МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)
02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)
Технология программирования
(направленность (профиль)/специализация)

# БАКАЛАВРСКАЯ РАБОТА

на тему «Разработка системы электронного голосования на базе технологии Blockchain»

| | | |
|---|---|---|
| Студент | И.А. Зязев | |
| | (И.О. Фамилия) | (личная подпись) |
| Руководитель | А.В. Очеповский | |
| | (И.О. Фамилия) | (личная подпись) |
| Консультанты | А.В. Москалюк | |
| | (И.О. Фамилия) | (личная подпись) |

**Допустить к защите**

Заведующий кафедрой к.т.н, доцент А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____»_____20_____г.

Тольятти 2018

**Аннотация**

Тема выпускной квалификационной работы «Разработка системы электронного голосования на основе технологии blockchain».

Объектом исследования является технология blockchain.

Предметом данной работы является децентрализованная система голосования.

В ходе работы были выполнены задачи: рассмотрение вопросов о разработке децентрализованных приложений; проводился анализ blockchain-платформ; проектировалось, разрабатывалось и тестировалось децентрализованное приложение для интернет-голосований.

В теоретической части рассматривается вопрос о разработке децентрализованных приложений на базе технологии blockchain. Так же, проводится сравнение между различными платформами blockchain. Итогом этого сравнения является то, что наиболее полно сформулированным требованиям соответствует платформа Ethereum. Практическая часть посвящена проектировке и разработке системы электронного голосования на основе блокчейн-платформы Ethereum.

В результате проделанной работы реализовано децентрализованное приложение на базе платформы Ethereum, в котором пользователи могут выражать свое волеизъявление по тому или иному вопросу, в целях совместного поиска удовлетворяющего всех участников системы подходящего решения.

Данная работа состоит из 60 страницах, включая в себя введение, три главы, заключение, список используемой литературы из 41 источников, 26 рисунков, 1 таблицы и двух приложений.

## ABSTRACT

The topic of the graduation work is development of an electronic voting system based on blockchain.

The object of the research is blockchain technology.

The subject of this work is an electronic voting system.

In the course of the work the following tasks are completed: examination of issues concerning development of decentralized application, analysis of blockchain-based distributed computing platforms, development and testing of the decentralized application for e-voting.

The theoretical part deals with the development of decentralized application based on blockchain technology. We also conduct a comparative analysis of different blockchain-based distributed computing platforms. We come to a conclusion that Ethereum fully meets the requirements formulated. The practical part is devoted to the modeling and developing the electronic voting system based on blockchain-based distributed computing platform Ethereum. Ethereum, a digital currency system based on the cryptography, is highly open and transparent for the individual transaction. In other words, anyone can access to the transaction contents via blockchain. Besides, regarding to anonymous way it trades, the transaction of Ethereum is untraceable.

As a result of the work done, a decentralized application based on Ethereum, an application that allows users to make a decision or express an opinion usually following discussions or debates is developed.

This work consists of 60 pages, including the introduction, three chapters, conclusion, a list of 41 references and 2 applications.

**ОГЛАВЛЕНИЕ**

# ВВЕДЕНИЕ

Информационные технологии со временем все теснее переплетаются с человеческой жизнью. Переход из реального мира в цифровое пространство обычно влечет за собой огромное количество положительных моментов.

Голосование - основа любой успешной демократии и, поэтому, оно должно быть доступным и безопасным для всех людей. Сегодняшние наиболее распространенные бумажные системы голосования доступные и дешевые, но имеют две основных проблемы. Согласно многим экспертам, такие системы не масштабируемы (поэтому, это приводит к таким основным проблемам, как точность), и они подразумевают “уверенность в процедурной безопасности организаторов, проводящих их правильно и честно” [7].

Нередко в новостях описываются события о взломе той или иной информационной системе, которые позволяют злоумышленникам получить несанкционированный доступ к конфиденциальной информации [13]. Актуальность работы заключается в том, что данные риски можно минимизировать благодаря стремительному прогрессу криптографии, в том числе благодаря развитию технологии blockchain. Blockchain мог бы предложить повсеместное масштабируемое решение текущих и устаревших избирательных методов, обеспечив безопасное и защищенное от фальсификаций цифровое голосование.

Новизна исследования состоит в том, что многие сложные вопросы безопасности, с которыми сегодня сталкиваются электронные системы голосования, могут быть преодолены, если в их разработке применить механизмы репликации, криптографии и верификации, которые использует технология blockchain. Применение данной технологии в области голосований должно положительно сказаться на безопасности и прозрачности таких систем, а, следовательно, и на доверии пользователей к ним.

Объект исследования: технология блокчейн.

Предмет исследования: система электронного голосования.

Цель работы: разработать систему электронного голосования на основе технологии блокчейн. Для достижения цели данной выпускной квалификационной работы необходимо решить следующие задачи:

1. Проанализировать проблемы существующих решений электронного голосования и изучить теоретическую часть, касающуюся технологии blockchain.
2. Исследовать существующие платформы blockchain и средства разработки децентрализованных приложения.
3. Разработать децентрализованное приложение, решающее проблемы современных систем голосования.

В первой главе проведен анализ информации: описание предметной области, обзор технологии blockchain, выполнен обзор криптографических протоколов и электронных систем голосования.

Во второй главе проведен анализ существующих блочкчейн-платформ и выполнено проектирование приложения, что позволит реализовать поставленную задачу.

Третья глава содержит разработку децентрализованной системы голосования.

В заключении формулируются выводы, полученные в процессе выполнения ВКР.

# Глава 1 АНАЛИЗ ИНФОРМАЦИИ И ПОСТАНОВКА ЗАДАЧИ

## 1.1 Бумажное и электронное голосование

Голосование – это способ принятия решений группой людей, в котором наиболее общее мнение выбирается путем подсчета волеизъявлений каждого из избирателей [1].

Голосование может быть конфиденциальным и открытым. В открытом голосовании, каждый участник знает, кто и как распорядился своим голосом. При конфиденциальном голосовании, напротив, принимаются меры, чтобы информация о волеизъявлении каждого из проголосовавших людей была недоступна третьим лицам. В таком голосовании повышается вероятность исключения возможности подкупа или давления на избирателя. Однако, в тайных голосованиях невозможно признать данную процедуру полностью честной, т.к. такой способ выборов нельзя верифицировать, если они масштабные. Поэтому, нередко на подобного рода процедурах тайного бумажного голосования можно встретить различные манипуляции, которые могут привести к дискредитированию результатов голосования.

Электронное голосование – процесс принятия решения, путём обычного голосования, с применением специальных электронных средств голосования и технических электронных средств для подсчета голосов и оглашения результата [1]. Одной из разновидности такой процедуры голосования являются выборы через сеть интернет. Технология электронных голосований позволяет сократить время, которые требуется для подсчета голосов, а также облегчить процедуру голосования людям, которым не нужно приходить на избирательный участок.

Голосование играет важную роль в построении демократического общества. Традиционное голосование требует, чтобы избиратели присутствовали при процедуре голосования, что обычно влечет за собой трату времени людей, которые участвуют в выборах, а также данная процедура требует больших финансовых вложений, если речь о выборах государственного масштаба.

Электронное голосование - это новая концепция онлайн-выборов, основанная на криптографии. Система поддерживает полнофункциональное онлайн-голосование на любых устройствах, а результаты опроса будут рассчитываться автоматически и анонимно. По сравнению с традиционным голосованием, электронное голосование – это более экономичная система, более прозрачная и беспристрастная.

Система электронного голосования в основном использует интернет-платформу. Важнейшая проблема для электронного голосования — это риски безопасности, которые могут возникнуть. В целях снижения рисков, за последние десятки лет было предложено множество различных протоколов связанных с конфиденциальностью бюллетеней, индивидуальной проверкой, надежностью, доступностью и т.д. Кроме того, в представленных протоколах реализованы различные технологии, такие как «слепая подпись», «кольцевая подпись», «гомоморфное шифрование», «Mix networks», «доказательство с нулевым разглашением» и т.п [6]. С появлением криптовалюты, шифрование, а, следовательно, и интернет-голосованием стало более совершенным.

