

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

Кафедра

вычислительной техники

«Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация
автоматизированных систем специального назначения»
Специализация: «Эксплуатация вычислительных машин, комплексов,
систем и сетей специального назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

**ANDROID – ПРИЛОЖЕНИЕ «БАЗА
ЭЛЕКТРОННЫХ КОМПОНЕНТОВ»**

Студент

Сетяев Сетяев Артем Николаевич
(подпись, дата) 10.05.2026 (ФИО полностью)

Руководитель

М. А. Митрохин
(подпись, дата) 10.05.2026 (фамилия, инициалы)

Нормоконтролёр

А. С. Бычков
(подпись, дата) 10.05.2026 (фамилия, инициалы)

Рецензент

зав. каф. Программирования ПК и ЭВМ А. И. Мартышкин
(должность, место работы) (подпись, дата) 10.05.2026 (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 г. № 14)

Заведующий кафедрой

М. А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от

17 июня 2026 г. № 13)

Секретарь ГЭК

Е. И. Гурин

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«14» 02 2026 г.

ЗАДАНИЕ

1. Студент Сетяев Артем Николаевич гр.21ВОЭ1 факультета вычислительной техники.

2. Тема работы: Android - приложение «База электронных компонентов»

Тема утверждена приказом ПГУ № 122/о от "11" февраля 2026 г.

3. Руководитель дипломной работы д.т.н., профессор кафедры ВТ Митрохин М.А.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Требуется разработать мобильное Android-приложение «База электронных компонентов» для автоматизации поиска и подбора электронных компонентов. Система должна обеспечивать:

- регистрацию и аутентификацию пользователей (инженеров, радиолюбителей);
- подбор электронных компонентов по параметрам (маркировка, тип корпуса, производитель);
- создание и отслеживание статуса заявок на подбор компонентов;
- просмотр всех заявок пользователя в личном кабинете;
- редактирование профиля пользователя (ФИО, должность, кабинет, телефон);
- просмотр списка всех пользователей администратором;
- отправку push-уведомлений об изменении статуса заявки;
- работу в режиме офлайн с локальным кэшированием данных;
- просмотр технических характеристик компонентов и даташитов.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.014.001 ПЗ

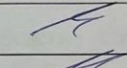
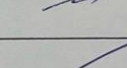
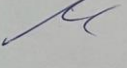
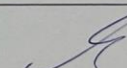
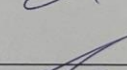
Лист

3

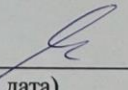
5. Перечень подлежащих разработке вопросов

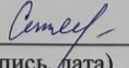
Требуется провести анализ предметной области и существующих аналогов мобильных приложений для работы с электронными компонентами, сформулировать требования к системе, обосновать выбор технологий. Необходимо спроектировать архитектуру мобильного приложения и структуру базы данных (Firebase Realtime Database), разработать клиентскую часть на платформе Android с использованием языка Java. Реализовать модули авторизации пользователей, личного кабинета, поиск и подбор электронных компонентов, создания и управление заявками на подбор компонентов, отслеживания статуса заявок с отправкой уведомлений (OneSignal), административной панели (просмотр всех пользователей и заявок), работы в офлайн-режиме с кэшированием данных

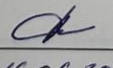
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	15	12.02.2026-15.02.2026	
2.	Анализ требований	5	16.02.2026-28.02.2026	
3.	Выбор технологий, применяемых при проектировании и разработке	10	01.03.2026-20.03.2026	
4.	Проектирование базы данных и пользовательского интерфейса	20	20.03.2026-30.03.2026	
5.	Разработка серверной и клиентской части	60	31.03.2026-30.04.2026	
6.	Оформление пояснительной записки	15	01.05.2026-20.05.2026	

Дата выдачи задания "11" февраля 2026 г.

Руководитель дипломной работы  Митрохин М.А.
(подпись, дата)

Задание к исполнению принял студент  Сетяев А.Н.
(подпись, дата)

Работу к защите допустить
Декан факультета  Фионова Л.Р.
16.06.2026

Реферат

Пояснительная записка содержит 106 листов, 32 рисунка, 18 таблицы, 18 источников и 1 приложение.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ,
РАЗРАБОТКА ПРИЛОЖЕНИЯ, БАЗА ДАННЫХ

Объектом работы является автоматизация поиска электронных компонентов.

Предметом дипломной работы выступает приложение для работы с электронными компонентами.

Целью работы является разработка мобильного Android приложения «База электронных компонентов».

В результате проделанной работы разработано программное средство для работы с электронными компонентами.

					ПГУ 1.090501.014.001 ПЗ		
Изм	Лист	№ докум.	Подпись	Дата			
Разраб.		Сетяев А.Н.	<i>Сетяев А.Н.</i>	16.06.20			
Пров.		Митрохин М.А.	<i>Митрохин М.А.</i>	16.06.20			
Н. Контр.		Бычков А.С.	<i>Бычков А.С.</i>	16.06.20			
Уте.		Митрохин М.А.	<i>Митрохин М.А.</i>	16.06.20			
					п	Лист	Листов
						2	106
					ФВТ, каф. ВТ гр. 21ВОЭ1		

Android - приложение «База
электронных компонентов»

Содержание

Перечень сокращений и обозначений.....	5
Введение.....	6
1 Анализ предметной области.....	8
1.1 Описание предметной области.....	8
1.2 Выбор CASE-средств для описания бизнес-процессов.....	10
1.3 Обзор и анализ аналогов мобильных приложений для для работы с электронными компонентами.....	12
2 Техническое задание.....	15
2.1 Разработка требований к мобильному приложению для работы с электронными компонентами.....	15
2.2 Постановка задачи на разработку мобильного приложения для работы с электронными компонентами.....	17
3 Проектирование мобильного приложения для работы с электронными компонентами.....	18
3.1 Выбор платформы реализации мобильного приложения.....	18
3.2 Выбор средств разработки.....	22
3.3 Проектирование модели данных.....	28
3.4 Архитектура программного продукта.....	31
4 Реализация мобильного приложения.....	36
4.1 Краткое описание разработанного решения.....	36
4.2 Реализация аутентификации и авторизации.....	42
4.3 Реализация личного кабинета	47
4.4 Тестирование мобильного приложения.....	55
5 Оценка затрат на разработку мобильного приложения.....	57
5.1 Расчет трудоемкости создания ПО.....	57
5.2 Расчет заработной платы по созданию программного продукта, содержанию и эксплуатации ПЭВМ.....	60
5.3 Расчет затрат на разработку программного средства.....	64

5.4 Расчет себестоимости и прибыли.....	68
6 Безопасность жизнедеятельности.....	73
Заключение.....	78
Список использованных источников.....	80
Приложение А – UML-диаграммы	82
Приложение Б – Листинг программы	90
Приложение В – Презентация.....	99

Перечень сокращений и обозначений

В настоящей работе применяются следующие сокращения и обозначения

БД – база данных

ИТ – информационные технологии

ПО – программное обеспечение

ПК – персональный компьютер

СУБД – система управления базами данных

ЭВМ – электронная вычислительная машина

ID (Identifier) – идентификатор сущности

PK (Primary Key) – первичный ключ

SQL (Structured Query Language) – язык структурированных запросов

UML (Unified Modeling Language) – унифицированный язык моделирования

					ПГУ 1.090501.014.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

Актуальность темы. Согласно статистике, количество пользователей мобильных устройств растет с каждым годом, что открывает новые возможности для бизнеса и разработчиков программного обеспечения. Мобильные приложения становятся неотъемлемой частью жизни людей, предоставляя им доступ к информации и услугам в любое время и в любом месте. В этой связи создание приложения для поиска электронных компонентов не только упростит процесс их поиска, но и сделает его более удобным и доступным для пользователей. Приложение сможет объединить в себе функции, которые помогут как профессиональным разработчикам, так и любителям, занимающимся сборкой и ремонтом электроники.

В настоящее время из-за санкций часть мобильных приложений с базами электронных компонентов стали недоступны в использовании для россиян. Поэтому разработка российского мобильного приложения является особенно актуальным.

Анализ рынка электронных компонентов показывает, что существует множество поставщиков, предлагающих широкий ассортимент товаров. Однако, несмотря на наличие большого количества ресурсов, пользователям зачастую сложно ориентироваться в этом многообразии. Они сталкиваются с проблемами поиска нужных компонентов, сопоставления цен, оценки качества и надежности поставщиков. В этом контексте мобильное приложение может стать эффективным инструментом, который упростит процесс поиска и выбора компонентов, предоставляя пользователям актуальную информацию о наличии, ценах и характеристиках товаров.

Техническое задание и функциональные требования к мобильному приложению должны учитывать потребности целевой аудитории. Важно определить, какие функции будут наиболее полезными для пользователей, а также как обеспечить удобный интерфейс и высокую скорость работы приложения. В процессе разработки необходимо учитывать различные аспекты,

					ПГУ 1.090501.014.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

такие как интеграция с базами данных поставщиков, возможность фильтрации и сортировки товаров, а также наличие системы отзывов и рейтингов, что позволит пользователям принимать обоснованные решения при выборе компонентов.

Таким образом, создание мобильного приложения для поиска электронных компонентов представляет собой актуальную задачу, которая может значительно упростить жизнь пользователям, работающим с электроникой.

Объектом работы является автоматизация поиска электронных компонентов.

Предметом дипломной работы выступает приложение для работы с электронными компонентами.

Целью работы является разработка мобильного Android приложения «База электронных компонентов».

Задачи исследования:

- проанализировать предметную область;
- разработать концепцию бизнес-процессов внутри приложения;
- создать макет мобильного приложения;
- спроектировать базу данных для приложения;
- разработать мобильное приложение и провести его тестирование;
- оценить эффективность разработки.

1 Анализ предметной области

1.1 Описание предметной области

Разработка мобильного приложения для поиска электронных компонентов необходима для мгновенного доступа к базе электронных компонентов, упрощения работы инженеров, сравнения характеристик и цен. Приложение обеспечивает автоматизацию поиска, позволяет добавлять необходимые компоненты в базу и предоставляет инженерам доступ к базе данных в реальном времени, что критично для оперативной работы.

Ключевые причины разработки:

- мобильность и скорость для быстрого поиска компонентов и доступа к даташитам (техническим паспортам) прямо с рабочего места или в полевых условиях.
- агрегация данных для объединения предложений от множества поставщиков в одном интерфейсе, для сравнения цен и наличия.
- автоматизация закупок для оптимизация процессов поиска, заказа и управления спецификациями (BOM), что особенно важно для опытных инженеров и снабженцев.
- функциональность обуславливается проектированием приложений и включает создание удобных интерфейсов, работу с базами данных и обеспечением высокой производительности.

Разработка такого приложения на платформе Android позволяет оперативно обрабатывать запросы пользователей.

Мобильное приложение для поиска электронных компонентов разрабатываются преимущественно для инженеров, менеджеров по закупкам (снабженцев) и технических специалистов.

Ассортимент в мобильных приложениях для поиска электронных компонентов охватывает миллионы позиций: от пассивных элементов (резисторы, конденсаторы) до активных полупроводников (транзисторы, микросхемы, датчики).

					ПГУ 1.090501.014.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

Основные категории ассортимента:

- активные компоненты: Транзисторы (MOSFET, IGBT, биполярные), диоды (Шоттки, TVS, выпрямительные), микросхемы (логика, память, микроконтроллеры), светодиоды, линейные стабилизаторы.
- пассивные компоненты: Резисторы, конденсаторы (керамические, электролитические), катушки индуктивности, трансформаторы.
- компоненты для ремонта SMD-коды, маркировка корпусов (поиск по корпусам TO-92, SOT-23 и т.д.).
- специализированные компоненты: Датчики, звуковая индикация, оптоэлектроника.

В настоящее время существует множество приложений, которые предлагают различные функции для поиска и заказа электронных компонентов. Тем не менее, многие из них имеют свои ограничения, такие как неудобный интерфейс, недостаточный объем информации о компонентах, отсутствие интеграции с поставщиками или высокие цены.

Важно, чтобы приложение не только удовлетворяло потребности пользователей, но и предлагало уникальные функции, которые отличали бы его от существующих решений. К примеру, приложение может включать в себя возможность фильтрации компонентов по различным параметрам, таким как цена, производитель, характеристики и наличие на складе. Также будет полезно реализовать функцию сравнения различных компонентов, что позволит пользователям принимать более обоснованные решения при выборе.

Мобильные приложения, предназначенные для поиска электронных компонентов, становятся все более популярными, предоставляя пользователям возможность быстро и удобно находить необходимые детали, сравнивать цены и характеристики, а также получать информацию о наличии компонентов на складах различных поставщиков. В этом разделе мы проведем анализ существующих решений на рынке, рассмотрим популярные мобильные приложения и их функциональные возможности, а также выявим ключевые аспекты, которые делают их успешными.

Анализ существующих решений на рынке показывает, что мобильные приложения для поиска электронных компонентов становятся все более популярными и востребованными. Успех этих приложений зависит от множества факторов, включая удобство интерфейса, доступность информации о компонентах, возможность интеграции с другими платформами и приложениями, а также наличие уникальных функций, которые делают процесс поиска более удобным и эффективным.

1.2 Выбор CASE-средств для описания бизнес-процессов

Разработка приложения для поиска электронных компонентов включает: анализ рынка (аналоги), проектирование UI/UX (удобные фильтры, база данных), бэкенд-разработку (интеграция с API дистрибьюторов), тестирование (точность поиска, скорость), публикацию и поддержку. Ключевой процесс - синхронизация данных о наличии и цене.

Выбор CASE-средства для описания бизнес-процессов является важным этапом при разработке мобильных приложений. CASE-средства предоставляют возможность описать бизнес-процессы, построить их модели и произвести анализ на основе полученных данных.

Существует множество CASE-средств, которые могут быть использованы для описания бизнес-процессов, таких как Bizagi, Lucidchart, Visual Paradigm, OpenText AppWorks, Diagrams.net и другие. Рассмотрим данные CASE-средства подробнее:

Bizagi Studio - CASE-средство, которое позволяет создавать модели бизнес-процессов и автоматизировать их выполнение. Bizagi Studio поддерживает стандарт BPMN 2.0, что позволяет использовать ее для моделирования сложных процессов и анализа их эффективности.

Visual Paradigm - CASE-средство, которое позволяет разрабатывать модели бизнес-процессов, анализировать их истребования и на основе этого создавать программное обеспечение. Visual Paradigm обладает простым

					ПГУ 1.090501.014.001 ПЗ	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

интерфейсом и поддерживает различные стандарты для моделирования бизнес-процессов.

OpenText AppWorks - это интегрированная платформа для создания, управления и автоматизации бизнес-процессов. Она использует стандарт BPMN 2.0 и включает возможности для автоматизации процессов и управления ими.

Diagrams.net – это онлайн-инструмент для создания диаграмм и блок-схем, который позволяет легко создавать наглядные модели бизнес-процессов [2]. Редактор diagrams.net предоставляет более 50 типов диаграмм, включая BPMN 2.0 - стандарт для моделирования бизнес-процессов.

Она имеет широкий набор инструментов для редактирования диаграмм, а также может быть интегрирована с различными облачными сервисами, в том числе с Firebase, используемым в разрабатываемом приложении.

Сравним данные CASE-средства по следующим критериям:

Функциональность: Все перечисленные сервисы предоставляют широкий набор инструментов для создания различных типов диаграмм. Lucidchart и Visual Paradigm, например, предоставляют более широкий спектр инструментов, чем Bizagi, но менее, чем Diagrams.net. OpenText AppWorks имеет узкую специализацию на BPMN диаграммах.

Интерфейс и удобство использования: все сервисы обладают простым и интуитивно понятным интерфейсом, который позволяет быстро создавать и редактировать диаграммы. Diagrams.net, благодаря своей простоте, позволяет более быстро начать работу с сервисом.

Интеграция с другими приложениями: Lucidchart и Visual Paradigm предоставляют более широкий спектр интеграций с другими приложениями, чем Bizagi и Diagrams.net. OpenText AppWorks также интегрируется с другими приложениями, но в основном с фокусом на BPMN.

Стоимость: Diagrams.net является бесплатным сервисом с открытым исходным кодом, в то время как Lucidchart и Visual Paradigm предлагают платные подписки, предоставляющие расширенные функциональные

					ПГУ 1.090501.014.001 ПЗ	Лист
						11
Изм.	Лист	№ докум.	Подпись	Дата		

возможности. Bizagi и OpenText AppWorks доступны только по запросу, что может усложнить процесс получения доступа.

Возможности совместной работы: все перечисленные сервисы обеспечивают возможность совместной работы над диаграммами, однако Lucidchart, Diagrams.net и Visual Paradigm предоставляют более удобные и гибкие инструменты для совместной работы.

Все вышеперечисленные свойства Программного обеспечения для наглядности были представлены в виде таблицы 1.

Таблица 1 – Свойства Программного обеспечения

Свойства	Bizagi	Lucidchart	Visual Paradigm	OpenText AppWorks	Diagrams.net
Функциональность	-	+	+	-	+
Интерфейс и удобство использования	+	+	+	+	+
Интеграция с другими приложениями	+	+	+	+	+
Стоимость	-	-	-	-	+
Возможности совместной работы	+	+	+	+	+

Для разработки мобильного приложения для работы с электронными компонентами использование diagrams.net будет наиболее эффективным и удобным решением.

1.3 Обзор и анализ аналогов мобильных приложений для для работы с электронными компонентами

Рынок мобильных приложений для работы с электронными компонентами представлен тремя основными категориями: справочники-

калькуляторы, симуляторы схем и комплексные инженерные пакеты. Ниже приведена сравнительная таблица актуальных аналогов.

Таблица 2 - Сравнительный анализ мобильных приложений для работы с электронными компонентами

Название приложения	Категория	Ключевые возможности	Платформы	Плюсы
Electrodoc (ранее ElectroDroid)	Справочник-калькулятор	Маркировка резисторов (3-6 полос) и SMD, расчеты закона Ома, фильтров, делителей напряжения, цоколевки разъемов (USB, HDMI и др.).	Android, iOS	Огромная база справочных данных, поддержка Arduino pin-outs, работа оффлайн.
EveryCircuit	Симулятор схем	Анимированная визуализация токов и напряжений в реальном времени, интерактивная настройка параметров компонентов во время работы схемы.	Android, iOS, Web	Наглядность процессов, большое сообщество с библиотекой готовых пользовательских схем.
Electronics Toolbox	Справочник-калькулятор	Калькуляторы цепей, конвертеры единиц, расчеты падения напряжения и ширины дорожек печатных плат.	Android, iOS	Широкий охват инженерных расчетов, включая ВЧ и расчеты трансформаторов.
iCircuit	Симулятор схем	Полноценный редактор и симулятор с поддержкой аналоговых и цифровых компонентов, осциллографом и мультиметром.	iOS, Android, Windows	Профессиональный уровень симуляции, возможность «горячего» изменения схемы без остановки симулятора.
Droid Tesla	SPICE-симулятор	Расширенное моделирование на базе SPICE-алгоритмов: транзисторы, операционные усилители, логические элементы.	Android	Точность расчетов, подходящая для профессиональных инженеров и студентов профильных вузов.

Лидером остается Electrodoc благодаря обширным справочным таблицам (AWG, таблицы сопротивлений материалов, цоколевки почти всех известных разъемов).

EveryCircuit выигрывает за счет динамической анимации - пользователь буквально «видит», как течет ток, что помогает быстро отладить концепцию.

Если важна точность параметров компонентов и снятие характеристик (графики переходных процессов), лучше использовать Droid Tesla или iCircuit.

					ПГУ 1.090501.014.001 ПЗ	Лист
						14
Изм.	Лист	№ докум.	Подпись	Дата		

2 Техническое задание

2.1 Разработка требований к мобильному приложению для работы с электронными компонентами

На основе проведенного анализа предметной области и аналогов мобильных приложений для работы с электронными компонентами с учётом классификации требований FURPS, были сформулированы следующие требования к разрабатываемому мобильному приложению:

- каталог компонентов для поиска по параметрам, отображению и техническим характеристикам;
- создание заявки на добавление необходимого компонента;
- синхронизация данных пользователя между устройствами.

Технические и качественные требования:

- оффлайн-режим для возможности просмотра скачанных даташитов и поиска по локальной базе без интернета;
- UI/UX минималистичный дизайн, позволяющий быстро находить информацию в полевых условиях.

Завершающим этапом разработки требований к мобильному приложению будет написание основных сценариев использования [16]. В рамках разработки мобильных приложений технику use – case применяют для описания взаимодействия приложения и пользователя, которое в дальнейшем поможет при непосредственной разработке и тестировании программы.

Таблица 3 - Участники взаимодействия описываемого use – case'a.

Система	Мобильное приложения поиска электронных компонентов
Основное действующее лицо	Пользователь
Цель	Найти желаемый компонент

Система	Мобильное приложения поиска электронных компонентов
Результат	Результаты поиска электронных компонентов

Таблица 4 - Основной поток событий описываемого use – case'a.

№ шага	Действующее лицо	Шаг	Комментарий
1	Приложение	Запрашивает фильтры для осуществления поиска	
2	Пользователь	Вводит желаемые фильтры	Фильтрами могут быть любые ключевые слова
3	Приложение	Делает поиск по полученным фильтрам	
4	Приложение	В случае успешного поиска и наличия подходящих компонентов переходит на новый экран и отображает пользователю полученный результат . В противном случае сообщает пользователю об отсутствии результатов поиска и предлагает добавить в базу необходимый компонент.	

№ шага	Действующее лицо	Шаг	Комментарий
5	Пользователь	В случае успешного отображения результатов поиска выбирает желаемый компонент	

2.2 Постановка задачи на разработку мобильного также приложения для работы с android компонентами

Целью является разработка мобильного приложения для инженеров, обеспечивающего быстрый доступ к справочным данным, расчету компонентов (резисторы, конденсаторы) и поиску аналогов. Приложение должно включать базу данных компонентов и возможность экспорта данных, повышая эффективность проектирования.

Основные элементы постановки задачи:

- целью является создание инструмента для удобной идентификации, поиска электронных компонентов (ЭК);
- к целевой аудитории относятся радиоинженеры, студенты технических вузов, радиолюбители;

Функциональные требования:

- поиск , маркировке, типу корпуса (резисторы, диоды, транзисторы, микросхемы);
- локальное хранение данных о наличии компонентов у пользователя (инвентаризация).

3 Проектирование мобильного приложения для работы с электронными компонентами

3.1 Выбор платформы реализации мобильного приложения

При выборе мобильной операционной системы для приложения для работы с электронными компонентами, необходимо учитывать ряд факторов, таких как функциональность, удобство использования, безопасность, стабильность, популярность и доступность.

Пользователи в настоящее время имеют широкий выбор мобильных операционных систем, предоставляющих доступ к различным сервисам и функциональности. Ниже приведены наиболее популярные мобильные операционные системы:

– Android - это операционная система для мобильных устройств, разработанная Google. Android является самой популярной мобильной платформой в мире, с долей рынка 68,61% на конец апреля 2025 года. Она предлагает широкий выбор устройств и возможностей для разработчиков [7];

– iOS - это мобильная операционная система, разработанная Apple для своих устройств, таких как iPhone и iPad [5]. Она имеет долю рынка около 30,61% на конец апреля 2025 года. iOS обеспечивает безопасность и надежность, а также предлагает богатый пользовательский опыт;

– Windows Phone - это операционная система для мобильных устройств, разработанная Microsoft. Она имеет долю рынка около 0,2% на конец апреля 2025 года. Windows Phone предлагает удобный интерфейс и интеграцию с другими продуктами Microsoft;

– BlackBerry OS - это операционная система для мобильных устройств, разработанная BlackBerry. Она имеет долю рынка около 0,1% на конец апреля 2025 года. BlackBerry OS предлагает высокий уровень безопасности и надежности.

Сравним данные мобильные операционные системы по следующим критериям: функциональность, удобство использования, безопасность, стабильность, популярность и доступность [13].

Функциональность:

– Android: предлагает обширный набор функций и возможностей, позволяющих разработчикам создавать разнообразные приложения и настраивать интерфейс под свои нужды;

– iOS: также предлагает широкий набор функциональных возможностей, но более строго контролирует доступ разработчиков к некоторым системным функциям и инструментам;

– Windows Phone: обладает функциональностью, сравнимой с Android и iOS, но может быть немного ограничен в некоторых аспектах, таких как доступность приложений;

– BlackBerry OS: имеет свои уникальные функции, такие как высокая безопасность и интеграция с корпоративными сервисами, но может быть менее разнообразным в функциональном плане.

