**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых» (ВлГУ)

Кафедра ВТиСУ

**КУРСОВАЯ РАБОТА**

по дисциплине «Разработка мобильных приложений»

на тему «Разработка мобильных приложений на платформе Android. Часть 4. Разработка мобильного приложения «Трекер книг»»

Студент: Андрианов Е.В.

Институт: ИИТЭ

Направление: 09.03.03

Группа: ПИ-121

Руководитель: Шутов А.В.

Владимир 2024 г.

Содержание

[ВВЕДЕНИЕ 3](#_Toc186055191)

[1. Теоретическая часть 5](#_Toc186055192)

[1.1. Анализ существующих решений 5](#_Toc186055193)

[1.2. Понятие и виды мобильных приложений 7](#_Toc186055194)

[1.3. Основы архитектуры приложений на платформе android 12](#_Toc186055195)

[1.4. Основные этапы разработки мобильных приложений 16](#_Toc186055196)

[1.5. Выбор технологий и инструментов разработки 21](#_Toc186055197)

[2. Практическая часть 24](#_Toc186055198)

[2.1. Описание поставленной задачи и предметной области 24](#_Toc186055199)

[2.2. Проектирование интерфейса пользователя 26](#_Toc186055200)

[2.3. Обзор пользовательского интерфейса 28](#_Toc186055201)

[ЗАКЛЮЧЕНИЕ 31](#_Toc186055202)

[Список используемой литературы 32](#_Toc186055203)

# **ВВЕДЕНИЕ**

В последние десятилетия с развитием технологий и ростом популярности мобильных устройств, мобильные приложения стали важной частью нашей повседневной жизни. Они охватывают широкий спектр задач, от социальных сетей до финансовых операций и организации личной жизни. Одной из самых ярких тенденций является создание приложений, которые помогают пользователю более эффективно управлять своим временем, ресурсами и информацией. Это стало возможным благодаря улучшению мобильных операционных систем, росту вычислительных мощностей устройств и повсеместному распространению интернета.

Одним из важнейших факторов при разработке мобильных приложений является создание интуитивно понятного и удобного интерфейса, который обеспечивает комфортное взаимодействие с пользователем. Важно учитывать не только функциональность, но и внешний вид приложения, его доступность для широкой аудитории и простоту в использовании. Успех приложения во многом зависит от того, насколько эффективно оно решает задачу пользователя и насколько комфортным является его использование в условиях современных технологий.

Целью данной курсовой работы является разработка мобильного приложения для платформы Android, которое будет предоставлять функционал для записи своих прочитанных книг или же книг, которые в будущем пользователь хочет прочитать. Данное приложение должно обеспечить удобный пользовательский интерфейс, простоту навигации и возможность оперативного получения необходимой информации.

Для достижения поставленной цели, будут решены следующие задачи:

1. Изучение инструментов и технологий для разработки приложений на платформе Android.
2. Разработка пользовательского интерфейса с учетом требований удобства и интуитивной понятности.
3. Реализация функционала для регистрации и авторизации пользователей.
4. Реализация функционала для управления списками книг.
5. Реализация функционала редактирования и удаления книг.

При выполнении работы используются следующие программные средства:

1. **Android Studio;**
2. Языки программирования Java и PHP;
3. Программная среда Open Server;
4. СУБД MySQL;

Таким образом, курсовая работа направлена на приобретение теоретических знаний и практических навыков в области разработки мобильных приложений для Android, что соответствует требованиям образовательной программы и способствует профессиональному развитию в сфере информационных технологий.

# **1. Теоретическая часть**

## **1.1. Анализ существующих решений**

На рынке мобильных приложений для трекинга книг существует множество решений, таких как Goodreads, Bookly, StoryGraph, Libib и другие. Эти приложения предлагают пользователям различные функции, которые помогают эффективно управлять информацией о своих прочитанных книгах. Однако каждое из этих приложений имеет свои особенности и ограничения, которые могут повлиять на выбор пользователя.

Goodreads - одна из крупнейших платформ для читателей, позволяющая отслеживать книги, составлять списки, оценивать книги, а также читать рецензии и делиться рекомендациями с друзьями. Это социальная сеть для любителей книг, где можно обмениваться мнениями, участвовать в обсуждениях и искать новые книги.

Недостатки:

* Перегруженность интерфейса

Пользователи могут быть перегружены многочисленными функциями и информацией на экране. Множество разделов и фильтров иногда делают навигацию сложной для новичков.

* Рекламные материалы и коммерция

Часто на платформе появляются платные рекомендации, что может затруднить нахождение независимых рецензий.

Bookly - приложение, предназначенное для отслеживания времени, проведенного за чтением. Оно ориентировано на людей, которые хотят фиксировать прогресс своего чтения, а также улучшать привычки, связанные с чтением книг.

Недостатки:

* Ограниченные функции для социальных взаимодействий Приложение в основном ориентировано на отслеживание прогресса в чтении и не имеет таких широких возможностей для социальных функций, как Goodreads.
* Платный доступ к дополнительным функциям

Для некоторых продвинутых функций (например, создания отчетов или синхронизации данных) требуется подписка.

StoryGraph - является менее известной альтернативой Goodreads, фокусируясь на улучшенной аналитике и статистике. Это приложение позволяет пользователям отслеживать свои книжные предпочтения и видеть подробную информацию о своих чтениях.

Недостатки:

* Меньше пользователей, чем у Goodreads

Из-за относительно меньшей популярности базы данных книг может быть не такой полной, как у Goodreads.

* Ограниченные социальные функции

Нет широких возможностей для общения с другими пользователями или создания групп.

Libib - приложение, предназначенное для создания каталогов книг. Оно ориентировано на тех пользователей, которые хотят детально управлять своей коллекцией книг и создавать персонализированные библиотеки.

Недостатки:

* Сложный интерфейс для новичков:

Приложение имеет довольно сложный интерфейс, который может не подойти тем пользователям, которые ищут простоту.

* Платный доступ к дополнительным функциям

Многие расширенные функции доступны только по подписке, что делает приложение менее доступным для всех пользователей.

Изучив существующие решения, можно выделить несколько ключевых тенденций и проблем, с которыми сталкиваются пользователи:

1. Большинство приложений перегружены множеством функций, которые могут не быть необходимыми для простого отслеживания книг.
2. Приложения вроде Goodreads и StoryGraph предлагают хорошую аналитику и социальные функции, но могут быть сложными для новичков.
3. Приложения, такие как Bookly и Libib, предлагают удобство для отслеживания времени и управления коллекциями, но ограничены в других аспектах, таких как социальное взаимодействие.

