Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем
Кафедра системного программирования

Холодаева Екатерина

# Разработка приложения для безопасного хранения криптовалют

Курсовая работа

Научный руководитель:
ст. преп. Я. А. Кириленко.
Консультант:
Ф. П. Долголев

Санкт-Петербург, 2020 г.

# Содержание

**Введение** **2**

**1 Постановка задачи** **4**

**2 Обзор существующих решений** **5**

**3 Стек технологий** **7**

**4 Функциональность приложения** **9**

**5 Архитектура** **11**

**6 Реализация** **13**
6.1 Bitcoin . . . . . 13
6.1.1 UTXO . . . . . 13
6.1.2 Транзакции в Bitcoin . . . . . 13
6.1.3 Multisig адрес в Bitcoin . . . . . 14
6.1.4 BIP 174 . . . . . 15
6.2 Хранение данных . . . . . 16
6.3 Шифрование . . . . . 18

**7 Результаты** **20**
7.1 Результаты работы . . . . . 20
7.2 Тестирование . . . . . 21
7.3 Дальнейшие планы развития . . . . . 22
7.3.1 Подпись Шнорра . . . . . 22
7.3.2 Новые криптовалюты . . . . . 22

**Заключение** **23**

**8 Благодарности** **24**

# Введение

Блокчейн – это многофункциональная и многоуровневая информационная технология, предназначенная для учета различных активов. Технология надежного распределенного хранения записей обо всех когда-либо совершенных транзакциях. Блокчейн представляет собой цепочку блоков данных, объем которой постоянно растет по мере добавления новых блоков с записями самых последних транзакций. В отличие от обычных баз данных, изменить или удалить эти записи нельзя.

Если в привычных нам электронных платежных системах типа PayPal есть центральное связующее звено, то у криптовалют такого звена нет. Конечно, это дает определенные преимущества в виде исключения вмешательства третьей стороны в финансовые операции пользователя и полного контроля над своими средствами, но у этой медали есть и обратная сторона – безопасность.

Отсутствие какого-либо контроля и невозможность отмены транзакций перекладывает всю ответственность за сохранность собственных средств на пользователя. Никто не сможет что-либо исправить, если средства были отправлены мошенникам или электронный кошелек был украден. Поэтому при работе с криптовалютой безопасность выходит на первый план.

И самым безопасным методом хранения криптовалюты является холодное хранение.

Холодное хранение криптовалюты – это способ содержания цифровых монет, при котором приватный ключ, дающий доступ к деньгам, находится без возможности подключения к любым внешним устройствам (будь то Internet, Bluetooth, Wi-Fi). Данный метод идеально подходит для сбережения крупных сумм с точки зрения безопасности.

Другим способом увеличить безопасность криптокошелька является использование мультиподписи.

Мультиподпись – это конфигурация кошелька, для которой требуется как минимум два ключа для валидации транзакции.

Схема использования multisig адреса m из n подразумевает, что любые m ключей из n заранее установленных должны быть использованы для проверки m подписей, представленных в качестве подтверждения владения цифровыми монетами. То есть, чтобы потратить монеты, нужно предоставить m подписей, которые будут проверены m открытыми ключами из изначальных n. Они могут принадлежать как одному владельцу,

так и разным.

Безопасность с помощью мультиподписи достигается за счет того, что ключи, необходимые для подписи транзакции, хранятся на разных устройствах. Например, один может храниться компьютере, второй – на смартфоне, а третий а на листке бумаги в сейфе. Для того чтобы завладеть средствами, злоумышленник должен получить доступ не только к компьютеру, но и к другим устройствам или местам хранения, в которых находятся остальные приватные ключи.

Также примером может служить ситуация, когда приватные ключи принадлежат разным людям. И чтобы потратить какую-то сумму со счета они должны прийти к согласию, при этом они могут находиться в разных точках земного шара.

В данной работе описывается разработка приложения, поддерживающего холодное хранение криптовалюты и работу с multisig адресами.

# 1 Постановка задачи

Целью данного проекта является разработка мультивалютного кроссплатформенного приложения для безопасного хранения криптоактивов с возможностью холодного хранения и поддержкой multisig адресов.Для выполнения обозначенной цели были выделены следующие задачи:

1. исследовать предметную область;
2. провести обзор решений для хранения криптовалют;
3. выбрать технологии и средства для создания приложений;
4. разработать архитектуру приложения и UI;
5. осуществить поддержку нескольких криптовалют;
6. осуществить поддержку стандартных и multisig адресов.

# 2 Обзор существующих решений

На данный момент существует множество приложений для хранения криптовалют. Сравнение проводилось по следующим критериям:

1. возможность холодного хранения;
2. наличие multisig-поддержки;
3. поддерживаемые ПО;
4. поддерживаемые валюты.

Самые известные криптокошельки это: Electrum[1], Copay[2], BitGo[3], Mist[4], Exodus[5].

Результаты анализа приведены в таблице 1.

Наличие такого большого количества приложений на рынке показывает актуальность проблемы, но среди всех исследованных приложений есть только одно, которое решало бы все поставленные проблемы, но у него есть существенные недостатки:

1. приложение доступно не во всех странах;
2. обязательной является централизованная аутентификация;
3. комиссия с каждой исходящей транзакции – 0,1%.