Удобство использования:

– Android: Интерфейс интуитивно понятен и привычен для пользователей, особенно для тех, кто привык к использованию продуктов Google;

– iOS: имеет простой и интуитивно понятный пользовательский интерфейс, что делает его легким в использовании для новых пользователей. Он предлагает стандартизированное взаимодействие и дизайн приложений;

– Windows Phone: Windows Phone также имеет простой и стильный пользовательский интерфейс, но может отличаться от привычного пользовательского опыта в Android и iOS;

– BlackBerry OS: имеет свою собственную систему навигации и управления, которая может быть несколько отличной от других операционных систем.

Безопасность:

					ПГУ 1.090501.014.001 ПЗ	Лист
						19
Изм.	Лист	№ докум.	Подпись	Дата		

– Android имеет мощные механизмы безопасности, включая систему разрешений, обновления безопасности и встроенную защиту от вредоносного ПО;

– iOS известен своей высокой безопасностью, благодаря строгим правилам и контролю со стороны Apple над приложениями и доступом к системным ресурсам;

– Windows Phone также предлагает некоторые механизмы безопасности, но из-за низкой популярности и ограниченной доступности сторонних приложений, экосистема Windows Phone менее подвержена атакам и вирусам;

BlackBerry OS славится своей высокой безопасностью, которая делает его популярным среди корпоративных пользователей.

Стабильность:

– стабильность Android-устройств может варьироваться в зависимости от производителя и модели. В некоторых случаях на различных устройствах могут возникать проблемы с производительностью или несовместимостью. Однако, с постоянным развитием платформы и улучшением аппаратного и программного обеспечения, стабильность Android-устройств постоянно улучшается;

– iOS известен своей стабильностью и плавной работой на всех устройствах Apple. Благодаря контролируемой экосистеме и оптимизации операционной системы для конкретных устройств, iOS обычно обеспечивает высокую стабильность и производительность;

– Windows Phone также предлагает стабильную работу, особенно на устройствах, разработанных Microsoft. Однако, с учетом того, что Windows Phone больше не активно поддерживается, новые обновления и исправления могут быть ограничены;

– BlackBerry OS изначально разрабатывалась с упором на стабильность и надежность. Эта операционная система обладает репутацией высокой стабильности, особенно в корпоративной среде.

Популярность и доступность:

					ПГУ 1.090501.014.001 ПЗ	Лист
						20
Изм.	Лист	№ докум.	Подпись	Дата		

– Android является наиболее популярной мобильной операционной системой в мире (68,61%). Его широкая распространенность обеспечивает доступность множества устройств различных производителей и различных ценовых категорий;

– iOS также имеет значительную популярность (30,61%), особенно среди пользователей Apple, которые предпочитают экосистему. Однако, доступность iOS ограничена только на устройствах Apple;

– Windows Phone имеет гораздо меньшую популярность по сравнению с Android и iOS (0,1%). Это может отразиться на доступности устройств и приложений;

– BlackBerry OS также имеет ограниченную популярность, особенно в широких массах пользователей (0,2%). Это может повлиять на доступность и поддержку приложений.

Ниже приведена таблица 4 для сравнения вышеперечисленных мобильных операционных систем.

Таблица 4 – Сравнение мобильных операционных систем

	Android	IOS	Windows Phone	BlackBerry OS
Функциональность	+	+	-	-
Удобство использования	+	+	+	+
Безопасность	+	+	+	+
Стабильность	+	+	+	+
Популярность и доступность	+	-	-	-

Сравнение различных операционных систем показывает, что Android является более гибкой и открытой платформой, что позволяет разработчикам более свободно работать с аппаратным обеспечением устройств и более быстро

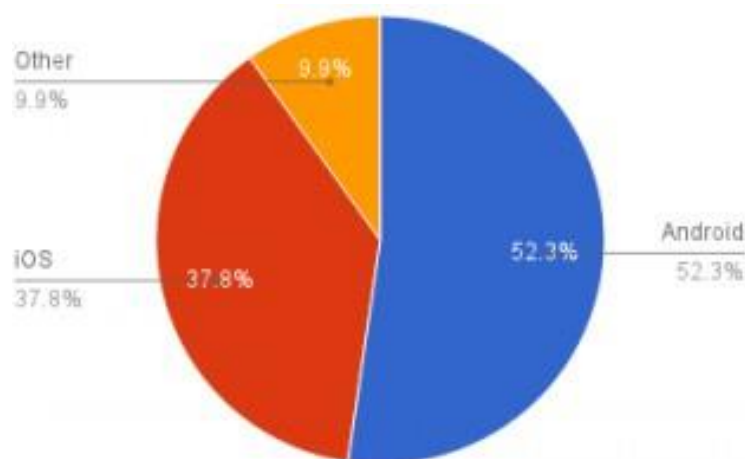
выпускать обновления и исправления ошибок. Кроме того, Android имеет большое количество устройств разных производителей, вследствие привлекает большее количество пользователей.

3.2 Выбор средств разработки

В самом начале разработки мобильного приложения было решено что, проект будет ориентирован на одну из самых распространенных и популярных операционных систем для мобильных устройств – Андроид. Данная операционная система имеет множество бесспорных преимуществ по сравнению с конкурентами.

Самые явные преимущества:

- андроид ОС является самой популярной в мире (рисунок 1)
- андроид ОС с открытым исходным кодом
- программное обеспечение для разработки мобильных приложений бесплатно и доступно на всех операционных системах



1 – Доля устройств на различных ОС для мобильных устройств

Одним из важнейших элементов разработки любого приложения является выбор интегрированной среды разработки – IDE. Для создания приложений на платформе Андроид подходят такие среды разработки как, Eclipse, NetBeans, Android studio. Для работы был выбран последний вариант. Android studio

является универсальным IDE от компании Google, так как позволяет создавать и протестировать приложения, которые будут работать на смартфонах, планшетах, портативных ПК, часах, которые работают на основе Андроид ОС. Данная среда разработки является по настоящему удобной по нескольким причинам:

- удобный редактор кода, который помогает в написании кода и предлагает завершение, рефракцию и анализ кода;
- наличие большой библиотеки готовых шаблонов и компонентов для разработки ПО;
- возможность предварительной проверки ПО на ошибки;
- встроенный эмулятор Android (рисунок 2).

Эмулятор Android - это виртуальное устройство, которое может представлять любое устройство, работающее на платформе Android. Используется эмулятор Android в качестве целевой платформы для запуска и тестирования мобильных приложений. Эмулятор обладает удобной отличительной особенностью – просмотр показателей производительности при запуске приложения (рисунок 3).

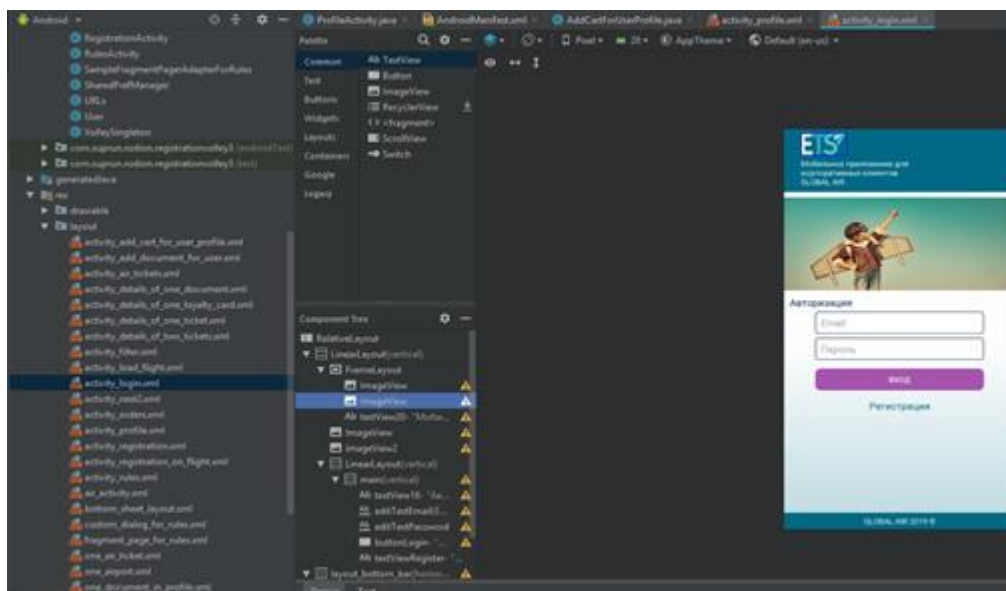


Рисунок 2 – Android studio

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.014.001 ПЗ

Лист

23



3 – Эмулятор в Android studio

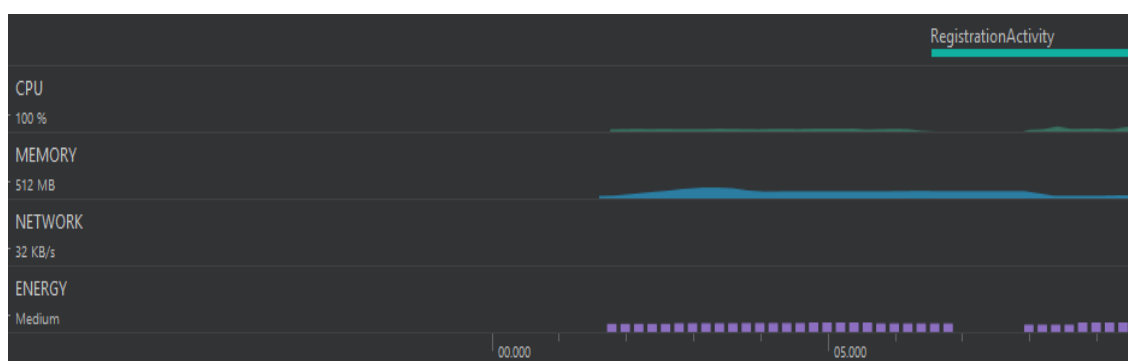


Рисунок 4 – Показатели производительности в Android studio

Android Studio был выбран в качестве среды разработки для создания мобильного приложения для работы с электронными компонентами по ряду причин:

– является официальной интегрированной средой разработки (IDE) для платформы Android, разработанной и поддерживаемой компанией Google. Благодаря этому, она обладает широким набором инструментов, ресурсов и обновлений, специально созданных для разработки приложений под Android [1];

– предоставляет простой и интуитивно понятный пользовательский интерфейс, что упрощает работу. Интуитивная навигация, автозаполнение кода и встроенные инструменты позволяют эффективно разрабатывать приложения;

– предлагает широкий спектр инструментов и функций, которые значительно облегчают процесс разработки. Это включает в себя эмуляторы устройств для тестирования приложений, визуальные редакторы пользовательского интерфейса, отладочные инструменты;

– предоставляет полную поддержку языка программирования Java, который является официальным языком разработки приложений под Android. Java является широко используемым языком с огромным сообществом разработчиков, множеством ресурсов и библиотек, что облегчает процесс разработки;

– интегрируется с другими инструментами и службами разработки, такими как Firebase, которая предоставляет широкий набор облачных услуг для разработки мобильных приложений. Это позволяет использовать Firebase для упрощения работы с базой данных, аутентификацией пользователей, обработкой уведомлений и другими функциями, что существенно ускоряет процесс разработки.

Исходя из этих преимуществ, выбор Android Studio является логичным и обоснованным для разработки мобильного приложения для работы с электронными компонентами.

Выбор языка программирования Java для разработки мобильного приложения для работы с электронными компонентами осуществлен по следующим причинам [9]:

– Android Studio, основная интегрированная среда разработки для платформы Android, обладает прекрасной поддержкой Java. Инструменты, редакторы и отладчики Android Studio предоставляют широкие возможности для разработки приложений на языке Java и облегчают процесс разработки;

– уже имеется опыт работы с языком программирования Java по учебному курсу университета. Это упрощает процесс разработки, позволяет использовать уже накопленный опыт и ускоряет процесс освоения различных Android-специфичных инструментов и фреймворков;

– имеет обширную базу ресурсов и документации, включая официальную документацию от Oracle, множество книг, учебных материалов и онлайн-ресурсов. Это обеспечивает легкий доступ к информации и помощи при разработке приложения;

– является одним из самых надежных и стабильных языков программирования, с многолетней историей применения в широком спектре проектов. Это обеспечивает стабильность и надежность разработанного приложения[12].

В целом, выбор Java в данном случае обоснован опытом, доступностью ресурсов и поддержкой Android-платформы, что обеспечит более комфортное и эффективное разработку мобильного приложения для работы с электронными компонентами компании.

Firebase был выбран в качестве инструмента разработки для мобильного приложения для работы с электронными компонентами по ряду причин [4]:

– предоставляет широкий набор облачных услуг, которые значительно упрощают разработку мобильных приложений. Он предлагает готовые решения для различных функций, таких как аутентификация пользователей, база данных в реальном времени, хостинг, облачные функции, хранение файлов, уведомления и многое другое. Благодаря этому, Firebase позволяет сосредоточиться на основной функциональности приложения, минимизируя необходимость в разработке и настройке собственных серверных компонентов;

– легко интегрируется с платформой Android и предоставляет удобные инструменты для быстрой и безболезненной интеграции с мобильными приложениями. Firebase SDK (Software Development Kit) предоставляет готовые API и компоненты, которые позволяют взаимодействовать с облачными услугами Firebase;

– предоставляет мощную базу данных в реальном времени, которая автоматически синхронизируется с клиентскими приложениями. Это позволяет создавать приложения, которые мгновенно реагируют на изменения данных и обеспечивают синхронизацию между разными устройствами и платформами;

– обеспечивает высокую масштабируемость и надежность своих облачных услуг. Использование Firebase позволяет избежать заботы о серверной инфраструктуре, масштабировании и обслуживании, поскольку все это обеспечивается Firebase;

– имеет большое количество документации, учебных ресурсов, сообществ и форумов, что позволяет достаточно легко научиться работать в данной системе.

В целом, выбор Firebase для разработки мобильного приложения для работы с электронными компонентами обусловлен его широкими возможностями, простотой интеграции, надежностью.

OneSignal был выбран для отправки уведомлений в приложении по нескольким причинам [8]:

– предоставляет простой и удобный способ интеграции сервиса уведомлений в мобильное приложение. Он предоставляет набор готовых библиотек и SDK для различных платформ, включая Android, что значительно упрощает процесс интеграции в приложение;

– OneSignal предлагает широкий набор функций для отправки уведомлений. Это включает гибкую настройку внешнего вида уведомлений, возможность отправки уведомлений по геолокации или группам пользователей, поддержку пуш-уведомлений с различными типами контента (текст,

изображения, звуки и т. д.), а также возможность отправки уведомлений на различные платформы (Android, iOS, веб);

– обеспечивает высокую масштабируемость и надежность в доставке уведомлений. Сервис имеет распределенную инфраструктуру и использует различные каналы доставки, чтобы гарантировать, что уведомления будут доставлены на целевые устройства надежно и вовремя.

Исходя из этих причин, OneSignal был выбран в качестве решения для отправки уведомлений в приложении, так как он сочетает в себе простоту интеграции, мощные функциональные возможности, надежность и масштабируемость.

3.3 Проектирование модели данных

В приложении не используется локальная база данных, так как нет необходимости хранить данных на самом устройстве. Вместо этого приложение берет все данные с БД сервера посредством HTTP протокола передачи данных и реализации API интерфейса на веб-сервере. Данное взаимодействие называется клиент - серверной архитектурой (рисунок 5).

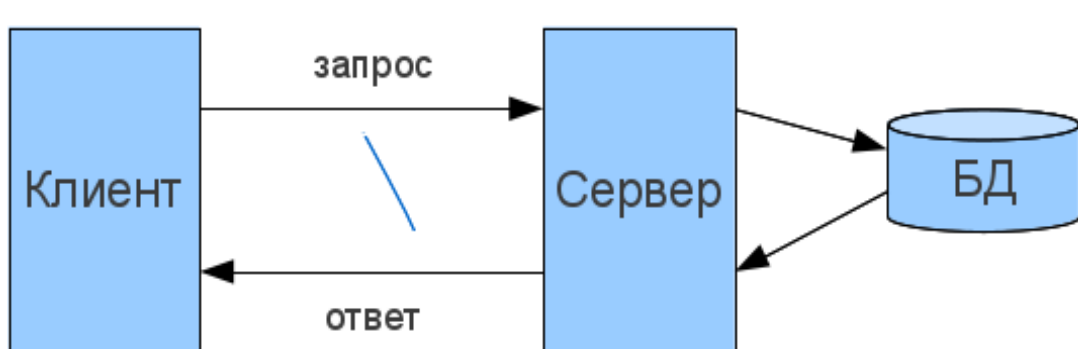


Рисунок 5 – Сетевая архитектура "клиент - сервер"

Сервер постоянно ждет запросы от клиента, и готов в любое время ответить на post или get запрос. Для отправки запросов на сервер в Android используются различные библиотеки такие как Retrofit, OkHttp, Volley.

Для работы была выбрана библиотека Volley, так как обладает множеством преимуществ, и самое главное библиотека легка в понимании и

использовании. Обычно в Android код сетевого вопроса помещался AsyncRask. Надо было позаботиться об отображении ProgressBar внутри функций OnsourceExecute() и OnPostExecute(). Благодаря Volley больше о такой рутине думать разработчикам нет необходимости. Volley предлагает множество функций, как указано в его документации:

- автоматическое планирование сетевых запросов;
- несколько одновременных сетевых подключений;
- Volley предоставляет инструменты отладки;
- поддержка приоритизации запросов;
- простота настройки, например, для повторных попыток и откатов.

Для отправки запросов с помощью библиотеки Volley используется 2 основных класса: Request и Request queue. Класс Request содержит необходимые детали для создания запроса такие как, как выбор типа запроса, URL, данные запроса. В своем приложении использовали StringRequest, который ответ от сервера преобразует в тип данных – строка (рисунок 6). Затем работаем с ответом, вытаскиваем Json – объекты, массивы и тд.

Класс Request queue используется для отправки запросов, в прямом смысле каждый запрос мы помещаем в очередь.

```

StringRequest stringRequest = new StringRequest(
    Request.Method.GET,
    url,
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            // Do something with response string
            mTextView.setText(response);
            requestQueue.stop();
        }
    },
    new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            // Do something when get error
            Snackbar.make(mCLayout, "Error.", Snackbar.LENGTH_LONG).show();
            requestQueue.stop();
        }
    }
);

// Add StringRequest to the RequestQueue
requestQueue.add(stringRequest);

```

Рисунок 6 – Синтаксис запроса с помощью библиотеки Volley

Для отправки уведомлений пользователям, приложение подключили к Firebase, который является поставщиком облачных услуг. Схема работы системы при отправлении уведомления представлена на рисунке 7.

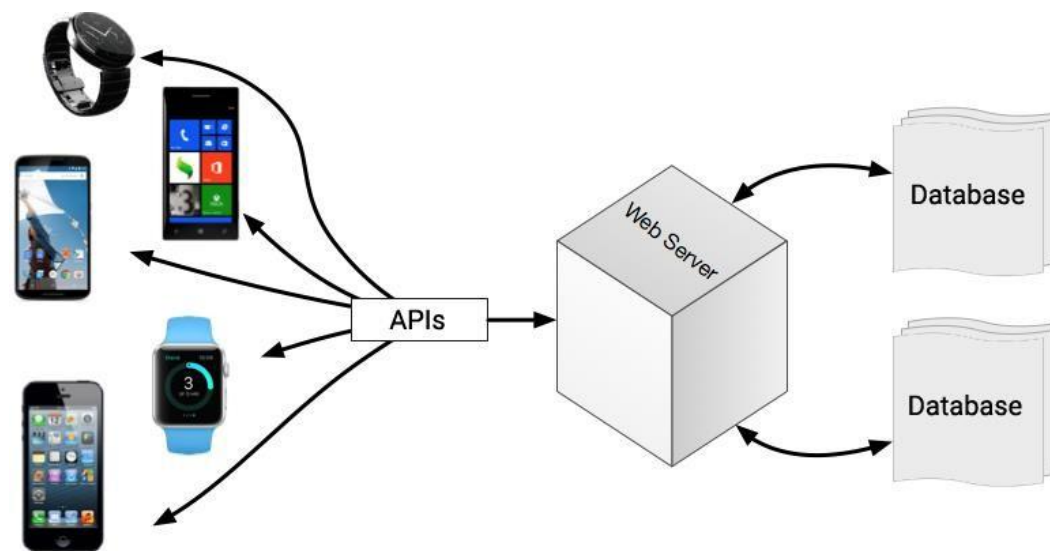


Рисунок 7 - Архитектура системы

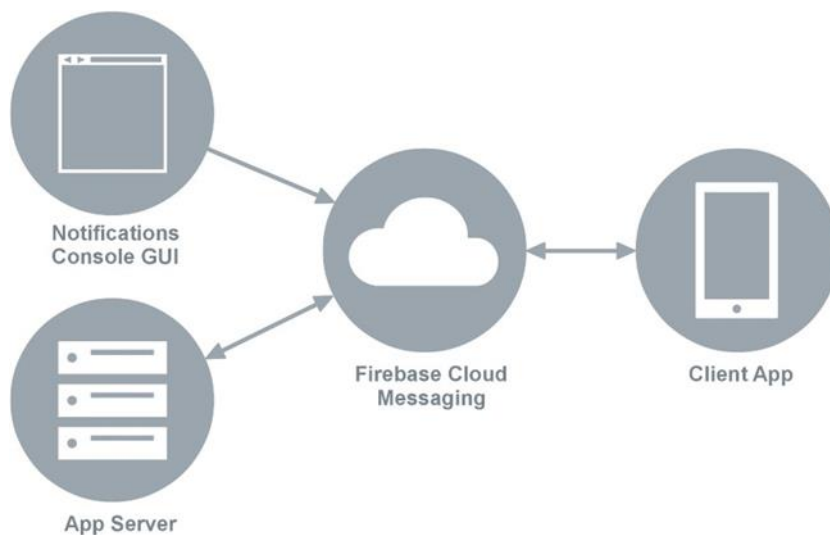


Рисунок 8 - Архитектура системы для отправления уведомлений

3.4 Архитектура программного продукта

Существует несколько видов архитектуры, которые могут быть использованы при разработке мобильных приложений [11]. Наиболее распространённые виды архитектуры мобильных приложений представлены ниже:

- MVC (Model-View-Controller). Архитектура, которая разделяет приложение на три основных компонента: модель (хранение данных и бизнес-логику), представление (отображение пользовательского интерфейса) и контроллер (управление взаимодействием между моделью и представлением). MVC обеспечивает разделение ответственностей и улучшает модульность и переиспользуемость кода;

- MVP (Model-View-Presenter). Архитектура основана на концепции MVC, но с некоторыми изменениями. Основное отличие заключается в том, что Presenter берет на себя большую часть ответственности за управление взаимодействием между моделью и представлением. MVP облегчает тестирование и обеспечивает лучшую разделение логики и представления;

- MVVM (Model-View-ViewModel). Архитектура также основана на концепции MVC, но включает в себя дополнительный компонент - ViewModel. ViewModel служит связующим звеном между моделью и представлением,

предоставляя данные и методы для их обработки. MVVM улучшает разделение интерфейса пользователя и бизнес-логики, а также облегчает тестирование и поддержку кода;

– Clean Architecture. Архитектурный подход, который стремится создать приложение, которое легко поддерживать, тестируемое и независимое от конкретных фреймворков или библиотек. Он использует принципы разделения интерфейсов и зависимостей, разделяя приложение на слои с четкими границами. Clean Architecture облегчает масштабирование приложения и его развитие в долгосрочной перспективе;

– Reactive Architecture. Архитектурный подход, который строится на идее реактивного программирования. Он основан на потоковой обработке данных и реакции на события. Реактивная архитектура позволяет создавать отзывчивые и масштабируемые приложения, которые легко адаптируются к изменяющимся условиям;

– «Основанное на стандартных компонентах Android». Это означает, что код и компоненты приложения могут быть организованы в соответствии с принятыми практиками разработки для платформы Android, но без строгого применения конкретного архитектурного паттерна.

Архитектура разработанного мобильного приложения основана на стандартных компонентах Android. Приложение организовано в соответствии с принятыми практиками разработки для платформы Android, используя предоставляемые Android SDK компоненты и функциональность. В рамках разработки были использованы активности, элементы интерфейса, обработчики событий, а также взаимодействие с Firebase и OneSignal для обеспечения функциональности для работы с электронными компонентами. Организация кода и компонентов приложения позволяет эффективно управлять его функциональностью, взаимодействием с пользователями и внешними сервисами.

В диаграмме развёртывания представлены компоненты, необходимые для функционирования мобильного приложения для работы с электронными компонентами. В развёртывании приложения участвуют следующие элементы:

- пользователи устанавливают мобильное приложение на свои устройства, такие как смартфоны или планшеты, работающие на операционной системе Android. Клиентское приложение предоставляет пользователю возможность создавать и управлять заявками;
- Firebase используется в качестве облачной платформы для хранения данных и обмена информацией между клиентскими устройствами и сервером. Firebase обеспечивает хранение и синхронизацию данных заявок, а также авторизацию пользователей. Он также предоставляет API для взаимодействия с базой данных и облачными функциями;
- в приложении используется интеграция с OneSignal. OneSignal используется для отправки уведомлений о новых заявках или изменении статуса существующих заявок.

