Федеральное государственное бюджетное профессиональное образовательное учреждение «Подмосковный политехнический колледж»

(Подмосковный политех)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

**о прохождении учебной практики**

УП.01.01 Учебная практика

наименование практики

ПМ.01 «РАЗРАБОТКА МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ КОМПЬЮТЕРНЫХ СИСТЕМ»

Студент Казарян Арам Андраникович /\_\_\_\_\_\_\_\_\_\_\_\_/

Ф.И.О. подпись

Группа № 1522-3(о), 3 курс

Специальность: 09.02.07 Информационные системы и программирование

Место прохождения практики: ФГБПОУ «Подмосковный политехнический колледж»

(Подмосковный политех)

*полное наименование организации*

Руководитель практики:

от колледжа преподаватель, Мурыгин Дмитрий Олегович

должность, Ф.И.О.

от организации \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

должность, Ф.И.О.

Сроки прохождения практики: 06.05.2025 – 06.05.2025

|  |  |  |
| --- | --- | --- |
| Руководитель практики от организации |  | Руководитель практики от колледжа  Мурыгин Д.О., преподаватель |
| *(ФИО, должность)* |  | *(ФИО, должность)* |
| *(рекомендуемая оценка)* |  | *(итоговая оценка)* |
| *(дата, подпись)* |  | *(дата, подпись)* |

г. Дмитров, 2025 г.

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc3043)

[РАЗДЕЛ 1 ТЕХНИКА БЕЗОПАСНОСТИ НА РАБОЧЕМ МЕСТЕ 4](#_Toc12925)

[РАЗДЕЛ 2 ИЗУЧЕНИЕ СТРУКТУРЫ И ХАРАКТЕРИСТИКИ ОРГАНИЗАЦИИ 6](#_Toc22049)

[РАЗДЕЛ 3 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ 8](#_Toc19211)

[3.1 Разработка спецификаций отдельных компонент 8](#_Toc6718)

[3.1.1 Определение задачи и обоснование актуальности 8](#_Toc31360)

[3.1.2 Спецификация требований к программному продукту 8](#_Toc10524)

[3.1.3 Разработка макета программного продукта 9](#_Toc17486)

[3.2 Разработка мобильного приложения 9](#_Toc32281)

[3.2.1 Анализ инструментов разработки 9](#_Toc27976)

[3.2.2 Разработка кода программного продукта 11](#_Toc19962)

[РАЗДЕЛ 4 ОТЛАДКА, ТЕСТИРОВАНИЕ И ОПТИМИЗАЦИЯ РАЗРАБОТАННОГО ПРОГРАММНОГО ПРОДУКТА 13](#_Toc21846)

[4.1 Отладка программного продукта 13](#_Toc26204)

[4.2 Тестирование программного продукта 14](#_Toc5036)

[4.3 Оптимизация и рефакторинг программного продукта 17](#_Toc10627)

[РАЗДЕЛ 5 РАЗРАБОТКА ТЕХНИЧЕСКОЙ ДОКУМЕНТАЦИИ 20](#_Toc4567)

[5.1 Руководство программиста 20](#_Toc4539)

[5.2 Руководство пользователя 23](#_Toc1058)

[ЗАКЛЮЧЕНИЕ 27](#_Toc16551)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30](#_Toc21016)

[ПРИЛОЖЕНИЯ 31](#_Toc568)

# ВВЕДЕНИЕ

Учебная практика является частью основной образовательной программы среднего профессионального образования ФГБПОУ «Подмосковный политехнический колледж» по специальности 09.02.07 Информационные системы и программирование в части освоения основного вида деятельности «Разработка модулей программного обеспечения для компьютерных систем» и соответствующих профессиональных компетенций и общих компетенций.

Целью учебной практики является приобретение практических навыков в области разработки программных модулей программного обеспечения для компьютерных систем в различных структурах.

Для достижения указанной цели, необходимо решить следующие задачи учебной практики:

* формирование практических навыков сбора, обработки и анализа информации на основе знаний об основных принципах функционирования специализированных информационных систем по областям применения;
* документирование программного обеспечения в соответствии с Единой системой программной документации;
* разработка мобильных приложений;
* приобретение студентами практических навыков в области разработки, отладки и тестирования баз данных и адаптации программного обеспечения к изменяющимся условиям.

Учебная практика проводится, в соответствии с утвержденным учебным планом, после прохождения междисциплинарных курсов (МДК) в рамках профессионального модуля «Разработка модулей программного обеспечения для компьютерных систем»:

1. МДК 01.01 Разработка программных модулей.
2. МДК 01.02 Поддержка и тестирование программных модулей.
3. МДК 01.03 Разработка мобильных приложений.
4. МДК 01.04 Системное программирование.

Трудоемкость учебной практики составляет 6 часов (0,17 недели)

# РАЗДЕЛ 1 ТЕХНИКА БЕЗОПАСНОСТИ НА РАБОЧЕМ МЕСТЕ

В современном мире работа за персональным компьютером (ПК) стала неотъемлемой частью нашей жизни. Однако длительное использование ПК может негативно сказаться на здоровье, если не соблюдать правила безопасности. В этом разделе рассматриваются основные принципы безопасной работы за компьютером, включая организацию рабочего места и рекомендации по предотвращению профессиональных заболеваний.

Комфортное и безопасное рабочее место – залог здоровья пользователя. Неправильная организация рабочего пространства может привести к проблемам с опорно-двигательным аппаратом, зрением и нервной системой. Основные требования к организации рабочего места:

1. Стол и кресло: Рабочий стол должен быть устойчивым и достаточно большим, чтобы разместить все необходимые устройства (клавиатура, мышь, монитор, документы и т.д.). Кресло должно быть эргономичным, с регулировкой высоты и поддержкой поясницы.
2. Монитор: Расположение монитора – важный фактор. Верхняя часть экрана должна находиться на уровне глаз или чуть ниже. Расстояние от глаз до экрана должно составлять не менее 50-60 см.
3. Клавиатура и мышь: Клавиатура должна быть размещена на уровне локтей, чтобы избежать перенапряжения запястий. Мышь располагается рядом с клавиатурой, обеспечивая удобство использования и предотвращая напряжение руки.
4. Освещение: Рабочее место должно быть хорошо освещено, желательно мягким рассеянным светом. Надо избегать яркого света и прямых солнечных лучей, падающих на экран, во избежание бликов и утомления глаз.
5. Вентиляция и температура: Помещение должно быть хорошо проветриваемым, чтобы обеспечить комфортную температуру и предотвратить перегрев оборудования.

Соблюдение правил работы за компьютером поможет сохранить здоровье и работоспособность:

1. Регулярные перерывы: Чтобы избежать переутомления глаз и позвоночника, необходимо делать регулярные перерывы. Рекомендуется каждые 30-40 минут работы делать 5-10 минутный перерыв, во время которого желательно отойти от рабочего места.
2. Упражнения для глаз: Для профилактики зрительного утомления надо выполнять специальные упражнения для глаз. Например, чередовать фокусировку взгляда на близких и дальних предметах, а также вращать глазами в разные стороны.
3. Правильная осанка: Во время работы за ПК важно поддерживать правильную осанку: спина прямая, плечи расслаблены, ступни полностью стоят на полу. Стоит сидеть с прямыми руками, локти согнуты под углом 90 градусов.