Из проведённого анализа можно выделить несколько ключевых требований, которые необходимо учитывать при разработке мобильного приложения для управления расписанием:

1. Интуитивно понятный интерфейс

Приложение должно быть простым в использовании, с минималистичным интерфейсом, где пользователи могут легко перейти от экрана к экрану (регистрация, авторизация, добавление книги и т.д.).

2. Функциональность управления книгами

Приложение должно поддерживать функции для добавления, редактирования и удаления книг в списке "Прочитанные" и "Ожидают прочтения". Пользователь должен иметь возможность управлять своими книгами и изменять их состояние.

3. Адаптивность под размер экрана и операционные системы

Приложение должно работать корректно на разных устройствах и операционных системах, с адаптивным интерфейсом.

## **1.2. Понятие и виды мобильных приложений**

Мобильные приложения — это программы, созданные для работы на мобильных устройствах, таких как смартфоны, планшеты и другие устройства с мобильными операционными системами. Эти приложения предоставляют пользователю широкий спектр функций, которые могут быть направлены на улучшение производительности, развлечение, организацию досуга, обучение и многие другие аспекты жизни. В отличие от веб-приложений, которые работают через браузер, мобильные приложения устанавливаются непосредственно на устройства и предоставляют более быстрый и гибкий доступ к различным функциям.

Мобильные устройства, такие как смартфоны и планшеты, становятся неотъемлемой частью повседневной жизни, и с каждым годом число пользователей мобильных приложений растет. В последние годы мобильные технологии значительно развились, что открыло новые горизонты для создания приложений, которые могут не только облегчить повседневные задачи, но и существенно повысить качество жизни. Мобильные приложения могут быть предназначены для различных сфер, начиная от бизнеса и социальных сетей и заканчивая обучением и здоровьем. Таким образом, мобильные приложения становятся важным инструментом в жизни как индивидуальных пользователей, так и организаций.

В современном мире мобильные приложения охватывают огромное количество сфер жизни и работают на различных операционных системах, таких как Android, iOS и других. Все это приводит к тому, что мобильные приложения становятся важным элементом экосистемы технологий и одним из главных инструментов взаимодействия с пользователем.

Виды мобильных приложений:

Мобильные приложения можно классифицировать по нескольким признакам, включая их функциональные особенности, технологии разработки и целевое назначение. Рассмотрим наиболее популярные категории мобильных приложений.

**1. Нативные приложения**

Нативные приложения — это приложения, которые разрабатываются специально для одной мобильной операционной системы, такой как Android или iOS. Они пишутся на языках программирования, поддерживаемых конкретной операционной системой, например, Java или Kotlin для Android, Swift или Objective-C для iOS. Такие приложения устанавливаются непосредственно на устройство и имеют доступ ко всем ресурсам и возможностям устройства, таким как камера, GPS, микрофон, сенсоры и другие.

Основным преимуществом нативных приложений является высокая производительность, поскольку они оптимизированы для работы с конкретной операционной системой. Также нативные приложения предоставляют пользователю максимальный уровень взаимодействия с устройством, а также могут работать в оффлайн-режиме, что делает их удобными для использования в любых условиях. Недостатком является необходимость разработки отдельной версии приложения для каждой операционной системы, что увеличивает затраты на создание и поддержку.

**2. Кроссплатформенные приложения**

Кроссплатформенные приложения — это приложения, которые разрабатываются с использованием технологий, позволяющих создавать один код для разных операционных систем. Такие приложения разрабатываются с помощью инструментов, которые создают универсальный код, который затем компилируется и адаптируется для Android, iOS и других платформ. Популярными фреймворками для разработки кроссплатформенных приложений являются React Native, Flutter, Xamarin.

Преимущество кроссплатформенных приложений заключается в том, что разработчики могут создавать одну версию приложения для нескольких операционных систем, что снижает затраты на разработку и тестирование. Однако кроссплатформенные приложения часто уступают нативным по производительности и функциональности, поскольку их код должен быть адаптирован для работы на различных устройствах.

**3. Веб-приложения**

Веб-приложения — это приложения, которые работают через мобильный браузер и не требуют установки на устройство. Такие приложения используют веб-технологии, такие как HTML, CSS и JavaScript, и могут работать на разных операционных системах без необходимости разрабатывать отдельные версии для каждой из них.

Основное преимущество веб-приложений заключается в том, что они не требуют установки и занимают мало места на устройстве. Кроме того, веб-приложения автоматически обновляются, что исключает необходимость пользователю вручную загружать новые версии. Однако веб-приложения ограничены функциональностью, предоставляемой браузером, и не имеют такого уровня доступа к системным ресурсам устройства, как нативные приложения. Кроме того, они зависят от стабильности интернет-соединения, что ограничивает их использование в оффлайн-режиме.

**4. Гибридные приложения**

Гибридные приложения — это промежуточный вариант между нативными и веб-приложениями. Они разрабатываются с использованием веб-технологий (HTML, CSS, JavaScript), но оборачиваются в нативный контейнер, который позволяет запускать приложение как обычное нативное. Такие приложения могут работать на разных операционных системах, при этом они сохраняют доступ к функционалу устройства через нативные элементы.

Гибридные приложения предоставляют определенные преимущества в плане разработки, так как позволяют создавать одну версию приложения для разных платформ, однако они уступают нативным приложениям по производительности и функциональности. Основное их преимущество заключается в том, что они обеспечивают доступ к большинству функций устройства при меньших затратах на разработку.

**5. Приложения для интернета вещей (IoT)**

С развитием технологий интернета вещей (IoT) появились приложения, которые управляют и взаимодействуют с различными устройствами через мобильное приложение. Такие приложения позволяют пользователю управлять умными домами, системами безопасности, носимыми устройствами и другими гаджетами. Эти приложения обычно имеют интерфейсы для управления устройствами, а также для мониторинга их состояния.

Основным требованием к приложениям для интернета вещей является высокая надежность связи между устройствами, а также обеспечение безопасности данных, передаваемых между ними. Такие приложения часто используют Bluetooth, Wi-Fi или другие протоколы связи для обмена данными между устройствами.

**6. Игровые приложения**

Игровые мобильные приложения занимают важное место в мобильной индустрии. Они предоставляют пользователю развлечение и взаимодействие в виртуальной среде, что является основным назначением игр. Игровые приложения могут варьироваться от простых аркад до сложных многопользовательских онлайн-игр с различными уровнями графики и взаимодействия.

Игровые приложения часто используют продвинутую графику, анимацию и звуковое оформление, чтобы обеспечить более насыщенный и интересный опыт. Одним из главных требований к игровым приложениям является высокая производительность, стабильная работа и минимизация времени отклика, чтобы обеспечить пользователю увлекательный игровой процесс.