Диаграмма развёртывания изображена на рисунке 9.

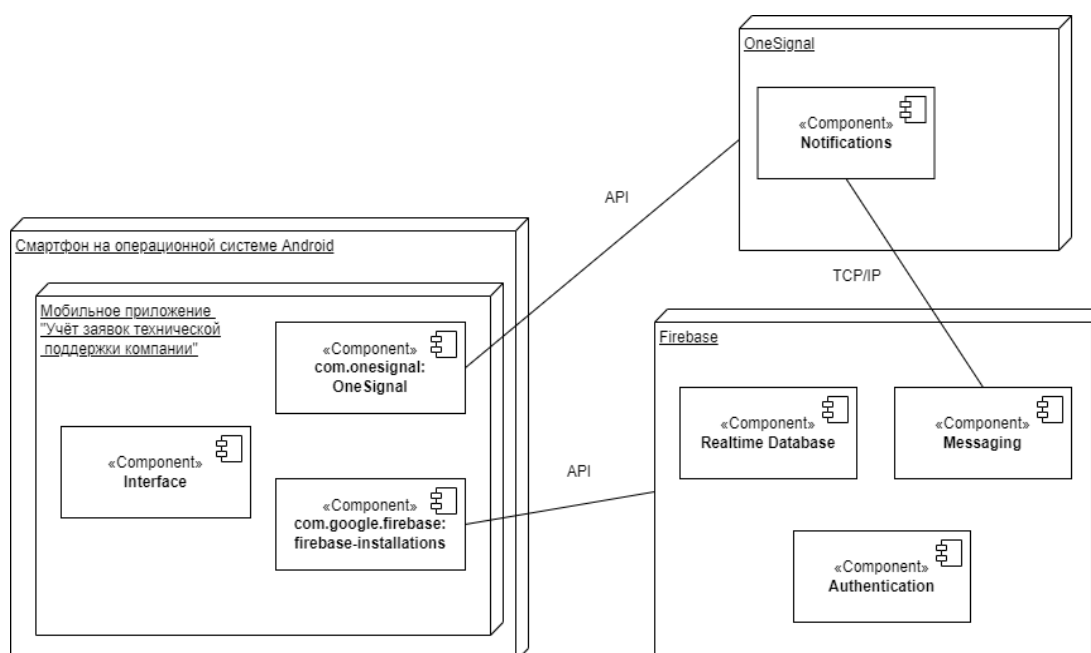


Рисунок 9 – Диаграмма развёртывания

Взаимодействие между этими компонентами позволяет приложению эффективно функционировать. Клиентские устройства обмениваются данными с Firebase, осуществляют создание и обновление статуса заявок. Firebase сохраняет и синхронизирует данные заявок между клиентскими устройствами и сервером. При необходимости OneSignal отправляет уведомления о новой заявке или об изменении статуса заявки.

Таким образом, диаграмма развёртывания показывает, как клиентские устройства взаимодействуют с Firebase и OneSignal, образуя основу для функционирования мобильного приложения.

компонентов используются для визуализации организации компонентов системы и зависимостей между ними. Они позволяют получить высокоуровневое представление о компонентах системы [10].

Диаграмма компонентов представлена на рисунке 10.

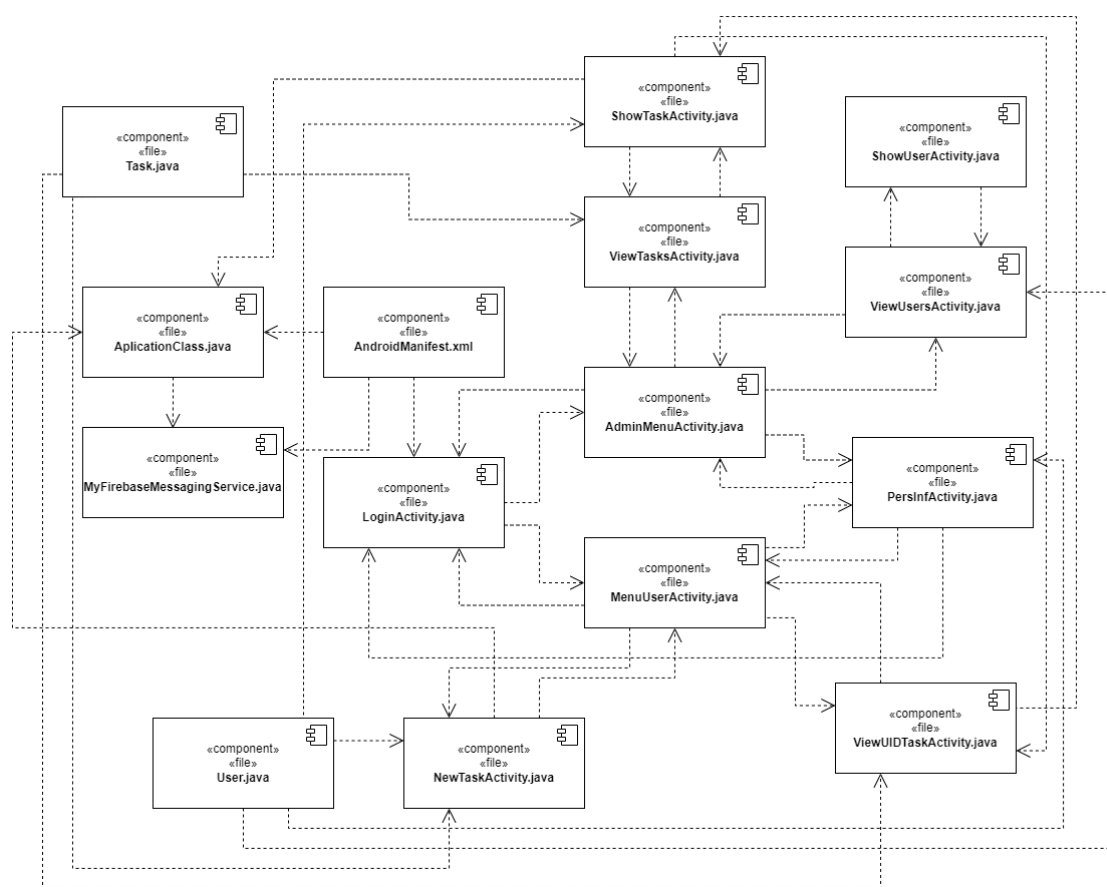


Рисунок 10 – Диаграмма компонентов

Взаимодействие между этими компонентами обеспечивает полноценное функционирование приложения. Диаграммы компонентов и развёртывания позволяют наглядно представить архитектурную структуру приложения, выделяя основные компоненты и их взаимосвязи.

В рамках главы был проведён сравнительный анализ мобильных операционных систем, в ходе которого мобильные операционные системы сравнивались по нескольким критериям. В результате данного анализа был выбран Android в качестве платформы для разработки мобильного приложения для работы с электронными компонентами. Выбор данной платформы был обоснован.

Далее были приведены средства разработки мобильного приложения: Android Studio, Firebase, OneSignal и язык программирования Java. Выбор средств разработки также был обоснован. После была спроектирована модель данных: концептуальная, логическая и физическая. Также были приведены виды архитектуры программного продукта. В итоге была выбрана архитектура разрабатываемого мобильного приложения, основанная на стандартных компонентах Android. В конце главы были представлены диаграмма развёртывания и диаграмма компонентов.

4 Реализация мобильного приложения

4.1 Краткое описание разработанного решения

Разработанное решение представляет собой мобильное приложение для работы с электронными компонентами. Оно разработано на языке программирования Java для платформы Android с использованием Android Studio, Firebase и OneSignal.

Разработанное решение полностью интегрировано с Firebase, платформой облачных сервисов от Google.

Также в приложении используется сервис OneSignal для отправки уведомлений клиентам о статусе и обновлениях их заявок, а сотрудникам технического отдела о появлении новой заявки. OneSignal предоставляет простой способ интеграции и управления уведомлениями, а также гибкие настройки для внешнего вида и содержания уведомлений.

Разработанное решение предоставляет компании эффективный инструмент для работы с электронными компонентами. В качестве акторов и систем используются:

- основной актер;
- интерфейс;
- обработчик логики и база данных;
- офлайн-кэш на устройстве.

Это главная функция приложения. Пользователь вводит параметры (номинал, корпус, производитель) и получает список подходящих компонентов (рисунок 11).

					ПГУ 1.090501.014.001 ПЗ	Лист
						36
Изм.	Лист	№ докум.	Подпись	Дата		

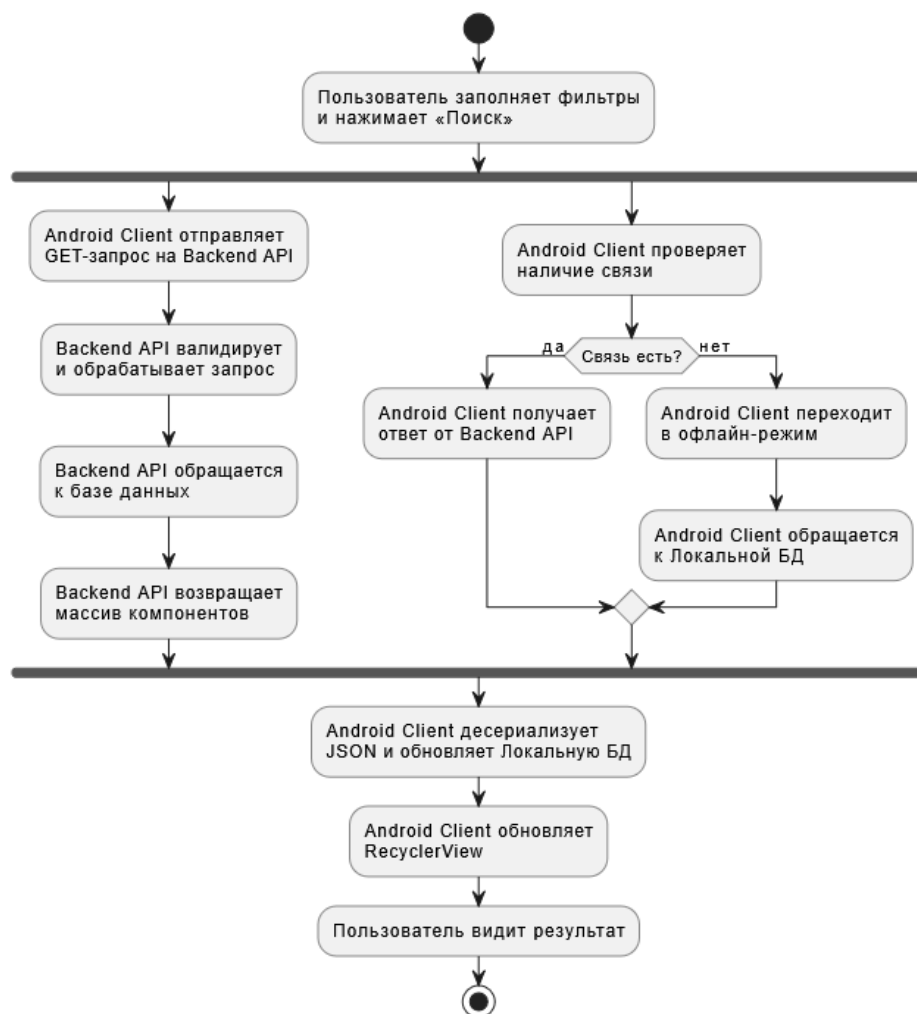


Рисунок 11 –Диаграмма последовательности: поиск компонента по параметрам (Основной сценарий)

Данная диаграмма описывает алгоритм обработки поискового запроса, который начинается с ввода пользователем фильтров и разделяется на два параллельных процесса: отправку GET-запроса к внешнему API и проверку доступности сети. В зависимости от наличия связи система либо получает актуальные данные с сервера, либо переходит в офлайн-режим для обращения к локальной базе данных, после чего результаты десериализуются, кэшируются и отображаются пользователю через графический интерфейс (RecyclerView). Пользователь нажимает на конкретный транзистор или микросхему в списке, чтобы увидеть подробности и открыть документацию (рисунок 12).

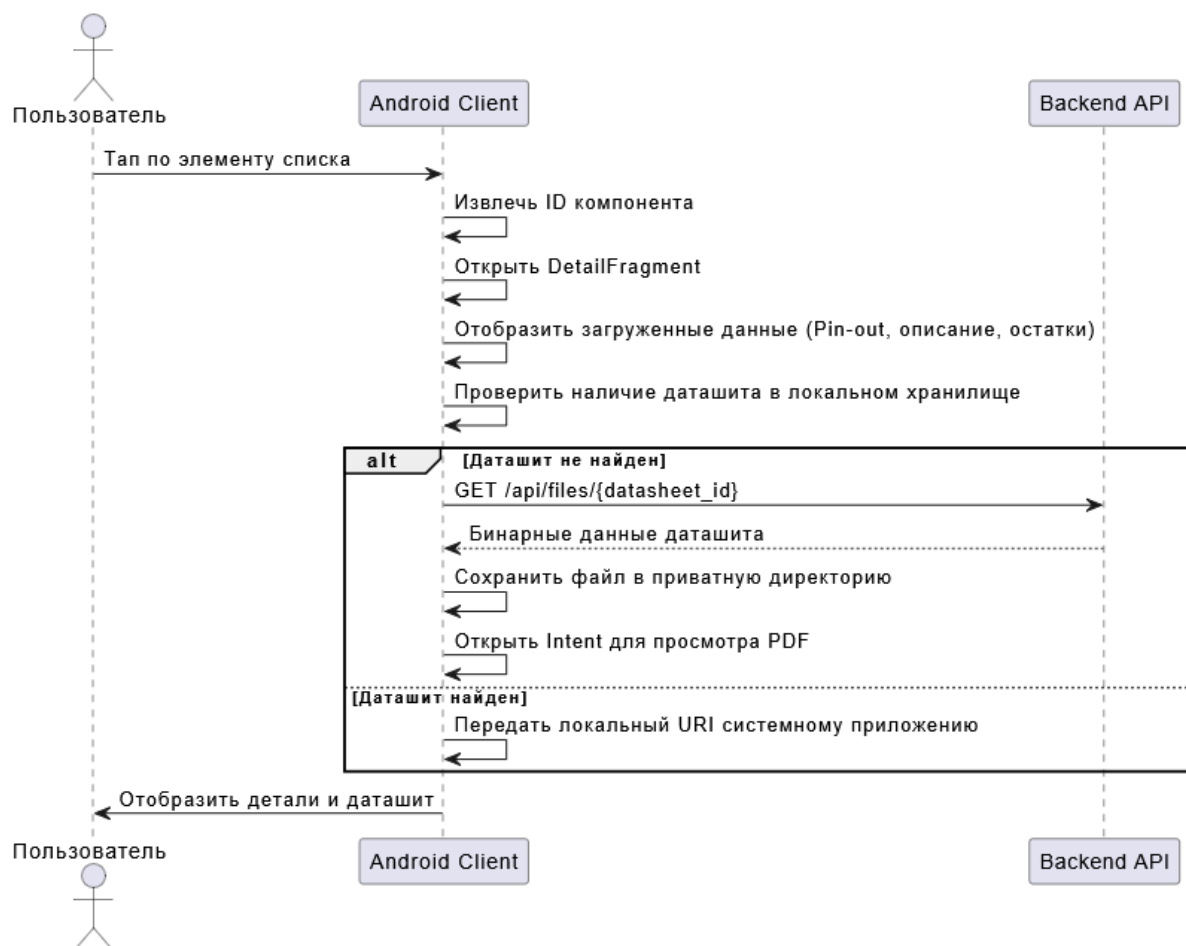


Рисунок 12 –Диаграмма последовательности: просмотр карточки компонента и даташита

Эта диаграмма последовательности визуализирует взаимодействие между пользователем, мобильным клиентом и сервером при открытии детальной информации о детали. При нажатии на элемент списка приложение извлекает ID компонента, открывает соответствующий фрагмент с характеристиками и проверяет наличие технической документации (даташита) в локальной памяти устройства. Если файл отсутствует, клиент отправляет запрос на сервер, скачивает бинарные данные, сохраняет их в приватную директорию и инициирует системное намерение (Intent) для открытия PDF-файла, предоставляя пользователю полный доступ к документации.

Сценарий, когда инженер находится на объекте без интернета и ему срочно нужен аналог компонента (рисунок 13).

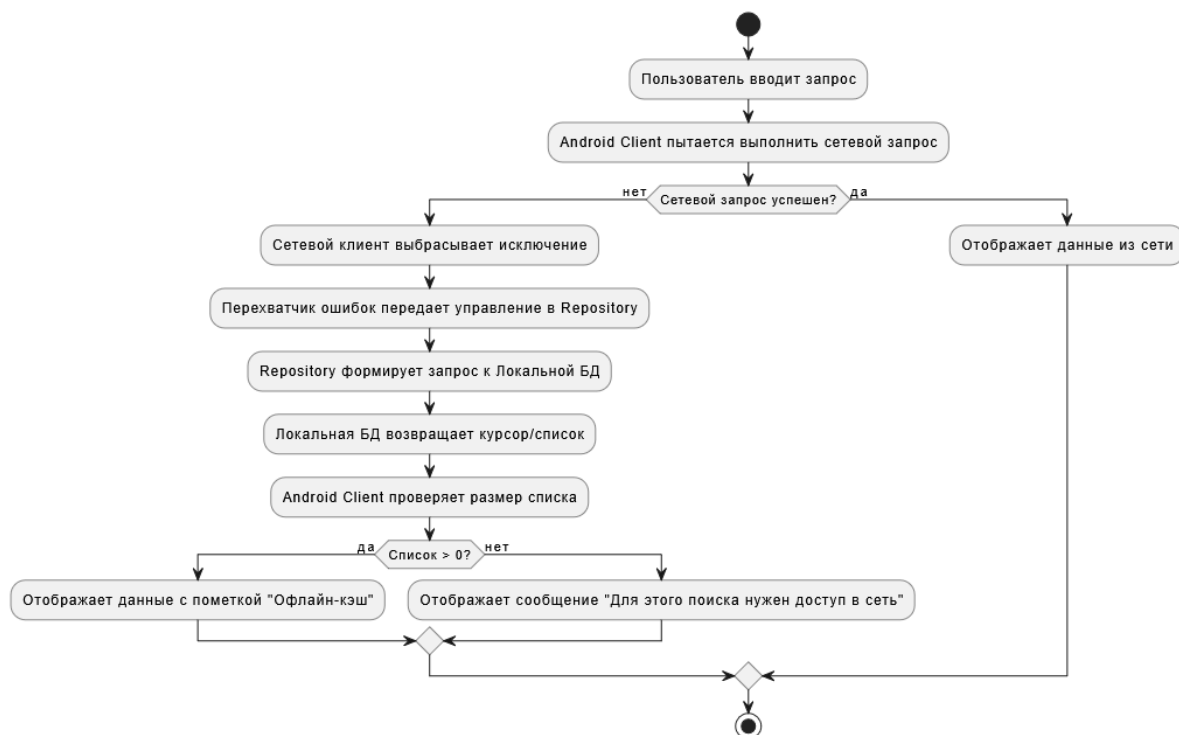


Рисунок 13 –Диаграмма последовательности: «Работа в офлайн-режиме (Fallback)»

Данная диаграмма демонстрирует логику работы приложения в условиях отсутствия интернета, где после неудачной попытки сетевого запроса управление передается перехватчику ошибок и слою Repository. Система выполняет поиск по локальному SQL-хранилищу и, если данные найдены, выводит их с пометкой «Офлайн-кэш», в противном случае уведомляя пользователя о невозможности выполнения операции без подключения к сети.

Пользователь сохраняет часто используемые позиции, чтобы иметь к ним доступ на разных устройствах (рисунок 14).

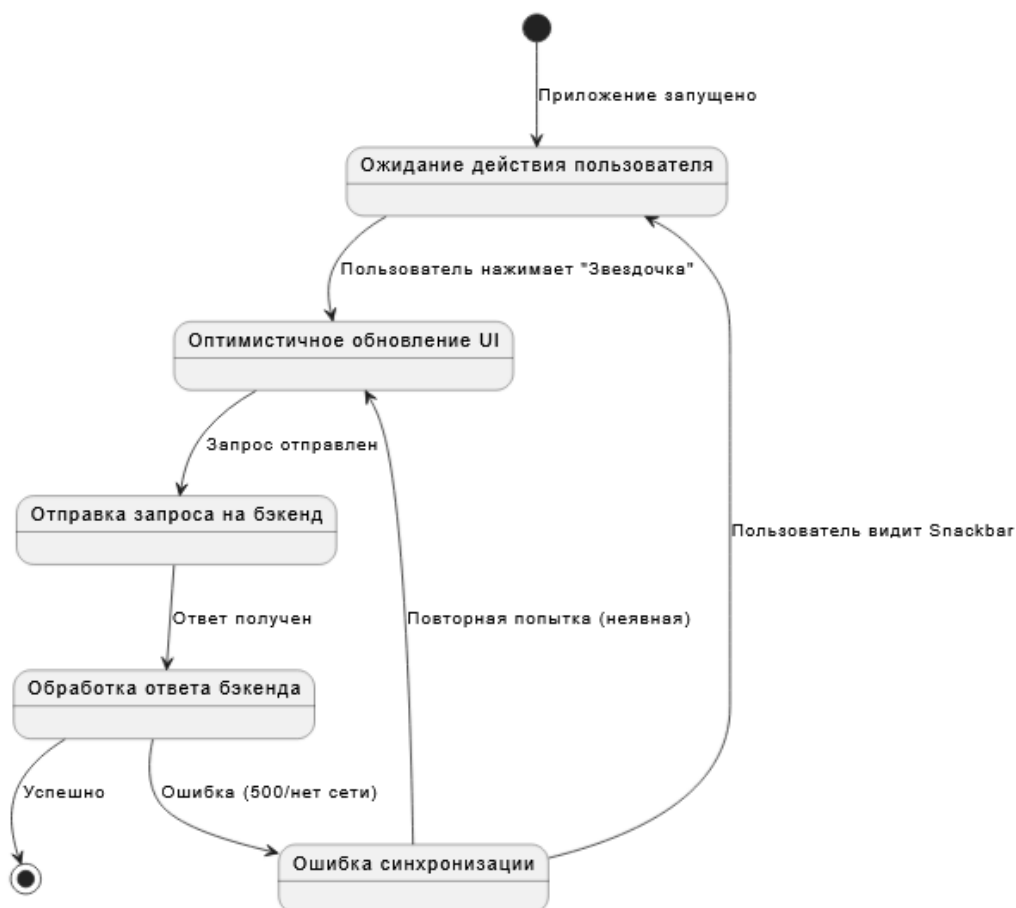


Рисунок 14 –Диаграмма последовательности: добавление компонента в «Избранное» и синхронизация

Данная диаграмма описывает логику синхронизации пользовательских предпочтений, используя механизм «оптимистичного обновления UI», при котором состояние иконки меняется мгновенно (до фактического подтверждения от сервера) для обеспечения высокой отзывчивости интерфейса. В процессе выполнения запроса на бэкенд система переходит в состояние ожидания ответа, и в случае успеха завершает операцию, а при возникновении ошибки (отсутствие сети или сбой 500) уведомляет пользователя через Snackbar и откатывает изменения либо инициирует повторную попытку синхронизации.

Сценарий загрузки обновлений с сервера, например, когда вышли новые серии микросхем (рисунок 15).