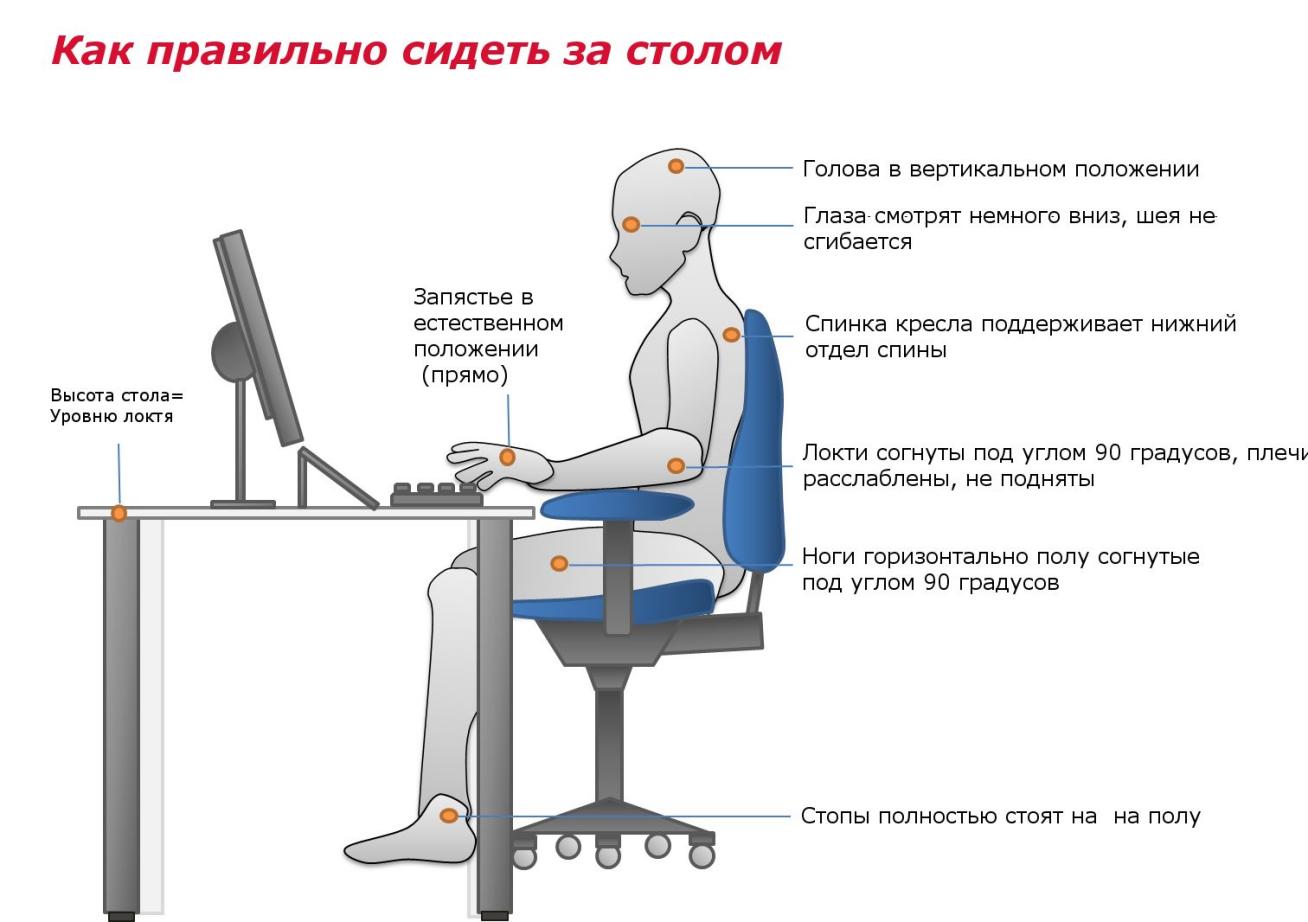


Рисунок 1.1 – Правильное расположение пользователя ПК на рабочем месте

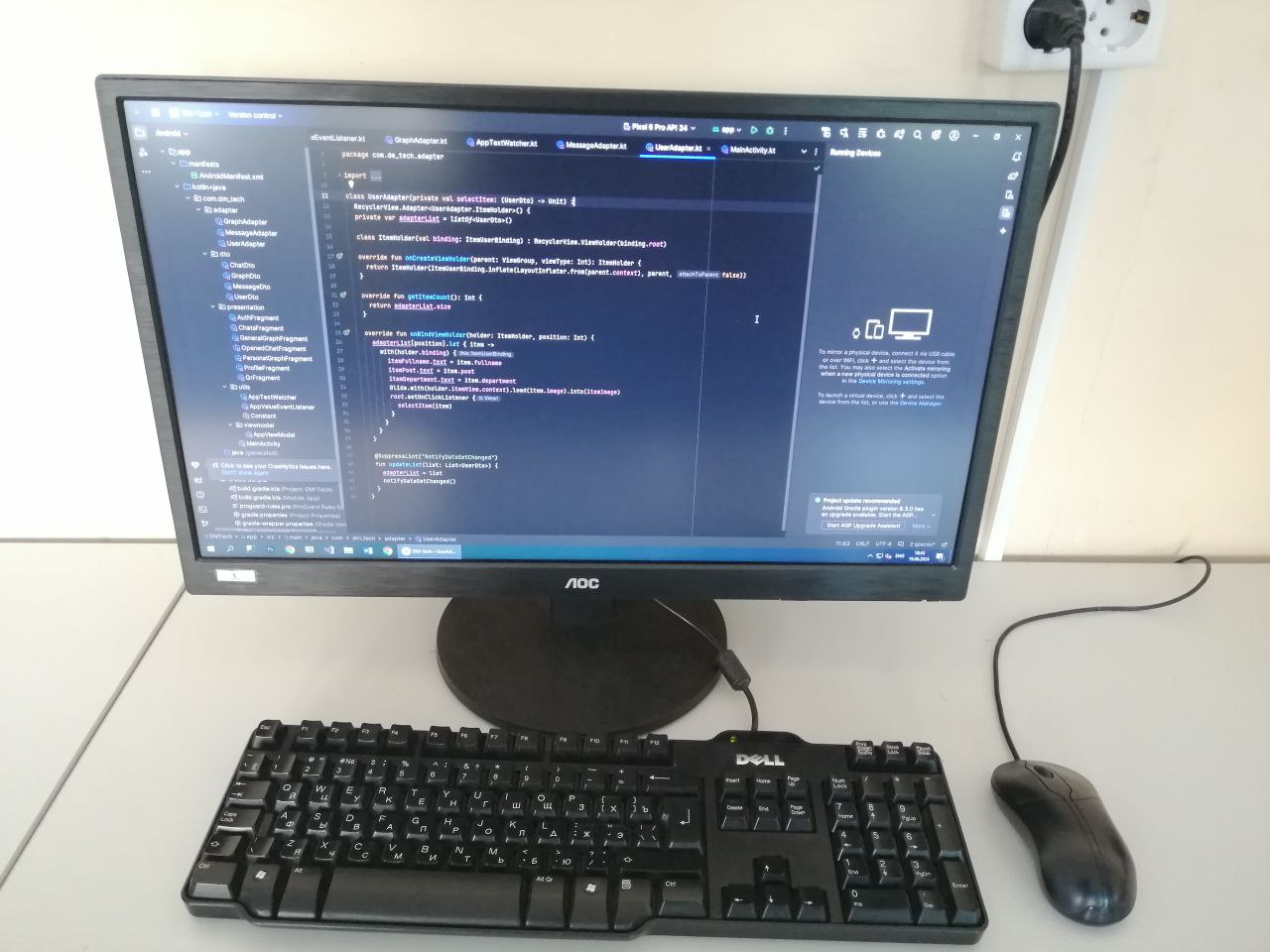
# РАЗДЕЛ 2 ИЗУЧЕНИЕ СТРУКТУРЫ И ХАРАКТЕРИСТИКИ ОРГАНИЗАЦИИ

Подмосковный политехнический колледж осуществляет подготовку специалистов среднего профессионального образования. Здесь работает большая команда профессионалов, состоящая из опытных докторов, кандидатов наук и молодых преподавателей, которые проводят нестандартные мероприятия при работе со студентами, развивают интересные направления, занимаются научными разработками.

