Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И.Ульянова (Ленина)
(СПбГЭТУ «ЛЭТИ»)

Направление	09.03.04 – Программная инженерия
Профиль	Разработка программно-информационных систем
Факультет	ФКТИ
Кафедра	МО ЭВМ

К защите допустить

Зав. кафедрой Кринкин К.В.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Тема: Разработка клиентской части мобильного приложения для организации работы с документами физических и юридических лиц на основе технологии блокчейн

Студент		_____ *подпись*	Кадыров Р.Р.
Руководитель	к.т.н., доцент *(Уч. степень, уч. звание)*	_____ *подпись*	Яценко И.В.
Консультанты	*(Уч. степень, уч. звание)*	_____ *подпись*	Барышников С.С.
	(Уч. степень, уч. звание)	_____ *подпись*	Лебедева Т.Н.
	к.т.н., доцент *(Уч. степень, уч. звание)*	_____ *подпись*	Лисс А.А.

Санкт-Петербург

2019

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Утверждаю
Зав. кафедрой МО ЭВМ
_____ Кринкин К.В.
«___»_____20___ г.

Студент Кадыров Р.Р. Группа 5303

Тема работы: Разработка клиентской части мобильного приложения для организации работы с документами физических и юридических лиц на основе технологии блокчейн

Место выполнения ВКР: ООО «Блокчейн Солюшнз Групп»

Исходные данные (технические требования): операционная система Android, язык программирования Java, база данных BigchainDB

Содержание ВКР: «Введение», «Обзор предметной области», «Выбор метода решения», «Описание метода решения», «Исследование свойств решения», «Технико-экономическое обоснование», «Заключение», «Приложение А», «Приложение Б», «Приложение В»

Перечень отчетных материалов: текст ВКР, иллюстративный материал

Дополнительные разделы: Экономическое обоснование ВКР

Дата выдачи задания
«_22_»__апреля__2019 г.

Дата представления ВКР к защите
«_19_»___июня__2019 г.

Студент _____ Кадыров Р.Р.

Руководитель _____ Яценко И.В.

Консультант _____ Барышников С.С.

КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю
Зав. кафедрой МО ЭВМ
_____ Кринкин К.В.
«___»_____20___ г.

Студент Кадыров Р.Р. Группа 5303
Тема работы: Разработка клиентской части мобильного приложения для организации работы с документами физических и юридических лиц на основе технологии блокчейн

№ п/п	Наименование работ	Срок выполнения
1	Обзор литературы по теме работы	22.04 – 24.04
2	Анализ предметной области	22.04 – 26.04
3	Проектирование бизнес-процессов производства	26.04 – 05.05
4	Разработка и тестирование системы	06.05 – 23.05
5	Оценка экономической эффективности проекта	24.05 – 28.05
6	Оформление пояснительной записки	28.05 – 15.06
7	Оформление иллюстративного материала	15.06 – 16.06
8	Предзащита	17.06

Студент _____ Кадыров Р.Р.
Руководитель _____ Яценко И.В.
Консультант _____ Барышников С.С.

РЕФЕРАТ

Пояснительная записка 111 стр., 24 рис., 11 табл., 25 ист., 03 прил.

БЛОКЧЕЙН, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, СМАРТ-КОНТРАКТ, ДОКУМЕНТООБОРОТ, КЛИЕНТСКАЯ ЧАСТЬ.

Объектом разработки является создание клиентской части мобильного приложения, которое будет работать на основе блокчейн и поддерживать функцию передачи активов посредством технологии смарт-контрактов.

Цель работы – раскрыть выбор решений в клиентской части мобильного приложения для организации работы с документами физических и юридических лиц на основе технологии блокчейн.

В данной работе были исследованы все классические решения, а также различные типы решений задачи документооборота с применением технологии блокчейн. Было доказано, почему разработанное решение Multipass.One превосходит остальные по заданным критериям и какие методы с клиентской точки зрения были применены для реализации данного продукта. В качестве элемента для управления продуктом было выбрано мобильное приложение. Была описана архитектура разработанного продукта, на примере документа показано устройство базы данных. В данной работе был оценен размер клиентской части приложения, скорость роста базы данных при добавлении новых данных, время поиска файла в блокчейне. Также было проведено исследование, в котором приняло участие 1073 респондента. Данное исследование помогло раскрыть Usability разработанного продукта на основе одного из сценария использования, а также было произведено сравнение полученных данных с результатами предельно быстрой скорости выполнения данного сценария. И в заключении были сделаны выводы о качестве разработанного продукта и дальнейшие перспективы его развития.

ABSTRACT

In this paper, all classical solutions were investigated, as well as various types of solutions for the workflow problem using blockchain technology. It was proved why the developed solution Multipass.One surpasses the rest according to specified criteria and which methods from a client point of view were applied to the implementation of this product. A mobile application has been selected as an element for product management. The architecture of the developed product was described, the example of the document shows the database device. In this work, we estimated the size of the client part of the application, the growth rate of the database when adding new data, the time it took to find the file in the blockchain. A study was also conducted in which 1073 respondents participated. This study helped to reveal the usability of the developed product on the basis of one of the use cases, and it also made a comparison of the data obtained with the results of the extremely fast speed of execution of this scenario. And in the conclusion conclusions were drawn about the quality of the developed product and further prospects for its development.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяют следующие термины с соответствующими определениями:

Блокчейн – выстроенная по определённым правилам цепочка из формируемых блоков транзакций, ориентированная на обеспечение взаимодействия большого количества пользователей между собой без использования «доверенных посредников». [1, 2]

Смарт-контракт – электронные алгоритмы, которые могут автоматически исполняться при определенных условиях. [3]

ЭЦП – Электронная цифровая подпись.

APK – Android Package.

BitIndex25 – Выбранный контур блокчейна.

iOS – iPhone Operating System.

JSON – JavaScript Object Notation.

Multipass.One – название разрабатываемого продукта в работе.

ВВЕДЕНИЕ

Актуальность решаемой в ВКР проблемы: одно из самых важных открытий и актуальных решений, используемых в IT-сфере - блокчейн-технология, прекрасно переносится на область документооборота, в которой уже давно назрели реформы. Однако, сейчас актуальные решения в этой области находятся на стадии низкой готовности, потому что разработчики таких решений либо только размещают свои идеи, либо их наработки находятся в стадии закрытого альфа-теста. Отсутствием точной законодательной базы, законченных решений и глубоких исследований в области взаимодействия документооборота и блокчейна мотивируется данная работа.

Цель работы: раскрыть выбор решений в клиентской части программного продукта, а именно мобильного приложения, для организации работы с документами физических и юридических лиц на основе технологии блокчейн, а также отображение интерфейса передачи активов посредством смарт-контрактов со стороны пользователя системы

Для достижения цели необходимо решить следующие **задачи:**

- Рассмотреть актуальные решения в сфере документооборота, и из анализа выделить необходимые требования для дальнейшей разработки.
- Реализовать клиентскую часть мобильного приложения на технологии блокчейн, предоставляющую возможность передачи активов посредством смарт-контрактов.
- Проанализировать и сделать вывод о качестве разработанного программного продукта по критериям и требованиям, введенным при выполнении первой задачи.

Объектом исследования является создание клиентской части мобильного приложения, которое будет работать на основе блокчейн и

поддерживать функцию передачи активов посредством технологии смарт-контрактов.

Предметом исследования является взаимодействие мобильного приложения с сетью блокчейна с последующим отображением полученных данных.

Практическая значимость решения заключается в том, что разработанный продукт можно использовать внутри коммерческих организаций или в повседневной жизни для увеличения скорости работы в сфере документооборота.

1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Принцип отбора аналогов

Блокчейн на современном этапе экономического развития интересен сфере документооборота. Помимо названной технологии существует еще три типа ведения данной деятельности (классический, электронный, облачный). На базе технологии блокчейн также присутствуют различные решения, однако выбраны были наиболее отличающиеся по подходу к реализации данной темы. Решения на блокчейне являются мобильными приложениями. Аналоги были отсортированы по историческому признаку.

Классический способ

Данный тип ведения документооборота общепринят и понятен. Классический способ ведения учета с помощью бумаг, в котором при заключении контракта должны присутствовать контрагенты и третье незаинтересованное лицо. Данный способ ведения дел очень дорог и имеет низкую скорость воспроизведения. [4] Для каждой сделки нужно подготавливать и проверять документы, также оплачивать услуги нотариуса. При ведении дел таким методом необходимо тратить деньги на услуги печати, на заверение доверенностей, а при нарушении условий дополнительно оплачивать повторную процедуру. Печатные документы легко подделать, также высока вероятность человеческой ошибки и присутствует риск потери информации от амортизации бумаги. Несмотря на то, что при классическом способе поддерживаются любые документы и можно передавать любые активы, имеют место и такие недостатки, как: необходимо обращаться во множество других органов для согласования документов, а также необходимо тратить большое количество времени для поиска нужной операции (документа).

Электронная почта

Способ, при котором бумаги отправляются на почту контрагенту. Данный способ удобен для обсуждения пунктов сделки, но не для

заключения ее, так как утвержденный документ необходимо либо предоставить в бумажном виде и заключать его классическим способом, либо подтвердить электронной цифровой подписью (ЭЦП) с помощью сторонних программ. [4] С использованием электронной почты процесс согласования значительно ускорился относительно классического способа, а также стал дешевле (например, не нужно тратить деньги на поездки на встречи и совещания для согласования). Документы при данном способе ведения дел стало сложнее подделать, однако фальсификация при скреплении договоров после согласования осталась. Никуда не делась вероятность человеческой ошибки, и к тому же, теперь появился риск взлома почты. Данный способ, как и классический, поддерживает любые типы документы и способен оформлять сделки с любыми типами активов, однако теперь необходимо использовать стороннюю программу для получения ЭЦП для согласования. Время, которое необходимо затратить на поиск нужного документа, значительно уменьшилось по сравнению с классическим способом, но необходима очень точная иерархия для быстрого поиска, чем часто пренебрегают при работе.

Облачное хранилище

Данный тип решения более универсален для ведения дел, чем предыдущие два аналога. В нем имеет место удобный способ работы с пунктами сделки, заключения ее и хранения. Однако облачное хранилище слабо защищено, так как максимально централизованно. [4] Достаточно взломать один сервер и злоумышленник будет иметь доступ ко всем документам. У данного способа нет возможности подделать документы после заключения сделки, однако в ходе оформления сделки также остается вероятность человеческой ошибки (документы контрагента необходимо проверять самостоятельно). Также присутствует огромный риск взлома облачного хранилища. Данный способ, как и предыдущие, поддерживает любые типы документов и способен оформлять сделки с любыми типами активов, а также необходимо обращаться в стороннюю программу (ЭЦП) для

согласования. Из преимуществ необходимо отметить малое количество времени для поиска нужной операции, так как данные технологии заточены под работу с документами.

Платформа DocSensus

Данное решение от компании Deloitte представляет собой одно из первых решений проблемы документооборота на базе блокчейна. Основной задачей разработчики данного решения на настоящий момент ставят упрощение корпоративного документооборота. [5] В перспективе создатели проекта видят развитие в переносе решений на другие сферы документооборота. Однако, решение DocSensus на базе блокчейна Emercoin [6] интересно рассмотреть, как представителя проектов, которые решают точечные (корпоративные, страховые, недвижимость) проблемы в большой сфере документооборота. Так как данное решение создано с использованием технологии блокчейн, то отсутствует возможность подделать документ после заключения сделки и риск взлома облачного хранилища. Также данное программное обеспечение исключает любую вероятность человеческой ошибки, так как делает все без участия пользователя. Однако, данная платформа поддерживает только корпоративные документы, в ней нельзя передавать активы. На платформе DocSensus все действия происходят в рамках одной программы. А также необходимо тратить малое количество времени для поиска нужной операции, так как решение заточено исключительно на решении поставленной задачи.

Платформа Endo

Решение от команды из России представляет собой протокол верификации данных. [7] Данное решение интересно рассмотреть, как представителя проектов, которые создают целую экосистему. На базе протокола разработчики создают целый ряд приложений по работе с документами, отзывами, услугами B2B (Business to Business), KYC (Know Your Customer) и т.д. Все документы, на момент совершения сделки, уже верифицированы, поэтому нет необходимости использовать третью сторону.

Аналогично DocSensus, данная платформа способна работать без контроля человека. Данное решение поддерживает любые документы, но имеет ограниченную возможность передавать активы. Из недостатков выделяется то, что все действия происходят в рамках множества программ (для проведения сделок одна программная среда, для работы с документами - другая [7]).

После выбора аналогов для реализации сравнения необходимо выбрать критерии.

1.2. Критерии сравнения аналогов

Стоимость операции

Любое решение должно экономить деньги людей, которые его используют. Если для того, чтобы воспроизвести операцию каждый раз необходимо тратить деньги на повторный сбор документов, деньги на ожидание ответа от принимающей стороны, то решение оказывается неинтересным для использования.

Защищенность операции

В данном критерии важно понять то, как защищен способ ведения документооборота, и вероятность возникновения ошибки при работе.

Удобство использования

В данный критерий помещены несколько разделов для анализа. Если выбранное решение поддерживает малое количество документов, не имеет возможности работы с активами, для реализации действий необходимо обращаться к другим решениям и проектам, а для поиска нужной операции тратить большое количество времени, то использование такого инструмента неудобно.

По вышеописанным критериям была составлена таблица сравнения приведенных аналогов (см. табл. 1.1).

Таблица 1.1 – Сравнение аналогов по критериям

Аналоги	Стоимость операции	Защищенность операции	Удобство использования
Классический	Высокая	Низкая	Низкое
Электронная почта	Средняя	Низкая	Среднее
Облачное хранилище	Средняя	Средняя	Среднее
DocSensus	Низкая	Высокая	Среднее
Endo	Низкая	Высокая	Среднее

Из приведенной таблицы необходимо выделить выводы сравнения выбранных аналогов.

1.3. Выводы по итогам сравнения

С помощью технологии блокчейн существует возможность быстрого и безопасного документооборота. На сегодняшний день, помимо блокчейн решения, существует еще 3 различных способа (классический, электронный, облачный), и все эти решения имеют недостатки представленные в таблице сравнения по критериям (см. табл. 1).

Однако, и на базе блокчейн возникают различные решения, которые пытаются исправить недостатки устаревших способов ведения документооборота. Решения на блокчейне можно разделить на три типа:

- Решение в определенной области.
- Решения на множестве областей, но разными проектами в рамках одной экосистемы.
- Решения, объединяющие разные области документооборота в одном проекте.

Решения, объединяющие разные области документооборота в одном проекте, являются наиболее предпочтительным по названным ранее критериям. Разрабатываемый проект Multipass.One является решением третьего типа. Данный продукт объединяет в себе преимущества классических методов решения задач документооборота и выгоды блокчейн-технологии, а также максимально устраняет недостатки более ранних аналогов. Особенно это видно по ранее введенному критерию «Удобства использования».

Решения на блокчейне не лишены недостатков, но связаны эти минусы исключительно с методами и способами реализации технической части, которые со временем будут преодолены. Также в рамках исследования аналогов должен быть проведен эксперимент, который бы сравнил скорость использования трех типов решений на блокчейн, но на момент написания данной работы, два решения (DocSensus, Endo, которые представляют мобильные приложения), из трех находятся в стадии «закрытого» тестирования, которое не предоставляет возможности массового теста в области Usability. Такой эксперимент будет произведен для разработанного решения Multipass.One в подразделе «Интерфейс» в разделе «Исследование свойств решения».

2. ВЫБОР МЕТОДА РЕШЕНИЯ

Выводы по итогам сравнения показали, что решения аналоги имеют недостатки, которые следует учитывать при разработке собственного продукта. Исходя из обзора аналогов можно сделать вывод о том, что решение должно представлять собой программный продукт (мобильное приложение), предоставляющий пользователю удобный интерфейс. В разработке данный продукт будет иметь название Mutipass.One. Чтобы решение подходило под все обозначенные критерии, оно должно обладать следующими качествами:

1. Доступ к документам. Приложение должно иметь возможность максимум в три «клика» получить доступ к любому документу, который хранится на платформе.
2. Скорость работы. Приложение должно обрабатывать запрос на выдачу нужных документов до 3 минут (обосновано скоростью работы сети блокчейн).
3. Защита данных. В клиентской части приложения не должно быть программных недоработок, которые бы позволили злоумышленникам украсть какие-либо данные.
4. Исключение ошибок. Продукт должен максимально исключить любые человеческие ошибки при работе с ним.
5. Типы документов. Приложение должно обрабатывать следующие типы документов [8, 9]:
    - Основные документы, удостоверяющие личность в Российской Федерации.
    - Национальные идентификационные номера.
    - Документы об актах гражданского состояния.
    - Документы об образовании.
    - Документы, связанные с воинской обязанностью.

- Документы, связанные с владением и эксплуатацией транспортных средств.
- Прочие документы.

6. Работа с активами. Приложение должно иметь возможность работать с активами (продажа, дарение, аренда, завещание).
7. Целостность. Разработанный продукт должен поддерживать все функции в едином приложении без обращения к сторонним проектам.
8. Наглядность. Приложение должно быть максимально понятно пользователю. Не должно быть лишних, неиспользуемых кнопок, окон и т.д.

На основе данных качеств необходимо описать метод решения.

3. ОПИСАНИЕ МЕТОДА РЕШЕНИЯ

3.1. Стек используемых технологий

Так как решение изначально разрабатывалось на систему Android, то выбранной платформой для реализации была Android Studio. Данная платформа является наиболее рентабельной на данный момент времени. [10] Данная платформа сочетает удобный пользовательский интерфейс, в ней присутствует поддержка необходимых языков программирования, а также, можно отметить, что эта среда разработки постоянно развивается и совершенствуется благодаря компании Google. [11] Но гораздо важнее то, что данная платформа является крайне надежной. [12] Разработанное мобильное приложение поддерживает Android версии от 4.4 (KitKat) отбросив 4,5% пользователей данной системы. [13] Однако, с течением времени, эта доля постоянно будет уменьшаться.

Используемый стек технологий:

- Java – выбранный язык программирования.
- Реализация архитектуры Model-View-Presenter с использованием Moxy.
- База данных BigchainDB.
- Dagger 2 для введения зависимостей.
- Ruby/Go/Php для реализации виртуальной машины блокчейн и узлов.
- Синтезированный аналог Solidity для реализации смарт-контрактов.

3.2. Структура программной реализации

Взаимодействие с блокчейном BitIndex25

В мобильном приложении Multipass.One транзакции подписываются офлайн и передаются публичному узлу для обработки. При таком подходе выполняется полное доверие данному узлу, что противоречит уровню безопасности, необходимому для реализации приложения в сфере документооборота. Поэтому данный риск максимально снижается путем

установки соединений сразу с несколькими узлами, отправляя одинаковые транзакции и запросы. Данный процесс отображен на рис. 3.1.

Рисунок 3.1 – Взаимодействие с блокчейном

Синхронизация клиентской и серверной частей с блокчейном

При некоторых операциях (например, при составлении условий смарт-контракта) понадобится взаимодействие трех сторон: клиента, сервера и блокчейна. При таком процессе клиент должен отслеживать только те события, которые напрямую относятся к его работе, а сервер может отслеживать все контракты. Процесс взаимодействия клиента, блокчейна и сервера отображен на рис. 3.2. Также возможен вариант, когда клиент отправлял бы уникальный идентификатор события сразу серверу, однако, с точки зрения криптоанализа, данный подход сильно ухудшает безопасность приложения.

Реагировать на события приложение начинает после получения достаточного числа подтверждений (8 блоков). Чтобы действовать на событие из блокчейна необходимо некоторое количество времени в зависимости от работы участников сети. Поэтому с клиентской точки зрение было принято решение уведомлять пользователя о том, что операция отправлена, но не подтверждена.

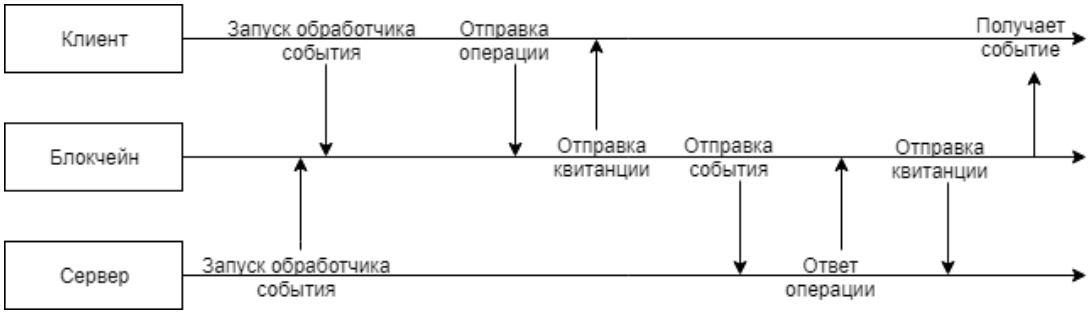

Рисунок 3.2 – Синхронизация «клиент-блокчейн-сервер»

Архитектура взаимодействия

При создании мобильного приложения Multipass.One использовался подход Model-View-Presenter (MVP) к архитектуре. MVP – шаблон проектирования пользовательского интерфейса, который был разработан для облегчения автоматического модульного тестирования и улучшения разделения ответственности в презентационной логике (отделения логики от отображения):

- Модель – хранит в себе всю бизнес-логику.
- Вид – реализует отображение данных (из Модели), обращается к Представителю за обновлениями.
- Представитель – реализует взаимодействие между Моделью и Видом.