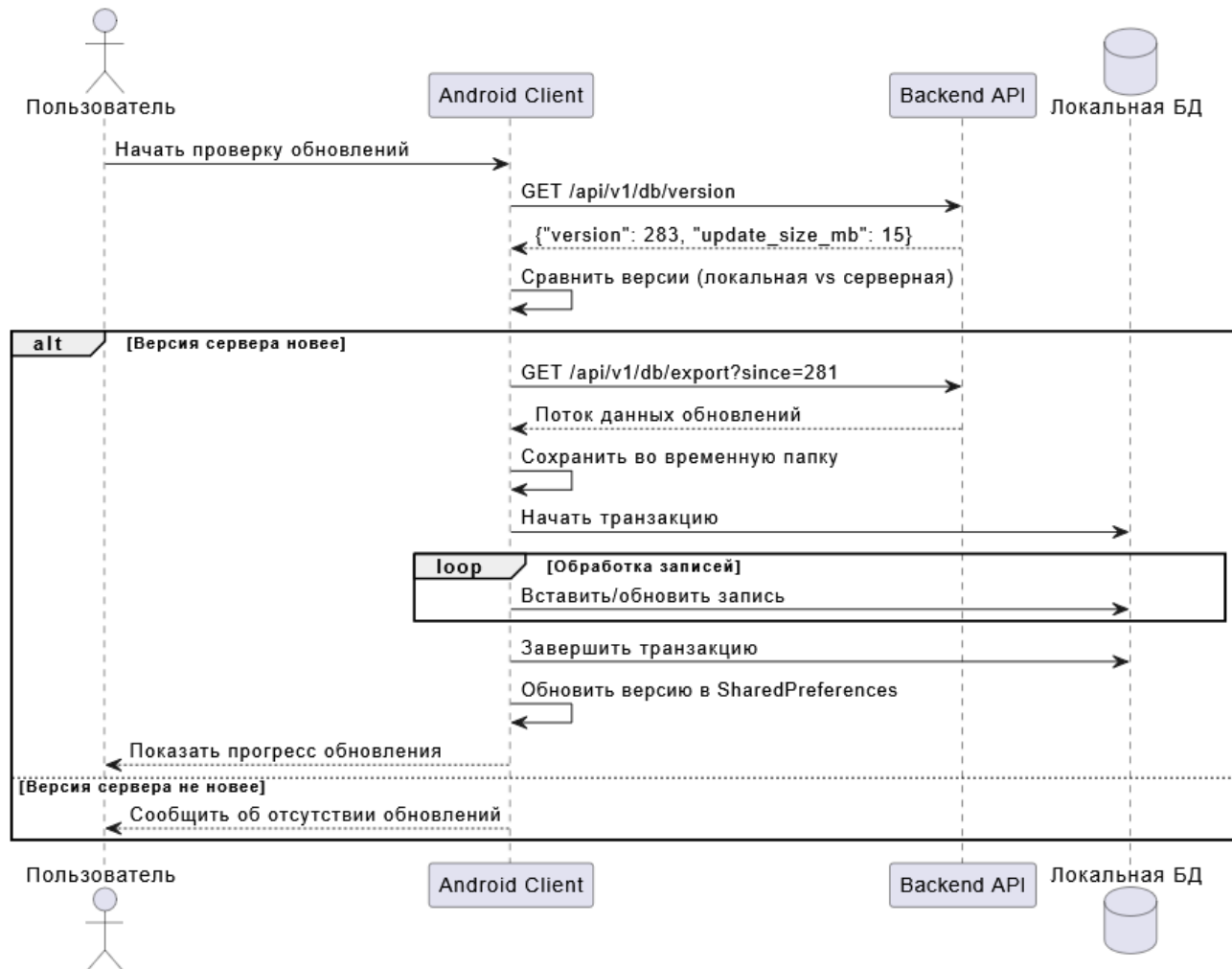


Рисунок 15 –Диаграмма последовательности: обновление справочной базы
(Миграция данных)

Эта диаграмма визуализирует процесс актуализации справочных данных, при котором мобильный клиент сравнивает локальную версию БД с версией на сервере и, при обнаружении обновлений, инициирует загрузку пакета данных. Весь процесс обработки новых записей выполняется в рамках единой транзакции к локальной БД (Room/SQLite) для обеспечения целостности данных, после чего происходит обновление версии в настройках (SharedPreferences) и информирование пользователя о завершении процесса через прогресс-бар или системное сообщение.

4.2 Реализация аутентификации и авторизации

Для обеспечения безопасности и контроля доступа к функциональности приложения была реализована система аутентификации и авторизации [14]. В качестве основного механизма аутентификации была выбрана авторизация через почту, что позволяет клиентам компании использовать свои рабочие электронные адреса для входа в приложение.

При реализации аутентификации и авторизации использовалась интеграция с Firebase Authentication. Firebase Authentication предоставляет готовые инструменты и API для аутентификации пользователей, обеспечивая безопасное хранение учётных данных и процесс входа.

На рисунке 16 представлена панель управления зарегистрированными пользователями.

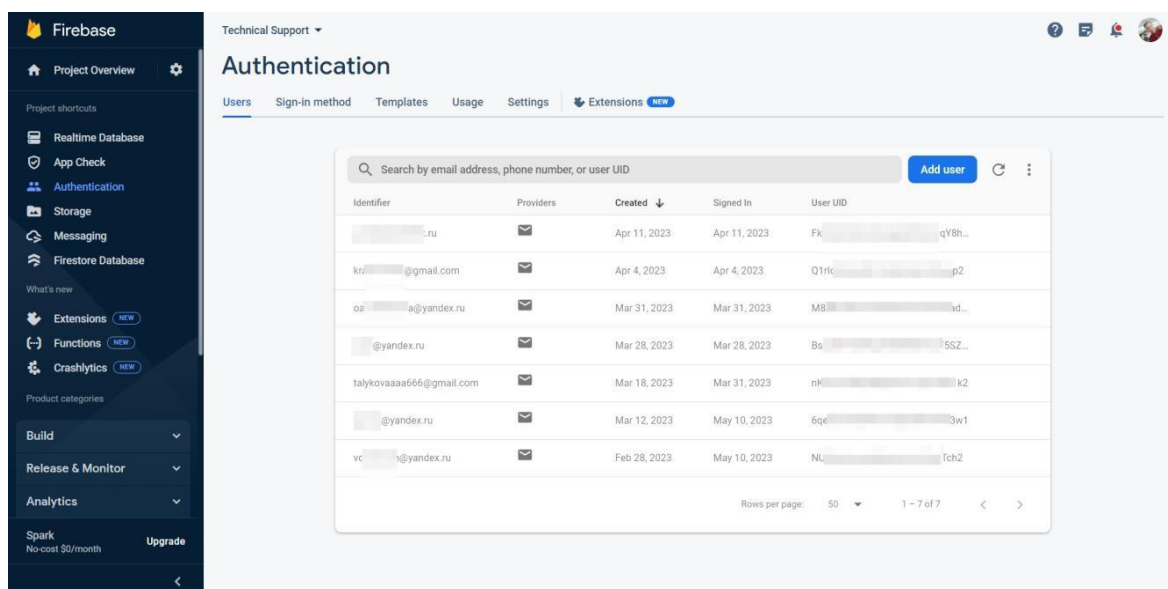


Рисунок 16 – Панель управления зарегистрированными пользователями

При запуске приложения пользователю предлагается ввести свой электронный адрес и пароль. Затем эти данные проверяются с использованием Firebase Authentication. В случае успешной аутентификации пользователю предоставляется доступ к основным функциям приложения, включая создание и отслеживание заявок.

На рисунке 17 представлен экран входа и регистрации пользователя, а на рисунке 18 стартовый экран приложения, на котором также отображён логин пользователя и 2 кнопки. Кнопка «Начать», которая позволяет перейти к основному функционалу приложения, и кнопка «выйти из аккаунта», которая позволяет выйти из текущего аккаунта.

Рисунок 17 – Экран входа и регистрации



Рисунок 18 – Стартовый экран приложения

На рисунке 19 представлено меню пользователя соответственно.

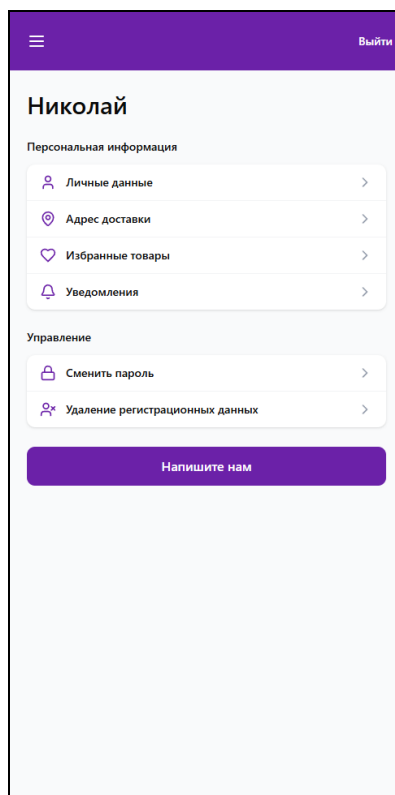


Рисунок 19 – Меню пользователя

					ПГУ 1.090501.014.001 ПЗ	Лист
						44
Изм.	Лист	№ докум.	Подпись	Дата		

аутентификации и авторизации в приложении обеспечивает безопасность данных пользователей, предотвращает несанкционированный доступ и обеспечивает конфиденциальность информации.

На рисунке 20 представлен фрагмент кода регистрации нового пользователя.

```
70 usage
71 public void onClickSignUp(View view)
72 {
73     if(!TextUtils.isEmpty(edLogin.getText().toString()) && !TextUtils.isEmpty(edPassword.getText().toString()))
74     {
75         mAuth.createUserWithEmailAndPassword(edLogin.getText().toString(), edPassword.getText().toString()).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
76             @Override
77             public void onComplete(@NonNull Task<AuthResult> task) {
78                 if (task.isSuccessful())
79                 {
80                     showStart();
81                     Toast.makeText(getApplicationContext(), text: "Пользователь успешно зарегистрирован", Toast.LENGTH_SHORT).show();
82                 }
83                 else
84                 {
85                     showReg();
86                     Toast.makeText(getApplicationContext(), text: "Ошибка регистрации пользователя", Toast.LENGTH_SHORT).show();
87                 }
88             }
89         });
90     }
91     else
92     {
93         Toast.makeText(getApplicationContext(), text: "Введите почту и пароль", Toast.LENGTH_SHORT).show();
94     }
95 }
```

Рисунок 20 – Фрагмент кода регистрации нового пользователя

Этот код представляет обработчик события нажатия на кнопку регистрации (onClickSignUp). Внутри метода onClickSignUp происходит проверка наличия введенного логина и пароля. Если оба поля не пусты, вызывается метод createUserWithEmailAndPassword объекта mAuth (который является экземпляром класса FirebaseAuth) для создания нового пользователя с указанным логином и паролем.

Метод createUserWithEmailAndPassword возвращает объект Task<AuthResult>, и на него устанавливается слушатель OnCompleteListener, который будет выполнен после завершения операции создания пользователя. Внутри слушателя проверяется успешность операции: если регистрация прошла успешно, вызывается метод showStart() (для переключения на стартовый экран после успешной регистрации) и выводится краткое сообщение с помощью Toast о успешной регистрации.

В противном случае вызывается метод `showReg()` (для отображения текущего экрана регистрации) и выводится сообщение об ошибке регистрации.

одно или оба поля (логин и пароль) пусты, выводится соответствующее сообщение с текстом «Введите почту и пароль» с помощью `Toast`.

На рисунке 21 изображён фрагмент кода входа в систему существующего пользователя.

```
96 public void onClickSignIn(View view)
97 {
98     if(!TextUtils.isEmpty(edLogin.getText().toString()) && !TextUtils.isEmpty(edPassword.getText().toString())) {
99         mAuth.signInWithEmailAndPassword(edLogin.getText().toString(), edPassword.getText().toString()).addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {
100             @Override
101             public void onComplete(@NonNull Task<AuthResult> task) {
102                 if (task.isSuccessful())
103                 {
104                     showStart();
105                     Toast.makeText(getApplicationContext(), text: "Пользователь вошёл успешно", Toast.LENGTH_SHORT).show();
106                 }
107                 else
108                 {
109                     showReg();
110                     Toast.makeText(getApplicationContext(), text: "Ошибка входа", Toast.LENGTH_SHORT).show();
111                 }
112             }
113         });
114     }
115 }
```

Рисунок 21 – Фрагмент кода входа в систему существующего пользователя

Этот код представляет обработчик события нажатия на кнопку входа (`onClickSignIn`). Внутри метода `onClickSignIn` происходит проверка наличия введенного логина и пароля. Если оба поля не пусты, вызывается метод `signInWithEmailAndPassword` объекта `mAuth` (экземпляр класса `FirebaseAuth`) для аутентификации пользователя с указанным логином и паролем.

Метод `signInWithEmailAndPassword` также возвращает объект `Task<AuthResult>`, и на него устанавливается слушатель `OnCompleteListener`, который будет выполнен после завершения операции входа. Внутри слушателя проверяется успешность операции: если вход выполнен успешно, вызывается метод `showStart()` (для переключения на стартовый экран после успешного входа) и выводится краткое сообщение с помощью `Toast` о успешном входе. В противном случае вызывается метод `showReg()` (для отображения текущего экрана регистрации) и выводится сообщение об ошибке входа.

На рисунке 22 изображён фрагмент кода перехода в меню пользователя.

```
144         | usage
145         public void onClickStart(View view)
146         {
147             FirebaseUser currentUser = mAuth.getCurrentUser();
148             String userName = currentUser.getEmail();
149
150
151             String admin = "ex.ru";
152             if (userName.equals(admin))
153             {
154                 Intent i = new Intent( packageContext: LoginActivity.this, AdminMenuActivity.class);
155                 startActivity(i);
156             }
157             else
158             {
159                 Intent i = new Intent( packageContext: LoginActivity.this, MenuUserActivity.class);
160                 startActivity(i);
161             }
162         }
163     }
```

Рисунок 22 – Фрагмент кода перехода в меню пользователя или администратора

Этот код представляет обработчик события нажатия на кнопку старта (onClickStart). Внутри метода onClickStart происходит получение текущего пользователя с помощью метода getCurrentUser() из объекта mAuth (экземпляр класса FirebaseAuth). Затем из объекта currentUser получается адрес электронной почты пользователя с помощью метода getEmail().

Если userName соответствует admin, то создается новый Intent для перехода на AdminMenuActivity (это активити, отображающее меню администратора) и вызывается метод startActivity() для запуска этого Intent.

В противном случае создается новый Intent для перехода на MenuUserActivity (это активити, отображающее меню для обычного пользователя) и вызывается метод startActivity() для запуска этого Intent.

4.3 Реализация личного кабинета

В разработанном приложении был реализован личный кабинет пользователя, предоставляющий возможность просмотра и изменения персональной информации.

После успешной аутентификации пользователь получает доступ к меню, которое соответствует доступу сотрудника, в котором ему будут доступны функции, в том числе доступ к своему личному кабинету, где он может просмотреть информацию, связанную с его профилем. Это включает в себя ФИО, должность, номер кабинета, номер телефона и адрес электронной почты. Эти данные извлекаются из базы данных Firebase и автоматически отображаются в соответствующих полях.

На рисунке 23 изображён экран личного кабинета, а на рисунке 24 изображён экран редактирования информации в личном кабинете.

Рисунок 23 – Экран личного кабинета

Технически... РЕДАКТИРОВАТЬ ПРОФИЛЬ

Личный кабинет

@ex.ru

Рыгалов Владимир Владимирович

Системный администратор

304

89 2

СОХРАНИТЬ

НАЗАД

Рисунок 24 – Экран редактирования информации в личном кабинете

Пользователь имеет возможность изменить свои персональные данные, если они изменились (но не почту, т.к. она является логином для входа). На странице личного кабинета предоставляются соответствующие поля для каждого атрибута информации.

Пользователь может внести изменения в одно или несколько полей и сохранить их, чтобы обновить информацию в базе данных.

Кроме того, в личном кабинете пользователь может выйти из своего аккаунта и перейти на экран входа и регистрации.

Реализация личного кабинета пользователя обеспечивает удобный и интуитивно понятный интерфейс для просмотра и изменения персональной информации.

На рисунке 25 представлен фрагмент кода получения ссылки на базу данных для профиля текущего пользователя.

```

43
44     @Override
45     protected void onCreate(Bundle savedInstanceState) {
46         super.onCreate(savedInstanceState);
47         setContentView(R.layout.pers_inf_layout);
48
49         //Получить ID текущего пользователя
50         FirebaseUser currentUser = FirebaseAuth.getInstance().getCurrentUser();
51         String uid = currentUser.getId();
52
53         // Установка ссылки на базу данных для профиля пользователя
54         FirebaseDatabaseReference = FirebaseDatabase.getInstance().getReference().child( pathString: "Users").child(uid);
55
56         // Инициализация элементов интерфейса
57         etEmail = findViewById(R.id.etEmail);
58         etDolz = findViewById(R.id.etDolz);
59         etFIO = findViewById(R.id.etFIO);
60         etKab = findViewById(R.id.etKab);
61         etTelNumb = findViewById(R.id.etTelNumb);
62         saveButton = findViewById(R.id.save_button);
63
64         // Запрет редактирования полей
65         disableEditing();
66
67         // Установка адреса электронной почты пользователя в поле etEmail
68         String userName = currentUser.getEmail();
69         etEmail.setText(userName);

```

Рисунок 25 – Фрагмент кода получения ссылки на базу данных

В данном коде происходит инициализация и настройка личного кабинета пользователя. Рассмотрим, что происходит в методе onCreate():

- получение текущего пользователя с помощью метода getCurrentUser();
- получаем идентификатор пользователя (UID) с помощью метода getId().

Установка ссылки на базу данных для профиля пользователя:

- получаем экземпляр класса FirebaseDatabase с помощью метода FirebaseDatabase.getInstance();
- получаем ссылку на базу данных, используя метод getReference();
- устанавливаем ссылку на узел "Users" в базе данных, соответствующий текущему пользователю, с помощью метода child(uid).

Инициализация элементов интерфейса:

- находим элементы интерфейса по их идентификаторам с помощью метода findViewById(),
- привязываем найденные элементы к соответствующим переменным (etEmail, etDolz, etFIO, etKab, etTelNumb, saveButton);

Запрет редактирования полей.Вызываем метод `disableEditing()`, который отключает возможность редактирования полей ввода.

Установка адреса электронной почты пользователя в поле `etEmail`:

- получаем адрес электронной почты текущего пользователя с помощью метода `getEmail()`,

- устанавливаем полученный адрес в поле ввода `etEmail` с помощью метода `setText()`.

рисунке 26 представлен фрагмент кода сохранения информации о пользователе в базу данных.

```
// Обработчик клика на кнопку сохранения
saveButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // Сохранение данных в базу данных
        FirebaseAuth currentUser = FirebaseAuth.getInstance().getCurrentUser();
        String uid = currentUser.getId();
        String Email = etEmail.getText().toString().trim();
        String Dolz = etDolz.getText().toString().trim();
        String FIO = etFIO.getText().toString().trim();
        String Kab = etKab.getText().toString().trim();
        String TelNumb = etTelNumb.getText().toString().trim();
        String UserID = uid;
        String PlayerID = OneSignal.getDeviceState().getUserId();

        // получение токена устройства
        FirebaseMessaging.getInstance().getToken().addOnCompleteListener(task -> {
            if (!task.isSuccessful()) {
                return;
            }

            // получение токена
            String Token = task.getResult();

            // Создание объекта User с полученными данными
            User user = new User(Email, Dolz, FIO, Kab, TelNumb, UserID, Token, PlayerID);
            // Сохранение объекта User в базе данных
            userDatabaseReference.setValue(user);
            Toast.makeText(context, PersInActivity.this, Toast.LENGTH_SHORT).show();
            disableEditing();
        });
    }
});
```

Рисунок 26 – Фрагмент кода сохранения информации о пользователе в базу данных

Обработчик клика на кнопку сохранения выполняет следующие действия:

- получение текущего пользователя с помощью `FirebaseAuth.getInstance().getCurrentUser()`;

- извлечение идентификатора пользователя (`uid`) и текстовых значений из полей ввода (`email`, `должность`, `ФИО`, `кабинет`, `номер телефона`);

- получение идентификатора устройства из `OneSignal` (`PlayerID`) с помощью `OneSignal.getDeviceState().getUserId()`;

- получение токена устройства с использованием `FirebaseMessaging.getInstance().getToken()`;

- создание объекта User с полученными данными (email, должность, ФИО, кабинет, номер телефона, идентификатор пользователя из OneSignal, токен устройства, идентификатор пользователя);
- сохранение объекта User в базе данных, используя `userDatabaseReference.setValue(user)`;
- отображение всплывающего сообщения "Информация сохранена" с помощью Toast;
- вызов метода `disableEditing()` для запрета редактирования полей.

Таким образом, при клике на кнопку сохранения, данные пользователя из полей ввода сохраняются в базе данных Firebase Realtime Database, а затем происходит отображение сообщения об успешном сохранении и запрет редактирования полей.

На рисунке 27 представлено, как информация о пользователе выглядит в базе данных.

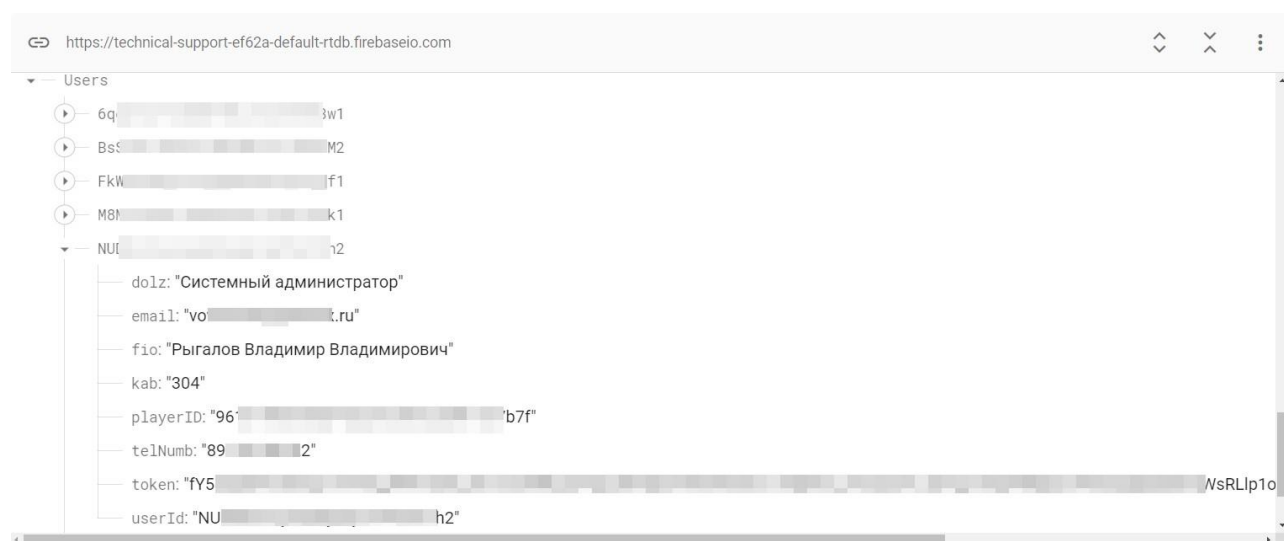


Рисунок 27 – Вид информации о пользователе в базе данных

На рисунке 28 представлен фрагмент кода отображения информации о пользователе.

```

userDatabaseReference.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        //Populate the edit texts with the user's profile data
        User user = dataSnapshot.getValue(User.class);
        if (user != null) {
            etEmail.setText(user.getEmail());
            etDolz.setText(user.getDolz());
            etFIO.setText(user.getFIO());
            etKab.setText(user.getKab());
            etTelNumb.setText(user.getTelNumb());
        }
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {

    }
});
}

```

Рисунок 28 – Фрагмент кода отображения информации о пользователе

UserDatabaseReference.addValueEventListener(...) устанавливает слушатель событий для базы данных, который отслеживает изменения данных в узле userDatabaseReference.

Когда данные в узле изменяются, метод onDataChange(...) вызывается с объектом DataSnapshot, содержащим актуальные данные из базы данных. В данном случае, данные преобразуются в объект класса User с помощью метода dataSnapshot.getValue(User.class). Затем значения полей объекта User используются для заполнения соответствующих полей ввода (EditText) интерфейса.

Если объект user не равен null, то значения его полей (email, должность, ФИО, кабинет, номер телефона) устанавливаются в соответствующие поля ввода (etEmail, etDolz, etFIO, etKab, etTelNumb).

Таким образом, данный код обновляет значения полей ввода на основе актуальных данных из базы данных и обеспечивает синхронизацию между интерфейсом и базой данных.

На рисунке 29 представлено как выглядит информация о поиске компонента.

