользователей определенных разрешений, прав доступа и привилегий в компьютерной системе.

База данных (БД) — совокупность хранимых в памяти компьютера данных, относящихся к определенному объему или кругу деятельности, специально организованных, обновляемых и логически связанных между собой.

Приложение — прикладная система или программа, предназначенная для решения задач в конкретной области техники.

Android — операционная система, разработанная компанией Google, предназначенная для мобильных устройств.

Android Studio — интегрированная среда разработки (IDE) от Google, предназначенная для создания Android-приложений.

Java — объектно-ориентированный язык программирования, используемый для разработки мобильных приложений на платформе Android.

Firebase Realtime Database — облачная база данных от Google, позволяющая хранить и синхронизировать данные в режиме реального времени.

SharedPreferences — механизм Android для хранения простых пар «ключ-значение» в локальной памяти устройства, применяемый для сохранения настроек пользователя и состояния приложения.

Single Activity – Multiple Fragments — архитектурный подход в Android-разработке, при котором всё приложение реализовано в рамках одной активности, содержащей несколько фрагментов для отображения различных интерфейсов.

Glide — библиотека для загрузки и отображения изображений в Android-приложениях с кэшированием и обработкой ошибок.

XML — расширяемый язык разметки, используемый для описания структуры пользовательского интерфейса в Android-приложениях.

Fragment — модульный компонент пользовательского интерфейса Android, предназначенный для повторного использования интерфейсов внутри активности.

RecyclerView — компонент пользовательского интерфейса Android, предназначенный для эффективного отображения прокручиваемых списков элементов.

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

XML – eXtensible Markup Language

UI – User Interface

UX – User Experience

ORM – Object-Relational Mapping

CRUD – Create, Read, Update, Delete

SDK – Software Development Kit

API – Application Programming Interface

RTDB – Realtime Database

# ВВЕДЕНИЕ

Целью данной курсовой работы является разработка мобильного приложения «Домашняя Библиотека» для платформы Android, предназначенного для удобного хранения, просмотра и управления личной коллекцией книг пользователей. Приложение предоставляет функциональный и интуитивно понятный интерфейс для взаимодействия с библиотекой, поиска книг по различным параметрам, просмотра детальной информации о произведениях, авторах, а также для чтения книг в формате PDF.

Актуальность темы обусловлена популяризацией цифрового контента и переходом от традиционных форм хранения книг к цифровым платформам. Современные пользователи стремятся иметь быстрый и постоянный доступ к литературе, независимо от местоположения и устройства. «Домашняя Библиотека» решает задачи хранения, систематизации и организации книжного каталога, обеспечивая удобство взаимодействия и упрощая поиск необходимых произведений.

В рамках реализации данного проекта решаются следующие задачи: создание удобного интерфейса для просмотра каталога книг, организация поиска по названию, возможность просмотра детальной информации о книге, реализация пользовательской авторизации, настройка профиля пользователя с возможностью персонализации, а также интеграция с облачной базой данных Firebase Realtime Database для хранения и получения актуальных данных.

Объектом исследования выступает информационная система личной библиотеки. Предметом исследования являются подходы и методы разработки клиентской части мобильных приложений, а также способы взаимодействия с облачными хранилищами данных.

В исследовательской части работы анализируются существующие приложения для хранения и чтения книг, выделяются их преимущества и недостатки. Выполняется моделирование основных процессов взаимодействия пользователя с системой с использованием диаграмм в нотациях IDEF0 и DFD. В проектной части сформулированы функциональные требования, выбрана архитектура приложения и технологии реализации. В технологической части приведено описание и реализация ключевых компонентов приложения, таких как экраны авторизации и профиля, лента книг, механизмы поиска, настройки профиля и взаимодействие с Firebase, а также проводится тестирование реализованного функционала.

# ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

## Описание предметной области

В предметную область данной курсовой работы входят особенности цифрового представления информации о книгах и авторах, а также механизмы взаимодействия пользователей с этой информацией через мобильное приложение «Домашняя Библиотека». В рамках указанной области рассматриваются процессы просмотра, поиска, чтения и хранения данных о литературных произведениях, а также возможности персонализации пользовательского опыта.

Предметная область включает:

* каталог книг, содержащий основную информацию о произведениях (название, автор, жанр, описание), ссылки на PDF-файлы книг, а также изображения обложек;
* просмотр детальной информации о каждой книге, включая сведения об авторах и возможность загрузки произведения для чтения;
* поиск книг по названию с отображением результатов в удобном для пользователя виде;
* авторизацию пользователей с использованием электронной почты и учетных записей Google, ведение и управление пользовательскими сессиями;
* персонализацию пользовательского профиля, включая возможность изменения аватара, никнейма, адреса электронной почты и настройки языка интерфейса;
* хранение и оперативное обновление данных пользователей и книг с помощью облачного хранилища Firebase Realtime Database;
* обеспечение возможности чтения книг в формате PDF с использованием сторонних приложений.

Все вышеуказанные функции охватываются разрабатываемой системой и реализуются с использованием соответствующих программных и информационных компонентов, таких как мобильный клиент на платформе Android, облачная база данных Firebase, визуальные компоненты интерфейса и механизмы авторизации.

## Анализ существующих аналогов

При разработке мобильного приложения «Домашняя Библиотека» важно изучить уже существующие решения на рынке [1] персональных библиотек, чтобы определить их сильные и слабые стороны и сформировать конкурентоспособный продукт. Ниже приведён анализ трёх популярных приложений, реализующих функции хранения, поиска и чтения книг.

### Libib

Libib — кроссплатформенное приложение для каталогизации личных коллекций (книг, фильмов, музыки и видеоигр). Основная цель — быстро создать собственную «библиотеку» с возможностью сканировать штрихкоды ISBN и синхронизироваться между веб- и мобильными клиентами.

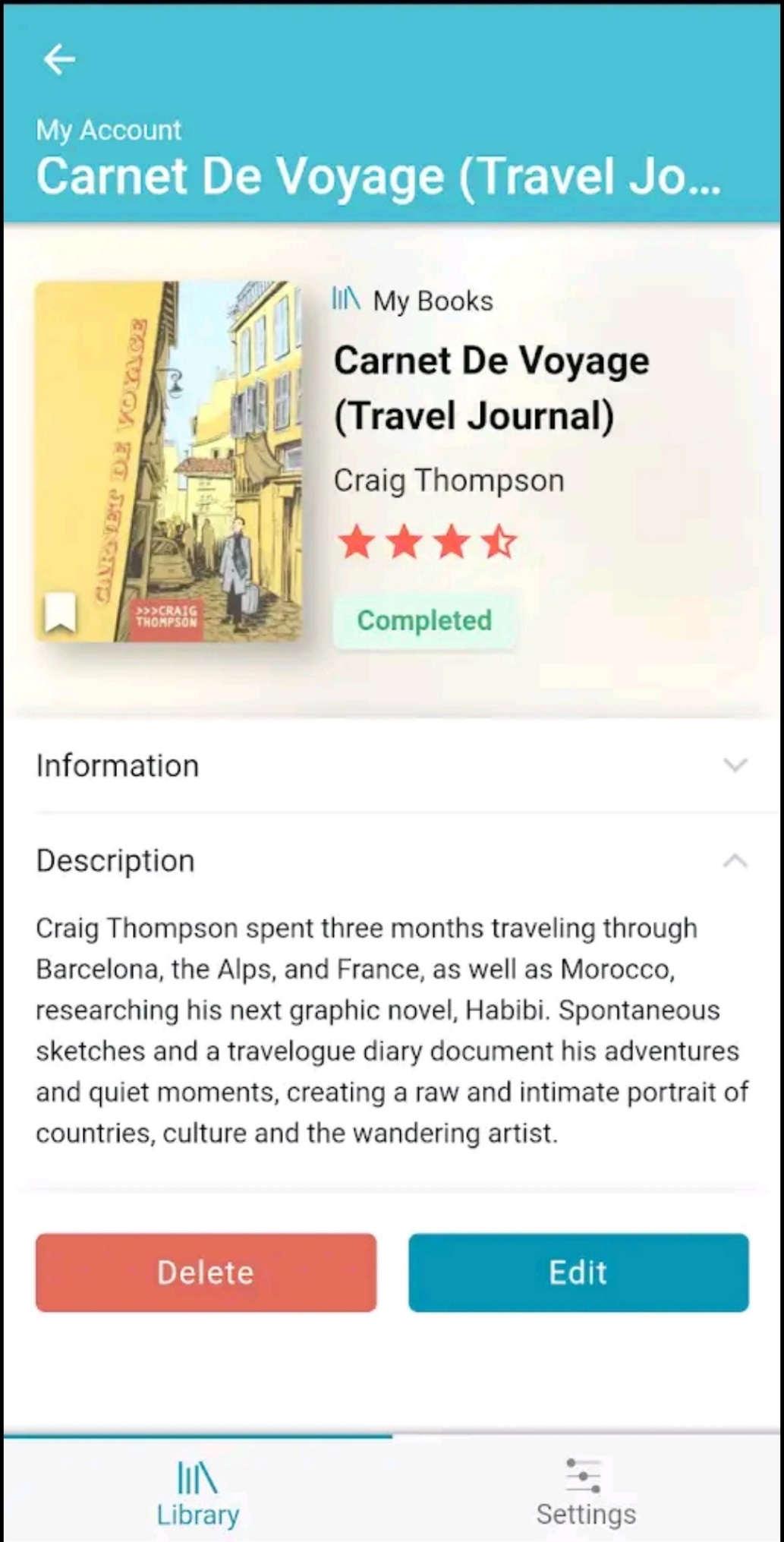
Достоинства:

* с**канирование штрихкодов (ISBN) – м**обильное приложение позволяет моментально считывать ISBN камерой и автоматически подтягивать данные о книге (название, обложка, автор) из онлайн-баз;
* множество коллекций – можно создать несколько «библиотек» (например, «Книги», «Фильмы», «Музыка») и вести их независимо. Бесплатный аккаунт поддерживает до 5 000 записей в каждом каталоге;
* импорт/экспорт – поддерживаются форматы XML/CSV/JSON для резервного копирования и передачи данных — удобно переносить коллекцию между разными приложениями;
* теги и фильтры – можно присваивать книгам набор тегов (жанр, тема, статус «прочитано/хочу прочитать») и быстро фильтровать коллекцию по любому тэгу.

Недостатки:

* нет встроенного чтения книг – Libib предназначен исключительно для каталогизации;
* ограничения бесплатной версии – бесплатный тариф ограничивает количество записей (до 5 000 книг на один каталог). Дополнительные возможности (история заимствований, расширенные отчёты) доступны только в Libib Pro;
* перегрузка функционала – для пользователя, заинтересованного лишь в хранении и чтении книг, часть опций (каталоги для фильмов, видеоигр и др.) может оказаться лишней, что усложняет интерфейс.

На Рисунке 1.2.1.1 продемонстрирован интерфейс мобильного приложения «Libib».

  
Рисунок 1.2.1.1 – Интерфейс Libib

### CLZ Books

CLZ Books — приложение от Collectorz.com, ориентированное исключительно на книги. Для работы требуется платная подписка: $1.95 / месяц или $19.95 / год (мобильная версия), а веб-версия стоит $39.95 / год.

