МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования
«Гомельский государственный университет
имени Франциска Скорины»

Факультет физики и информационных технологий

Кафедра общей физики

Допущена к защите
Зав. кафедрой _____Е.Б. Шершнев

"____"_____2024 г.

# Разработка платформы для пиринговой аренды вещей

Дипломная работа

Исполнитель:
студент группы МС–42 _____ В.Ю. Шевцов

Научный руководитель:
старший преподаватель
кафедры общей физики _____ В.В. Грищенко

Рецензент:
старший преподаватель
кафедры АСОИ _____ В.Н. Кулинченко

Гомель 2024

# РЕФЕРАТ

Дипломная работа 49 страниц, 32 рисунка, 15 источников, 2 приложения.

Ключевые слова: пиринговая аренда, аренда вещей, мобильное приложение, Firebase, React Native, Google Maps API, тестирование, оценка масштабируемости.

Объект исследования: платформа для пиринговой аренды вещей, включающая инструменты для подачи объявлений, оформления сделок, общения с пользователями.

Метод исследования: анализ существующих решений и современных трендов в пиринговой аренде, проектирование архитектуры программного обеспечения, язык программирования JavaScript, фреймворк React Native, тестирование и оценка эффективности платформы.

Цель дипломной работы: разработка платформы для пиринговой аренды вещей, которая обеспечит удобное и безопасное взаимодействие между пользователями.

В данной дипломной работе рассмотрены вопросы разработки платформы для пиринговой аренды вещей. Были изучены основные концепции и принципы организации систем пиринговой экономики, проанализированы существующие решения в этой сфере. На основе проведенного анализа разработана структурная модель платформы для аренды вещей, определены ее ключевые компоненты и функциональные возможности. Особое внимание уделено вопросам обеспечения безопасности транзакций, репутационных механизмов и защиты прав пользователей. В работе представлены результаты проектирования архитектуры платформы, выбора технологического стека, разработки прототипа и проведения его тестирования. По результатам проведенной работы были подготовлены и опубликованы тезисы, что свидетельствует о практической значимости и научной новизне полученных результатов.

СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

В современном мире понятие совместного потребления набирает все большую популярность. Одним из ярких его примеров является пиринговая аренда вещей, которая позволяет людям делиться своими ресурсами и использовать вещи совместно, вместо того чтобы покупать их. Это не только экономит деньги, но и способствует более рациональному использованию ресурсов, что актуально в условиях растущей экологической осведомленности общества.

Пиринговая аренда вещей представляет собой систему, в которой люди могут сдавать в аренду свои личные вещи другим пользователям за определенную плату. Это могут быть самые разнообразные предметы: от электроники и бытовой техники до спортивного инвентаря и инструментов для ремонта. Основное преимущество такой аренды заключается в том, что она позволяет пользователям получить доступ к вещам, которые им нужны на кратковременный период, без необходимости их покупки.

Современная платформа для пиринговой аренды вещей в первую очередь должна быть удобной и безопасной для всех участников. Она должна предлагать пользователям интуитивно понятный интерфейс, возможность быстрой и простой регистрации, а также широкие возможности по поиску и бронированию необходимых вещей. Важным аспектом также является система рейтингов и отзывов, которая помогает пользователям оценивать надежность арендодателей и арендаторов, что способствует созданию доверительной среды.

Платформа для пиринговой аренды вещей позволит людям экономить деньги, арендуя вещи вместо их покупки. Это особенно актуально для предметов, которые используются редко или на короткий период времени. Такая платформа поспособствует более рациональному использованию ресурсов, уменьшая количество производимых и покупаемых вещей, что, в свою очередь, снижает нагрузку на окружающую среду. Также она позволит людям зарабатывать деньги, сдавая в аренду свои вещи, которые в противном случае могли бы просто лежать без дела.

Кроме того, платформа для пиринговой аренды вещей может служить отличным инструментом для построения локальных сообществ. Она способствует взаимодействию и взаимодействию между людьми, которые живут в одном районе или городе, что может положительно сказаться на социальной сплоченности и взаимопомощи.

В целом, разработка платформы для пиринговой аренды вещей представляет собой важный шаг в направлении более устойчивого и экономически выгодного потребления. Она предоставляет пользователям возможность эффективно управлять своими ресурсами, экономить деньги и поддерживать окружающую среду. Такая платформа может стать полезным

инструментом как для индивидуальных пользователей, так и для малого бизнеса, предлагая им новые возможности для роста и развития.

Цель и задачи работы: целью данной дипломной работы является разработка платформы для пиринговой аренды вещей, которая обеспечит удобное и безопасное взаимодействие между пользователями. В рамках выполнения работы будут решены следующие задачи:

– анализ предметной области;

– выбор и описание технологий и инструментов;

– проектирование и разработка платформы;

– тестирование и оценка эффективности платформы.

Таким образом, представленная работа направлена на создание инновационного инструмента, который поможет пользователям эффективно управлять своими ресурсами и взаимодействовать с другими людьми с максимальной выгодой и удобством.

# 1 Анализ предметной области

## 1.1 Актуальность и проблематика платформы для пиринговой аренды вещей

Модели совместного потребления и аренды набирают все большую популярность во всем мире, особенно среди молодого поколения. Это связано с рядом объективных предпосылок:

– экономические факторы. Многие люди все чаще отдают предпочтение аренде вместо покупки товаров. Это позволяет им экономить средства, избегать ненужных трат и более рационально распоряжаться своим бюджетом. Кроме того, пользование вещами на условиях аренды или совместного доступа дает возможность опробовать новые продукты и услуги без необходимости их полной оплаты;

– экологические факторы. Модели совместного потребления способствуют снижению объемов производства новых товаров, что положительно сказывается на экологической обстановке. Вместо того, чтобы покупать новые вещи, люди могут арендовать их у других пользователей, продлевая срок службы существующих предметов;

– социальные факторы. Платформы пиринговой аренды вещей создают новые возможности для коммуникации, обмена опытом и формирования сообществ людей, разделяющих ценности устойчивого образа жизни. Это способствует укреплению социальных связей и повышению качества жизни;

– технологические факторы. Развитие цифровых платформ, мобильных приложений и технологий блокчейна значительно упрощает организацию пиринговых моделей аренды, обеспечивая удобство, безопасность и прозрачность операций для всех участников.

Несмотря на очевидную актуальность и перспективность концепции, существует ряд сложностей, которые необходимо учитывать при создании и развитии платформ для пиринговой аренды вещей:

– вопросы доверия и безопасности. Для успешной реализации пиринговой модели необходимо сформировать атмосферу доверия между арендодателями и арендаторами. Это подразумевает наличие надежных механизмов идентификации пользователей, верификации качества товаров, обеспечения сохранности вещей во время аренды и т.д.;

– правовые аспекты. Существует необходимость в разработке четкой нормативно-правовой базы, регулирующей отношения между участниками пиринговой аренды. Это касается вопросов ответственности сторон, налогообложения, страхования, защиты прав потребителей и т.п.;

– операционные сложности. Организация эффективных логистических процессов доставки, возврата и обслуживания арендуемых вещей является непростой задачей. Важно обеспечить высокую скорость и качество этих операций для поддержания лояльности пользователей;

– маркетинг и масштабирование. Для достижения критической массы пользователей и обеспечения ликвидности платформы требуются значительные маркетинговые усилия. Кроме того, необходимо разработать эффективные стратегии привлечения новых арендодателей и арендаторов, а также масштабирования бизнеса;

– конкуренция. На рынке уже присутствуют игроки, предлагающие различные модели пиринговой аренды. Для успешного развития проекта необходимо тщательно анализировать предложения конкурентов и предлагать уникальные конкурентные преимущества.

Таким образом, платформа для пиринговой аренды вещей обладает высокой актуальностью, вызванной растущим спросом на модели совместного потребления и аренды. Вместе с тем, реализация такого проекта сопряжена с рядом серьезных проблем, требующих комплексного и тщательного подхода к их решению. Только при условии учета всех ключевых аспектов можно рассчитывать на успешное создание и масштабирование платформы для пиринговой аренды вещей.

### 1.1.1 Обзор существующих платформ для пиринговой аренды вещей

В последние годы популярность пиринговой аренды вещей значительно выросла. Такие платформы дают людям возможность зарабатывать, сдавая в аренду личные вещи, а тем, кто ищет эти вещи, предоставляют доступ к разнообразному ассортименту по более низким ценам, чем в традиционных магазинах. Рассмотрим наиболее известные и успешные платформы для пиринговой аренды вещей.

Peerby — это нидерландская компания, основанная в 2012 году, которая стала одним из первопроходцев в сфере пиринговой аренды вещей. Платформа позволяет пользователям арендовать различные предметы у местных жителей, таких как инструменты, спортивное оборудование, бытовую технику и многое другое. Одной из сильных сторон Peerby является простота использования и эффективная система поиска необходимых вещей. Компания активно развивается, и на данный момент ее сервис доступен в Нидерландах, Бельгии, Германии и Великобритании.[1]

Одним из ключевых факторов успеха Peerby является ее фокус на создании доверительных отношений между арендодателями и арендаторами. Платформа предоставляет подробные профили пользователей, систему отзывов и рейтингов, а также гарантии безопасности сделок. Это позволяет людям чувствовать себя комфортно при использовании сервиса и арендовать вещи у незнакомцев. Кроме того, Peerby активно работает над расширением ассортимента доступных предметов, постоянно привлекая новых арендодателей. Компания также уделяет большое внимание экологической устойчивости, поощряя модель повторного использования вместо покупки новых вещей. Интерфейс приложения Peerby представлен на рисунке 1.

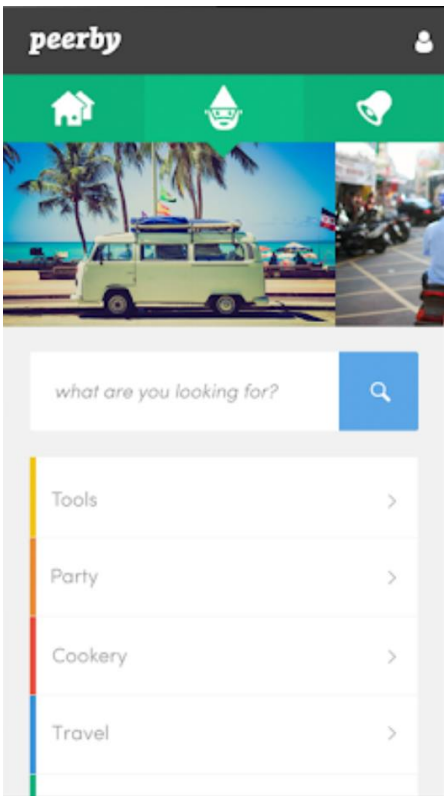



**Рисунок 1 – Интерфейс приложения Peerby**

Fat Llama - британская компания, основанная в 2016 году. Она предлагает широкий ассортимент товаров, включая фото - и видеооборудование, музыкальные инструменты, спортивный инвентарь и даже дроны. Платформа отличается высоким уровнем безопасности, предлагая страхование арендуемых вещей и тщательную проверку пользователей. Fat Llama работает только в Великобритании.[2] Интерфейс приложения Fat Llama представлен на рисунке 2.

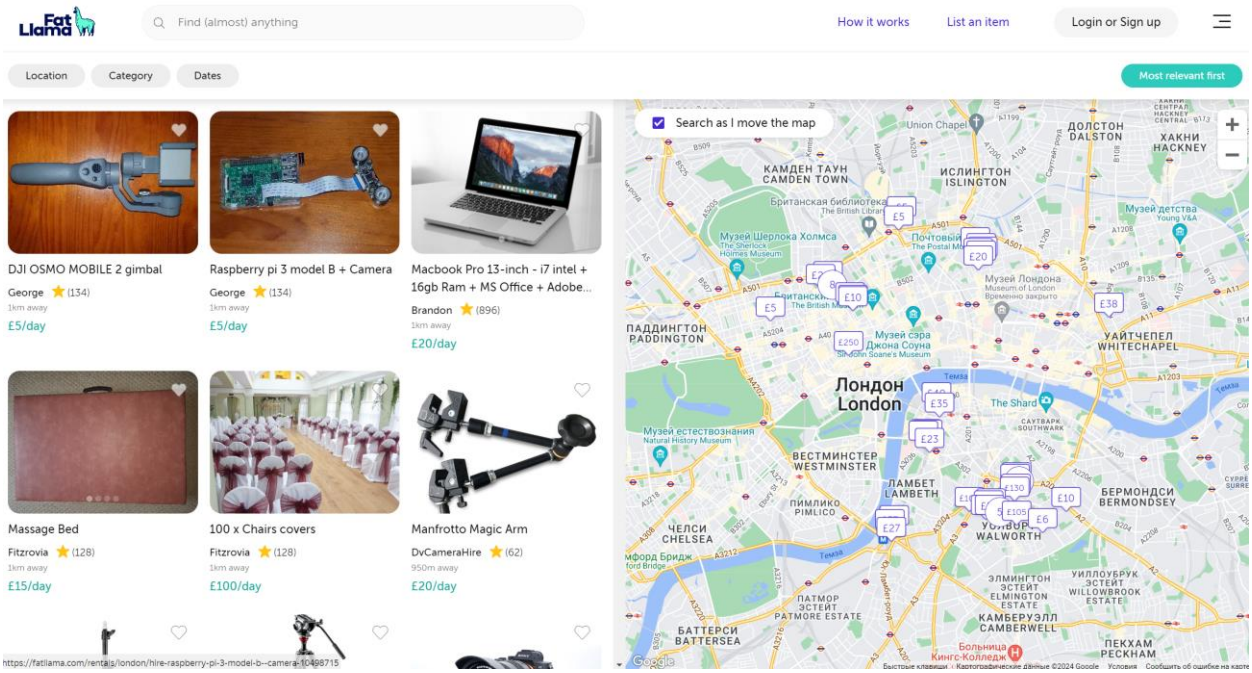



**Рисунок 2 – Интерфейс приложения Fat Llama**

Spinlister - американская платформа, специализирующаяся на аренде велосипедов, сноубордов и другого спортивного оборудования. Она была запущена в 2012 году и предлагает простой и удобный способ найти и арендовать необходимое снаряжение у местных жителей. Spinlister работает по всей территории Соединенных Штатов, а также в некоторых странах Европы.[3] Интерфейс приложения Spinlister представлен на рисунке 3.