Таблица 1 – Сравнение электронного и бумажного голосования

| | Бумажное голосование | Электронное голосование |
|---|---|---|
| Высокая скорость обработки бюллетеней | - | + |
| Экономия времени, при заполнении бюллетеней | - | + |
| Отображение хода голосования в реальном времени | - | + |
| Эффективная масштабируемость | - | + |

Не смотря на положительные стороны, у электронного голосования, по некоторым оценкам, еще больше рисков фальсификаций и

компрометирования результатов, чем у обычного, т.к. система может быть взломана третьими лицами, а подсчет голосов производится на машине, доступ к которой есть у администраторов, а значит, что они тоже могут иметь доступ к результатам и изменять их по своему усмотрению [15, 18, 17, 11].

### 1.1.1 Первые системы электронного голосования

В течение долгого времени многие исследователя посвящали себя разработке протокола для безопасного и эффективного голосования. Одно из первых положений, связанных с криптографическим протоколом электронного голосования, было опубликовано Дэвидом Чаумом в 1981 году [22]. Он использовал анонимный канал коммутации для шифрования бюллетеня. С развитием криптографии было предложено множество протоколов со своими собственными свойствами. В 1982 году Ричард Де Милло также предложил протокол, который требует, чтобы все избиратели участвовали [26]. Каждая бюллетень зашифровывалась и в конце голоса подсчитывались. В 1985 году Коэн Дж. и Фишер М. предложили похожий криптографический протокол, который может проводить избирательные выборы в форме голосования. Однако, он требовал, чтобы все избиратели голосовали одновременно [24]. Протокол шифровал бы бюллетень используя теорему гомоморфизма, а правительство в конце публиковало бы итоговый результат.

В 1992 году Фудзиока, Окамото и Охта предложили практическую секретную схему электронного голосования(FOO), она используется для проведения масштабных выборов, которые могут обеспечить конфиденциальность избирателей и справедливость голосования. Эта схема использовала слепую подпись для слепого сообщения, которое избиратель отправлял администратору [28]. После выхода этой основополагающей работы многие разработчики использовали ее для обеспечения электронного голосования в своем программном обеспечении, например, EVOX и SENSUS. В данном случае, такая система тоже имеет свои слабости, она

требует, чтобы в процессе голосования участвовали абсолютно все избиратели, и, если кто-то воздерживается от голосования, то в результат можно было вмешаться. Также, система была построена таким образом, что администратор не мог узнать, кто именно вмешался в результат. В 1996 году Хуанг и Лэй предложили голосовать на основе слепой подписи, но также требуют, чтобы все присутствовали на мероприятии, посвященным голосованию.

Через 3 года, М. Окубо, Ф. Миура и М. Абе продвигали схему FOO, используя пороговый протокол шифрования и канал связи Mix-Network, который может поддерживать конфиденциальность избирателя. Избирателям необязательно участвовать в подсчете голосов, и они могут уйти сразу после голосования [37].

С развитием системы электронного голосования появилось множество способов фальсификаций и дискредитации выборов. Можно воздействовать на результаты с помощью угроз избирателю, либо покупать его голос. Чтобы справиться с этой проблемой, были предложены многие новые требования для участия в электронном голосовании, например, такие как «receipt-freeness» («освобождение от квитанций») и «coercion-resistance» («устойчивость к принуждению»).

Receipt-freeness означает, что после процесса голосования, избиратель не имеет никакой информации, например, в виде квитанции, которая может быть использована в качестве подтверждения того, что он проголосовал тем или иным образом. Термин «receipt-freeness» предложил Джош Бенало в 1994 году [20]. Хотя Бенало использовал гомоморфное шифрование, чтобы реализовать receipt-freeness, Мартин Гирт утверждал, что это не будет работать должным образом, если в выборах будет более одного органа, подсчитывающего голоса [20]. В 1995 году В. Ниэми и А. Ренвалл разработали схему, адоптировав квитанцию о голосовании так, что избиратель не мог доказать за кого или за что он отдал свой голос. В то же время К. Сако и Дж. Килиан предложили первый протокол на основе Mix-

Net, который должен был удовлетворять требованиям receipt-freeness [40]. Но этот протокол основан на предположении, что между избирательным участком и избирателем нет частного канала. Поэтому, этот протокол показал, что он всё-таки не удовлетворяет требованиям receipt-freeness в полной мере. В 2001 году О. Бодрон предложил новую схему, удовлетворяющую этим свойствам, используя криптосистему Пайе и протокол Zero-knowledge proof [30].

В последние годы многие исследователи сосредотачиваются на receipt-freeness и coercion-resistance в сфере электронных голосований. В 2010 году А. Джулс представил новое направление в требованиях к электронным голосованиям. Оно имеет название «Coercion-Resistant Electronic Elections»(«Электронные выборы с сопротивлением к принуждению»). В 2012 году О. Шпихер и Р. Кениг продвигают схему Джулса с добавлением к ней случайного целого числа f. Предлагаемая схема получит зашифрованное целое число C. Орган подсчета голосов может судить о наличии поддельного бюллетеня путём дешифровки C с помощью f [32]. Таким образом, устойчивость к принуждающим воздействиям на избирателя было удовлетворено.

Однако, даже такое голосование, в котором предусмотрено большое количество факторов, в которых нуждается процедура честного голосования, не может сделать его достаточно прозрачным, чтобы избиратели могли доверять системам электронного голосования. [17, 5, 6].

### 1.1.2 Проведенные масштабные электронные голосования и их критика

В настоящее время системы электронного голосования широко используются. Ниже приведены самые известные случаи применения таких систем.

В 2000 году электронное голосование использовалось в выборах президента США. Хотя этот эксперимент проводился только в некоторых

районах штата Флорида, это стало важным событием в развитии электронного голосования [9].

В 2002 году в Соединенном Королевстве была опробована система электронного голосования. Более, чем 18 государственных органов были награждены за создание системы этой системы [35].

В 2004 году выборы в США впервые использовали систему электронного голосования DRE. Индия использует эту систему для парламентских выборов в национальном масштабе [5].

В 2007 году во Франции политическая партия UMP вошла в историю, проведя интернет-голосование. Более, чем 31 000 избирателей проголосовали на выборах президента в UMP в 2007 году. Это было первое массовое электронное голосование в истории [27].

В 2014 году на выборах в Министерство Национального образования во Франции было получено 1 760 000 бюллетеней [33]. Эти выборы взяли на себя ведущую роль в юридическом и защищенном сетевом голосовании, тем самым популяризировали системы электронного голосования.

Несмотря на то, что системы электронных голосований во многом удобнее, эффективнее и экономичнее(Таблица 1) традиционных систем, во многих из этих случаев применения электронного голосования возникали некоторые проблемы и сбои в работе, которые мешали общему процессу выборов [9, 11, 15, 18, 8].

То есть, зачастую, в сбоях таких систем проблемой являлось то, что машина, которая обрабатывает голоса выводилась из строя, что приводило к нарушению процесса голосования, например, целого штата [9].

Кроме того, в настоящее время ведутся споры о том, что электронное голосование может быть подвержено большим нарушениям, чем при аналоговом, т.к. система может быть взломана третьими лицами, а подсчет голосов производится на машине, доступ к которой имеется у

администраторов, а значит, что они тоже могут иметь возможность манипулировать результатами голосования [17, 5, 6].

## 1.2 Технология blockchain

Проанализировав критику существующих решений систем электронного голосования, можно сформулировать следующие недостатки таких систем и некоторые стороны, которые можно улучшить:

- недостаточная прозрачность процесса голосования;
- недостаточная отказоустойчивость системы;
- повысить устойчивость к взлому.

Исходя из общих требований безопасности и отказоустойчивости, всё чаще предлагается протокол на основе технологии Blockchain, который должен положительно сказаться сформулированных выше недостатках.