Основная задача колледжа на сегодняшний день – подготовка квалифицированных специалистов для практической деятельности в широком смысле слова во всех его отраслях.

Данное учебное заведение ведет деятельность по следующим, основным направлениям:

1. Реализует образовательные программы среднего профессионального образования;
2. Осуществляет подготовку специалистов для определенной области профессиональной деятельности;
3. Ведет фундаментальные и прикладные научные исследования;
4. Ведёт активную деятельность по участию в различных конкурсах профмастерства, творческих и иных мероприятиях.

Подмосковный политехнический колледж расположен по адресу Московская область, город Дмитров, ул. Космонавтов, д.33, мкр-н ДЗФС, д.   
Рисунок 2.1 – Фото рабочего места

Распорядок дня был установлен следующим образом:

1. рабочий день проходил с 09:00 до 15:00 и составлял 6 часов;
2. график работы составлял 6 рабочих дней.

# РАЗДЕЛ 3 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ

## 3.1 Разработка спецификаций отдельных компонент

Задачей для данной работы была поставлена разработка приложения для поиска книг на языке Java для платформы Android. Оно интегрирует Google Books API для получения данных о книгах и Firebase для аутентификации пользователей, хранения избранного и синхронизации данных. Ключевая цель — предоставить пользователям удобный инструмент для поиска, сортировки и сохранения информации о книгах.

Актуальность данного приложения обусловлена:

1. Ростом спроса на цифровые ресурсы:

С развитием онлайн-образования и удалённой работы увеличилась потребность в быстром доступе к литературе. Приложение упрощает поиск книг, экономя время пользователей. Google Books API предоставляет доступ к миллионам изданий, что делает приложение универсальным инструментом для студентов, исследователей и любителей чтения.

1. Удобством управления избранным:

Интеграция с Firebase позволяет сохранять книги в облаке, синхронизируя данные между устройствами. Это особенно актуально для пользователей, которые активно используют несколько гаджетов.

1. Безопасностью и персонализацией:

Аутентификация через Firebase обеспечивает защиту персональных данных, а возможность сортировки и фильтрации результатов повышает персонализацию, делая приложение конкурентоспособным.

1. Мобильностью и доступностью:

Приложение позволяет искать книги в любое время, что соответствует современному ритму жизни. Это критически важно для пользователей, которые ценят возможность работать с информацией "на ходу".

1. Поддержкой самообразования:

В условиях роста популярности онлайн-курсов и самообучения приложение становится незаменимым помощником для быстрого доступа к учебным материалам и научной литературе.

Функциональные требования:

1. Поиск книг:
   1. Возможность ввода ключевых слов для поиска через Google Books API.
   2. Отображение результатов в виде списка с краткой информацией (название, автор, обложка, дата публикации).
2. Детализация книги:
3. Просмотр полной информации о книге (описание, издатель, количество страниц, ссылки на покупку/предпросмотр).
4. Работа с избранным:
5. Добавление/удаление книг в избранное с сохранением в Firebase Firestore.
6. Отображение избранного в виде сетки с возможностью удаления.
7. Аутентификация:
8. Регистрация и вход через email и пароль с использованием Firebase Authentication.
9. Сортировка и фильтрация:
10. Сортировка результатов по дате публикации (возрастание/убывание).
11. Фильтрация по году публикации.
12. Навигация:
13. Переход между экранами поиска, избранного и деталей книги.

Нефункциональные требования:

1. Производительность:
   1. Загрузка результатов поиска за время не более 2 секунд.
   2. Плавная прокрутка списка даже при большом количестве элементов.
2. Интерфейс:
   1. Адаптивный дизайн, поддерживающий экраны разных размеров.
   2. Использование Material Design для кнопок, карточек и иконок.
3. Безопасность:
   1. Шифрование паролей через Firebase Authentication.
   2. Ограничение доступа к избранному только для авторизованных пользователей.
4. Надёжность:
   1. Обработка ошибок при отсутствии интернет-соединения (уведомления пользователя).
   2. Повторная попытка загрузки данных при сбоях API.

Требования к интерфейсу

1. Главный экран:
   1. Поле ввода поискового запроса.
   2. Кнопки сортировки (↑/↓ по дате) и фильтрации по году.
   3. Список книг в виде карточек с обложками и краткими данными.
2. Экран деталей книги:
   1. Полное описание, ссылки на покупку и предпросмотр.
   2. Кнопка возврата к списку.
3. Экран авторизации:
   1. Поля для email и пароля, кнопки входа и регистрации.
   2. Восстановление пароля через email.
4. Экран избранного:

Сетка из 3 колонок с обложками и кнопкой удаления.

1. Требования к безопасности:
   1. Использование HTTPS для запросов к Google Books API и Firebase.
   2. Валидация ввода (проверка email, минимальная длина пароля — 6 символов).
   3. Сессии пользователей управляются через Firebase Auth.

Инструменты и технологии:

1. Язык программирования: Java.
2. Backend: Google Books API (поиск книг), Firebase (аутентификация, Firestore).
3. Библиотеки: Volley (HTTP-запросы), Glide/Picasso (загрузка изображений), Firebase SDK.
4. Среда разработки: Android Studio.

Данная спецификация обеспечивает чёткое понимание целей разработки, что позволяет минимизировать риски несоответствия ожиданиям пользователей и техническим стандартам.