**Рисунок 3 – Интерфейс приложения Spinlister**

Эти платформы представляют лишь небольшую часть быстрорастущего рынка пиринговой аренды вещей. Каждая из них имеет свои уникальные особенности и предлагает различные категории товаров. Однако все они объединены общей целью - предоставить людям доступ к разнообразным вещам по доступным ценам, одновременно создавая новые возможности для заработка.

### 1.1.2 Анализ функциональных возможностей популярных решений

Рассмотрев основные платформы для пиринговой аренды вещей, перейдем к более детальному анализу их ключевых функциональных возможностей.

1) Регистрация и учетные записи:

– все платформы предлагают простую регистрацию для арендодателей и арендаторов, как правило, с использованием электронной почты и социальных сетей;

– некоторые сервисы, такие как Fat Llama и Rentything, проводят более тщательную проверку пользователей для повышения безопасности;

– учетные записи позволяют пользователям публиковать объявления, управлять арендами, оставлять отзывы и рейтинги.

2) Поиск и фильтрация:

– платформы предлагают различные инструменты для поиска необходимых вещей, включая категории, фильтры по местоположению, типу товара, цене и наличию;

– некоторые сервисы, например Peerby и Zilok, имеют расширенные функции поиска с учетом доступности, условий аренды и рейтинга арендодателей.

3) Публикация объявлений:

– арендодателям предоставляется возможность публиковать объявления о сдаче вещей в аренду с подробными описаниями, фотографиями, ценами и условиями;

– платформы, такие как Rentything и Fat Llama, помогают арендодателям в создании привлекательных объявлений.

4) Управление арендой:

– пользователи могут бронировать, оплачивать и возвращать арендованные вещи через интуитивно понятные интерфейсы платформ;

– некоторые сервисы, например Spinlister, предлагают возможность удаленной разблокировки/блокировки арендованных предметов;

– платформы обеспечивают безопасные методы оплаты и страхование арендованных вещей.

5) Отзывы и рейтинги:

– пользователи могут оставлять отзывы и рейтинги о своем опыте аренды, что помогает повысить доверие к сервису;

– рейтинги арендодателей и арендаторов видны на их профилях, что способствует построению надежных отношений между участниками.

Проанализировав функциональные возможности популярных платформ для пиринговой аренды вещей, можно выделить следующие ключевые особенности и преимущества:

– разнообразие товаров: платформы предлагают широкий ассортимент вещей для аренды, от бытовой техники и инструментов до спортивного снаряжения и даже автомобилей;

– удобство и доступность: интуитивно понятные интерфейсы и мобильные приложения обеспечивают легкий поиск, бронирование и управление арендам;

– безопасность и доверие: тщательная проверка пользователей, система отзывов и рейтингов, а также страхование арендованных вещей повышают безопасность сделок;

– возможность заработка: арендодатели могут монетизировать свои неиспользуемые вещи, получая дополнительный доход;

– экономичность: арендаторы могут получить доступ к разнообразным товарам по более низким ценам, чем при покупке;

– экологичность: пиринговая аренда способствует более рациональному использованию и сокращению отходов.

Эти ключевые особенности и преимущества делают платформы для пиринговой аренды вещей привлекательными как для арендодателей, так и для арендаторов, способствуя росту популярности и развитию этого сегмента рынка.

### 1.1.3 Выявление недостатков и проблем существующих платформ

Несмотря на растущую популярность и успех существующих платформ для пиринговой аренды вещей, они все еще сталкиваются с рядом значительных недостатков и проблем, которые необходимо решить для дальнейшего развития этой сферы.

Одной из ключевых проблем является вопрос безопасности. Многие пользователи все еще с опаской относятся к идее передачи своих личных вещей незнакомым людям. Существующие платформы предпринимают шаги для повышения доверия, такие как проверка пользователей и предоставление страхования, однако этих мер часто бывает недостаточно, особенно для аренды более ценных или уникальных предметов. Необходимы более надежные механизмы верификации личности и обеспечения сохранности арендованных вещей.

Другая проблема связана с ограниченностью ассортимента. Несмотря на то, что платформы предлагают широкий выбор товаров, он все еще зачастую не отвечает всем потребностям пользователей. Многие предметы, особенно профессиональное оборудование или редкие коллекционные вещи, попросту отсутствуют в предложениях. Расширение каталога и привлечение большего числа арендодателей является важной задачей для развития этого рынка.

Еще одна проблема — это неравномерность географического распределения платформ. Большинство успешных сервисов ограничены лишь несколькими странами или регионами, в то время как во многих других местах они попросту недоступны. Для полноценного развития пиринговой аренды необходимо обеспечить глобальное присутствие платформ и их равномерное распространение по всему миру.

Кроме того, существует проблема с качеством и состоянием арендуемых предметов. Не всегда арендодатели надлежащим образом следят за своими вещами, что может приводить к повреждениям или неисправностям.

Улучшение системы проверки и контроля качества арендуемых товаров является важным шагом для повышения удовлетворенности пользователей.

Наконец, одной из ключевых проблем является недостаточная осведомленность населения о возможностях пиринговой аренды. Многие люди попросту не знают о существовании таких платформ или не понимают их преимуществ. Повышение информированности и популяризация концепции пиринговой аренды вещей могли бы значительно расширить клиентскую базу и ускорить развитие этого рынка.

Решение этих проблем является необходимым условием для дальнейшего роста и совершенствования платформ для пиринговой аренды вещей. Разработка более надежных систем безопасности, расширение ассортимента, глобальная экспансия, улучшение контроля качества и более активное продвижение концепции - все эти направления требуют пристального внимания и инноваций. Только комплексный подход позволит раскрыть полный потенциал пиринговой аренды и сделать ее удобным, безопасным и привлекательным вариантом для широкого круга пользователей.

## 1.2 Требования к платформе

Ключевые функциональные требования к платформе для пиринговой аренды вещей включают в себя:

1) Регистрация и авторизация пользователей:

– возможность регистрации пользователей через различные методы (email, google);

– простой и интуитивно понятный процесс регистрации;

– надежная система авторизации с высоким уровнем безопасности.

2) Публикация объявлений о сдаче в аренду:

– возможность для пользователей создавать объявления о сдаче в аренду своих вещей;

– способность добавлять подробную информацию об арендуемом предмете (описание, фотографии, местоположение, условия аренды, стоимость);

– функционал редактирования и удаления созданных объявлений.

3) Поиск и просмотр объявлений:

– удобная система поиска и фильтрации объявлений по различным критериям (категория, местоположение, диапазон цен и т.д.);

– детальное отображение информации об арендуемом предмете в объявлении;

– возможность просмотра рейтингов и отзывов о продавцах.

4) Бронирование и аренда вещей:

– функционал бронирования арендуемых предметов пользователями;

– интеграция с платежными системами для совершения арендных платежей;

– система уведомлений о статусе бронирования и сроках аренды.

5) Управление арендами:

– возможность для арендодателей управлять своими активными арендами;

– отслеживание статуса аренды, сроков возврата и оплаты;

– функционал продления или завершения аренды.

6) Отзывы и рейтинги:

– система отзывов и рейтингов для арендодателей и арендаторов;

– использование рейтингов в качестве критерия при поиске и бронировании.

7) Система сообщений:

– встроенный функционал обмена сообщениями между арендодателями и арендаторами;

– возможность задавать вопросы, обсуждать детали аренды и т.д.

8) Управление профилем:

– возможность редактирования личной информации пользователей;

– добавление фотографий профиля, описания и других данных;

– управление списком арендованных/сданных в аренду вещей.

## 1.3 Выбор и описание технологий и инструментов

При разработке мобильного приложения платформы для пиринговой аренды вещей было принято решение использовать фреймворк React Native. Этот выбор обусловлен рядом преимуществ, которые React Native предоставляет по сравнению с другими подходами к разработке мобильных приложений.

– производительность. React Native использует нативные компоненты платформ iOS и Android, что обеспечивает высокую производительность и плавность анимаций, сравнимую с нативными приложениями. Это важно для обеспечения комфортного пользовательского опыта в мобильном приложении, особенно при работе с интерактивными элементами и визуализацией данных;

– кроссплатформенность. React Native позволяет разрабатывать приложения для обеих мобильных платформ, используя один и тот же JavaScript-код с небольшими платформенно-специфичными вставками. Это значительно сокращает временные и финансовые затраты на разработку и поддержку двух отдельных версий приложения;

– экосистема и сообщество. React Native активно развивается компанией Facebook и имеет обширное сообщество разработчиков. Это означает наличие богатой экосистемы готовых библиотек и компонентов, а также обилие

обучающих материалов и примеров кода. Это ускоряет разработку и упрощает решение типовых задач;

– разработка на знакомом стеке. Многие члены команды разработки уже имели опыт работы с React.js, JavaScript и TypeScript. Использование React Native позволило им применить эти знания и навыки в контексте мобильной разработки, что сократило время на изучение новых технологий и инструментов;

– горячая перезагрузка и отладка. React Native предоставляет удобные средства для быстрой итеративной разработки, включая горячую перезагрузку и встроенные инструменты отладки. Это повышает продуктивность разработчиков и ускоряет процесс создания и тестирования нового функционала;

– доступ к нативным API. Несмотря на использование единого кодового базиса, React Native позволяет получать доступ к нативным API платформ, таким как камера, геолокация, push-уведомления и другие. Это дает возможность реализовать полноценные мобильные приложения, использующие все возможности устройств;

– обновление без перезагрузки. React Native поддерживает технологию Live Updates, которая позволяет развертывать обновления приложения без необходимости повторной загрузки и установки. Это значительно упрощает процесс обновления и повышает доступность приложения для пользователей;

– зрелость и поддержка. React Native активно развивается с 2015 года и используется такими крупными компаниями, как Facebook, Instagram, Airbnb, Skype и многими другими. Это свидетельствует о зрелости фреймворка и гарантирует долгосрочную поддержку и развитие.

Таким образом, выбор React Native в качестве основы для разработки мобильного приложения платформы для пиринговой аренды вещей был обоснован множеством преимуществ, которые этот фреймворк предоставляет: кроссплатформенность, высокая производительность, богатая экосистема, знакомый стек технологий, удобные инструменты разработки и отладки, а также зрелость и поддержка проекта. Эти факторы позволили команде разработчиков создать высококачественное мобильное приложение, отвечающее всем требованиям бизнеса и пользователей.[4-6]

### 1.3.1 Обоснование выбора Firebase для бэкенда и хранения данных

При разработке платформы для пиринговой аренды вещей был выбран сервис Google Firebase в качестве основы для бэкенда и системы хранения данных. Это решение было принято по ряду весомых причин:

– быстрое развертывание и масштабирование. Firebase предоставляет комплексную платформу "Backend-as-a-Service", которая позволяет быстро развернуть надежную серверную инфраструктуру без необходимости построения собственного бэкенда "с нуля". Платформа автоматически

масштабируется в соответствии с потребностями растущего приложения, что позволяет сосредоточиться на разработке функциональности, не беспокоясь об управлении серверами;

– интегрированные сервисы. Firebase включает в себя широкий спектр интегрированных сервисов, покрывающих практически все потребности типичного мобильного приложения: аутентификация пользователей, облачное хранилище данных, хостинг статического контента, облачные функции, аналитика, push-уведомления и многое другое. Использование этого комплексного решения позволило значительно ускорить разработку, избежать интеграции множества разрозненных сторонних сервисов и унифицировать управление данными;

– кроссплатформенность и единый стек. Благодаря поддержке Firebase SDK для платформ iOS, Android и веб-приложений, команда разработчиков смогла использовать единый стек технологий и API для реализации клиентских приложений и бэкенда. Это упростило разработку, тестирование и поддержку как мобильного, так и веб-приложения платформы для пиринговой аренды вещей;

– безопасность и управление доступом. Firebase предлагает мощные средства для обеспечения безопасности данных и управления доступом к ним. Встроенные механизмы аутентификации, авторизации и правила безопасности позволили реализовать надежную защиту от несанкционированного доступа к критически важной информации платформы;

– удобство разработки и DevOps. Экосистема Firebase включает в себя удобные инструменты для разработки, отладки и развертывания приложений, такие как Firebase Console, Firebase CLI, Firebase Hosting и Firebase Functions. Это позволило команде сосредоточиться на бизнес-логике, не тратя время на рутинные DevOps-операции;

– аналитика и монетизация. Firebase предоставляет встроенную аналитическую платформу Firebase Analytics, которая помогает отслеживать поведение пользователей, выявлять ключевые метрики и принимать обоснованные решения. Кроме того, Firebase Crashlytics обеспечивает мониторинг стабильности приложения и помогает быстро исправлять возникающие ошибки. Платформа также предлагает инструменты монетизации, такие как Firebase Remote Config и Firebase A/B Testing, которые могут быть использованы для оптимизации бизнес-показателей;

– надежность и поддержка. Google Firebase является зрелым и надежным сервисом, поддерживаемым одним из лидеров IT-индустрии - компанией Google. Это гарантирует долгосрочную доступность платформы, регулярное обновление функциональности и высокий уровень технической поддержки.