Для более удобного восприятия архитектуры продукта показана схема взаимодействий на рис. 3.3.

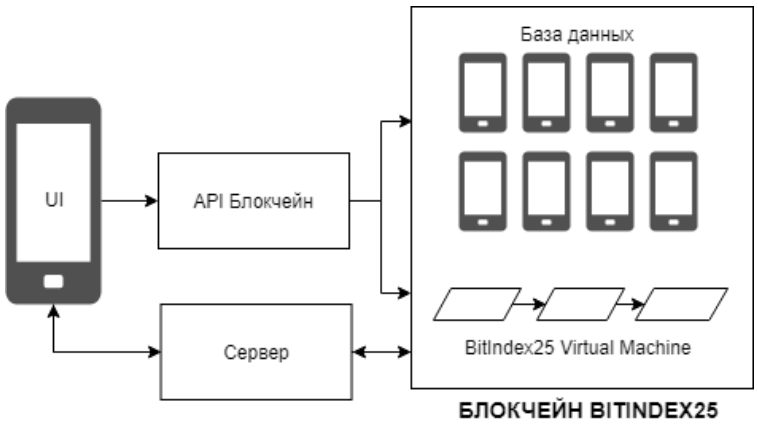

Рисунок 3.3 – Архитектура взаимодействия

Все пункты из данной схемы более подробно будут разобраны далее в работе. Однако, нужно отметить, что рассмотрены данные моменты будут не основательно, так как работа посвящена клиентской части и взаимодействию с остальными слоями.

Из схемы нужно учесть, что база данных и серверный код расположены в блокчейне. Однако, не совсем ясно назначение сервера. В классических приложениях сервер выполняет роль хранилища данных, отвечает за бизнес

логику и координацию работы. Но решение данных задач перенесено на блокчейн. Тем не менее, сервер позволяет работать напрямую с внешними сервисами (курсы криптовалют, внутреннее общение на платформе при заключении смарт-контрактов и др.). Но для подтверждения операций сервер не предназначен.

Для подтверждения транзакций и их защиты используется метод Proof-of-stake (PoS). Это метод защиты, основанный на том, что вероятность формирования участником очередного блока в блокчейне пропорциональна доле, которую составляют принадлежащие этому участнику расчётные единицы данной криптовалюты от их общего количества. Рассматриваемый блокчейн BitIndex25 имеет немного отличный метод от классического PoS метода. В выбранной реализации участники не только подтверждают транзакции выбранного блокчейна, но и участвуют в одобрении операций других монет, на основе которых создаются заявки на покупку внутренней валюты BitIndex25.

Архитектура классов Multipass.One

Для рассмотрения архитектуры классов были отображены UML-диаграммы классов для следующих процессов: получение документов, передача актива. Данные диаграммы представлены соответственно на рис. 3.4 и рис. 3.5. Получение документа происходит при запросе с фрагмента. На активности отображаются разные фрагменты в зависимости от выбранных пунктов пользователем на фрагментах выбора типа документа. Документ может быть 3 типов: физический документ, юридическое лицо, документ «Multipass». Разные типы документов делятся на различные подтипы документов. Реализовано это с помощью фабричного метода. Таким образом, появляется возможность добавлять новые типы документов без переписывания уже существующего кода.

Вторая диаграмма показывает структуру процесса «передачи актива». В нем присутствует класс смарт-контракта, который составит из следующих полей, являющимися классами: «User» - аккаунт пользователя, «Partner» -

аккаунт контрагента, «Asset» - целевой актив для передачи, «Demand» - требование. Активов и требований в одной сделки может быть множество. «Demand», как и класс документа, реализован с помощью фабричного метода. Из-за того, что пользователь может создавать самые разные условия, то для этих условий соответствующих классов будет большое количество. На диаграмме, например, отображены требования типов: срок, деньги, третье лицо, верификация документа, процент от актива.

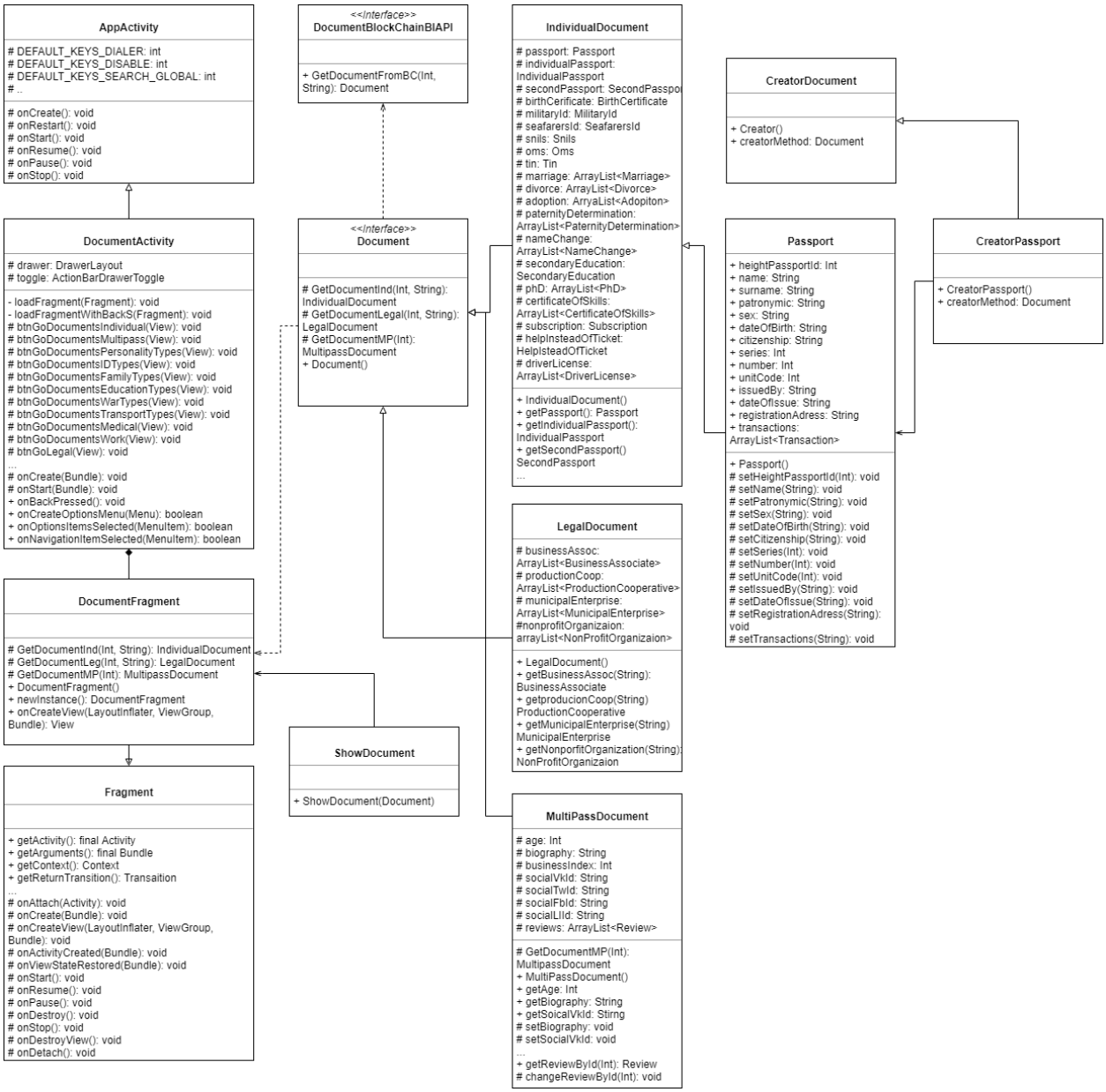

Рисунок 3.4 – UML-диаграмма классов «получения документов»

Сформированный смарт-контракт передается на сервера в блокчейне. И, когда все условия будут выполнены, у сущности «Asset» будет изменено

поле «Owner». Что, повлечёт изменения в документах этого актива с последующим одобрением 8 блоков сети блокчейн. Затем, при обращении пользователя или системы к данному активу, будут загружены уже обновленные данные.

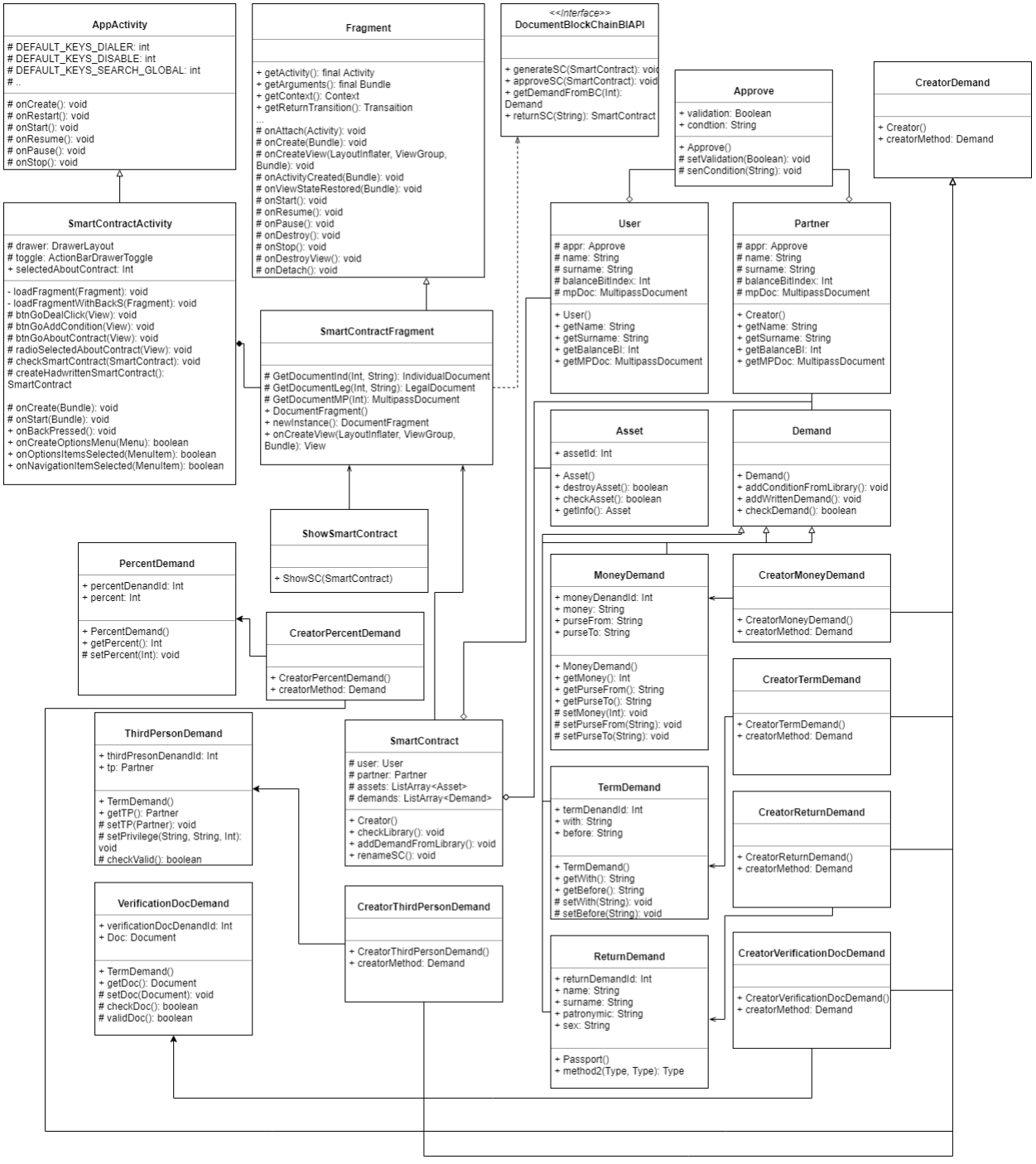

Рисунок 3.5 – UML-диаграмма классов «передача актива»

Особо важным для безопасности является процесс регистрации. Так как именно на этом шаге создаются основные параметры, по которым

пользователь, в дальнейшем, будет пользоваться мобильным приложением. Поэтому следует рассмотреть данный процесс подробнее.

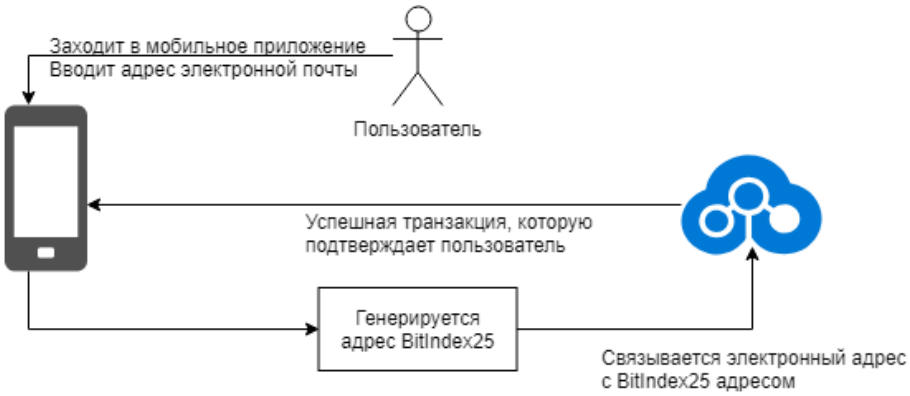

Рисунок 3.6 – Процесс регистрации

Регистрация – первый шаг, когда пользователь пытается использовать мобильное приложение (см. рис. 3.6). Для начала пользователь открывает мобильное приложение. Затем пользователь вводит свой адрес электронной почты. Новый адрес кошелька BitIndex25 автоматически генерируется мобильным приложением. Этот адрес отправляется пользователю на адрес электронной почты для удобства. Автоматически устанавливается связь между основным адресом BitIndex25 и этим дополнительным адресом. Для этого у пользователя считывается небольшое количество монет BitIndex25, так как на установку связи тратится транзакция. Этот подход затрудняет использование одноразовых учетных записей. Как только связь между адресами установлена, мобильное приложение покажет диалоговое окно подтверждения. Если пользователь подтверждает данную операцию отпечатком пальца, сопоставление установлено. Тогда же происходит привязка отпечатка пальца к аккаунту.

3.3. Модель данных

Децентрализованная база данных представляет собой фундаментальный сдвиг в эффективности крупномасштабного хранения. Устранение центрального контроля позволяет пользователям хранить и делиться данными без использования стороннего поставщика услуг

хранения. Децентрализация смягчает риск сбоев и сбоев данных при одновременном повышении безопасности и конфиденциальности хранения объекта. Это также позволяет оптимизировать данные для более дешевого хранения.

Выбранная база данных (BigchainDB) не предназначена для децентрализованного хранилища данных. Выбранная база данных лучше всего подходит для хранения, индексации и запроса структурированных данных, а не файлов.

Блок - это объект JSON с определенной схемой. Блок должен содержать следующие ключи JSON: «высота»: «<высота блока>», "транзакции": ["<Список транзакций>"]. Блок "высота" (целое число) обозначает высоту блокчейна, когда данный блок был зафиксирован. Поскольку высота блокчейна монотонно увеличивается, высоту блока можно рассматривать как его идентификатор.

Для рассмотрения модели данных были взяты две ER-диаграммы: «документы», «активы». В диаграмме для документов, отображенной на рис. 3.7, не показаны все возможные документы. Были максимально раскрыты все поля на примере следующих данных: документ типа «Паспорт», документ типа «Заграничный паспорт». Для юридических лиц был показан пример для организации с ограниченной ответственностью. Все остальные документы и типы юридических лиц созданы по аналогии показанных документов на диаграмме. Также были показаны поля документа типа «Multipass». В диаграмме отображены только те поля, которые относятся к диаграмме документов. Так, например, из объекта «User» были убраны поля, связанные с активами. Сделано это для более наглядного отображения данных. В ER-диаграмме для документов отображены следующие таблицы и связи. Объект «User» имеет обязательные связи с объектами «IndividualDocument», «LegalDocument», «MultipassDocument». В «IndividualDocument» имеются связи с всеми документы физических лиц. В «LegalDocument» имеются связи с юридическими лицами, аффилированными с пользователем, а также

показатели данных юридических лиц. В «MultipassDocument» находится общая информация о пользователе, а также отзывы о сделках с ним.

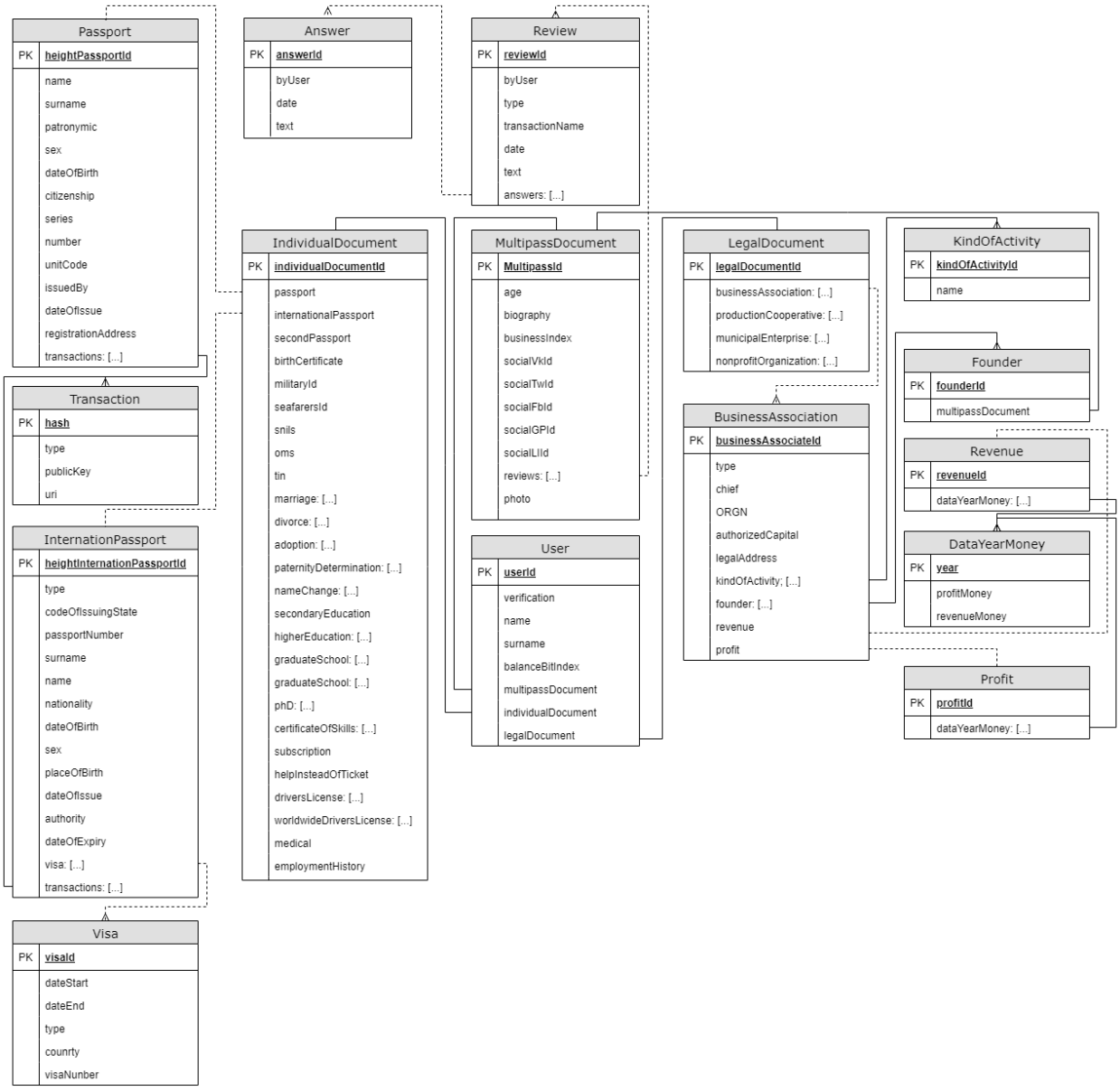

Рисунок 3.7 – ER-диаграмма базы данных документов на примере паспорта

ER-диаграмма, представленная на рис. 3.8, является моделью активов. Пользователь имеет обязательную связь с объектами «FinancialAsset», «ImmaterialAsset», «RealAsset». В диаграмме отображен только актив типа «Недвижимость». У недвижимости есть владельцы текущие и предыдущие, а также различные документы. Также могут быть долги различных типов. Остальные типы активов созданы по аналогии с недвижимостью. В данный момент для вещественных активов существуют следующие типы:

недвижимость, транспорт. Для невещественных: патент, торговый знак, гудвилл. Для финансовых: доля в бизнесе, акция, накопления, страховка.