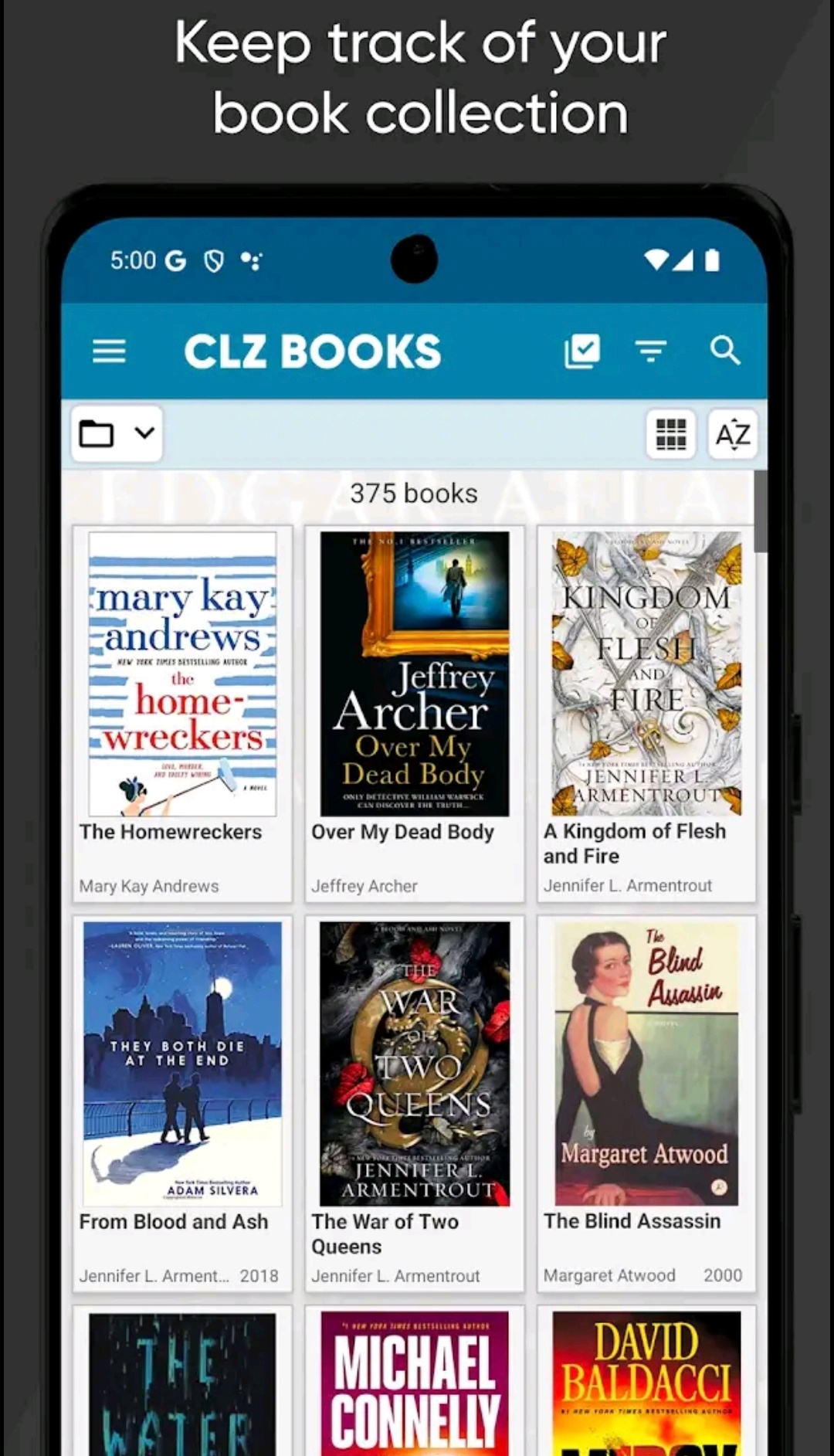
Достоинства:

* **очень подробные метаданные – п**ри сканировании ISBN загружаются полные сведения: серия, жанры, субъекты, описания, рейтинги, даже цены книг;
* **п**олная кастомизация – можно редактировать любые поля, менять обложки, добавлять примечания, состояние книги (состояние, цена покупки, место хранения).

Недостатки:

* отсутствие бесплатной версии – CLZ Books требует $19.95 / год только за мобильную версию.

На Рисунке 1.2.2.1 продемонстрирован интерфейс мобильного приложения «CLZ Books»

  
Рисунок 1.2.2.1 – Интерфейс CLZ Books

### ReadEra

ReadEra — бесплатное мобильное приложение-ридер, ориентированное на оффлайн-чтение. В отличие от облачных каталогизаторов, ReadEra не предоставляет автоматических онлайн-баз данных: все книги пользователь добавляет самостоятельно из локальной памяти устройства. Основное назначение — чтение PDF, EPUB, DOC и других форматов, с упором на комфорт и скорость обработки.

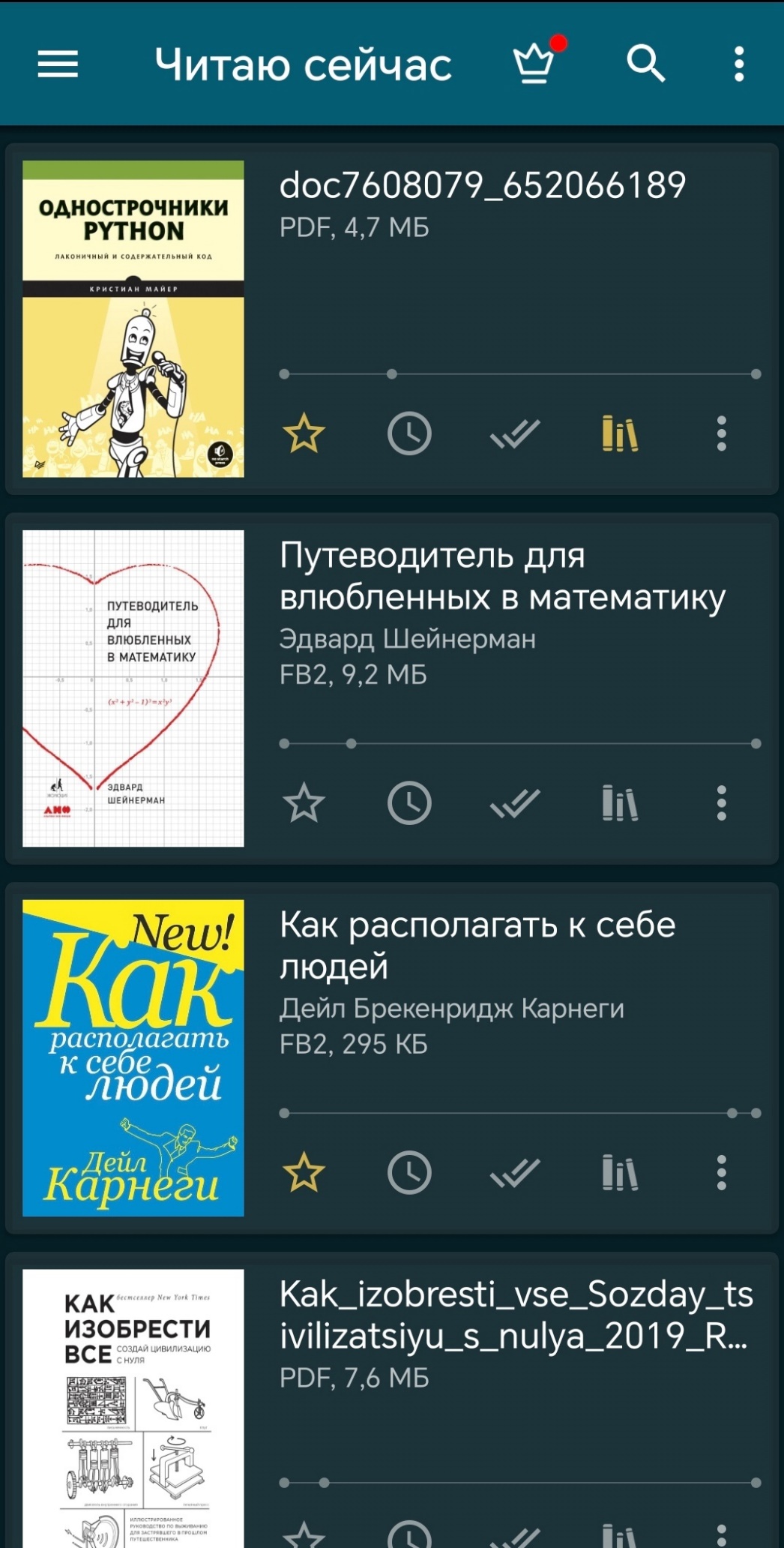
Достоинства:

* полностью бесплатен и без рекламы – отсутствуют платные подписки;
* оффлайн-режим – все книги хранятся локально, приложение не требует подключения к интернету;
* поддержка множества форматов – пользователь может читать любой формат без конвертации;
* удобный набор функций чтения – ночной режим, настройка параметров шрифта и поиска по тексту.

Недостатки:

* н**ет онлайн-каталога и метаданных – в**се книги требуется добавлять вручную — нет автозаполнения информации об авторах или обложек;
* отсутствие интеграции с облаком – все книги хранятся только на мобильном устройстве.

На Рисунке 1.2.3.1 показан интерфейс мобильного приложения «ReadEra».

  
Рисунок 1.2.3.1 – Интерфейс ReadEra

### Сравнение аналогов

Соединим полученные данные в таблицу сравнения (таблица 1.2.4.1).

*Таблица 1.2.4.1 – Таблица сравнения существующих аналогов*

|  |  |  |  |
| --- | --- | --- | --- |
| Приложение | Основные функции | Достоинства | Недостатки |
| Libib | Каталогизация, сканирование ISBN, синхронизация | Мультикаталоги, гибкий импорт/экспорт, сканирование ISBN | Отсутствие возможности чтения, лимиты бесплатного аккаунта, избыточность функционала |
| CLZ Books | Сканирование ISBN, полные метаданные, синхронизация, редактирование полей и обложек | Подробные метаданные, кастомизация записей, мощная синхронизация | Высокая стоимость, отсутствие бесплатной версии |
| ReadEra | Автоматическое чтение локальных файлов, оффлайн-чтение, поддержка множества форматов | Полностью бесплатное приложение, оффлайн-режим, богатые настройки чтения | Отсутствие онлайн-каталога и метаданных, отсутствие облачной базы |

## Техническое задание

### Определение пользовательских требований

На основании анализа существующих решений (Libib, CLZ Books, ReadEra), можно выделить следующие пользовательские ожидания и предпочтения:

* быстрый доступ к любым книгам из личной коллекции;
* возможность открытия книги для чтения;
* удобный поиск по названию книги;
* отсутствие или минимум ограничений в бесплатной версии;
* персонализация профиля (никнейм, аватар, язык интерфейса);
* отсутствие навязчивой рекламы.

В связи с этим, пользователи ожидают от создаваемого приложения:

* удобный и лаконичный интерфейс для просмотра каталога книг;
* оперативный поиск по названию книги;
* возможность чтения книги в PDF-формате;
* просмотр детальной информации о книге (жанр, обложка, описание, список авторов);
* работа без жёстких лимитов — бесплатно читать любую книгу;
* простая авторизация (e-mail/пароль) и сохранение сессии;
* настройка личного профиля (никнейм, аватар, смена языка) без сложных форм.

### Определение функциональных требований

На основе пользовательских требований и анализа аналогов, формулируются следующие функциональные требования к мобильному приложению:

1. Каталог книг:

* загрузка списка книг из Firebase Realtime Database;
* отображение книг в виде прокручиваемой ленты карточек (обложка + название + жанр);
* реализация поиска по названию книги (регистронезависимо).