## **1.3. Основы архитектуры приложений на платформе android**

Android — это одна из самых популярных операционных систем для мобильных устройств, разработанная компанией Google. Она основана на ядре Linux и предоставляет широкие возможности для создания мобильных приложений с использованием различных технологий. Архитектура приложений на платформе Android представляет собой структуру, которая описывает взаимодействие компонентов приложения, их жизненный цикл, а также способы их организации и взаимодействия с пользователем и системой. Для успешной разработки приложений важно понимать основы архитектуры Android, а также подходы, которые используются для упрощения и улучшения разработки.

Основные компоненты приложения Android:

Каждое приложение на платформе Android состоит из нескольких ключевых компонентов, которые обеспечивают его работу и взаимодействие с пользователем и операционной системой. На платформе Android выделяют четыре основные компоненты:

1. Activity (Активность)

Activity является основным компонентом для создания пользовательского интерфейса в Android. Это класс, который представляет собой одно окно приложения, с помощью которого взаимодействует пользователь. Каждое приложение может содержать несколько активностей, каждая из которых может отвечать за отображение определенной информации или за выполнение определенной задачи (например, экран с настройками или экран с деталями продукта).

Основной задачей активности является создание и управление пользовательским интерфейсом, обработка пользовательских действий и взаимодействие с другими компонентами приложения. Каждая активность имеет свой жизненный цикл, который управляется системой и может изменяться в зависимости от состояния приложения (например, переход из фона в активное состояние или его закрытие).

1. Service (Служба)

Служба — это компонент приложения, который выполняет долгосрочные операции в фоновом режиме. В отличие от активности, служба не взаимодействует с пользователем напрямую, но может продолжать работать даже когда приложение не активно или когда экран устройства выключен. Например, служба может использоваться для воспроизведения музыки, обработки сетевых запросов или выполнения других операций, которые должны продолжаться в фоновом режиме.

Службы также имеют свой жизненный цикл и могут быть запущены или остановлены системой по требованию. Существуют два типа служб: Foreground Service — служба, которая выполняется в переднем плане и требует постоянного внимания пользователя, и Background Service — служба, которая выполняется в фоновом режиме и не требует взаимодействия с пользователем.

1. Broadcast Receiver (Приемник широковещательных сообщений)

Broadcast Receiver — это компонент, который позволяет приложениям слушать и реагировать на широковещательные сообщения или события, происходящие в системе или других приложениях. Эти события могут быть как системными, так и пользовательскими. Например, приемник широковещательных сообщений может отслеживать изменения состояния сети, батареи, или получать уведомления о входящих сообщениях и других системных событиях.

Приемники могут быть активированы в ответ на конкретные события или же работать в фоновом режиме, обрабатывая события, происходящие в системе. Важной особенностью является то, что они не создают пользовательский интерфейс, а выполняют работу по обработке событий.

1. Content Provider (Поставщик контента)

Content Provider — это компонент, который управляет доступом к данным, предоставляя интерфейс для работы с ними для других приложений. Поставщик контента позволяет одному приложению передавать данные другому, защищая внутренние данные приложения от прямого доступа.

Примером использования поставщика контента может быть доступ к данным контактов на устройстве, истории звонков или галерее изображений. Content Provider может использоваться для того, чтобы абстрагировать работу с данными и предоставить унифицированный способ доступа к ним.

Жизненный цикл компонентов:

Каждый компонент в Android имеет свой жизненный цикл, который управляется операционной системой. Важно понимать, как система взаимодействует с каждым компонентом, чтобы эффективно управлять его состоянием, энергопотреблением и ресурсами устройства.

Активности: Жизненный цикл активности включает несколько стадий: создание, начало, приостановка, возобновление и уничтожение. Управление этими стадиями позволяет разработчику правильно обрабатывать пользовательский ввод, взаимодействовать с фоновыми процессами и управлять ресурсами.

Службы: Жизненный цикл службы начинается с ее запуска и продолжается до тех пор, пока служба не будет остановлена либо системой, либо самим приложением. Службы могут работать в фоне, что важно для создания долгосрочных задач, таких как синхронизация данных или фоновые вычисления.

Приемники широковещательных сообщений: Эти компоненты не имеют стандартного жизненного цикла, так как они активируются в ответ на события. Они начинают работать, когда система или приложение посылает широковещательное сообщение, и завершают работу после обработки сообщения.

Поставщики контента: Поставщики контента работают с данными и предоставляют интерфейс для их получения. Обычно поставщики контента создаются для работы с базами данных или файловыми системами, и жизненный цикл этих компонентов зависит от того, как осуществляется доступ к данным.

Архитектурные подходы в Android-разработке:

Для упрощения разработки и улучшения качества кода, в Android-разработке существует несколько архитектурных подходов, которые помогают организовать структуру приложения.

Основными подходами являются:

1. Model-View-Controller (MVC)

MVC — это классическая архитектурная модель, в которой приложение разделяется на три компонента: модель (Model), представление (View) и контроллер (Controller). Модель отвечает за хранение и управление данными, представление отображает данные пользователю, а контроллер обрабатывает взаимодействие с пользователем и управляет логикой приложения. Однако, несмотря на свою популярность, MVC в Android-разработке может создавать проблемы с управлением состоянием и трудностью поддержания кода.

1. Model-View-Presenter (MVP)

MVP — это более развитая архитектурная модель, которая решает проблемы MVC, улучшая разделение логики и UI. В этой модели представление (View) представляет собой интерфейс пользователя, модель (Model) управляет данными, а презентер (Presenter) управляет логикой приложения и отвечает за взаимодействие между представлением и моделью.

1. Model-View-ViewModel (MVVM)

MVVM — это современный подход, который широко используется в Android-разработке. В этой модели представление (View) отвечает за отображение UI, модель (Model) управляет данными, а ViewModel представляет собой слой, который связывает модель и представление, предоставляя данные в удобном формате для отображения. MVVM поддерживает использование LiveData и DataBinding для автоматического обновления UI при изменении данных.

1. Clean Architecture

Clean Architecture — это архитектурный подход, который стремится создать четкую иерархию слоев в приложении, разделяя код на независимые части. В этой модели данные, логика и представление разделены на отдельные слои, что улучшает тестируемость и поддержку приложения. Clean Architecture основывается на принципах SOLID и помогает разработчикам создавать масштабируемые и поддерживаемые приложения.

## **1.4. Основные этапы разработки мобильных приложений**

Разработка мобильных приложений — это многозадачный процесс, который включает несколько ключевых этапов. Каждый этап требует внимания к деталям, планирования и четкой организации для того, чтобы приложение было успешным, удовлетворяло потребности пользователей и работало стабильно. Процесс разработки можно разделить на несколько последовательных этапов, каждый из которых играет важную роль в создании качественного продукта.