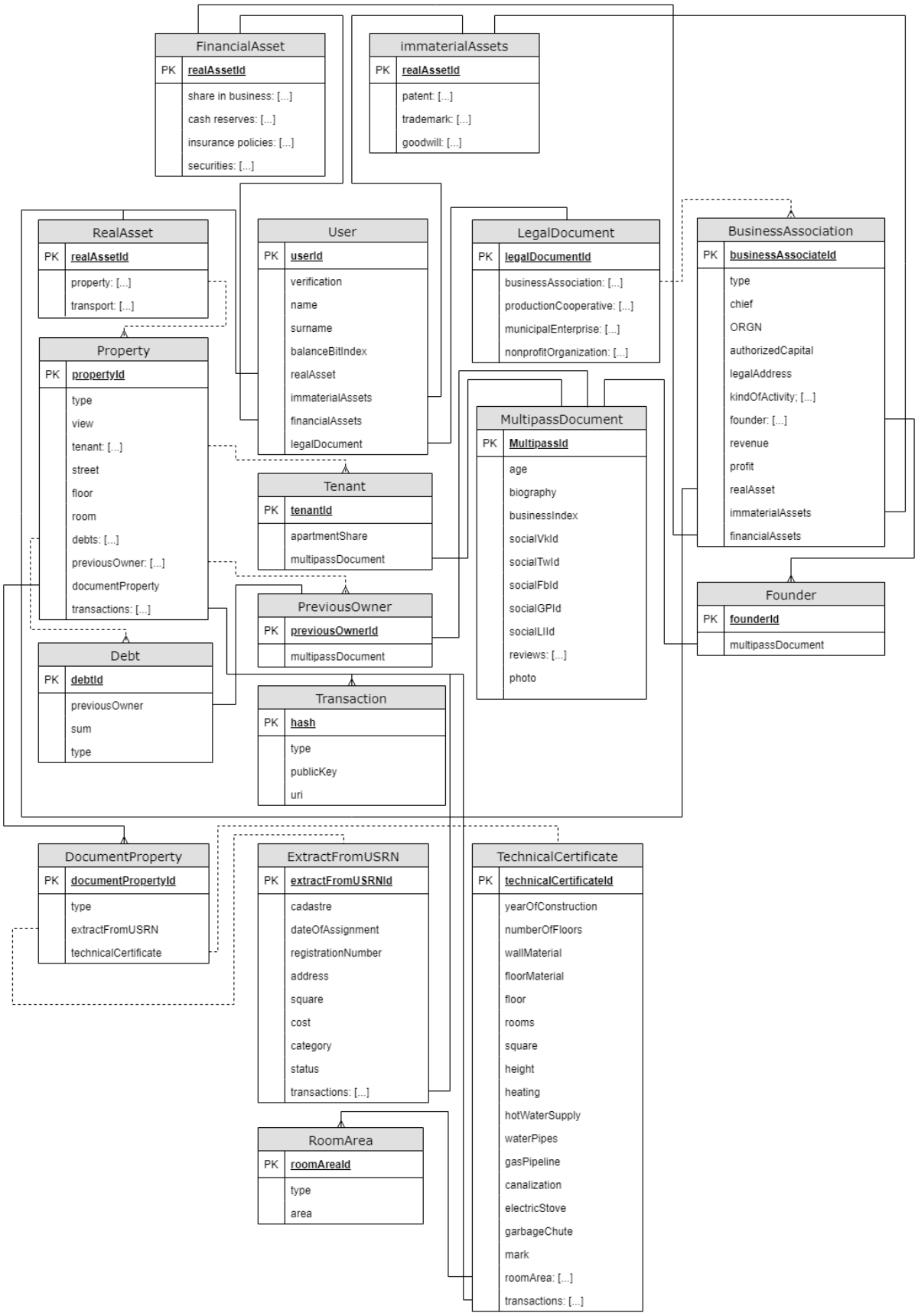

Рисунок 3.8 – ER-диаграмма базы данных активов на примере квартиры

Для рассмотрения одного из объектов был выбран документ «Паспорт». Помимо списка транзакций и высоты, такой документ должен хранить и изменяемые данные, которые применяются только после того, как будет достигнуто достаточное количество подтверждений участников (поля: фамилия, имя, отчество, пол, дата рождения, гражданство, серия, номер, код подразделения, кем выдан, когда, адрес регистрации, семейное положение). Средний объем такого документа составляет 286 байт. Важно понимать, что в данной системе данные хранятся не на собственном мобильном устройстве, а на множестве в виде распределенных хэшей. Пример запроса приведен в приложении Б.

3.4. Методы безопасности

Важной частью разработки мобильных приложений, взаимодействующих с критичными ресурсами, является этап анализа информационной защищенности. При разработке мобильного приложения существуют риски частичной либо полной компрометации системы и конфиденциальных данных. Важно отметить: так как в рамках данной работы разрабатывалась только клиентская часть мобильного приложения, то аспект безопасности распределенной сети блокчейн необходимо опустить.

Все данные, с которыми работает клиентская часть, получены с сети блокчейн и не хранятся на устройстве. Отсюда, важным моментом для рассмотрения безопасности является выбор метода защиты физического доступа к данным и одобрение операции. Для одобрения совершения транзакции или доступа к информации были выбраны два метода:

- Отпечаток пальца.
- PIN-код. (Применяется, в случае отсутствия аппаратной возможности применения первого метода).

Данные два способа были выбраны вовсе не случайно. Оба этих способа удобны и требуют малого количества времени для применения, а

также, каждый из них выделяется уровнем безопасности относительно конкурирующих способов. [14]

Необходимо рассмотреть реализацию работы алгоритма одобрения операции с помощью отпечатка пальца, используя диаграмму активности. Данная диаграмма приведена на рис. 3.9.

В данном алгоритме важным является то, что полученный публичный ключ при сканировании отпечатка из-за своей уникальности может являться паролем к любой операции. Данный пароль хранится в сети блокчейн и является криптозащищенным.

Также, при разработке были применены методы устранения уязвимостей мобильных приложений. Одним из важных этапов осуществления безопасности в приложении Multipass.One является использование защищенного протокола передачи информации https для осуществления межсетевых запросов. Помимо протокола было уделено внимание настройке защищенности соединения, например были убраны алгоритмы сжатия. [15]

Помимо защищенности каналов важно осуществить защищенность самих данных при обмене информацией. Для создания шифрования данных использован 256-битовый симметричный алгоритм AES [16], 2048-битовый RSA [17] алгоритм, и безопасный обмен ключами на основе протокола Диффи-Хеллмана [18].

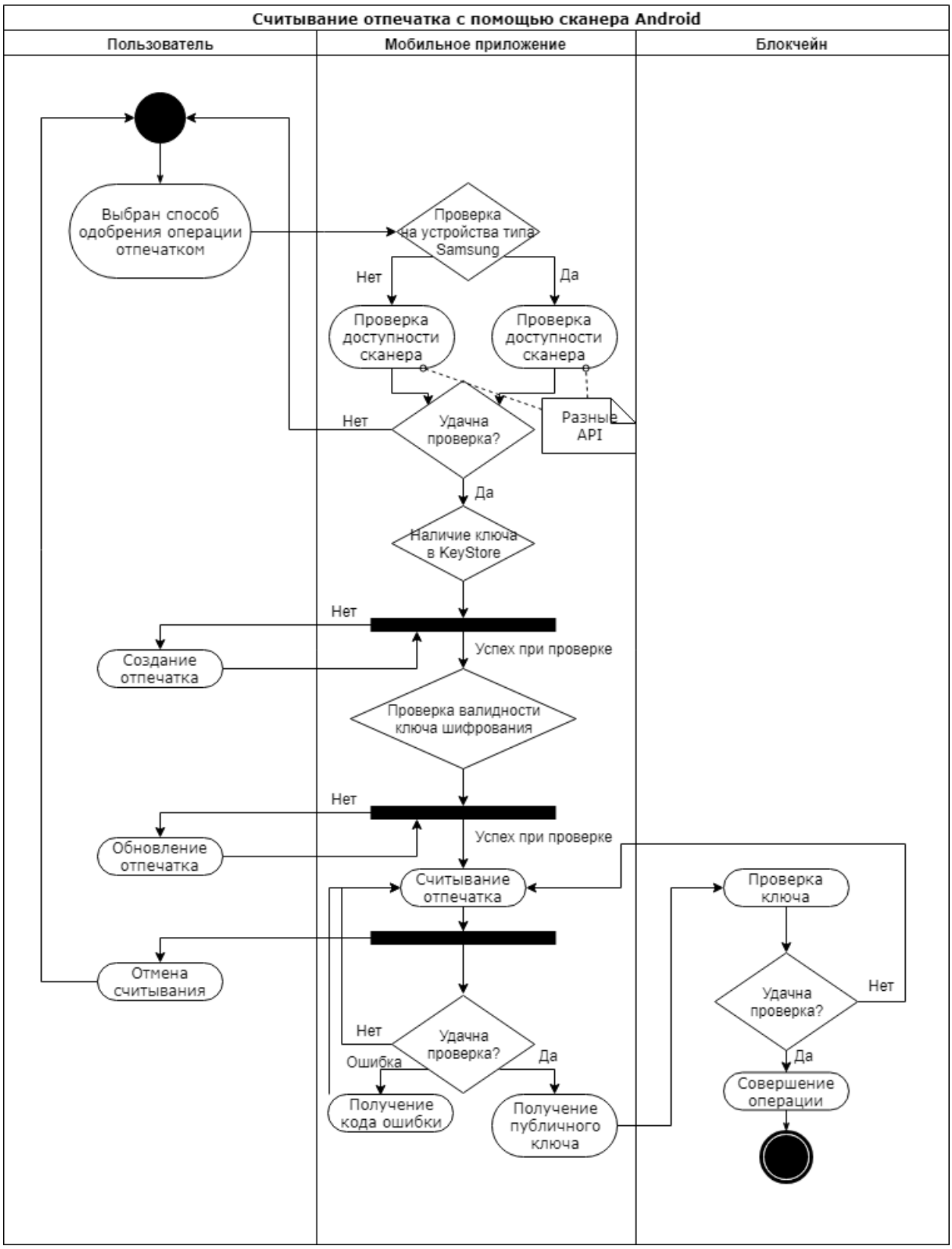

Рисунок 3.9 – Диаграмма активности сканера отпечатка

4. ИССЛЕДОВАНИЯ СВОЙСТВ РЕШЕНИЯ

4.1. Время поиска объекта в блокчейне

Отсюда, по мере роста количества сегментов в сети, становится все более трудно найти данный файл без предварительного знания мест его частей. Вероятность найти целевой файл (4.1), состоящий из k шардов при обработке n случайных расположений в сети в 1 миллисекунду, которая состоит из N шардов всего, моделируется как гипергеометрическое распределение с $K = k$ (см. табл. 4.1). Диаграмма распределения отображена на рис. 4.1. Также, в ходе построения данной модели было учтено, что сложность поиска в блокчейне BitIndex25, в зависимости от количества шардов, равна $O(s \cdot \log(s) + t)$, где s – количество узлов, t – количество связей. Время поиска определяется по формуле (4.2).

$$P_{succ}(N,k,n) = \frac{\binom{N-k}{n-k}}{\binom{N}{n}}. \quad (4.1)$$

$$T_{поиск} = P_{succ}(N,1,n) \cdot 10^{-3} \cdot 8. \quad (4.2)$$

На хранение всех файлов, изначально, отводится 10 мегабайт на диске пользователя, однако, для более щедрых денежных вознаграждений за хранение файлов и участие в одобрении транзакций пользователь может отдать под хранение файлов больше дискового пространства. Отсюда, рост объема базы данных при добавлении новых данных производится незначительно, так как хранятся данные на других устройствах и серверах.

Таблица 4.1 – Расчет вероятности успеха и времени поиска

N	k	n	P_{succ}	$T_{поиск}, сек$
100	8	64	0,0237	0,0125
1000	8	63,996	1,8346E-10	0,1250

Окончание таблицы 4.1

10000	8	63,95	1,7778E-18	1,2509
100000	8	63,4	1,6703E-26	12,6183
1000000	8	57	6,6627E-35	140,3509

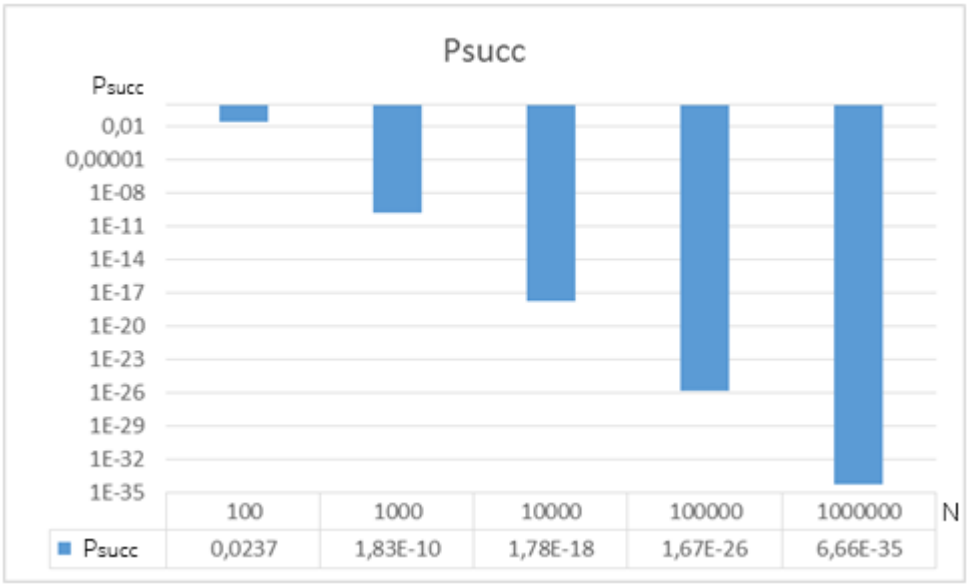

Рисунок 4.1 – Распределение вероятности нахождения файла в блокчейне

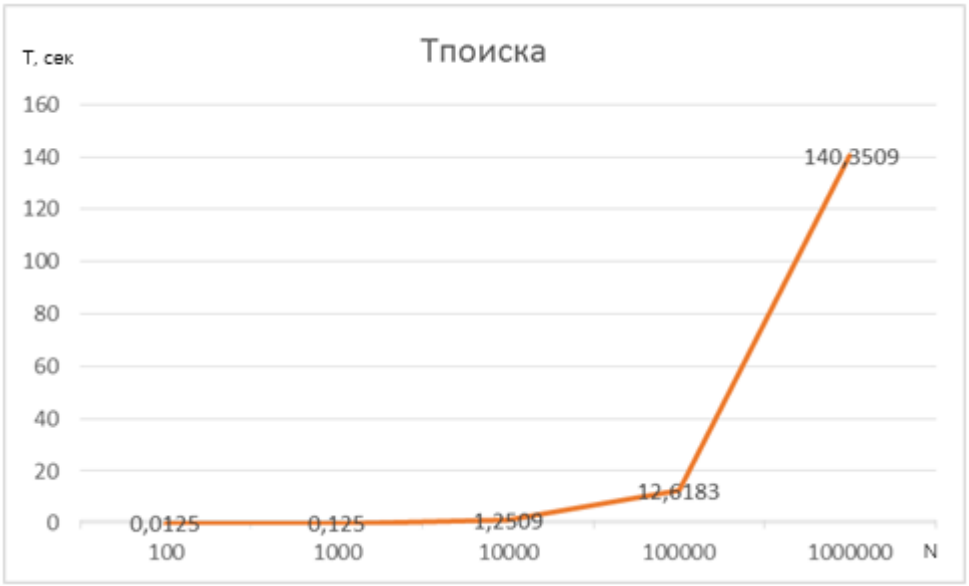

Рисунок 4.2 – Скорость нахождение целевого файла в блокчейне

Полученная модель на рис. 4.2 описывает максимально возможное время поиска файла в сети блокчейн. Для проверки полученных данных,

были получены замеры времени со 100 различных устройств при разных параметрах сети блокчейн. Результат представлен в табл. 4.2.

Таблица 4.2 – Сравнение полученной модели в теории и на практике

$T_{теор}, сек$	$T_{практ}, сек$	Разница, %
0,0125	0,0091	27.2
0,1250	0,0845	32.4
1,2509	0,7856	37.2
12,6183	8,9968	28.7
140,3509	101,6141	27.6

Так как полученный в теоритических расчетах результат, является максимально возможным, то на практике, процедура поиска заканчивается раньше приблизительно на 30%. Данная зависимость получена из-за того, что целевые шарды, как правило, не находятся в самым крайних узлах сети блокчейн.

4.2. Защита приложения от атак

Так как самой важной частью такого приложения является безопасность данных пользователя, то было проведено исследование на предмет защищенности мобильного приложения. Для этого использовались следующие методы анализа [19]:

- Анализ исходного кода.
- Декомпиляция.
- Анализ метаданных.
- Анализ работы приложения с файловой системой.
- Анализ работы приложения с инструментами, предоставляемыми операционной системой Android.

Данные методы были использованы в ходе проведения следующих типов атак: «вредоносная программа», «атака посередине», «атака на NFC».

В ходе данного исследования был исключен такой внешний фактор, как наличие внешних вредоносных программ.

При атаке типа «вредоносная программа» на устройство было загружено вредоносное программное обеспечение. Так как данные хранятся в защищенном хранилище Android, то без root-прав получить данные оказалось невозможным. Однако, при наличии этих прав, забрать целевые файлы, хранимые на устройстве, с помощью механизма декомпиляции оказалось возможным. Однако, данный процесс был максимально затруднен из-за применения механизма ProGuard. Данный инструмент оптимизирует байт-код, и производит переименование классов, методов, переменных. Тем самым, усложняя чтение кода при декомпиляции. Также, так как данные хранятся на устройстве в зашифрованном гибридном алгоритме AES и RSA, то расшифровать эти данные в ходе исследования не получилось. Также, важно учесть, что полученные данные, являются лишь частью необходимой информации из 8 необходимых блоков.

В ходе проведения атаки «человек посередине» было проделано следующее: в рамках одной сети Wi-Fi к подключению между клиентом и блокчейном была произведена попытка подменить хост. Это очень опасная атака, при которой злоумышленник может перехватить все части данных целевого запроса, приходящих из сети блокчейн. Однако, данная атака не прошла с успехом. Связано это с тем, что используется 2 метода защиты от данной атаки. Первый метод - это использование механизма SSL-рукопожатия, в рамках которого клиент и сервер в сети блокчейн обмениваются ключами и сертификатами. В ходе данного «рукопожатия» происходит проверка сертификата на его подпись, на соответствие имени и домена сертификата, а также является ли сертификат истекшим или отозванным. Второй метод – это использование цепочки сертификатов. Начиная с версии Android 4.0, все сертификаты хранятся в разных файлах. Отсюда, предоставляется возможность для нахождения на устройстве множества корневых сертификатов. Проверка сертификатов происходит по

цепочке. Сначала проверяется присланный на устройство сертификат на имя хоста, сроки и другие параметры, а затем проверяются корневые сертификаты.

В «атаке на NFC», используя технологию NFC, можно украсть деньги, либо информацию. В ходе данной атаки злоумышленнику необходимо будет использовать только интерфейс платы. То есть, отсутствует необходимость взламывать само устройство. Для проведения атаки было сымитировано то, что необходимый интерфейс украден. На самом деле, с помощью исходного устройства была произведена попытка считать документ типа «паспорт» другим человеком. Данная атака прошла неудачно, так как пройти защиту отпечатком пальца не удалось.

4.3. Исследование объема APK файла

Размер файла .APK [20] мобильного приложения крайне важен. Если размер мобильного приложения слишком велик, его загрузка может быть сложной и дорогой для пользователей. Первое впечатление, которое могло бы произвести приложение, может быть испорчено, если пользователь не смог запустить продукт из-за размера, так как на диске устройства пользователя может быть очень мало места. Также пользователи часто избегают загрузки приложений, которые кажутся слишком большими, особенно на развивающихся рынках, где устройства подключаются к сетям 2G и 3G или работают по планам с оплатой за каждый байт.

Исходя из исследования [21], которое провел Бошель, можно сделать вывод, что за последний месяц средний объем .APK приложений в Android Play составил 15 мегабайт. Объем .APK разработанного мобильного приложения Multipass.One составляет 32 мегабайта. Данный результат вдвое больше среднего объема по рынку Android приложений. Однако, это не является отрицательным результатом со всех точек зрения. Если рассматривать это же исследование, то можно увидеть, что мобильные приложения для Android с большим количеством загрузок, как правило,

также имеют большой размер файла. Самые популярные мобильные приложения, как правило, многофункциональны. А это включает дополнительный код и ресурсы, которые будут увеличивать размеры файлов. Разработанное приложение Multipass.One с размером .APK файла попадает в ту выборку приложений, которую скачивают более 10 млн раз. Что, несомненно, является позитивным результатом.

Однако, несмотря на такой такой результат, существует исследование [22], которое доказывает, что при уменьшении размера .APK файла на 6 мегабайт конверсия установок увеличивается на 1%. Отсюда, можно сделать вывод, что следует по возможности уменьшать размер .APK файла разработанного мобильного приложения Multipass.One, применив следующие методы:

- Удаление неиспользуемых ресурсов.
- Минимизация использования ресурсов из библиотек.
- Сжатие файлов PNG и JPEG.
- Использование векторной графики.
- Удаление ненужного сгенерированного кода.

Применение этих методов способно уменьшить размер .APK файла на 44%. [23] Это особенно важно для приложения, которое будет работать с разнообразной целевой аудиторией в области документооборота.

4.4. Интерфейс

При разработке клиентской части мобильного приложения огромная роль отводится разработке интерфейса. Также «удобство использования», о котором речь шла ранее, очень важно для принятия инновационного способа документооборота. Проблема разработки интерфейсов мобильного приложения заключается в том, что практически любые изменения на поздних этапах приводят к большим временным и денежным потерям. Чтобы этого избежать необходимо уделить внимание к таким этапам разработки

клиентской части, как прототипирование и написание сценариев использования.