Интерфейс сервиса Google Firebase представлен на рисунке 4.

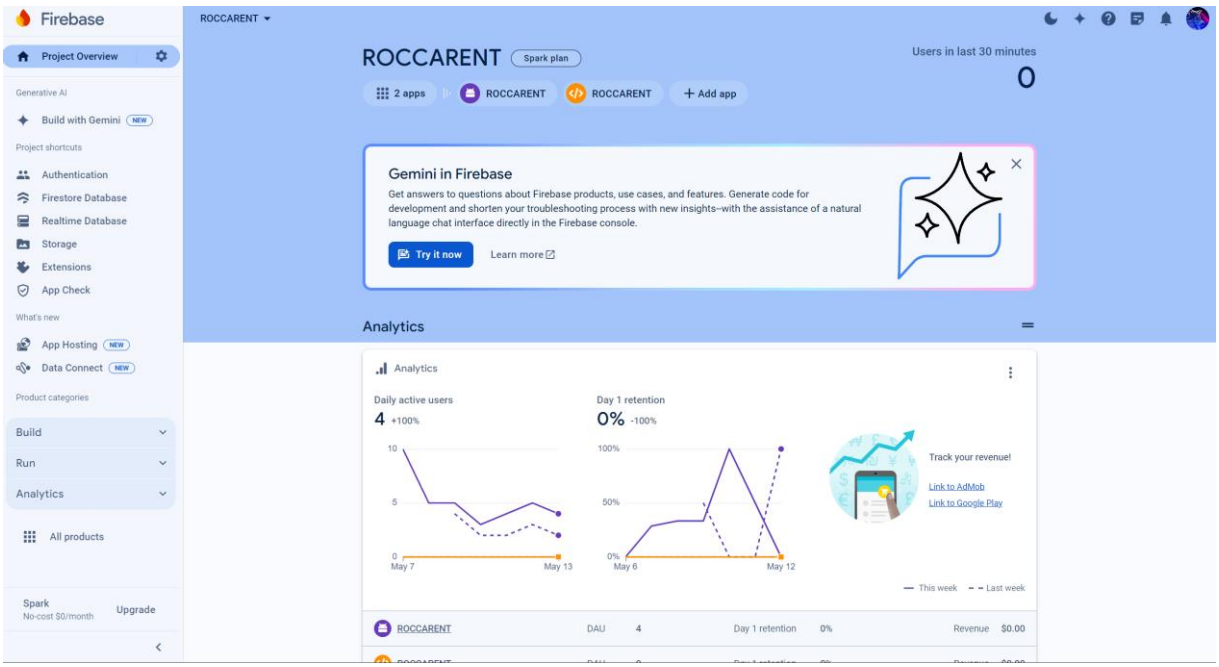



**Рисунок 4 – Интерфейс сервиса Google Firebase**

Таким образом, выбор Firebase в качестве основы для бэкенда и хранения данных платформы для пиринговой аренды вещей был обусловлен множеством преимуществ. Быстрое развертывание, широкий спектр интегрированных сервисов, кроссплатформенность, безопасность, удобство разработки и DevOps, а также надежность и поддержка со стороны Google позволили команде значительно ускорить разработку, сосредоточившись на реализации бизнес-логики, а не инфраструктурных задачах.[7-9]

### 1.3.2 IntelliJ IDEA

IntelliJ IDEA — это интегрированная среда разработки (IDE), которая широко используется для создания программного обеспечения на JavaScript и других языках программирования. Она предоставляет множество полезных функций, облегчающих разработку, такие как:

– умный код-ассистент, который помогает писать код быстрее и с меньшим количеством ошибок;

– встроенная поддержка популярных фреймворков и библиотек, включая Maven, Gradle и многие другие;

– удобная система управления проектами и сборки;

– интегрированная система контроля версий, поддерживающая Git, Subversion и другие;

– расширенные средства отладки и профилирования;

– широкие возможности настройки и расширения функционала с помощью плагинов.[10]

Интерфейс IDE IntelliJ IDEA представлен на рисунке 5.

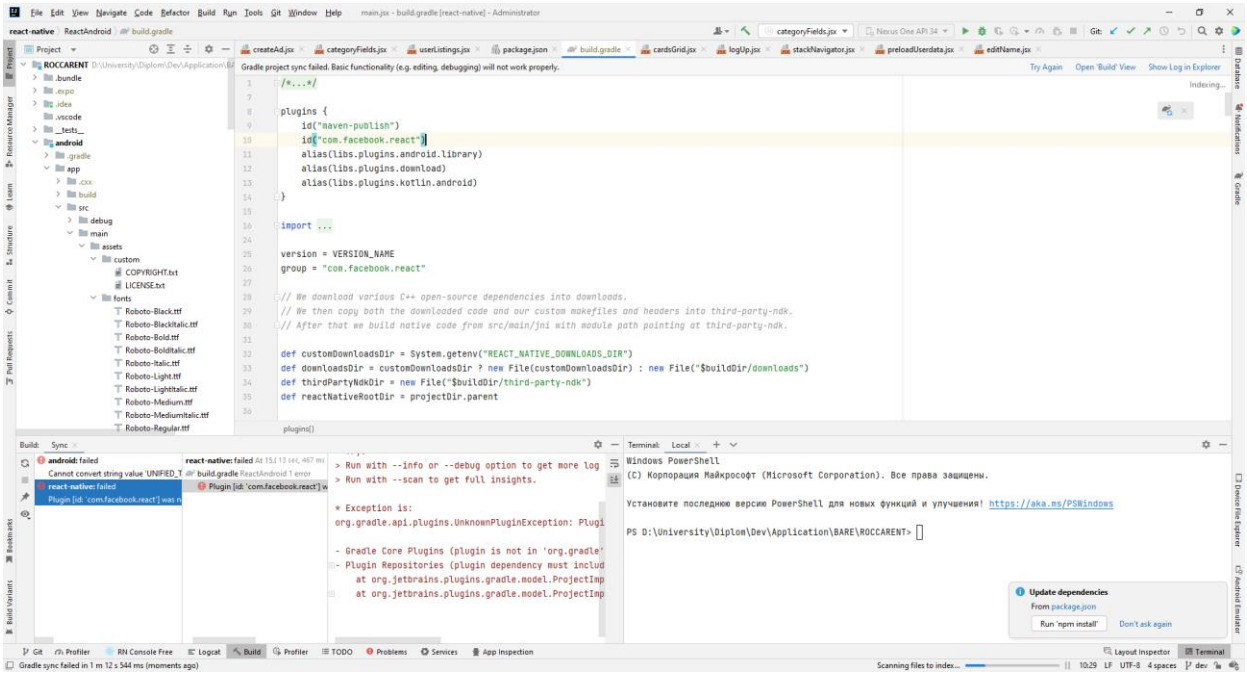


**Рисунок 5 – Интерфейс IDE IntelliJ IDEA**

В рамках этого проекта в качестве среды разработки выбрана IDE IntelliJ IDEA благодаря своим широким возможностям отладки и профилирования.

### 1.3.3 Google Maps API

Google Maps API — это набор инструментов, предоставляемых компанией Google, которые позволяют интегрировать картографические возможности Google Maps в веб-сайты и мобильные приложения. Основные возможности, задействованные в этом проекте:

– отображение интерактивных карт с возможностью панорамирования и масштабирования;

– геокодирования - конвертация адресов в географические координаты и обратно;

– маршрутизация - построение оптимальных маршрутов между точками;

– поиск информации о местах (адреса, часы работы, отзывы и т.д.).

В нашем приложении Google Maps API используется для отображения карт, поиска адресов сдаваемых в аренду вещей.[11]

### 1.3.4 Geocoding API

Geocoding API — это сервис, предоставляющий бесплатный доступ к геокодированию адресов в географические координаты и обратному процессу - reverse geocoding. Основные возможности:

– конвертация адресов в широту/долготу и обратно;

– поддержка различных форматов ввода/вывода данных (JSON, XML, GeoJSON);

– гибкие возможности фильтрации и уточнения запросов;

– бесплатный тарифный план с лимитом в 1 млн запросов в месяц.

Этот сервис используется в проекте для геокодирования адресов арендодателей, чтобы отображать их на интерактивной карте.[12]

### 1.3.5 react-native-ml-kit/text-recognition

@react-native-ml-kit/text-recognition — это библиотека для платформы React Native, которая предоставляет доступ к возможностям распознавания текста на изображениях с использованием Google ML Kit. Основные возможности:

– извлечение текста из изображений, сделанных камерой устройства;

– поддержка многоязычного распознавания текста;

– высокая точность и скорость распознавания;

– простой и интуитивно понятный API.

В мобильном приложении этот компонент используется для того, чтобы обезопасить пользователей путем верификации паспортных данных, просто сфотографировав их с помощью камеры устройства.[13]

Таким образом, представленные выше дополнительные библиотеки и инструменты помогут в разработке платформы для пиринговой аренды вещей, предоставив широкий спектр функциональных возможностей, от удобной среды разработки до продвинутых алгоритмов обработки данных.

### 1.3.6 Архитектура приложения и интеграция с Firebase

При разработке платформы для пиринговой аренды вещей используется многослойная архитектура приложения, состоящая из следующих компонентов:

1) Презентационный слой. Презентационный слой отвечает за взаимодействие с пользователем и включает в себя:

– мобильное приложение, разработанное с использованием React Native. Оно предоставляет интуитивно понятный пользовательский интерфейс для поиска, аренды и сдачи в аренду вещей.

2) Бизнес-логика. Бизнес-логика реализована на серверной части платформы, написанной на языке программирования JavaScript. Она включает в себя следующие основные функции:

– функции для работы с HTTP запросами;

– функции для работы с Firebase;

– функции для работы со сторонними API.

3) Интеграция с Firebase. Для реализации взаимодействия между серверной и клиентской частями платформы используется интеграция с Firebase. Это позволяет достичь следующих результатов:

– унификация системы аутентификации пользователей. Пользователи могут входить в систему, используя учетные записи Google, Facebook или email/пароль, и эта информация синхронизируется между мобильным/веб-приложениями и серверной частью;

– единое хранилище данных. Firestore используется как основное хранилище данных как для мобильного, так и для веб-приложения, что обеспечивает согласованность информации;

– синхронизация данных в реальном времени. Изменения в Firestore мгновенно отражаются во всех клиентских приложениях, благодаря встроенным механизмам подписки и обновления данных.

Схема интеграции платформы с Firebase представлена на рисунке 6.