1. Идея и планирование

Первый этап разработки мобильного приложения начинается с идеи. Важно понять, какую проблему будет решать приложение, кто является целевой аудиторией, какие функции и возможности будут в нем реализованы. На этом этапе необходимо провести анализ рынка и конкурентов, чтобы выявить потребности пользователей и понять, чем приложение будет отличаться от существующих решений.

Планирование также включает определение функционала, который должен быть включен в приложение. Это может быть, например, возможность регистрации пользователей, наличие профилей, система уведомлений, геолокация или интеграция с другими сервисами. Важно на этом этапе четко обозначить цели и задачи, которые приложение должно выполнить.

Кроме того, на этом этапе нужно составить проектный план, определив основные этапы работы, сроки, ресурсы и бюджет, если речь идет о коммерческом проекте. Планирование и разработка стратегии поможет команде сосредоточиться на ключевых аспектах, таких как приоритеты функционала и возможность тестирования на различных устройствах.

2. Разработка концепции и дизайна

После того как основные идеи и цели были определены, следующим шагом является разработка концепции и дизайна приложения. Это ключевой этап, на котором закладывается пользовательский интерфейс (UI) и пользовательский опыт (UX).

Процесс дизайна начинается с создания макетов или прототипов экрана. Прототипирование помогает визуализировать, как приложение будет выглядеть, какие элементы интерфейса будут размещены на каждом экране, как пользователи будут взаимодействовать с приложением. Прототипы могут быть выполнены с использованием таких инструментов, как Figma, Adobe XD или Sketch, которые позволяют создавать интерактивные прототипы с возможностью тестирования.

Дизайн приложения должен учитывать требования целевой аудитории, соответствовать общим тенденциям и следовать принципам удобства использования. Важно, чтобы интерфейс был интуитивно понятным, а взаимодействие с ним — логичным и удобным. Это включает в себя выбор шрифтов, цветовых схем, иконок и размещение кнопок. Пользовательский опыт должен быть ориентирован на максимально простое и быстрое выполнение задач.

На этом этапе также создаются визуальные материалы, такие как иконки, логотипы и другие графические элементы, которые будут использоваться в приложении. Каждый из этих элементов должен быть проработан таким образом, чтобы приложение выглядело привлекательно и профессионально.

3. Разработка и программирование

После завершения этапа проектирования и утверждения макетов начинается этап разработки и программирования. На этом этапе разрабатываются все основные функциональные части приложения: создание базы данных, интеграция с внешними сервисами, написание логики работы приложения и реализация интерфейса.

Сначала разрабатывается архитектура приложения и выбирается подходящая платформа — Android или iOS, либо обе. Программирование может вести как один разработчик, так и группа специалистов, в зависимости от сложности проекта. Важно, чтобы приложение было оптимизировано под каждую платформу, учитывая особенности операционных систем, их интерфейс и технические ограничения.

Разработка включает в себя несколько этапов:

* **Фронтенд-разработка (клиентская часть):** на этом этапе создаются экраны приложения, все элементы пользовательского интерфейса, а также взаимодействие с пользователем через элементы управления (кнопки, поля ввода и т.д.).
* **Бэкенд-разработка (серверная часть):** на этом этапе создается серверная логика, базы данных, API для взаимодействия с сервером и другие элементы, которые обеспечивают работу приложения на серверной стороне. Это может включать создание базы данных для хранения пользовательских данных, а также интеграцию с внешними сервисами, такими как платежные системы, карты, социальные сети и т.д.

Процесс разработки требует активного тестирования и отладки кода для выявления ошибок, и их устранения. Это важный этап, который помогает убедиться, что приложение работает корректно и не вызывает сбоев на разных устройствах.

4. Тестирование и отладка

После того как приложение собрано и основные функции реализованы, необходимо провести тестирование. Тестирование — это один из самых важных этапов разработки, который позволяет убедиться в работоспособности приложения, выявить ошибки и недочеты, а также улучшить его функциональность.

Тестирование можно разделить на несколько видов:

**Функциональное тестирование** – проверка правильности работы всех функций приложения. Это включает в себя тестирование регистрации, авторизации, работы всех кнопок, форм и других элементов интерфейса.

**Тестирование производительности** – проверка скорости работы приложения, его отклика и стабильности при высокой нагрузке. Это включает в себя тесты на скорость запуска, загрузку данных и работу с большими объемами информации.

**Тестирование совместимости** – проверка работы приложения на разных устройствах с разными версиями операционных систем. Это важно, так как Android и iOS имеют различные устройства, размеры экранов и аппаратные характеристики, что может повлиять на работу приложения.

**UI/UX тестирование** – проверка удобства использования интерфейса. Это позволяет понять, насколько интуитивно понятным и удобным является пользовательский интерфейс, соответствует ли он ожиданиям пользователей.

Ошибки, выявленные в ходе тестирования, исправляются на этапе отладки. Также на этом этапе часто проводится бета-тестирование, которое включает проверку приложения на ограниченной группе пользователей для выявления возможных недочетов, которые могут быть не заметны разработчикам.

5. Развертывание и публикация

После того как приложение прошло тестирование и все ошибки были исправлены, оно готово к развертыванию и публикации. На этом этапе приложение загружается в соответствующий магазин приложений, например, Google Play для Android или App Store для iOS.

Для публикации в Google Play или App Store необходимо создать учетную запись разработчика, подготовить все необходимые материалы, такие как описание приложения, скриншоты, иконки и видеообзор. После этого приложение проходит проверку модераторами магазина, что может занять некоторое время. В случае с iOS проверка может быть более строгой, и приложение может быть отклонено, если оно не соответствует стандартам Apple.

6. Поддержка и обновления

После того как приложение опубликовано и доступно для пользователей, процесс разработки не заканчивается. Важно следить за его работой, собирать обратную связь от пользователей и устранять возможные проблемы. Это включает в себя исправление багов, улучшение производительности и добавление нового функционала в ответ на запросы пользователей.

Регулярные обновления помогают поддерживать интерес к приложению и обеспечивать его соответствие современным требованиям. Важно постоянно улучшать приложение, чтобы оно оставалось актуальным и конкурентоспособным.

## **1.5. Выбор технологий и инструментов разработки**

Для реализации данного проекта было принято решение использовать проверенные и высокоэффективные технологии, которые обеспечат стабильную и бесперебойную работу мобильного приложения, соответствующую современным стандартам разработки. Выбор этих технологий основан на их популярности, доступности, широком функционале и высокой совместимости, что позволяет создать качественное и удобное приложение для управления расписанием занятий.