Также, мобильное приложение должно смотреться гармонично на любом устройстве, то есть быть «гибким». [24] Для создания такого интерфейса во время разработки необходимо применять метод модульного программирования. Суть такого метода заключается в том, чтобы значимые пункты дизайна разбивались на отдельные самостоятельные части. При таком подходе, необходимо будет менять лишь малую часть кода, в случае необходимых изменений. Важно обращать внимание, при разработке такой системы ведения документооборота, на то, какие пользователи будут пользоваться системой. Например, 49% пользователей смартфонов держат устройство одной рукой и управляются большим пальцем. [25] Если учитывать эти данные, то необходимо правильно рассчитывать площадь и пропорции в экране.

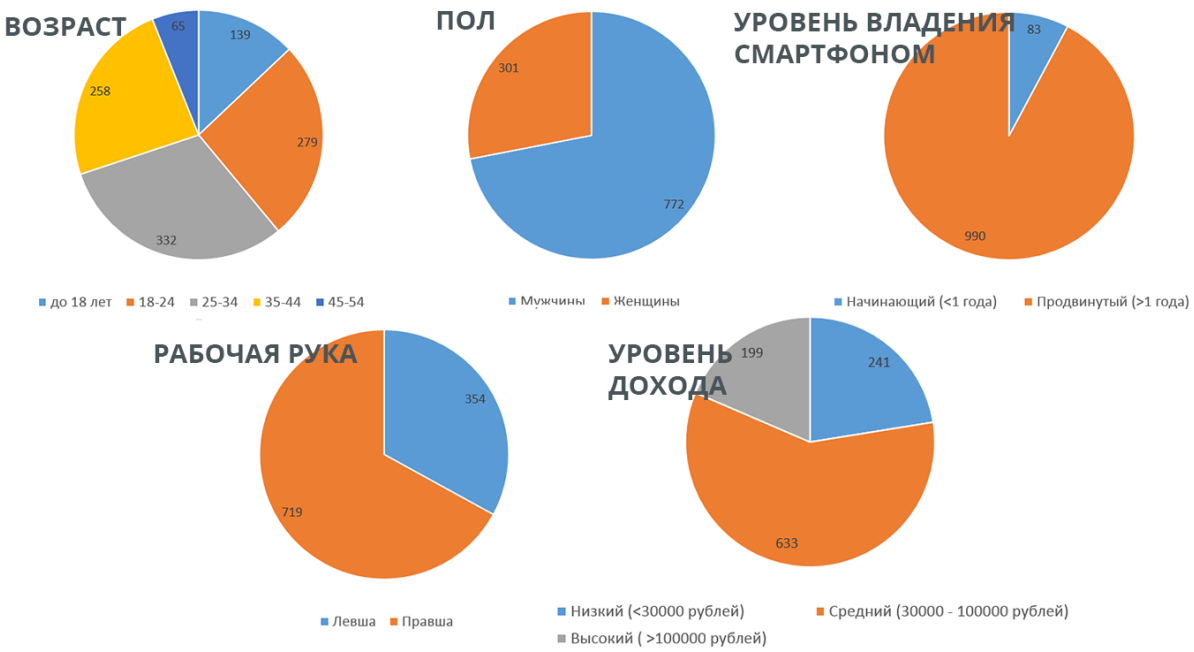

Рисунок 4.3 – Портрет аудитории исследования

Помимо данных методов, важно учитывать прямое мнение пользователей. Тестирование на пользователях позволяет качественно улучшить те детали, которые кажутся неважными при разработке. Например, после разработки интерфейса Multipass.One было проведено исследование, в

котором приняло участие 1073 респондента. Портрет аудитории представлен на рис. 4.3. Параметры, выбранные для демонстрации портрета аудитории следующие: возраст, пол, уровень владения смартфоном, рабочая рука, уровень дохода. Первые два параметра (возраст и пол) влияют на культурное и психологическое восприятие мира. Параметры «уровень владения смартфоном» и «рабочая рука» важны с технической точки зрения. Благодаря разнообразности аудитории в этих параметрах можно понять насколько удобен и понятен разработанный интерфейс. Параметр «уровень дохода» интересен тем, что благодаря градации по нему, есть возможность определить насколько разработанное приложение удобно для людей, которые взаимодействуют с документами несколько реже остальных и уровень компетенций ниже.

Из портрета аудитории, который представлен на рис. 4.3, можно сделать вывод о том, что аудитория была задействована самая разнообразная с различными культурными ценностями и психологическими аспектами. Это было важно сделать, так как разрабатываемое мобильное приложение Multipass.One создано, чтобы взаимодействовать с документами и проводить сделки. А этими процессами занимаются практически все люди вне зависимости от их предпочтений. Однако, для дальнейшего развития, все равно необходимо будет выделить основные группы целевой аудитории, так как удовлетворять каждую группу пользователей персонально нерационально.

Респондентам были предложены макеты в разных цветовых гаммах. Из них 63,65% опрошенных выбрали основными цветами красный и белый, что изменило изначально выбранную цветовую гамму.

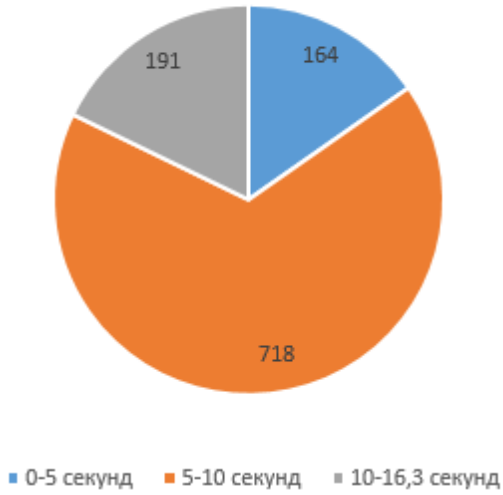

Рисунок 4.4 – Временные интервалы, которые затратили респонденты на поиск документа «Свидетельство о рождении»

Учитывая представленные методы и данные, с помощью платформы Adobe XD были разработаны макеты экранов и состояний клиентской части мобильного приложения Multipass.One.

В упомянутом ранее исследовании участникам также необходимо было в кратчайшие сроки найти документ типа «Свидетельство о рождении». Результат представлен на рис. 4.4. Упрощенный сценарий использования подразумевал следующее:

1. Пользователь находится на странице «Документы», в котором есть возможность для выбора трех вариантов типов документов:
   - Multipass.One (международный документ платформы).
   - Физическое лицо (документооборот физического лица).
   - Юридическое лицо (аффилированные с пользователем юридические лица).

Пользователь выбирает «Физическое лицо» и переходит на страницу типов документов физического лица.

2. Пользователь должен выбрать один из предложенных типов (личность, идентификационные номера, гражданское состояние, образование, воинская обязанность, транспорт, медицина, трудовая книжка).

Пользователь выбирает тип «Личность» и переходит на страницу подтипов документов физического лица типа «Личность».

3. Пользователь должен выбрать один из предложенных подтипов (паспорт, второе гражданство, заграничный паспорт, свидетельство о рождении, удостоверение военнослужащего, паспорт моряка). Пользователь выбирает подтип «Свидетельство о рождении» и переходит на выбранный документ.

Как видно из полученных результатов, представленных на рис. 4.4, подавляющее большинство респондентов справилось с заданным сценарием тестирования за 5-10 секунд. Важно отметить, что среднее время загрузки каждого из экранов 1,2 секунды. Отсюда, 2,4 секунды пользователи тратили на ожидание загрузки двух экранов. Значит, на ознакомление с информацией на экране пользователи, в среднем, тратили от 2,6 секунд до 7,6 секунд. Важно заметить, что замер времени производился автоматически с помощью функции, которая начинала отсчет при нажатии на тип документа «Физическое лицо» и заканчивала при нажатии на документ «Свидетельство о рождении». То есть, не учитывалось время, которое было затрачено на ознакомление с первым экраном. Модели устройств, на которых проводилось исследование представлены в табл. 4.3.

Следует отметить, что скрипт, написанный специально для того, чтобы протестировать скорость работы, исключая человеческий фактор, справился с задачей за 2,9 секунд. А человек, который знал где находится каждый из элементов, затратил 3,8 секунды. То есть, 1,4 секунды времени ушло на нажатия нужных элементов на двух экранах. Отсюда, разница между средним затраченным временем на заданный сценарий использования и временем работы человека, знакомым с системой, оказалась равной от 1,2 секунд до 6,2 секунд.

Данное исследование показывает, что для обычного пользователя не составит труда найти нужный документ без какой-либо подготовки. Процесс получения документа представлен на рис. 4.5. Мобильное приложение

соответствует поставленным ранее условиям об получении любого документа в три «клика».

Таблица 4.3 – Модели устройств, процессор, а также процент использования данных устройств в исследовании

Устройство	Процессор	Процент использования
Samsung S9	Exynos 9810, 4×2,8 ГГц и 4×1,7 ГГц	50
Xiaomi A1	Qualcomm Snapdragon 625, 2000 МГц, 8-ми ядерный	30
OnePlus 6	Qualcomm Snapdragon 845, 2800МГц, 8-ми ядерный	20

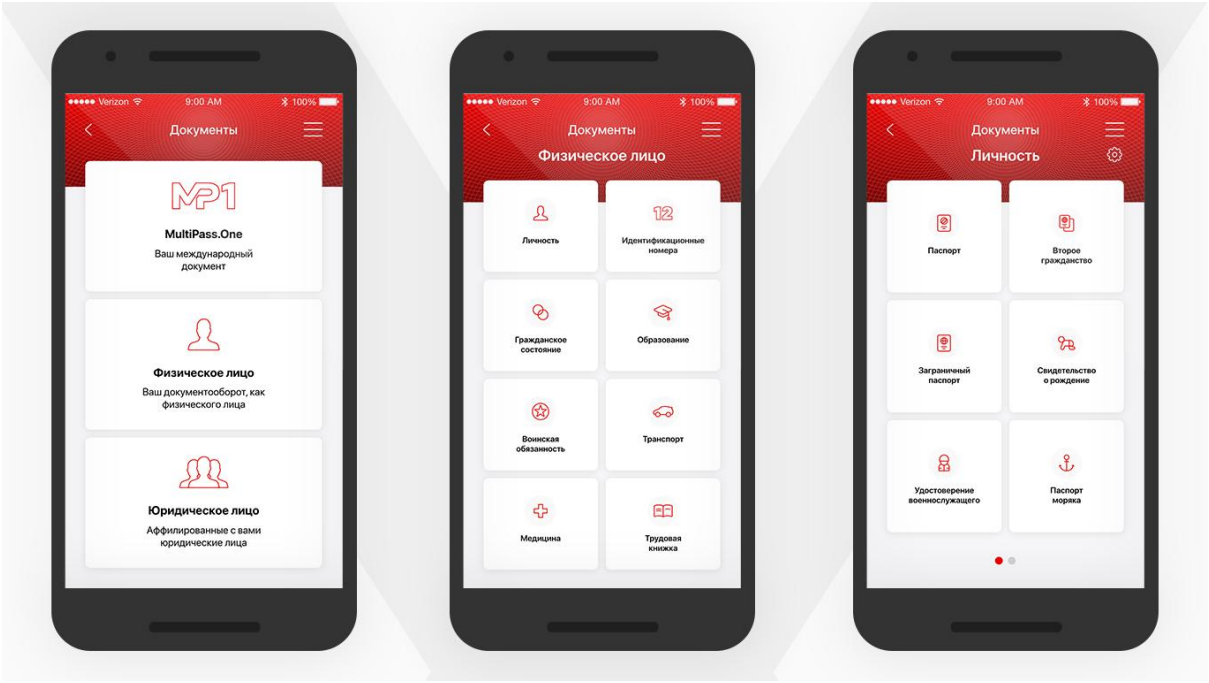

Рисунок 4.5 – Демонстрация получения документа в мобильном приложении Multipass.One

Для обычного пользователя также не составит труда заключить контракт на стандартный вид услуг. Также пользователю не нужно обладать какими-либо навыками в юриспруденции, так как приложение

самостоятельно проанализирует, каких документов недостаточно для заключения сделки, а те документы, что нужны и загружены в систему, уже проверены валидаторами. Также, в систему включены заранее подготовленные варианты договоров и стандартных условий, о которых принято договариваться людям при классических сделках. Процесс заключения контракта представлен на рис. 4.6.

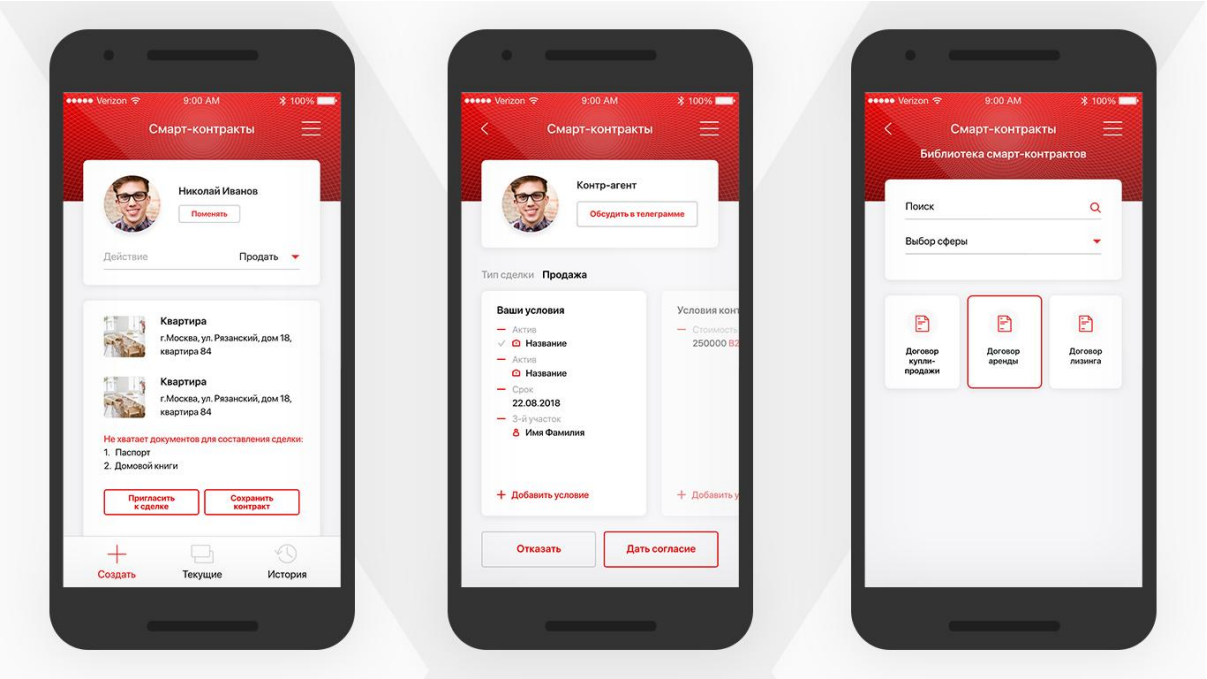

Рисунок 4.6 – Демонстрация работы с активами в мобильном приложении Multipass.One

4.5. Тестирование интерфейса

В качестве тестирования интерфейсов был выбран метод ручного тестирования. Для проведения такого тестирования необходимо было составить возможные варианты поведения пользователя. Данные истории поведения представлены в приложении В.

В ходе тестирования все истории поведения пользователя были проверены. Ошибок выявлено не было.

На основе раздела «Исследование свойств решения» был сделан вывод в разделе «Заключение».

5. ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ

5.1. План-график выполнения работ

Выпускная квалификационная работа посвящена разработке клиентской части мобильного приложения для организации работы с документами физических и юридических лиц на основе технологии блокчейн. При разработке использовался итеративный способ. Отсюда, будет рассмотрена только текущая итерация. В ходе выполнения работы было разработано мобильное приложение. В работе применялись технологии, которые были обозначены в разделе «Стек используемых технологий».

На основе полученных результатов работы, были выделены следующие этапы разработки:

1. Анализ предметной области и поиск информации.
2. Составление технического задания.
3. Проектирование архитектуры программного продукта.
4. Разработка клиентской части мобильного приложения.
5. Тестирование и проведение атак на разработанный продукт.
6. Разработка документации.

На этапе «Анализ предметной области и поиск информации» производились следующие работы: получение знаний о текущем состоянии рынка; анализ конкурентных продуктов; изучение необходимых технологий; а также выбор вектора разработки.

На этапе «Составление технического задания» производились следующие работы: формализация задачи; выбор технологий; структурирование функциональных требований; отбор ограничений на продукт.

На этапе «Проектирование архитектуры программного продукта» производились следующие работы: составление структуры базы данных; выбор решений в архитектурных вопросах продукта; создание бизнес-

процессов; выбор методов обработки полученных данных; выбор алгоритмов шифрования и защиты.

На этапе «Разработка клиентской части мобильного приложения» производились следующие работы: создание внешнего вида разрабатываемого продукта; написание кода для обработки элементов интерфейса; создание модулей для работы с полученными данными; написание кода для защиты мобильного приложения от атак.

На этапе «Тестирование и проведение атак на разработанный продукт» производились следующие работы: тестирование полученного продукта; устранение неисправностей.

На этапе «Разработка документации» производились следующие работы: создание документации для отдельных модулей мобильного приложения; создание документации для работы с продуктом.

На основе этих данных необходимо составить табл. 5.1. Всего на выполнение работы было потрачено 58 дней.

Таблица 5.1 – Длительность этапа разработки

Название задачи	Длительность
Анализ предметной области и поиск информации	2 дня
Составление технического задания	2 дня
Проектирование архитектуры программного продукта	9 дней
Разработка клиентской части мобильного приложения	21 день
Тестирование и проведение атак на разработанный продукт	4 дня

Окончание таблицы 5.1

Разработка документации	18 дней

Далее была составлена табл. 5.2, которая показывает трудоемкость выполняемых работ и ставки соответствующих исполнителей. Для расчета были взяты данные сайта trud.com. С данного сайта были получены следующие средние значения заработные платы:

1. Для инженера в Санкт-Петербурге – 37000 рублей.
2. Для научного руководителя в Санкт-Петербурге – 44000 рублей.

Таблица 5.2 – Расчет трудоемкости и продолжительности работ

№	Название задачи	Исполнитель	Объем работ, в днях	Затраты на оплату труда, руб.
1	Анализ предметной области и поиск информации	Руководитель	1	2095
		Разработчик	2	3524
2	Составление технического задания	Руководитель	1	2095
		Разработчик	2	3524
3	Проектирование архитектуры программного продукта	Руководитель	2	4190
		Разработчик	9	15857

Окончание таблицы 5.2

4	Разработка клиентской части мобильного приложения	Руководитель	2	4190
		Разработчик	21	37000
5	Тестирование и проведение атак на разработанный продукт	Руководитель	1	2095
		Разработчик	4	7048
6	Разработка документации	Руководитель	3	6286
		Разработчик	18	31714
Итого		Руководитель	10	20951
		Разработчик	58	98667

В полученных данных использовался расчет дневной ставки исполнителей. Так, для инженера данная ставка составляет:

$$C_1 = \frac{37000 \text{ руб}}{21 \text{ день}} = 1762 \text{ руб/день}.$$

Для научного руководителя дневная ставка составляет:

$$C_2 = \frac{44000 \text{ руб}}{21 \text{ день}} = 2095 \text{ руб/день}.$$

Из полученных данных, был сделан вывод, что на основную оплату работы научного руководителя и разработчика, в общем, необходимо затратить 119618 рублей.

Также, были вычислены расходы на дополнительную заработную плату по формуле (5.1):

$$З_{доп.з/пл} = З_{осн.з/пл} \cdot \frac{Н_{доп}}{100}. \quad (5.1)$$

где $З_{доп.з/пл}$ – расходы на дополнительную заработную плату (руб.);

$З_{осн.з/пл}$ – расходы на основную заработную плату (руб.);

$Н_{доп}$ – норматив дополнительной заработной платы (%), в рамках данной работы принимается равным 14%.

Данный расчет представлен в табл. 5.3.

Таблица 5.3 – Расчет совокупной заработной платы

Исполнитель	Основная заработная плата, руб.	Дополнительная заработная плата, руб.	Совокупная заработная плата, руб.
Руководитель	20951	2933	23884
Разработчик	98667	13813	112480
Итого	119618	16746	136364

5.2. Расчет социальных отчислений

Производятся отчисления на страховые взносы на обязательные медицинское, социальное, пенсионное страхование по формуле (5.2):

$$З_{соц} = (З_{осн.з/пл} + З_{доп.з/пл}) \cdot \frac{Н_{соц}}{100}. \quad (5.2)$$

где $З_{соц}$ – отчисления на социальные нужды с заработной платы (руб.);

$З_{осн.з/пл}$ – расходы на основную заработную плату (руб.);

$З_{доп.з/пл}$ – расходы на дополнительную заработную плату (руб.);

$Н_{соц}$ – норматив страховых отчислений (%), в рамках данной работы принимается равным 30%.

Отсюда, по формуле (5.2) были вычислены отчисления на страховые взносы:

$$З_{соц} = (119618 \text{ руб } + 16746 \text{ руб}) \cdot \frac{30}{100} = 40909 \text{ руб}.$$

5.3. Расчет материальных затрат

К данным затратам необходимо отнести расходы на комплектующие, материалы, полуфабрикаты. Данные продукты расходуются во время реализации проекта. Также, сюда включаются транспортно-заготовительные расходы, которые в рамках данной работы равны 10%. Расчет материальных затрат отображен в табл. 5.4.

Таблица 5.4 – Материальные затраты

№	Наименование материала	Единицы измерения	Количество	Цена за единицу, руб.	Стоимость, руб.
1	Бумага А4	пачка	1	295	295
2	Картридж для принтера	шт	1	3800	3800
3	Презентация	шт	5	400	2000
4	SD-карта, 8Гб	шт	1	500	500

Окончание таблицы 5.4

Итого за материалы	6595
Транспортно-заготовительные расходы	660
Итого	7255

Отсюда, затраты на комплектующие материалы составляет 7255 рублей.