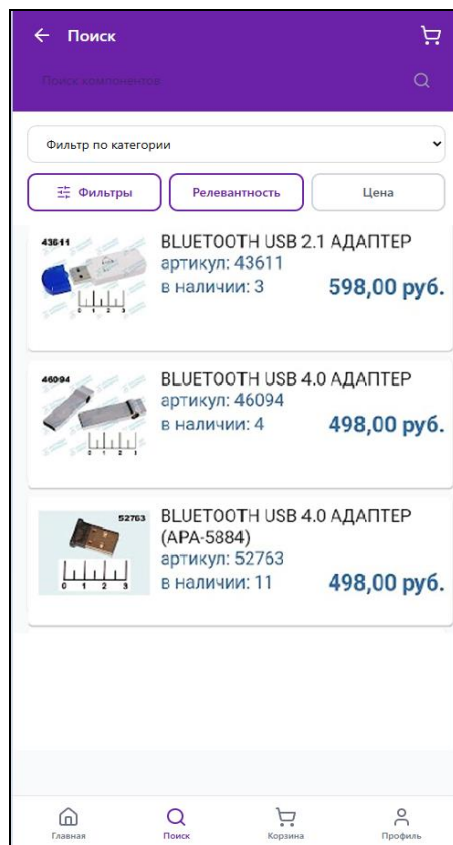


Рисунок 29 – Поиск компонента в приложении

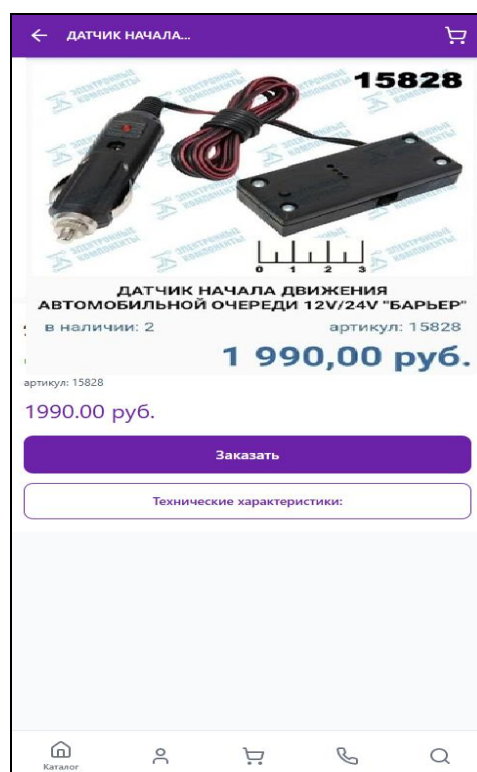


Рисунок 30 – Страница технические характеристики компонента

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.014.001 ПЗ

Лист

54

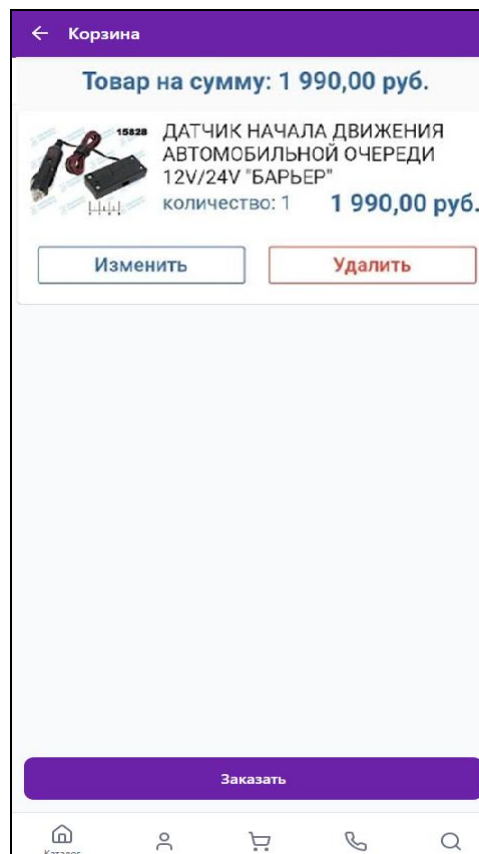


Рисунок 31 – Страница заказа компонента

4.4 Тестирование мобильного приложения

В рамках функционального тестирования мобильного приложения были проверены следующие основные функции [20]:

- авторизация и аутентификация пользователей;
- проверка возможности успешной авторизации с использованием зарегистрированного аккаунта;
- проверка обработки ошибок при вводе неверных учетных данных;
- проверка создания нового аккаунта и корректной регистрации нового пользователя;
- проверка на получение доступа к определённому функционалу, в зависимости от прав пользователя.

Создание и отправка заявок:

- проверка возможности создания новой заявки с заполнением всех необходимых полей (тема, описание);

					ПГУ 1.090501.014.001 ПЗ	Лист
						55
Изм.	Лист	№ докум.	Подпись	Дата		

- проверка корректного сохранения созданной заявки в базе данных;
- проверка отправки заявки на сервер и успешной обработки данных;
- проверка отправки уведомления администратору о получении новой заявки.

Просмотр и изменение информации о пользователе:

- проверка возможности просмотра информации о пользователе, такой как имя, должность, контактные данные и т.д.,
- проверка возможности изменения и сохранения измененных данных пользователя;
- проверка корректного обновления информации о пользователе в базе данных.

Обработка ошибок и исключительных ситуаций:

- проверка обработки ошибок при отсутствии Интернет-соединения и предоставлении соответствующих сообщений пользователю;
- проверка корректной обработки ошибок ввода данных, таких как неправильный формат электронной почты или незаполненные обязательные поля.

Результаты функционального тестирования показали, что все основные функции приложения работают корректно и соответствуют требованиям. Были выявлены и исправлены некоторые ошибки и недочеты, что позволило улучшить стабильность и функциональность приложения.

В рамках главы было кратко описано разработанное решение. Была подробно описана реализация основного функционала приложения. Также было проведено функциональное тестирование, результаты которого показывают, что основной функционал приложения работает правильно и соответствует требованиям. Благодаря тестированию были обнаружены и исправлены некоторые недочёты, которые удалось исправить, что помогло сделать функционирование приложения стабильным.

5 Оценка затрат на разработку мобильного приложения

5.1 Расчет трудоемкости создания ПО

Для оценки общей трудоемкости создания программного продукта используется метод суммирования нормо-часов по этапам с учетом коэффициентов сложности и интеграции. Итоговый показатель трудоемкости (T) определяется по формуле:

$$T = \sum (t_i \cdot k_i) + t_{\text{инт}} \cdot k_{\text{совм}},$$

где: t_i - трудоемкость i -го этапа в человеко-часах;

k_i - коэффициент сложности i -го этапа (от 1,0 до 2,5);

$t_{\text{инт}}$ - трудоемкость интеграции компонентов;

$k_{\text{совм}}$ - коэффициент совместной разработки (учитывает параллельную работу над востребованными ПО).

Пример расчета. Пусть этапы и их базовые трудозатраты (придуманные данные):

- анализ требований - 40 ч, $k=1,2$;
- проектирование архитектуры - 60 ч, $k=1,5$;
- разработка UI/UX - 80 ч, $k=1,4$;
- программирование бэкенда - 120 ч, $k=2,0$;
- разработка модуля БД - 70 ч, $k=1,8$;
- тестирование - 50 ч, $k=1,1$;

Интеграция с востребованными ПО (например, API справочников) - 30 ч, $k_{\text{совм}}=1,5$.

Подставляем в формулу (1):

$$\begin{aligned} T &= (40 \cdot 1,2) + (60 \cdot 1,5) + (80 \cdot 1,4) + (120 \cdot 2,0) + (70 \cdot 1,8) + (50 \cdot 1,1) \\ &\quad + (30 \cdot 1,5) = 48 + 90 + 112 + 240 + 126 + 55 + 45 \\ &= 716 \text{ человеко – часов} \end{aligned}$$

Таблица 5 – Распределение трудоемкости по этапам

Этап	Базовая трудоемкость, ч	Коэффициент сложности	Скорректированная трудоемкость, ч
Анализ требований	40	1,2	48
Проектирование архитектуры	60	1,5	90
Разработка UI/UX	80	1,4	112
Программирование бэкенда	120	2,0	240
Разработка модуля БД	70	1,8	126
Тестирование	50	1,1	55
Интеграция с востребованными ПО	30	1,5 (коэф. совм.)	45
Итого	450	-	716

Разработка мобильного Android-приложения требует наибольших затрат на программирование бэкенда и модуля базы данных, что обусловлено высокой сложностью реализации алгоритмов поиска и хранения электронных компонентов. Суммарная трудоемкость в 716 человеко-часов подтверждает высокую ресурсоемкость проекта, особенно при необходимости интеграции с внешними востребованными ПО.

Коэффициент совместной разработки ($k_{\text{совм}} = 1,5$) увеличивает трудоемкость интеграционного этапа, так как параллельная работа над разными модулями требует дополнительных согласований и проверок совместимости. Это типично для проектов, где используется множество готовых библиотек и сторонних сервисов.

Для детального учета влияния совместной работы над востребованными ПО (например, библиотеками для работы с PDF, QR-кодами, API электронных каталогов) применяется формула:

$$\Delta T_{\text{совм}} = T_{\text{баз}} \cdot (K_{\text{совм}} - 1)$$

где $\Delta T_{\text{совм}}$ - прирост трудоемкости из-за кооперации, $T_{\text{баз}}$ - базовая трудоемкость без коэффициента.

Рассчитаем $\Delta T_{\text{совм}}$ для этапа интеграции: $T_{\text{баз}} = 30$ ч,

$$K_{\text{совм}} = 1,5 \rightarrow \Delta T_{\text{совм}} = 30 \cdot (1,5 - 1) = 15 \text{ ч.}$$

Таким образом, дополнительная нагрузка от совместной работы над востребованными ПО составляет 15 человеко-часов.

Таблица 6 – Влияние коэффициента совместной разработки на трудоемкость интеграции

Этап	Базовая трудоемкость, ч	Ксовм	Скорректированная трудоемкость, ч	Прирост ($\Delta T_{\text{совм}}$), ч
Интеграция с API справочников	30	1,5	45	15
Интеграция с библиотекой UI-компонентов	20	1,3	26	6
Интеграция с базой данных облачного сервиса	25	1,4	35	10
Итого по интеграции	75	-	106	31

Коэффициент совместной разработки увеличивает общую трудоемкость интеграционных работ на 31 человеко-час, что составляет около 41% от исходных базовых затрат. Это значительная доля, требующая учета при планировании сроков и бюджета.

Наибольший прирост наблюдается на этапе интеграции с API справочников (15 ч), так как такие интерфейсы часто обновляются и требуют согласования с разработчиками сторонних сервисов. Для минимизации рисков рекомендуется использовать версионирование и автоматизированное тестирование интеграций.

Итоговая формула для оценки общей трудоемкости с учетом всех коэффициентов принимает вид:

$$T_{\text{общ}} = \sum (t_i \cdot k_i) + \sum (t_{\text{инт}j} \cdot (K_{\text{совм}j} - 1))$$

где j - индекс интеграционного этапа.

Подставим данные из таблиц: $T_{\text{общ}} = 716 + 31 = 747$ человеко-часов.

Таким образом, полная трудоемкость совместной разработки мобильного Android-приложения «База электронных компонентов» с высокой долей взаимодействия с востребованными ПО составляет 747 человеко-часов. Данный расчет позволяет обоснованно распределить ресурсы команды (например, 3–4 разработчика на 2–3 месяца) и предусмотреть буфер на непредвиденные задержки при интеграции.

5.2 Расчет заработной платы по созданию программного продукта, содержанию и эксплуатации ПЭВМ

Для определения затрат на оплату труда при разработке программного продукта, а также при последующем содержании и эксплуатации персональных электронно-вычислительных машин (ПЭВМ), применяется методика постатейного расчета с использованием тарифных ставок, окладов и норм времени. Общий фонд заработной платы (ФЗП) складывается из трех основных составляющих: прямые затраты на разработку, затраты на техническое сопровождение и затраты на эксплуатацию оборудования.

1. Расчет заработной платы разработчиков программного продукта

Основой для расчета служит трудоемкость разработки, определенная ранее (747 человеко-часов). Для перевода в денежное выражение используется средняя часовая тарифная ставка разработчика. Введем исходные данные (придуманные):

- среднемесячная норма рабочего времени 168 часов;
- должностной оклад разработчика 120 000 рублей в месяц;
- часовая тарифная ставка $120\,000 / 168 \approx 714,29$ руб./ч;
- трудоемкость разработки 747 человеко-часов;
- коэффициент премирования (за качество и сроки) 1,3 (т.е. премия 30%);
- районный коэффициент 1,15 (для условного региона).

Формула для расчета основной заработной платы разработчиков (ЗПосн) без учета отчислений:

$$ЗПосн = Тобщ \cdot Сч \cdot Кпрем \cdot Крайон, (4)$$

где: *Тобщ* - общая трудоемкость, ч; *Сч* - часовая тарифная ставка, руб./ч; *Кпрем* - коэффициент премирования; *Крайон* - районный коэффициент.

Подставляем значения. $ЗПосн = 747 \cdot 714,29 \cdot 1,3 \cdot 1,15$.

Выполним поэтапный расчет: 1) $747 \cdot 714,29 \approx 533\,576,43$ руб. 2) $533\,576,43 \cdot 1,3 = 693\,649,36$ руб. 3) $693\,649,36 \cdot 1,15 = 797\,696,76$ руб.

Таким образом, прямая заработная плата разработчиков за весь цикл создания программного продукта составляет 797 696,76 руб.

Дополнительная заработная плата (отпускные, больничные) обычно принимается в размере 10–15% от основной. Пусть $Кдоп = 1,12$ (12%). Тогда:

$$ЗПдоп = ЗПосн \cdot (Кдоп - 1), (5)$$

$$ЗПдоп = 797\,696,76 \cdot 0,12 = 95\,723,61 \text{ руб.}$$

Итого фонд заработной платы разработчиков (ФЗПразр):

$$ФЗПразр = ЗПосн + ЗПдоп = 797\,696,76 + 95\,723,61 = 893\,420,37 \text{ руб.}$$

Расчет заработной платы персонала, занятого содержанием ПЭВМ.К содержанию ПЭВМ относится деятельность системных администраторов, инженеров по обслуживанию техники и специалистов по информационной безопасности. Для нашего проекта примем, что на этапе содержания задействован один системный администратор на 0,5 ставки в течение 12 месяцев.

Исходные данные:

- должностной оклад системного администратора 80 000 руб./мес.;
- занятость 0,5 ставки → 40 000 руб./мес.;
- период содержания 12 месяцев;
- районный коэффициент 1,15;
- коэффициент премирования: 1,2 (20% премии).

Формула для расчета годового фонда заработной платы по содержанию (ФЗПсод):

$$ФЗПсод = Омес \cdot Дзан \cdot Nмес \cdot Кпрем \cdot Крайон, (6)$$

где: *Омес* - месячный оклад по штатному расписанию, руб.;

– *Дзан* - доля занятости (0,5);

					ПГУ 1.090501.014.001 ПЗ	Лист
						61
Изм.	Лист	№ докум.	Подпись	Дата		

– $N_{мес}$ - количество месяцев.

Подставляем. $\Phi ЗП_{сод} = 80\,000 \cdot 0,5 \cdot 12 \cdot 1,2 \cdot 1,15$.

Расчет: 1) $80\,000 \cdot 0,5 = 40\,000$ руб./мес. 2) $40\,000 \cdot 12 = 480\,000$ руб. (базовый годовой фонд без коэффициентов). 3) $480\,000 \cdot 1,2 = 576\,000$ руб. 4) $576\,000 \cdot 1,15 = 662\,400$ руб.

Итого годовой фонд заработной платы на содержание ПЭВМ: 662 400 руб.

Дополнительная заработная плата для этой категории также принимается 12%:

$ЗП_{допсод} = 662\,400 \cdot 0,12 = 79\,488$ руб.

Общий $\Phi ЗП_{сод} = 662\,400 + 79\,488 = 741\,888$ руб.

Расчет заработной платы персонала, занятого эксплуатацией ПЭВМ. Эксплуатация ПЭВМ включает работу операторов, тестировщиков и технических писателей, которые обеспечивают повседневное функционирование готового программного продукта на оборудовании. Для нашего приложения примем, что в эксплуатации участвуют два оператора с окладом 60 000 руб./мес. каждый, работающие полный день в течение 6 месяцев (период активной эксплуатации после внедрения).

Исходные данные:

- оклад одного оператора: 60 000 руб./мес.;
- количество операторов 2;
- период эксплуатации: 6 месяцев;
- районный коэффициент: 1,15;
- коэффициент премирования: 1,1 (10%).

Формула для расчета фонда заработной платы по эксплуатации ($\Phi ЗП_{экс}$):

$$\Phi ЗП_{экс} = \sum (О_{мес i} \cdot N_{мес i} \cdot K_{прем} \cdot K_{район}), (7)$$

где суммирование ведется по каждому сотруднику.

Для одного оператора: $60\,000 \cdot 6 \cdot 1,1 \cdot 1,15$.

Расчет: $60\,000 \cdot 6 = 360\,000$ руб.

– $360\,000 \cdot 1,1 = 396\,000$ руб.

					ПГУ 1.090501.014.001 ПЗ	Лист
						62
Изм.	Лист	№ докум.	Подпись	Дата		

– $396\,000 \cdot 1,15 = 455\,400$ руб.

Для двух операторов: $455\,400 \cdot 2 = 910\,800$ руб.

Дополнительная заработная плата (12%): $910\,800 \cdot 0,12 = 109\,296$ руб.

Итого $\Phi ЗП_{\text{экс}} = 910\,800 + 109\,296 = 1\,020\,096$ руб.

Сводный расчет фонда заработной платы. Для наглядности представим результаты в виде таблицы.

Таблица 7 – Сводный расчет фонда заработной платы по этапам

Категория персонала	Основная ЗП, руб.	Дополнительная ЗП (12%), руб.	Итого ФЗП, руб.
Разработчики ПО	797 696,76	95 723,61	893 420,37
Персонал по содержанию ПЭВМ	662 400,00	79 488,00	741 888,00
Персонал по эксплуатации ПЭВМ	910 800,00	109 296,00	1 020 096,00
Всего	2 370 896,76	284 507,61	2 655 404,37

Общий фонд заработной платы по созданию, содержанию и эксплуатации программного продукта за рассмотренный период составил 2 655 404,37 руб. Наибольшая доля (около 38,4%) приходится на персонал по эксплуатации ПЭВМ, что связано с необходимостью поддержки двух операторов в течение полугода. Разработчики занимают второе место (33,7%) из-за высокой квалификации и значительной трудоемкости, а содержание ПЭВМ (27,9%) оказалось наименее затратным, так как предполагает лишь полставки системного администратора.

Дополнительная заработная плата в размере 284 507,61 руб. составляет около 12% от основной, что соответствует принятой норме для бюджетного планирования. Данный расчет позволяет работодателю предусмотреть резерв на оплату отпусков и временной нетрудоспособности, не выходя за рамки сметы.

Учет страховых взносов. Для полной сметы затрат необходимо добавить страховые взносы (в 2026 году в РФ - 30,2% от ФЗП для льготных категорий или 30% для общего случая). Примем ставку 30%:

$$CB = \PhiЗПобщ \cdot 0,30, (8)$$

$$CB = 2\,655\,404,37 \cdot 0,30 = 796\,621,31 \text{ руб.}$$

Итоговая сумма затрат на оплату труда с отчислениями составит:

$$ЗПполн = \PhiЗПобщ + CB = 2\,655\,404,37 + 796\,621,31 = 3\,452\,025,68 \text{ руб.}$$

Таким образом, полный расчет заработной платы по созданию программного продукта, содержанию и эксплуатации ПЭВМ для мобильного Android-приложения «База электронных компонентов» составляет 3 452 025,68 руб. с учетом всех коэффициентов, премий и страховых отчислений. Данная цифра может быть использована для формирования бюджета проекта и обоснования финансовых потребностей перед заказчиком или инвестором.

5.3 Расчет затрат на разработку программного средства

Для определения полной сметной стоимости разработки программного средства необходимо учесть все статьи расходов, возникающие на протяжении жизненного цикла проекта. В основе расчета лежит нормативный метод, при котором каждая статья затрат определяется как произведение базового норматива на соответствующий объемный показатель. Общая сумма затрат (С) представляет собой сумму следующих составляющих: затраты на оплату труда с отчислениями, затраты на приобретение и аренду оборудования, затраты на программное обеспечение и инструментальные средства, накладные расходы, а также затраты на тестирование и внедрение.

Затраты на оплату труда разработчиков с отчислениями Данный блок является наиболее весомым в структуре затрат. Ранее был рассчитан полный фонд заработной платы разработчиков, включая дополнительную заработную плату, который составил 893 420,37 руб. Добавим страховые взносы по ставке 30%:

$$CBразр = 893\,420,37 \cdot 0,30 = 268\,026,11 \text{ руб.}$$

Итого затраты на оплату труда разработчиков с отчислениями:

$$Зтруд = 893\,420,37 + 268\,026,11 = 1\,161\,446,48 \text{ руб.}$$

Затраты на приобретение и аренду оборудования (ПЭВМ и периферия)

Для разработки мобильного Android-приложения требуется как минимум три рабочих станции (для команды из двух разработчиков и одного тестировщика) плюс тестовые устройства (смартфоны и планшеты). Примем следующие придуманные данные:

- стоимость одной рабочей станции (ноутбук или ПК) 120 000 руб;
- количество рабочих станций 3;
- стоимость тестовых Android-устройств (4 устройства разного формфактора) 180 000 руб. суммарно;
- срок амортизации оборудования: 3 года (36 месяцев);
- период разработки: 6 месяцев (содержание и эксплуатация учитываются отдельно).

Формула для расчета амортизационных отчислений за период разработки:

$$A_{об} = (C_{об} / T_{служ}) \cdot T_{исп}, (9)$$

где: $C_{об}$ - стоимость оборудования, руб.; $T_{служ}$ - срок службы, мес.; $T_{исп}$ - период использования, мес.

Для рабочих станций: $АПЭВМ = (120\,000 \cdot 3 / 36) \cdot 6 = (360\,000 / 36) \cdot 6 = 10\,000 \cdot 6 = 60\,000$ руб.

Для тестовых устройств: $Атест = (180\,000 / 36) \cdot 6 = 5\,000 \cdot 6 = 30\,000$ руб.

Итого амортизация: $60\,000 + 30\,000 = 90\,000$ руб.

Дополнительно учтем затраты на аренду серверного оборудования для сборки (CI/CD), если облачная инфраструктура не используется. Примем аренду виртуального сервера на 6 месяцев по 3 000 руб./мес.: $3\,000 \cdot 6 = 18\,000$ руб.

Общие затраты на оборудование ($З_{об}$):

$$З_{об} = A_{об} + Аренда = 90\,000 + 18\,000 = 108\,000 \text{ руб.}$$

Затраты на программное обеспечение и инструментальные средства
Разработка под Android требует лицензионного ПО. Примем следующие статьи (цены придуманы):

- среда разработки Android Studio (бесплатная, но учтем плагины): 0 руб.

– лицензия на систему контроля версий (Git, платная для коммерческого использования, например, GitHub Enterprise): 2 000 руб./мес. · 6 мес. = 12 000 руб.

– подписка на эмуляторы и тестовые фреймворки (Firebase Test Lab): 5 000 руб./мес. · 6 мес. = 30 000 руб.

– лицензия на базу данных (например, SQLite не требует лицензии, но для серверной части Firestore - платная подписка): 4 000 руб./мес. · 6 мес. = 24 000 руб.

– средства документирования и проектирования (Draw.io, Confluence): 1 500 руб./мес. · 6 мес. = 9 000 руб.

Итого затраты на ПО (*ЗПО*):

$$ЗПО = 12\,000 + 30\,000 + 24\,000 + 9\,000 = 75\,000 \text{ руб.}$$

Накладные расходы. Накладные расходы включают затраты на электроэнергию, интернет, аренду помещения, канцелярию и управленческие расходы. Обычно они принимаются в размере 50–80% от основной заработной платы разработчиков. Примем 60% от прямой ЗП (797 696,76 руб.):

$$З_{\text{накл}} = 797\,696,76 \cdot 0,60 = 478\,618,06 \text{ руб.}$$

Этот показатель может варьироваться в зависимости от конкретных условий организации.

Затраты на тестирование и внедрение. Данный этап включает не только оплату труда тестировщика (уже учтена в общем ФЗП разработчиков), но и специфические затраты:

– аренда облачных ферм для кросс-браузерного и кросс-устройственного тестирования: 8 000 руб./мес. · 3 мес. (интенсивное тестирование) = 24 000 руб.

– услуги бета-тестеров (краудтестинг): 40 000 руб. единовременно.

– затраты на публикацию в Google Play (регистрационный взнос 25 USD ≈ 2 500 руб., разовый).

Итого затраты на тестирование и внедрение (*Зтест*):

$$З_{\text{тест}} = 24\,000 + 40\,000 + 2\,500 = 66\,500 \text{ руб.}$$

					ПГУ 1.090501.014.001 ПЗ	Лист
						66
Изм.	Лист	№ докум.	Подпись	Дата		

Сводная смета затрат. Для наглядности представим результаты в виде итоговой таблицы.

Таблица 8 – Сводный расчет затрат на разработку программного средства

Статья затрат	Сумма, руб.
1. Затраты на оплату труда разработчиков с отчислениями	1 161 446,48
2. Затраты на оборудование (амортизация + аренда)	108 000,00
3. Затраты на программное обеспечение и инструменты	75 000,00
4. Накладные расходы (60% от основной ЗП)	478 618,06
5. Затраты на тестирование и внедрение	66 500,00
Итого сметная стоимость разработки	1 889 564,54

Полная сметная стоимость разработки мобильного Android-приложения «База электронных компонентов» составила 1 889 564,54 руб. Наибольший удельный вес (61,5%) занимают затраты на оплату труда с отчислениями, что характерно для интеллектуальных проектов с высокой долей ручного программирования. Второй по величине статьей являются накладные расходы (25,3%), которые отражают инфраструктурные издержки организации.