1. **Android Studio**: это интегрированная среда разработки (IDE), специально предназначенная для создания приложений на платформе Android. Android Studio предоставляет все необходимые инструменты для разработки, включая мощные средства визуального проектирования интерфейса, анализатор кода, встроенные инструменты для отладки и тестирования, а также поддержку эмуляторов устройств. Вся эта функциональность значительно ускоряет процесс разработки и позволяет сосредоточиться на создании высококачественного приложения, удовлетворяющего требованиям пользователей.
2. **Java**: Один из самых популярных и надёжных языков программирования для разработки мобильных приложений на платформе Android. Java обеспечивает отличную стабильность и совместимость с Android SDK, что делает её идеальным выбором для создания мобильных решений. Этот язык обладает богатой экосистемой библиотек и фреймворков, что ускоряет процесс разработки и улучшает производительность приложений. Он также упрощает решение таких задач, как обработка данных, управление ресурсами устройства и взаимодействие с базами данных.
3. **Volley**: Библиотека для выполнения HTTP-запросов и взаимодействия с сервером, которая существенно упрощает работу с сетевыми запросами. Volley позволяет эффективно взаимодействовать с REST API, предоставляя встроенные механизмы кэширования и обработки запросов. Эта библиотека помогает быстро отправлять и получать данные с сервера, минимизируя задержки, что особенно важно для приложений с динамическими данными, такими как расписания занятий.
4. **PHP**: Популярный серверный язык программирования, широко используемый для создания API и обработки серверных запросов. PHP отличается гибкостью, простотой в использовании и возможностью интеграции с различными базами данных. В рамках данного проекта PHP применяется для разработки серверной части приложения, обеспечивая взаимодействие с базой данных и обработку данных, поступающих от клиента. Этот язык идеально подходит для обработки запросов, что гарантирует эффективное управление данными в реальном времени.
5. **MySQL**: Реляционная база данных, подходящая для хранения и управления информацией о пользователях и расписаниях. MySQL отличается высокой производительностью, масштабируемостью и надёжностью, что делает её отличным выбором для работы с большими объёмами данных. Использование индексов и связей между таблицами позволяет оптимизировать запросы и минимизировать вероятность ошибок, что критично для приложений с большим количеством пользователей и постоянно обновляемой информацией, как в случае с расписаниями занятий.

Выбор этих технологий был обусловлен их совместимостью и возможностью интеграции друг с другом, что позволяет создать устойчивую и масштабируемую архитектуру приложения. В совокупности эти инструменты обеспечивают надёжную платформу для разработки функционального и удобного приложения, способного эффективно управлять расписанием занятий.

# **2. Практическая часть**

## **2.1. Описание поставленной цели и предметной области**

В рамках курсовой работы была поставлена цель разработки мобильного приложения для отслеживания прочитанных книг и книг, которые хотелось бы прочитать. Приложение должно быть удобным и интуитивно понятным, обеспечивать возможность регистрации пользователей, авторизации, добавления, редактирования и удаления информации о книгах, а также отображать список прочитанных и запланированных для прочтения книг.

Система должна позволять пользователю управлять своими данными, сохраняя их в базе данных, а также предоставлять простой и эффективный интерфейс для работы с книгами.

Приложение должно предоставлять пользователю следующие основные функции:

1. **Регистрация и авторизация пользователей**: Приложение должно поддерживать создание новых учетных записей и возможность входа в систему с уже существующей учетной записью. Для этого будет реализован экран регистрации с полями для ввода логина и пароля, а также экран авторизации для ввода логина и пароля для входа в систему.
2. **Управление списками книг**: после успешной авторизации пользователь должен получить доступ к двум основным разделам — "Прочитанные книги" и "Ожидающие прочтения". В каждом из этих разделов будет отображаться список книг, а также предоставляться возможность добавления новых книг в каждый список. Для этого будет реализована форма для ввода данных о книге, включая название, автора и описание.
3. **Редактирование и удаление книг**: Пользователь должен иметь возможность изменять или удалять книги, которые уже добавлены в список прочитанных или ожидающих прочтения. Для этого будет предусмотрен интерфейс для редактирования данных книги и кнопка для её удаления.
4. **Хранение данных**: Все добавленные пользователем книги должны сохраняться в базе данных, что обеспечит возможность их просмотра и редактирования в будущем. Важно также, чтобы данные сохранялись даже после закрытия приложения, что требует реализации функционала работы с локальной базой данных.
5. **Удобный и понятный интерфейс**: Интерфейс приложения должен быть простым и интуитивно понятным, чтобы пользователи могли быстро и легко выполнять все операции. Для этого будет использована стандартная структура интерфейса Android, включающая кнопки, текстовые поля и списки.

Основной задачей в рамках разработки является создание приложения, которое будет удобным и функциональным инструментом для управления личной библиотекой, а также обеспечит безопасность данных пользователей.

Предметная область данного приложения связана с управлением списками книг, что является важной частью повседневной жизни для многих людей. С каждым годом всё больше людей используют различные способы для отслеживания прочитанных и планируемых к прочтению книг. Одним из популярных методов является использование мобильных приложений, которые позволяют не только вести учет книг, но и рекомендовать новые, отслеживать прогресс чтения и взаимодействовать с другими пользователями.

В данном контексте приложение направлено на создание удобного инструмента для людей, которые хотят организовать свои книжные коллекции. Оно будет полезно как для любителей чтения, так и для тех, кто следит за своими книжными планами, отслеживает перечень прочитанных произведений и желает упорядочить свою коллекцию. Важным аспектом является то, что приложение будет доступно на мобильных устройствах, что позволит пользователю всегда иметь доступ к своим спискам книг и быстро обновлять их.

Кроме того, предметная область связана с понятием управления личными данными, где важнейшими требованиями являются безопасность, хранение данных и доступность информации на разных устройствах. В этом проекте будет использована базовая работа с локальной базой данных для хранения информации о книгах и пользователях, что гарантирует сохранность данных и возможность их редактирования в любой момент времени.

Таким образом, предметная область охватывает несколько ключевых аспектов: управление книгами, безопасность данных и удобство использования мобильных приложений для организации личной информации. Приложение, разрабатываемое в рамках курсовой работы, будет отвечать всем этим требованиям и предоставлять пользователю необходимые инструменты для эффективного управления личной книжной коллекцией.

## **2.2. Проектирование интерфейса пользователя**

Для приложения, отслеживающего книги, интерфейс должен быть логично структурирован и разделен на несколько основных экранов.