Работа велась на протяжении 58 дней по 8 часов в день. Структура серверов блокчейна также обеспечивалась 58 дней по 8 часов в день. Отсюда, в табл. 5.5 представлена оценка затрат на содержание и эксплуатацию оборудования.

Таблица 5.5 – Оценка затрат на содержание и эксплуатацию оборудования

Устройство	Тариф на электроэнер-гию, руб. за кВт/ч.	Потребля-емая мощность, кВт/ч.	Себестои-мость машинно-часа, руб.	Количест-во машинно - часов	Сумма, руб.
Ноутбук Asus ROG	4,32	0,23	0,99	464	461,34
Сервера для блокчейна SuperMicro SYS-1018GR-T	4,32	21	90,72	464	42094,08
Итого					42555,42

Для данного оборудования также важно учесть амортизационные отчисления. Для этого сначала были вычислены амортизационные отчисления за 1 год, а затем за время написания дипломной работы, а именно, 2 месяца. Расчет данных отчислений представлен в табл. 5.6.

Таблица 5.6 – Оценка амортизационных отчислений за время написания дипломной работы

Устройство	Стоимость, руб.	Срок полезного использования, год	Годовая амортизация, %	Отчисления за год, руб.	Время на диплом, мес.	Отчисления за диплом, руб.
Ноутбук Asus ROG	157610	3	33,33	52531	2	4377
Сервера для блокчейна SuperMicro SYS-1018GR-T	600000	3	33,33	199980	2	33330
Итого						37707

Помимо затрат на оборудование существуют затраты на сторонние услуги. Так, например, важно включить оплату за Интернет-соединение для серверов и устройств для разработки. За данную услугу оплачивалось два соединения. Каждый из них стоил по 870 рублей в месяц. После вычета НДС в 20% сумма составила 725 рублей в месяц. Для расчета затрат, необходимо ввести формулу (5.3):

$$З_{инт} = Т_{инт} \cdot \frac{К_{дн}}{30}. \quad (5.3)$$

где $З_{инт}$ – затраты на интернет (руб.);

$T_{инт}$ – тарифный план (руб.);

$K_{дней}$ – количество дней работы с интернетом (руб.).

Отсюда, были посчитаны затраты за время написания дипломной работы:

$$З_{инт} = 1450 \text{ руб} \cdot \frac{58}{30} = 2803 \text{ руб}.$$

Значит, итоговые затраты на содержание и эксплуатацию равны:

$$З_{исэ} = 42555,42 \text{ руб} + 2803 \text{ руб} = 45358,42 \text{ руб}.$$

5.4. Расчет накладных расходов

Накладные расходы являются дополнительными затратами, которые рассчитываются по следующей формуле (5.4):

$$З_{накл} = (З_{осн.з/пл} + З_{доп.з/пл}) \cdot \frac{Н_{накл}}{100}. \quad (5.4)$$

где $З_{накл}$ – накладные расходы (руб.);

$З_{осн.з/пл}$ – расходы на основную заработную плату (руб.);

$З_{доп.з/пл}$ – расходы на дополнительную заработную плату (руб.);

$Н_{накл}$ – норматив накладных расходов (%), в рамках данной работы принимается равным 40%.

Отсюда, значение накладных расходов равно:

$$З_{накл} = (119618 \text{ руб} + 16746 \text{ руб}) \cdot \frac{40}{100} = 54546 \text{ руб}.$$

5.5. Расчет полной стоимости работы

Для подсчета полной стоимости выполнения дипломной работы была составлена табл. 5.7.

Таблица 5.7 – Полная стоимость работы

Категория затрат	Сумма, руб.	Доля расходов, %
Материалы	7255	2,25
Оплата труда	136364	42,26
Социальные взносы	40909	12,68
Содержание и эксплуатация оборудования	45358	14,23
Амортизационные отчисления	37707	11,68
Накладные расходы	54546	16,90
Итого	322700	100

5.6. Вывод по разделу

В ходе выполнения данного раздела дипломного проекта была дана экономическая оценка. Данная оценка дает понять необходимые вложения в разработку с точки зрения финансов.

Полная стоимость разработки клиентской части мобильного приложения составляет 322700 рублей. Большой статьей расхода являются статьи расхода, связанные с оборудованием: содержание и эксплуатация (14,23%), амортизационные отчисления (11,68%). Связано это с дорогим оборудованием, необходимым для обеспечения бесперебойной работы сети блокчейн. Также, данное оборудование создает децентрализованную систему блокчейна, поэтому данное оборудование включает целый комплекс

одинаковых серверов. Основной частью расходов является статья оплаты труда. Она составляет 42,26% от общих затрат.

Программное обеспечение, разрабатываемое в ходе выполнения дипломной работы, создается внешней коммерческой компанией, которая получила необходимые инвестиции для разработки. При успешном окончании разработки данный продукт будет внедряться в различные сторонние компании, а также, при возможности, в государственные органы. Финансовые средства, которые были выделены на клиентскую часть, намного больше полученных результатов в рамках дипломного продукта. Связано это с тем, что продукт будет масштабироваться и улучшаться в последствии.

Отсюда, можно сделать вывод о том, что затраты на разработку данного продукта оправданы. Значит, с экономической точки зрения, разработка клиентской части мобильного приложения для организации работы с физическими и юридическими лицами на основе технологии блокчейн, полностью целесообразна.

ЗАКЛЮЧЕНИЕ

В рамках данной работы были раскрыты решения в клиентской части мобильного приложения для организации работы с документами физических и юридических лиц на основе технологии блокчейн. Также было реализовано отображение интерфейса передачи активов посредством смарт-контрактов со стороны пользователя системы.

При рассмотрении предметной области было доказано, что классические методы ведения документооборота (классический, электронный, облачный) устарели, а на смену им пришли различные решения на технологии блокчейн.

Разработанная платформа Multipass.One по названным ранее критериям показала, что является наиболее рентабельным решением в заданной области по указанным ранее критериям (стоимость операции, защищенность операции, удобство использования), так как:

- Разрабатываемое решение Multipass.One объединяет все вышеуказанные мобильные приложения в одном едином проекте.
- Решение отвечает на целый ряд проблем, связанных с верификацией, валидацией, работой с документами, активами, голосованиями и т.д.
- Данное решение включает преимущества других решений на блокчейне, а также поддерживает любые документы, любые активы, все действия в рамках одной программы.

Была описана архитектура разработанного продукта, процесс взаимодействия клиента и сервера с блокчейном, а также на примере документа “Паспорт” показано устройство базы данных. В данной работе был оценен размер клиентской части приложения, скорость роста при добавлении новых данных, дана оценка времени роста времени поиска документа с увеличением количества документов. Результат всех проведенных оценок оказался удовлетворительным.

Мобильное приложение Multipass.One находится в высокой стадии готовности (альфа-тест), но еще нигде не внедрено в использование, так как использование такого продукта подразумевает вовлечение правительства, что недоступно на данном этапе развития.

На мобильное приложение были произведены ряд атак, которые прошли неудачно. Отсюда, в рамках поставленных критериев, приложение можно назвать достаточно безопасным.

В перспективе, решение будет масштабироваться, сохраняя свое единообразие со стороны клиентской части. Так, планируется создание iOS, web, desktop версий продукта. В дальнейших исследованиях по данной теме могут быть проведены более глубокие тесты на объемы, ошибки и другие данные клиентской части мобильного приложения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Артем Генкин, Алексей Михеев. Блокчейн. Как это работает и что ждет нас завтра. // Альпина Паблишер. 2017. — 592 с. — ISBN 978-5-9614-6558-7.

2. Stone H. How Do Users Really Hold Mobile Devices. // 2013. — 4 февраля [Электронный ресурс]. URL: http://www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php (дата обращения: 04.12.2018).

3. Mayukh M. Ethereum Smart Contract Development: Build blockchain-based decentralized applications using solidity // 2018. — 288 p. — ISBN 978-1788473040.

4. Парасоцкая, Н. Н. Документооборот в учете безналичных денежных средств // Бухгалтерский учет в бюджетных и некоммерческих организациях. – 2012. – №23. – С. 7-10.

5. Редакция ForkLog. Проект DocSensus: с появлением блокчейна бумажный документооборот уйдет в прошлое // 2017. — 23 марта [Электронный ресурс]. URL: https://forklog.com/proekt-docsensus-s-poyavleniem-blokchejna-bumazhnyj-dokumentooborot-ujdet-v-proshloe/ (дата обращения: 19.11.2018).

6. Павленко Д., Бондаренко С. DocSensus Документооборот в эпоху блокчейн // 2017. — 17 ноября [Электронный ресурс]. URL: https://deloitte.com/content/dam/Deloitte/ua/Documents/technology/Docsensus%20Astana%2014.06.2017.pdf (дата обращения: 19.11.2018).

7. Пальмачинский Я. Endo WhitePaper // 2018. — 24 июля [Электронный ресурс]. URL: https://endo.im/ru/whitepaper/ (дата обращения: 19.11.2018).

8. Приказ ФМС России N 288 // 2012. — 11 сентября [Электронный ресурс]. URL:

https://normativ.kontur.ru/document?moduleId=1&documentId=217058 (дата обращения: 19.11.2018).

9. Приказ ФНС России N ЯК-7-6/488 // 2011. — 11 августа [Электронный ресурс]. URL: https://www.nalog.ru/rn77/about_fts/docs/3938344/ (дата обращения: 19.11.2018)

10. Фадеев А.Ю., Волкова Е.А. Сравнительный анализ программного обеспечения для разработки мобильных приложений // Наука и перспективы. — 2016. — №3. — 1 марта [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-programmnogo-obespecheniya-dlya-razrabotki-mobilnyh-prilozheniy (дата обращения: 10.12.2018).

11. Meet Android Studio // 2017. — 19 августа [Электронный ресурс] URL: https://developer.android.com/studio/intro/index.html (дата обращения: 10.12.2018).

12. Бурзуева Н. Н., Мостовой Я. А. Анализ надежности среды разработки Android Studio // Интернет-журнал Науковедение. — 2017. №6 (43) — 3 июня [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/analiz-nadezhnosti-sredy-razrabotki-android-studio (дата обращения: 10.12.2018).

13. Статистика распределения версий ОС Android // 2018. — 3 октября [Электронный ресурс]. URL: https://droidbug.com/statistika-raspredeleniya-versiy-os-android-za-oktyabr-2018/ (дата обращения: 03.12.2018).

14. Сидорова Марина Андреевна Защита данных мобильных устройств на базе ОС Android // Научный журнал. — 2017. — №6-2 — 1 июня [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/zaschita-dannyh-mobilnyh-ustroystv-na-baze-os-android (дата обращения: 09.01.2019).

15. Цыганенко Н. П. Статический анализ кода мобильных приложений как средство выявления его уязвимостей // Физико-математические науки и информатика. — 2015. — № 6 — 1 июня [Электронный ресурс]. URL:

http://cyberleninka.ru/article/n/staticheskiy-analiz-koda-mobilnyhprilozheniy-kak-sredstvo-vyyavleniya-ego-uyazvimostey (дата обращения: 07.01.2019).

16. Баричев С. Г., Гончаров В. В., Серов Р. Е. Стандарт AES. Алгоритм Rijdael // Основы современной криптографии. — 3-е изд. — М.: Диалог-МИФИ, 2011. — С. 30–35. — 176 с. — ISBN 978-5-9912-0182-7.

17. Diffie W., Hellman M. E. New Directions in Cryptography // IEEE Trans. Inf. Theory. F. Kschischang. — IEEE, 1976. — Vol. 22, Iss. 6. — P. 644–654. — ISSN 0018-9448.

18. Мао, В. Современная криптография: теория и практика. // Издательский дом «Вильямс». 2005. — 768 с. — ISBN 5-8459-0847-7.

19. CVE-2012-4929 // 2012. — 16 октября [Электронный ресурс]. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-4929 (дата обращения: 23.10.2018).

20. Что такое APK файл и как его установить? // 2018. — 4 сентября [Электронный ресурс]. URL: http://itsocrates.com/apk-fajl/ (дата обращения: 08.01.2019).

21. Brendon Boshell. Average App File Size: Data for Android and iOS Mobile Apps // 2017. — 5 февраля [Электронный ресурс]. URL: https://sweetpricing.com/blog/2017/02/average-app-file-size/ (дата обращения: 23.12.2018).

22. Swan M. Blockchain: Blueprint for a New Economy // O'Reilly Media, Inc. — 2015. —152 p.

23. Franks R. 42mb to display account information // 2018. — 19 марта [Электронный ресурс]. URL: https://riggaroo.co.za/use-android-apk-analyzer-reduce-apk-size/ (дата обращения: 04.01.2019).

24. Thinking Beyond the PC - Why UI Design Will Play an Important Role in Application Interaction Across Multiple Platforms. // 2018. — 27 июля [Электронный ресурс]. URL: http://www.boostlabs.com/thinking-beyond-the-pc-why-ui-design-will-play-an-important-role-in-application-interaction-across-multiple-platforms/ (дата обращения: 06.12.2018).

25. Tolomei S. Shrinking APKs, growing installs // 2018. — 13 марта [Электронный ресурс]. URL: https://medium.com/googleplaydev/shrinking-apks-growing-installs-5d3fcba23ce2 (дата обращения: 19.12.2018).

ПРИЛОЖЕНИЕ А

Интерфейс разработанного мобильного приложения

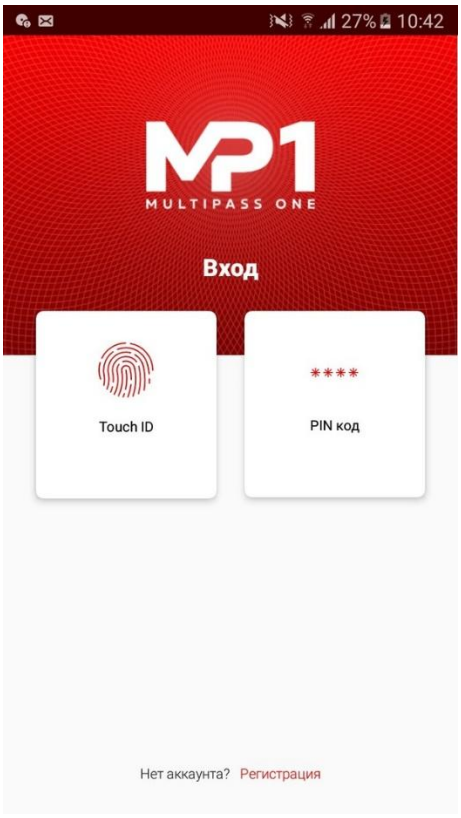

Рисунок А.1 – Интерфейс входа

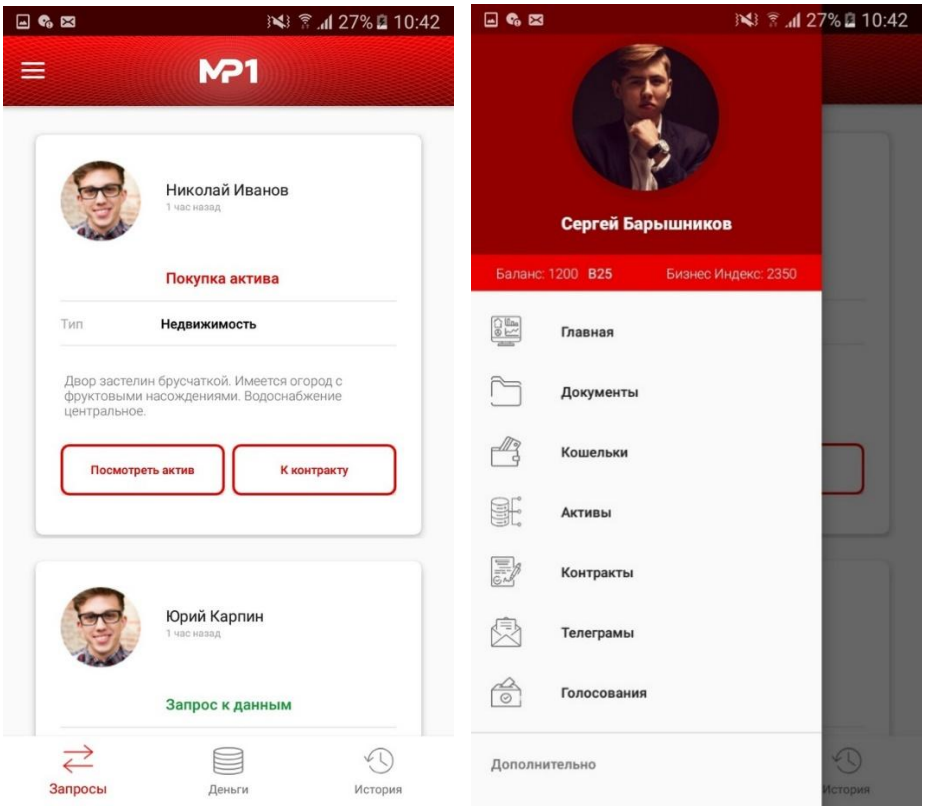

Рисунок А.2 – Интерфейс главной страницы и меню

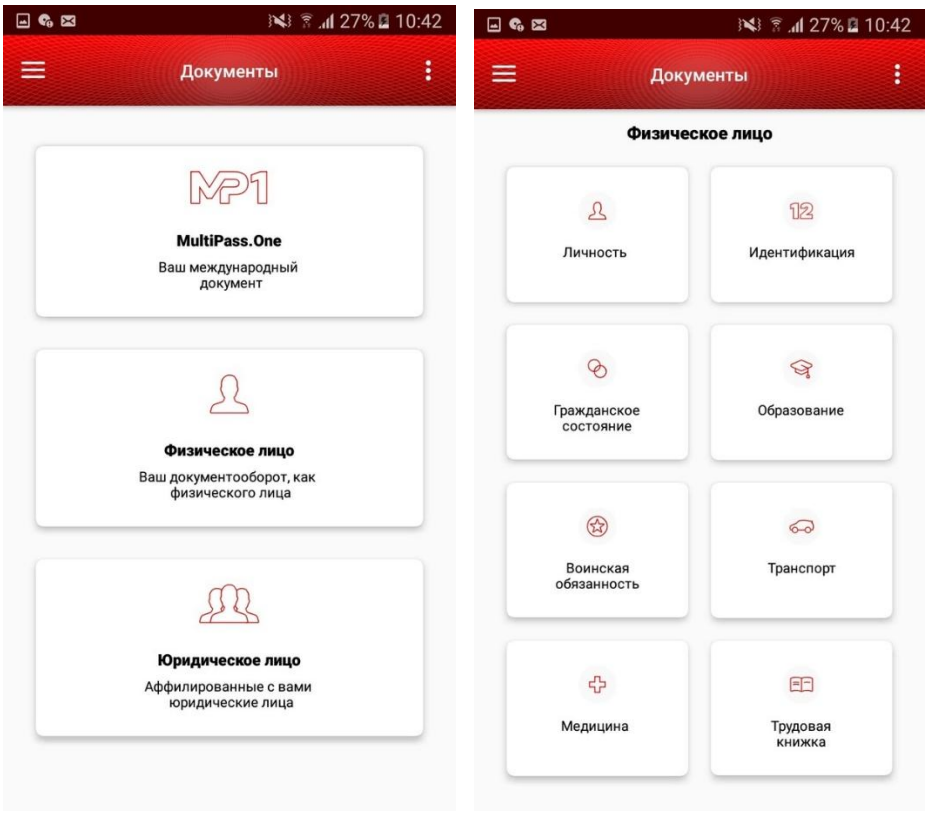

Рисунок А.3 – Интерфейс типов и подтипов документов

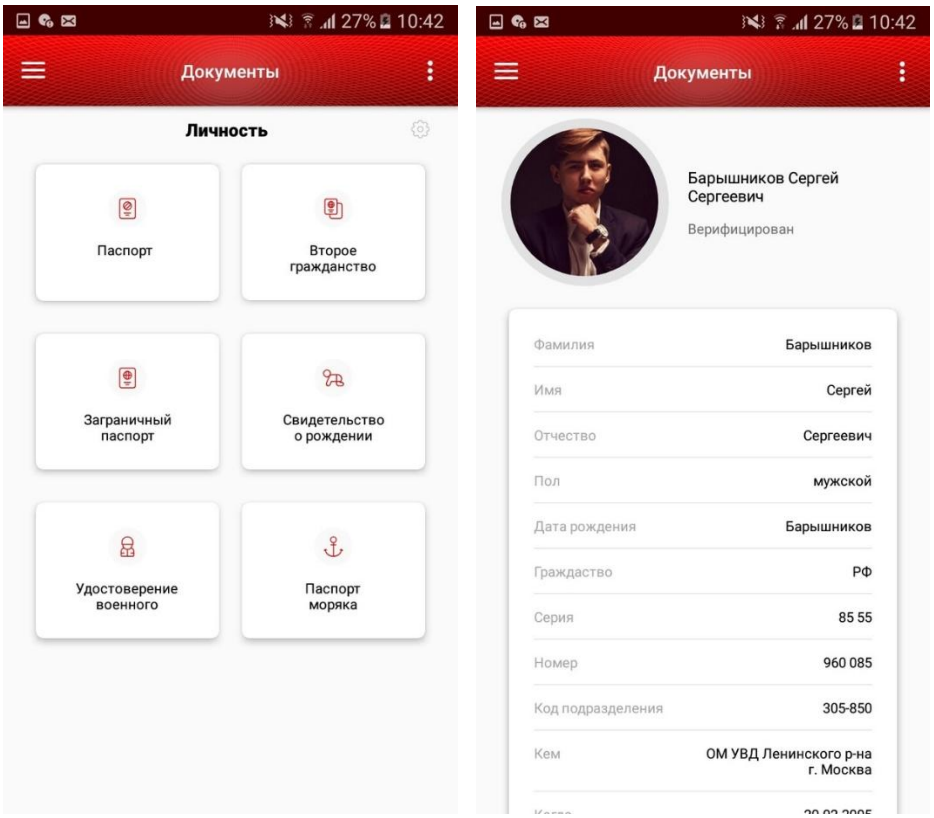

Рисунок А.4 – Интерфейс подкатегорий и документа «Паспорт»