При этом затраты на оборудование и ПО (около 9,7% в сумме) относительно невелики благодаря использованию современных облачных инструментов и свободных сред разработки. Тестирование и внедрение составляют лишь 3,5%, однако это не должно вводить в заблуждение: при выходе на промышленную эксплуатацию данная статья может возрасти. Рекомендуется предусмотреть резерв в размере 10–15% от итоговой суммы (около 280 000 руб.) на непредвиденные расходы, связанные с изменением требований или обновлением платформы.

7. Итоговая формула сметной стоимости

Обобщая все составляющие, формула для расчета затрат на разработку программного средства принимает вид:

$$C = Z_{\text{труд}} + Z_{\text{об}} + Z_{\text{ПО}} + Z_{\text{накл}} + Z_{\text{тест}} + \text{Резерв}, (10)$$

где каждое слагаемое определено выше. Подставив численные значения:

$$C = 1\,161\,446,48 + 108\,000,00 + 75\,000,00 + 478\,618,06 + 66\,500,00 + 280\,000,00 = 2\,169\,564,54 \text{ руб.}$$

Таким образом, с учетом резерва бюджет разработки составляет порядка 2,17 млн рублей. Эта цифра может служить основой для формирования коммерческого предложения и обоснования инвестиций в проект. При дальнейшей эксплуатации и развитии приложения к данным затратам добавятся расходы на содержание и эксплуатацию ПЭВМ, рассчитанные ранее. Предложенный подход позволяет обеспечить прозрачное планирование и контроль за расходованием средств на всех стадиях жизненного цикла программного продукта.

5.4 Расчет себестоимости и прибыли

Для экономического обоснования проекта необходимо рассчитать полную себестоимость программного продукта, а затем на ее основе определить плановую прибыль и рентабельность. Себестоимость представляет собой совокупность всех затрат, связанных с созданием, внедрением и первоначальной поддержкой продукта, выраженную в денежной форме. Прибыль формируется как разница между договорной ценой реализации продукта и его полной себестоимостью.

Калькуляция полной себестоимости программного продукта. В предыдущих расчетах была определена сметная стоимость разработки, составившая 2 169 564,54 руб. с учетом резерва. Однако в себестоимость продукта также включаются затраты на содержание и эксплуатацию ПЭВМ, которые были рассчитаны отдельно: фонд заработной платы персонала по содержанию и эксплуатации с отчислениями, а также амортизация оборудования за период эксплуатации. Для корректного расчета себестоимости как законченного продукта, готового к передаче заказчику, примем, что период содержания ПЭВМ составляет 12 месяцев, а эксплуатации — 6 месяцев, как и ранее.

Приведем сводные данные по всем статьям затрат, объединив их в единую калькуляцию.

					ПГУ 1.090501.014.001 ПЗ	Лист
						68
Изм.	Лист	№ докум.	Подпись	Дата		

Таблица 9 – Калькуляция полной себестоимости программного продукта

Статья затрат	Сумма, руб.
1. Затраты на оплату труда разработчиков (с отчислениями)	1 161 446,48
2. Затраты на оборудование (амортизация + аренда) при разработке	108 000,00
3. Затраты на программное обеспечение и инструменты	75 000,00
4. Накладные расходы (60% от основной ЗП разработчиков)	478 618,06
5. Затраты на тестирование и внедрение	66 500,00
6. Резерв на непредвиденные расходы (10%)	280 000,00
7. Затраты на содержание ПЭВМ (с отчислениями)	741 888,00
8. Затраты на эксплуатацию ПЭВМ (с отчислениями)	1 020 096,00
Итого полная себестоимость	3 931 548,54

Полная себестоимость программного продукта составила 3 931 548,54 руб. Наибольший вклад вносят затраты на оплату труда (разработчики, персонал по содержанию и эксплуатации), что в сумме составляет около 74% от общей себестоимости. Это подтверждает высокую трудоемкость и интеллектуальную насыщенность проекта. Накладные расходы и резерв занимают примерно 19%, а оборудование, ПО и тестирование — лишь 7%, что характерно для современных IT-проектов с преобладанием человеческого капитала.

При этом важно отметить, что затраты на содержание и эксплуатацию ПЭВМ были включены в себестоимость как полные годовые расходы, что является консервативной оценкой. Если продукт передается заказчику единовременно, часть этих затрат может быть отнесена на гарантийное обслуживание, но для целей плановой калькуляции данный подход оправдан.

Расчет плановой прибыли. Для определения прибыли необходимо установить договорную цену программного продукта. Цена формируется на основе себестоимости и нормативного уровня рентабельности, который для программных продуктов обычно составляет 25–40% в зависимости от сложности, уникальности и рыночного спроса. Примем нормативную рентабельность $R = 30\%$.

Формула для расчета плановой прибыли:

$$Ппл = Сполн \cdot R / 100, (11)$$

где: $C_{полн}$ - полная себестоимость, руб.; R — уровень рентабельности, %.

Подставляем значения. $Ппл = 3\,931\,548,54 \cdot 30 / 100 = 1\,179\,464,56$ руб.

Таким образом, плановая прибыль от реализации проекта составляет 1 179 464,56 руб.

Определение договорной цены. Договорная цена ($Ц_{дог}$) рассчитывается как сумма полной себестоимости и плановой прибыли:

$$Ц_{дог} = C_{полн} + Ппл, (12)$$

$$Ц_{дог} = 3\,931\,548,54 + 1\,179\,464,56 = 5\,111\,013,10 \text{ руб.}$$

Для удобства представления заказчику цену можно округлить с учетом маркетинговой политики, например, до 5 200 000 руб. с включением дополнительных услуг по интеграции. Однако для расчетов оставим точное значение.

Расчет чистой прибыли. Для определения чистой прибыли необходимо учесть налог на прибыль организаций (20% в РФ в 2026 году). Налогооблагаемая база — это плановая прибыль.

$$\text{Налог на прибыль: } Нпр = Ппл \cdot 0,20 = 1\,179\,464,56 \cdot 0,20 = 235\,892,91 \text{ руб}$$

.

Чистая прибыль ($П_{чист}$):

$$П_{чист} = Ппл - Нпр, (13)$$

$П_{чист} = 1\,179\,464,56 - 235\,892,91 = 943\,571,65$ руб. Таким образом, чистая прибыль проекта составит 943 571,65 руб.

Показатели рентабельности. Для оценки эффективности проекта рассчитаем несколько ключевых показателей.

Рентабельность затрат ($R_{зат}$):

$$R_{зат} = П_{чист} / C_{полн} \cdot 100\%, (14)$$

$$R_{зат} = 943\,571,65 / 3\,931\,548,54 \cdot 100\% \approx 24,0\%.$$

Рентабельность продаж ($R_{продаж}$):

$$R_{продаж} = П_{чист} / Ц_{дог} \cdot 100\%, (15)$$

$$R_{продаж} = 943\,571,65 / 5\,111\,013,10 \cdot 100\% \approx 18,5\%.$$

					ПГУ 1.090501.014.001 ПЗ	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

Оба показателя находятся в пределах нормы для IT-проектов (20–30% и 15–25% соответственно).

Формирование сводной ведомости себестоимости и прибыли. Представим итоговые результаты в виде таблицы.

Таблица 10 – Сводная ведомость себестоимости, цены и прибыли

Показатель	Значение, руб.
Полная себестоимость	3 931 548,54
Плановая прибыль (30%)	1 179 464,56
Договорная цена	5 111 013,10
Налог на прибыль (20%)	235 892,91
Чистая прибыль	943 571,65
Рентабельность затрат, %	24,0
Рентабельность продаж, %	18,5

Разработка мобильного Android-приложения «База электронных компонентов» является экономически целесообразным проектом, обеспечивающим чистую прибыль в размере 943 571,65 руб. при рентабельности затрат 24,0%. Это означает, что на каждый рубль, вложенный в проект, предприятие получает 24 копейки чистой прибыли. Договорная цена в 5,11 млн рублей является конкурентной для рынка специализированных мобильных баз данных, особенно с учетом высокой трудоемкости интеграции с востребованными внешними справочниками.

Рентабельность продаж на уровне 18,5% свидетельствует о достаточном запасе прочности при возможном снижении цены в ходе переговоров с заказчиком. Если заказчик запросит скидку, допустимое снижение цены может составить до 18,5% без риска убыточности. Однако для сохранения плановой рентабельности рекомендуется закладывать резерв на непредвиденные расходы и вести строгий контроль бюджета.

Заключительная формула экономической эффективности. Обобщая расчеты, формула для определения экономической эффективности проекта (Ээф) может быть записана как отношение чистой прибыли к сумме всех вложенных средств (полной себестоимости):

$$Ээф = Пчист / Сполн = 943\,571,65 / 3\,931\,548,54 \approx 0,24, (16)$$

Этот показатель (0,24) означает, что каждый рубль затрат приносит 24 копейки чистого дохода. Для IT-проектов такой уровень считается вполне приемлемым, особенно с учетом того, что разработанный продукт может в дальнейшем приносить доход от лицензионных отчислений, обновлений и дополнительных модулей. Таким образом, проект обладает достаточной инвестиционной привлекательностью и может быть рекомендован к реализации

.

					ПГУ 1.090501.014.001 ПЗ	Лист
						72
Изм.	Лист	№ докум.	Подпись	Дата		

6 Безопасность жизнедеятельности

При разработке мобильных приложений сотрудник выполняет свою работу преимущественно сидя за своим рабочим местом. Соответственно организацию рабочего места в данном случае будет регулировать ГОСТ 12.2.032-78 «Система стандартов безопасности труда. Рабочее место при выполнении работ сидя. Общие эргономические требования» [15].

Основные положения:

Конструкцией рабочего места должно быть обеспечено выполнение трудовых операций в пределах зоны досягаемости моторного поля.

Высота рабочей поверхности при организации рабочего места для работы данной специфики, выполняемой женщинами, – 630 мм, а высота сидения – 400 мм.

При работе двумя руками органы управления размещают с таким расчетом, чтобы не было перекрещивания рук.

Очень часто используемые средства отображения информации, требующие точного и быстрого считывания показаний, следует располагать в вертикальной плоскости под углом $\pm 15^\circ$ от нормальной линии взгляда и в горизонтальной плоскости под углом $\pm 15^\circ$ от сагиттальной плоскости.

Также немаловажным является правильное освещение рабочего места, данные стандарты регулирует СП 52.13330.2016 «Естественное и искусственное освещение» [16].

При выполнении выпускной квалификационной работы правовых и организационных нарушений по указанным требованиям не было выявлено, рабочее место было оборудовано согласно всем нормам и правилам.

Условия труда, в которых разрабатывается мобильное приложение для проектирования рациона питания, и устройства, с помощью которых осуществляется разработка, могут спровоцировать появление вредных и опасных факторов производства.

Согласно «ГОСТ 12.0.003-2015 Система стандартов безопасности труда (ССБТ). Опасные и вредные производственные факторы. Классификация», при выполнении работ на персональном компьютере воздействуют факторы, представленные в таблице 16. [17].

Таблица 16 – Возможные опасные и вредные производственные факторы на рабочем месте разработчика мобильных приложений

Факторы (ГОСТ 12.0.003-2015)	Нормативные документы
Отсутствие или недостатков необходимого искусственного освещения	СП 52.13330.2016 Естественное и искусственное освещение. Актуализированная редакция СНиП 23-05-95*
Аномальные микроклиматические параметры воздушной среды на местонахождении рабочего	СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания»
Нервно-психические перегрузки, связанные с монотонностью труда	Трудовой кодекс Российской Федерации от 30.12.2001 №197-ФЗ (ред. от 25.02.2022)
Нервно-психические нагрузки, связанные с перенапряжением зрительных анализаторов	СанПиН 1.2.3685-21 Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания
Производственные факторы, связанные с электрическим током, вызываемым разницей электрических потенциалов, под действие которого попадает рабочий	ГОСТ 12.1.038-82 Система стандартов безопасности труда (ССБТ). Электробезопасность. Предельно допустимые уровни напряжений прикосновения и токов

Недостаточная освещенность оказывает большую нагрузку на зрение и является одной из причин нарушения зрительной функции. Искусственное освещение в помещениях для эксплуатации персонального компьютера должно осуществляться системой общего равномерного освещения. В случаях преимущественной работы с персональным компьютером следует применять

системы комбинированного освещения. Окна в помещениях, где эксплуатируется вычислительная техника, преимущественно должны быть ориентированы на север и северо-восток.

Согласно СП 52.13330.2016 [16], зрительную работу разработчика программного обеспечения можно характеризовать как работу разряда Б – высокой точности (наименьший эквивалентный размер объекта различения составляет 0,3-0,5 мм), подразряда 1 (относительная продолжительность зрительной работы при направлении зрения на рабочую поверхность не менее 70%). В таблице 17 представлены требования к освещению рабочего помещения для указанного разряда.

Таблица 17 – Требования к освещению рабочего помещения для разряда Б1

Искусственное освещение				Естественное освещение	
Освещенность на рабочей поверхности системы общего освещения, лк	Цилиндрическая освещенность, лк	Объединенный показатель дискомфорта, не более	Коэффициент пульсации освещенности, Кп, %, не более	Коэффициент естественной освещенности, %, при	
				верхнем или комбинированном	боковом
300	100	21	15	3	1

Если следовать приведенным мерам, то удастся минимизировать вероятность нарушения зрительной функции.

Работа программиста тесно связана с малоподвижной, однотипной деятельностью, что является показателем монотонного труда. При большом количестве монотонной работы у рабочего могут наблюдаться следующие негативные последствия [14]:

- бессонница;
- депрессия;
- сонливость;
- повышенная утомляемость.

Для снижения уровня монотонности работы разработчика программного обеспечения следует принимать следующие меры:

- во время рабочего дня делать регулярные перерывы;
- вне рабочего времени стоит проводить время на свежем воздухе;
- стараться организовывать свое рабочее время таким образом, чтобы не заниматься одной задачей на протяжении длительного времени.

Работа на ПК сопровождается постоянным и значительным напряжением функций зрительного анализатора. Одной из основных особенностей является иной принцип чтения информации, чем при обычном чтении. Допустимые уровни ультрафиолетового излучения, создаваемые изделиями, в соответствии с СанПиНом 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания» указаны в таблице 18 [18].

Таблица 18 – Допустимые уровни ультрафиолетового излучения

Вид изделий	Спектральный диапазон длин волн, нм	Допустимая интенсивность облучения, Вт/м ²
Экраны телевизоров, видеомониторов, осциллографов измерительных и других приборов, средств отображения информации с визуальным контролем	Свыше 315 до 400	Не более 0,1
	Свыше 280 до 315	Не более 0,0001
	От 200 до 280	Не допускается

В результате выполнения работы были изучены правовые и организационные вопросы обеспечения безопасности, правовые нормы трудового законодательства и требования к организации рабочей зоны, в которой проводилась разработка приложения. Были проанализированы вредные и опасные факторы, которые могут возникнуть при разработке и эксплуатации проектируемого решения, и предложены мероприятия по защите разработчика от их действия. Проанализирован характер воздействия проектируемого

решения на окружающую среду и чрезвычайные ситуации, которые могут возникнуть при разработке и эксплуатации мобильного приложения.

					ПГУ 1.090501.014.001 ПЗ	Лист
						77
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В рамках данной дипломной работы была поставлена цель разработки мобильного приложения для работы с электронными компонентами. Исходя из актуальности данной задачи и необходимости повышения эффективности работы и уровня удовлетворенности специалистов компании, разработка такого приложения имеет высокую практическую значимость.

В рамках работы был проведен анализ бизнес-процессов предприятия, включающий описание самого предприятия и выбор CASE-средств для описания бизнес-процессов. Также был проведен анализ модели текущих бизнес-процессов с использованием диаграмм, что позволило более детально изучить их особенности и выявить потенциальные улучшения.

На основе проведенного анализа были сформулированы требования к мобильному приложению для работы с электронными компонентами. Кроме того, был проведен обзор и анализ аналогов существующих мобильных приложений для учета заявок, что позволило выявить их преимущества и недостатки.

На основе сформулированных требований и проведенного анализа была поставлена задача на разработку мобильного приложения, произведено проектирование приложения, выбрана платформы реализации и средства разработки, а также было произведено проектирование модели данных. Архитектура разработанного мобильного приложения основана на стандартных компонентах Android.

Реализация мобильного приложения выполнена с использованием выбранной платформы и средств разработки. Проведено тестирование, в результате которого была проверена его функциональность, стабильность и соответствие поставленным требованиям.

В результате работы была достигнута поставленная цель – разработано мобильное приложение для работы с электронными компонентами.

Разработанное приложение позволяет упростить и улучшить процесс учета и обработки заявок, повышая эффективность работы и уровень удовлетворенности клиентов компании.

					ПГУ 1.090501.014.001 ПЗ	Лист
						79
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. Android Studio. [Электронный ресурс] / Режим доступа: URL: <https://developer.android.com/studio/intro> (дата обращения 04.01.2026).
2. Diagrams.net. [Электронный ресурс] / Режим доступа: URL: <https://en.wikipedia.org/wiki/Diagrams.net> (дата обращения 02.01.2026).
3. Eeles P. Capturing architectural requirements //IBM Rational developer works. – 2005.
4. Firebase for Android. [Электронный ресурс] / Режим доступа: URL: <https://firebase.google.com/docs/android/setup?authuser=0&hl=en> (дата обращения 27.01.2026).
5. Goadrich M. H., Rogers M. P. Smart smartphone development: iOS versus Android //Proceedings of the 42nd ACM technical symposium on Computer science education. – 2011. – С. 607-612.
6. Mobile Operating System Market Share. [Электронный ресурс] / Режим доступа: URL: <https://gs.statcounter.com/os-market-share/mobile/worldwide> (дата обращения 15.01.2026).
7. Nimodia C., Deshmukh H. R. Android operating system //Software Engineering. – 2012. – Т. 3. – №. 1. – С. 10.
8. OneSignal SDK. [Электронный ресурс] / Режим доступа: URL: <https://documentation.onesignal.com/docs/android-sdk-setup> (дата обращения 17.01.2026).
9. Афанасьев А. В. Актуальность языка программирования java в разработке мобильных приложений в 2020 году //Информационные технологии в процессе подготовки современного специалиста. – 2020. – С. 25- 29.
10. Афанасьев А. Н., Войт Н. Н. Разработка компонентно-сервисной платформы обучения: диаграммы классов программного компонента сценария на UML-языке //Вестник Ульяновского государственного технического университета. – 2012. – №. 2 (58). – С. 32-36.

11. Вязовик Н., Жилин Е. Программирование на Java //М.: Интуит. ру. – 2003.
12. Зиятдинова А., Староверова Н. А. Аналитический обзор и сравнение возможностей операционных систем для мобильных устройств //Фундаментальные исследования. – 2015. – №. 9-2. – С. 227-231.
13. Иванов В. В., Лубова Е. С., Черкасов Д. Ю. Аутентификация и авторизация //Проблемы современной науки и образования. – 2017. – №. 2 (84). – С. 31-33.
14. Микляев И. А., Ундозерова А. Н., Кудасева М. В. Универсальная логическая модель базы данных //Arctic Environmental Research. – 2010. – №. 1. – С. 93-98.
15. Об учреждении «Арена». [Электронный ресурс] / Режим доступа: URL: <https://www> (дата обращения 24.01.2026).
16. Оптимизация бизнес-процессов предприятия. [Электронный ресурс] / Режим доступа: URL: <https://piter-consult.ru/home/Articles/Simply-about-the-difficult/Let-entrust-business-processes.html> (дата обращения 24.01.2026).
17. Тарасов С. В. СУБД для программиста. Базы данных изнутри. – 2015.
18. Юрченко А. Н. Функциональное тестирование как одна из методологий тестирования программного обеспечения //Вестник современных исследований. – 2018. – №. 1.1. – С. 155-156.

Приложение А

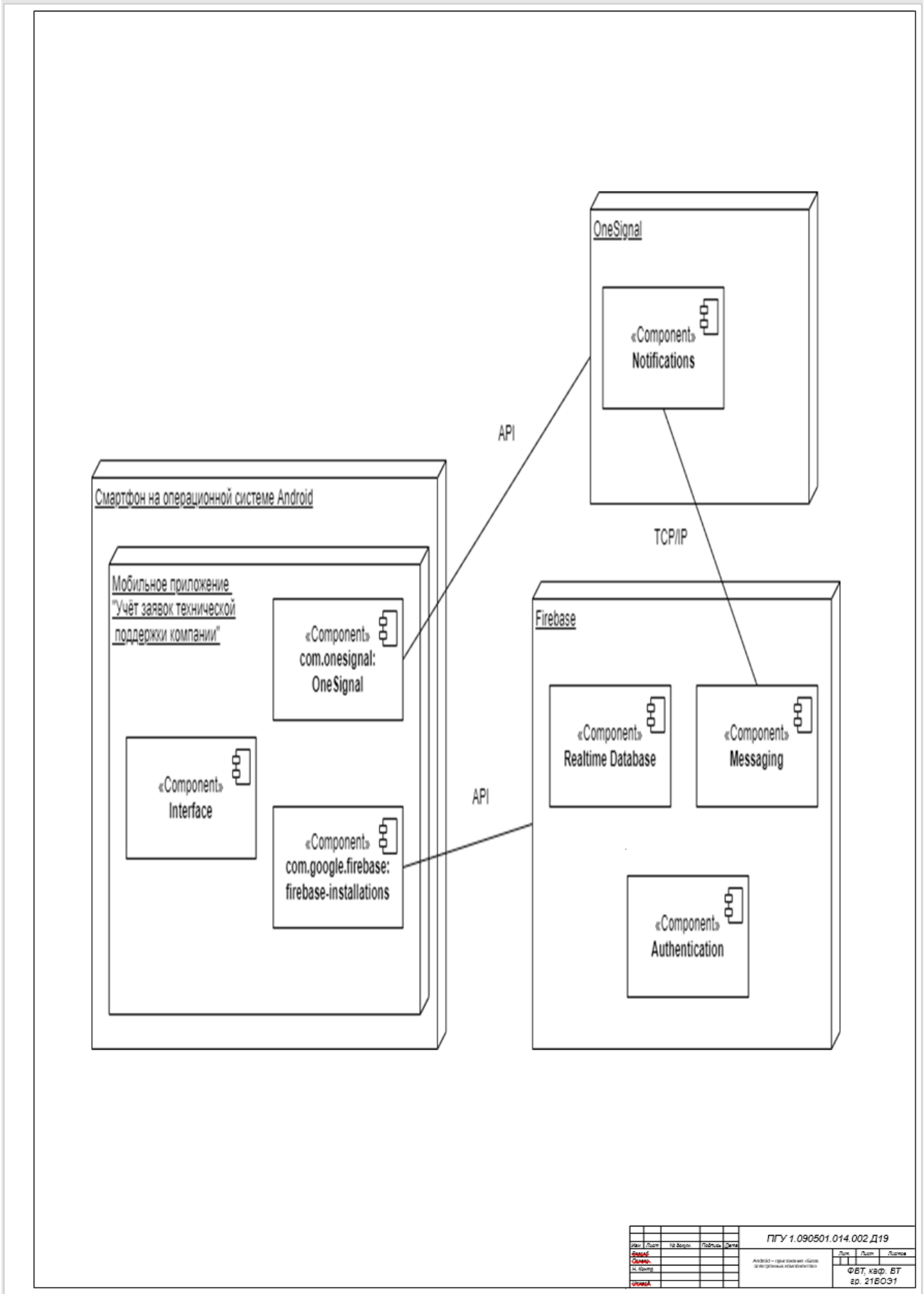
(обязательное)

Anroid – приложение «База электронных компонентов»