1. Детальная информация о книге:

* просмотр полной информации: название, жанр, описание, обложка, список авторов;
* подгрузка имён авторов по их ID;
* кнопка «Читать» — загрузка PDF в локальное хранилище и открытие PDF-ридера;

1. Система авторизации и регистрации:

* регистрация пользователя по e-mail и паролю (Firebase Auth);
* авторизация и вход существующего пользователя;
* сохранение активной сессии (Firebase Auth + SharedPreferences, чтобы не запрашивать вход при каждом запуске).

1. Работа с PDF-файлами:

* скачивание PDF через DownloadManager с индикацией прогресса, сохранение в папку «/documents/books/»;
* открытие PDF во внешнем ридере (Intent);
* если файл уже загружен, открывать локальную копию без повторного скачивания.

1. Настройки приложения:

* изменение никнейма (сохранение в RTDB);
* изменение e-mail с отправкой письма подтверждения;
* изменение аватара через загрузку url;
* изменение языка приложения;
* выход из аккаунта в приложении (возврат к экрану авторизации).

1. Работа с базами данных:

* использование Firebase Realtime Database;
* узлы books (метаданные книг), authors (данные об авторах), users (профили пользователей);
* ValueEventListener для «живых» обновлений списка книг и данных профиля;
* SharedPreferences для хранения UID текущего пользователя и выбранного языка.

1. Интерфейс и визуальные компоненты:

* использование Material Design (Toolbar, BottomNavigation, CardView, RecyclerView);
* XML-макеты;
* загрузка изображений через Glide с placeholder;
* Navigation Component для навигации между фрагментами и сохранения back-stack.

1. Производительность и UX:

* высокая скорость отклика при скролле списка (RecyclerView с GridLayoutManager или LinearLayoutManager);
* обработка ошибок (отображать Toast «Ошибка авторизации»);
* загрузка изображений через Glide с placeholder;
* стабильная работа при медленном интернете (показ placeholder’ов);

# ПРОЕКТНАЯ ЧАСТЬ

## Описание процесса в нотации IDEF0

Функциональная модель в нотации IDEF0 описывает совокупность взаимосвязанных процессов (функций) [2], реализуемых внутри приложения «Домашняя Библиотека», а также их взаимодействие с внешней средой. Каждый функциональный блок на такой диаграмме представлен прямоугольником и отвечает за конкретную функцию системы. По стандарту IDEF0 на одном уровне декомпозиции помещают от трёх до шести блоков, а на контекстном уровне обычно изображают один главный блок, отражающий всё приложение.

Каждый тип стрелок на диаграмме несёт свою семантику:

* вход (input) – данные, преобразуемые функцией для получения результата;
* выход (output) – результат преобразования, создаваемый функцией;
* управление (control) – правила, ограничения и условия, определяющие поведение функции;
* механизм (mechanism) – ресурсы или инструменты, обеспечивающие выполнение функции.

Ниже – описание контекстной диаграммы приложения «Домашняя Библиотека».

Центральный блок («Домашняя Библиотека») обобщённо описывает весь процесс работы мобильного приложения – от приёма пользовательских действий до формирования интерфейса.

Входные данные (input):

* список книг из Firebase;
* введённые пользователем данные авторизации;
* идентификатор книги, заинтересовавшей пользователя;
* введённые пользователем новые данные настроек.

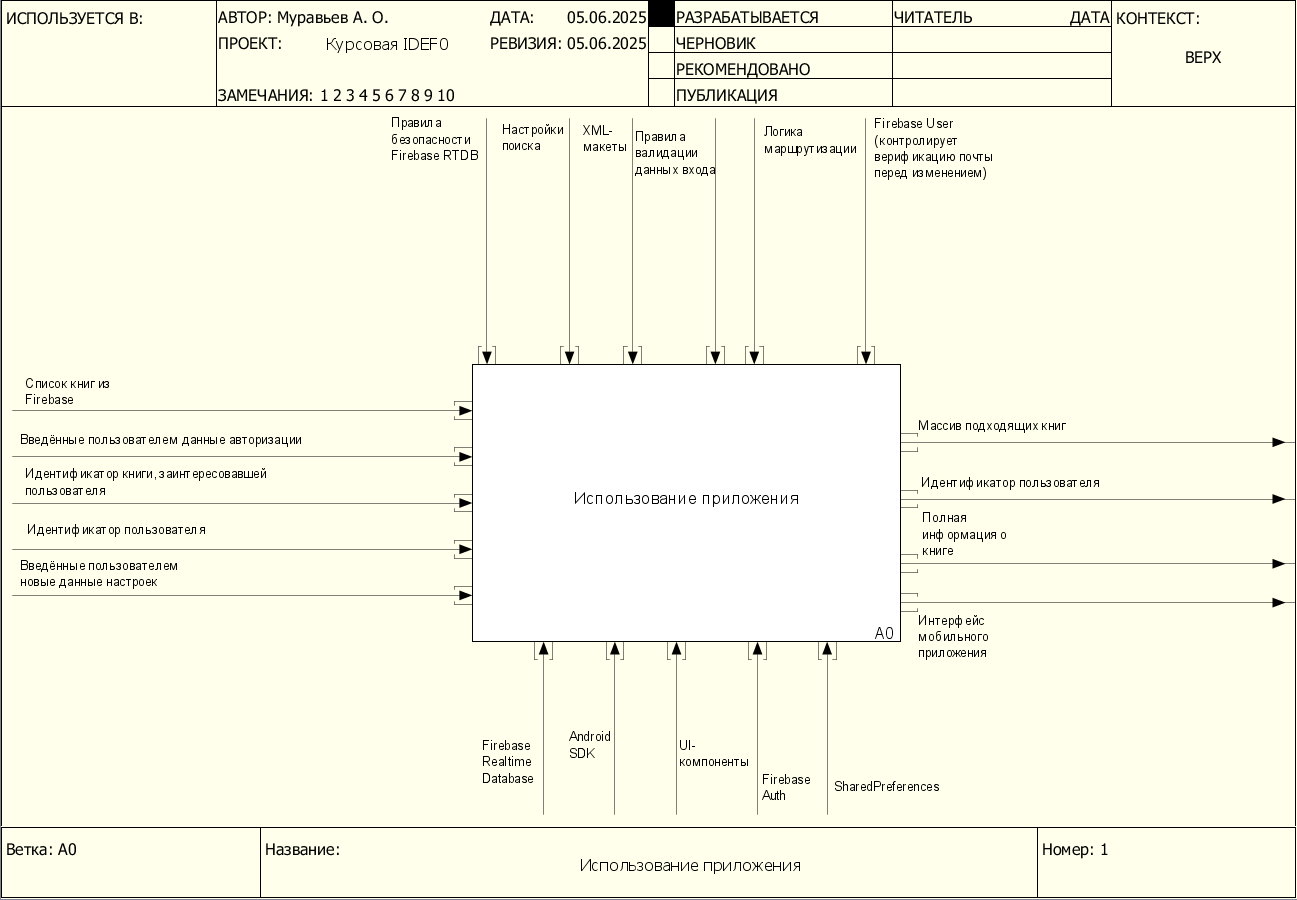
Управляющие воздействия (control):

* правила безопасности Firebase RTDB;
* настройки поиска;
* XML-макеты;
* правила валидации данных входа;
* логика маршрутизации;
* Firebase User (контролирует верификацию почты перед изменениями настроек).

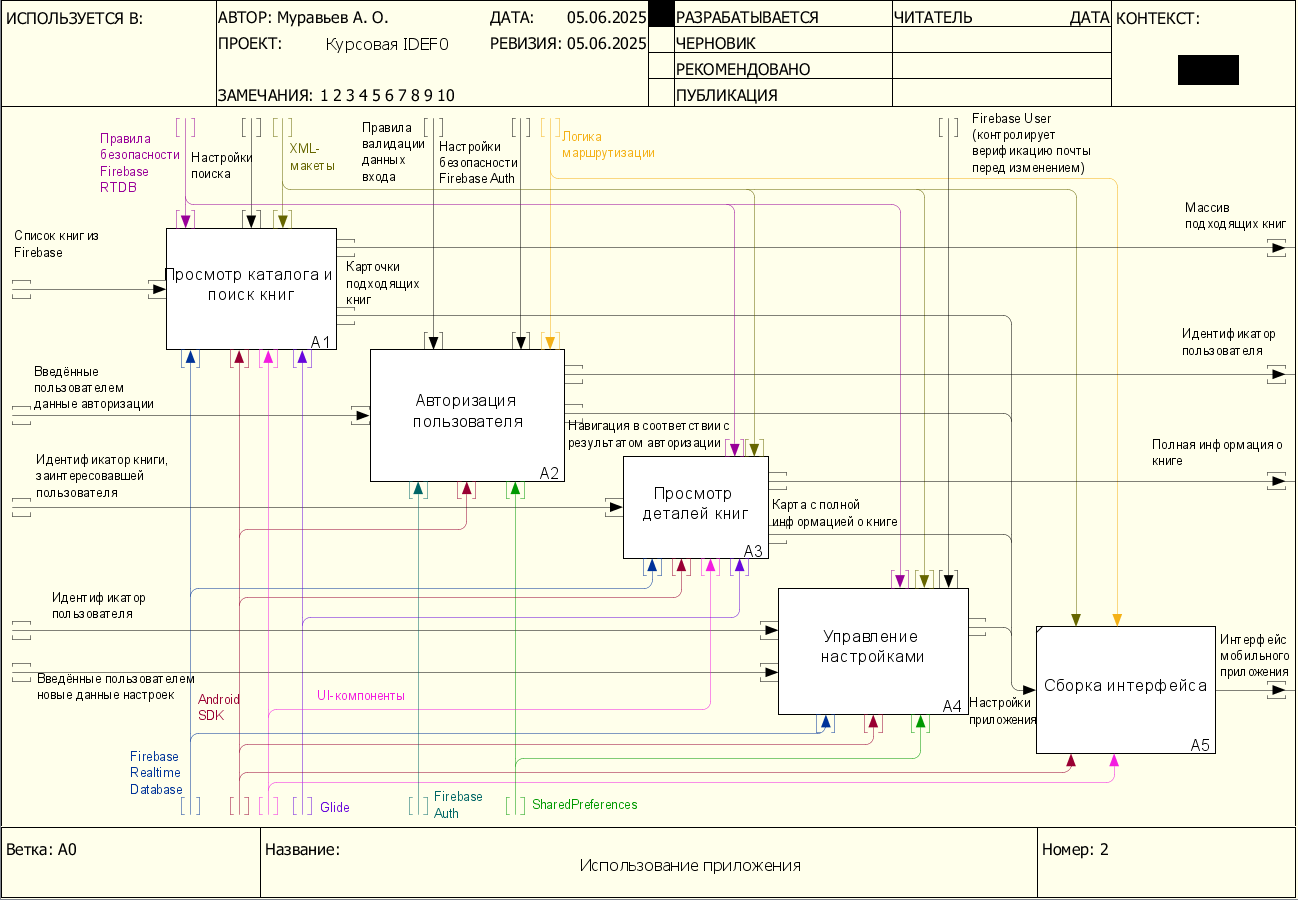
Механизмы (mechanism):

* Firebase Realtime Database;
* Android SDK;
* UI-компоненты;
* Firebase Auth;
* SharedPreferences.