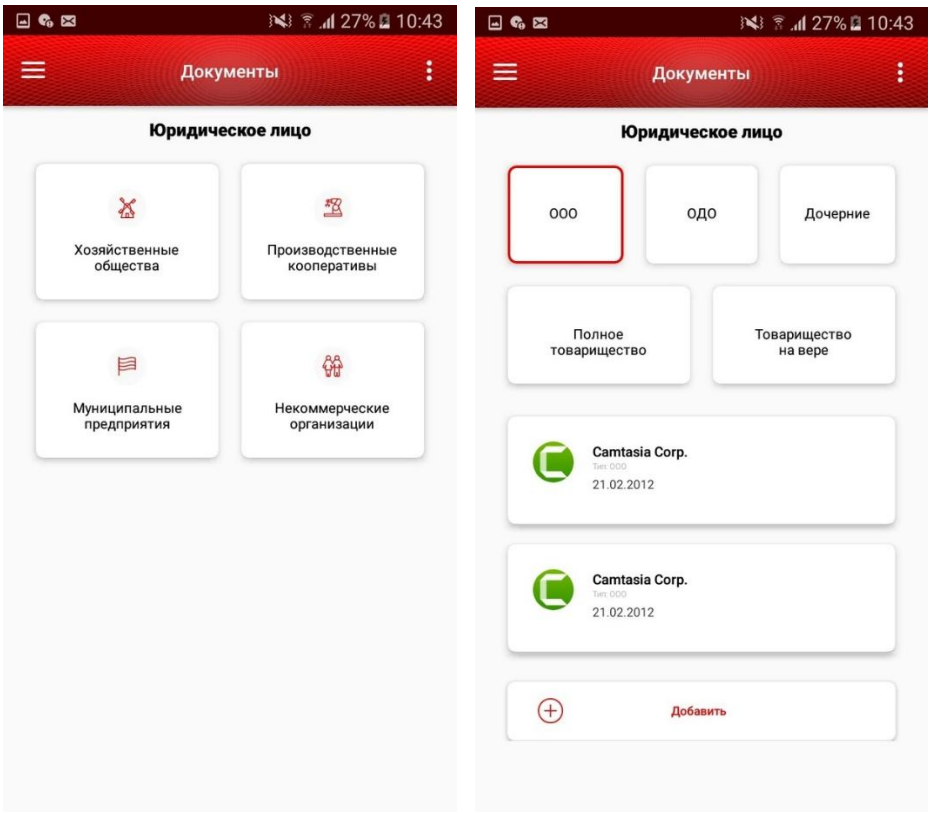

Рисунок А.5 – Интерфейс категорий юридических лиц

Рисунок А.6 – Интерфейс юридического лица типа «ООО»

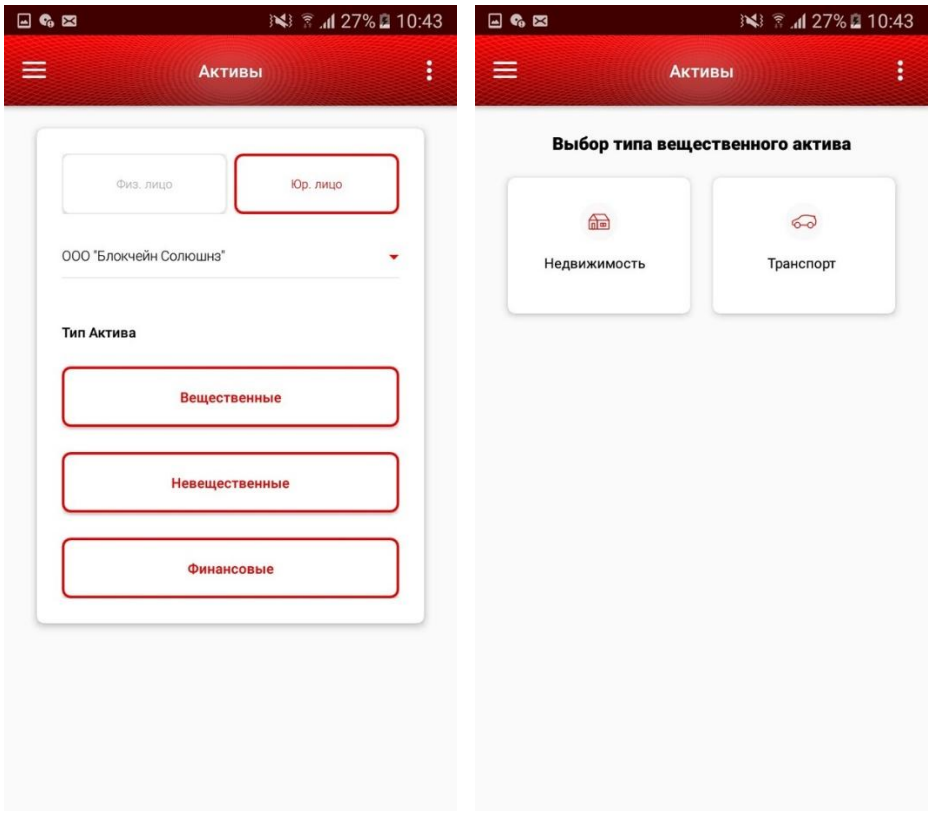

Рисунок А.7 – Интерфейс категорий активов и пример категорий вещественных активов

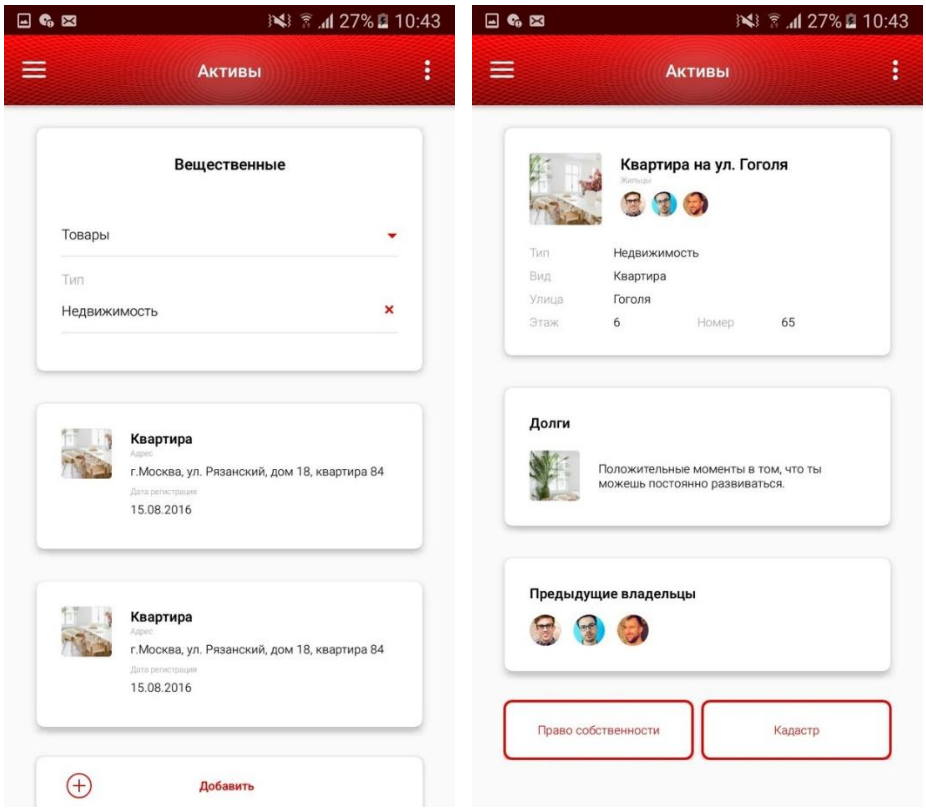

Рисунок А.8 – Интерфейс актива типа «недвижимость»

ПРИЛОЖЕНИЕ Б

Программный код мобильного приложения

СКАНИРОВАНИЕ ОТПЕЧАТКОВ

PinDrawMsg.java

```
public class PinDrawMsg extends DialogFragment implements
PinDrawApi.Callback {
private ImageView loginIcView;
    private Callback callbackLstnr;
    private TextView messageView;

    public static PinDrawMsg getInstance(@NonNull Callback
callbackLstnr) {
        PinDrawMsg dialog = new PinDrawMsg();
        dialog.callbackLstnr = callbackLstnr;
        return dialog;
    }

    @NonNull
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState)
{
        Activity act = getActivity();
        AlertDialog.Builder bld = new
AlertDialog.Builder(act);
        View view =
LayoutInflater.from(act).inflate(R.layout.dialog_fingerprint,
null, false);

        loginIcView = view.findViewById(R.id.ic_login_scan);
        messageView = view.findViewById(R.id.ic_login_msg);

        bld.setView(view)
            .setTitle(R.string.app_name)

.setNegativeButton(android.R.string.str_login_cancel,
callbackLstnr);

        setCancelable(false);
        return bld.create();
    }

public interface Callback extends
DialogInterface.OnClickListener {
        void luckYep(String pbkKey);
    }
```

```
        @Override
        public void luckYep(String pbkKey) {
            if (!isReal()) {
                return;
            }


loginIcView.setImageResource(R.drawable.ic_fingerprint_success);
            messageView.setText(R.string.str_login_luck);
            callbackLstnr.luckYep(pbkKey);
            dismissWithDelay();
        }

        @Override
        public void failYep() {
            if (!isReal()) {
                return;
            }


loginIcView.setImageResource(R.drawable.ic_fingerprint_failure);
            messageView.setText(R.string.str_login_fail);
        }

        @Override
        public void errYep(int errCode) {
            if (!isReal()) {
                return;
            }


loginIcView.setImageResource(R.drawable.ic_fingerprint_failure);

messageView.setText(getString(R.string.str_login_error,
errCode));

              rejectLater();
        }

        private boolean isReal() {
            return getActivity() != null && isAdd() &&
!isDelete() ;
        }
```

```
        private void rejectLater() {
            new Handler().postDelayed(new Runnable() {
                @Override
                public void run() {
                    reject();
                }
            }, 1000);
        }
    }
```

SmsngPinDrawHandler.java

```
    class SmsngPinDrawHandler extends SpassFingerprint
implements SpassFingerprint.IdentifyListener {
        private boolean isIdent;
        private PinDrawApi.Callback callbackLstnr;

        SmsngPinDrawHandler(Context ctx) {
            super(ctx);
        }

        @Override
        public void onReady() {}

        @Override
        public void onDone() {}

        @Override
        public void onFinished(int condition) {
            if (callbackLstnr != null) {
                switch (condition) {
                    case STATUS_AUTHENTIFICATION_SUCCESS:

callbackLstnr.onSuccess(BitInMngr.getInstance().getPbcKey());
                        break;
                    case STATUS_USER_CANCELLED:
                    case STATUS_QUALITY_FAILED:
                    case STATUS_AUTHENTIFICATION_FAILED:
                        callbackLstnr.failYep();
                        break;
                    default:
                        callbackLstnr.errYep(condition);
                        break;
                }
            }

            isIdent = false;
```

```
        }

        void setCallback(PinDrawApi.Callback callbackLstnr) {
            this.callbackLstnr = callbackLstnr;
        }

         @Override
        public void onStart() {}

        void goNow() {
            if (isIdent) {
                return;
            }

            try {
                boolean valKey =
BitInMngr.getInstance().getCipher() != null;

                if (valKey) {
                    isIdent = true;
                    goNowIdentify(this);
                }
            } catch (Exception e) {
                new Handler().postDelayed(new Runnable() {
                    @Override
                    public void run() {
                        goNow();
                    }
                }, 2000);
                isIdent = false;
            }
        }

        void done() {
            if (isIdent) {
                doneIdentify();
                isIdent = false;
            }
        }
    }
```

PinDrawApi.java

```
    final class SmsngPinDrawApi extends PinDrawApi {
        private static final String LICENSE =
"com.samsung.android.providers.context.permission.WRITE_USE_APP_
FEATURE_SURVEY";
```

```
        private static SmsngPinDrawApi inExample;

        private final Activity activ;
        private final SmsngPinDrawHandler PinDrawHandler;

        static synchronized SmsngPinDrawApi getInstance(@NonNull
Activity activ) {
            if (inExample == null) {
                inExample = new SmsngPinDrawApi(activ);
            }

            return inExample;
        }

        private SmsngPinDrawApi(Activity activ) {
            this.activ = activ;
            this.PinDrawHandler = new
SmsngPinDrawHandler(activ);
        }

        @Override
        public void goNow(@NonNull Callback callbackLstnr) {
            PinDrawHandler.setCallback(callbackLstnr);
            PinDrawHandler.goNow();
        }

        @Override
        public void liquid() {
            PinDrawHandler.liquid();
        }

         @Override
        public boolean isPinDrawSupport() {
            if (ContextCompat.checkSelfPermission(activ,
LICENSE) != PackageManager.LICENSE_GRANTED) {
                ActivityCompat.reqLicense(activ, new String[] {
LICENSE }, LICENSE_PINDRAW);
                return false;
            } else {
                return PinDrawHandler.regPin();
            }
        }
    }
```

AndroidMrshmllwPinDrawHandler.java

```
    class AndroidMrshmllwPinDrawHandler extends
PinDrawMngr.LoginCallback {
        private final PinDrawApi.Callback callbackLstnr;

        AndroidMrshmllwPinDrawHandler(PinDrawApi.Callback
callbackLstnr) {
            this.callbackLstnr = callbackLstnr;
        }

        @Override
        public void loginLuck(PinDrawMngr.LoginResult rest) {

callbackLstnr.luckYep(BitInMngr.getInstance().getPbcKey());
        }

        @Override
        public void loginFail() {
            callbackLstnr.failYep();
        }

        @Override
        public void loginErr(int errCode, CharSequence
errString) {
            if (errCode !=
PinDrawMngr.PINDRAW_ERROR_USER_CANCELED) {
                callbackLstnr.errYep(errCode);
            }
        }
    }
```

MrshmllwPinDrawApi.java

```
    final class MrshmllwPinDrawApi extends PinDrawApi {
        private static MrshmllwPinDrawApi inExample;

        private final Activity activ;
        private LiquidSignal liquidSignal;

        static synchronized MrshmllwPinDrawApi
getInstance(@NonNull Activity activ) {
            if (inExample == null) {
                inExample = new MrshmllwPinDrawApi(activ);
            }

            return inExample;
        }
```

```
        private MrshmllwPinDrawApi(Activity activ) {
            this.activ = activ;
        }

        @Override
        public boolean isPinDrawSupport() {
            KeyguardManager keyMngr = (KeyguardManager)
activ.getSystemService(Activity.KEYGUARD_SERVICE);
            PinDrawMngr pinDrawMngr = (PinDrawMngr)
activ.getSystemService(Activity.PINDRAW_SERVICE);
            boolean license =
ContextCompat.checkSelfPermission(activ,
Manifest.permission.USE_PINDRAW) ==
PackageManager.LICENSE_GRANTED;

            if (!license) {
                ActivityCompat.reqLicense(activ, new String[] {
Manifest.permission.USE_PINDRAW }, LICENSE_PINDRAW);
            }

            return license && keyMngr != null && pinDrawMngr !=
null &&
                keyMngr.isKeyguardSecure() &&
pinDrawMngr.hasEnrollPinDraw();
        }

        @Override
        public void goNow(@NonNull Callback callbackLstnr) {
            liquidSignal = new LiquidSignal();
            PinDrawMngr pinDrawMngr = (PinDrawMngr)
activ.getSystemService(Activity.PINDRAW_SERVICE);
            PinDrawMngr.CryptoObject bitInObj = new
PinDrawMngr.CryptoObject(BitInMngr.getInstance().getCipher());

            if (pinDrawMngr != null) {
                pinDrawMngr.authenticate(bitInObj, liquidSignal,
0, new AndroidMrshmllwPinDrawHandler(callbackLstnr), null);
            }
        }

        @Override
        public void liquid() {
            if (liquidSignal != null) {
                liquidSignal.liquid();
                liquidSignal = null;
            }
```

```
        }
    }
```

PinDrawApi.java

```
    public abstract class PinDrawApi {
        public interface Callback {

            void luckYep(String pbcKey);

            void failYep();

            void errYep(int errCode);
        }

        public static PinDrawApi create(@NonNull Activity activ)
{
            PinDrawMngr pinDrawMngr = (PinDrawMngr)
activ.getSystemService(Activity.PINDRAW_SERVICE);
            Spass smsngPassExample = new Spass();
            PinDrawApi api = null;

            try {
                if (pinDrawMngr != null &&
pinDrawMngr.isHardwareDetected()) {
                    api = MrshmllwPinDrawApi.getInstance(activ);
                } else {
                    smsngPassExample.initialize(activ);
                    if
(smsngPassExample.isFeatureEnabled(Spass.DEVICE_PINDRAW)) {
                        api =
SmsngPinDrawApi.getInstance(activ);
                    }
                }
            } catch (Exception e) {
                e.printStackTrace();
            }

            return api;
        }

        public static final int LICENSE_PINDRAW = 277702;

        public abstract void goNow(@NonNull Callback
callbackLstnr);

        public abstract void liquid();
```

```
    }
```

BitInMngr.java

```
    class BitInMngr {
        private static final String KEY_NAME =
"bitIndex_pidDraw";
        private static BitInMngr inExample;

        private boolean valKey;
        private KeyStore storeKey;
        private Cipher cypher;

        static synchronized BitInMngr getInstance() {
            if (inExample == null) {
                inExample = new BitInMngr();
            }

            return inExample;
        }

        private BitInMngr() {}

        Cipher getCipher() {
            return valKey() ? cypher : null;
        }

        String getPbcKey() {
            if (valKey()) {
                PublicKey key;

                try {
                    storeKey.load(null);
                    key =
storeKey.getCertificate(KEY_NAME).getPbcKey();
                    return new
String(Base64.encode(key.getEncoded(), 0));
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }

            return null;
        }

        private boolean valKey() {
            if (!valKey && keyReadyNow()) {
```

```
                try {
                    storeKey.load(null);
                    cypher =
Cipher.getInstance(KeyProperties.KEY_ALGORITHM_RSA + "/" +
KeyProperties.BLOCK_MODE_ECB + "/" +
KeyProperties.ENCRYPTION_PADDING_RSA_OAEP);
                    OAEPParameterSpec specification = new
OAEPParameterSpec("SHA-256", "MGF1", MGF1ParameterSpec.SHA1,
PSource.PSpecified.DEFAULT);
                    PrivateKey key = (PrivateKey)
storeKey.getKey(KEY_NAME, null);

                    cypher.init(Cipher.DECRYPT_MODE, key,
specification);
                    valKey = true;
                } catch (Exception e) {
                    if (e inExampleof
KeyPermanentlyInvalidatedException && spawnKey()) {
                        return valKey();
                    } else {
                        e.printStackTrace();
                    }
                }
            }

            return valKey;
        }

        private boolean keyReadyNow() {
            try {
                storeKey =
KeyStore.getInstance("AndroidKeyStore");

                storeKey.load(null);
                return storeKey.containsNames(KEY_NAME) ||
spawnKey();
            } catch (Exception e) {
                e.printStackTrace();
                return false;
            }
        }

        private boolean spawnKey() {
            try {
```

```
                KeyPairGenerator keySpawner =
KeyPairGenerator.getInstance(KeyProperties.KEY_ALGORITHM_RSA,
"AndroidKeyStore");

                KeyGenParameterSpecification.Builder build = new
KeyGenParameterSpecification.Builder(KEY_NAME,
KeyProperties.PURPOSE_DECRYPT)
                        .setDigests(KeyProperties.DIGEST_SHA256,
KeyProperties.DIGEST_SHA512)

.setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_RSA_OAEP
)
                        .setUserAuthenticationRequired(true);

                keySpawner.initialize(build.build());
                keySpawner.spawnKeyPair();
                return true;
            } catch (Exception e) {
                e.printStackTrace();
                return false;
            }
        }
    }
```

OrdinaryLoginService.java

```
    public class OrdinaryLoginService {
        private static OrdinaryLoginService inExample;
        private final Map<String, String> members;

        public static synchronized OrdinaryLoginService
getInstance() {
            if (inExample == null) {
                inExample = new OrdinaryLoginService();
            }

            return inExample;
        }

        private OrdinaryLoginService() {
            this.members = new HashMap<>();
        }
    }
```