Рассмотрим, как будет организован интерфейс этого приложения:

1. На главном экране будет два основных элемента управления: кнопки "Регистрация" и "Авторизация". Этот экран будет служить точкой входа в приложение, обеспечивая доступ к функционалу для новых пользователей и для тех, кто уже зарегистрирован. Кнопки будут расположены по центру экрана, чтобы привлечь внимание пользователя. (рис.2)
2. **После** нажатия кнопки "Регистрация" пользователю будет предложено заполнить поля для ввода логина, пароля и повторения пароля. Экран регистрации будет минималистичным, чтобы сосредоточить внимание на необходимых действиях. Под полями ввода будет расположена кнопка "Зарегистрироваться", которая инициирует процесс регистрации. (рис.1)
3. Авторизация – это экран, который будет доступен после нажатия кнопки "Авторизация". Он будет включать два поля для ввода — логин и пароль, а также кнопку "Авторизоваться", которая позволит пользователю войти в систему. (рис.1)

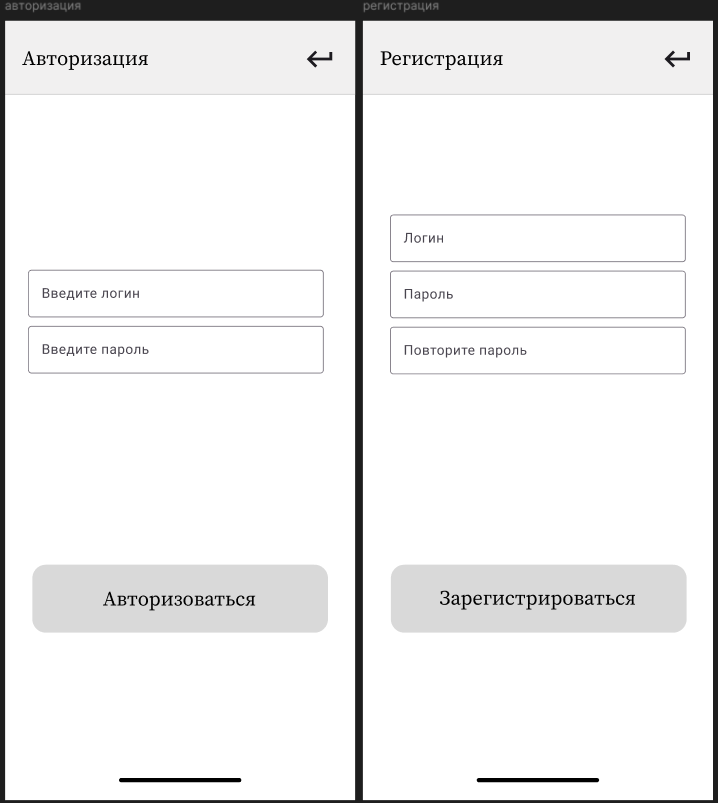


Рисунок 1 – Экраны авторизации и регистрации

1. **Экран “Выбор раздела”** после успешной авторизации, куда пользователь попадает после авторизации, имеет две кнопки: "Прочитанные книги" и "Ожидают прочтения". Эти кнопки нужны, чтобы пользователь без труда мог выбрать нужную категорию книг. Каждый раздел будет иметь собственный экран со списком книг, которые относятся к данной категории. (рис.2)



Рисунок 2 – Главный экран и экран выбора раздела

1. Экран **«Список прочитанных книг»** отображает все книги, которые были прочитаны пользователем, будет содержать список с короткими аннотациями и кнопкой для удаления каждой книги. В правом верхнем углу экрана будет кнопка "+" для добавления новой книги в список прочитанных. (рис.3), (рис.4)
2. **Экран «Список ожидающих прочтения книг»** будет аналогичен предыдущему, но предназначен для книг, которые пользователь планирует прочитать в будущем. Здесь также будет отображаться список книг с возможностью их редактирования или удаления, а также кнопка "+" для добавления новой книги в этот список. (рис.3), (рис.4)

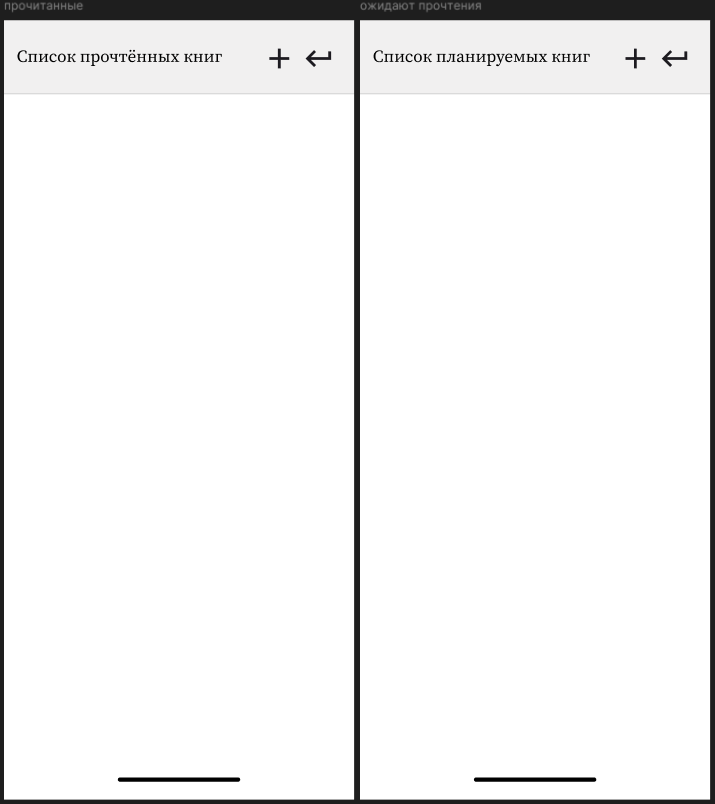


Рисунок 3 – Экраны прочитанные и ожидают прочтения

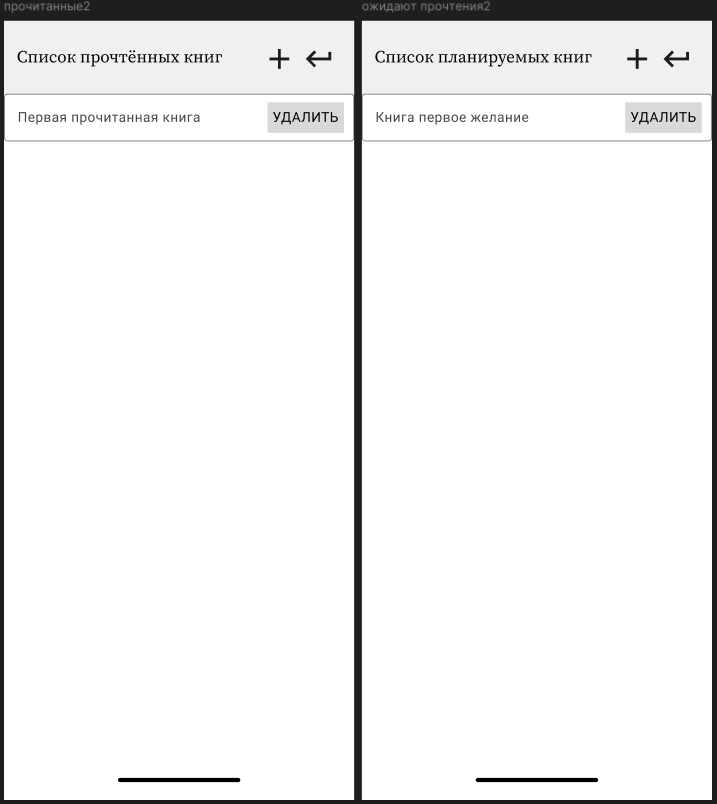


Рисунок 4 – Экраны прочитанные и ожидают прочтения с добавлением

1. **Экран добавления книги** открывается при нажатии на кнопку "+" и будет содержать поля для ввода данных о книге:

1. название, автор, описание

2. название, автор.