Таблица 1: Анализ существующих решений

| Криптовалюты | Возможность холодного хранения | Наличие multisig-поддержки | Поддерживаемые ПО | Поддерживаемые валюты |
|---|---|---|---|---|
| Electrum | есть | есть | Linux, MacOS, Windows, Android, portable version | Bitcoin |
| Copay | нет | есть | iOS, Android, Windows Phone, Linux, Windows, MacOS, Chrome (plugin) | Bitcoin и Bitcoin Cash |
| BitGo | для ограниченной группы пользователей | есть | MacOS, Windows, Linux | более 100 криптовалют |
| Mist | есть | есть | Windows, Mac OS, Linux | Ethereum |
| Exodus | есть | нет | Windows(x64), Linux, MacOS, iOS | Более 95 криптовалют |

# 3 Стек технологий

В рамках данного проекта необходимо было создать приложение, поддерживаемое несколькими платформами, но создание нативных приложений более длительно и более затратно. Поэтому после изучения проблем выбора языков и технологий для разработки приложений и, исходя из требований к данному проекту, была выбрана кроссплатформенная разработка. В соответствии с этим, рассматривались наиболее популярные и распространенные фреймворки: React Native[6] и Quasar[7], Flutter[8]. Основными критериями выбора были:

1. поддерживаемые платформы;
2. совместимость с node.js;
3. производительность;
4. работа с графическими элементами.

Результаты анализа приведены в таблице 2.

После анализа преимуществ и недостатков вышеперечисленных фреймворков, был выбран Quasar, так как его существенным преимуществом является возможность создания универсального кода — как для веб приложения, так и для десктопного и мобильного приложений (под iOS и под Android устройства одновременно).

Таблица 2: Анализ фреймворков

| | Работа с графическими элементами | Поддерживаемые платформы | Совместимость с node.js | Произв-ть | Другие недостатки |
|---|---|---|---|---|---|
| React Native | использует реальные Android и iOS компоненты | iOS и Android | невозможно создавать браузерные версии библиотек, требующих node.js | высокая, близкая к нативной | |
| Quasar | есть компонент почти для каждой потребности в веб-разработке | SPAs, SSR, PWAs, BEX, Mobile Apps(Android, iOS) through Cordova or Capacitor, Multi-platform Desktop Apps (using Electron) | да | высокая | |
| Flutter | в составе виртуальной машины есть собственный графический движок, он рисует интерфейс приложения со всеми переходами между экранами, диалогами, фрагментами и т.д. | Android и iOS | использует Dart | очень высокая | в конечный установочный пакет добавляется виртуальная машина Dart |

# 4 Функциональность приложения

Основная цель при проектировании приложения состояла в том, чтобы сделать его наиболее безопасным, поэтому устройство, на котором будет установлено данное приложение, не будет иметь доступа в интернет (чтобы избежать получения злоумышленниками доступа к приватным ключам). А импорт и экспорт данных будет производиться посредством ручного ввода и с помощью QR-кодов.

Почему помимо ввода информации вручную использовались QR-коды?

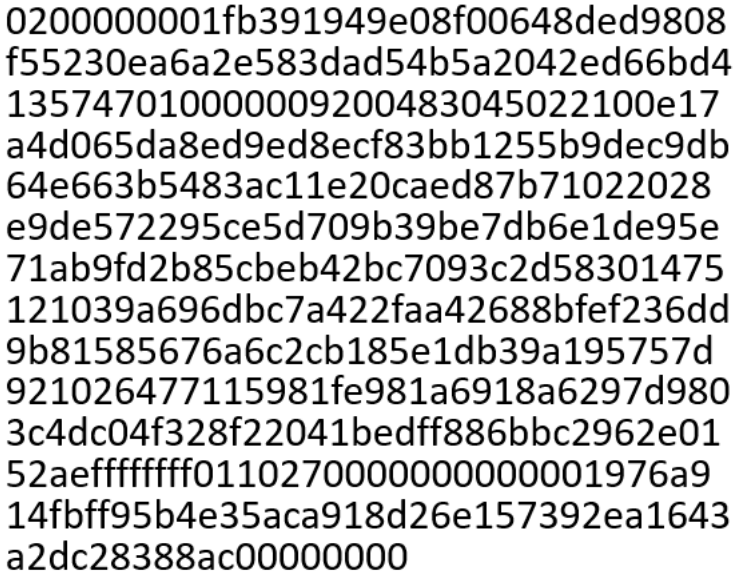


Рис. 1: Пример сырой транзакции.

На рисунке 1 представлен пример сырой транзакции. Данные довольно объемные для переписывания их вручную, а также велика вероятность совершить ошибку, потеряв при этом важную информацию. А так как камера есть почти на каждом устройстве, или же ее достаточно легко к нему подключить. использование QR-кодов является оптимальным решением данной проблемы.

Функциональность приложения:

1. поддержка множества валют;
2. хранение приватных ключей;
3. генерация стандартного адреса или адреса с multisig-поддержкой;

4. подписывание транзакции одним/несколькими ключами;
5. ввод/вывод информации с помощью QR-кода.