Технология, известная, как «блокчейн» была впервые представлена Сатоши Накомото в его статье “Bitcoin: A Peer-to-Peer Electronic Cash System”, в которой изложена математическая основа для криптовалюты «Bitcoin» [12]. Хотя это была новаторская статья, она никогда не представлялась в традиционном рецензируемом журнале, а подлинная личность автора неизвестна. Технология блокчейн не только лежит в основе всех криптовалют, но и находит широкое применение в более традиционной финансовой индустрии. Она также открыла двери для новых приложений, таких как «смарт-контракты».

Проблема, которую Накомото разрешил с помощью цепочки блоков, заключалась в установлении доверия к децентрализованной(распределенной) системе. Конкретнее, проблема создания распределенного хранилища временных файлов, в то время, как ни одна из сторон не может вмешиваться в содержание данных или в отметки времени без обнаружения. Стоит обратить внимание на то, что эта проблема ортогональна(параллельна) проблемам аутентификации, целостности и ненарушения, которые решаются

с помощью цифровых подписей. Если сторона создает электронную подпись для документа, она устанавливает только проверяемую связь между стороной и документом. Наличие действительной цифровой подписи доказывает, что сторона действительно намерена подписать документ, и что документ не был изменен. Однако цифровая подпись не гарантирует ничего о времени, в которое документ был подписан: отметка времени требует доверия к стороне, которая ее подписала. В случае финансовых операций и других форм юридических договоров время имеет существенное значение, и порядок этих финансовых транзакций должен быть независимо сертифицирован для проведения аудита [12].

Положительные качества блокчейна и биткоина можно рассмотреть на сделках, связанных с недвижимостью. Владелец может быть определен как сторона, которой последний раз был продан дом, но право собственности может быть проверено только с полного бумажного исследования всех транзакций, связанных с домом, в некотором роде «бумажной дорожкой», которая обычно хранится и проверяется титульными компаниями. Стоит заметить, что система не полностью предотвращает мошеннические транзакции (например, человек, продающий дом, которым он не владеет, или продает одно и то же имущество более чем одной стороне), но мошеннические действия в конечном итоге обнаруживаются, как и реальное право собственности. Та же проверка собственности возникает в финансовых транзакциях – обязательно, при продаже криптовалюты, а также в продаже любого другого традиционного финансового инструмента. Обычно проблема решается путем записи всех транзакций в одном доверенном централизованном регистре, но «книга» не всегда является практическим решением, поскольку она не масштабируется для большого количества частых транзакций и потому она требует, чтобы все стороны доверяли хранителю книги. Точно так же вам нужно доверять своему банку свои деньги (а сотрудники банка, крадущие средства клиентов не редкость). Чтобы устранить это, блокчейн предоставляет механизм распределенного

доверия: несколько сторон хранят записи транзакций, и каждая сторона может проверить, не были ли изменены порядок и отметки времени транзакций [34].

Единица биткоина – это не что иное, как число, но только некоторые цифры из него представляют из себя подтвержденный биткоин. Эти числа являются решениями четко определенного уравнения, и тот, кто находит его решение, получает какое-то количество биткоинов. Этот процесс называется майнингом. Как только биткоин обнаружен, он может участвовать в транзакциях, которые будут храниться в базе данных. Сделки подписываются цифровой подписью с учетными данными продавца, чтобы избежать отказа [34].

Централизованной базы данных в такой системе не существует, поскольку пользователи не доверяют ей, а также потому, что существует слишком много транзакций для хранения их всех в одном месте. Поэтому биткоин и другие криптовалюты предоставляют распределенную базу данных, в которой каждый компьютер, участвующий в транзакции определенной монеты, хранит копию истории транзакции этой монеты. Технология блокчейн гарантирует, что ни одна сторона, хранящая эту историю, не может вмешиваться в нее, оставшись незамеченным. Децентрализованность такой системы обеспечивается технологией одноранговой сети peer-to-peer(P2P). Именно он отвечает за такую характеристику блокчейна, как хранение цепочки блоков транзакций, в которой периодически происходит добавление новых блоков у каждого участника такой системы [34]. Простейшая структура сети изображена на рисунке 1.1.

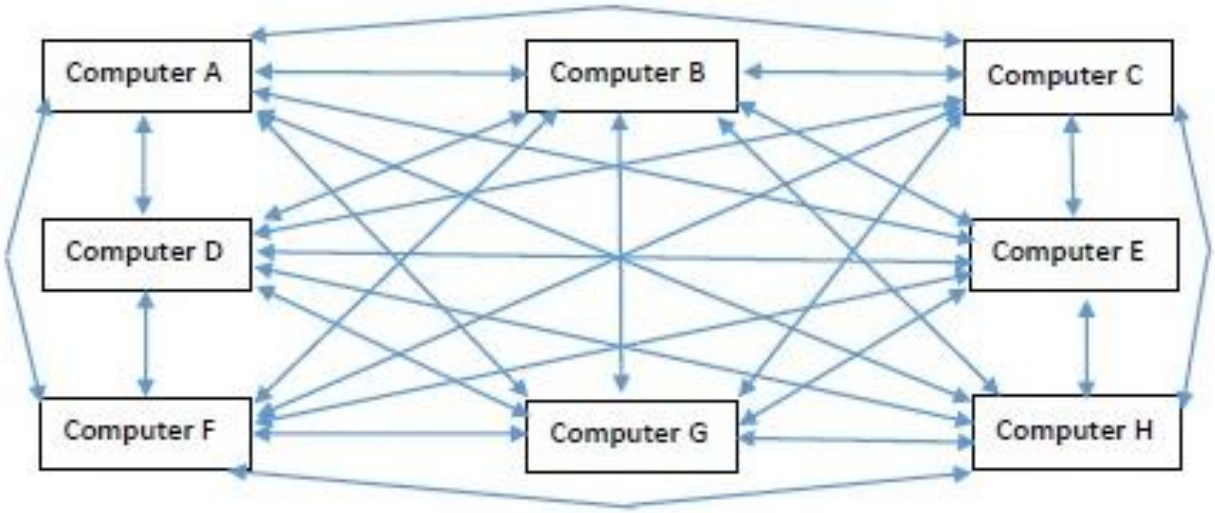


Рисунок 1.1 – Схематичное представление P2P сети

Транзакции – это единицы данных, содержащие сведения о транзакции и отметке времени. Оба могут быть представлены в виде вычисленных чисел и строк. Блокчейн можно рассматривать как цепочку блоков с тремя столбцами, где каждая строка представляет отдельную транзакцию. В первом столбце хранится метка времени о транзакции. Во втором столбце хранятся сведения о транзакции. А третий столбец содержит в себе свой хэш и хэш предыдущего блока, так все блоки представляют собой единую цепочку, то есть историю всех действий в системе с начала её старта [34].


Рисунок 1.2 – Цепочка блоков

Когда новая запись появляется в блокчейне, последний вычисленный хэш передается каждой заинтересованной стороне. Необязательно, чтобы каждая сторона хранила копию всей истории транзакций, достаточно, чтобы это сделали несколько сторон. Цепочка проходит проверку целостности на устройстве у каждого клиента сети. Описанная структура служит гарантией того, что любое несанкционированное действие будет отвержено и признано

невалидным, при такой проверке пользователями, которые хранят у себя валидные цепочки. Единственный способ вмешаться в данные при сохранении хэша – найти коллизию. Но такой способ потребует слишком много вычислительной мощности, что не будет целесообразным, и, следовательно, такой метод практически невозможен.