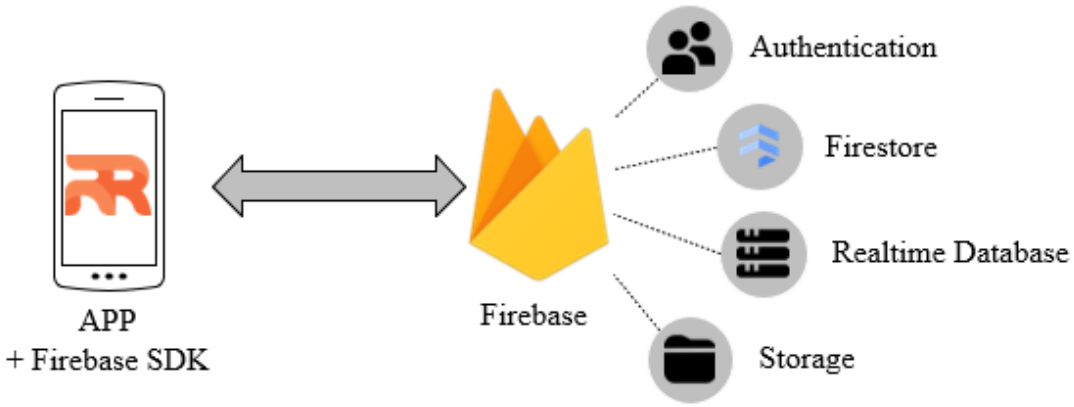



**Рисунок 6 – Схема интеграции платформы с Firebase**

4) Хранилище данных. В качестве хранилища данных используется облачная база данных Firebase Firestore от Google. Firebase Firestore — это NoSQL документно-ориентированная база данных, которая предоставляет следующие преимущества:

– высокая масштабируемость и доступность;

– встроенная синхронизация данных в реальном времени;

– простой и удобный API для взаимодействия с данными;

– интеграция с другими сервисами экосистемы Firebase.

В Firestore хранятся следующие основные сущности платформы:

– объявления о сдаче вещей в аренду;

– профили пользователей;

– истории аренды;

– отзывы и рейтинги.

Пример хранения данных в Firestore представлен на рисунке 7.

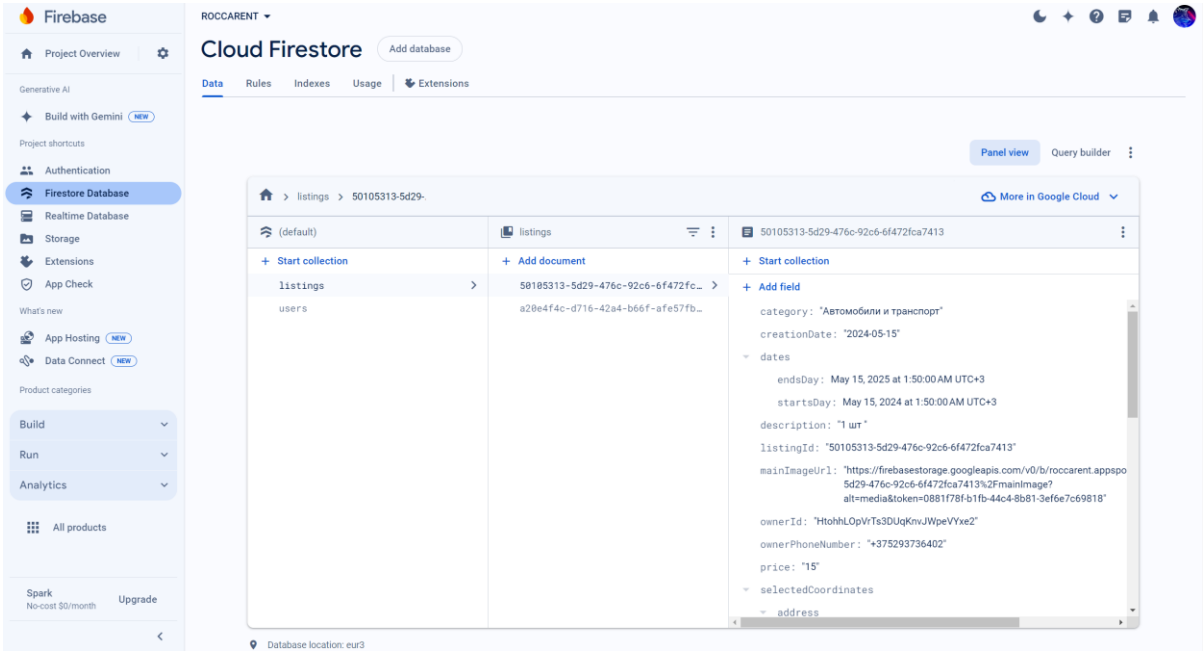



**Рисунок 7 – Пример хранения данных в Firestore**

Таким образом, архитектура платформы, основанная на разделении ответственностей между презентационным, бизнес-логическим слоями и хранилищем данных, в сочетании с интеграцией с Firebase, обеспечивает высокую гибкость, масштабируемость и надежность всей системы.

# 2 Проектирование и разработка платформы

## 2.1 Структура проекта

Проект платформы для пиринговой аренды вещей на React Native имеет структуру, представленную на рисунке 8.

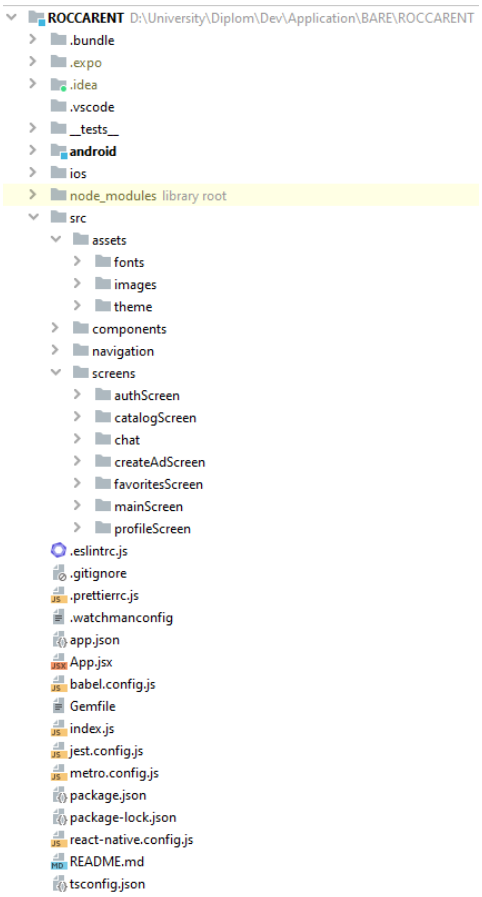



**Рисунок 8 – Структура проекта платформы для пиринговой аренды вещей**

Описание структуры проекта:

– android/ и ios/. Содержат нативные файлы для платформ Android и iOS соответственно;

– node_modules/. Хранит все установленные зависимости проекта;

– src/. Корневая папка для исходного кода приложения;

– assets/. Содержит изображения, шрифты, темы и другие статические ресурсы, используемые в приложении;

– components/. Хранит многоразовые компоненты, такие как заголовок, футер, карточки категорий и предметов;

– navigation/. Хранит компоненты, отвечающие за навигацию в приложении;

– screens/. Содержит основные экраны приложения, например, главный экран, экран категорий, экран деталей предмета и профиля;

– App.jsx. Главный файл приложения;

– babel.config.js и metro.config.js. Конфигурационные файлы для Babel и Metro bundler.

Эта структура проекта обеспечивает удобную организацию и модульность кода, что способствует масштабированию, поддержке и разработке мобильного приложения на React Native.

## 2.2 Основные компоненты и их функциональность

Схема работы платформы приведена в Приложении А. Рассмотрим более подробно основные компоненты приложения и их функциональность.

### 2.2.1 Корневой компонент

Корневой компонент платформы является точкой входа в приложение. Он отвечает за организацию навигации и управление потоком между ключевыми экранами приложения. Также этот компонент автоматически при первом запуске приложения регистрирует анонимную учетную запись пользователя, после чего пользователь может просматривать объявлений и добавлять их в избранное. Таким образом после успешной регистрации в приложении с использованием email или google данные анонимного пользователя сохраняются. Листинг кода корневого компонента представлен на рисунке 9.

```
import ...

export const AppContext = createContext();

const App = React.memo( Component: () => {

  const { theme, toggleMode, loadMode } = useCustomTheme();
  const [userdata, setUserdata] = useState( initialState: null);
  const [initializing, setInitializing] = useState( initialState: true);
  const [loadingScreenText, setLoadingScreenText] = useState( initialState: null);
  const [isInternetConnected, setIsInternetConnected] = useState( initialState: true);

  StatusBar.setTranslucent( translucent: true);
  StatusBar.setBackgroundColor(theme.colors.accent);
  StatusBar.setBarStyle( style: "light-content",  animated: true);
  changeNavigationBarColor(theme.colors.background,  light: theme.mode !== "dark",  animated: false);

  useEffect( effect: () => {...},  deps: []);
  useEffect( effect: () => {...},  deps: []);
  useEffect( effect: () => {...},  deps: [auth().currentUser]);
  useEffect( effect: () => {...},  deps: []);

  // Загрузка сохраненной темы при запуске приложения
  const loadTheme = async () => {...};
  async function preloadImages() {...}
  const loadUserdata = async () => {...};
  const waitForEmailVerification = async () => {...};

  GoogleSignin.configure({webClientId: "771592361046-c50gd0p0heu9i02kp2j8j3s27m45h8cl.apps.googleusercontent.com"...});
  const onGoogleButtonPress = async () => {...};

  if (!isInternetConnected) {
    return ( <SafeAreaProvider> <AppLoadingScreen theme={theme} text={"Нет подключения к интернету"} textColor={theme.colors.error} /> </SafeAreaProvider> );
  }
  if (initializing) { return <AppLoadingScreen theme={theme} text={loadingScreenText} /> }

  return (<GestureHandlerRootView style={{ flex: 1 }}...>);
});

export default App;
```

**Рисунок 9 – Листинг кода корневого компонента**

### 2.2.2 Компонент темы

Компонент темы отвечает за установку, загрузку и смену темы в приложении. Листинг кода компонента темы представлен на рисунке 10.

```
import ...

export const useCustomTheme = () => {
  const [mode, setMode] = useState( initialState: "light");

  const toggleMode = useCallback( callback: () => {...}, deps: []);

  const loadMode = useCallback( callback: (newMode) => {...}, deps: []);

  const theme = createTheme({mode: mode...});

  return { theme, toggleMode, loadMode };
};
```

**Рисунок 10 – Листинг кода компонента темы**

### 2.2.3 Компоненты навигации

Ключевым компонентом навигации в приложении является Navigator, который содержит в себе BottomTabNavigator, отвечающий за навигацию с помощью нижней панели навигации и содержащий в себе ссылки на стековые навигаторы из StackNavigator и экран создания объявлений CreateAd. Листинг кода компонента навигации Navigator представлен на рисунке 11.

```
import ...

// Массив с конфигурациями вкладок
const TabArr = [...];

const Tab = createBottomTabNavigator();

// Компонент кнопки вкладки
const TabButton = (props) => {...};

// Компонент нижней навигации по вкладкам
const BottomTabNavigator = React.memo( Component: ({ theme, toggleMode, isDarkMode, routeName, setInitializing, setLoadingScreenText }) => {...})

const ref = createNavigationContainerRef();
// Компонент навигации приложения
const AppNavigator = ({ setInitializing, theme, toggleMode, setLoadingScreenText }) => {...};

export default AppNavigator;

const styles = StyleSheet.create({...});
```

**Рисунок 11 – Листинг компонента навигации Navigator**

StackNavigator содержит в себе несколько навигаторов, обеспечивающих стековую структуру навигации для модулей приложения:

– MainStackNavigator управляет навигацией на главном экране приложения;

– CatalogStackNavigator отвечает за навигацию в каталоге объявлений;

– FavoritesStackNavigator управляет навигацией на экране избранного;

– ProfileStackNavigator отвечает за навигацию на экране профиля пользователя. Листинг кода компонента навигации StackNavigator представлен на рисунке 12.

```
import ...
const Stack = createNativeStackNavigator();

export const ProfileStackNavigator = ({
                                        user,
                                        theme,
                                        toggleMode,
                                        setInitializing,
                                        setLoadingScreenText,
                                    }) => {...};

export const MainStackNavigator = ({user,
                                    theme,
                                    toggleMode,
                                    setInitializing,
                                    setLoadingScreenText,}) => {...}

export const CatalogStackNavigator = ({user,
                                       theme,
                                       toggleMode,
                                       setInitializing,
                                       setLoadingScreenText,}) => {...}

export const FavoritesStackNavigator = ({user,
                                        theme,
                                        toggleMode,
                                        setInitializing,
                                        setLoadingScreenText,}) => {...}
```

**Рисунок 12 – Листинг кода компонента навигации StackNavigator**

### 2.2.4 Компоненты экранов входа и регистрации

Экран входа позволяет пользователям авторизоваться в приложении, использую email и пароль, или войти через Google. Листинг кода экрана входа представлен на рисунке 13.

```
import ...

const LogIn = ({ theme, onGoogleButtonPress, setInitializing, navigation }) => {
  const [isPasswordSecure, setIsPasswordSecure] = useState( initialState: true);
  const [authBtnDisabled, setAuthBtnDisabled] = useState( initialState: true);
  const passwordRef = useRef( initialValue: null);

  const [email, setEmail] = useState( initialState: "");
  const [password, setPassword] = useState( initialState: "");

  const backColor = theme.colors.background;
  const textColor = theme.colors.text;

  const [hasValidPasswordLength, setHasValidPasswordLength] = useState( initialState: false);
  const [hasValidPassword, setHasValidPassword] = useState( initialState: false);
  const [hasValidEmail, setHasValidEmail] = useState( initialState: false);

  useEffect( effect: () => {...}, deps: []);


  useEffect( effect: () => {...}, deps: [email, password]);

  const handleEmailClear = () => {...};

  const handleEmailChange = (value) => {...};

  const handlePasswordChange = (value) => {...};

  const handleLogInBtn = async () => {...};

  const styles = StyleSheet.create({...});

  return (<SafeAreaView style={{ flex: 1 }}...>);
};

export default LogIn;
```