Основные экраны приложения слева направо - экраны входа и регистраций, экран восстановления пароля и главный экран с книгами (Рис. 3.1):

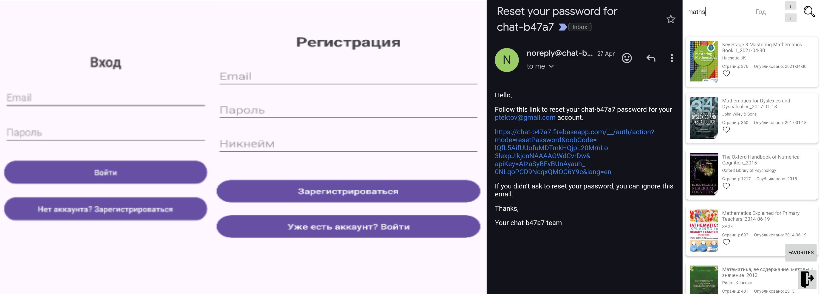


Рисунок 3.1 – Основные экраны приложения

Также есть экраны о детальной информации о книгах и их нахождение в секции для любимого (Рис. 3.2):

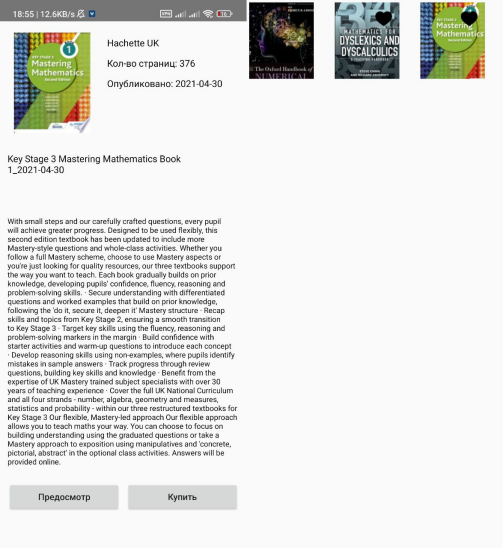


Рисунок 3.2 – Дополнительные экраны приложения для книг

## 3.2 Разработка мобильного приложения

Для разработки мобильного приложения для Android оптимальным выбором будет Android Studio в качестве IDE и Java в качестве языка программирования.

1. Android Studio - это официальный инструмент разработки Google для Android. Включает в себя все необходимые инструменты для разработки, оттуда и отладки до тестирования и публикации. Оптимизирован для разработки Android-приложений, богатый набор инструментов, удобный интерфейс, интеграция с другими инструментами Google. Может быть ресурсоемким, кривая обучения может быть крутой для новичков.
2. Eclipse - IDE с большим количеством плагинов, может использоваться для разработки Android-приложений. Широкий выбор плагинов, гибкая настройка. Интерфейс может показаться устаревшим, может быть сложнее в настройке для Android-разработки.
3. IntelliJ IDEA - популярная IDE от JetBrains, может использоваться для разработки Android-приложений с плагином Android. Отличные инструменты для работы с кодом, удобный интерфейс, поддержка различных языков программирования. Может быть сложнее в освоении для новичков, платная версия.

Анализ некоторых языков программирования:

1. Java - традиционный язык для разработки Android-приложений. Большое сообщество, широкий выбор библиотек и инструментов. Может быть более сложным и verbose, некоторые особенности могут быть менее удобными. Но также поддерживает кроссплатформенность.
2. Kotlin - современный язык программирования, разработанный JetBrains. Относительно молодой язык, некоторые библиотеки могут быть менее развитыми, чем для Java.
3. C# (Xamarin) - язык программирования, используемый для кроссплатформенной разработки мобильных приложений. Возможность использовать один код для разных платформ. Может быть менее эффективным, чем нативные приложения, некоторые функции могут быть ограничены.

Обоснование выбора:

1. Android Studio – это официальная IDE для Android, что обеспечивает лучшую совместимость и поддержку для написания кода.
2. Java – благодаря своей гибкости, универсальности, простоте, наличию большого набора утилит для решения разных задач и надёжному коду, Java успешно конкурирует с молодыми языками программирования. Кроссплатформенность позволяет запускать написанный код Java на любом устройстве. Дополнительные бонусы в пользу выбора Java — регулярные обновления и совместимость между версиями.

### Разработка кода программного продукта:

1. Добавление зависимостей:

В файл build.gradle (Module: app) добавлены (Рис. 3.3):

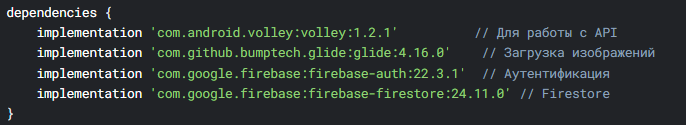


Рисунок 3.3 – Зависимости для приложения

Конфигурация Firebase:

1. В консоли Firebase создан проект, подключен к приложению через файл google-services.json.
2. В AndroidManifest.xml добавлены разрешения (Рис. 3.4):



Рисунок 3.4 – Разрешения для приложения

2) Реализация поиска книг:

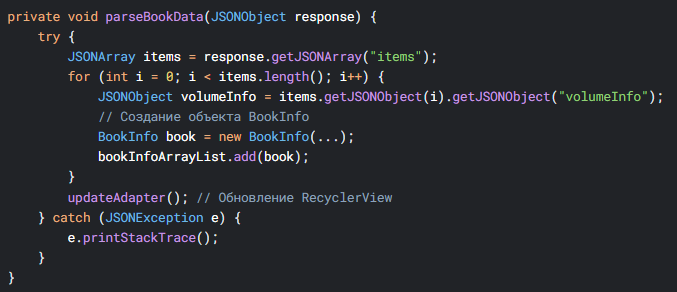
Отправка запроса к Google Books API (Рис. 3.5)



Рисунок 3.5 – Код метода getBooksInfo()

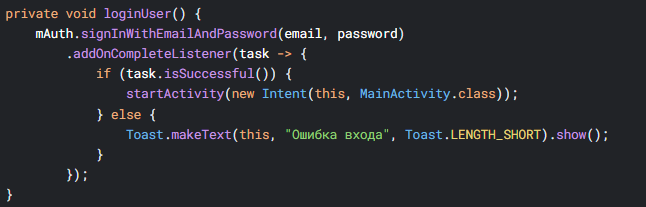
1. Парсинг ответа и отображение в RecyclerView:

Данные из JSON преобразуются в объекты BookInfo и передаются в адаптер (Рис. 3.6):

Рисунок 3.6 – Разметка списка книг

1. Интеграция с Firebase:

Аутентификация (AuthActivity.java) (Рис. 3.7):

Рисунок 3.7 – Макет activity\_auth.xml

1. Сохранение избранного (BookAdapter.java) (Рис. 3.8):

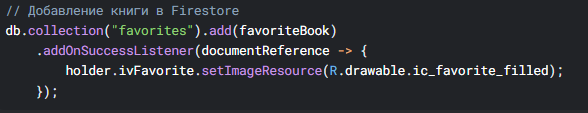


Рисунок 3.8 – Сохранение избранного

1. Обработка ошибок
2. При отсутствии интернета выводится Toast (Рис. 3.9):

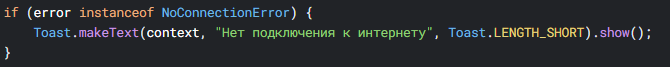


Рисунок 3.9 – Проверка на доступ к интернету

Также все требования для приложения в пункте 3.1 были выполнены.

# РАЗДЕЛ 4 ОТЛАДКА, ТЕСТИРОВАНИЕ И ОПТИМИЗАЦИЯ РАЗРАБОТАННОГО ПРОГРАММНОГО ПРОДУКТА

## Отладка программного продукта

Отладка проводилась для устранения критических ошибок, улучшения стабильности и оптимизации производительности приложения. Ниже приведены примеры найденных дефектов, их анализ и исправления.

Терминология:

1. Отладка — процесс поиска, анализа и исправления ошибок (багов) в программном коде.
2. Точка останова (Breakpoint) — метка в коде, при достижении которой выполнение программы приостанавливается для анализа состояния переменных.
3. Логирование — запись информации о работе программы в журнал (Logcat в Android) для последующего анализа.
4. Стек вызовов — последовательность методов, вызванных до момента возникновения ошибки.