На рисунке 2 приведена диаграмма, отображающая UX-сценарии

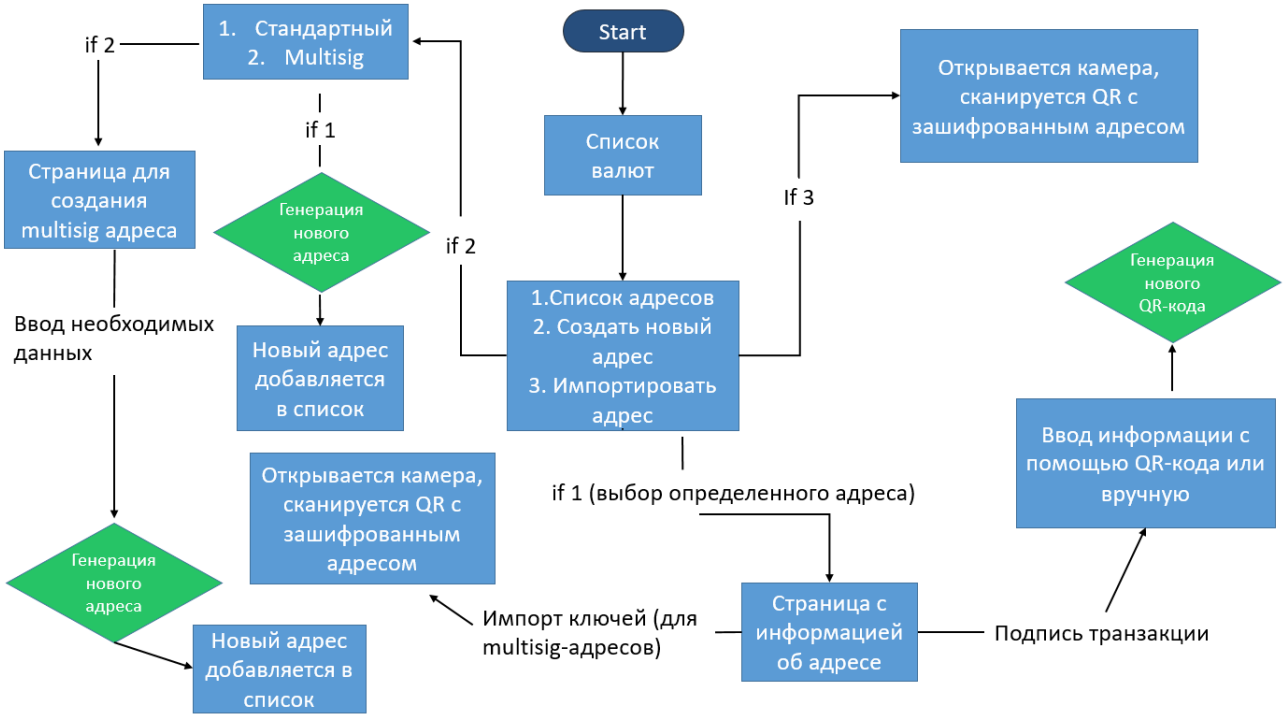


Рис. 2: UX-сценарии.

# 5 Архитектура

Работа с разными криптовалютами существенно различается, поэтому для каждой валюты было решено создать отдельный компонент, который включает в себя классы Transaction, Settings, CurrencyAddress и CurrencyMultisigAddress.

Для того, чтобы пользователь получил доступ к своим кошелькам,была создана форма для введения пароля, который также является и ключом для декодирования данных при извлечении их из хранилища.

При переходе на страницу, относящуюся к конкретной валюте, данные загружаются из класса DataStore, который отвечает за получения необходимой информации из места хранения, поэтому при дальнейшей разработке можно менять способ хранения без необходимости вносить серьезные изменения в сам проект.

На данном этапе хранение производится в базе данных SQLite.На рисунке 3 изображена ее ER-диаграмма.

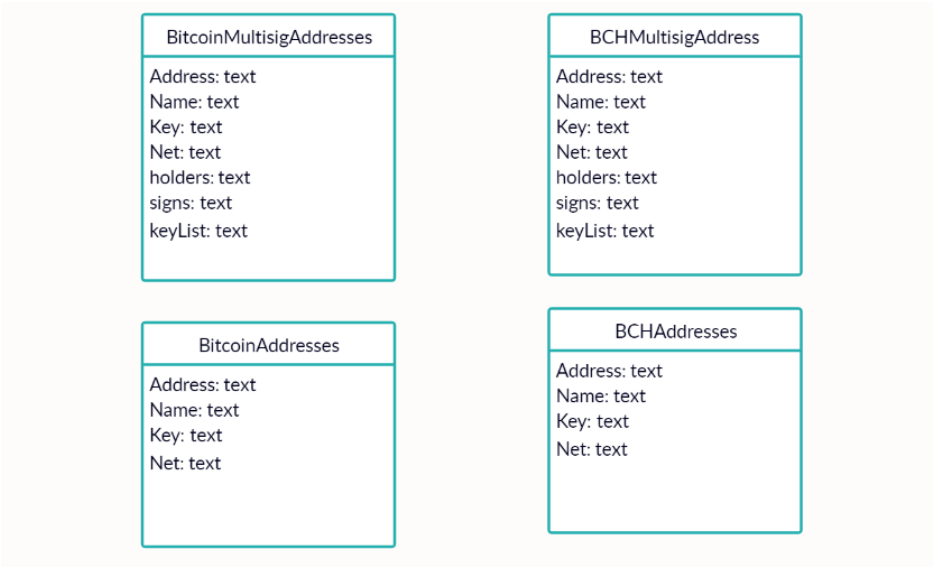


Рис. 3: ER-диаграмма базы данных.