Хэш может рассматриваться как необратимо зашифрованная версия исходной строки. На самом деле, один из способов вычислить хэш строки – это зашифровать ее, выполнить некоторые скремблирования и выполнить битовые операции XOR. Математически хэш создается хэш-функцией f, которая должна иметь два важных свойства: размер входного и выходного пространства должен быть большим; коллизии должны быть практически невозможными, то есть 2 входные переменные $x_1$и $x_2$, не должны производить две одинаковые функции на выходе $f\ x_1\ = f(x_2)$. Типичное применение хэш-функции — это хранение паролей. Например, при регистрации на веб-сайтах, может возникнуть ситуация, в которой пароль p хранится в базе данных веб-сайта в незашифрованном виде. В таком случае, каждый, кто имеет доступ к базе данных, может посмотреть пароль p любого пользователя. Чтобы этого не случалось, на сайте должны храниться хэши паролей $f\ p\ = y$. В таком случае, при каждом входе в систему входной пароль каждый раз хэшируется и сравнивается с изначальным значением $f\ p\ = y$. Вероятность того, что неверный пароль будет иметь такое же значение хэша, как у фактического пароля практически равна нулю. Примерами хэш-функций являются безопасные хэш-алгоритмы (SHA1, SHA128, SHA512 и т.д.), которые реализуются модулем hashlib в языке программирования Python. Он может принимать любую строку в качестве входных данных и всегда создавать выходную строку, которая представлена в шестнадцатеричном виде с фиксированной длиной [34].

### 1.2.1 Принципы технологии blockchain

Итак, технология blockchain основывается на 4 важных принципах, которые благотворно скажутся на применении этой технологии в сфере электронного голосования.

1) Проверка транзакций в сети с использованием криптографии.

На сегодняшний день, в сферах, где используется передача данных, после отправки некоторого сообщения со сведениями, который нужно синхронизировать с получателем, получатель должен обновить свой собственный реестр с данными. На сегодняшний день не существует удобного, не затратного по времени и эффективного решения, чтобы сверять такие копии. И именно технология blockchain позволяет справиться с этой проблемой, с помощью различных способов. К примеру, при обмене одними и теми же базовыми данными или при предоставлении подтверждающих элементов с целью верификации данных. Пользователи blockchain реестра достигают соглашения в отношении верификации изменения данных используют некоторые алгоритмы консенсуса, такие как Proof of Stake, Proof Of Work, Proof Of Activity, Proof of Weight и т.д [12, 34].

2) Распространение копий информации о транзакциях среди большого количества участников системы.

База данных в системе на блокчейн либо целиком, либо частями распределяется между устройствами участников системы. Это позволяет избежать критических ошибок, которые могли бы случиться на одном устройстве. На данный момент, мультиплицирование является довольно большой проблемой для нынешних технологий организации баз данных, и это приводит к внеочередным сложностям и затратам в ходе реализаций различных проектов. Кроме того, такое достоинство блокчейна позволяет сохранить целым все копии, если в одной из них произошел сбой. Многие участники могут также подтверждать добавление новых транзакций в ходе проверки сопоставления собственными силами [12, 34].

3) Децентрализованный контроль доступа

В реестрах с децентрализованных системах используются ключи и подписи в целях предоставить определенным участникам реестра права на определенные манипуляции в общей базе данных. Такие ключи могут позволить выполнять определенные действия, которые будут возможны только при соблюдении некоторого ряда условий. Так, регулятор может иметь ключ для чтения, который даст ему возможность просмотра информации о транзакциях соответствующей организации. Но это возможно только при условии, если владелец предоставит регулятору соответствующий ключ [12, 34].

4) Конфиденциальность и прозрачность

Поскольку, большое количество сторон участвует в верификации каждой транзакции, а копия данных находится у многих участников, такая система обеспечивает высокий уровень прозрачности. Данное свойство предоставляет возможность убедиться в том, что реестр не редактировался несанкционированным путем. Каждая транзакция производится с использованием уникальной цифровой подписи, благодаря чему подтверждается факт того, что определенный пользователь совершил транзакцию в соответствии с существующими в системе правилами [12, 34].

### 1.2.2 Процесс верификации в blockchain

Структура блока в технологии blockchain представляет из себя список транзакций(тело) и заголовка, который содержит в себе ключи транзакций.

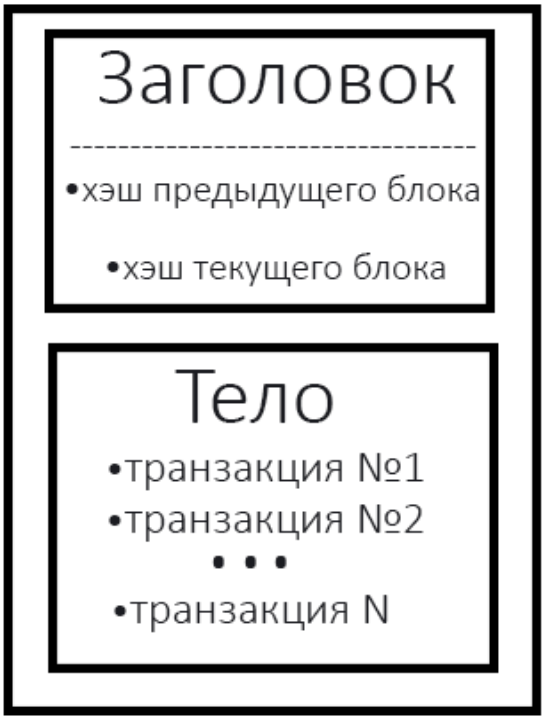


Рисунок 1.3 – Содержание блока

Тело блока содержит в себе список всех предыдущих транзакций.

Заголовок каждого блока хранит результат выполнения хэш-функции текущей и предыдущей транзакции. С помощью хэша идентификатора блока каждый блок связывается с предыдущим и последующим блоком. Данное техническое решение позволяет сохранять надежность системы.

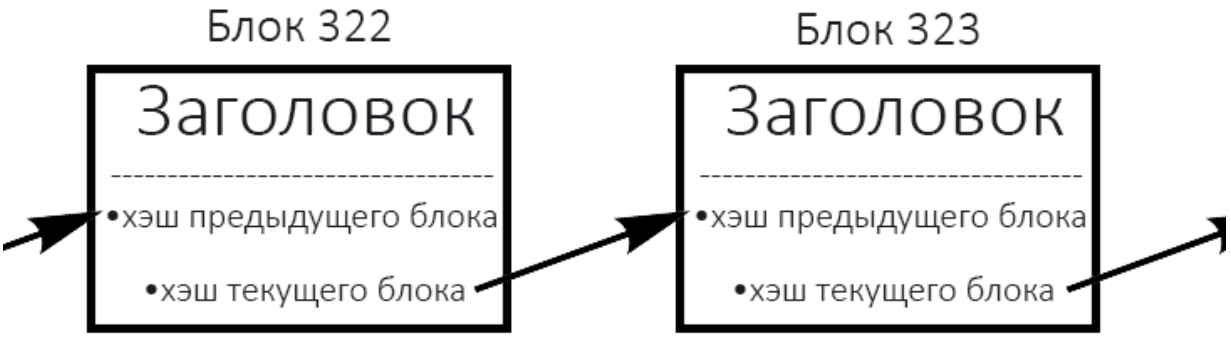


Рисунок 1.4 – Связь блоков между собой

Хэш идентификаторов рассчитывается из хэша транзакций текущего блока и хэша идентификатора предшествующего блока. То есть, в заголовке блоков зашифрована информация о данных из предыдущих блоков. Отсюда следует, что в такой системе есть возможность отследить все корректировки в

данных в каждом из блоков, т.к. если внести изменения в данные в одном блоке, изменится идентификатор всех следующих боков.

Получается, имея цепь из блоков, в которых содержится хэш любого блока, можно проверить оригинальность цепи блоков, корректны ли данные в том или ином блоке, присутствуют ли в цепи поддельные блоки и т.п.

Чтобы достичь соглашения между участниками системы касательно внесения изменений в блокчейн используются алгоритмы консенсуса, такие как proof-of-work(доказательство выполнения работы) и proof-of-stake(Доказательство доли владения).