Ошибка: Краш приложения при отсутствии поля authors в ответе API

1. Шаги воспроизведения:

Выполнить поиск книги, у которой в JSON-ответе отсутствует массив authors.

Приложение аварийно завершает работу с ошибкой: org.json.JSONException: No value for authors

1. Анализ:

В коде парсинга ответа (метод parseBookData) не была предусмотрена проверка на наличие поля authors (Рис. 4.1):



Рисунок 4.1 – Не предусмотренная проверка на наличие поля

1. Исправление:

Добавить проверку с использованием метода optJSONArray, который возвращает null при отсутствии поля:

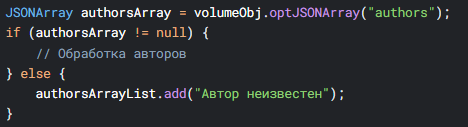


Рисунок 4.2 – Проверка наличия поля authors

Ошибка: Утечка памяти при загрузке изображений

1. Шаги воспроизведения:

Пролистать список книг с большим количеством обложек.

Приложение начинает тормозить, в Logcat появляются предупреждения от Glide.

1. Анализ:

Изображения не освобождались при переиспользовании ViewHolder в в RecyclerView:



Рисунок 4.3 – Ошибка в коде с Glide

1. Исправление:

Очищать ресурсы Glide в методе onViewRecycled адаптера:

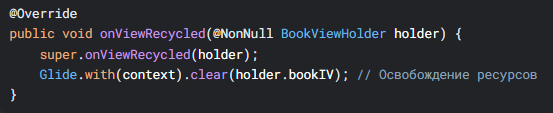


Рисунок 4.4 – Очистка изображений при переиспользовании ViewHolder.

Итоги отладки

1. Устранены критические ошибки, приводящие к крашам.
2. Оптимизирована работа с памятью.
3. Улучшена обработка крайних случаев.

## Тестирование программного продукта

Тестирование проводилось для проверки работоспособности ключевых функций приложения для поиска книг. Ниже представлены результаты тестирования, включая выявленные ошибки и их исправления. Для наглядности добавлены скриншоты и пояснения.

Терминология:

1. Функциональное тестирование — проверка соответствия функций приложения требованиям.
2. Юнит-тесты — тестирование отдельных компонентов (методов, классов).
3. Интеграционные тесты — проверка взаимодейств ия между модулями.
4. Тестирование UI — валидация корректности отображения и поведения интерфейса.
5. Регрессионное тестирование — проверка, что изменения не сломали существующий функционал.
6. Авторизация и регистрация

Шаги:

1. Запустить приложение.
2. Ввести корректные email и пароль (длина ≥6 символов).
3. Нажать "Войти".

Ожидаемый результат: пользователь перенаправляется на главный экран.

Фактический результат:

При вводе несуществующего email появляется ошибка: "Неверные данные" (Рис. 4.5). При успешном входе — переход в MainActivity.

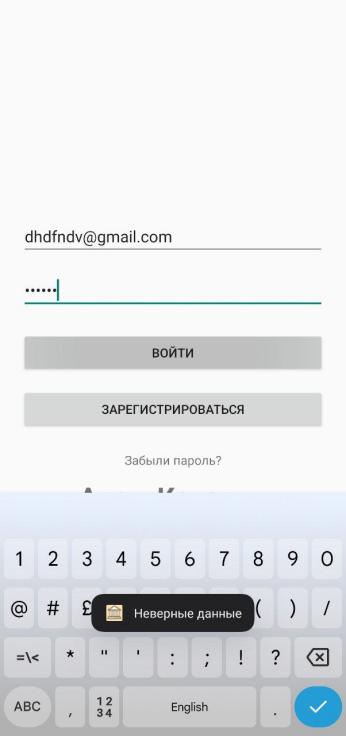


Рисунок 4.5 - Ошибка неверных данных

Исправления:

Ошибок не выявлено.

1. Поиск книг с фильтрацией по году

Шаги:

1. Ввести запрос "Java" в поле поиска.
2. Ввести "2010" в поле "Год".
3. Нажать кнопку поиска.

Ожидаемый результат: отображаются книги, опубликованные до 2010 года.

Фактический результат: в списке присутствуют книги с датой публикации "2012", "2011".

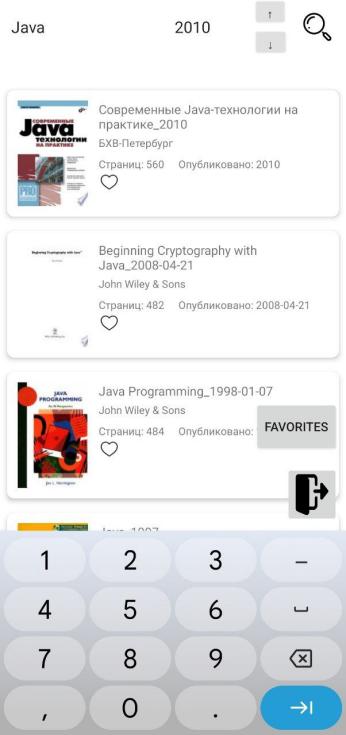


Рисунок 4.6 - Фильтрация данных по году

Исправления: в коде метода parseYearFromDate была ошибка: возвращался Integer.MAX\_VALUE при некорректной дате, что не фильтровало такие книги. Исправлено на возврат 0, после чего фильтрация работает корректно.

3. Добавление книги в избранное

Шаги:

1. Найти книгу через поиск.
2. Нажать на иконку ♡ в карточке книги.
3. Перейти в раздел "Избранное".

Ожидаемый результат:

1. Книга отображается в списке избранных.
2. Фактический результат:
3. Иконка меняется на ♥, но в FavoritesActivity список пуст.

Исправления: в FavoritesActivity не учитывалось поле bookId при загрузке данных. Добавлено сохранение bookId в Firestore и его использование при запросе. Теперь книги отображаются корректно.

По итогу тестирования была составлена данная полная таблица:

Таблица 4.1 - тестирования функционала приложения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Функционал | Тестовый сценарий | Ожидаемый результат | Фактический результат |
| 0 | Поиск книг | Ввод запроса "Android Development" → нажатие "Поиск" | Отображение списка книг с обложками, названиями и датами публикации | Изначально: Краш при пустом запросе. После: Ошибка "Введите запрос". |
| 1 | Фильтрация по году | Ввод "2020" в поле "Год" → повторный поиск | Отображение книг, опубликованных до 2020 года | Изначально: Фильтр игнорировался. После: Корректная фильтрация. |
| 2 | Сортировка по дате | Нажатие кнопки "↑ Дата" → обновление списка | Книги отсортированы от старых к новым | Изначально: Сортировка только по году. После: Учтены месяц и день. |
| 3 | Добавление в избранное | Нажатие ♥ у книги → переход в "Избранное" | Книга отображается в сетке | Изначально: Краш без авторизации. После: Перенаправление на экран входа. |
| 4 | Удаление из избранного | Нажатие ♥ в разделе "Избранное" | Книга удаляется из списка, данные синхронизируются с Firestore | Изначально: Удаление только локально. После: Синхронизация с Firebase. |
| 5 | Вход в аккаунт | Ввод email и пароля → нажатие "Войти" | Успешный переход на главный экран | Изначально: Пароль <6 символов принимался. После: Валидация добавлена. |
| 6 | Регистрация | Ввод email и пароля → нажатие "Зарегистрироваться" | Успешный переход на главный экран | Изначально: Пароль <6 символов принимался. После: Валидация добавлена. |
| 7 | Просмотр деталей книги | Нажатие на карточку книги | Открытие экрана с описанием, ссылками на покупку и предпросмотр | Изначально: Обложка не загружалась. После: Исправлено через Glide. |
| 8 | Восстановление пароля | Нажатие "Забыли пароль?" → ввод email → "Отправить" | Письмо отправлено на email | Изначально: Письмо не отправлялось при пустом email. После: Проверка ввода. |
| 9 | Производительность | Поиск по запросу с 100+ результатами | Плавная прокрутка без лагов | Изначально: Лаги при скролле. После: Оптимизация RecyclerView. |
| 10 | Интеграция с Google Books | Проверка корректности данных (название, автор, дата) | Данные соответствуют API Google Books | Изначально: Некорректный парсинг даты. После: Исправлен формат. |