**Рисунок 13 – Листинг кода экрана входа**

Экран регистрации дает возможность новым пользователям создать учетную запись, заполнив необходимые данные, такие как, имя, фамилия, email, пароль. После успешной регистрации пользователь автоматически переходит на главный экран приложения. Листинг кода экрана регистрации представлен на рисунке 14.

```
import ...

const LogUp = ({ theme, setInitializing, setLoadingScreenText, onGoogleButtonPress, navigation }) => {

    const [isPasswordSecure, setIsPasswordSecure] = useState( initialState: true);
    const [isConfirmPasswordSecure, setIsConfirmPasswordSecure] = useState( initialState: true);
    const [authBtnDisabled, setAuthBtnDisabled] = useState( initialState: true);

    const [name, setName] = useState( initialState: "");
    const [surname, setSurname] = useState( initialState: "");
    const [email, setEmail] = useState( initialState: "");
    const [password, setPassword] = useState( initialState: "");
    const [passwordConfirmation, setPasswordConfirmation] = useState( initialState: "");
    const [termsAccepted, setTermsAccepted] = useState( initialState: false);
    const surnameRef = useRef( initialValue: null);
    const emailRef = useRef( initialValue: null);
    const passwordRef = useRef( initialValue: null);
    const passwordConfirmationRef = useRef( initialValue: null);

    const backColor = theme.colors.background;
    const textColor = theme.colors.text;
    const accentColor = theme.colors.accent;

    const [hasValidPasswordLength, setHasValidPasswordLength] = useState( initialState: false);
    const [hasValidPassword, setHasValidPassword] = useState( initialState: false);
    const [hasValidEmail, setHasValidEmail] = useState( initialState: false);
    const [hasValidName, setHasValidName] = useState( initialState: false);
    const [hasValidSurname, setHasValidSurname] = useState( initialState: false);
    const [passwordsMatch, setPaswordsMatch] = useState( initialState: false);
    useEffect( effect: () => {...},  deps: [name, surname, email, password, passwordConfirmation, termsAccepted]);

    const handleNameClear = () => {...};

    const handleNameChange = (value) => {...};

    const handleSurnameClear = () => {...};

    const handleSurnameChange = (value) => {...};

    const handleEmailClear = () => {...};

    const handleEmailChange = (value) => {...};

    const handlePasswordChange = (value) => {...};

    const handlePasswordConfirmationChange = (value) => {...};

    const handleTermsToggle = (isChecked) => {...};

    const handleLogUpBtn = async () => {...};

    const styles = StyleSheet.create({...});

    return (
      <SafeAreaView style={{ flex: 1 }}...>
    );
  }
;

export default LogUp;
```

**Рисунок 14 – Листинг кода экрана регистрации**

Оба экрана обеспечивают безопасность личных данных пользователей за счет использования надежных методов аутентификации и шифрования.

### 2.2.5 Компонент главного экрана

Главный экран отображает список объявлений. Листинг кода главного экрана представлен на рисунке 15.

```
import ...

const Main = ({ theme }) => {

  const { loadUserdata } = useContext(AppContext);

  const [listings, setListings] = useState( initialState: []);

  useEffect( effect: () => {...}, deps: []);

  const handleReloadBtn = async () => {...};

  const loadListingList = async () => {...};

  const styles = StyleSheet.create({...});

  return (
    <SafeAreaProvider...>
  );
};

export default Main;
```

**Рисунок 15 – Листинг кода главного экрана**

Для незарегистрированных пользователей на главном экране присутствует кнопка-подсказка, предлагающая пройти процесс регистрации.

Для пользователей, не прошедших верификацию, отображается кнопка, предлагающая пройти верификацию.

### 2.2.6 Компонент экрана каталога

Экран каталога позволяет пользователям просматривать полный список доступных для аренды вещей, фильтры и поиск по названиям объявлений. Листинг кода экрана каталога представлен на рисунке 16.

```
import ...

const Catalog = ({theme}) => {

  const { loadUserdata } = useContext(AppContext);

  const [listings, setListings] = useState( initialState: []);

  useEffect( effect: () => {...}, deps: []);

  const handleReloadBtn = async () => {...};

  const loadListingList = async () => {...};

  const styles = StyleSheet.create({...});

  return (
    <SafeAreaView style={{flex: 1}}...>
  );
};

export default Catalog;
```

**Рисунок 16 – Листинг кода экрана каталога**

### 2.2.7 Компонент экрана подачи объявлений

Экран подачи объявлений позволяет пользователям размещать свои вещи для сдачи в аренду. Листинг кода экрана подачи объявлений представлен на рисунке 17.

```
import ...

const CreateAd = ({ theme, navigation }) => {

  const { userdata, loadUserdata } = useContext(AppContext);
  const [listingUploading, setListingUploading] = useState( initialState: false);
  const [isSubmitBtnEnabled, setIsSubmitBtnEnabled] = useState( initialState: false);
  const [isModalVisible, setModalVisible] = useState( initialState: false);
  const [selectedCategory, setSelectedCategory] = useState( initialState: null);
  const [selectedSubcategory, setSelectedSubcategory] = useState( initialState: null);
  const [fieldsData, setFieldsData] = useState( initialState: {});
  const [selectedImages, setSelectedImages] = useState( initialState: []);
  const [isAddingImageInProcess, setIsAddingImageInProcess] = useState( initialState: false);
  const [userCoordinates, setUserCoordinates] = useState({...});
  const [markerCoordinates, setMarkerCoordinates] = useState(userCoordinates);
  const [selectedCoordinates, setSelectedCoordinates] = useState( initialState: {});
  const [userLocationLoading, setUserLocationLoading] = useState( initialState: false);
  const mapRef = useRef( initialValue: null);
  const [isMapOpen, setIsMapOpen] = useState( initialState: false);
  const titleRef = useRef( initialValue: null);
  const descriptionRef = useRef( initialValue: null);
  const priceRef = useRef( initialValue: null);
  const categories = [...];

  const handleFieldsChange = (fieldName, value) => {...};
  const handleCleanBtn = () => {...};
  const handleAddImageBtn = async () => {...};
  const handleDeleteImageBtn = (image, uri) => {...};
  const renderItem = ({ item, drag, isActive }: RenderItemParams<Item>) => {...};
  const handleCategoryModal = (mode) => {...};
  const handleCategoryPress = (category) => {...};
  const handleSubcategoryPress = (subcategory) => {...};
  const handleStartsDayChoose = async (value) => {...};
  const handleEndsDayChoose = async (value) => {...};
  const handleStartsDayModalOpen = () => {...};
  const handleEndsDayModalOpen = () => {...};
  const handleLocation = async () => {...};
  const checkLocationPermission = () => {...};
  const getCurrentUserLocation = async () => {...};
  const handleOpenMap = async () => {...};
  const handleCloseMap = () => {...};
  const handleMapReady = () => {...};
  const handleMapPress = (event) => {...};
  const reverseGeocode = async (latitude, longitude) => {...};
  const handleSaveLocation = async () => {...};
  const handleCreateListing = async () => {...};

  useEffect( effect: () => {...}, deps: [
      selectedImages, selectedCategory, selectedSubcategory, fieldsData, selectedCoordinates
  ]);

  const styles = StyleSheet.create({...});

  if (userLocationLoading || listingUploading) {...}

  const AnonymousBtn = () => ...

  const NotVerified = () => ...

  return (
    <SafeAreaView style={{ flex: 1 }}...>
  );
};

export default CreateAd;
```

**Рисунок 17 – Листинг кода экрана подачи объявлений**

Пользователь может загрузить фотографии предмета, указать подробное описание, установить цену и условия аренды.

После заполнения всех необходимых данных объявление отправляется на модерацию и, при одобрении, публикуется в каталоге для просмотра другими пользователями.

### 2.2.8 Компонент экрана избранных объявлений

Экран избранных объявлений дает возможность пользователям сохранять предметы в личный список для дальнейшего просмотра. Для верифицированных пользователей отображается раздел «Активные сделки». Листинг кода экрана избранных объявлений представлен на рисунке 18.

```
import ...

const Favorites = ({ theme, navigation }) => {

  const { userdata } = useContext(AppContext);

  const [listingsLoading, setListingsLoading] = useState( initialState: true);
  const [favoriteListings, setFavoriteListings] = useState( initialState: []);

  useEffect( effect: () => {...}, deps: [userdata && userdata.favoriteListings]);

  const fetchFavoriteListings = async () => {...};

  const styles = StyleSheet.create({...});

  return (
    <SafeAreaView style={{ flex: 1 }}...>
  );
};

export default Favorites;
```

**Рисунок 18 – Листинг кода экрана избранных объявлений**

Пользователи могут добавлять предметы в избранное, удалять их оттуда, а также просматривать полный список сохраненных объявлений.

### 2.2.9 Компонент экрана профиля пользователя

Экран профиля пользователя позволяет пользователям управлять своей учетной записью и персональной информацией.

Здесь пользователи могут редактировать свои личные данные, такие как имя, аватар, контактная информация.

Также на экране профиля отображается история активности пользователя, включая размещенные объявления, арендованные и сданные в аренду предметы, и кнопка обращения в поддержку с переходом на экран с формой для заполнения и отправки в службу поддержки. Листинг кода экрана профиля пользователя представлен на рисунке 19.

```
import ...

const Profile = ({ theme, toggleMode, navigation, setInitializing }) => {

  const { userdata, loadUserdata } = useContext(AppContext);

  const [isAvatarLoading, setIsAvatarLoading] = useState( initialState: true);
  const [visible, setVisible] = useState( initialState: false);
  const [isSun, setIsSun] = useState( initialState: theme.mode === "light");

  useLayoutEffect( effect: () => {...}, deps: []);

  const handleToggleModePress = () => {...};

  const toggleOverlay = () => {...};

  const handleLogInBtnPress = () => {...};

  const handleLogUpBtnPress = () => {...};

  const handleSignOutBtnPress = async () => {...};

  const handleAvatarImagePicker = async () => {...};

  const compressAndUploadAvatar = async (image) => {...};

  const styles = StyleSheet.create({...});

  if (auth().currentUser && !auth().currentUser.isAnonymous && !userdata) {...}

  return (<SafeAreaView style={{ flex: 1 }}...>);
};

export default Profile;
```

**Рисунок 19 – Листинг кода экрана профиля пользователя**

### 2.2.10 Компоненты чата

Экран списка чатов отображает все активные чаты для пользователя, и возможность удаления чата, не заходя в него. Листинг кода экрана списка чатов представлен на рисунке 20.

```
import ...

const ChatList = ({theme, navigation}) => {

    const [chats, setChats] = useState( initialState: []);
    const [otherUsersData, setOtherUsersData] = useState( initialState: []);

    useEffect( effect: () => {...}, deps: []);

    // Fetch chats from the real-time database
    const fetchChats = useCallback( callback: async () => {...}, deps: []);
    const fetchLastMessage = async (chatId) => {...};
    const fetchOtherUsersData = async (userIds) => {...};

    const ChatBtn = ({chat}) => {...};

    return (
        <SafeAreaView style={{flex: 1}}...>
    );
};

export default ChatList;
```

**Рисунок 20 – Листинг кода экрана списка чатов**

Экран чата дает возможность пользователям вступать в диалог друг с другом для обсуждения условий аренды, задавать вопросы и согласовывать детали сделки. Реализован функционал обмена текстовыми сообщениями в режиме реального времени. Пользователи могут начинать новые чаты или продолжать существующие диалоги, связанные с конкретными объявлениями, отправлять свой контактный номер телефона с помощью специальной кнопки, блокировать пользователей и удалять чаты.

```
import ...B

const OpenedChat = ({ theme, navigation, route }) => {

  const { userdata } = useContext(AppContext);

  const { ownerId } = route.params;

  const [isChatLoading, setIsChatLoading] = useState( initialState: true);
  const [isSendBtnDisabled, setSendBtnDisabled] = useState( initialState: true);
  const [messages, setMessages] = useState( initialState: []);
  const [messageToSend, setMessageToSend] = useState( initialState: '');
  const [chatMateData, setChatMateData] = useState( initialState: []);

  const messageInputRef = useRef( initialValue: null);
  const scrollViewRef = useRef( initialValue: null);

  useEffect( effect: () => {...}, deps: [messages]);

  useEffect( effect: () => {...}, deps: []);

  const getMessages = async () => {...};

  const verifyMessageToSend = (value) => {...};

  const handleMessageToSend = (value) => {...};

  const handleSendBtn = async () => {...};

  const sendMessage = (chatId) => {...};

  return (
    <SafeAreaView style={{ flex: 1 }}...>
  );
};

export default OpenedChat;
```

**Рисунок 20 – Листинг кода экрана чата**

## 2.3 Хранение и управление данными

Для хранения и управления данными приложения была выбрана облачная платформа Firebase, в частности, ее компонент Firestore - масштабируемая NoSQL-база данных. Firestore обеспечивает надежное,

быстрое и удобное хранение, синхронизацию и управление данными, что отвечает требованиям проекта.