Также будет кнопка "Добавить", которая позволит сохранить книгу в соответствующий список (прочитанных или ожидающих прочтения). Важно, чтобы на этом экране пользователь имел возможность легко заполнить все необходимые поля и добавить книгу в список. (рис.5)

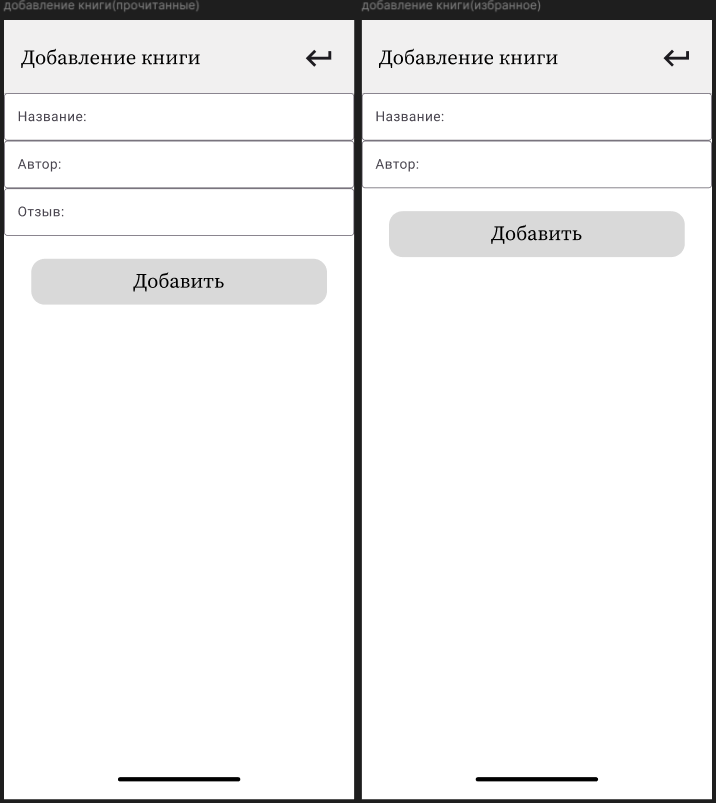


Рисунок 5 – Добавление прочитанных и желаемых книг

1. **Навигация и управление приложением** – все экраны приложения будут иметь кнопку "Назад" для возврата на предыдущий экран. Это позволит пользователю легко перемещаться по приложению без потери данных и без необходимости повторно входить в систему или добавлять книги.

Для создания прототипа интерфейса будет использована среда Figma, которая предоставляет все необходимые инструменты для визуального проектирования экранов и элементов интерфейса. Важно, чтобы элементы были распределены равномерно и гармонично, а также использовались стандартные элементы интерфейса Android, такие как кнопки, поля ввода и списки, что обеспечит консистентность приложения и улучшит его восприятие пользователями.

Цветовая гамма будет выбранной с учетом удобства восприятия и сочетаемости. Кнопки и текстовые поля будут контрастировать с фоном, чтобы обеспечить четкость и удобство взаимодействия.

## **2.3. Обзор пользовательского интерфейса**

Интерфейс был спроектирован с учетом удобства и интуитивной понятности, чтобы обеспечить легкость в навигации и выполнение всех необходимых действий. Ниже приведено описание основных экранов приложения с привязкой к скриншотам, которые демонстрируют внешний вид каждого из них.

1. **Главный экран**

На главном экране приложения расположены две ключевые кнопки: "Зарегистрироваться" и "Авторизация", которые являются основными точками входа в приложение. Этот экран является первым, который видит пользователь при запуске приложения, и его цель — предоставить простой и понятный интерфейс для выбора дальнейшего действия. (рис. 6)

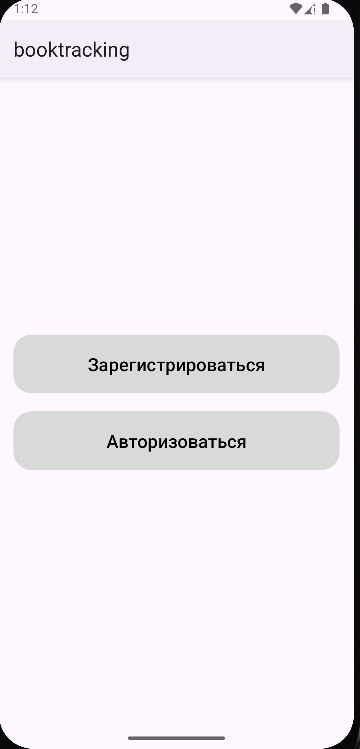


Рисунок 6 – Главный экран с кнопками регистрации и авторизации.

1. **Экран регистрации**

После нажатия на кнопку "Зарегистрироваться" пользователь переходит на экран с полями для ввода логина, пароля и повторения пароля. Важно, что все поля расположены компактно и логично, что позволяет пользователю быстро заполнить данные и перейти к следующему шагу. (рис. 7)

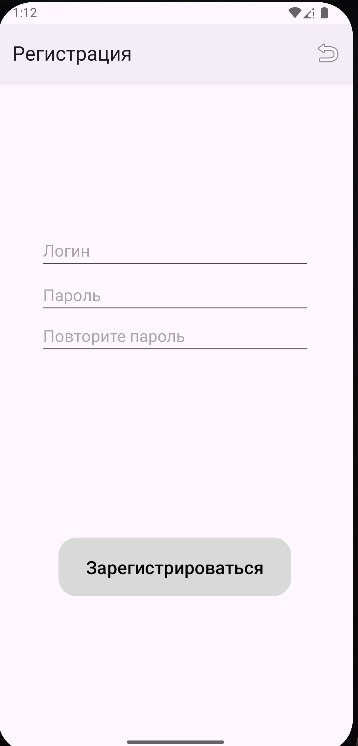


Рисунок 7 – Экран регистрации с полями для ввода данных.