Покажем на Рисунке 2.1.1 диаграмму процесса в нотации IDEF0.

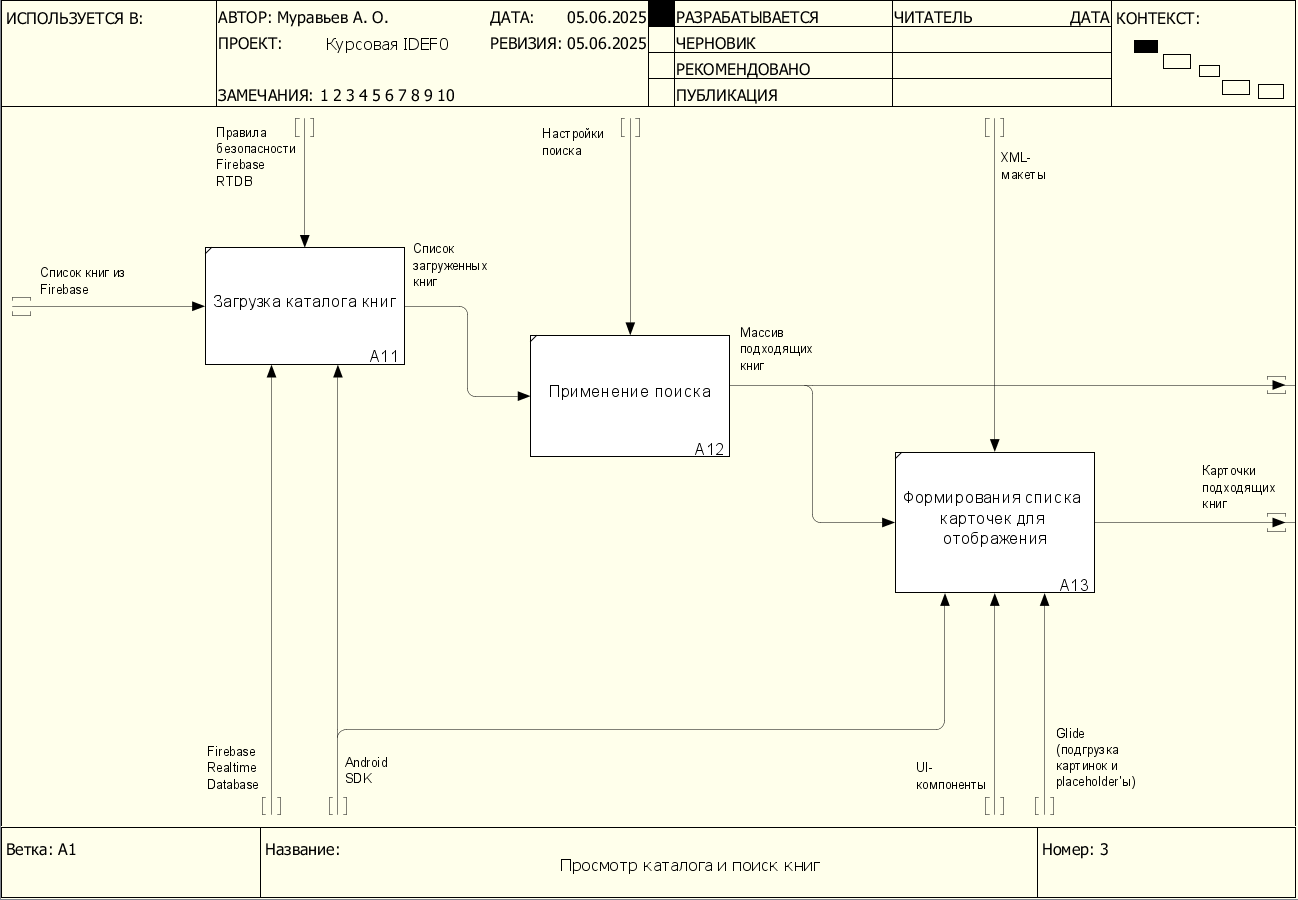
  
Рисунок 2.1.1 – Диаграмма А0 в нотации IDEF0

Декомпозиция процесса использования прототипа приложения представлена на Рисунке 2.1.2.

  
Рисунок 2.1.2 – Декомпозиция процесса «Использование приложения» в нотации IDEF0

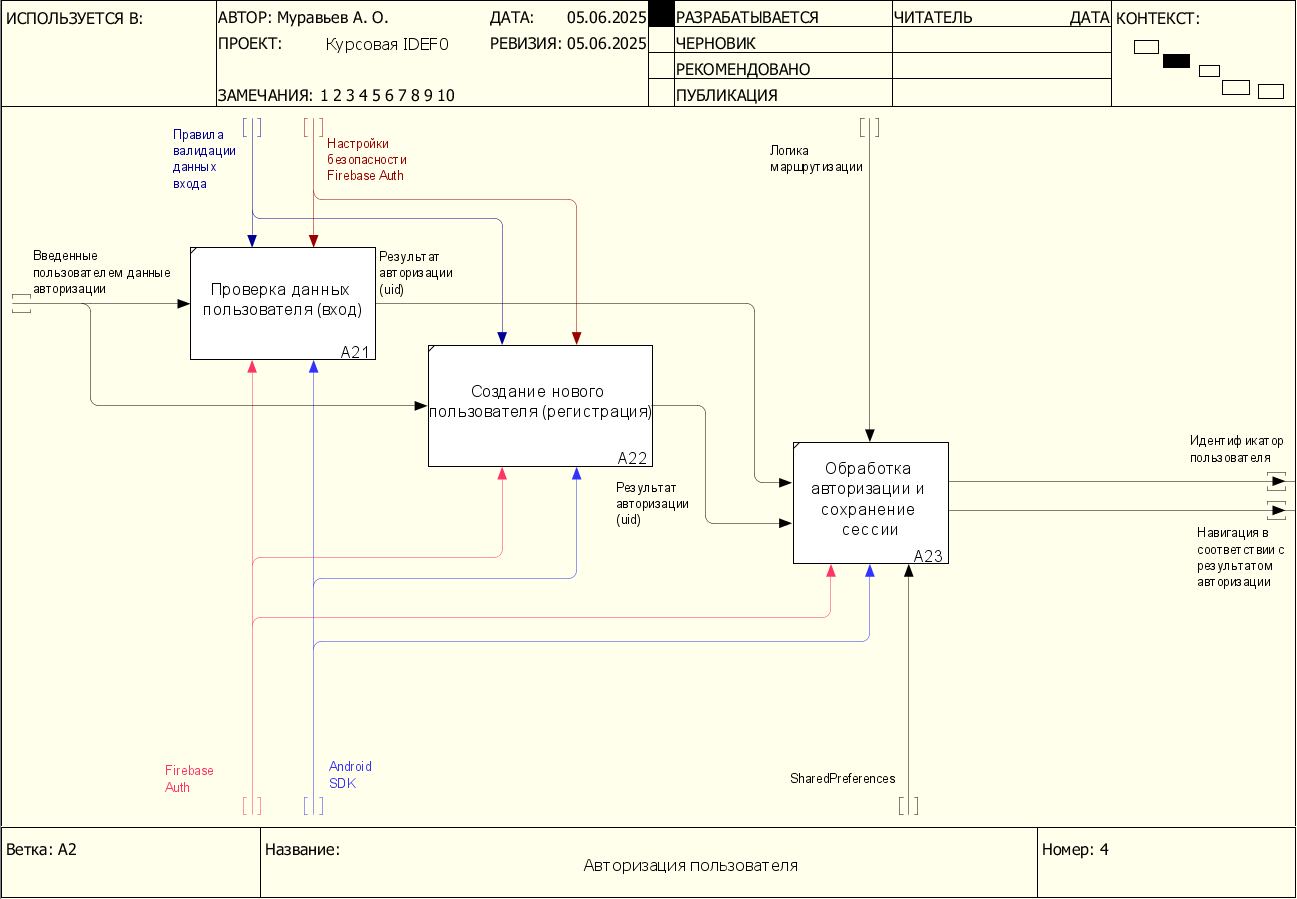
Процесс просмотра каталога и поиска книг включает в себя загрузку каталога книг из Firebase с учетом правил безопасности Firebase RTDB, применение условий поиска c использованием настроек поиска и формирование списка карточек для отображения по XML-макетам. В самом процессе участвуют Firebase Realtime Database, Android SDK, UI-компоненты и Glide (для подгрузки картинок и placeholder’ов). На выход подаются массив подходящих книг и их карточки.

Декомпозицию процесса просмотра каталога и поиска книг можно увидеть на Рисунке 2.1.3.

  
Рисунок 2.1.3 – Декомпозиция процесса «Просмотр каталога и поиск книг» в нотации IDEF0

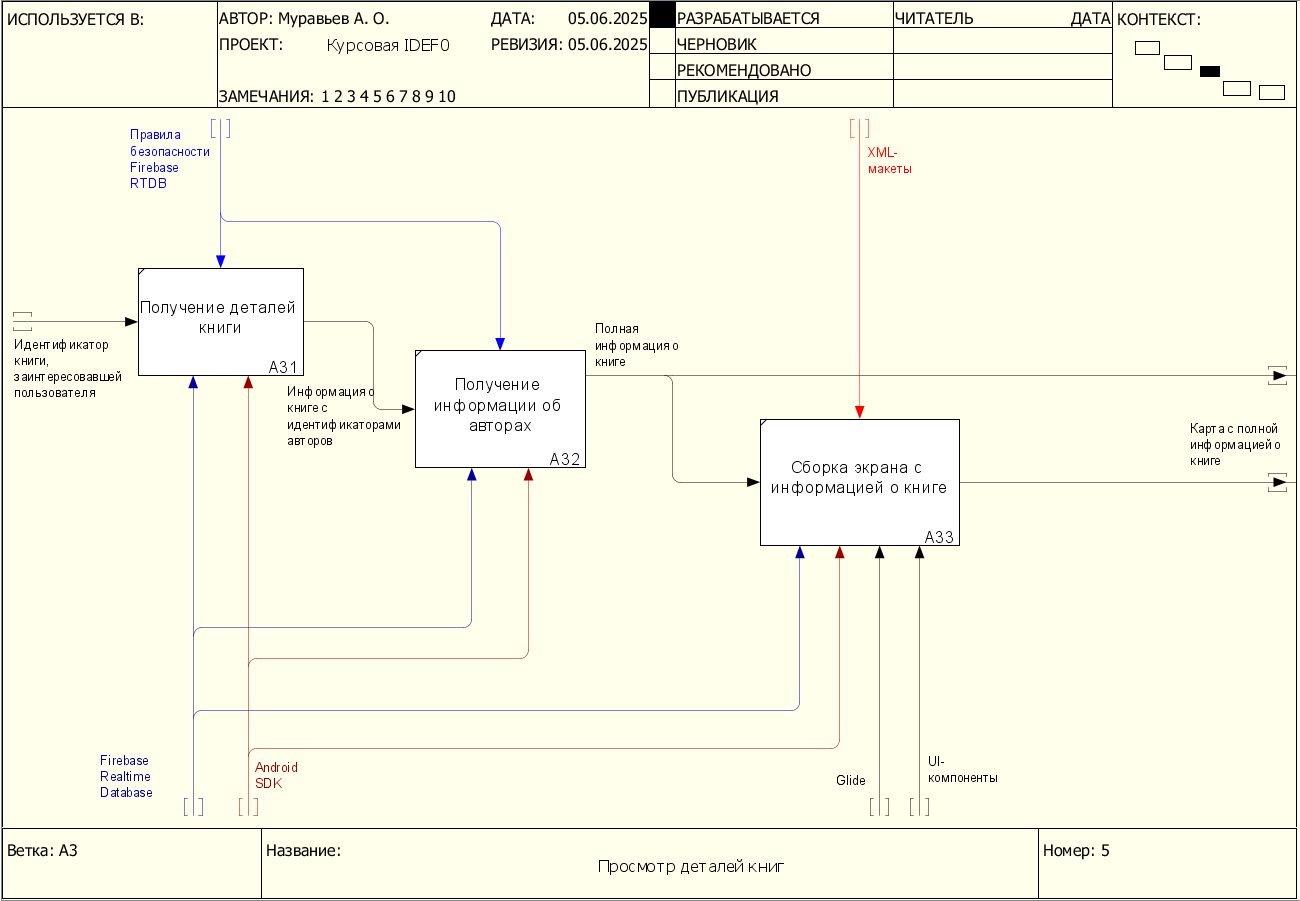
Для авторизации в системе на вход необходимы введённые пользователем данные авторизации, которые проходят правила валидации данных входа и настройки безопасности Firebase Auth. Данные необходимы и для регистрации, и для входа. Они проходят через обработку авторизации и сохранение сессии. На выход с авторизации поступает идентификатор пользователя и дальнейшую навигацию в соответствии с результатом авторизации с учетом логики маршрутизации. В процессе также участвуют Firebase Auth, Android SDK и SharedPreferences.