Также были созданы страницы для стандартного и multisig адресов каждой валюты. На них отображаются публичный и приватный ключи, адрес, сеть, в формате которой сгенерирован адрес.

Для multisig адреса создана отдельная форма генерации, так как при его создании необхоимо ввести дополнительные параметры, такие как:

1. количество владельцев;
2. число подписей, необходимое для валидации;
3. сеть.

Транзакция подписывается в специальном классе Transaction, который изолирован от классов адресов. Это было сделано для возможности вызова методов для подписания транзакций из разных частей кода.

Создан класс Settings для настройки параметров для каждой валюты, например сеть, для которой нужно сгенерировать адрес, так как форматы ключей отличаются для разных сетей некоторых валют.

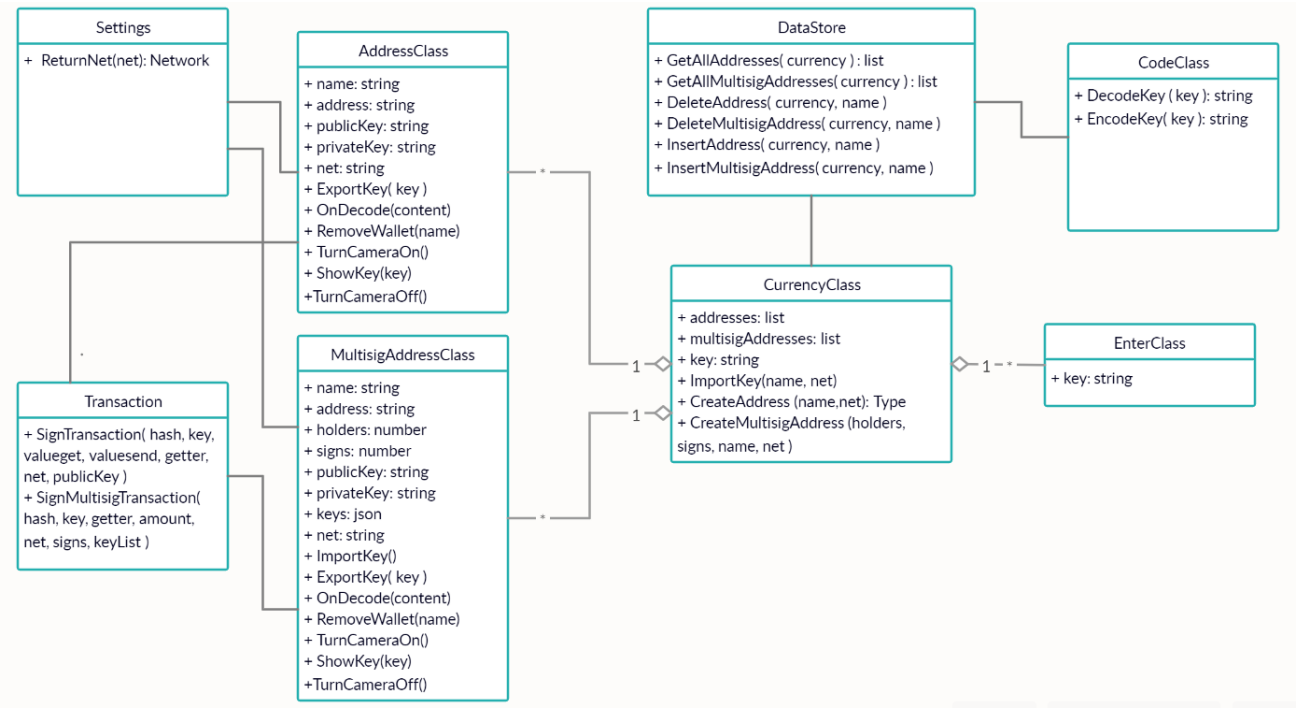


Рис. 4: UML-диаграмма классов

# 6 Реализация

## 6.1 Bitcoin

Биткойн – это первая в мире децентрализованная цифровая криптовалюта. Сеть Bitcoin появилась в 2009 году, она работает за счет майнинга в результате которого и добываются монеты. Транзакции в сети Bitcoin происходят по принципу peer-to-peer, что означает прямую передачу средств от одного пользователя другому, без участия третьих лиц.

### 6.1.1 UTXO

UTXO (Unspent Transaction Output) – неизрасходованные выходы в сети Bitcoin. При совершении транзакции выходы предыдущих транзакций расходуются и создаются новые, которые будут использованы в будущих транзакциях. Каждый из таких выходов может расходоваться только один раз, а конечный баланс в биткоин-кошельке будет складываться из всех оставшихся UTXO.