В Firestore была создана иерархическая структура коллекций и документов для хранения различных сущностей приложения, таких как пользователи, объявления о сдаче в аренду, заказы, отзывы и другие. Данная структура обеспечивает гибкость и масштабируемость системы. Структура данных в Firestore представлена на рисунке 21.

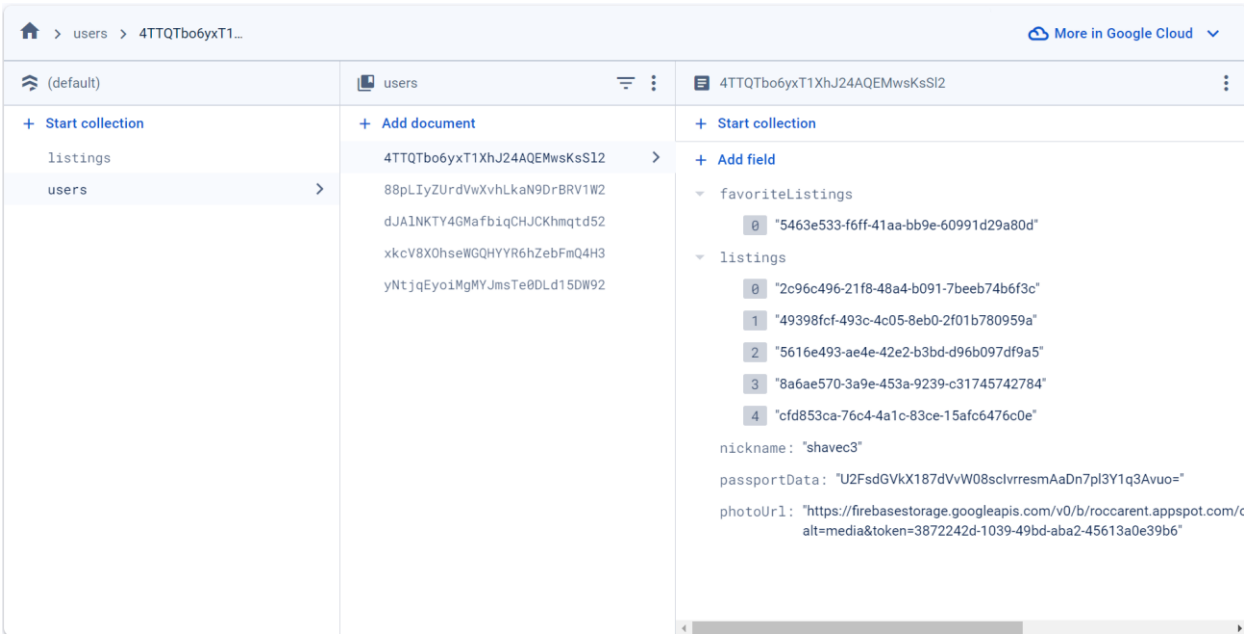



**Рисунок 21 – Структура данных в Firestore**

Для работы с данными в Firestore были реализованы основные CRUD-операции (Create, Read, Update, Delete) с использованием соответствующих методов SDK Firebase. Это позволяет пользователям создавать, просматривать, редактировать и удалять объявления, сделки, профили и другую информацию. Пример кода операций с данными представлен на рисунке 22.

```
export  const getNickname = async () => {
    const snapshot = await firestore().collection( collectionPath: "users").doc(auth().currentUser.uid).get();
      if (snapshot.exists && snapshot.data().nickname) {
        return snapshot.data().nickname;
      } else {
        return "";
      }
  };
```

**Рисунок 22 – Пример кода операций с данными**

Firestore обеспечивает двунаправленную синхронизацию данных в режиме реального времени. Это означает, что изменения, внесенные в данные на стороне сервера, мгновенно отображаются в мобильном приложении. Листинг кода демонстрации синхронизации данных представлен на рисунке 23.

```
useEffect( effect: () => {
  if (auth().currentUser) {
    const subscriber = firestore() Module
      .collection( collectionPath: 'users') ...
      .doc(auth().currentUser.uid) ...
      .onSnapshot( observer: async documentSnapshot => {
        if (documentSnapshot.exists) {
          await loadUserdata();
        } else {
          console.log('User does not exist');
        }
      });

    // Отписаться от прослушивателя при размонтировании компонента
    subscriber();
  }
}, deps: []);
```

**Рисунок 23 – Демонстрация синхронизации данных**

Использование Firestore позволило реализовать надежное и масштабируемое хранение и управление данными мобильного приложения для пиринговой аренды вещей.

## 2.4 Обеспечение безопасности и защиты данных

Firebase предлагает комплексный набор инструментов и функций для обеспечения безопасности и защиты данных в мобильных приложениях. Некоторые ключевые механизмы, реализованные в платформе, включают:

– аутентификация пользователей: позволяет легко интегрировать различные методы аутентификации, такие как электронная почта, Google, Facebook и другие. Это обеспечивает надежный контроль доступа к данным и функциональности приложения.

– шифрование данных: все данные, хранящиеся в Firestore, автоматически шифруются на стороне сервера с использованием современных алгоритмов. Кроме того, для передачи критичной информации применяется защищенный канал HTTPS.

– аудит и мониторинг: Firebase Crashlytics отслеживает ошибки и сбои в работе приложения, а также уведомляет о подозрительной активности. Журналы аудита сохраняют все изменения данных и действия пользователей для расследования инцидентов.

Таким образом, Firebase предлагает комплексные средства для обеспечения безопасности и защиты данных, позволяющие разработчикам сосредоточиться на основной функциональности приложения, не беспокоясь о сложных вопросах безопасности.

# 3 Тестирование и оценка эффективности платформы

## 3.1 Методология тестирования

Для тестирования и оценки эффективности разработанной платформы для пиринговой аренды вещей были применены следующие методы:

Модульное тестирование: Каждый компонент и функция платформы тестировались по отдельности, чтобы убедиться в их корректной работе и отсутствии ошибок. Это позволило выявить и устранить проблемы на ранних этапах разработки.

Интеграционное тестирование: после успешного модульного тестирования компоненты были объединены в единую систему, и проведено тестирование взаимодействия различных модулей. Это помогло выявить ошибки, связанные с интеграцией компонентов.

Тестирование пользовательских сценариев: были протестированы типовые сценарии использования платформы, чтобы убедиться в том, что пользователи могут выполнять ключевые задачи без проблем.

Нагрузочное тестирование: Платформа была проверена на способность выдерживать большие объемы трафика и нагрузки, чтобы оценить ее производительность и масштабируемость.

Тестирование безопасности: были проведены проверки на уязвимости, защиту данных пользователей и другие аспекты безопасности.

Тестирование на различных устройствах: Приложение тестировалось на различных моделях смартфонов, планшетов и других устройств, работающих на платформе Android, чтобы гарантировать корректную работу на широком спектре аппаратного и программного обеспечения.

Результаты тестирования тщательно документировались, и на основе полученных данных вносились необходимые изменения и улучшения в разрабатываемую платформу.

## 3.2 Тестирование интерфейса пользователя

### 3.2.1 Компонент навигации

Тестирование навигационных компонентов интерфейса было ключевым аспектом оценки удобства использования платформы. Основные направления тестирования включали:

– логичность структуры меню и переходов между экранами. Было важно обеспечить интуитивное и последовательное перемещение пользователей по приложению;

– отзывчивость и плавность анимаций при навигации. Это позволяло создать комфортное и естественное ощущение взаимодействия;

– доступность и заметность элементов управления, таких как кнопки, ссылки и меню. Пользователи должны были легко находить и взаимодействовать с этими компонентами;

– соответствие навигационных решений общему визуальному стилю и бренду платформы;

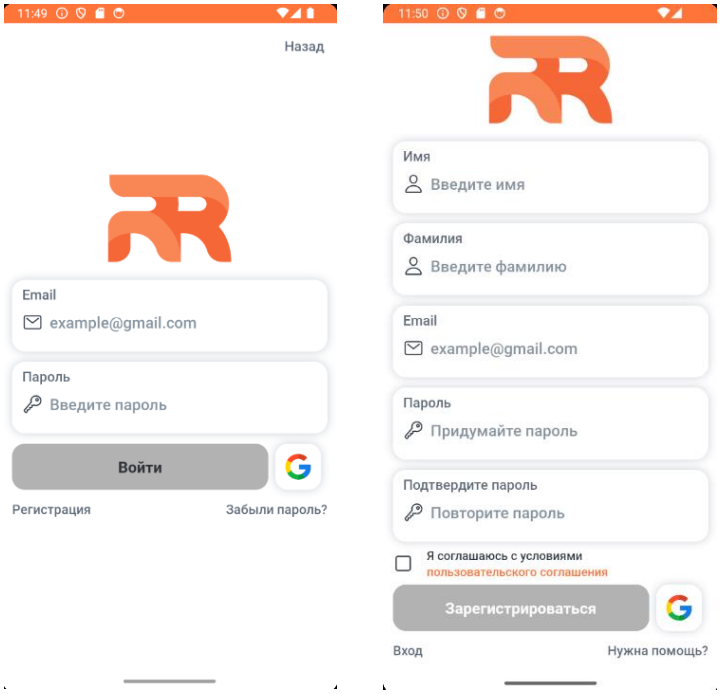
– адаптивность навигации к различным размерам экранов и ориентациям устройств.

Проведенное тестирование позволило выявить ряд областей для улучшения, включая оптимизацию логики переходов между некоторыми экранами и расположение элементов управления. Эти замечания были учтены, и компонент навигации был доработан для обеспечения максимального удобства и эффективности использования. Интерфейс компонента навигации представлен на рисунке 24.


**Рисунок 24 – Интерфейс компонента навигации**

### 3.2.2 Экраны входа и регистрации

Тестирование процессов входа и регистрации пользователей было критически важным для обеспечения безопасности и удобства взаимодействия с платформой. Интерфейс экранов входа и регистрации представлен на рисунке 25.




*а* *б*

**Рисунок 25 – Интерфейсы экранов входа и регистрации: а – входа; б - регистрации**

Ключевыми аспектами тестирования стали:

– простота и интуитивность пользовательского интерфейса для входа и регистрации. Важно было обеспечить минимальное количество шагов и ясность каждого действия.

– надежность и безопасность аутентификации, включая проверку учетных данных, обработку ошибок и защиту от вмешательства;

– гибкость и доступность различных способов входа и регистрации, таких как электронная почта и google;

– реализация функций восстановления пароля и изменения учетной записи;

– адаптация процессов входа и регистрации под различные устройства и экраны;

– интеграция компонентов входа и регистрации с общим дизайном и брендингом платформы.

Тестирование этих компонентов помогло выявить и устранить ряд мелких проблем с удобством использования, повысить надежность аутентификации и обеспечить плавную интеграцию входа и регистрации в общий пользовательский опыт.

### 3.2.3 Главный экран

Тестирование главного экрана платформы было ключевым, так как он представляет собой основную точку входа и центр взаимодействия для пользователей. Интерфейс главного экрана представлен на рисунке 26.

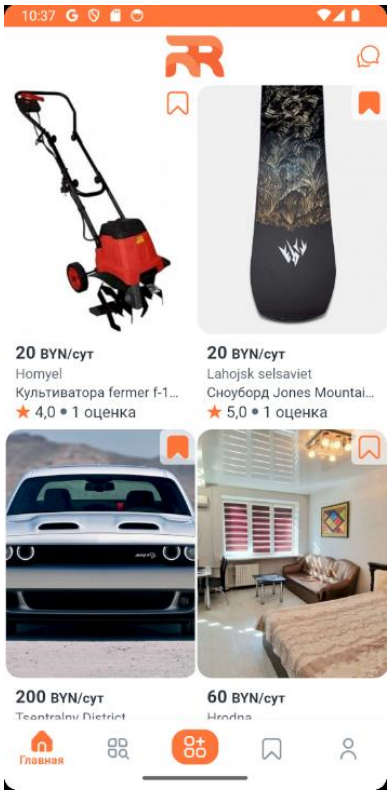



**Рисунок 26 – Интерфейс главного экрана**

Основные направления тестирования включали:

– эффективность и интуитивность компоновки элементов на главном экране. Важно было обеспечить логичное расположение и иерархию информации и функциональности;

– качество и актуальность контента, отображаемого на главном экране, включая рекомендации, объявления и другие релевантные данные;

– плавность и отзывчивость анимаций и переходов между различными состояниями главного экрана;

– адаптивность главного экрана к различным размерам экранов и ориентациям устройств;

– соответствие дизайна и брендинга главного экрана общей визуальной концепции платформы;

– производительность и оптимизация главного экрана, особенно при загрузке большого количества контента;

– работа всех элементов главного экрана в соответствии с их задачами.