Доказательство выполнения работы(PoW) – это принцип защиты сетевых систем от злоупотреблений услугами, например, DoS-атак и других видов атак. Данный метод основан на решении сложной математической задачи одной стороной, которая требует некоторых вычислительных мощностей и времени. А результат этого решения может быстро проверить другая сторона. То есть особенностью таких вычислений является асимметричность затрат по времени. Данная особенность обеспечивает сильную защиту системы от фальсификаций. Хэш идентификатора блока должен удовлетворять определенному условию, которое устанавливает сложность вычислений.

Доказательство доли владения(PoS) – принцип защиты, который основан на доказательстве искомых средств. Этот метод является альтернативой proof-of-work. В случае использования данного принципа защиты, в цепочку блоков, с большей вероятностью, запишется блок, учетная запись которого имеет на балансе больше токенов.

Процесс верификации новых блоков называется майнингом(mining). А участник сети блокчейн, который занимается майнингом именуется майнером.

Майнер получает от других пользователей системы информацию о новых записях в реест, собирает их вместе и пытается сформировать новый блок. Для этого ему требуется вычислить хэш идентификатора нового блока.

Например, если после первого вычисления получился результат, который не удовлетворяет требованиям безопасности системы, в блоке существует специальное поле «nonce», которое изначально равно нулю. Так, чтобы получить результат, который будет удовлетворять требованиям безопасности, манер делает перерасчет хэша с измененным полем «nonce» столько раз, сколько потребуется для того, чтобы хэш получился подходящим.

Чтобы вычислить хэш, который будет удовлетворять требованиям, майнер чаще всего совершает многочисленные вычисления. И в тот момент, когда подходящий хэш вычислен, блок верифицируется и отправляется другим участникам сети.

Процесс майнинга особенен тем, что независимо от того, сколько повторных вычислений провел майнер, вероятность вычисления нужного решения одинакова у всех. Данная особенность делает невозможным предварительный расчет хэша или сохранения результата всех расчетов, чтобы использовать их в дальнейшем. Это делает всех участников майнинга равноправными.

За каждое успешное вычисление майнер получает вознаграждение, комиссию от транзакции, которую он рассчитывал. Данное вознаграждение и есть причина, по которой он выполняет столь трудоемкую задачу [12].

### 1.2.3 Голосование, основанное на blockchain

В последние годы активно развиваются децентрализованные цифровые валюты, в которых главными преимуществами является защищенность от вмешательства третьих лиц в такие финансовые системы и прозрачность. Некоторые исследователи искали способ голосования и подсчета голосов посредством технологии блокчейн. В 2015 году Дж. С. Чеплач в IT Университете Копенгагена представил доклад об областях использования блокчейна где, в том числе утверждал, что блокчейн может использоваться для электронного голосования [25]. В то же время З. Жао и Т. Чан предложили способ голосования с использованием технологий блокчейн и zk-SNARK. Такой способ голосования имел свойства конфиденциальности,

проверяемости и неизменяемости [41]. В 2016 году был предложен протокол с использованием Zerocoin, который обеспечивал бы большинство требований электронного голосования. В том же году, П. С. Джейсон, и К. Ючи предложили протокол с использованием слепой подписи и карт Биткоин [30].

Однако, использование Zerocoin сложно реализовать в программном обеспечении в связи с недостаточным литературным базисом и непопулярностью данной платформа. Zerocoin специализируется в разработке именно финансовых систем и среди разработчиков децентрализованных приложений, не связанных с финансами, не имеет высокого спроса.

Недостатками системы голосования основанной на платформе Bitcoin являются низкая скорость транзакций, ввиду недостаточной пропускной способности ее сети. Из-за чрезмерной популярности криптовалюты, в очереди транзакций скапливается большое количество транзакций, как следствие, растет время обработки транзакций и стоимость комиссий.

Поэтому, данные предложения не позволяют реализовать систему голосования на основе технологии blockchain достаточно эффективно.

### 1.3 Требования к голосованию с применением blockchain

Итак, технология blockchain, благодаря своей децентрализованности, репликации базы данных между участниками, неизменяемости данных и сохранению всех тразакций в виде цепочки блоков, позволяет использовать данные положительные качества в электронной системе голосования в целях устранения недостатков систем электронного голосования, сформулированных в п. 1.3.

Система голосования на blockchain должна выполнять следующие требования:

- возможность создать опросы и списки объектов голосования к ним;
- регистрация участников для каждого созданного опроса;

- возможность голосования;
- обеспечение прозрачности;
- обеспечение отказоустойчивости;
- отсутствие возможности вносить несанкционированные изменения, влияющие на результат голосования.

То есть, в такой системе у пользователя должна быть возможность создать нужный опрос, после чего создателю опроса будет присуждаться статус администратора опроса, с помощью которого он сможет ограничивать круг лиц, которые будут иметь разрешение на участие в созданном голосовании.

Кроме того, каждый пользователь такой системы должен иметь возможность просмотра результатов голосования в реальном времени и просмотра цепочки блоков, чтобы удостовериться в прозрачности голосования.

Помимо этого, система должна быть отказоустойчивой, то есть при выходе из строя устройства с базой данных голосования, система должна продолжать работать, т.к. база данных будет реплицироваться на устройства участников сети, в которой запущенно децентрализованное приложение.

Система также должна быть взломоустойчивой, то есть у злоумышленника не должно быть возможности влиять на ход голосования и его результаты.

# Глава 2 АНАЛИЗ ПЛАТФОРМЫ BLOCKCHAIN И ПРОЕКТИРОВАНИЕ СИСТЕМЫ ЭЛЕКТРОННОГО ГОЛОСОВАНИЯ

## 2.1 Платформа blockchain

На данный момент существуют значительное количество реализаций блокчейнов, среди которых присутствуют такие платформы, как Bitcoin, Ethereum, zCash, Wave, Ripple, Cordano, Stellar, IOTA и др [16].

Из перечисленных выше самых популярных платформ, такие платформы, как Ethereum, Wave и имеют под собой цель создать платформы, позволяющие решать самые разные задачи с помощью «смарт-контрактов», которые обеспечивают возможность различных манипуляций с данными в блокчейне.

Каждая из этих платформ имеет свои положительные стороны, но на данный момент при разработке децентрализованных приложений эффективнее всего будет обратиться к Ethereum, поскольку остальные платформы созданы относительно недавно, имеют меньшее количество обучающих материалов, а в средствах для разработки на данных платформах часто происходят изменения. Поэтому реализация децентрализованных приложений на данных платформах будет затруднительным и неэффективным.

### 2.1.1 Ethereum

В конце 2013 года канадский программист Бутерин В. Д. представил общественности единую централизованную виртуальную машину, которая называлась Ethereum. Первый запуск сети с открытым исходным кодом состоялся 30 июля 2015 года.

Создатель платформы поставил перед собой цель дать разработчикам возможность реализовывать различные децентрализованные приложения, обладающие свойствами масштабируемости, совместимости и простоты разработки. Достижение цели позволило разрабатывать децентрализованные приложения и смарт-контракты, обладающие форматами транзакций, правилами владения и функциями изменения состояния.

Большая популярность платформы Ethereum среди разработчиков и пользователей обусловлена, в том числе тем, что такие алгоритмы, как смарт-контракты впервые стали применяться именно в проекте Ethereum.

### 2.1.2 Смарт-контракты

Возможность выполнять различные манипуляции с данными в цепочке блоков предоставляется специальным компьютерным алгоритмом, который называется смарт контракт. Смарт контракт задает правила для хранения данных, описывает наборы функций для операций над ними. Все эти манипуляции осуществляются посредством интерфейса, который предоставляется смарт контрактом. Этот интерфейс создается из исходного кода отдельно от компиляции, а затем предоставляет возможность выполнять двоичный код. Как и предполагается блокчейну, все данные в Ethereum предоставляются для каждого участника, т.к. они реплицируются и распределяются между пользователями сети. Внесение изменений происходит в виде транзакций. В Ethereum конструкция транзакции выглядит следующим образом:

- Получатель.
- Электронная подпись отправителя.
- Сумма перевода в валюте «ETH», используемой в системе как токен.
- Комментарий.
- Лимит «топлива» на транзакцию(gasLimit).
- Стоимость единицы «топлива»(gasPrice).