Декомпозиция процесса авторизации пользователя продемонстрирована на Рисунке 2.1.4.

  
Рисунок 2.1.4 – Декомпозиция процесса «Авторизация пользователя» в нотации IDEF0

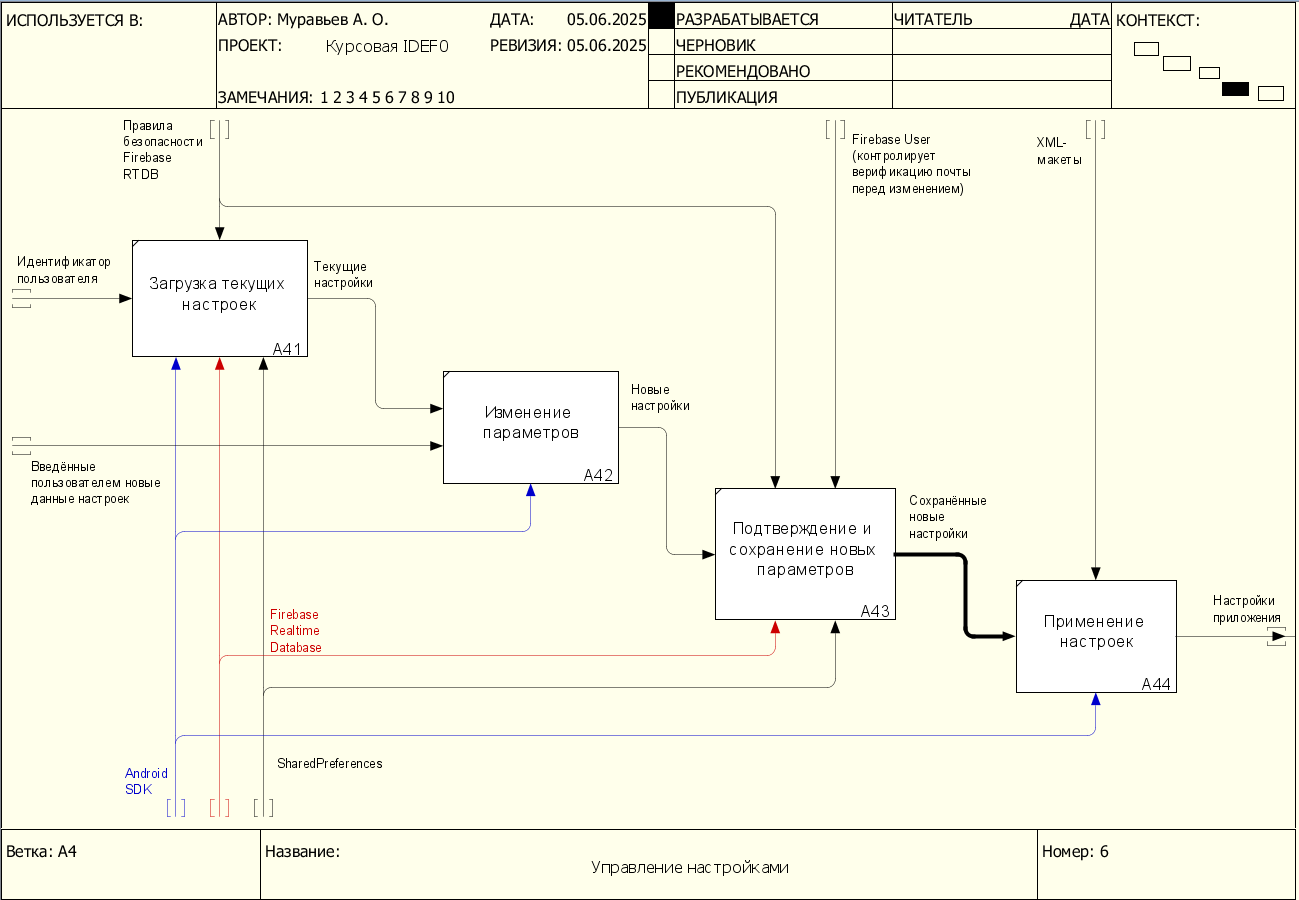
Процесс просмотра деталей книги принимает в себя идентификатор книги, с помощью которого получает детали книги с учетом правил безопасности Firebase RTDB из базы данных, получает информацию об авторах книг, собирает экран со всей информацией о книге через XML-макеты и возвращает полную информацию о книге и её графическую карту.

Декомпозиция этого процесса показана на Рисунке 2.1.5.

  
Рисунок 2.1.5 – Декомпозиция процесса «Просмотр деталей книг» в нотации IDEF0

Управление настройками, принимая в себя идентификатор пользователя и настройки, которые нужно изменить, загружает текущие настройки пользователя с учётом правил безопасности Firebase RTDB, создаёт новые настройки, подтверждает их, проверяя в Firebase User, сохраняет их и применяет, возвращая новые настройки.

Декомпозицию процесса управления настройками можно наблюдать на Рисунке 2.1.6.

  
Рисунок 2.1.6 – Декомпозиция процесса «Управление настройками» в нотации IDEF0

## Описание процесса в нотации DFD

Диаграмма в нотации DFD иллюстрирует движение информационных потоков в рамках работы мобильного приложения «Домашняя библиотека». Стрелками обозначены данные, входящие в процесс и исходящие из него. Процессы представлены прямоугольниками со скруглёнными краями. Хранилища данных обозначены прямоугольниками без правой стороны, а внешние сущности – прямоугольниками с толстыми линиями.

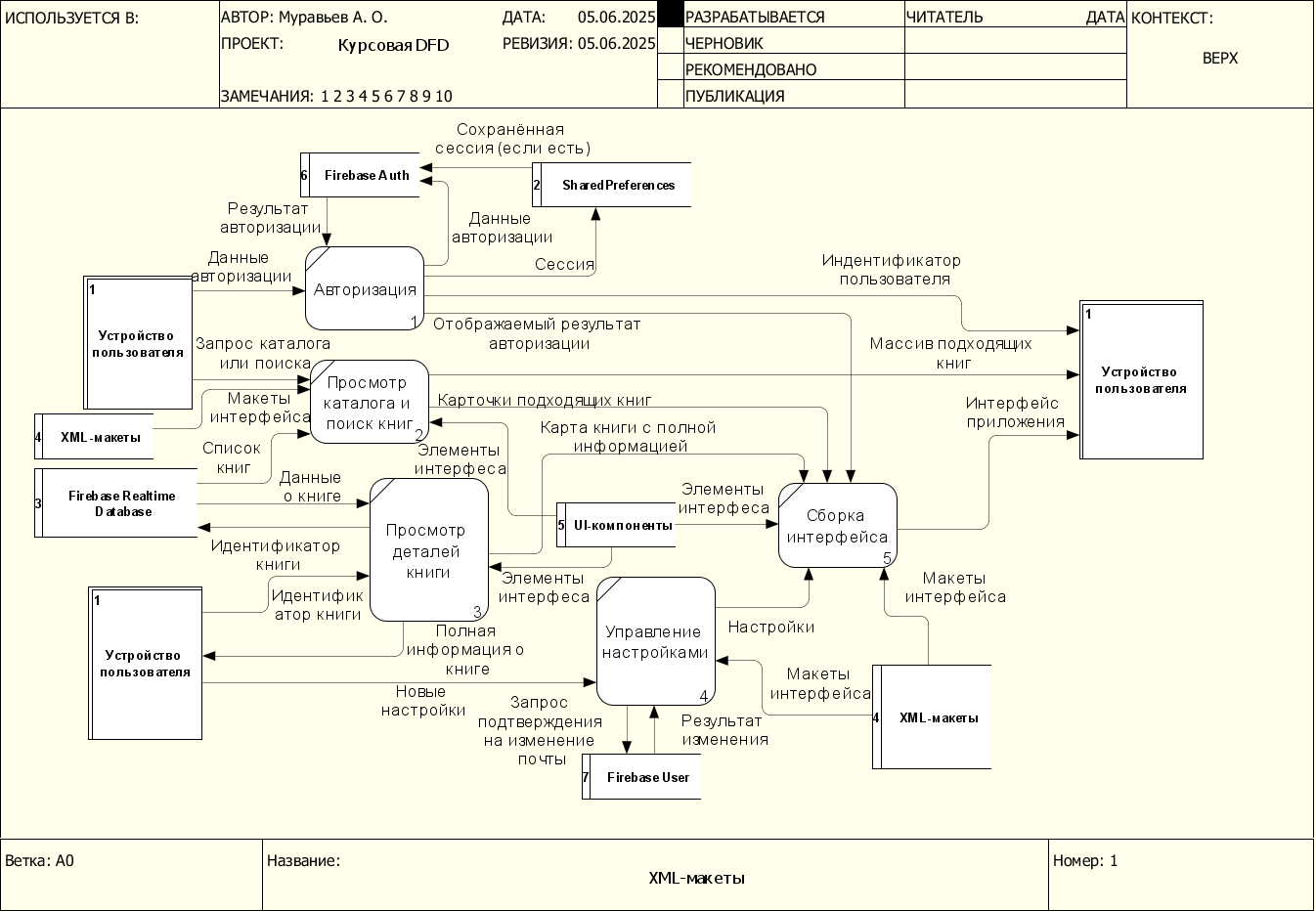
Сначала приложение собирает и обрабатывает информацию из внешних и внутренних источников. Входными данными выступают список книг из внешнего источника (Firebase), авторизационные данные пользователя, идентификаторы книг и пользовательские настройки. Эти данные, после обработки и проверки, передаются в базу данных приложения.

Процесс авторизации объединяет регистрацию новых пользователей и вход существующих, включая возможность восстановления доступа. На вход этого процесса поступают электронная почта, пароль и текущие пользовательские данные. В результате формируется токен авторизации, пользовательский идентификатор и данные сессии, которые сохраняются локально.

Процесс просмотра каталога и поиска книг начинается с запроса от устройства пользователя. Приложение загружает каталог книг из хранилища данных Firebase с учётом заданных параметров поиска и правил безопасности. Обработанный список книг передаётся обратно на устройство пользователя для отображения.