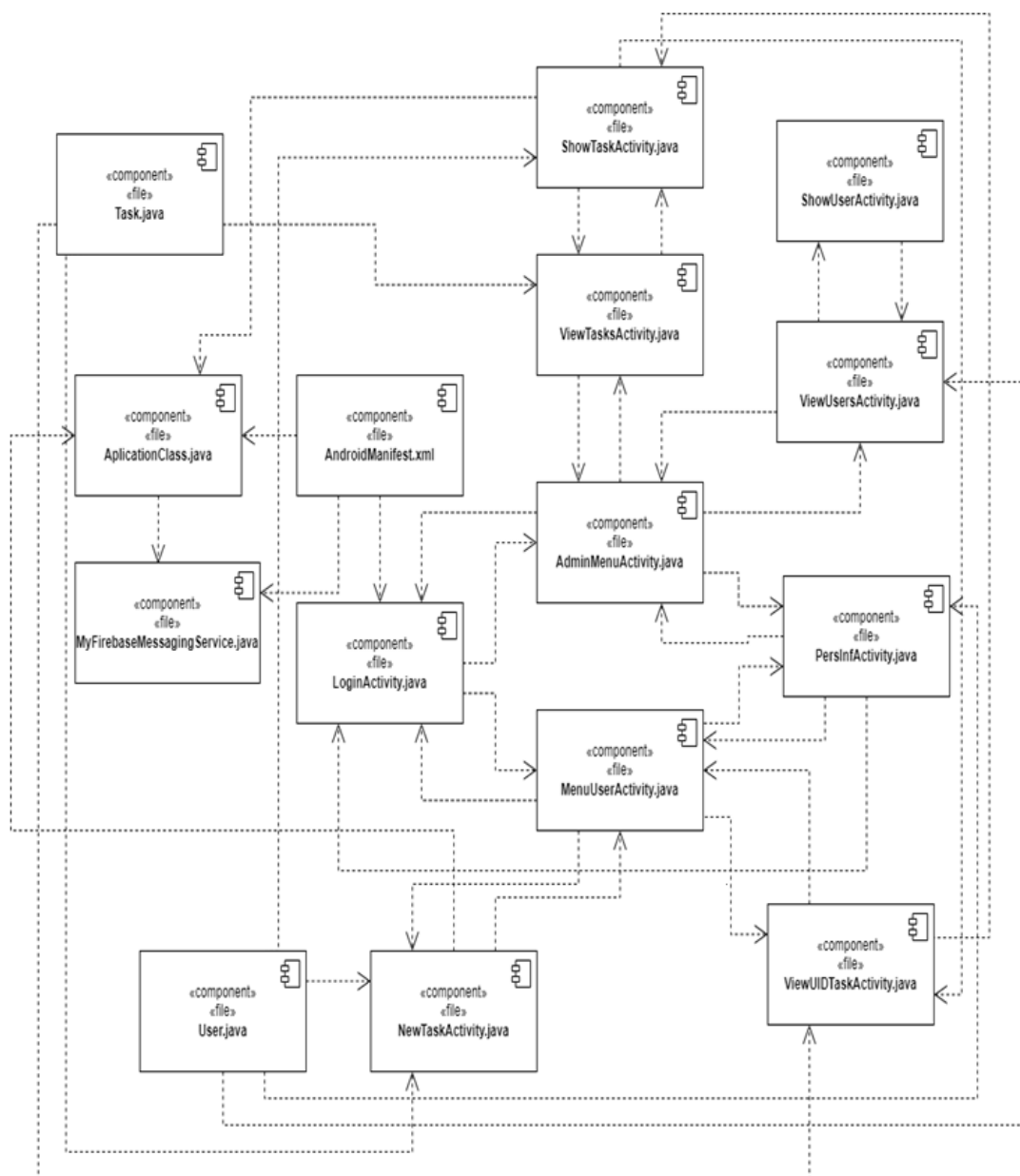
UML–диаграммы

					ПГУ 1.090501.014.001 ПЗ	Лист
						82
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1 – Диаграмма развертывания



Приложение А.2 – Диаграмма компонентов



ПГУ 1.090501.014.003 Д18			
Имя	Лист	№ докум.	Подпись
Создан			
Изменен			
Сл. Измен.			
Утвержден			

Android — это товарный знак
и зарегистрированный товарный знак

ФБТ, кат. БТ
ед. 216031

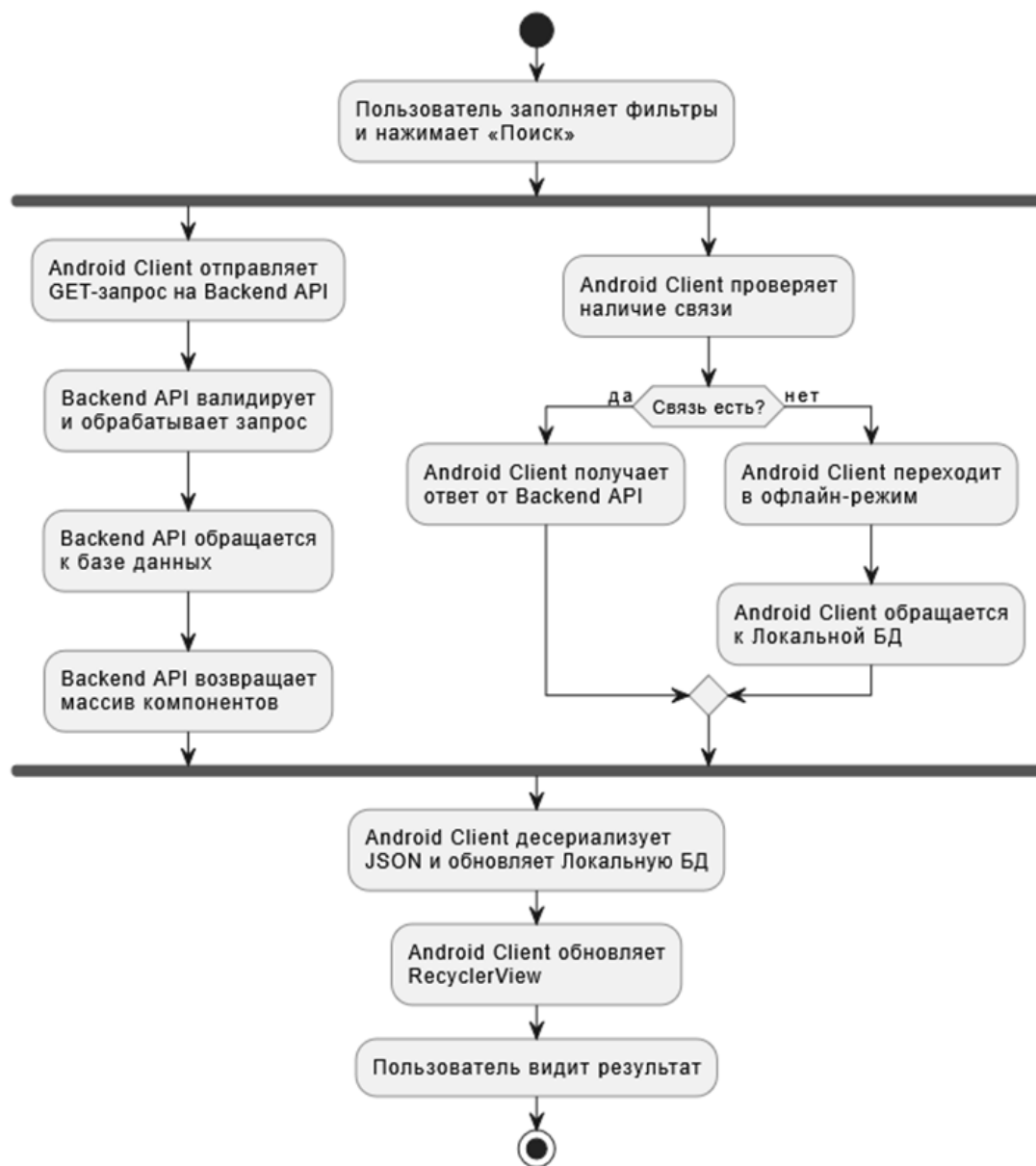
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.014.001 ПЗ

Лист

84

Приложение А.3 – Диаграмма последовательности



ПГУ 1.090501.014.004 Д14				
Имя	Фамилия	Имя	Фамилия	Подпись
Степанов	Иван	Иванов	Иван	Иванов
М. Козлов				
Иванов				

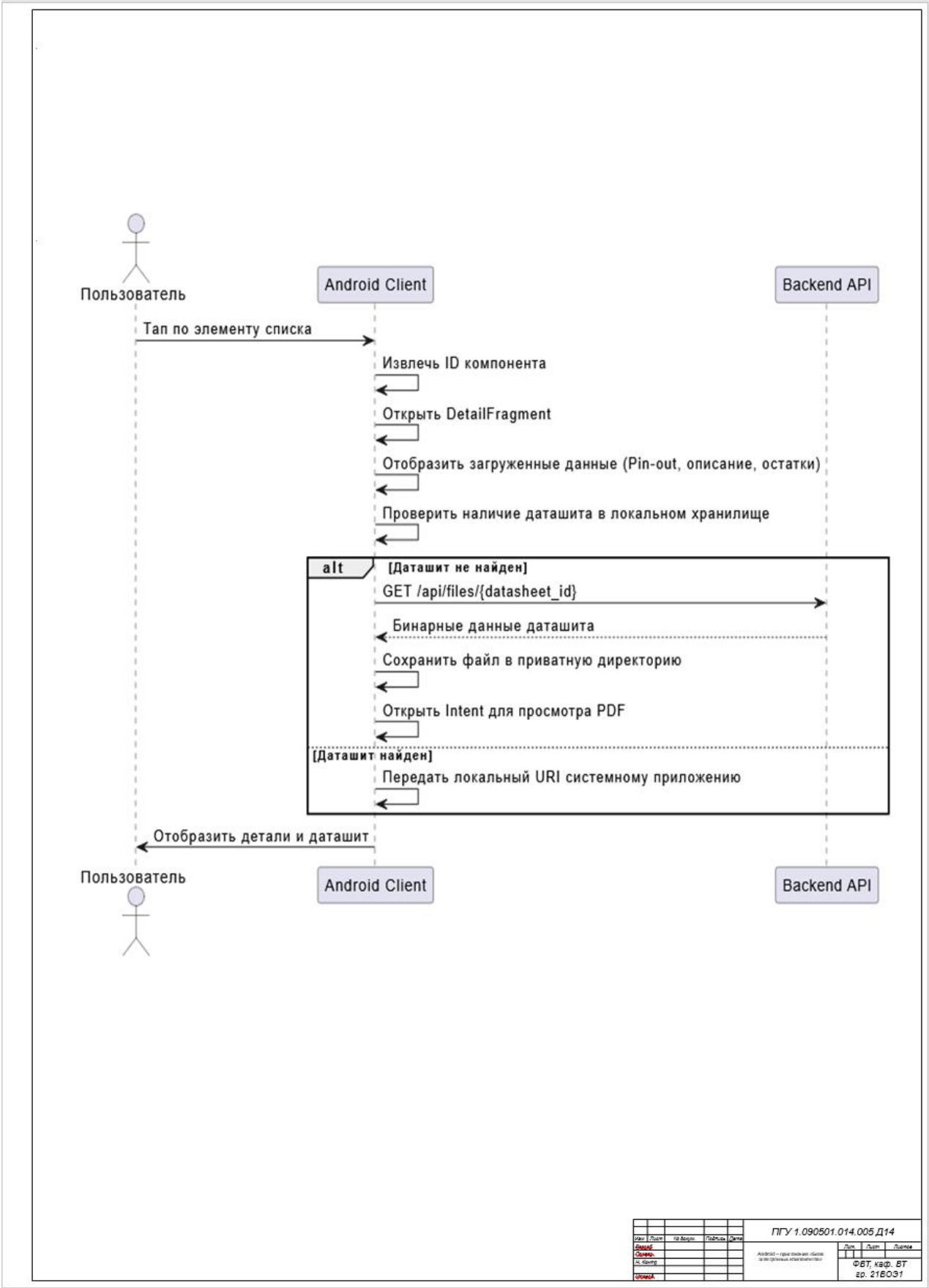
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.014.001 ПЗ

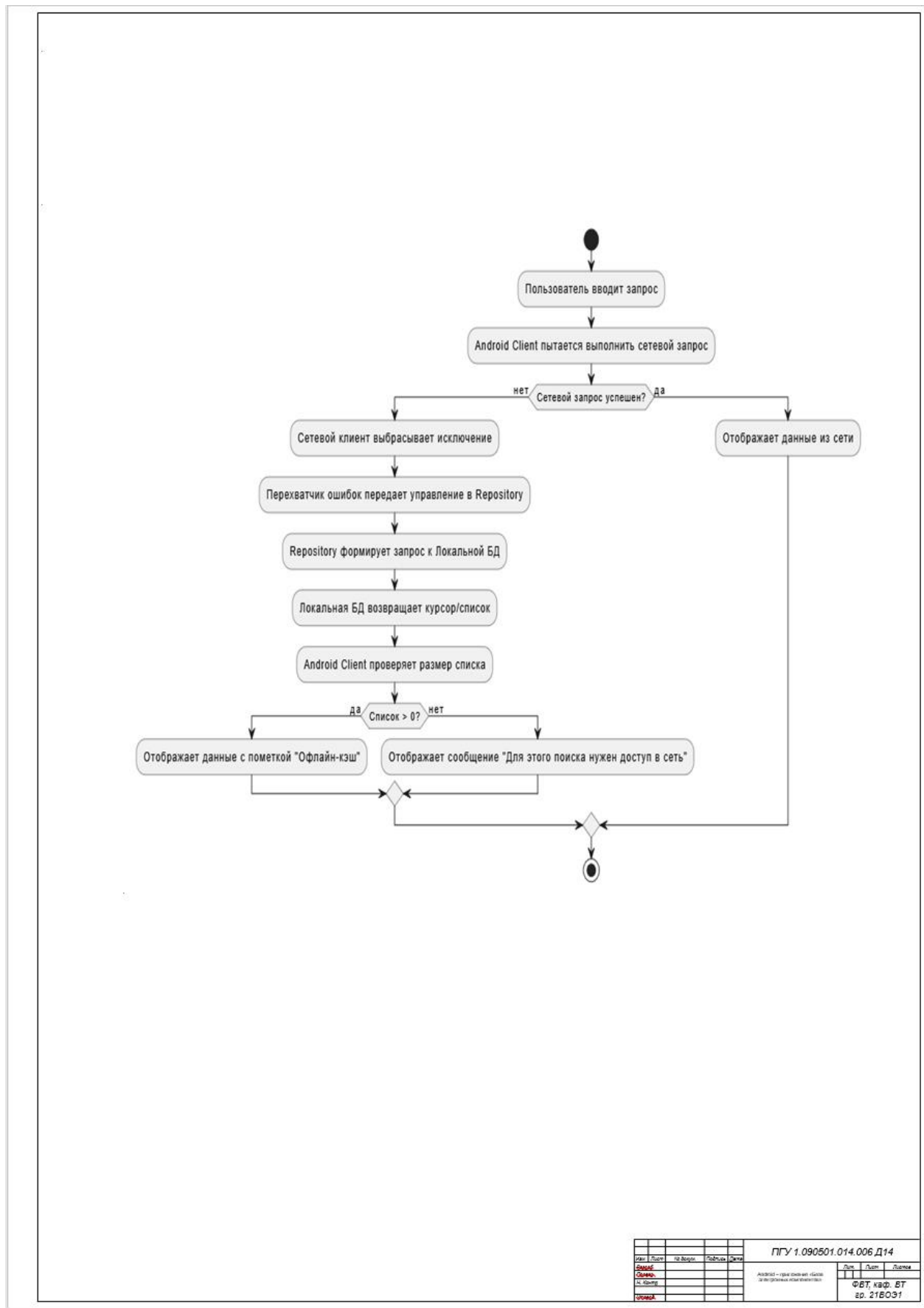
Лист

85

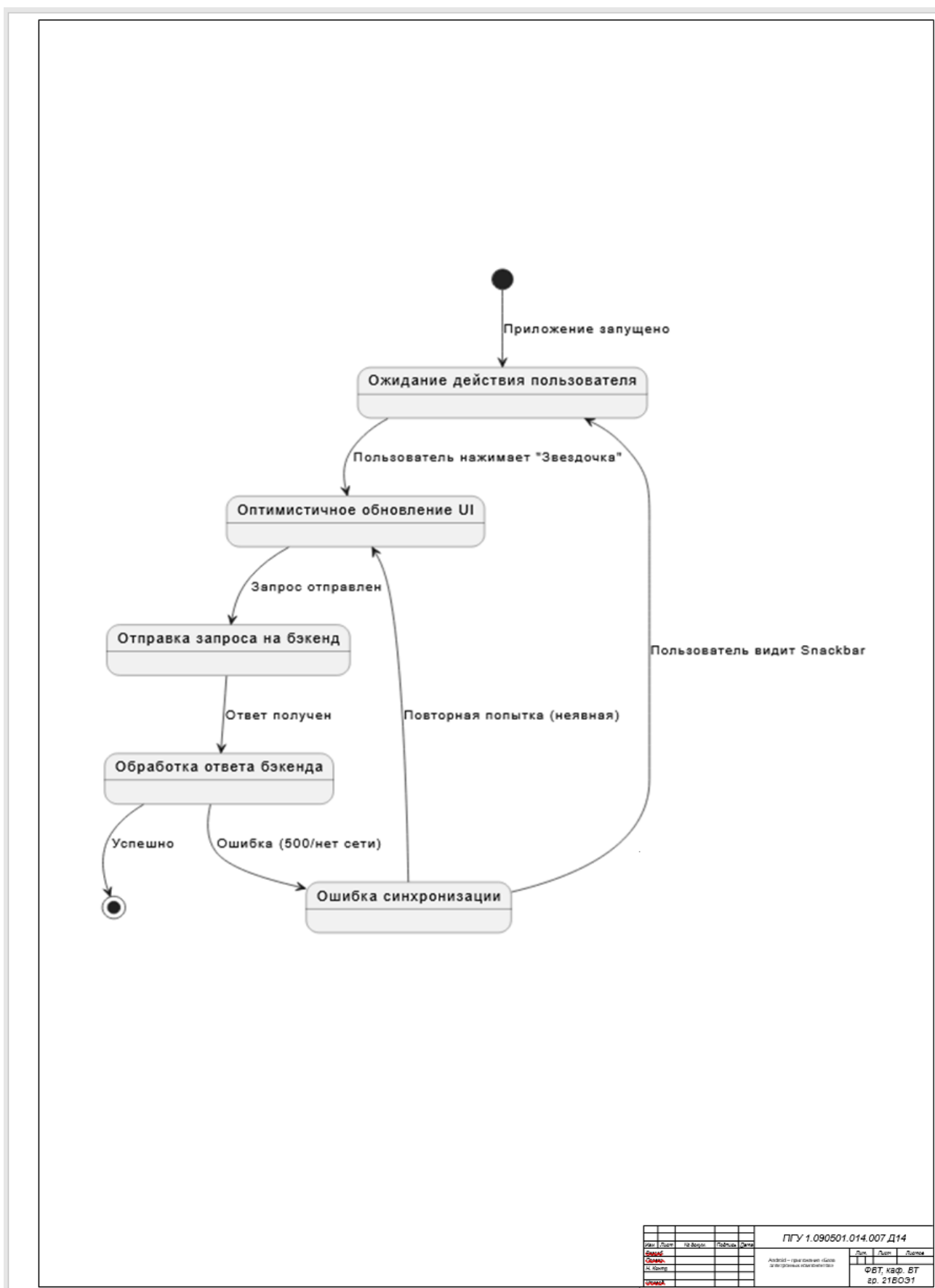
Приложение А.4 – Диаграмма последовательности: просмотр карточки компонента и даташита



Приложение А.5 – Диаграмма последовательности: Работа в офлайн-режиме (Fallback)



Приложение А.6 – Диаграмма последовательности: добавление компонента в
«Избранное» и синхронизация



ПГУ 1.090501.014.007 Д14				
Имя	Фамилия	Имя	Фамилия	Имя
Степанов	Иван	Степанов	Иван	Степанов
И. Степанов				
Степанов				

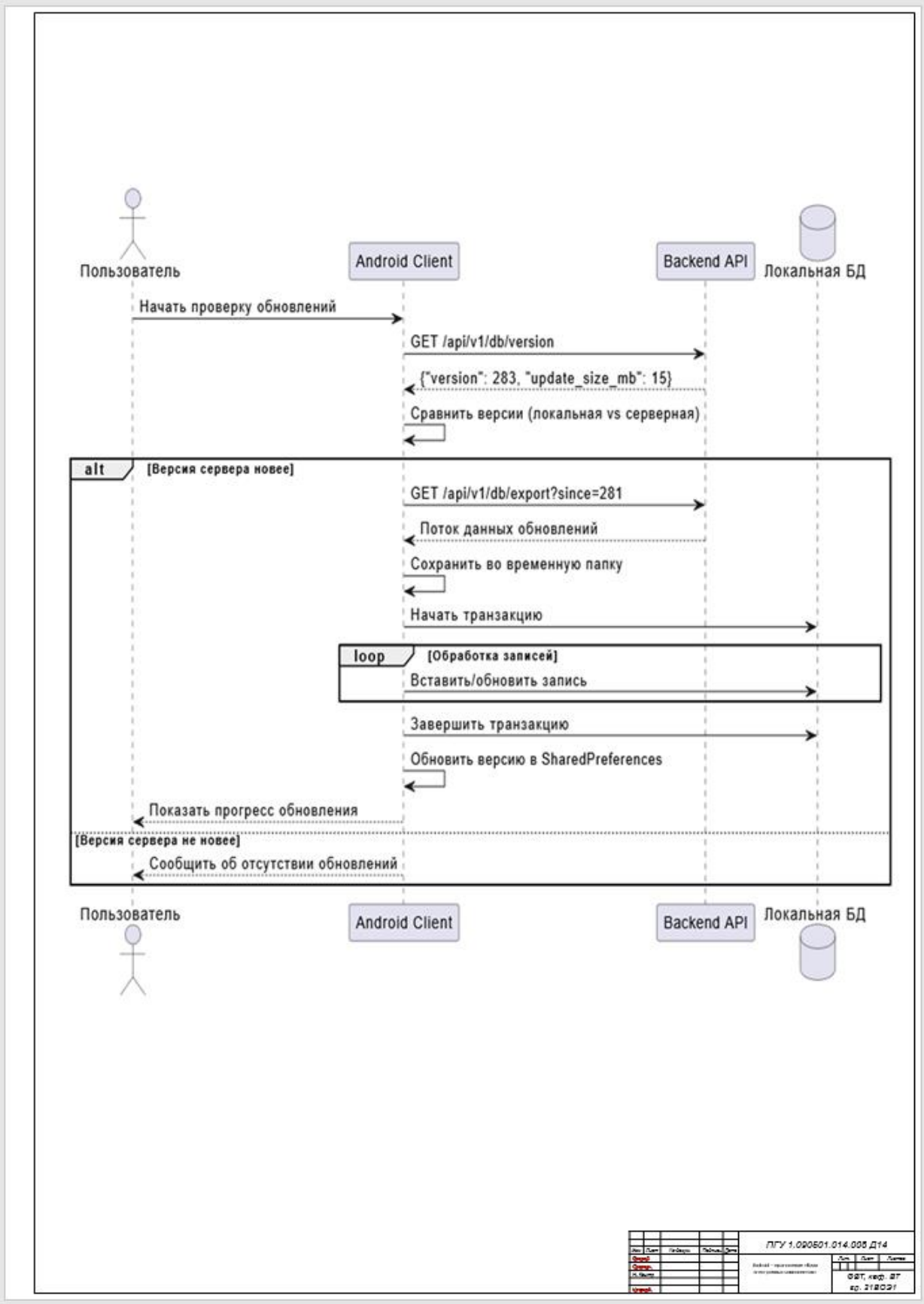
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.014.001 ПЗ

Лист

88

А.7 – Диаграмма последовательности: обновление справочной базы (Миграция данных)



Приложение Б

(обязательное)

Android – приложение «База электронных компонентов»

Листинги программы

					ПГУ 1.090501.014.001 ПЗ	Лист
						90
Изм.	Лист	№ докум.	Подпись	Дата		

Листинг А.1 – Фрагмент кода отображения списка пользователей

```
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import java.util.ArrayList;
import java.util.List;

public class ViewUsersActivity extends AppCompatActivity {
    private ListView listView;
    private ArrayAdapter<String> adapter;
    private List<String> listData;
    private List<User> listTemp;
    private DatabaseReference userDatabaseReference;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.viewuserslayout);
        init();
        getDataFromDB();
        setOnClickItem();
    }

    private void init() {
        {
            listView = findViewById(R.id.listView);
            listData = new ArrayList<>();
            listTemp = new ArrayList<>();
            adapter = new ArrayAdapter<>(this, android.R.layout
            .simplelistitem1,
            listData);
            listView.setAdapter(adapter);
            userDatabaseReference =
            FirebaseDatabase.getInstance().getReference().child("Users");
        }

        private void getDataFromDB() {
            {
                ValueEventListener vListener = new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                        {
                            if(listData.size() > 0)listData.clear();