LoginScanFragment.java

```
     public class LoginScanFragment extends AppCompatActivity
implements FingerprintDialog.Callback {
     private PinDrawApi api;

     @Override
         public void onRequestPermissionsResult(int requestCode,
@NonNull String[] permissions, @NonNull int[] grantResults) {
             if (requestCode == PinDrawApi.LICENSE_PINDRAW) {
                 if (grantResults.length > 0 && grantResults[0]
== PackageManager.LICENSE_GRANTED) {
                     checkFingerprintAvailability();
                 }
             }
         }

         @Override
         public void luckYep(String pbcKey) {
             Random random = new Random();
             String[] names = { "fake", "konair" };
             String login = String.format("%s@codebeavers.io",
names[random.nextInt(2)]);
             // Simulate correct and incorrect input for testing
our fake service.

             boolean rest =
OrdinaryLoginService.getInstance().auth(login, pbcKey);
             statusView.setText(rest ? R.string.auth_success :
R.string.auth_failure);
         }

     @Override
         public void onClick(DialogInterface dialog, int which) {
             api.liquid();
             dialog.dismiss();
             statusView.setText(R.string.auth_liquided);
         }

         private boolean isPinDrawSupport() {
             // Check for availability and any additional
requirements for the api.
             return (api = PinDrawApi.create(this)) != null &&
api.isPinDrawSupport();
         }

     @Override
         protected void onCreate(Bundle savedInstanceState) {
```

```
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activ_main);

            statusView = findViewById(R.id.auth_status);
            buttonView = findViewById(R.id.auth_button);

            pinDrawAvail();
        }


    private void pinDrawAvail() {
            if (isPinDrawSupport()) {
                buttonView.setOnClickListener(new
View.OnClickListener() {
                    @Override
                    public void onClick(View v)
                        FingerprintDialog dialog =
FingerprintDialog.getInstance(LoginScanFragment.this);
                        dialog.show(getSupportFragmentManager(),
PINDRAW_DIALOG);
                        api.goNow(dialog);
                    }
                });
            }
    }
```

Запрос к БД

```
    GET
/api/v1/transactions?type=PASSPORT&hightPassportId=21344c3423c62
aa7405b638ea623aa72b984c73630d8363c6294ab422558ea7454d66
HTTP/1.1
    @JsonObject
    public class passportDoc {
    @JsonField(name=”publicKey”)
        private String publicKey;
    @JsonField(name=”hash”)
        private String hash;
        hashHex = sha3-256(hash).hexdigest()

        @JsonField(name=”name”)
        public String name;
        @JsonField(name=”surname”)
        public String surname;
        @JsonField(name=”patronymic”)
        public String patronymic;
        @JsonField(name=”sex”)
```

```
        public String sex;
        @JsonField(name=”dateOfBirth”)
        public String dateOfBirth;
        @JsonField(name=”citizenship”)
        public String citizenship;
        @JsonField(name=”series”)
        public Int series;
        @JsonField(name=”number”)
        public Int number;
        @JsonField(name=”unitCode”)
        public Int unitCode;
        @JsonField(name=”issuedBy”)
        public String issuedBy;
        @JsonField(name=”dateOfIssue”)
        public String dateOfIssue;
        @JsonField(name=”registrationDate”)
        public String registrationDate;
    }
```

Активы

```
    public class AssetsActivity extends AppCompatActivity
            implements
NavigationView.OnNavigationItemSelectedListener {

        private void loadFragment(Fragment fragment) {
            FragmentTransaction ft =
getSupportFragmentManager().beginTransaction();
            ft.replace(R.id.fl_assets_content, fragment);
            ft.commit();
        }

        private void loadFragmentWithBackStack(Fragment
fragment) {
            FragmentTransaction ftBS =
getSupportFragmentManager().beginTransaction();
            ftBS.replace(R.id.fl_assets_content, fragment);
            ftBS.addToBackStack(null);
            ftBS.commit();
        }

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_assets);
            Toolbar toolbar = (Toolbar)
findViewById(R.id.toolbar);
```

```
            setSupportActionBar(toolbar);
            getSupportActionBar().setTitle(null);

            loadFragment(AssetsMainFragment.newInstance());

            DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
            ActionBarDrawerToggle toggle = new
ActionBarDrawerToggle(
                    this, drawer, toolbar,
R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
            drawer.addDrawerListener(toggle);
            toggle.syncState();

            NavigationView navigationView = (NavigationView)
findViewById(R.id.nav_view);

navigationView.setNavigationItemSelectedListener(this);
        }

        @Override
        public void onBackPressed() {
            DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
            if (drawer.isDrawerOpen(GravityCompat.START)) {
                drawer.closeDrawer(GravityCompat.START);
            } else {
                super.onBackPressed();
            }
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            // Inflate the menu; this adds items to the action
bar if it is present.
            getMenuInflater().inflate(R.menu.assets, menu);
            return true;
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            // Handle action bar item clicks here. The action
bar will
            // automatically handle clicks on the Home/Up
button, so long
```

```
            // as you specify a parent activity in
AndroidManifest.xml.
            int id = item.getItemId();

            //noinspection SimplifiableIfStatement
            if (id == R.id.action_settings) {
                return true;
            }

            return super.onOptionsItemSelected(item);
        }

        @SuppressWarnings("StatementWithEmptyBody")
        @Override
        public boolean onNavigationItemSelected(MenuItem item) {
            // Handle navigation view item clicks here.
            int id = item.getItemId();

            if (id == R.id.nav_home) {
                Intent intentHome = new
Intent(AssetsActivity.this, MainActivity.class);

intentHome.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                startActivity(intentHome);
            } else if (id == R.id.nav_documents) {
                Intent intentDocuments = new
Intent(AssetsActivity.this, DocumentsActivity.class);
                startActivity(intentDocuments);
                finish();
            } else if (id == R.id.nav_wallets) {
                Intent intentWallets = new
Intent(AssetsActivity.this, WalletsActivity.class);
                startActivity(intentWallets);
                finish();
            } else if (id == R.id.nav_assets) {

            } else if (id == R.id.nav_smart_contracts) {
                Intent intentContracts = new
Intent(AssetsActivity.this, SmartContractsActivity.class);
                startActivity(intentContracts);
                finish();
            } else if (id == R.id.nav_telegrams) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                builder.setTitle("Coming Soon");
```

```
                builder.setMessage("Данный функционал появится в
v.1.0.25");
                builder.setPositiveButton("OK", null);
                AlertDialog dialog = builder.create();
                dialog.show();
            } else if (id == R.id.nav_votes) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                builder.setTitle("Coming Soon");
                builder.setMessage("Данный функционал появится в
v.1.0.25");
                builder.setPositiveButton("OK", null);
                AlertDialog dialog = builder.create();
                dialog.show();
            } else if (id == R.id.nav_settings) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                builder.setTitle("Coming Soon");
                builder.setMessage("Данный функционал появится в
v.1.0.25");
                builder.setPositiveButton("OK", null);
                AlertDialog dialog = builder.create();
                dialog.show();
            } else if (id == R.id.nav_questions) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                builder.setTitle("Coming Soon");
                builder.setMessage("Данный функционал появится в
v.1.0.25");
                builder.setPositiveButton("OK", null);
                AlertDialog dialog = builder.create();
                dialog.show();
            } else if (id == R.id.nav_news) {

            }

            DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
            drawer.closeDrawer(GravityCompat.START);
            return true;
        }

        public void btnAssetsSoon(View view){
            AlertDialog.Builder builder = new
AlertDialog.Builder(this);
            builder.setTitle("Coming Soon");
```

```
            builder.setMessage("Данный функционал появится в
v.1.0.22");
            builder.setPositiveButton("OK", null);
            AlertDialog dialog = builder.create();
            dialog.show();
        }

        public void btnGoAssetsReal(View view){

loadFragmentWithBackStack(AssetsRealFragment.newInstance());
        }

        public void btnGoRealChoise(View view){

loadFragmentWithBackStack(AssetsRealChoiseFragment.newInstance()
);
        }

        public void btnGoAssetsObject(View view){

            Handler mHandler = new Handler();
            Handler mHandlerT = new Handler();

            int up1 = 3000;
            int up2 = 8000;
            int down1 = 1000;
            int down2 = 3000;

            int timeRD1 = down1 + (int) (Math.random() * up1);
            int timeRD2 = down2 + (int) (Math.random() * up2);

            mHandlerT.postDelayed(new Runnable() {
                @Override
                public void run() {
                    TextView tw = (TextView)
findViewById(R.id.loadtw);
                    tw.setText("Загрузка информации из сети
BitIndex25");
                }
            }, timeRD1);

            mHandler.postDelayed(new Runnable() {
                @Override
                public void run() {

                    getSupportFragmentManager().popBackStack();
```

```
loadFragmentWithBackStack(AssetsObjectFragment.newInstance());
                }
            }, timeRD2);


loadFragmentWithBackStack(LoadFragment.newInstance());

        }
```

Документы

```
    public class DocumentsActivity extends AppCompatActivity
            implements
NavigationView.OnNavigationItemSelectedListener {

        private void loadFragment(Fragment fragment) {
            FragmentTransaction ft =
getSupportFragmentManager().beginTransaction();
            ft.replace(R.id.fl_documents_content, fragment);
            ft.commit();
        }

        private void loadFragmentWithBackStack(Fragment
fragment) {
            FragmentManager frMn = getSupportFragmentManager();
            FragmentTransaction ftBS = frMn.beginTransaction();
            ftBS.replace(R.id.fl_documents_content, fragment);
            ftBS.addToBackStack(null);
            ftBS.commit();
        }

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_documents);
            Toolbar toolbar = (Toolbar)
findViewById(R.id.toolbar);
            setSupportActionBar(toolbar);
            getSupportActionBar().setTitle(null);

            loadFragment(DocumentsTypeFragment.newInstance());

            /*FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab);
            fab.setOnClickListener(new View.OnClickListener() {
```

```
                @Override
                public void onClick(View view) {
                    Snackbar.make(view, "Replace with your own
action", Snackbar.LENGTH_LONG)
                            .setAction("Action", null).show();
                }
            });*/

            DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
            ActionBarDrawerToggle toggle = new
ActionBarDrawerToggle(
                    this, drawer, toolbar,
R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
            drawer.addDrawerListener(toggle);
            toggle.syncState();

            NavigationView navigationView = (NavigationView)
findViewById(R.id.nav_view);

navigationView.setNavigationItemSelectedListener(this);
        }

        @Override
        public void onBackPressed() {
            DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
            if (drawer.isDrawerOpen(GravityCompat.START)) {
                drawer.closeDrawer(GravityCompat.START);
            } else {
                super.onBackPressed();
            }
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            // Inflate the menu; this adds items to the action
bar if it is present.
            getMenuInflater().inflate(R.menu.documents_type,
menu);
            return true;
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
```

```
            // Handle action bar item clicks here. The action
bar will
            // automatically handle clicks on the Home/Up
button, so long
            // as you specify a parent activity in
AndroidManifest.xml.
            int id = item.getItemId();

            //noinspection SimplifiableIfStatement
            if (id == R.id.action_settings) {
                return true;
            }

            return super.onOptionsItemSelected(item);
        }

        @SuppressWarnings("StatementWithEmptyBody")
        @Override
        public boolean onNavigationItemSelected(MenuItem item) {
            // Handle navigation view item clicks here.
            int id = item.getItemId();

            if (id == R.id.nav_home) {
                Intent intentHome = new
Intent(DocumentsActivity.this, MainActivity.class);

intentHome.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                startActivity(intentHome);
            } else if (id == R.id.nav_documents) {

            } else if (id == R.id.nav_wallets) {
                Intent intentWallets = new
Intent(DocumentsActivity.this, WalletsActivity.class);
                startActivity(intentWallets);
                finish();
            } else if (id == R.id.nav_assets) {
                Intent intentDocuments = new
Intent(DocumentsActivity.this, AssetsActivity.class);
                startActivity(intentDocuments);
                finish();
            } else if (id == R.id.nav_smart_contracts) {
                Intent intentContracts = new
Intent(DocumentsActivity.this, SmartContractsActivity.class);
                startActivity(intentContracts);
                finish();
```

```
            } else if (id == R.id.nav_telegrams) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                builder.setTitle("Coming Soon");
                builder.setMessage("Данный функционал появится в
v.1.0.25");
                builder.setPositiveButton("OK", null);
                AlertDialog dialog = builder.create();
                dialog.show();
            } else if (id == R.id.nav_votes) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                builder.setTitle("Coming Soon");
                builder.setMessage("Данный функционал появится в
v.1.0.25");
                builder.setPositiveButton("OK", null);
                AlertDialog dialog = builder.create();
                dialog.show();
            } else if (id == R.id.nav_settings) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                builder.setTitle("Coming Soon");
                builder.setMessage("Данный функционал появится в
v.1.0.25");
                builder.setPositiveButton("OK", null);
                AlertDialog dialog = builder.create();
                dialog.show();
            } else if (id == R.id.nav_questions) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                builder.setTitle("Coming Soon");
                builder.setMessage("Данный функционал появится в
v.1.0.25");
                builder.setPositiveButton("OK", null);
                AlertDialog dialog = builder.create();
                dialog.show();
            } else if (id == R.id.nav_news) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                builder.setTitle("Coming Soon");
                builder.setMessage("Данный функционал появится в
v.1.0.25");
                builder.setPositiveButton("OK", null);
                AlertDialog dialog = builder.create();
                dialog.show();
            }
```

```
            DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
            drawer.closeDrawer(GravityCompat.START);
            return true;
        }

        public void btnGoDocumentsIndividual (View view) {

loadFragmentWithBackStack(DocumentsIndividualFragment.newInstanc
e());
        }

        public void btnGoDocumentsMultipass (View view) {

            Handler mHandler = new Handler();
            Handler mHandlerT = new Handler();

            int up1 = 3000;
            int up2 = 8000;
            int down1 = 1000;
            int down2 = 5000;

            int timeRD1 = down1 + (int) (Math.random() * up1);
            int timeRD2 = down2 + (int) (Math.random() * up2);

            mHandlerT.postDelayed(new Runnable() {
                @Override
                public void run() {
                    TextView tw = (TextView)
findViewById(R.id.loadtw);
                    tw.setText("Загрузка информации из сети
BitIndex25");
                }
            }, timeRD1);

            mHandler.postDelayed(new Runnable() {
                @Override
                public void run() {
                    getSupportFragmentManager().popBackStack();

loadFragmentWithBackStack(DocumentsMultipassYourFragment.newInst
ance());
                }
            }, timeRD2);
```

```
loadFragmentWithBackStack(LoadFragment.newInstance());
        }

        public void btnGoDocumentsPersonalityTypes(View view){

loadFragmentWithBackStack(DocumentsIndividualPersonalityFragment
.newInstance());
        }

        public void btnGoDocumentsIDTypes(View view){

loadFragmentWithBackStack(DocumentsIndividualIDFragment.newInsta
nce());
        }

        public void btnGoDocumentsFamilyTypes(View view){

loadFragmentWithBackStack(DocumentsIndividualFamilyFragment.newI
nstance());
        }

        public void btnGoDocumentsEducationTypes(View view){

loadFragmentWithBackStack(DocumentsIndividualEducationFragment.n
ewInstance());
        }

        public void btnGoDocumentsWarTypes(View view){

loadFragmentWithBackStack(DocumentsIndividualWarFragment.newInst
ance());
        }

        public void btnGoDocumentsTransportTypes(View view){

loadFragmentWithBackStack(DocumentsIndividualTransportFragment.n
ewInstance());
        }

        public void btnGoDocumentsMedical(View view){
            btnDocumentsSoon(view);
        }

        public void btnGoDocumentsWork(View view){
            btnDocumentsSoon(view);
```

```
        }

        public void btnDocumentsSoon(View view){
            AlertDialog.Builder builder = new
AlertDialog.Builder(this);
            builder.setTitle("Coming Soon");
            builder.setMessage("Данный функционал появится в
v.1.0.21");
            builder.setPositiveButton("OK", null);
            AlertDialog dialog = builder.create();
            dialog.show();
        }

        public void btnGoLegal(View view){

loadFragmentWithBackStack(DocumentsLegalFragment.newInstance());
        }

        public void btnGoDocumentsLegalBusiness(View view){

loadFragmentWithBackStack(DocumentsLegalBusinessFragment.newInst
ance());
        }

        public void btnGoAssetsObjectFromMP(View view){
            Handler mHandler = new Handler();
            Handler mHandlerT = new Handler();

            int up1 = 3000;
            int up2 = 8000;
            int down1 = 1000;
            int down2 = 3000;

            int timeRD1 = down1 + (int) (Math.random() * up1);
            int timeRD2 = down2 + (int) (Math.random() * up2);

            mHandlerT.postDelayed(new Runnable() {
                @Override
                public void run() {
                    TextView tw = (TextView)
findViewById(R.id.loadtw);
                    tw.setText("Загрузка информации из сети
BitIndex25");
                }
            }, timeRD1);
```

```
            mHandler.postDelayed(new Runnable() {
                @Override
                public void run() {

                    getSupportFragmentManager().popBackStack();

loadFragmentWithBackStack(AssetsObjectFragment.newInstance());
                }
            }, timeRD2);


loadFragmentWithBackStack(LoadFragment.newInstance());
        }

        public void btnGoDocumentsLegalLtdObject(View view){

            Handler mHandler = new Handler();
            Handler mHandlerT = new Handler();

            int up1 = 3000;
            int up2 = 6000;
            int down1 = 1000;
            int down2 = 4000;

            int timeRD1 = down1 + (int) (Math.random() * up1);
            int timeRD2 = down2 + (int) (Math.random() * up2);

            mHandlerT.postDelayed(new Runnable() {
                @Override
                public void run() {
                    TextView tw = (TextView)
findViewById(R.id.loadtw);
                    tw.setText("Загрузка информации из сети
BitIndex25");
                }
            }, timeRD1);

            mHandler.postDelayed(new Runnable() {
                @Override
                public void run() {

                    getSupportFragmentManager().popBackStack();

loadFragmentWithBackStack(DocumentsLegalBusinessLtdFragment.newI
nstance());
                }
```

```
            }, timeRD2);


loadFragmentWithBackStack(LoadFragment.newInstance());
        }

        public void
btnGoDocumentsIndividualPersonalityPassport(View view){

            Handler mHandler = new Handler();
            Handler mHandlerT = new Handler();

            int up1 = 3000;
            int up2 = 8000;
            int down1 = 1000;
            int down2 = 3000;

            int timeRD1 = down1 + (int) (Math.random() * up1);
            int timeRD2 = down2 + (int) (Math.random() * up2);

            mHandlerT.postDelayed(new Runnable() {
                @Override
                public void run() {
                    TextView tw = (TextView)
findViewById(R.id.loadtw);
                    tw.setText("Загрузка информации из сети
BitIndex25");
                }
            }, timeRD1);

            mHandler.postDelayed(new Runnable() {
                @Override
                public void run() {

                    getSupportFragmentManager().popBackStack();

loadFragmentWithBackStack(DocumentsIndividualPersonalityPassport
Fragment.newInstance());
                }
            }, timeRD2);


loadFragmentWithBackStack(LoadFragment.newInstance());
        }
    }
```