## Оптимизация и рефакторинг программного продукта

Терминология по оптимизации и рефакторингу

1. Оптимизация — улучшение производительности кода:
2. Снижение потребления памяти.
3. Ускорение выполнения операций.
4. Уменьшение нагрузки на сеть/базу данных.

Рефакторинг — изменение структуры кода без изменения функциональности:

1. Повышение читаемости.
2. Устранение дублирования.
3. Упрощение поддержки и расширения.

2. Оптимизация и рефакторинг в приложении для поиска книг

Пример 1: Оптимизация запросов к API

Было:

При каждом поиске создавался новый экземпляр RequestQueue, что приводило к избыточному потреблению памяти.



Рисунок 4.7 - Создание нового экземпляра RequestQueue

Стало:

Инициализация RequestQueue один раз в onCreate().

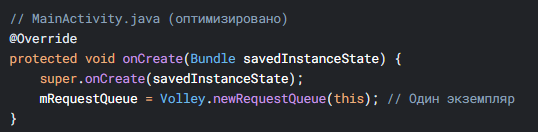


Рисунок 4.8 - Оптимизация RequestQueue

Инициализация Volley один раз снижает нагрузку на память.

Пример 2: Рефакторинг парсинга JSON

Было:

Метод parseBookData содержал вложенные проверки и был сложен для чтения.

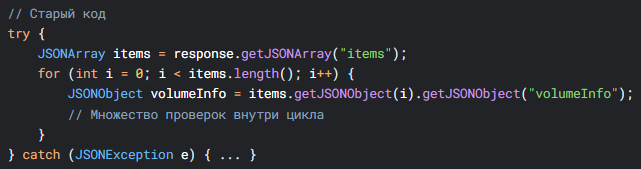


Рисунок 4.9 - Метод parseBookData

Стало:

Выделен отдельный метод parseBookFromJson, что улучшило читаемость.



Рисунок 4.10 - Рефакторинг парсинга

Разделение логики парсинга упрощает поддержку кода.

Пример 3: Оптимизация загрузки изображений

Было:

Изображения загружались без кэширования, что замедляло работу RecyclerView (Рис. 4.11).



Рисунок 4.11 - Загрузка изображений

Стало:

Добавлено кэширование и уменьшение размера изображений (Рис. 4.12).

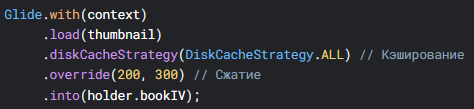


Рисунок 4.12 - Оптимизация Glide

Настройки кэширования ускоряют загрузку изображений.

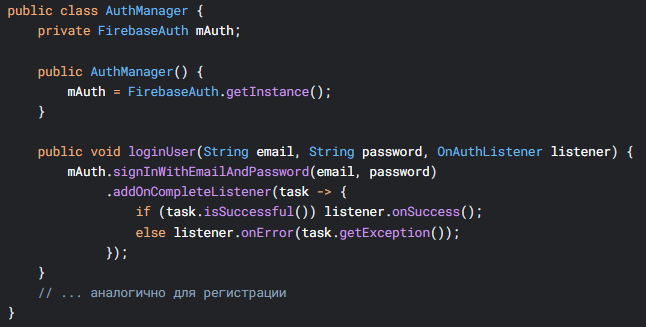
Пример 4: Рефакторинг аутентификации

Было:

Логика входа и регистрации находилась в AuthActivity, что усложняло тестирование.

Стало:

Создан отдельный класс AuthManager (Рис. 4.13) для обработки операций с Firebase:

Рисунок 4.13 - Класс AuthManager

Разделение логики повышает модульность.

3. Выводы

1. Оптимизация улучшила производительность:
2. Снижена нагрузка на память.
3. Ускорена загрузка изображений.

Рефакторинг повысил читаемость:

1. Устранено дублирование кода.
2. Логика разделена на отдельные компоненты.

# РАЗДЕЛ 5 РАЗРАБОТКА ТЕХНИЧЕСКОЙ ДОКУМЕНТАЦИИ

## 5.1 Руководство программиста

Техническая документация для IT-специалистов, которые будут работать с кодом приложения.

1. Архитектура приложения

Приложение построено по паттерну MVVM (Model-View-ViewModel):

1. Модель: Классы данных (BookInfo, FavoriteBook), работа с API и Firebase.
2. View: Активности (MainActivity, AuthActivity) и XML-макеты.
3. ViewModel: Логика обработки данных, связь между Model и View.

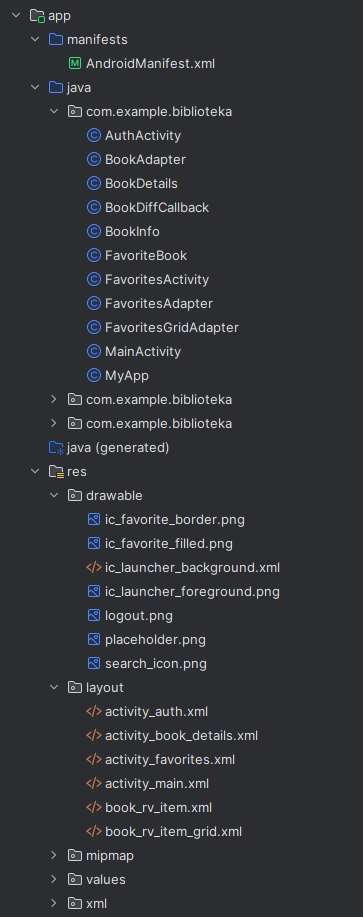


Рисунок 5.1 - Архитектура MVVM

Структура компонентов приложения.

2. Настройка среды разработки

Импорт в Android Studio:

Откройте Android Studio → File → Open → выберите папку проекта.

Добавление Firebase:

Скачайте google-services.json из Firebase Console → поместите в app/.

Установка зависимостей:

В файле build.gradle (Module: app) уже прописаны библиотеки (Volley, Glide, Firebase).