### 6.1.2 Транзакции в Bitcoin

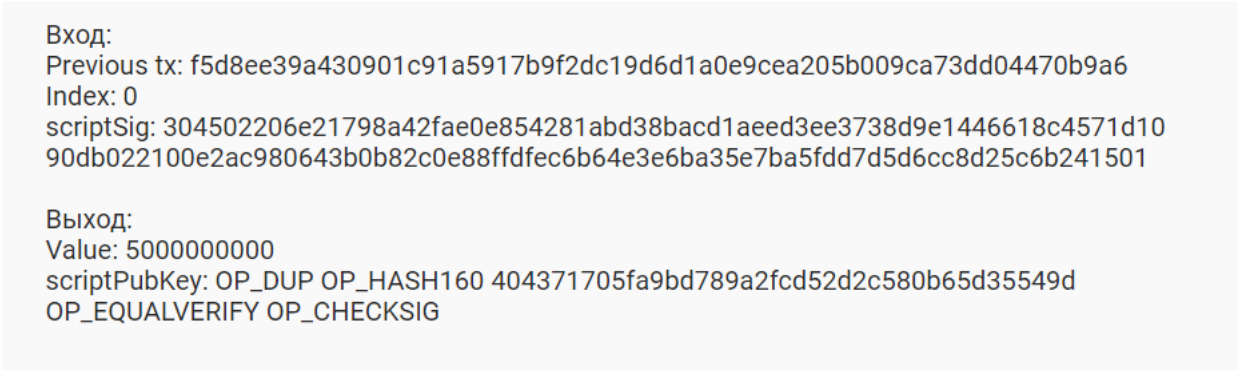


Рис. 5: Пример транзакции биткойна.

Транзакция в сети Bitcoin состоит из входов и выходов. На рисунке 5 приведен пример транзакции с одним входом и одним выходом.

Вход – это ссылка на выход другой транзакции. Часто в транзакции может быть несколько входов, в таком случае значения всех выходов предыдущих сделок суммируются и общая сумма записывается на вход текущей транзакции.

Previous tx – хэш предыдущей транзакции.

Index – выход транзакции, на которую ссылаются.

ScriptSig – подпись( первая половина Скрипта).

Скрипт состоит из двух частей: подписи(ScriptSig) и открытого ключа(scriptPubKey). Подпись (ScriptSig) получена из хэша транзакции по алгоритму ECDSA(алгоритм с открытым ключом для создания цифровой подписи). Второй компонент – открытый ключ, который принадлежит инициатору транзакции и подтверждает, что он обладает суммой, необходимой для выполнения транзакции. Вместе они подтверждают, что транзакция была создана конкретным владельцем Bitcoin-адреса.

Выход

Выход содержит инструкции на перевод биткоинов.

Value – количество Сатоши( 1 биткоин = 100000000 Сатоши) участвующих в данной транзакции.

ScriptPubKey – это вторая половина Скрипта. Транзакция также может содержать больше одного выхода. Если на выходах транзакции обрабатывается не вся сумма биткоинов, указанная на входе, то остаток считается комиссией за транзакцию. А именно, эти биткоины получит майнер, в чей блок будет включена запись о данной транзакции.

Подтверждение

Для подтверждения транзакции ScriptSig на входе и ссылающийся на него ScriptPubKey на выходе оцениваются. Если проверка происходит корректно, то результат проверки — true.

### 6.1.3 Multisig адрес в Bitcoin

Multisignature address (multisig address, если сокращенно) — это такой Биткоин адрес, к которому привязано сразу несколько пар ECDSA ключей. Каждая пара состоит из личного и открытого ключей. Схемы комбинаций, согласно которым можно использовать эти ключи, могут быть различными.  На схеме 6 изображена транзакция, которая тратит криптовалюту с multisig адреса. В поле scriptSig входа перечислено два открытых

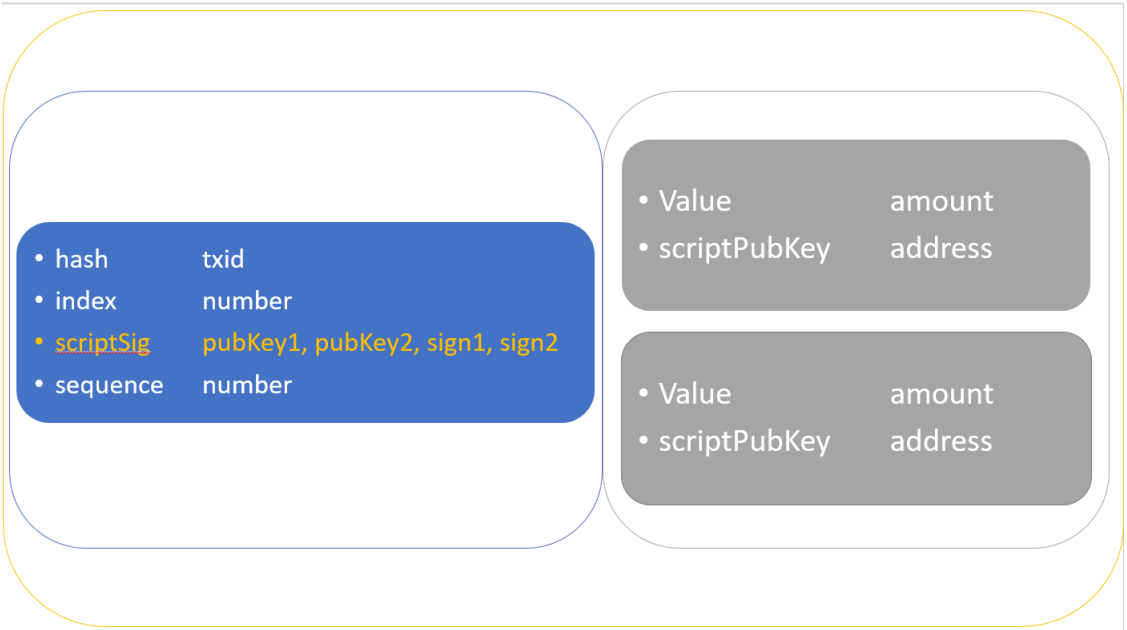


Рис. 6: Схема транзакции, которая тратит монеты с multisig адреса.