«Топливо» - это единица измерения, которая используется для определения оплаты по конкретному вычислению. Стоимость «топлива» - это количество ETH, которое вы можете потратить на единицу «топлива». Стоимость «топилва» измеряется в «gwei». 1 Gwei равен 0,000000001 ETH.

Для проведения всех транзакций отправитель должен установить лимит топлива и его стоимость. Цена на топливо и лимит топлива – это

максимальная сумма в Gwei, которую отправитель готов заплатить за перевод токенов.

Так, например, установив gasLimit в 47 000 gwei, а gasPrice в 25 gwei, отправитель готова заплатить 1 175 000 gwei, или 0,001175 ETH.

То есть, для совершения транзакции требуются затраты «топлива», которые выплатятся майнеру. Подтверждение транзакции происходит на устройстве майнера, в виртуальной машине Ethereum Virtual Machine, где выполняется смарт контракт.

### 2.1.3 Децентрализованное приложение

Децентрализованное приложение(Dapp) – приложение, которое запускается участниками децентрализованной сети с протоколами защиты(PoW, PoS). Приложение может базироваться на разных технологиях, но сейчас под Dapp, в основном, подразумевается приложение на blockchain с использованием смарт контрактов. Логика современного Dapp это работа смарт контракта и UI для взаимодействия с приложением. ранение более-менее объемных данных и обмен сообщениями в идеальном DApp тоже должны быть децентрализованными, однако эти технологии только начинают появляться и заслуживают отдельной статьи. Блокчейн же обеспечивает хранение текущего состояния и реализует бизнес-логику через смарт-контракты.

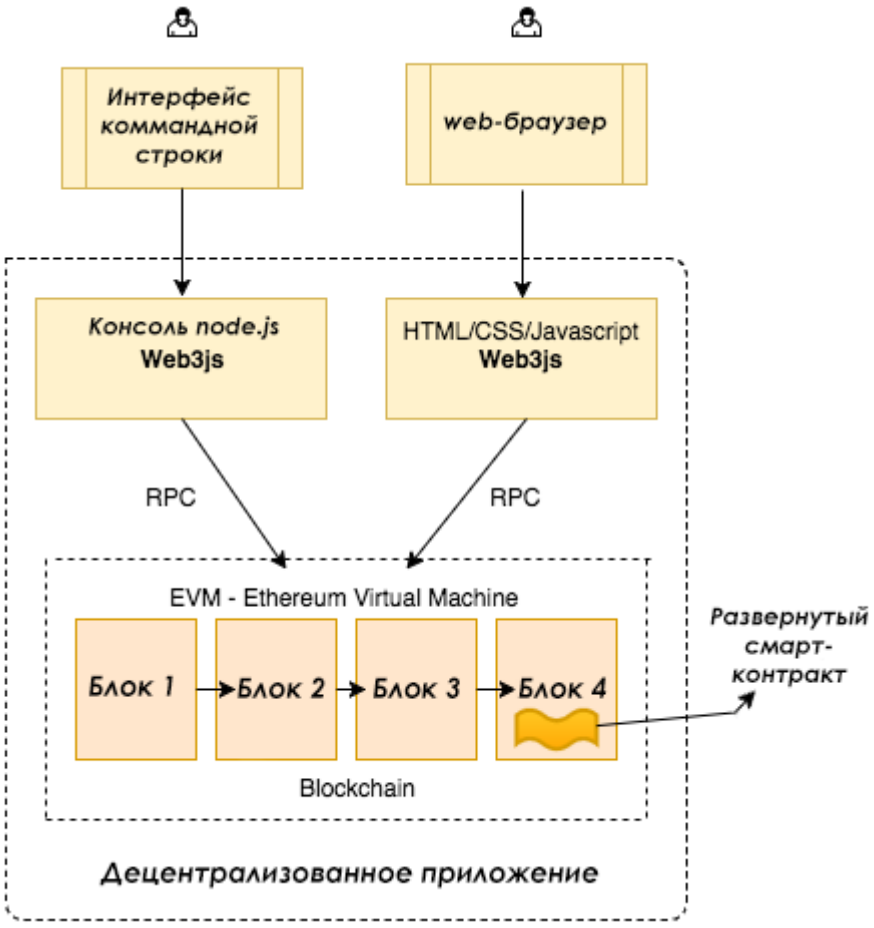


Рисунок 2.1 – Взаимодействие с Ethereum Dapp

На рисунке 2.1 представлен обзор взаимодействия пользователя с децентрализованным приложением.

Взаимодействуя с Ethereum Dapp, пользователь имеет возможность обращается непосредственно к блокчейну через специальное программное обеспечение или через интерфейс командной строки на своем устройстве.

## 2.2 Проектирование приложения

Разработка приложения начинается с его проектирования. Проектировка разрабатываемого приложение подразумевает под собой продумывание базовых функций, которое будет иметь электронное голосование и полномочия пользователей.

### 2.2.1 Функциональное моделирование приложения

Проектируемое мобильное приложение для электронного голосования, должно отвечать некоторым функциональным требованиям. На рисунке 2.2

представлена модель системы голосования, где показаны основные функции приложения.

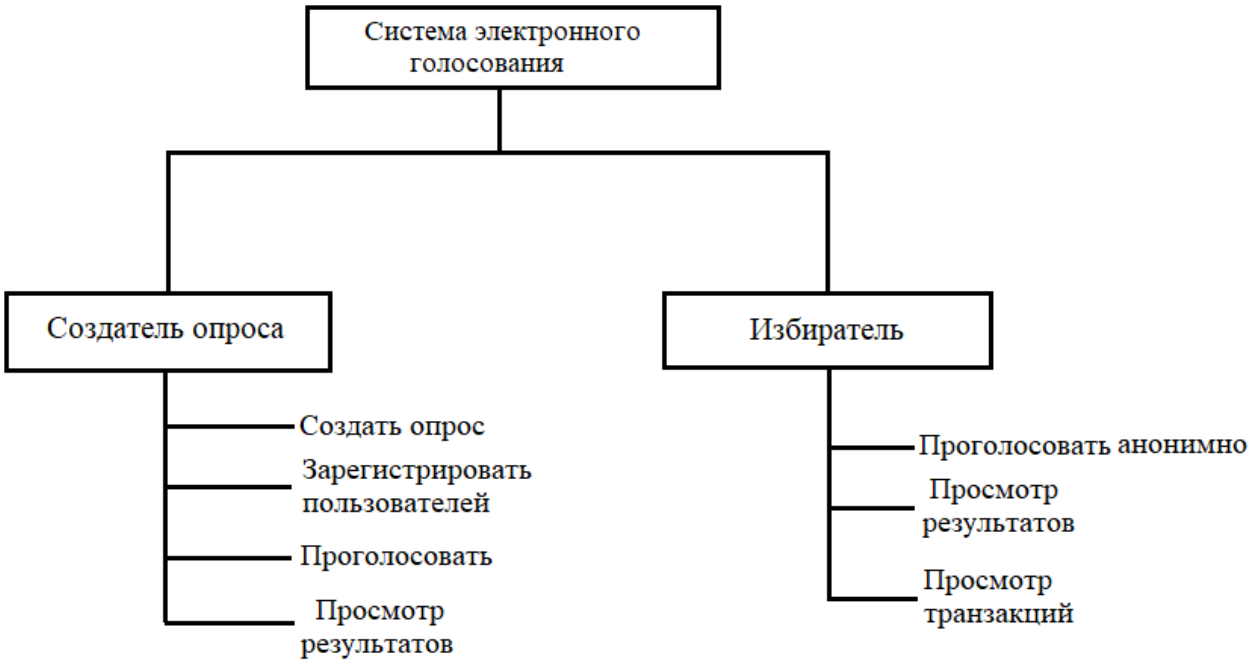


Рисунок 2.2 – Модель разрабатываемого приложения

Основные функции, которая предоставляет система голосования:

- создать опрос и быть его администратором с возможностью регистрировать избирателей;
- отдавать свой голос за тот или иной объект голосования;
- просмотреть результаты голосования;
- просмотреть связанные со смарт-контрактом транзакции, чтобы удостовериться в их честности.