Просмотр деталей конкретной книги осуществляется при получении идентификатора книги от пользователя. Приложение обращается к базе данных за информацией о книге и её авторах, формирует экран с полной информацией и передаёт данные обратно пользователю, включая графическое представление книги.

Управление настройками предполагает принятие новых параметров от пользователя вместе с его идентификатором. Текущие настройки загружаются и проверяются, новые настройки создаются и подтверждаются через механизм безопасности Firebase. После сохранения новых параметров обновлённые настройки направляются обратно пользователю.

  
Рисунок 2.2.1 – Декомпозиция использования приложения в нотации DFD

## Описание и обоснование выбора программного обеспечения

Для разработки мобильного приложения «Домашняя библиотека» выбрана среда Android Studio, официальная IDE от Google, предоставляющая полный набор инструментов для создания и отладки Android-приложений. Основным языком программирования является Java, так как он поддерживает обширную библиотечную базу Android SDK, обладает стабильностью, хорошей документацией и простотой интеграции с фреймворком Android.

В проекте активно используются компоненты Firebase (Auth и Realtime Database), которые обеспечивают быструю и безопасную работу с данными, авторизацию пользователей и хранение контента. Также применяются SharedPreferences для хранения пользовательских настроек и AndroidX Navigation Component для удобной и надёжной организации навигации между экранами приложения. Для реализации интерфейса выбраны Material Components, RecyclerView и LinearLayout, которые позволяют создавать удобный и адаптивный пользовательский опыт. Для быстрой и оптимизированной загрузки изображений используется библиотека Glide.

## Архитектура программной среды

В ходе разработки проекта структура Java-классов была логически организована по пакетам, каждый из которых отвечает за отдельную область функциональности:

1. data – содержит все классы и пакеты для работы с базой данных:

* DBManager – менеджер для базовых операций с базой данных;
* AuthManager – менеджер для работы с авторизацией через Firebase Auth;
* BookDownloadManager – менеджер для работы с файлами книг;
* SearchManager – менеджер для работы с поиском по базе данных.

1. data.models – содержит модели-схемы для работы с базой данных:

* Author – схема данных для автора книги;
* Book – схема данных для книги;
* User – схема данных для пользователя;

1. ui – содержит все классы и пакеты для работы интерфейса приложения:

* MainActivity – входная точка приложения, в которой оно запускается.

1. ui.auth – содержит все классы, относящиеся к интерфейсу авторизации:

* AuthHostFragment – общий класс-фрагмент, на котором действуют другие фрагменты авторизации;
* AuthViewModel – общий класс для выполнения логики авторизации и взаимодействия с менеджерами;
* AuthFragment – абстрактный класс-фрагмент для создания различных экранов авторизации (вход и регистрация);
* LoginFragment – класс-фрагмент для входа через e-mail и пароль;
* RegisterFragment – класс-фрагмент для регистрации через e-mail и пароль.

1. ui.book – содержит классы, связанные с отображением книг на экране:

* BookCardFragment – класс-фрагмент для отображения небольшой карточки книги;
* BookDetailFragment – класс-фрагмент для отображения подробной информации о книге.

1. ui.common – содержит вспомогательные классы для работы с интерфейсом:

* AvatarUtil – ненаследуемый класс, созданный для генерации и работы с аватарами пользователей;
* LocaleUtil – ненаследуемый класс, созданный для взаимодействия с локализацией приложения и переключения языка.

1. ui.home – хранит классы домашней страницы:

* HomeHostFragment – общий класс-фрагмент, на котором действуют другие фрагменты домашней страницы
* LibraryFragment – класс-фрагмент для отображения экрана библиотеки;
* BookFeedFragment – класс-фрагмент для отображения ленты из карточек книг;
* SettingsFragment – класс-фрагмент, отвечающий за настройки приложения;
* ProfileFragment – класс-фрагмент для отображения экрана профиля.

В директории res используется стандартная структура:

* drawable/ – изображения и иконки;
* layout/ – XML-макеты экранов и фрагментов;
* menu/ – хранит XML-файл навигации нижней панели;
* mipmap/ – хранит иконки запуска приложения в разных разрешениях (например, ic\_launcher);
* navigation/ – содержит графы навигации, которые описывают переходы между экранами;
* xml/ – используется для хранения дополнительных настроек, таких как prefs\_settings.xml.

# ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

## Описание моделей и структур данных

Класс Book (пакет data.models) имеет достаточно простую структуру.

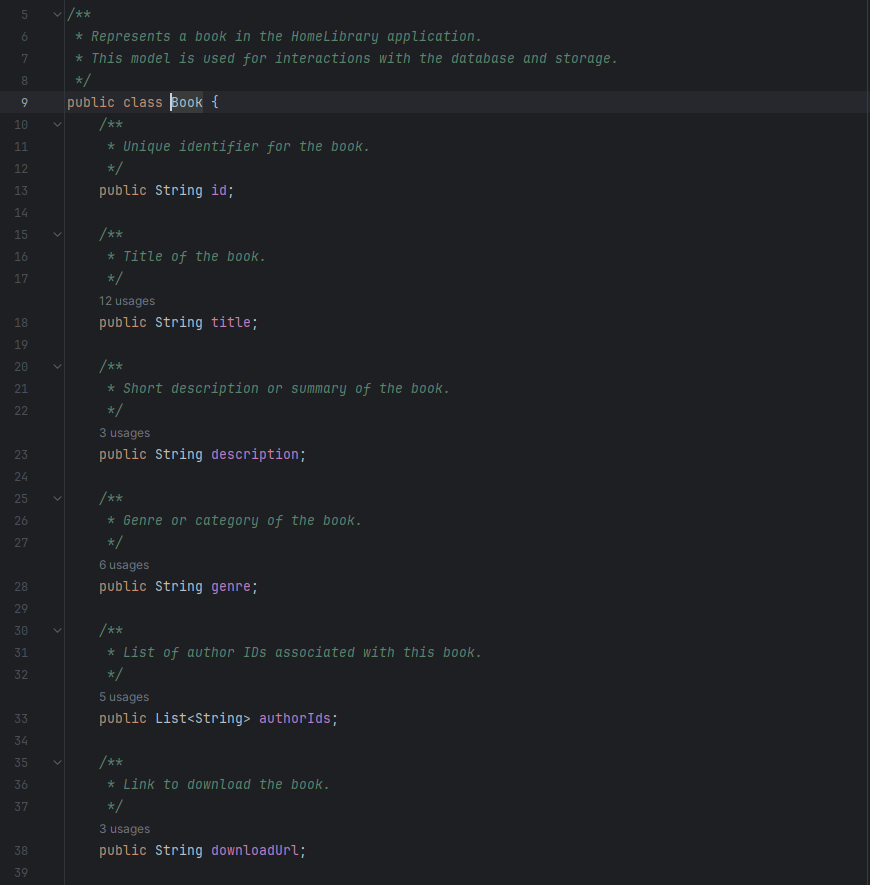
Основные поля включают:

* id – первичный ключ (генерируется автоматически);
* title – название книги;
* description – описание книги;
* genre – жанр книги;
* auhorIds – идентификаторы авторов книги;
* downloadUrl – ссылка для скачивания книги;
* imageUrl – ссылка на обложку книги.

А также два конструктора:

* пустой конструктор без параметров, требуемый Firebase;
* конструктор с параметрами id, title, description, genre, authorIds, downloadUrl, imageUrl – для создания экземпляра класса внутри программы.

Покажем первую часть кода класса Book на Рисунке 3.1.1.

  
Рисунок 3.1.1 – Код класса Book, часть 1

Вторая часть кода класса Book продемонстрирована на Рисунке 3.1.2.

  
Рисунок 3.1.2 – Код класса Book, часть 2

Другие модели устроены по тому же принципу.

Класс DBManager написан по паттерну проектирования singleton и содержит статическое приватное поле instance, доступ к которому организован через метод getInstance(). Геттер-метод имеет ключевое слово synchronized, улучшая потокобезопасность класса.

DBManager управляет данными авторов, книг и пользователей в базе данных через инструменты Firebase.

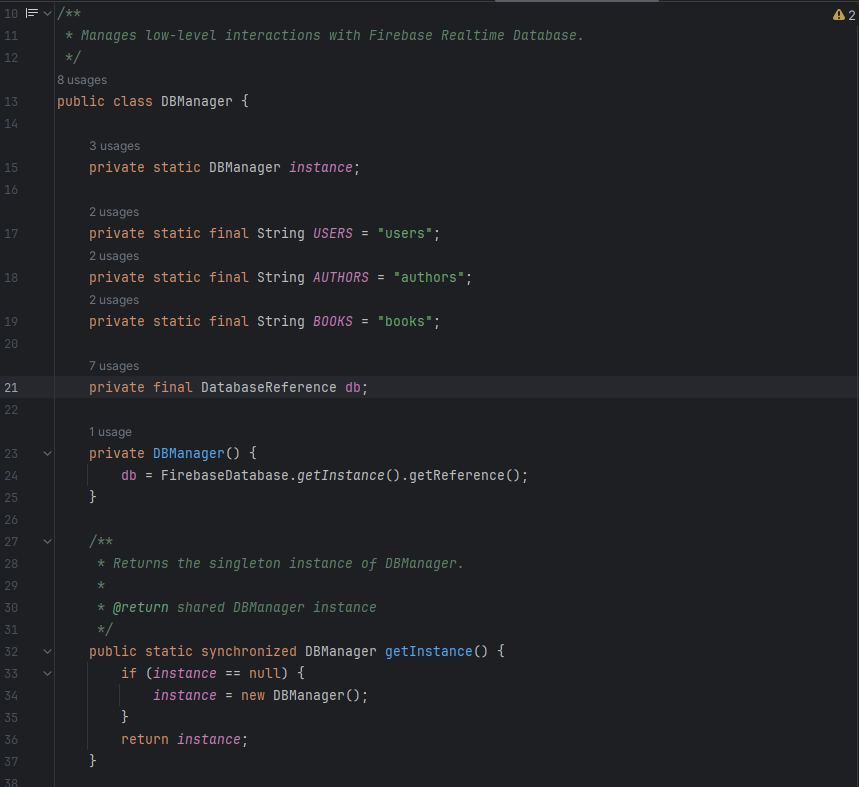
Другие поля DBManager включают:

* USERS – статическое приватное неизменяемое поле, хранящее название ветви в базе данных, отвечающей за пользователей;
* AUTHORS – статическое приватное неизменяемое поле, хранящее название ветви в базе данных, отвечающей за авторов;
* BOOKS – статическое приватное неизменяемое поле, хранящее название ветви в базе данных, отвечающей за книги;
* db – приватное неизменяемое поля, хранящее ссылку на базу данных Firebase Realtime Database.