ключа и две подписи. Они должны проверяться этими открытыми ключами соответственно. Это и есть тот вход транзакции, который тратит монеты с multisig адреса. Именно так будет выглядеть доказательство владения активами.

Для упрощения обмена данными для работы с multisig адресами было предложен проект BIP 174.

### 6.1.4 BIP 174

BIP 174 – предложение по модернизации сети Bitcoin, которое только недавно было добавлено в репозиторий проекта [9]. Данная разработка позволит Blockchain поддерживать PSBT(Partially Signed Bitcoin Transactions) – так называемые частично подписанные биткоин-транзакции, то есть формат данных, который позволяет кошелькам и другим инструментам обмениваться информацией о транзакции и подписями, необходимыми для ее валидации. Он предназначен для упрощения рабочих процессов, когда нескольким сторонам необходимо сотрудничать для осуществления транзакции.

В целом, построение полностью подписанной биткойн-транзакции проходит следующие этапы:

1. Creator создает PSBT, который содержит определенные входы и выходы, но не содержит дополнительных метаданных;

2. для каждого входа Updater добавляет информацию об UTXO, используемых тран-

закцией, в PSBT. Также добавляется информация о скриптах и открытых ключах, участвующих в каждом из входов (и, возможно, выходов) PSBT;

3. Signers проверяют транзакцию и ее метаданные. Если они согласны с транзакцией, то каждый из них производит частичную подпись для входных данных, для которых у него есть соответствующие ключи;

4. Finalizer запускается для каждого ввода, чтобы преобразовать частичные подписи и, возможно, информацию скрипта в финальные scriptSig и/или scriptWitness;

5. Extractor создает из PSBT, для которого все входные данные подтверждены, окончательную биткоин-транзакцию.

Где Creator, Updater, Signers, Finalizer, Extractor – названия ролей, определенных в BIP 174. Но на практике программные и аппаратные реализации обычно исполняют несколько ролей одновременно.

Как правило, каждый из вышеперечисленных (исключая Creator и Extractor) будет просто добавлять все больше и больше данных в конкретный PSBT, пока все входные данные не окажутся полностью подписанными. PSBT последовательно проходит все этапы и в конечном итоге Extractor преобразует его в окончательную валидную транзакцию.

## 6.2 Хранение данных

Чтобы пользователю не было необходимости каждый раз заново импортировать в приложение свои ключи, все важные сведения хранятся на устройстве, на котором установлено приложение.

При разработке настольного приложения чаще всего используются 2 способа хранения данных:

1. файлы;

2. базы данных.

Файловые системы обычно применяются для хранение слабо структурированной информации, базы данных же решают ряд проблем, связанных с файлами:

1. так как данные хранятся в разных файлах, то чтобы собрать информацию из них приходится организовывать синхронную обработку сразу нескольких файлов;
2. из-за хранения данных в разных файлах, часто приходится повторять некоторую информацию, чтобы не потерять смысл;
3. в файловых системах внесения изменений в данные или в структуру требует значительных усилий.

При разработке мобильного приложения также важно выбрать правильный механизм для локального хранения данных. Метод должен быть безопасным и не замедлять работу приложения. Наиболее часто используемыми способами хранения данных являются [10]:

1. Shared Preferences(для Android) и UserDefaults(для iPhone);
2. обычные файлы;
3. базы данных SQLite.

Каждый из способов имеет свои недостатки и преимущества, что обусловлено их предназначением.

Первый способ представляет собой хранение данных в виде ”ключ-значение”в XML файле. Он используется преимущественно для хранения настроек и не подходит для хранения структурированных данных.

Второй способ ипользуется для хранения больших объемов данных и является наиболее оптимальным для хранения изображений медиа и других файлов, чтение которых осуществляется без пропусков. Почти каждое устройство имеет 2 области памяти: внутреннюю и внешнюю.То, где будет хранится файл в каждом конкретном случае определяется целью проекта. Например, хранение во внутренней области более безопасно, так как не допускает изменений файлов пользователем и другими приложениями, но не трубе разрешения на запись, файлы же хранящиеся во внешнем хранилище доступны везде, но чтобы произвести запись во внешнюю память требуется разрешение.

Последний из вышеназванных способов используется для хранения больших объемов повторяющихся и сложноструктурированных данных, поэтому идеален в рамках поставленной задачи. База данных считается частью приложения, которое ее создало, поэтому упрощается работа с данными и нивелируются задержки при выполнении

транзакции. Также доступ к базе данных есть только у приложения, поэтому хранение информации в базе безопаснее.

Так как объемы информации невелики, на данном этапе проектирования приложения не уделялось много внимания выбору способа хранения данных.В рамках проекта был использован плагин cordova-sqlite-storage [11]. Его достоинства:

1. можно использовать на Android, IOS, Windows;
2. удобный инструментарий;
3. достаточно высокая производительность.

## 6.3 Шифрование

Первостепенное значение при использовании приложения имеет безопасность, поэтому данные при добавлении в базу шифруются.

Анализируемыми алгоритмами шифрования были:

1. криптография на эллиптических кривых(ECDSA и ECDH);
2. RSA.