При обращении к смарт контракту, он позволяет пользователю создать опрос, передав массив с элементами голосования, такими как: название опроса, обсуждаемый вопрос, варианты голосования. После создания опроса он может регистрировать новых пользователей в своем опросе, которые после регистрации будут иметь возможность проголосовать в этом голосовании. Также, смарт контракт предоставляет возможность обратиться к блоку с данными о результатах голосования. Кроме того, пользователь

имеет возможность просмотреть историю транзакции, такую возможность предоставляет сеть блокчейн, в которой развернут смарт-контракт.

### 2.2.2 Логическое моделирование системы голосования

Логическое моделирование представляет собой осуществление проверки функционирования логической схемы.

Основная цель заключается в осуществлении проведения проверки функций проектируемого приложения без полной реализации на данном этапе разработки. Преимущества данной модели заключается в том, что осуществляется проверка, как логических функций разрабатываемого приложения, так и её временные соотношения. Для осуществления моделирования необходимо построить диаграмму последовательности, диаграмму вариантов использования приложения.

На рисунке 2.3 показана диаграмма последовательности, на которой отображено взаимодействие объектов.

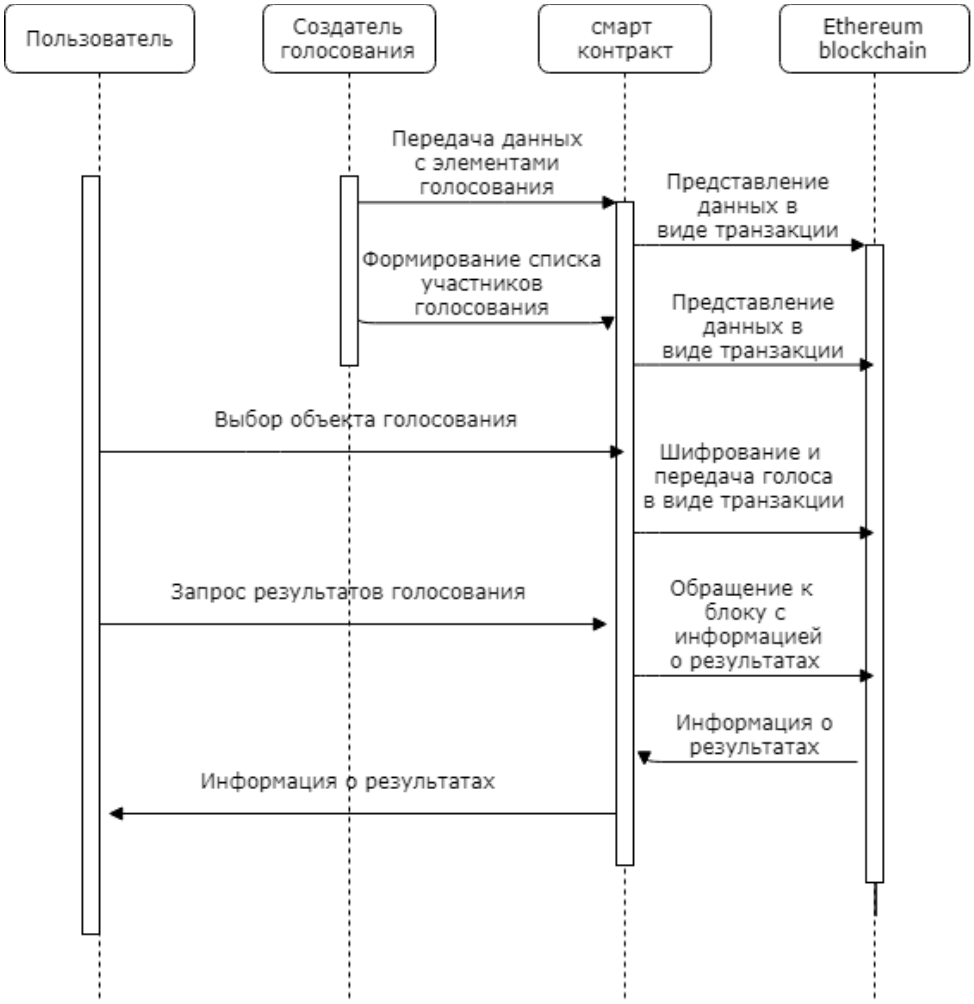


Рисунок 2.3 – Диаграмма последовательности работы системы электронного голосования

Взаимодействие между объектами и субъектами происходит следующим образом:

1. К разработанному смарт-контракту обращается создатель нового голосования и создает новый опрос.
2. Такие данные, как «Название голосования», «Обсуждаемый вопрос», «Варианты голосования» формируются в блок и отправляются в сеть blockchain.
3. Создатель голосования регистрирует избирателей в систему, путём добавления адресов их аккаунтов в список избирателей, что даёт им право на голосование в этом опросе.
4. Сформированный список избирателей отправляется в blockchain.

5. Пользователь обращается к смарт-контракту и выбирает в опросе объект голосования.

6. Имя объекта голосования, за которого был отдан голос шифруется и отправляется в блокчейн.

7. Пользователь обращается к смарт контракту, для того чтобы узнать результаты голосования.

8. Смарт-контракт обращается к нужному блоку с информацией в блокчейне.

9. Информация о результатах голосования выводится пользователю.

Таким образом, была осуществлена проверка логики функционирования приложения. Описана взаимосвязь и последовательность работы в приложении.

На рисунке 2.4 представлена диаграмма вариантов использования системы голосования пользователем.

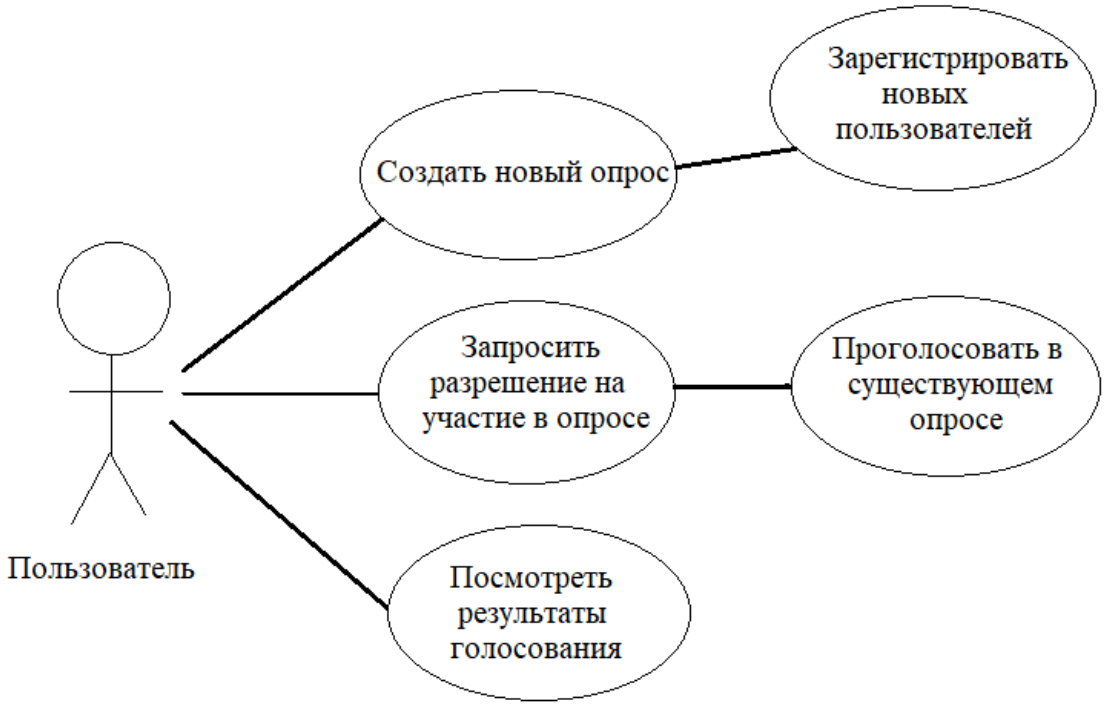


Рисунок 2.4 – Диаграмма вариантов использования

После того, как получено представление об основных функциях системы голосования, можно приступить к её реализации.