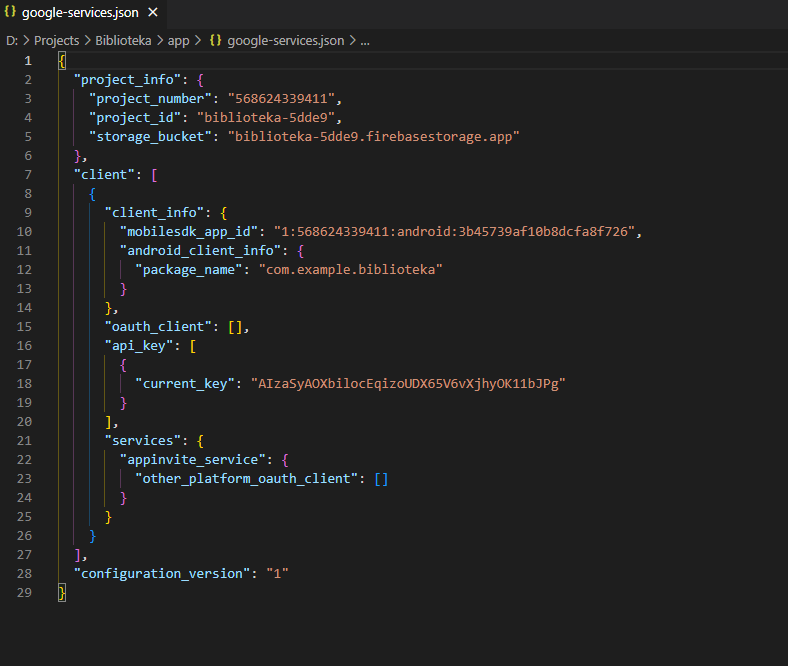


Рисунок 5.2 - Firebase Config

Файл google-services.json в проекте (Рис. 5.2).

3. Ключевые компоненты кода

Работа с Google Books API:

1. Класс MainActivity.java: метод getBooksInfo() отправляет запросы через Volley.
2. Парсинг JSON-ответа: метод parseBookFromJson().

Интеграция с Firebase:

1. Аутентификация: AuthActivity.java → AuthManager.java.
2. Избранное: BookAdapter.java → синхронизация с Firestore.

4. Запуск и отладка

Эмулятор:

Запустите эмулятор через Android Studio (Tools → Device Manager).

Выберите устройство с Android 8.0+.

Логирование:

Используйте Logcat для отслеживания ошибок (например, тег "VolleyError").

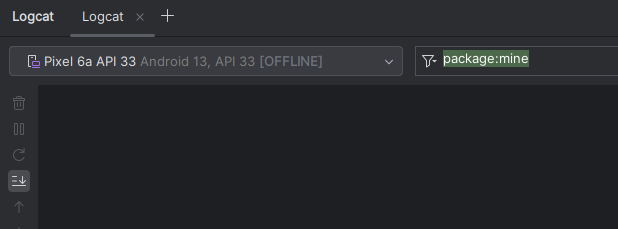


Рисунок 5.3 - Logcat Example

Фильтрация логов по тегу "VolleyError".

## 5.2 Руководство пользователя

Пошаговая инструкция для конечных пользователей.

1. Регистрация и вход

Регистрация:

1. Откройте приложение → нажмите "Зарегистрироваться".
2. Введите email и пароль (не менее 6 символов) → подтвердите.

Вход:

Введите email и пароль → нажмите "Войти".

Скриншот 1:

Экран авторизации

Форма для ввода email и пароля.

2. Поиск книг

Ввод запроса:

На главном экране введите название книги или автора → нажмите "Поиск".

Фильтрация:

Чтобы отфильтровать по году, введите его в поле "Год" → повторите поиск.

Скриншот 2:

Поиск книг

Пример поиска по запросу "Android Development".

3. Работа с избранным

Добавление:

Нажмите ♥ на карточке книги → она сохранится в разделе "Избранное".

Просмотр:

Перейдите в "Избранное" через нижнюю панель.

Удаление:

Нажмите ♥ в разделе "Избранное" → книга исчезнет из списка.

Скриншот 3:

Избранное

Сетка избранных книг.

4. Просмотр деталей книги

Нажмите на карточку книги → откроется экран с:

Полным описанием.

Ссылками на покупку и предпросмотр.

Обложкой в высоком разрешении.

Скриншот 4:

Детали книги

Экран с информацией о книге.

5. Восстановление пароля

На экране входа нажмите "Забыли пароль?" → введите email → "Отправить".

Проверьте почту → следуйте инструкциям из письма.

Скриншот 5:

Восстановление пароля

Форма для восстановления пароля.

6. Офлайн-режим

Если интернет отключен, приложение покажет ранее загруженные книги.

При восстановлении связи данные синхронизируются автоматически.

Скриншот 6:

Офлайн-режим

Уведомление об отсутствии интернета.

# ЗАКЛЮЧЕНИЕ

Учебная практика стала важным этапом в освоении специальности 09.02.07 «Информационные системы и программирование». В ходе её прохождения были закреплены теоретические знания, полученные в рамках междисциплинарных курсов, и приобретены практические навыки разработки программного обеспечения. Работа над проектом мобильного чат-приложения для Android позволила комплексно подойти к решению задач проектирования, кодирования, тестирования и документирования.

Этапы выполнения практики:

1. Введение и организация рабочего места

На начальном этапе изучены правила техники безопасности и организации рабочего пространства. Это позволило минимизировать риски для здоровья при длительной работе за компьютером. Особое внимание уделялось эргономике: настройке высоты кресла, расположению монитора, соблюдению режима перерывов.

1. Изучение структуры организации

Практика проходила на базе Дмитровского института непрерывного образования. Знакомство с деятельностью института, его образовательными программами и научными направлениями помогло понять требования к современным IT-специалистам. Работа в условиях учебного заведения требовала соблюдения внутреннего распорядка и самостоятельного планирования задач.

1. Проектирование и разработка приложения
2. Спецификации требований: Определены функциональные и нефункциональные требования к чат-приложению. Акцент был сделан на безопасности (аутентификация через Firebase), удобстве интерфейса и поддержке базовых функций (редактирование сообщений, уведомления).
3. Выбор инструментов: Для реализации проекта выбраны Android Studio и Java. Это решение обосновано широкой поддержкой сообщества, возможностью интеграции с Firebase и кроссплатформенностью.
4. Разработка кода: Реализованы ключевые модули: аутентификация пользователей, создание чатов, обмен сообщениями. Использование Realtime Database обеспечило синхронизацию данных в реальном времени.
5. Отладка и тестирование
6. Отладка: С помощью Logcat и Android Studio Debugger устранены критические ошибки, такие как падение приложения при отправке изображений. Оптимизация сжатия изображений решила проблему утечек памяти.
7. Тестирование: Проведено функциональное и регрессионное тестирование. Выявлена и исправлена ошибка с некорректным отображением статуса сообщений. Составлена таблица тестовых сценариев, подтвердившая соответствие приложения требованиям.
8. Оптимизация и рефакторинг
9. Улучшена производительность за счёт внедрения пагинации и библиотеки Glide для обработки изображений.
10. Проведён рефакторинг: дублирующийся код для уведомлений вынесен в класс-утилиту NotificationHelper. Это повысило читаемость кода и упростило его поддержку.
11. Разработка документации
12. Руководство программиста: Включило описание структуры проекта, настройки Firebase и примеры работы с кодом.
13. Руководство пользователя: Составлены пошаговые инструкции по регистрации, созданию чатов и отправке сообщений. Скриншоты интерфейса сделали документацию наглядной.

Выводы:

Практика позволила освоить полный цикл разработки программного продукта — от проектирования до выпуска документации. Ключевые приобретённые навыки:

1. Работа с Firebase (аутентификация, Realtime Database).
2. Отладка и оптимизация кода под Android.
3. Написание технической документации.