Выбор был сделан в пользу ECDSA, так как у алгоритмов асимметричной криптографии на эллиптических кривых в сравнении с RSA есть следующие преимущества:

1. гораздо меньше длина ключа по сравнению к «классической» асимметричной криптографией. И, следовательно, пользователю будет проще запомнить код;
2. скорость работы эллиптических алгоритмов гораздо выше, чем у классических. Так происходит потому, что элементы поля $GF(2^m)$ могут быть легко представлены в виде n-битных кодовых слов;
3. из-за небольшой длины ключа и высокой скорости работы, алгоритмы асимметричной криптографии на эллиптических кривых могут использоваться в устройствах с ограниченными вычислительными ресурсами. А в контексте рассматриваемой задачи это условие является важным, так как приложение должно корректно работать и на устройствах с низкой производительностью.

Также алгоритм ECDSA используется Bitcoin для гарантии того, что средства могут быть потрачены только их законными владельцами.

# 7 Результаты

## 7.1 Результаты работы

В результате курсовой работы было получено приложение, поддерживающее холодное хранение валют Bitcoin, Bitcoin Cash. На рисунке 7 изображен экран с доступными пользователю кошельками.

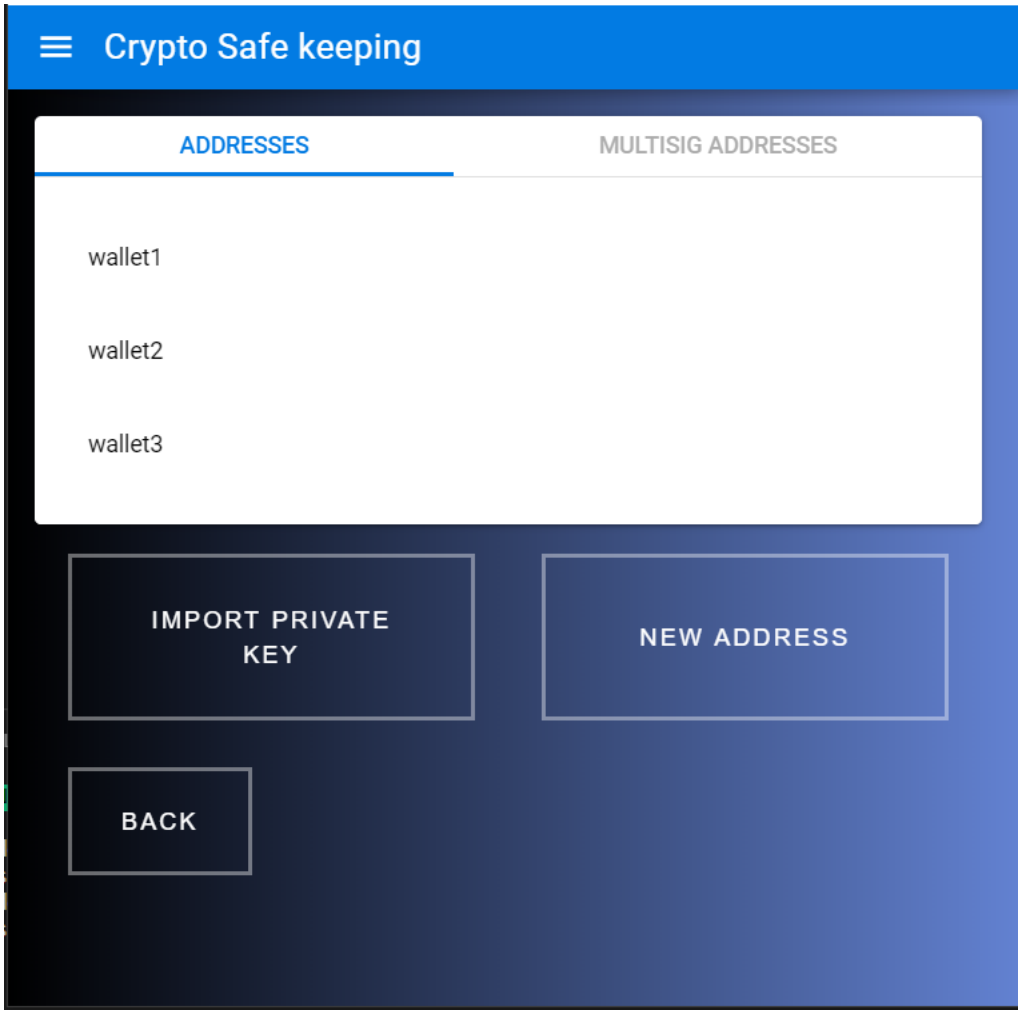


Рис. 7: Экран с доступными пользователю кошельками.

А также работу как со стандартными, так и с multisig адресами для вышеперечисленных валют. На рисунке 8 изображен экран генерации нового multisig адреса для Bitcoin.