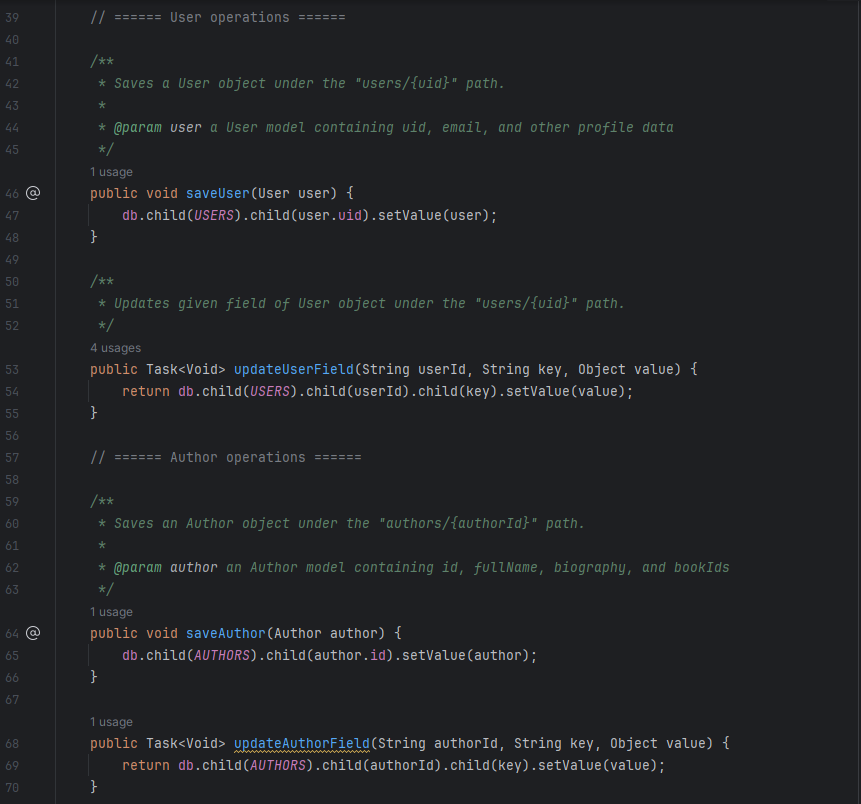
Также класс включает в себя несколько методов для работы с данными. Вот основные:

* saveUser() – метод для сохранения объекта пользователя;
* updateUserField() – метод для обновления определённого поля пользователя.

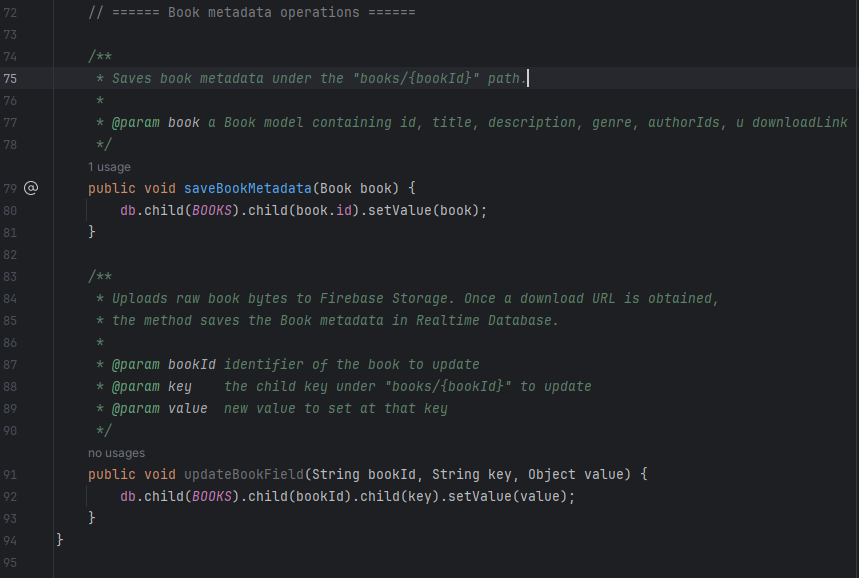
А также аналогичные методы для авторов и книг. Первая часть класса DBManager показана на Рисунке 3.1.3.

  
Рисунок 3.1.3 – Код класса DBManager, часть 1

Вторая часть кода класса DBManager на Рисунке 3.1.4.

  
Рисунок 3.1.4 – Код класса DBManager, часть 2

Последняя часть кода класса DBManager на Рисунке 3.1.5.

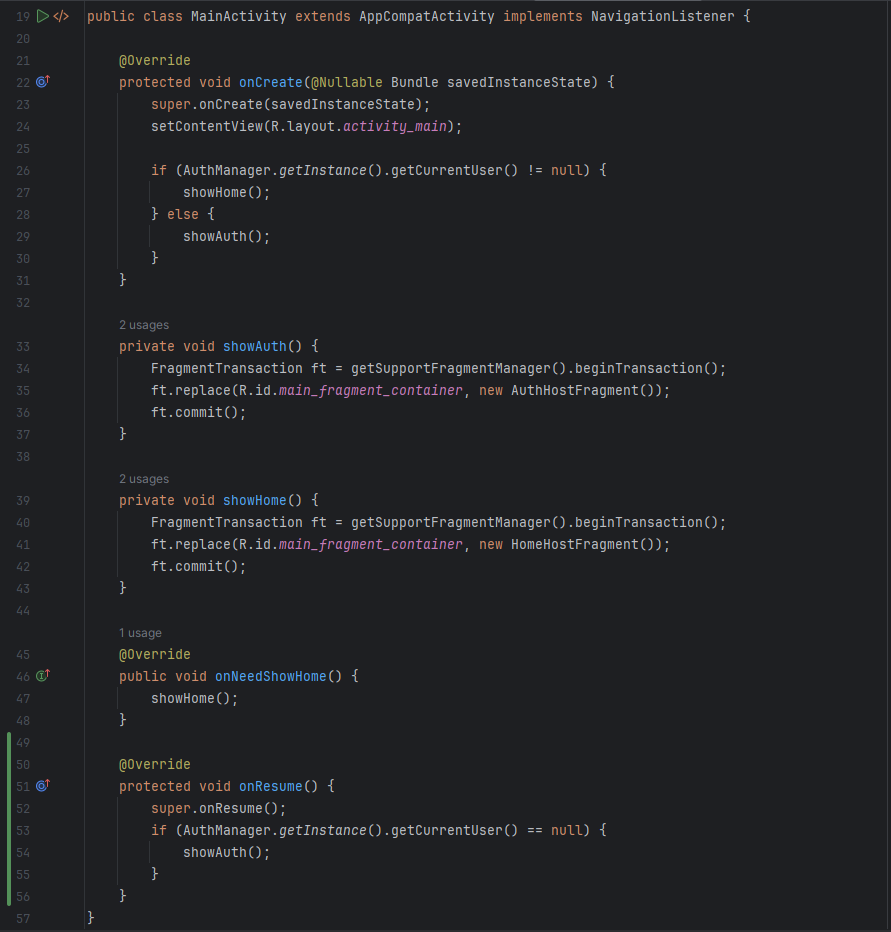
  
Рисунок 3.1.5 – Код класса DBManager, часть 3

Другие менеджеры построены по схожей схеме, но имеют более специфические и гибкие методы для работы с базой данных или локальными данными (как в случае с BookDownloadManager).

Класс MainActivity, являясь главным и единственным активити, загружает фрагменты на экран пользователя.

В зависимости от того, авторизован пользователь или нет, класс загружает фрагмент авторизации или домашний фрагмент.

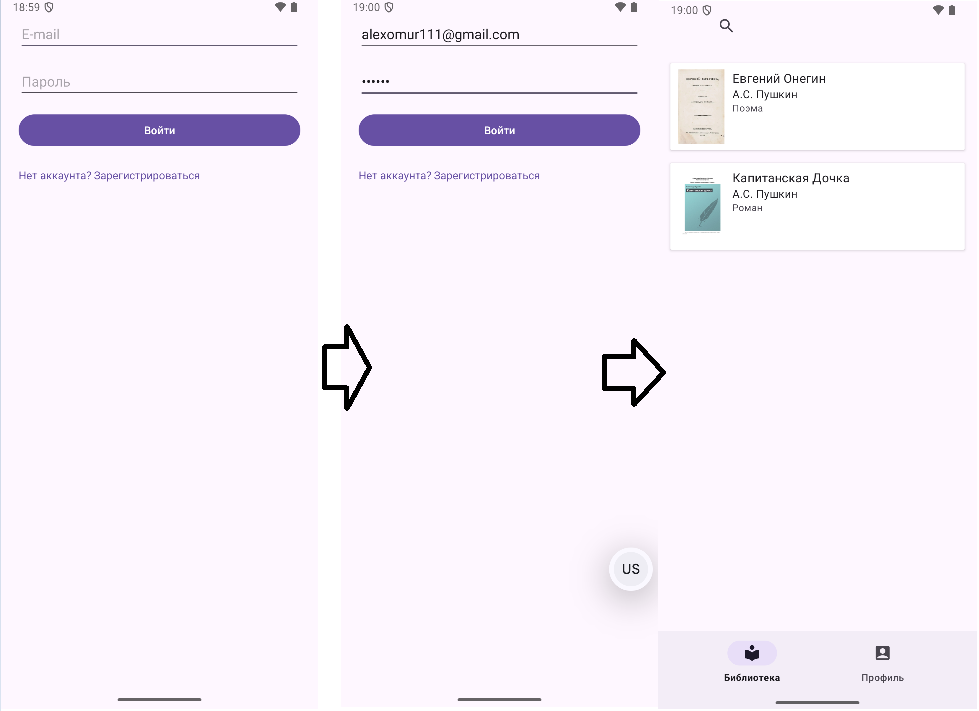
Код класса MainActivity представлен на Рисунке 3.1.6.

  
Рисунок 3.1.6 – Код класса MainActivity

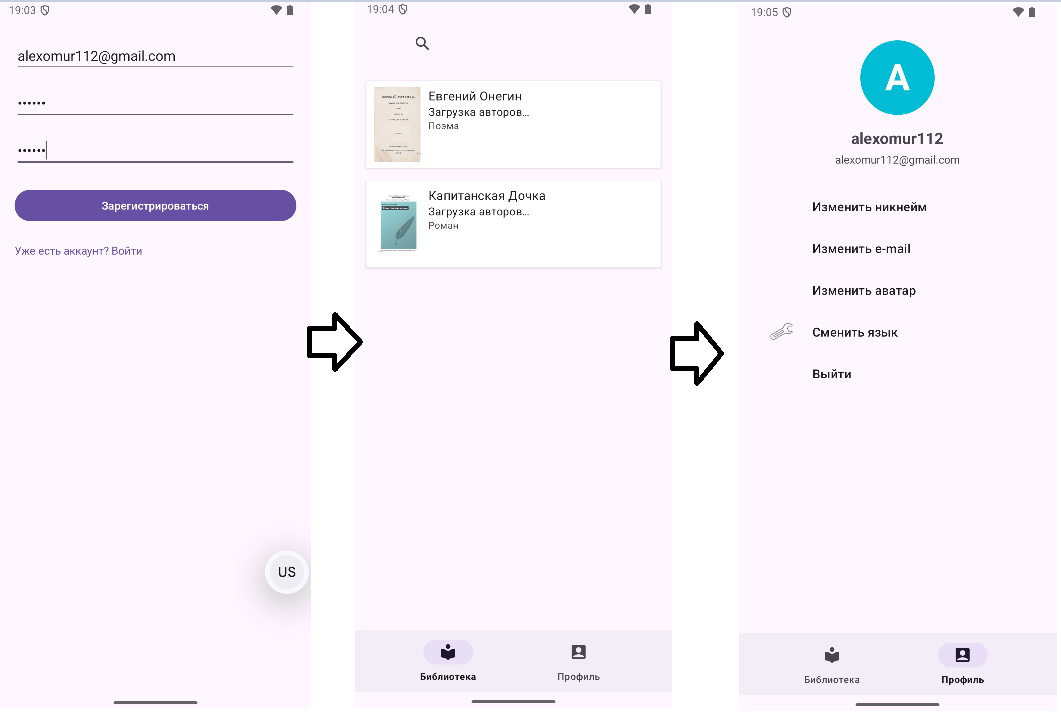
## Тестирование программного продукта

Для проведения тестирования мобильного приложения был использован метод черного ящика – метод, при котором тестировщику неизвестно внутреннее устройство тестируемой системы. Для этого нескольким пользователям было выдано приложение, в ходе проверки которого были протестированы все функциональные возможности системы.