```

					ПГУ 1.090501.014.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		91

```

if(listTemp.size() > 0)listTemp.clear();
for(DataSnapshot ds : dataSnapshot.getChildren())
{
User user = ds.getValue(User.class);
assert user != null;
listData.add(user.getEmail() + '\n' + user.getFIO());
listTemp.add(user);
}
adapter.notifyDataSetChanged();

}
@Override
public void onCancelled(@NonNull DatabaseError error) {
}
};
userDatabaseReference.addValueEventListener(vListener);
}
private void setOnClickItem()
{
ListView.setOnItemClickListener(new AdapterView
.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View view, int
position,
long id) {
User user = listTemp.get(position);
Intent i = new Intent(ViewUsersActivity.this,
ShowUserActivity.class);
i.putExtra("userEmail", user.getEmail());
i.putExtra("userFIO", user.getFIO());
i.putExtra("userDolz", user.getDolz());
i.putExtra("userKab", user.getKab());
i.putExtra("userTelNum", user.getTelNumb());
i.putExtra("userToken", user.getToken());
i.putExtra("userPlayerID", user.getPlayerID());
startActivity(i);
}
});
}

public void onClickNazad(View view)
{
FirebaseUser currentUser = FirebaseAuth.getInstance
().getCurrentUser();
String userName = currentUser.getEmail();
String admin = "admin@yandex.ru";
if (userName.equals(admin))
{
Intent i = new Intent(ViewUsersActivity.this,
AdminMenuActivity.class);
startActivity(i);
}
}

```

					ПГУ 1.090501.014.001 ПЗ	Лист
						92
Изм.	Лист	№ докум.	Подпись	Дата		

```

    }
    else
83
    {
        Intent i = new Intent(ViewUsersActivity.this, MenuUserActivity
        .class);
        startActivity(i);
    }
}
}

```

Листинг А.2 – Фрагмент кода главной страницы приложения (MainActivity.java)

```

package com.example.electronicssdb;

import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.electronicssdb.adapters.ComponentAdapter;
import com.example.electronicssdb.models.Component;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {

    // UI элементы
    private EditText etSearch;
    private Button btnSearch;
    private RecyclerView rvComponents;
    private ProgressBar progressBar;

    // Адаптер и данные
    private ComponentAdapter adapter;

```

```

private List<Component> componentList;
private List<Component> searchList;

// Firebase
private FirebaseAuth mAuth;
private DatabaseReference databaseReference;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // Инициализация Firebase
    mAuth = FirebaseAuth.getInstance();
    databaseReference = FirebaseDatabase
    .getInstance().getReference();

    // Инициализация UI
    initView();
    setupRecyclerView();
    setupListeners();

    // Загрузка компонентов
    loadComponents();
}

private void initView() {
    etSearch = findViewById(R.id.etSearch);
    btnSearch = findViewById(R.id.btnSearch);
    rvComponents = findViewById(R.id.rvComponents);
    progressBar = findViewById(R.id.progressBar);
}

private void setupRecyclerView() {
    componentList = new ArrayList<>();
    searchList = new ArrayList<>();
    adapter = new ComponentAdapter(this, searchList);
    rvComponents.setLayoutManager(new LinearLayoutManager(this
));
    rvComponents.setAdapter(adapter);
}

private void setupListeners() {
    // Обработчик поиска
    btnSearch.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String query = etSearch.getText().toString
().trim().toLowerCase();
            performSearch(query);
        }
    });
}

```

```

        // Обработчик клика по компоненту
        adapter.setOnItemClickListener(new ComponentAdapter
.OnItemClickListener() {
            @Override
            public void onItemClick(Component component) {
                Intent intent = new Intent(MainActivity.this,
ComponentDetailActivity.class);
                intent.putExtra("component_id", component
.getId());
                intent.putExtra("component_name", component
.getName());
                intent.putExtra("component_type", component
.getType());
                intent.putExtra("component_manufacturer",
component.getManufacturer());
                intent.putExtra("component_specifications",
component.getSpecifications());
                intent.putExtra("component_datasheet", component
.getDatasheetUrl());
                intent.putExtra("component_price", component
.getPrice());
                startActivity(intent);
            }
        });
    }

    private void loadComponents() {
        progressBar.setVisibility(View.VISIBLE);

        databaseReference.child
("Components").addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot
snapshot) {
                componentList.clear();
                for (DataSnapshot data : snapshot.getChildren()) {
                    Component component = data.getValue
(Component.class);
                    if (component != null) {
                        component.setId(data.getKey());
                        componentList.add(component);
                    }
                }
                // Показываем все компоненты по умолчанию
                searchList.clear();
                searchList.addAll(componentList);
                adapter.notifyDataSetChanged();
                progressBar.setVisibility(View.GONE);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error)
{

```

					ПГУ 1.090501.014.001 ПЗ	Лист
						95
Изм.	Лист	№ докум.	Подпись	Дата		

```

        progressBar.setVisibility(View.GONE);
        Toast.makeText(MainActivity.this,
            "Ошибка загрузки: " + error.getMessage(),
            Toast.LENGTH_SHORT).show();
    }
});
}

private void performSearch(String query) {
    searchList.clear();

    if (query.isEmpty()) {
        searchList.addAll(componentList);
    } else {
        for (Component component : componentList) {
            if (component.getName().toLowerCase().contains
(query) ||
                component.getType().toLowerCase
().contains(query) ||
                component.getManufacturer
().toLowerCase().contains(query)) {
                searchList.add(component);
            }
        }

        adapter.notifyDataSetChanged();

        if (searchList.isEmpty()) {
            Toast.makeText(this, "Компоненты не найдены", Toast
.LENGTH_SHORT).show();
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        int id = item.getItemId();

        if (id == R.id.menu_profile) {
            startActivity(new Intent(MainActivity.this,
ProfileActivity.class));
            return true;
        } else if (id == R.id.menu_my_tasks) {
            startActivity(new Intent(MainActivity.this,
MyTasksActivity.class));
            return true;
        } else if (id == R.id.menu_logout) {

```

```

        logout();
        return true;
    } else if (id == R.id.menu_refresh) {
        loadComponents();
        return true;
    }

    return super.onOptionsItemSelected(item);
}

private void logout() {
    mAuth.signOut();
    Intent intent = new Intent(MainActivity.this,
LoginActivity.class);
    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent
.FLAG_ACTIVITY_CLEAR_TASK);
    startActivity(intent);
    finish();
}

@Override
protected void onStart() {
    super.onStart();
    // Проверка авторизации
    FirebaseUser currentUser = mAuth.getCurrentUser();
    if (currentUser == null) {
        Intent intent = new Intent(MainActivity.this,
LoginActivity.class);
        startActivity(intent);
        finish();
    }
}
}

```

Листинг А.3 – Фрагмент кода структуры базы данных в Firebase

```

{
  "Components": {
    "-N1234567890": {
      "name": "STM32F103C8T6",
      "type": "Микроконтроллер",
      "manufacturer": "STMicroelectronics",
      "specifications": "ARM Cortex-M3, 72MHz, 64KB Flash, 20KB
RAM",
      "datasheetUrl": "https://example.com/datasheet/stm32f103
.pdf",
      "price": 350.00
    },
    "-N1234567891": {
      "name": "ESP8266",
      "type": "WiFi модуль",
      "manufacturer": "Espressif",
      "specifications": "802.11 b/g/n, 32-bit, 80MHz",

```

```

        "datasheetUrl": "https://example.com/datasheet/esp8266.pdf",
        "price": 280.00
    },
    "Users": {
        "uid123": {
            "email": "user@example.com",
            "fio": "Иванов Иван Иванович",
            "dolz": "Инженер",
            "kab": "101",
            "telNumb": "+7-999-123-45-67",
            "playerID": "onesignal_id",
            "token": "fcm_token"
        }
    },
    "Tasks": {
        "-T123456": {
            "userID": "uid123",
            "tema": "Поиск аналога",
            "opis": "Нужен аналог транзистора 2N2222",
            "status": "В обработке",
            "time": "2026-06-10 14:30:00"
        }
    }
}

```

					ПГУ 1.090501.014.001 ПЗ	Лист
						98
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В

(обязательное)

Автоматизированная система психодиагностики для задач профотбора

Презентация

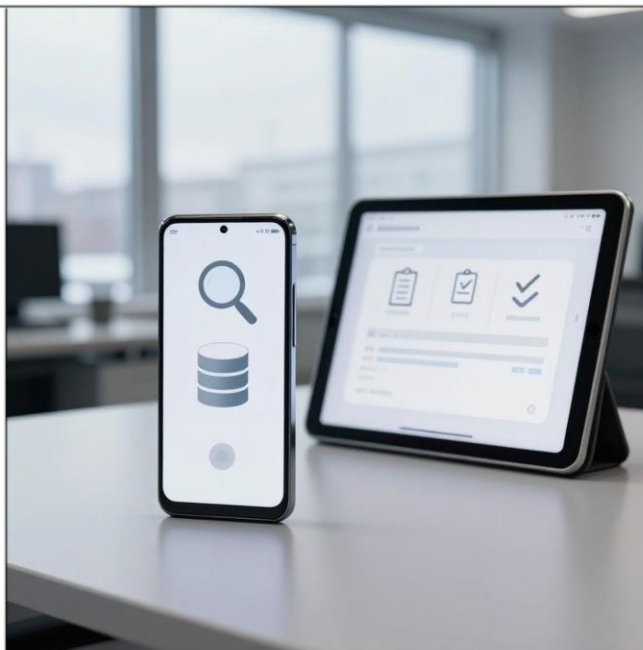
					ПГУ 1.090501.014.001 ПЗ	Лист
						99
Изм.	Лист	№ докум.	Подпись	Дата		

Дипломный проект

на тему ANDROID – ПРИЛОЖЕНИЕ «БАЗА ЭЛЕКТРОННЫХ КОМПОНЕНТОВ»

Выполнил: ст. Сетяев Артем Николаевич
Группа: 21В031

Руководитель работы: д.т.н., профессор
Митрохин М.А.



Актуальность

Подбор электронных компонентов часто требует сравнения множества источников. Предлагаемое приложение объединяет каталог, поиск, заявки и офлайн-работу, сокращая время выбора и упрощая работу инженеров и студентов.



2

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.014.001 ПЗ

Лист

100

Задачи работы

1

Анализ предметной области

Изучены типовые сценарии поиска электронных компонентов и существующие мобильные решения, чтобы определить ограничения аналогов и сформулировать требования к будущему приложению.

2

Проектирование системы

Спроектированы архитектура приложения и структура Firebase Realtime Database, обеспечивающие хранение данных, синхронизацию между устройствами и поддержку офлайн-режима.

3

Разработка клиента

Создано нативное Android-приложение на Java в Android Studio с основными экранами, навигацией и логикой работы с каталогом, профилем и заявками.

4

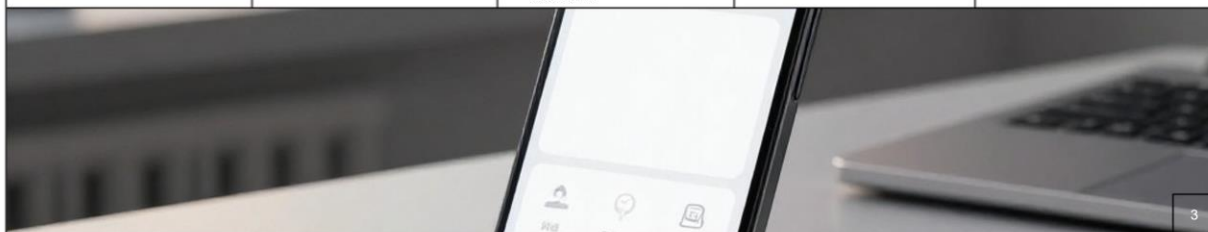
Кабинеты пользователей

Реализованы отдельные личные кабинеты для пользователя и администратора с различными правами доступа, что позволило разделить сценарии работы и повысить удобство использования.

5

Заявки и уведомления

Добавлен модуль создания и отслеживания заявок на подбор компонентов, а также push-уведомления об изменении статуса и локальное кэширование данных.



3 0 0 0 0 0

Сравнение с существующими решениями

Electrodoc

Сильная сторона решения — большая справочная база и офлайн-режим. Однако отсутствуют заявки на подбор и синхронизация данных между устройствами.

EveryCircuit

Приложение удобно для визуализации и симуляции электрических схем, но оно не ориентировано на поиск компонентов и работу с заявками.

Droid Tesla

Решение отличается высокой точностью расчетов и подходит инженерам, но имеет более сложный интерфейс и не поддерживает уведомления.

Потребность рынка

Практике требуется приложение, которое одновременно дает каталог компонентов, простой поиск и систему заявок без лишней сложности.

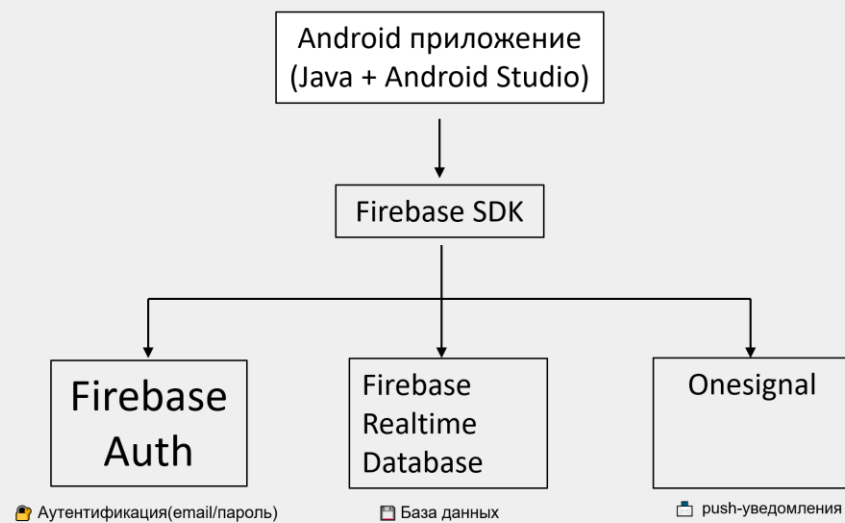
Вывод для проекта

Именно сочетание справочника, офлайн-доступа и управления заявками стало основой разработанного решения.

3 0 0 0 0 0

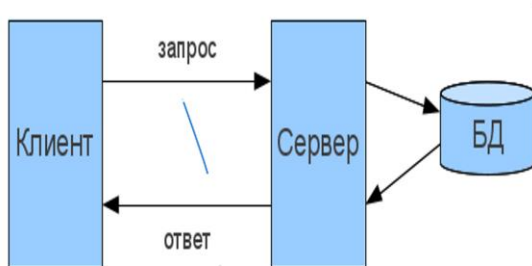
4

Архитектура и технологический стек



5

Клиент-серверная архитектура



Клиентская часть:

- Android-приложение (Java, Android Studio)
- Отправка HTTP-запросов через Firebase SDK / Volley
- Отображение данных (RecyclerView, CardView)

Серверная часть:

- Firebase Authentication — управление пользователями
- Firebase Realtime Database — хранение данных
- OneSignal — push-уведомления



6

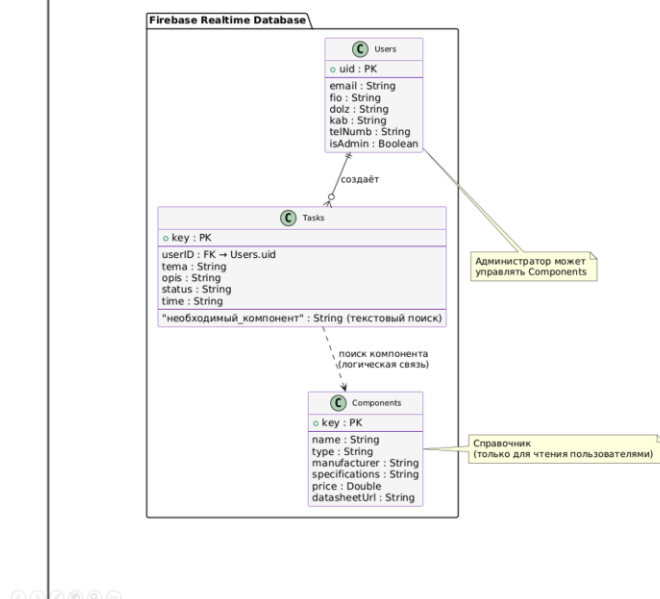
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.014.001 ПЗ

Лист

102

Проектирование базы данных



Узел	Основные поля	Назначение
Components	Название, тип, цена	Каталог компонентов
Users	Email, ФИО, телефон	Профили пользователей
Tasks	Тема, статус, ID	Заявки на подбор

Модель данных проста и разделена по сущностям, что удобно для синхронизации и офлайн-доступа.

Таблица показывает основные узлы базы и назначение хранимых полей.

7

Роли пользователей в системе

Пользователь

После регистрации и входа пользователь ищет компоненты по параметрам, создает заявку на подбор, просматривает ее статус и при необходимости редактирует личные данные.

Каталог и заявки

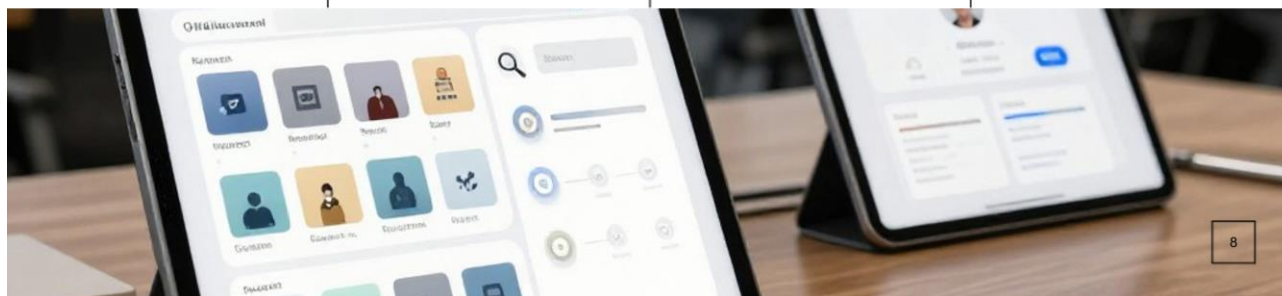
Для обычного пользователя доступны только рабочие сценарии поиска, формирования обращения и контроля собственных заявок без доступа к административным разделам.

Администратор

Администратор просматривает всех зарегистрированных пользователей, открывает их карточки, контролирует общий поток заявок и меняет их статусы в зависимости от результата обработки.

Разделение прав

Такое распределение функций исключает смешение ролей и делает интерфейс понятным: каждый участник системы видит только те инструменты, которые ему необходимы.



8

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.014.001 ПЗ

Лист

103

Функционал личного кабинета пользователя

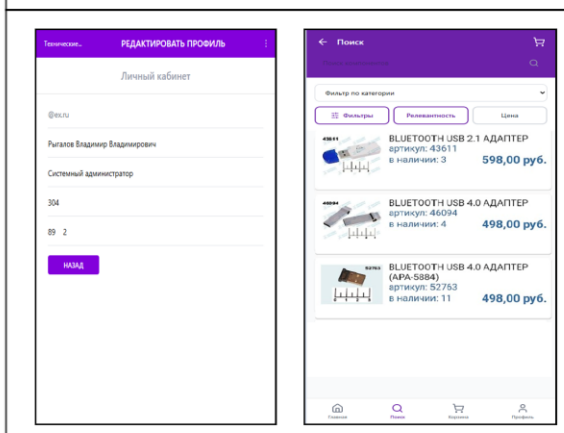


Таблица отражает основные разделы пользовательского кабинета и их назначение.

Кабинет пользователя представляет актуализацию персональных данных в одном месте.

Раздел	Функции	Содержимое
Каталог	Поиск и фильтры	Название, тип, производитель
Мои заявки	Создание и статус	Тема, описание, статус
Профиль	Редактирование данных	ФИО, должность, кабинет, телефон

9

Функционал личного кабинета администратора

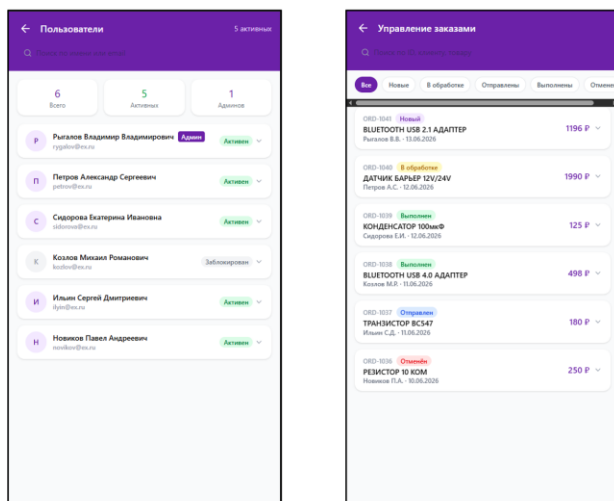


Таблица показывает ключевые административные операции и их результат.

Администратор управляет системой централизованно, а изменение статуса сразу отражается для пользователя через уведомление.

Раздел	Действие	Результат
Пользователи	Просмотр карточек	Контроль аккаунтов
Все заявки	Изменение статуса	Новый/в обработке
Уведомления	OneSignal	Сообщение пользователю

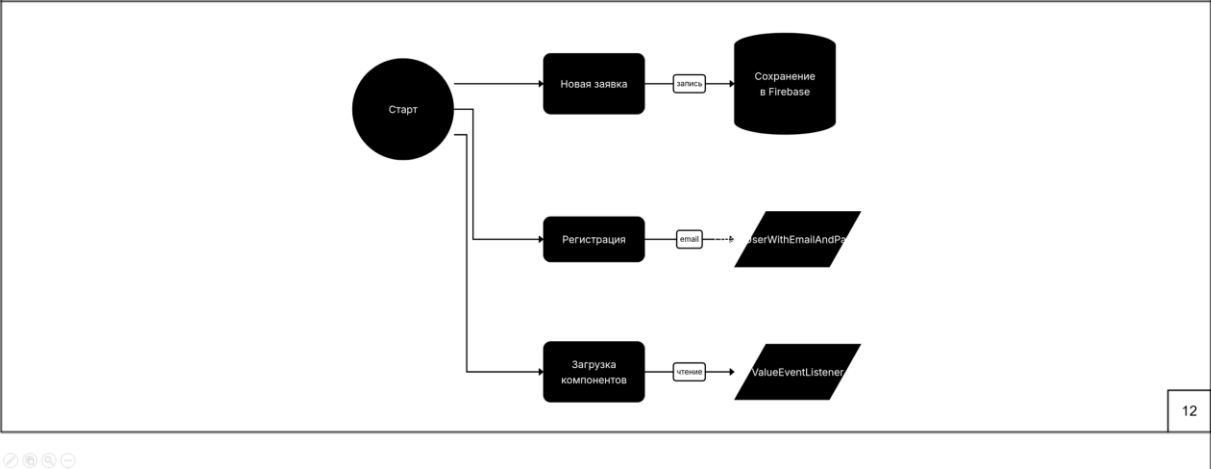
10

Ключевые облачные сервисы

	Firestore Authentication Отвечает за регистрацию и вход по email и паролю, проверяет учетные данные на стороне сервера и выдает токен доступа для защищенной работы с приложением.
	Realtime Database Используется как основное хранилище приложения; данные доступны для синхронизации в реальном времени и продолжают работать при временном отсутствии сети.
	OneSignal Сервис доставляет push-уведомления пользователю при изменении статуса заявки, обеспечивая быстрый отклик и прозрачность обработки обращения.
	Сокращение бэкенда Применение готовой облачной платформы позволило не разрабатывать отдельный сервер, сосредоточиться на логике приложения и упростить дальнейшее сопровождение.

Сценарии работы клиентского приложения

Три ключевых процесса: регистрация пользователя, загрузка каталога из Firebase и создание заявки с сохранением в базу данных.



Безопасность и разграничение доступа

1 Проверка учетных данных

Вход выполняется через Firebase Authentication: пользователь вводит email и пароль, после чего сервер проверяет их корректность и выдает токен доступа для дальнейших авторизованных запросов.

2 Ограничение прав пользователя

Обычный пользователь получает доступ только к своему профилю и собственным заявкам, поэтому видит лишь персональные данные и может работать только в пределах своей учетной записи.

3 Проверка учетных данных и разграничение доступа

Вход выполняется через Firebase Authentication: приложение отправляет email и пароль, затем сервис подтверждает личность пользователя и выдает JWT. Без корректного токена доступ к API и экрану заявок невозможен.

4 Защита передачи и восстановление

Передача данных выполняется по HTTPS, а восстановление пароля организовано через отправку ссылки на email. Такой подход снижает риск несанкционированного доступа к системе.

13

⏮ ⏪ ⏩ ⏭ 🔍 ⏴ ⏵ ⏸ ⏹

Заключение

Итог работы:

1. Разработано готовое к использованию Android-приложение.
2. Решены проблемы поиска компонентов и заявок на подбор.
3. Реализованы личные кабинеты пользователя и администратора.
4. Поддержка офлайн-режима с кэшированием.

Перспективы развития:

- 📦 Расширение базы до тысяч компонентов
- 🔍 Сканирование маркировки через камеру (OCR)
- 🛒 Интеграция с онлайн-магазинами для заказа компонентов
- 🍏 Создание версии для iOS на Swift

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.014.001 ПЗ

Лист

106