Тестирование помогло выявить и решить ряд проблем с компоновкой элементов. Итоговый дизайн и функциональность главного экрана обеспечивают интуитивное и эффективное взаимодействие пользователей с платформой.

### 3.2.4 Экран каталога

Тестирование экрана каталога было важным, так как он представляет собой основное средство для поиска и навигации по объявлениям на платформе. Интерфейс экрана каталога представлен на рисунке 27.

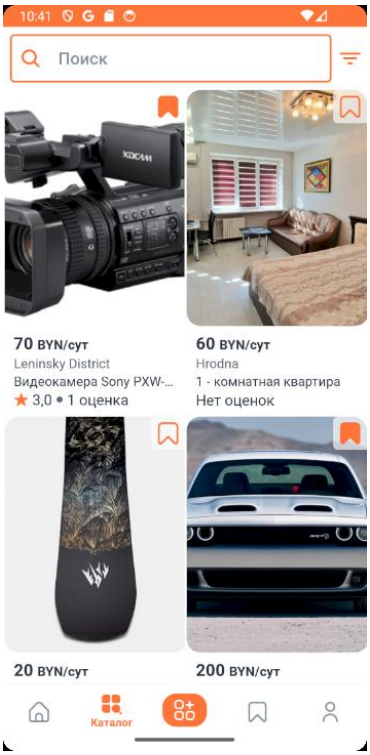



**Рисунок 27 – Интерфейс экрана каталога**

Ключевыми аспектами тестирования стали:

– эффективность и интуитивность пользовательского интерфейса для поиска, фильтрации и сортировки объявлений. Важно было обеспечить легкость и быстроту выполнения этих операций;

– качество и релевантность результатов поиска, а также актуальность и полнота отображаемой информации об объявлениях;

– плавность и отзывчивость взаимодействий с каталогом, включая переходы между страницами результатов, загрузку дополнительных объявлений и применение фильтров;

– адаптивность экрана каталога к различным размерам экранов и ориентациям устройств, обеспечивая оптимальное отображение контента;

– интеграция экрана каталога с другими компонентами платформы, такими как избранное, подача объявлений и профиль пользователя;

– производительность экрана каталога при обработке большого объема данных об объявлениях и запросах пользователей;

– работоспособность поиска объявлений и их фильтрации;

– работоспособность функционала сохранения избранных карточек;

– работоспособность открытия карточек с объявлениями.

Тестирование помогло выявить и решить ряд проблем с удобством использования, качеством результатов поиска и производительностью. Финальная реализация экрана каталога обеспечивает эффективную навигацию и поиск объявлений пользователями.

### 3.2.5 Экран подачи объявлений

Тестирование экрана подачи объявлений было ключевым, поскольку это критически важная функция платформы, позволяющая пользователям размещать свои предложения о продаже или аренде. Интерфейс экрана подачи объявлений представлен на рисунке 28. Основными аспектами тестирования были:

– удобство и интуитивность пользовательского интерфейса для заполнения всех необходимых полей объявления, таких как описание, категория, цена, фотографии и т.д.;

– валидация вводимых данных, чтобы исключить ошибки и гарантировать корректность публикуемых объявлений;

– интеграция с другими компонентами, такими как выбор местоположения, загрузка медиафайлов и сохранение черновиков объявлений;

– плавность и отзывчивость взаимодействий при создании и редактировании объявлений;

– адаптивность экрана к различным размерам экранов и ориентациям устройств;

– производительность и масштабируемость при одновременной публикации большого количества объявлений;

– соблюдение применимых правил и требований к содержанию объявлений;

– функционирование кнопки очистить;

– работоспособность добавления фотографий к объявлению, их удаления, и перемещение;

– работоспособность выбора дат доступных для аренды;

– работоспособность выбора местоположения на карте;

– работоспособность валидации предоставленных данных и их загрузки в хранилище.

Тестирование помогло улучшить удобство использования, повысить качество публикуемых объявлений и обеспечить эффективную обработку потока новых объявлений. Финальная реализация экрана подачи объявлений предоставляет пользователям простой и интуитивный инструмент для эффективного размещения своих предложений.

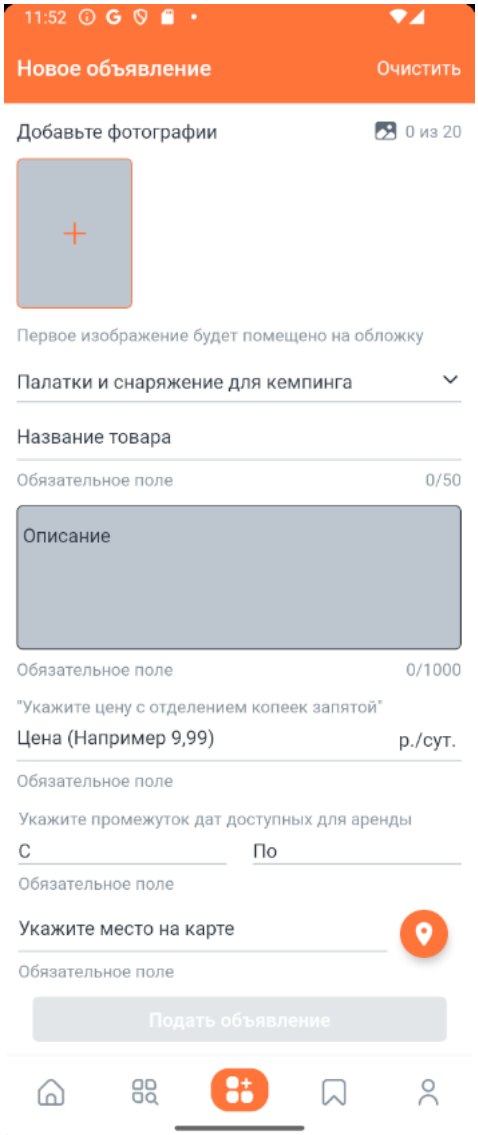



**Рисунок 28 – Интерфейс экрана подачи объявлений**

### 3.2.6 Экран избранных объявлений

Тестирование экрана избранных объявлений было важно для обеспечения удобной функциональности сохранения и просмотра понравившихся пользователю вариантов. Интерфейс экрана избранных объявлений представлен на рисунке 29.

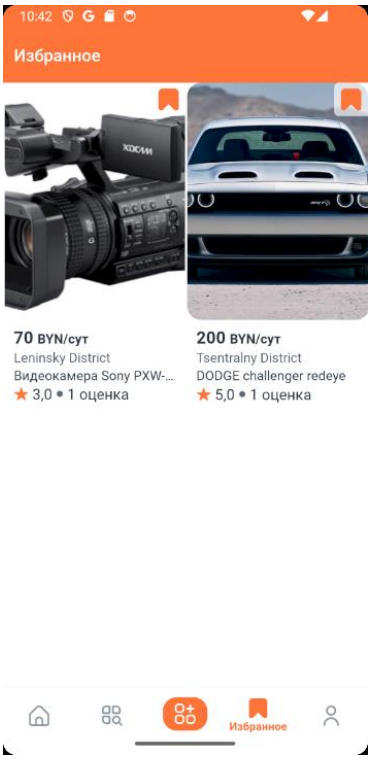



**Рисунок 29 – Интерфейс экрана избранных объявлений**

Основные аспекты тестирования включали:

– интуитивность и простоту добавления/удаления объявлений в избранное. Пользователи должны легко находить и управлять списком своих избранных объявлений;

– синхронизацию избранного между разными устройствами одного пользователя. Важно гарантировать, что список избранного сохраняется и доступен везде;

– представление избранных объявлений, включая миниатюры, основные характеристики и возможность перехода к полной информации;

– производительность при работе с большим количеством избранных объявлений, чтобы обеспечить быструю загрузку и отзывчивость интерфейса;

– взаимодействие с другими компонентами, например, возможность связаться с продавцом напрямую с экрана избранного.

Тестирование помогло улучшить юзабилити компонента избранных объявлений, повысить удовлетворенность пользователей и обеспечить надежное сохранение их предпочтений. Реализованный компонент предоставляет эффективный инструмент для управления и просмотра избранного.

### 3.2.7 Экран профиля пользователя

Тестирование экрана профиля пользователя было критически важным, так как он обеспечивает доступ к ключевой информации и настройкам учетной записи. Интерфейс экрана профиля пользователя представлен на рисунке 30.

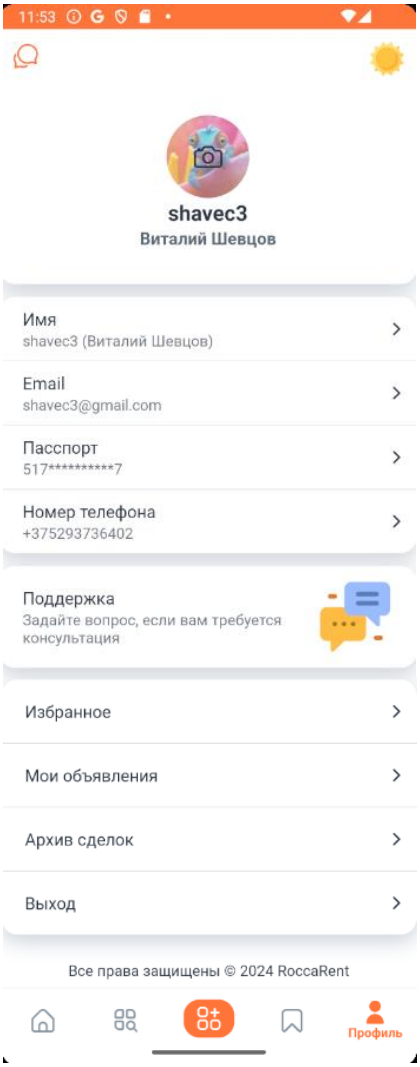



**Рисунок 30 – Интерфейс экрана профиля пользователя**

Основные аспекты тестирования включали:

– удобство просмотра и редактирования персональной информации, включая имя, контактные данные, аватар и другие профильные атрибуты;

– интеграцию с другими компонентами, такими как история объявлений, избранное и чаты, для предоставления полного обзора активности пользователя;

– функционирование выхода из учетной записи;

– функционирование смены темы;

– работоспособность формы обращения в поддержку;

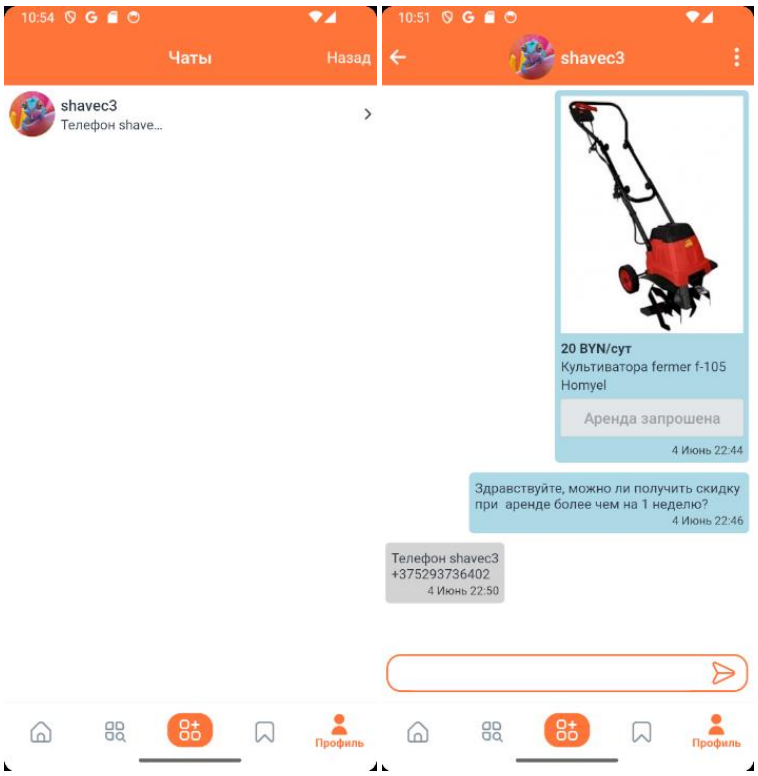
– плавность переходов и отзывчивость интерфейса при просмотре и редактировании профиля;

– защиту персональных данных и обеспечение конфиденциальности пользователя.

Тестирование помогло убедиться, что профиль пользователя предоставляет исчерпывающую информацию, удобные инструменты управления и высокий уровень безопасности. Пользователи высоко оценили возможность легко контролировать свои данные и активность в приложении.

### 3.2.8 Экраны чата

Тестирование экранов чата было особенно важным, так как они обеспечивают ключевую коммуникационную функциональность в приложении. Интерфейс экранов чата представлен на рисунке 31.