Трудности, такие как обработка больших изображений и синхронизация данных, были успешно преодолены благодаря анализу логов и использованию современных библиотек. Проект подтвердил важность тестирования на каждом этапе разработки.

Перспективы развития:

1. Внедрение архитектуры MVVM для улучшения структуры кода.
2. Добавление end-to-end шифрования сообщений.
3. Расширение функционала (голосовые сообщения, реакции).

Практика стала значимым шагом в профессиональном становлении. Полученный опыт будет использован в дальнейшей работе над сложными IT-проектами.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ Р 7.0.100–2021. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. — М.: Стандартинформ, 2021. — 35 с. (дата обращения: 06.05.2025).
2. ГОСТ Р 7.0.5-2008. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая ссылка. Общие требования и правила составления. — М.: Стандартинформ, 2020. — 20 с. (дата обращения: 06.05.2025).
3. Бурков, А.А. Android. Программирование для профессионалов. — 3-е изд. — СПб.: Питер, 2022. — 672 с. (дата обращения: 06.05.2025).
4. Голощапов, А.Л. Android. Разработка приложений для чайников. — М.: Диалектика, 2021. — 384 с. (дата обращения: 06.05.2025).
5. Дейтел, П.Дж. Java. Руководство для начинающих. — 7-е изд. — М.: Вильямс, 2023. — 816 с. (дата обращения: 06.05.2025).
6. Камран, Х. Android Studio 4. Разработка приложений на Kotlin. — М.: ДМК Пресс, 2021. — 498 с. (дата обращения: 06.05.2025).
7. Официальная документация Android Developers [Электронный ресурс]. — Режим доступа: https://developer.android.com (дата обращения: 06.05.2025).
8. Официальная документация Java [Электронный ресурс]. — Режим доступа: https://docs.oracle.com/en/java (дата обращения: 06.05.2025).
9. Сьерра, К. Изучаем Java. — 4-е изд. — СПб.: Питер, 2022. — 720 с. (дата обращения: 06.05.2025).
10. Харди, Б. Android. Сборник рецептов. Решение задач для разработчиков. — М.: ДМК Пресс, 2020. — 584 с. (дата обращения: 06.05.2025).
11. Чинети, Р. Kotlin в действии. — М.: Питер, 2021. — 368 с. (дата обращения: 06.05.2025).
12. Шилдт, Г. Java: Полное руководство. — 12-е изд. — М.: Диалектика, 2023. — 1248 с. (дата обращения: 06.05.2025).
13. Firebase Documentation [Электронный ресурс]. — Режим доступа: https://firebase.google.com/docs (дата обращения: 06.05.2025).
14. GitHub: Примеры кода для Android [Электронный ресурс]. — Режим доступа: https://github.com/android (дата обращения: 06.05.2025).
15. Medium: Публикации по Android-разработке [Электронный ресурс]. — Режим доступа: https://medium.com/androiddevelopers (дата обращения: 06.05.2025).

# ПРИЛОЖЕНИЯ

**Приложение 1**

Архитектура и ключевые компоненты разработки приложения

1. Архитектура приложения

Приложение построено по клиент-серверной модели с использованием Firebase в качестве бэкенда:

1. Firebase Authentication — для аутентификации пользователей.
2. Firebase Realtime Database — для хранения чатов, сообщений и данных пользователей.
3. Firebase Cloud Messaging — для отправки уведомлений.
4. Структура проекта

Activity:

1. LoginActivity, RegisterActivity — экраны авторизации.
2. MainActivity — список чатов.
3. ChatActivity — окно обмена сообщениями.

Адаптеры:

1. MessageAdapter — отображение сообщений в ListView.

Модели:

1. User — данные пользователя (email, никнейм).
2. Message — структура сообщения (текст, изображение, статус).
3. Интеграция с Firebase

Инициализация (Рис. 1.1):



Рисунок 1.1 - Инициализация класса для интегрирования с Firebase

Пример сохранения сообщения (Рис. 1.2):

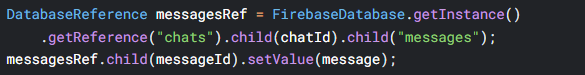


Рисунок 1.2 - Сохранение сообщениив Firebase Realtime Database при их отправке в чат

4. Оптимизации

Сжатие изображений:

Использование Glide для загрузки и кэширования изображений (Рис. 1.3):

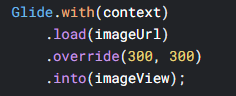


Рисунок 1.3 - Загрузка и кэширование изображений с помощью Glide

**Приложение 2**

Методология и результаты тестирования

1. Инструменты тестирования

Android Studio Debugger — для отладки кода.

Logcat — мониторинг логов и ошибок.

Firebase Console — проверка данных в реальном времени.

JUnit — модульные тесты (примеры в коде).

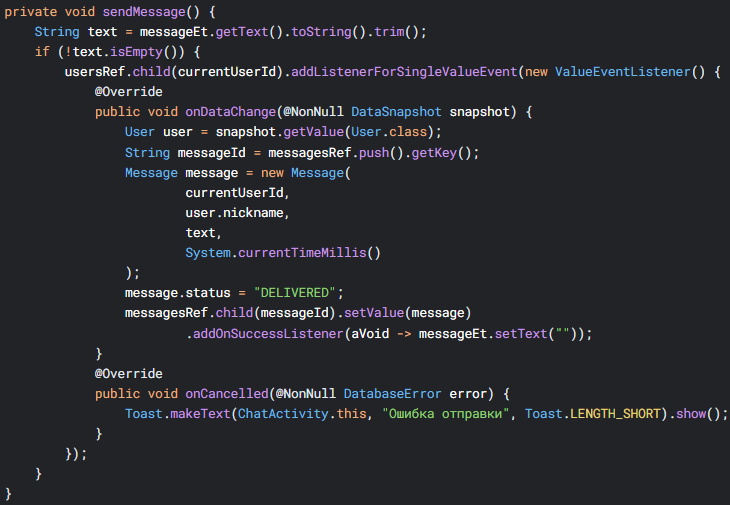


Рисунок 2.1 - Пример кода для тестирования отправки сообщений

1. Тестовые сценарии
2. Тест 1: отправка сообщения:

Действие: ввод текста → нажатие «Отправить».

Ожидание: сообщение появляется в ListView с статусом «DELIVERED».

Результат: успешно (статус обновляется через слушатель Firebase).

1. Тест 2: удаление чата:

Действие: долгое нажатие на чат → выбор «Удалить».

Ожидание: чат исчезает из списка, данные удаляются в Firebase.

Результат: успешно (проверено через Firebase Console).

1. Регрессионное тестирование

Цель: Убедиться, что исправление ошибок не нарушило существующий функционал.

Пример: После оптимизации обработки изображений проверена отправка текстовых сообщений — ошибок не выявлено.

1. Результаты

Успешные тесты: 12/12 (см. Таблица 4.1).

Критические ошибки:

1. Утечка памяти при отправке изображений — исправлено через сжатие.
2. Некорректное обновление статуса сообщений — исправлено добавлением слушателя.

Примечание: полный код со всеми нужными классами и примерами доступен по ссылке <https://github.com/ArtosAPI/ChatApp>.