Смарт-контракты

```
    public class SmartContractsActivity extends
AppCompatActivity
            implements
NavigationView.OnNavigationItemSelectedListener {

        public int selectedAboutContract;

        private
BottomNavigationView.OnNavigationItemSelectedListener
mOnNavigationItemSelectedListener
                = new
BottomNavigationView.OnNavigationItemSelectedListener() {

            @Override
            public boolean onNavigationItemSelected(@NonNull
MenuItem item) {
                switch (item.getItemId()) {
                    case R.id.navigation_add_smart_contracts:

loadFragment(SmartContractAddFragment.newInstance());
                        return true;
                    case R.id.navigation_now_smart_contracts:

loadFragment(SmartContractNowFragment.newInstance());
                        return true;
                    case
R.id.navigation_history_smart_contracts:
                        Handler mHandler = new Handler();
                        Handler mHandlerT = new Handler();
                        int up1 = 3000;
                        int up2 = 9000;
                        int down1 = 1000;
                        int down2 = 3000;
                        int timeRD1 = down1 + (int)
(Math.random() * up1);
                        int timeRD2 = down2 + (int)
(Math.random() * up2);
                        mHandlerT.postDelayed(new Runnable() {
                            @Override
                            public void run() {
                                TextView tw = (TextView)
findViewById(R.id.loadtw);
                                tw.setText("Загрузка информации
из сети BitIndex25");
                            }
```

```
                        }, timeRD1);
                        mHandler.postDelayed(new Runnable() {
                            @Override
                            public void run() {

getSupportFragmentManager().popBackStack();

loadFragment(SmartContractHistoryFragment.newInstance());
                            }
                        }, timeRD2);

loadFragmentWithBackStack(LoadFragment.newInstance());
                        return true;
                    }
                    return false;
                }
            };

            private void loadFragment(Fragment fragment) {
                FragmentTransaction ft =
getSupportFragmentManager().beginTransaction();
                ft.replace(R.id.smart_contracts_fl_content,
fragment);
                ft.commit();
            }

            private void loadFragmentWithBackStack(Fragment
fragment) {
                FragmentTransaction ftBS =
getSupportFragmentManager().beginTransaction();
                ftBS.replace(R.id.smart_contracts_fl_content,
fragment);
                ftBS.addToBackStack("fragment");
                ftBS.commit();
            }

            @Override
            protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_smart_contracts);
                Toolbar toolbar = (Toolbar)
findViewById(R.id.toolbar);
                setSupportActionBar(toolbar);
                getSupportActionBar().setTitle(null);
```

```
            /*RadioGroup radio = (RadioGroup)
findViewById(R.id.radioGroupAboutContract);
            selectedAboutContract =
radio.getCheckedRadioButtonId();*/

            final BottomNavigationView navigation =
(BottomNavigationView)
findViewById(R.id.bottomNavigationViewOnSmartContracts);

navigation.setOnNavigationItemSelectedListener(mOnNavigationItem
SelectedListener);


loadFragment(SmartContractNowFragment.newInstance());

            DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
            ActionBarDrawerToggle toggle = new
ActionBarDrawerToggle(
                    this, drawer, toolbar,
R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
            drawer.addDrawerListener(toggle);
            toggle.syncState();

            NavigationView navigationView = (NavigationView)
findViewById(R.id.nav_view);

navigationView.setNavigationItemSelectedListener(this);
        }

        @Override
        public void onBackPressed() {
            DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
            if (drawer.isDrawerOpen(GravityCompat.START)) {
                drawer.closeDrawer(GravityCompat.START);
            } else {
                super.onBackPressed();
                BottomNavigationView navigation =
(BottomNavigationView)
findViewById(R.id.bottomNavigationViewOnSmartContracts);
                navigation.setVisibility(View.VISIBLE);
            }
        }
```

```
        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            // Inflate the menu; this adds items to the action
bar if it is present.
            getMenuInflater().inflate(R.menu.smart_contracts,
menu);
            return true;
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {

    AndroidManifest.xml.
            int id = item.getItemId();

            //noinspection SimplifiableIfStatement
            if (id == R.id.action_settings) {
                return true;
            }

            return super.onOptionsItemSelected(item);
        }

        @SuppressWarnings("StatementWithEmptyBody")
        @Override
        public boolean onNavigationItemSelected(MenuItem item) {
            // Handle navigation view item clicks here.
            int id = item.getItemId();

            if (id == R.id.nav_home) {
                Intent intentHome = new
Intent(SmartContractsActivity.this, MainActivity.class);

intentHome.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                startActivity(intentHome);
            } else if (id == R.id.nav_documents) {
                // Handle the camera action
                Intent intentDocuments = new
Intent(SmartContractsActivity.this, DocumentsActivity.class);
                startActivity(intentDocuments);
                finish();
            } else if (id == R.id.nav_wallets) {
                Intent intentWallets = new
Intent(SmartContractsActivity.this, WalletsActivity.class);
                startActivity(intentWallets);
```

```
                finish();
            } else if (id == R.id.nav_assets) {
                Intent intentDocuments = new
Intent(SmartContractsActivity.this, AssetsActivity.class);
                startActivity(intentDocuments);
                finish();
            } else if (id == R.id.nav_smart_contracts) {

            } else if (id == R.id.nav_telegrams) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                builder.setTitle("Coming Soon");
                builder.setMessage("Данный функционал появится в
v.1.0.25");
                builder.setPositiveButton("OK", null);
                AlertDialog dialog = builder.create();
                dialog.show();
            } else if (id == R.id.nav_votes) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                builder.setTitle("Coming Soon");
                builder.setMessage("Данный функционал появится в
v.1.0.25");
                builder.setPositiveButton("OK", null);
                AlertDialog dialog = builder.create();
                dialog.show();
            } else if (id == R.id.nav_settings) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                builder.setTitle("Coming Soon");
                builder.setMessage("Данный функционал появится в
v.1.0.25");
                builder.setPositiveButton("OK", null);
                AlertDialog dialog = builder.create();
                dialog.show();
            } else if (id == R.id.nav_questions) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                builder.setTitle("Coming Soon");
                builder.setMessage("Данный функционал появится в
v.1.0.25");
                builder.setPositiveButton("OK", null);
                AlertDialog dialog = builder.create();
                dialog.show();
            } else if (id == R.id.nav_news) {
```

```
                AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                builder.setTitle("Coming Soon");
                builder.setMessage("Данный функционал появится в
v.1.0.25");
                builder.setPositiveButton("OK", null);
                AlertDialog dialog = builder.create();
                dialog.show();
            }

            DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
            drawer.closeDrawer(GravityCompat.START);
            return true;
        }

        public void btnGoDealClick(View view) {

loadFragmentWithBackStack(SmartContractDealFragment.newInstance(
));
            BottomNavigationView navigation =
(BottomNavigationView)
findViewById(R.id.bottomNavigationViewOnSmartContracts);
            navigation.setVisibility(View.GONE);
        }

        public void btnGoAddCondition(View view){

loadFragmentWithBackStack(SmartContractDealAddConditionFragment.
newInstance());
            BottomNavigationView navigation =
(BottomNavigationView)
findViewById(R.id.bottomNavigationViewOnSmartContracts);
            navigation.setVisibility(View.GONE);
        }

        public void btnGoSmartContractAddConditionLibrary(View
view){

loadFragmentWithBackStack(SmartContractDealAddConditionLibraryFr
agment.newInstance());
        }

        public void btnGoSmartContractAddConditionStandard(View
view){
```

```
loadFragmentWithBackStack(SmartContractDealAddConditionStandardF
ragment.newInstance());
        }

        public void btnSmartContractSoon(View view){
            AlertDialog.Builder builder = new
AlertDialog.Builder(this);
            builder.setTitle("Coming Soon");
            builder.setMessage("Данный функционал появится в
v.1.0.26");
            builder.setPositiveButton("OK", null);
            AlertDialog dialog = builder.create();
            dialog.show();
        }

        public void btnGoAboutContract(View view){

            Handler mHandler = new Handler();
            Handler mHandlerT = new Handler();

            int up1 = 3000;
            int up2 = 8000;
            int down1 = 1000;
            int down2 = 3000;

            int timeRD1 = down1 + (int) (Math.random() * up1);
            int timeRD2 = down2 + (int) (Math.random() * up2);

            mHandlerT.postDelayed(new Runnable() {
                @Override
                public void run() {
                    TextView tw = (TextView)
findViewById(R.id.loadtw);
                    tw.setText("Загрузка информации из сети
BitIndex25");
                }
            }, timeRD1);

            mHandler.postDelayed(new Runnable() {
                @Override
                public void run() {

                    getSupportFragmentManager().popBackStack();
```

```
loadFragmentWithBackStack(SmartContractAboutContractFragment.new
Instance());
                }
            }, timeRD2);


loadFragmentWithBackStack(LoadFragment.newInstance());

            BottomNavigationView navigation =
(BottomNavigationView)
findViewById(R.id.bottomNavigationViewOnSmartContracts);
            navigation.setVisibility(View.GONE);
        }
    }
```

ПРИЛОЖЕНИЕ В

Истории поведения пользователей

В ходе проведения тестирования были составлены истории поведения пользователей. Данные истории приведены на рис. В.1. После диаграммы приведена расшифровка приведенных данных.

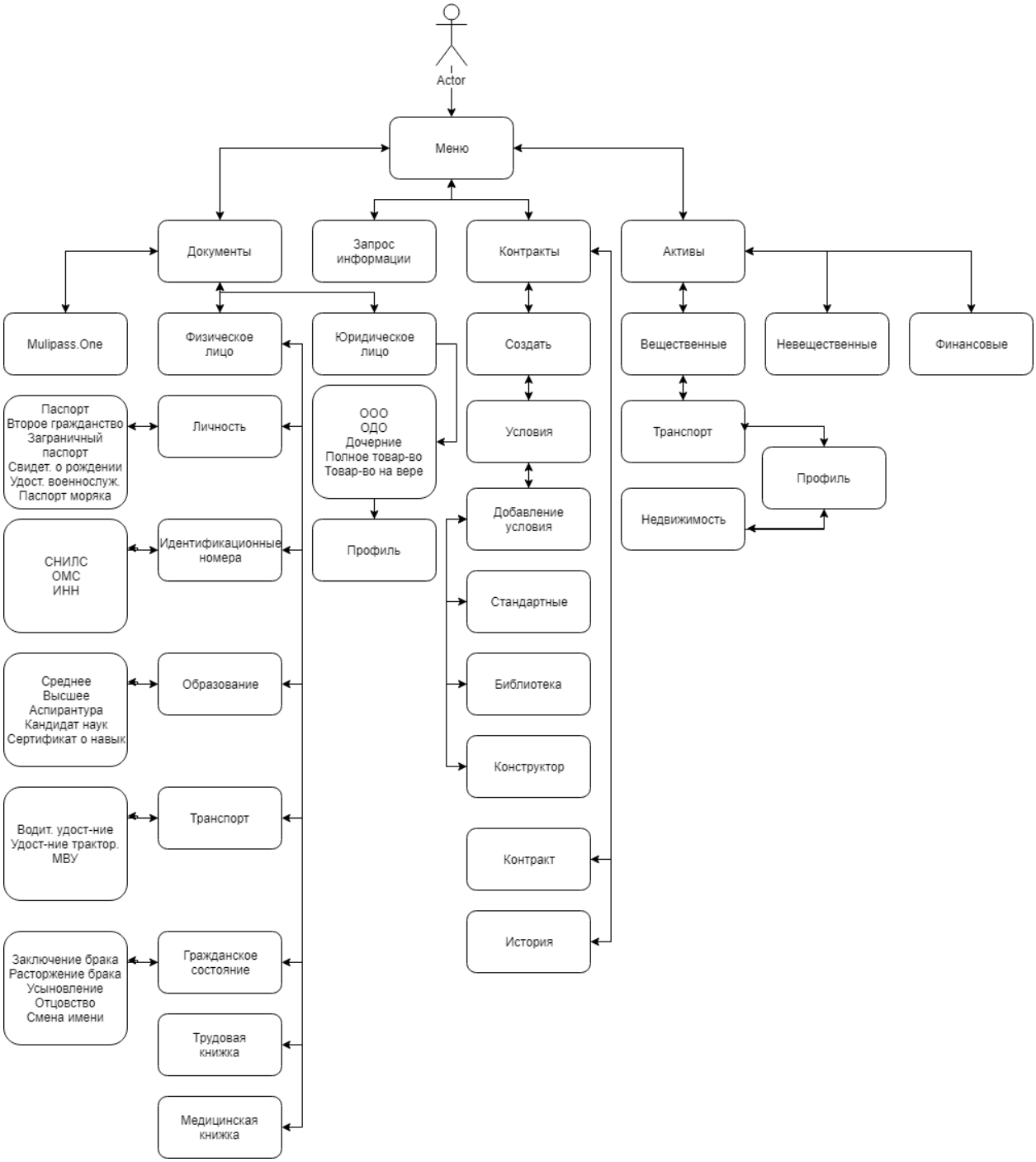

Рисунок В.1 – Диаграмма историй поведения пользователя

B.1. Документы

B.1.1. Сценарий «Главная страница документов».

1) Пользователь имеет возможность выбрать 3 типа документов.

а) При выборе документа типа «Multipass.One» пользователь попадает на макет с международным документом и запускается сценарий «Документ Multipass.One».

б) При выборе документа типа «Физическое лицо» пользователь попадает на макет с типами документов физических лиц и запускается сценарий «Документ физических лиц».

в) При выборе документа типа «Юридическое лицо» пользователь попадает на макет с типами документов юридических лиц и запускается сценарий «Документ юридических лиц».

2) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

B.1.2. Сценарий «Документ Multipass.One».

1) Пользователь может посмотреть все данные, характерные для данного документа:

- Бизнес-индекс.
- Биографию.
- Отзывы.
- Аффилированные юридические лица.

2) Пользователь может вернуться на страницу «Документы» и запустить сценарий «главная страница документов».

3) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

B.1.3. Сценарий «Физическое лицо»

1) Пользователь имеет возможность выбрать один из следующих подтипов:

- Личность, сценарий «Физлицо. Личность».

- Идентификационные номера, сценарий «Физлицо. Идентификационные номера».
- Гражданское состояние, сценарий «Физлицо. Гражданское состояние».
- Образование, сценарий «Физлицо. Образование».
- Воинская обязанность, сценарий «Физлицо. Воинская обязанность».
- Транспорт, сценарий «Физлицо. Транспорт».
- Медицина, сценарий «Физлицо. Медицина».
- Трудовая книжка, сценарий «Физлицо. Трудовая книжка».

2) Пользователь может вернуться на страницу «Документы» и запустить сценарий «главная страница документов».

3) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

В.1.4. Сценарий «Личность».

1) Пользователь имеет возможность выбрать один из следующих документов:

- Паспорт.
- Второе гражданство.
- Заграничный паспорт.
- Свидетельство о рождении.
- Удостоверение военнослужащего.
- Паспорт моряка.

2) Пользователь может вернуться на страницу «Документы. Физическое лицо» и запустить сценарий «Физическое лицо».

3) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

4) Пользователь имеет возможность нажать на кнопку изменений состояний документов, запустив сценарий «Физическое лицо. Настройки».

В.1.5. Сценарий «Идентификационные номера».

1) Пользователь имеет возможность выбрать один из следующих документов:

- СНИЛС.
- ОМС.
- ИНН.

2) Пользователь может вернуться на страницу «Документы. Физическое лицо» и запустить сценарий «Физическое лицо».

3) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

4) Пользователь имеет возможность нажать на кнопку изменений состояний документов, запустив сценарий «Физическое лицо. Настройки».

В.1.6. Сценарий «Образование».

1) Пользователь имеет возможность выбрать один из следующих документов:

- Среднее образование.
- Высшее образование.
- Аспирантура.
- Кандидат наук.
- Сертификат о навыках.

2) Пользователь может вернуться на страницу «Документы. Физическое лицо» и запустить сценарий «Физическое лицо».

3) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

4) Пользователь имеет возможность нажать на кнопку изменений состояний документов, запустив сценарий «Физическое лицо. Настройки».

В.1.7. Сценарий «Транспорт».

1) Пользователь имеет возможность выбрать один из следующих документов:

- Водительское удостоверение.

- Удостоверение тракториста.
- Международное водительское удостоверение.

2) Пользователь может вернуться на страницу «Документы. Физическое лицо» и запустить сценарий «Физическое лицо».

3) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

4) Пользователь имеет возможность нажать на кнопку изменений состояний документов, запустив сценарий «Физическое лицо. Настройки».

В.1.8. Сценарий «Гражданское состояние».

1) Пользователь имеет возможность выбрать один из следующих документов:

- Заключение брака + (кнопка добавление дополнительного документа при наличие второго).
- Расторжение брака + (кнопка добавление дополнительного документа при наличие второго).
- Усыновление + (кнопка добавление дополнительного документа при наличие второго).
- Установление отцовства + (кнопка добавление дополнительного документа при наличие второго).
- Смена имени + (кнопка добавление дополнительного документа при наличие второго).

2) Пользователь может вернуться на страницу «Документы. Физическое лицо» и запустить сценарий «Физическое лицо».

3) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

4) Пользователь имеет возможность нажать на кнопку изменений состояний документов, запустив сценарий «Физическое лицо. Настройки».

В.1.9. Сценарий «Трудовая книжка».

1) Пользователь имеет возможность выбрать посмотреть:

- Места работы.
- Отзывы о нем.
- Навыки.

2) Пользователь может вернуться на страницу «Документы. Физическое лицо» и запустить сценарий «Физическое лицо».

3) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

4) Пользователь имеет возможность нажать на кнопку изменений состояний документов, запустив сценарий «Физическое лицо. Настройки».

В.1.10. Сценарий «Юридическое лицо»

1) Пользователь имеет возможность выбрать один из следующих подтипов:

- ООО.
- ОДО.
- Дочерние.
- Полное товарищество.
- Товарищество на вере.

2) в зависимости от выбора подтипов на этом же макете появляются доступные юридические лица. Пользователь может выбрать любой из них, запустив сценарий «Юридическое лицо. Профиль.»

3) в зависимости от выбора подтипов на этом же макете пользователь может добавить юридические лица.

4) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

В.1.11. Сценарий «Юридическое лицо. Профиль.»

1) Пользователь имеет возможность выбрать один из следующих элементов взаимодействия с юридическим лицом:

- Сделка, запустив сценарий «Контракты. Создание».
- Запрос информации, запустив сценарий «Запрос информации».

2) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

3) Пользователь может вернуться на страницу «Документы. Юридическое лицо» и запустить сценарий «Юридическое лицо».

В.1.12. Сценарий «Меню»

1) Пользователь имеет возможность выбрать один из следующих элементов для перехода:

- Документы, запустив сценарий «Документы. Главная страница».
- Кошельки.
- Активы, запустив сценарий «Активы. Главная страница».
- Контракты, запустив сценарий «Контракты».
- Голосования.
- Карты.
- Поиск.

2) Пользователь имеет возможность закрыть окно, оставив предыдущий сценарий активным.

В.2. Запрос информации

В.2.1. Сценарий «Запрос информации»

1) Пользователь работает с тем лицом, которого выбрал для запроса информации. Пользователь видит контрагента на макете.

2) Пользователь может выбрать тип информации:

- Документ
- Актив
- Отзыв

3) При выборе одного из типов информации необходимо выбрать подтип информации.

Для документа: ранее описанные подтипы.

Для активов: вещественные и невещественные.

Для отзывов: отзывы.

4) При выборе подтипа (для отзывов и активов), пользователь может выбрать конкретный подтип.

5) Пользователь имеет возможность добавить еще один пункт для запроса информации, нажав на кнопку «добавления» и вернувшись на 2 шаг данного сценария.

6) Пользователь имеет возможность выбрать кнопку «Запросить».

7) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

В.3. Смарт-контракты

В.3.1. Сценарий «Контракты».

1) Пользователь видит перед собой все текущие контракты, снизу панель с возможность выбрать:

- Создать контракт, сценарий «Контракты. Создать»
- Текущие
- История, сценарий «Контракты. История»

2) Из текущих пользователь может, в зависимости от состояния смарт-контракта:

- Согласовать условия.
- Выбрать контрагента.
- Посмотреть смарт-контракт.
- Оставить отзыв.

3) Пользователь имеет возможность просмотреть все текущие открытые незаконченные сделки.

4) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

В.3.2. Сценарий «Контракты. Создать».

1) Пользователь выбирает контрагента.

2) Пользователь может выбрать действие:

- Продать.
- Сдать.
- Подарить.
- Наследство.

3) Пользователь выбирает актив.

4) Пользователь имеет возможность сохранить контракт и заняться им позже, перейти на сценарий «Контракты», либо пригласить контрагента для обсуждений условий, запустив сценарий «Контракты. Условия».

5) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

6) Пользователь имеет возможность вернуться на страницу «Контракты. Текущие», запустив сценарий «Контракты».

В.3.3. Сценарий «Контракты. Условия».

1) Пользователь видит конртагента и имеет возможность перейти для общения в Telegram. Пользователь видит тип сделки. Пользователь имеет возможость просмотреть свои условия и условия контрагента. Пользователь имеет возможность добавить уловия, запустив сценарий «Контракты. Добавление условия».

2) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

3) Пользователь имеет возможность вернуться на страницу «Контракты. Текущие», запустив сценарий «Контракты».

В.3.4. Сценарий «Контракты. Условия».

1) Пользователь видит конртагента и имеет возможность перейти для общения в Telegram. Пользователь видит тип сделки. Пользователь имеет возможость просмотреть свои условия и условия контрагента. Пользователь

имеет возможность добавить уловия, запустив сценарий «Контракты. Добавление условия».

2) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

3) Пользователь имеет возможность вернуться на страницу «Контракты. Текущие», запустив сценарий «Контракты».

В.3.5. Сценарий «Контракты. Добавление условия».

1) Пользователь имеет возможность выбрать один из вариантов добавления условий:

- Стандартные, сценарий «Контракты. Условия. Стандартные».
- Из библиотеки, сценарий «Контракты. Условия. Из библиотеки».
- Написать самому.

2) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

3) Пользователь имеет возможность вернуться на страницу «Контракты. Текущие», запустив сценарий «Контракты».

В.3.6. Сценарий «Контракты. Добавление условия. Стандартные».

1) При выборе стандартных условий, пользователь имеет возможность выбрать один из предустановленных вариантов, ввести значение и добавить условие в список условий:

- Срок.
- Стоимость.
- Возврат.
- 3-й участник.
- Добавить актив.
- Верификация до срока.
- Указать процент покупки.

2) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

3) Пользователь имеет возможность вернуться на страницу «Контракты. Добавление условия», запустив сценарий «Контракты. Добавление условия».

В.3.7. Сценарий «Контракты. Добавление условия. Библиотека».

1) При выборе условий из библиотеки, пользователь имеет возможность выбрать один из документов с условиями, загруженных создателями системы или другими пользователями.

2) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

3) Пользователь имеет возможность вернуться на страницу «Контракты. Добавление условия», запустив сценарий «Контракты. Добавление условия».

В.4. Активы

В.4.1. Сценарий «Активы».

1) Пользователь имеет возможность выбрать принадлежность актива к:

- Физическое лицо.
- Юридическое лицо.

2) В зависимости от выбора на макете появляется выбор нужного лица принадлежность.

3) Пользователь имеет возможность выбрать тип актива:

- Вещественные, сценарий «Активы. Вещественные».
- Невещественные.
- Финансовые.

4) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

В.4.2. Сценарий «Активы. Вещественные».

1) Пользователь имеет возможность выбрать тип вещественного актива:

- Недвижимость, сценарий «Активы. Профиль недвижимость».
- Транспорт.

2) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

3) Пользователь имеет возможность вернуться на страницу «Активы», запустив сценарий «Активы».

В.4.3. Сценарий «Активы. Профиль недвижимость».

1) Пользователь имеет возможность просмотреть недвижимость, фотографии, жильцов, информацию, предыдущих владельцев, документы.

2) Пользователь имеет возможность открыть окно меню и выбрать другой интересующие его раздел, запустив сценарий «Меню».

3) Пользователь имеет возможность вернуться на страницу «Активы. Вещественные», запустив сценарий «Активы. Вещественные».