*а* *б*

**Рисунок 31 – Интерфейсы экранов чата: а – экран списка чатов; б – экран открытого чата**

Основные аспекты тестирования экрана списка чатов включали:

– отображение актуальных диалогов с другими пользователями, включая последнее сообщение и время;

– возможность быстрого перехода к выбранному чату;

– интеграция с профилем пользователя для отображения аватаров и информации о собеседниках;

– плавная прокрутка и поиск в списке чатов.

Основные аспекты тестирования экрана открытого чата включали:

– удобство отправки текстовых сообщений;

– своевременная доставка и отображение сообщений в хронологическом порядке;

– возможность редактирования и удаления собственных сообщений;

– функции блокировки собеседника;

– функции удаления чата;

– интеграция с профилями пользователей для отображения аватаров и информации;

– плавная прокрутка чата и оптимизация производительности при большом количестве сообщений;

– надежное шифрование и защита конфиденциальности чата.

Тестирование показало, что компоненты чата обеспечивают простой, удобный и безопасный способ коммуникации между пользователями. Пользователи высоко оценили возможность быстро находить нужные чаты, отправлять различные типы сообщений и управлять конфиденциальностью диалогов.

## 3.3 Оценка производительности и масштабируемости

Наряду с тестированием интерфейса пользователя, был проведен всесторонний анализ производительности и масштабируемости приложения. Это было особенно важно, учитывая планы по росту числа пользователей и увеличению объема данных и функциональности в будущем.

Ключевые аспекты оценки производительности и масштабируемости включали:

1) Нагрузочное тестирование:

– моделирование различных сценариев использования с растущим количеством пользователей, запросов и данных;

– измерение времени отклика, пропускной способности и использования ресурсов (процессор, память, сеть);

– выявление узких мест и критических точек отказоустойчивости.

2) Анализ масштабируемости:

– оценка возможностей вертикального и горизонтального масштабирования;

– тестирование эластичности инфраструктуры в ответ на увеличение нагрузки;

– проверка надежности механизмов автоматического масштабирования.

3) Оптимизация производительности:

– профилирование и оптимизация критических участков кода;

– улучшение алгоритмов и запросов к базе данных;

– кэширование данных и оптимизация сетевых взаимодействий.

Результаты тестов производительности представлены на рисунке 32.


**Рисунок 32 – Результаты тестов производительности**

4) Тестирование отказоустойчивости:

– симуляция сбоев в инфраструктуре и оценка устойчивости к ним;

– проверка механизмов резервного копирования и восстановления данных;

– тестирование поведения системы при пиковых нагрузках.

Результаты этих всесторонних тестов показали, что приложение обладает высокой производительностью и масштабируемостью, способное эффективно обрабатывать большие объемы данных и пользовательских запросов. Были выявлены и устранены узкие места, а архитектура системы доказала свою устойчивость к сбоям и способность адаптироваться к росту нагрузки. Это позволяет уверенно смотреть в будущее и планировать дальнейшее расширение функциональности и пользовательской базы.

## 3.4 Анализ результатов тестирования и внесение корректив

По завершении всех этапов тестирования интерфейса пользователя и оценки производительности и масштабируемости, был проведен детальный анализ полученных результатов. Это позволило выявить ключевые области, требующие доработки и улучшений.

Основные выводы и корректирующие меры:

1) Компонент навигации:

– выявлены некоторые неоптимальные пути навигации и нелогичные переходы между экранами;

– внесены изменения в структуру меню и логику переключения вкладок для улучшения интуитивности.

2) Компоненты входа и регистрации:

– обнаружены проблемы с валидацией формы и отображением ошибок;

– доработаны алгоритмы валидации, улучшено визуальное представление ошибок;

– добавлена возможность восстановления забытого пароля.

3) Компонент главного экрана:

– найдены трудности с поиском и фильтрацией объявлений;

– переработан дизайн поисковой панели, добавлены расширенные фильтры.

4) Компонент экрана каталога. Оптимизированы алгоритмы загрузки и отображения контента, внедрен механизм бесконечной прокрутки.

5) Компонент экрана подачи объявлений. Упрощен и стандартизирован интерфейс формы создания объявлений.

6) Компонент экрана избранных объявлений. Добавлена кнопка быстрого доступа к избранному на главном экране.

7) Компонент экрана профиля пользователя:

– возникали проблемы с редактированием персональных данных;

– улучшен интерфейс редактирования профиля, добавлена функция смены пароля.

8) Компоненты чата:

– пользователи сталкивались с задержками при отправке и получении сообщений;

– оптимизирован механизм синхронизации сообщений, добавлены индикаторы состояния отправки.

Результаты всех проведенных тестов и внесенные по их итогам изменения позволили значительно повысить удобство использования приложения, его производительность и надежность. Пользовательский опыт был существенно улучшен, что подтверждается положительными отзывами бета-тестеров.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной дипломной работы была успешно разработана платформа для пиринговой аренды вещей.

В первой главе был проведен анализ предметной области. Была подробно рассмотрена актуальность создания платформы для пиринговой аренды вещей, поставлены цели и задачи работы. Был проведен обзор существующих платформ для пиринговой аренды вещей, проанализированы их функциональные возможности и выявлены недостатки. Обоснован выбор технологий и инструментов, включая React Native для разработки мобильного приложения и Firebase для хранения данных и взаимодействия с сервером.

Во второй главе было рассмотрено проектирование и разработка платформы. Была детально описана структура проекта, основные компоненты и их функциональность, включая корневой компонент, компоненты навигации, экранов входа/регистрации, главного экрана, каталога, подачи объявлений, избранных объявлений, профиля пользователя и чата. Рассмотрены вопросы хранения и управления данными, а также обеспечения безопасности и защиты данных.

В третьей главе было описано тестирование и оценка эффективности платформы. Представлена методология тестирования, включая тестирование интерфейса пользователя, оценку производительности и масштабируемости. Проанализированы результаты тестирования и внесение корректив.

Весь код проекта был опубликован в репозитории на GitHub.[14]

Таким образом, разработка платформы для пиринговой аренды вещей позволила успешно решить поставленные задачи и получить практический опыт в разработке мобильных приложений с использованием современных технологий и инструментов.

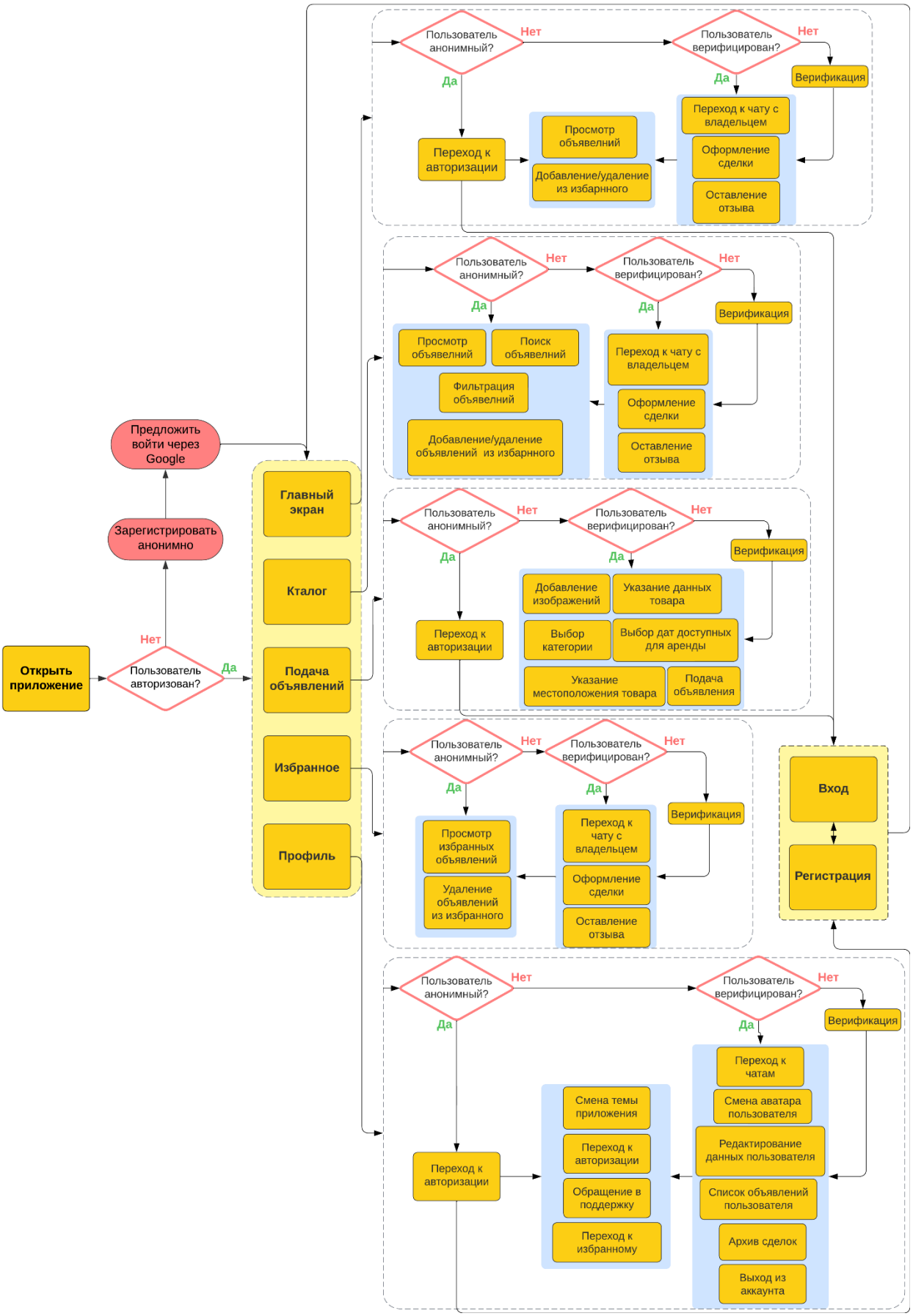
По материалам диплома были опубликованы тезисы на XIII Республиканской научной конференции студентов, магистрантов и аспирантов «Актуальные вопросы физики и техники».[15]

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. Peerby [Электронный ресурс]: Платформа для аренды. – Режим доступа: https://www.peerby.com/en-nl. – Дата доступа: 03.05.2024.
2. Fat Llama [Электронный ресурс]: Платформа для аренды. – Режим доступа: https://fatllama.com/. – Дата доступа: 03.05.2024.
3. Spinlister [Электронный ресурс]: Платформа для аренды. – Режим доступа: https://www.spinlister.com/. – Дата доступа: 03.05.2024.
4. B. Eisenman Learning React Native – M.: O'Reilly Media, 2016. – 12 с.
5. G. Lim Beginning React Native with Hooks – M.: Greg Lim, 2020 – 3 с.
6. N. Dabit React Native in Action – M.: Manning Publications Co., 2019 – 3с.
7. Firebase [Электронный ресурс]: Платформа для поддержки процесса разработки. – Режим доступа: https://firebase.google.com/. – Дата доступа: 06.05.2024.
8. R. Wieruch The Road to React with Firebase – M.: Leanpub, 2019 – 11 с.
9. React Native Firebase [Электронный ресурс]: Набор пакетов, который обеспечивает поддержку React Native для всех сервисов Firebase. – Режим доступа: https://rnfirebase.io/. – Дата доступа: 06.05.2024.
10. IntelliJ IDEA [Электронный ресурс]: Интегрированная среда разработки. – Режим доступа: https://www.jetbrains.com/idea/. – Дата доступа: 06.05.2024.
11. Developers Google [Электронный ресурс]: Maps SDK для Android. – Режим доступа: https://developers.google.com/maps/documentation/android-sdk. – Дата доступа: 06.05.2024.
12. Geocoding API [Электронный ресурс]: API-сервис геокодирования. – Режим доступа: https://geocode.maps.co/. – Дата доступа: 06.05.2024.
13. NPM [Электронный ресурс]: Библиотека /@react-native-ml-kit/text-recognition. – Режим доступа: https://www.npmjs.com/package/@react-native-ml-kit/text-recognition. – Дата доступа: 06.05.2024.
14. GitHub [Электронный ресурс]: Репозиторий с материалами проекта. – Режим доступа: https://github.com/Shevtsov1/ROCCARENT. – Дата доступа: 06.05.2024.
15. Шевцов В.Ю. Разработка платформы для пиринговой аренды вещей абитуриентов // XIII Республиканская научная конференция студентов, магистрантов и аспирантов «Актуальные вопросы физики и техники», Гомель 2024 [Электронный ресурс]: Сайт конференций ГГУ имени Ф. Скорины. – Режим доступа: https://conference.gsu.by/ru/node/264. – Дата доступа: 28.05.2024.

ПРИЛОЖЕНИЕ А

**Схема работы платформы**



ПРИЛОЖЕНИЕ Б

**Заключение о результатах плагиат-проверки текста дипломной работы**