1. **Экран авторизации**

Экран авторизации позволяет пользователю войти в свою учетную запись, введя логин и пароль. Здесь также находится кнопка для входа в систему. Интерфейс продуман так, чтобы минимизировать количество шагов, необходимых для авторизации, и сделать процесс входа максимально быстрым и удобным. (рис. 8)

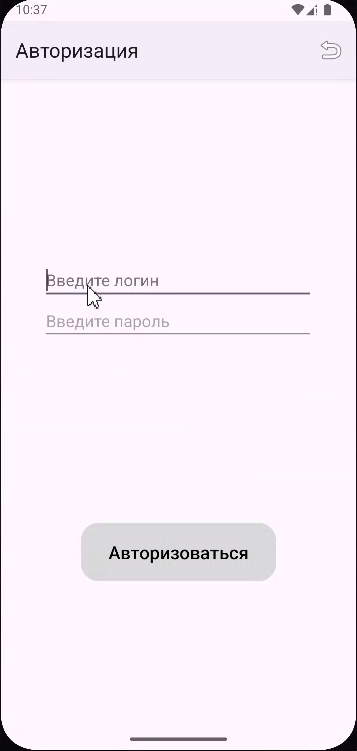


Рисунок 8 – Экран авторизации с полями для логина и пароля.

1. **Экран после авторизации**

После успешной авторизации пользователь попадает на основной экран, где ему предоставляется выбор между двумя разделами: "Прочитанные книги" и "Ожидающие прочтения". Эти кнопки заметно выделяются и легко доступны. (рис. 9)

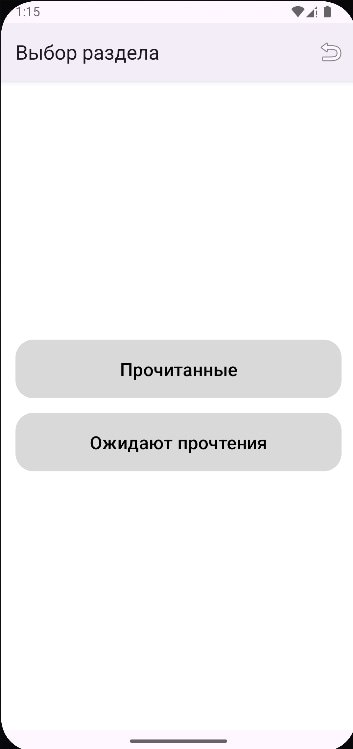


Рисунок 9 – Экран с кнопками для выбора разделов.

1. **Экран списка прочитанных книг**

Экран "Прочитанных книг" отображает все книги, которые были прочитаны пользователем. Здесь также имеется кнопка для добавления новой книги, а также возможность редактировать или удалить книги из списка. (рис. 10)



Рисунок 10 – Экран с перечнем прочитанных книг

1. **Экран списка ожидающих прочтения книг**

На экране "Ожидающие прочтения" представлены книги, которые пользователь планирует прочитать в будущем. Он имеет такой же функционал, как и экран "Прочитанных книг", включая добавление, редактирование и удаление книг. (рис. 11)

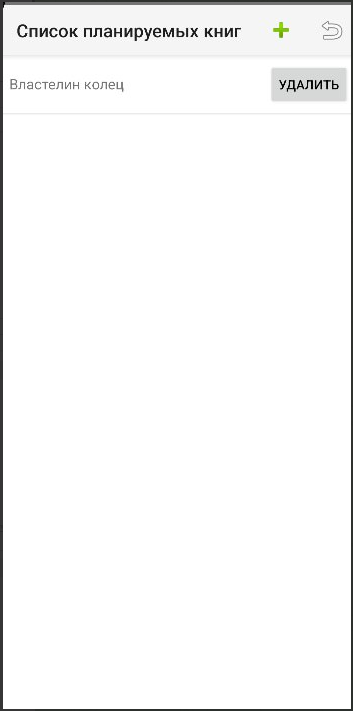


Рисунок 11 – Экран с перечнем планируемых книг.

1. **Экран добавления книги**

Экран добавления книги позволяет пользователю вводить название, автора и описание книги. Этот экран также включает кнопку для добавления книги в соответствующий список, что делает процесс добавления простым и быстрым. (рис. 12)

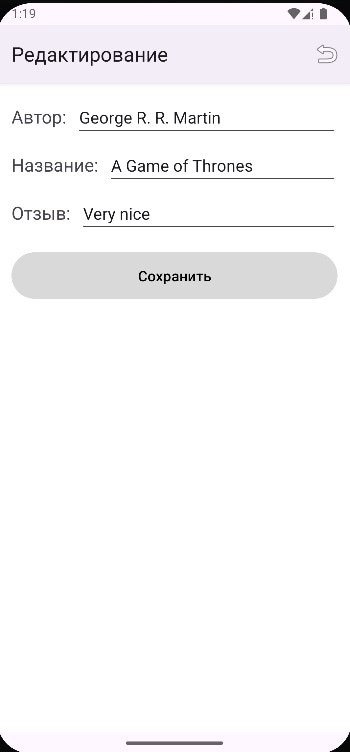


Рисунок 12 – Экран добавления книги с полями для ввода данных.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы был разработано мобильное приложение для трекинга книг, которое включает функции авторизации, регистрации, добавления и редактирования расписания, а также отображения списка занятий. Все поставленные цели были достигнуты.

В процессе разработки использовались современные инструменты и технологии, такие как Android Studio, Java, Volley, PHP, MySQL, что обеспечило стабильную работу приложения и его интеграцию с серверной частью. Основное внимание уделялось удобству пользовательского интерфейса, что было подтверждено тестированием и улучшением взаимодействия с пользователем.

Таким образом, в рамках курсовой работы была создана полностью работоспособная версия мобильного приложения для управления расписанием, что является значительным достижением в освоении мобильной разработки.

# **Список используемой литературы**

* 1. ГОСТ 34.602-89. Система разработки и постановки на производство программных средств. Стандарты программирования. — М.: Издательство стандартов, 1989. — 160 с.
  2. Швецов, И. В. Разработка мобильных приложений на платформе Android / И. В. Швецов, И. Н. Павлов. — М.: МГТУ, 2017. — 256 с.
  3. Канторович, Ю. А. Программирование мобильных приложений: учебник для вузов / Ю. А. Канторович. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2016. — 320 с.
  4. Спектор, А. С. Основы разработки приложений для Android / А. С. Спектор, Е. В. Ермолов. — М.: Курс, 2018. — 300 с.
  5. Android Studio: Руководство по использованию / Google. — М.: Google, 2020. — 48 с.