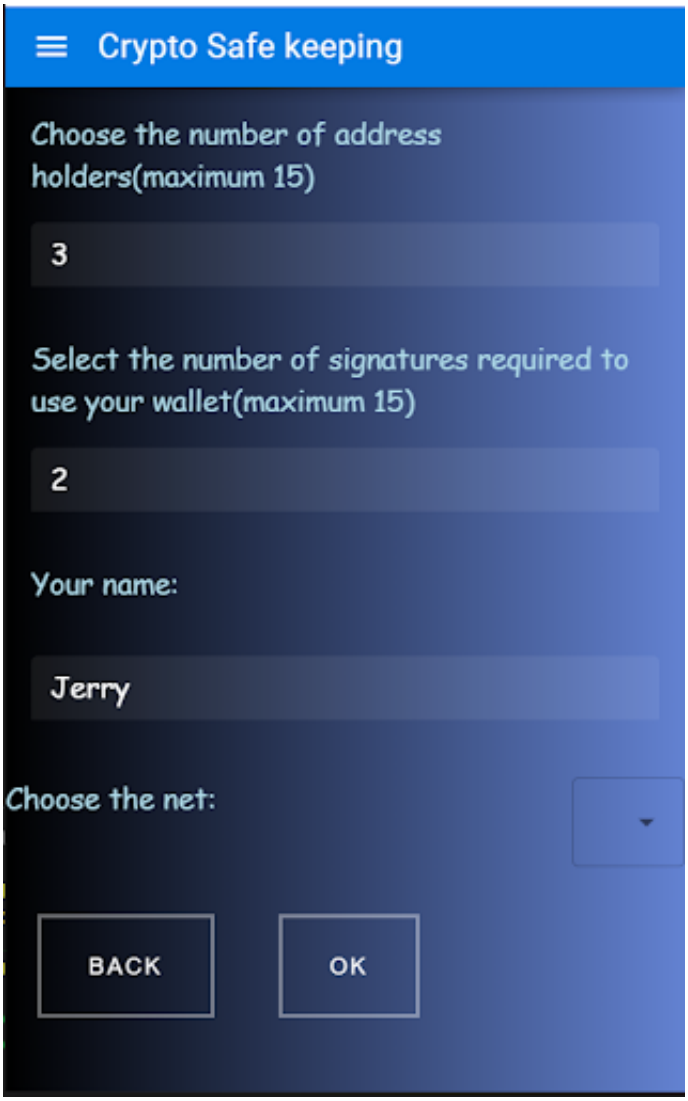


Рис. 8: Экран генерации нового multisig адреса для Bitcoin.

## 7.2 Тестирование

Для удобства тестирования и понимания принципов работы приложения были использованы тестнеты [12] – инструмент для разработчиков, позволяющий имитировать работу с настоящими монетами. Для этого создавались адреса в специальном формате, а при формировании транзакций тестовая сеть указывалась в качестве параметра. Далее подписанные транзакции отправлялись в тестнет. Для модульного тестирования использовался фреймворк Jest [13].

## 7.3 Дальнейшие планы развития

### 7.3.1 Подпись Шнорра

Технология Шнорра – одно из многих возможных направлений развития блокчейна Bitcoin.

В настоящее время для каждой транзакции используется своя цифровая подпись, занимающая много места на блокчейне. В результате увеличиваются требования к размеру транзакции и возрастает комиссия за нее.

Технология мультиподписей Шнорра предлагает следующее решение – систематизировать цифровые подписи и объединить ключи. В результате вместо действующей схемы генерации индивидуальных подписей к каждой новой транзакции подписанты могут использовать только одну. Эта единственная подпись может подтверждаться кем угодно, кто, разумеется, знает сообщение и открытые ключи всех участников. Одним из возможных направлений развития является внедрение технологии Шнорра в приложение.

### 7.3.2 Новые криптовалюты

Другим возможным вариантом развития является поддержка других криптовалют, например, Litecoin.

# Заключение

В рамках данного проекта мною были выполнены следующие задачи:

1. исследована предметная область и сделан обзор уже существующих решений;
2. выбран стек технологий;
3. разработана функциональность приложения;
4. реализована поддержка стандартных и multisig адресов для Bitcoin;
5. реализована поддержка стандартных и multisig адресов для Bitcoin Cash;
6. разработан прототип UI;

Ссылка на репозиторий, где ведется разработка:
https://github.com/dsx-tech/CryptoSafekeeping

# 8 Благодарности

Хочу выразить благодарность компании DSX Technologies и, в частности, руководившему моей работой, Долголеву Филиппу Петровичу за оказание помощи в освоении предметной области и решении возникавших проблем с реализацией проекта.

# Список литературы


[1] Electrum. Bitcoin wallet. 2011. URL: https://electrum.org/home.

[2] Copay. Bitcoin and Bitcoin Cash wallet. 2014. URL: https://Copay.Io/.

[3] BitGo. Digital asset trust company and security company. 2013. URL: https://www.bitgo.com/.

[4] Mist. Ethereum wallet. 2015. URL: https://github.com/ethereum/mist/releases.

[5] Exodus. Beautiful Wallet. 2015. URL: https://www.exodus.io/.

[6] React Native. React Native. 2015. URL: https://facebook.github.io/react-native/.

[7] Quasar. Quasar Framework. 2018. URL: https://quasar.dev/.

[8] Flutter. Flutter. 2015. URL: https://https://flutter.dev//.

[9] Bitcoin. Bitcoin. 2009. URL: https://github.com/bitcoin/bitcoin.

[10] Сидора А. А. Способы хранения данных в приложениях Android OS // Программные средства и информационные технологии. 2015. С. 248–250.

[11] cordova-sqlite-storage. cordova-sqlite-storage. 2016. URL: https://github.com/xpbrew/cordova-sqlite-storage.

[12] Bitcoin Testnet Explorer. Bitcoin Testnet Explorer. 2013. URL: https://live.blockcypher.com/btc-testnet/.

[13] Jest. Delightful JavaScript Testing Framework. 2016. URL: https://jestjs.io/.