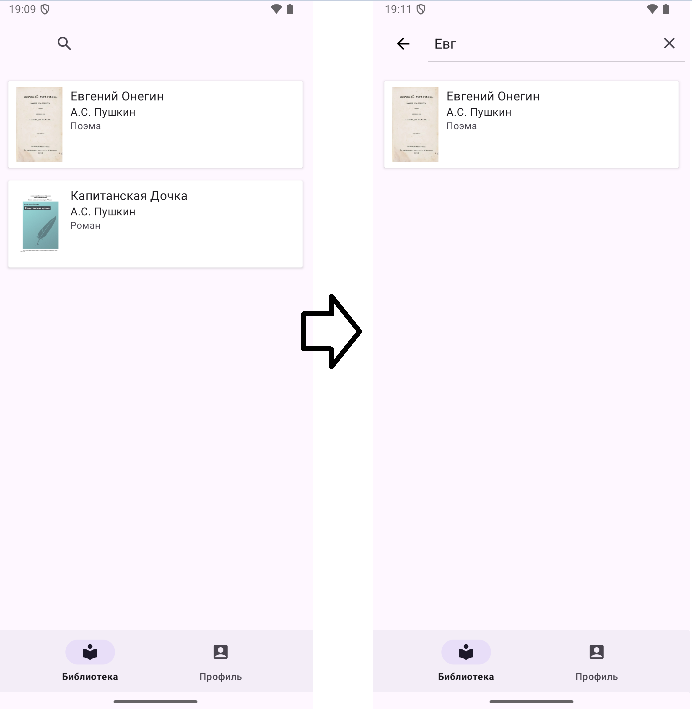
На Рисунке 3.2.1 результат тестирования экранов авторизации.

  
Рисунок 3.2.1 – Тестирование авторизации

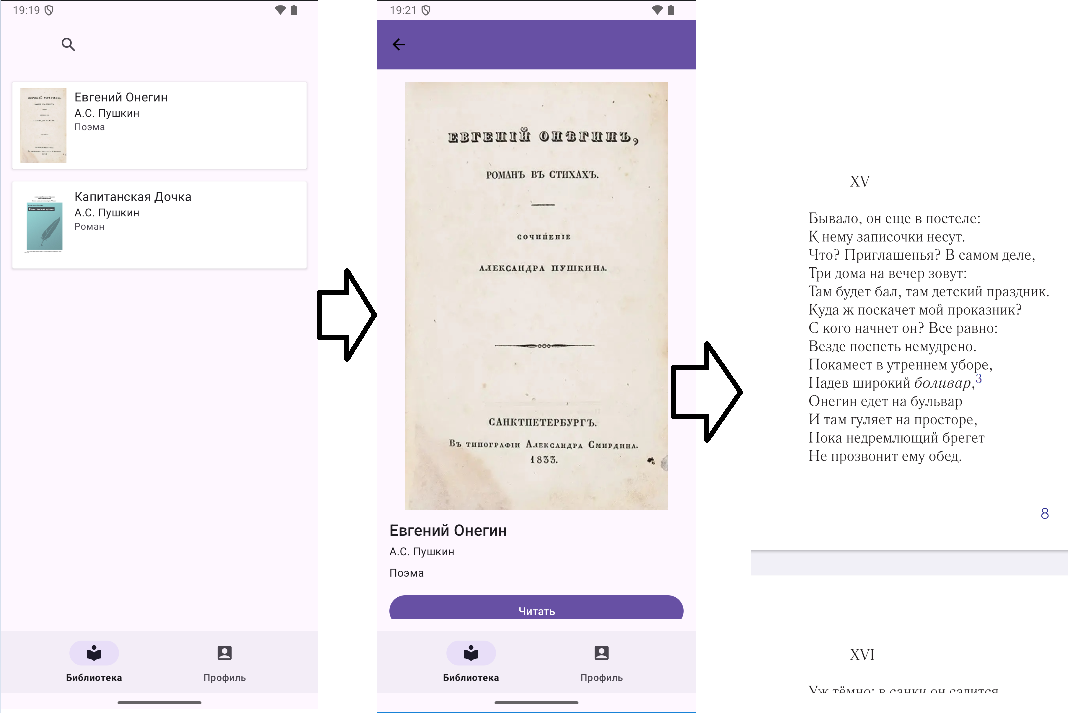
На Рисунке 3.2.2 продемонстрировано тестирование регистрации.

  
Рисунок 3.2.2 – Тестирование регистрации

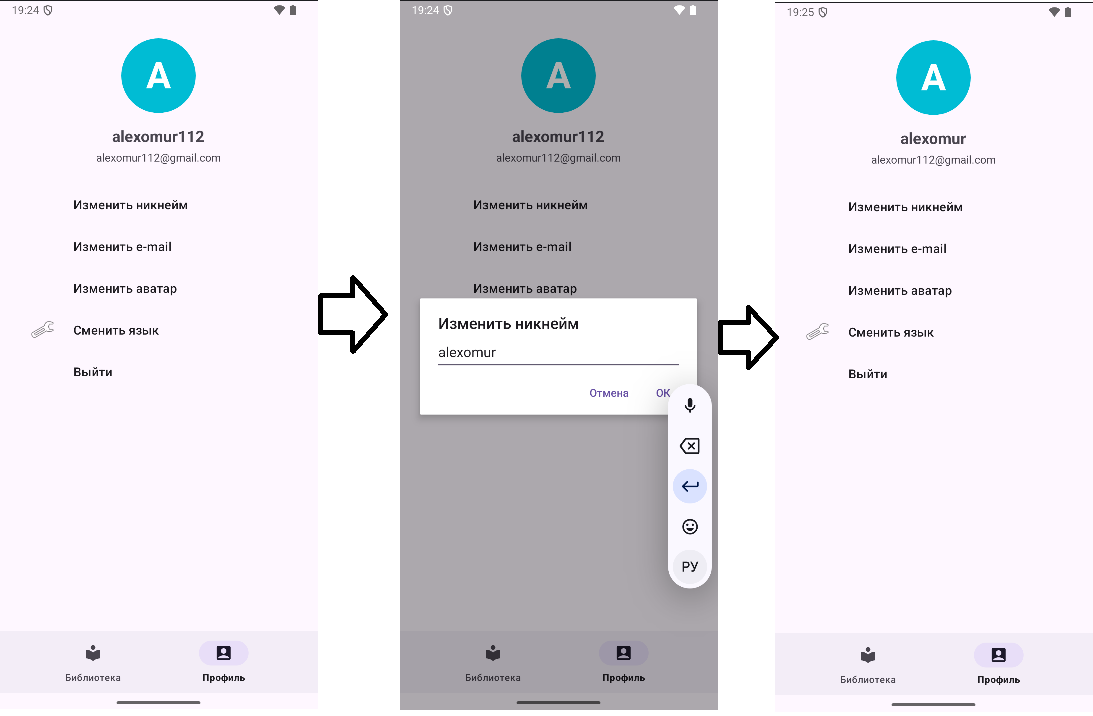
На Рисунке 3.2.3 показано тестирование каталога и поиска.

  
Рисунок 3.2.3 – Тестирование каталога и поиска

На Рисунке 3.2.4 показано тестирование экрана с детальной информацией о книге и открытия внешнего PDF-ридера.

  
Рисунок 3.2.4 – Тестирование экрана с детальной информацией о книге и открытия внешнего PDF-ридера

На Рисунке 3.2.5 продемонстрировано тестирование изменения профиля пользователя.

  
Рисунок 3.2.5 – Тестированиее изменения профиля пользователя

# ЗАКЛЮЧЕНИЕ

В рамках данной работы разработано и протестировано мобильное приложение для управления домашней библиотекой на платформе Android. Приложение предоставляет удобный и интуитивно понятный интерфейс для просмотра каталога книг, поиска по названию, изучения детальной информации о книгах и их авторах, а также возможности чтения в формате PDF.

Была реализована система регистрации и авторизации пользователей с использованием Firebase Auth, а также сохранение активных сессий с помощью SharedPreferences. Каталог книг, авторские данные и пользовательские настройки хранятся и синхронизируются через Firebase Realtime Database. Архитектура приложения построена на принципе «Single Activity — Multiple Fragments», обеспечивая модульность, масштабируемость и простоту поддержки кода.

Для реализации проекта были использованы Android Studio, язык Java, Firebase SDK, Navigation Component, Glide для эффективной загрузки изображений и UI-компоненты Material Design. Визуальный интерфейс был выполнен с использованием XML-разметки и адаптивных drawable-ресурсов.

Тестирование показало стабильную и производительную работу приложения на различных устройствах. Разработанное решение может быть использовано для дальнейших учебных целей или послужить основой для создания полнофункционального мобильного продукта для домашней библиотеки. Проект позволил улучшить навыки проектирования, разработки интерфейсов и взаимодействия с облачными сервисами.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Приложения для домашней библиотеки [Электронный ресурс]. – URL: <https://blog.mann-ivanov-ferber.ru/2020/05/27/prilozheniya-dlya-domashnej-biblioteki/>, режим доступа: свободный (дата обращения: 05.06.2025).
2. IDEF0: знакомство с нотацией и пример использования [Электронный ресурс]. – URL: <https://www.trinion.org/blog/idef0-znakomstvo-s-notaciey-i-primer-ispolzovaniya>, режим доступа: свободный (дата обращения: 05.06.2025).
3. Что такое DFD (диаграммы потоков данных) [Электронный ресурс] / Хабр – URL: <https://habr.com/ru/articles/668684/>, режим доступа: свободный (дата обращения: 05.06.2025).
4. Фрагменты [Электронный ресурс] – URL: [https://metanit.com/java/android/8.1.php](https://metanit.com/java/android/8.1.php%20), режим доступа: свободный (дата обращения 05.06.2025).
5. Основы создания интерфейса Android Studio [Электронный ресурс] – URL: [https://metanit.com/java/android/3.1.php](https://metanit.com/java/android/3.1.php%20), режим доступа: свободный (дата обращения: 05.06.2025).
6. Сохраните данные в локальной базе данных с помощью Room [Электронный ресурс] – URL: <https://developer.android.com/training/data-storage/room?hl=ru>, режим доступа: свободный (дата обращения: 04.06.2025).
7. Использование SharedPreferences для хранения данных [Электронный ресурс] – URL: <https://developer.android.com/training/data-storage/shared-preferences?hl=ru>, режим доступа: свободный (дата обращения: 05.06.2025).
8. Официальная документация Firebase Realtime Database [Электронный ресурс] – URL: <https://firebase.google.com/docs/database>, режим доступа: свободный (дата обращения: 05.06.2025).
9. Glide – загрузка изображений в Android [Электронный ресурс] – URL: [https://bumptech.github.io/glide/](https://bumptech.github.io/glide/%20), режим доступа: свободный (дата обращения: 05.06.2025).

# ПРИЛОЖЕНИЕ

Репозиторий разрабатываемого прототипа мобильного приложения находится по адресу: <https://github.com/alexomur/HomeLibrary>