# Глава 3 РЕАЛИЗАЦИЯ СИСТЕМЫ ЭЛЕКТРОННОГО ГОЛОСОВАНИЯ

При разработке Dapp можно выделить 3 основных этапа: написание смарт-контракта, развертывание смарт-контракта в сеть Ethereum blockchain и подключение графического интерфейса пользователя.

Смарт-контракт играет ключевую роль в разработке децентрализованного приложения, на нем описываются все функции, которые будут позволять пользователю выполнять те или иные манипуляции с данными.

## 3.1 Выбор средств разработки смарт-контракта

### 3.1.1 Язык программирования Solidity

Для разработки смарт контракта для платформы Ethereum чаще всего используется объектно-ориентированный, предметно-ориентированный язык программирования Solidity. Данный язык программирования был представлен в 2014 году и рекомендуется к использованию командой проекта Ethereum, в официальном репозитории платформы Ethereum содержится документация и руководство по данному языку программирования.

Solidity – высокоуровневый язык для EVM с синтаксисом, приближенным к языку программирования JavaScript. Его схожесть с JavaScript позволяет быстро адаптироваться к написанию смарт-контрактов разработчикам, которые занимаются web-программированием.

Поскольку этот язык рекомендуется разработчиками проекта Ethereum, мною было решено выбрать именно этот язык для написания смарт-контракта.

### 3.1.2 Интегрированная среда разработки Remix

В официальном репозитории Ethereum, в руководстве к языку программирования Solidity также прилагается ссылка на интегрированную среду разработки под названием Remix Solidity.

Remix – самая популярная среда разработки смарт-контрактов. Ее популярность обуславливается его простотой. Так, для того чтобы начать

написание контракта на языке Solidity, с данной средой разработки не требуется никаких сложных манипуляций. Она работает в браузере, и для того чтобы начать разработку, достаточно зайти на веб-сайт remix.ethereum.org, где разработчика уже ждет открытый текстовый редактор, в котором можно начинать писать программный код.

Remix поддерживает множество функций, полезных для разработки Dapp, например, такие как:

- Пошаговый отладчик;
- Подсчет стоимости «топлива» для исполнения функций;
- Компиляция смарт-контракта;
- Подключение к желаемому RPC серверу;
- И прочее.

### 3.1.3 Программная платформа Node.js

Помимо среды программирования для разработки Dapp также может понадобиться программная платформа node.js, которая имеет множество полезных библиотек для разработки Dapp.

При разработке системы электронного голосования эффективно будет применять инструмент Ganache-CLI, который позволяет с одной команды поднять у себя на устройстве симулятор блокчейна, с включенным RPC протоколом. Этот инструмент создает 10 пробных аккаунтов в данном блокчейне. Баланс каждой тестовой учетной записи имеет 100 ETH. Данная процедура позволяет ускорить тестирование написанной программы, т.к. отсутствует потребность тратить время на поднятие реального частного блокчейна, создание аккаунтов и т.д.

Помимо Ganache-CLI, node.js имеет такую важную для разработки библиотеку, как Web3.js. Данная библиотека позволяет использовать API эфириума с помощью обычного JavaScript.

## 3.2 Разработка смарт-контракта

После того, как выбраны все необходимые средства для реализации, можно приступить к его написанию.

### 3.2.1 Базовые требования к смарт-контракту

Для начала необходимо сформировать базовые требования к смарт контракту, то есть определиться, какие манипуляции с данными в блокчейн он позволяет выполнять, и далее, отталкиваясь от этих функций, писать программный код смарт-контракта.

В самом начале в смарт-контракте нужно будет реализовать 3 простых функции:

- конструктор, которому мы будем передавать массив объектов, за которые будет идти голосование;
- присваивание создателю опроса статуса администратора;
- функция, которая позволит отдавать голос за нужный объект;
- шифрование выбранного имени объекта голосования для конфиденциальности выбора;
- функция, которая возвращает число голосов у того или иного объекта голосования.

### 3.2.2 Реализация базовых функций

В силу того, что Solidity довольно молодой и стремительно развивающийся язык, он накладывает некоторые неудобства в разработке, одним из таких неудобств является невозможность передавать массив строк в конструктор. Для того, чтобы хранить список объектов голосования, название опроса и обсуждаемый в нем вопрос было решено использовать массив bytes30 длиной в 30 байт, которой хватит для этих нужд. Однако в процессе разработки, в язык Solidity была добавлена возможность использования массива string, после чего в код были внесены некоторые изменения.

В случае разрабатываемой системы голосования, в начале контракта используется структура, то есть комплексный тип данных, к которым будут ссылаться основные функции.

```
struct Start {
    string title;
    string question;
    string option1;
    string option2;
    uint count1;
    uint count2;
    mapping (address => bool) voted;
    bool exists;
}
```

Рисунок 3.1 – Структура, к которой будут ссылаться функции

В данной структуре будут содержаться такие данные, как «Название опроса», «Обсуждаемый вопрос», «Объект голосования 1», «Объект голосования 2», «Объект голосования 3», счетчики голосов за каждого из кандидатов, статус о том, проголосовал тот или иной аккаунт, или нет и создан ли опрос или нет.

Далее идёт конструктор, который вызывается при развертывании смарт-контракта, или в момент, когда необходимо создать новый опрос, в который будет передаваться массив с объектами голосования, названием опроса и обсуждаемым вопросом, то есть все элементы голосования, описанные в структуре.

```
function createVoting (string _question, string _name, string _option1, string _option2, string _option3) public {
        polls[_name].title = _name;
        polls[_name].question = _question;
        polls[_name].option1 = _option1;
        polls[_name].option2 = _option2;
        polls[_name].option3 = _option3;
        polls[_name].count1 = 0;
        polls[_name].count2 = 0;
        polls[_name].count3 = 0;
        polls[_name].exists = true;
    }
```

Рисунок 3.2 – Конструктор, в который передается информация об элементах голосования

Далее, создана функция, которая позволяет добавить голос избирателя к счетчику голосов объекта голосования с применением протокола шифрования SHA-3, делается это с помощью функции, предоставляемой языком Solidity «keccak256».

```
function vote (string optionName, string pollName) public {
    require(doesPollExist(pollName));
    require(hasAlreadyVoted(pollName));

    polls[pollName].voted[msg.sender] = true;

    if (keccak256(polls[pollName].option1) == keccak256(optionName)) {
        polls[pollName].count1 += 1; }
     if (keccak256(polls[pollName].option2) == keccak256(optionName)) {
        polls[pollName].count2 += 1; }
    if (keccak256(polls[pollName].option3) == keccak256(optionName)) {
        polls[pollName].count3 += 1; }
}
```

Рисунок 3.3 – Функция «Голосовать за»

Данная функция позволяет добавить голос к тому или иному объекту голосования, предварительно проверив 2 условия:

1) существует ли опрос, в котором участвует избиратель;
2) была ли задействована эта функция раннее с этого аккаунта.

После проверки этих условий, функция присуждает голосующему аккаунту статус «проголосовавший» и прибавляет один голос к счетчику одного из трех объектов голосования.

Описанные в функции голосования условия, определяются функцией, которая проверяет существует ли запрашиваемый опрос и функцией, которая проверяет статус избирателя, была ли произведена передача голоса объекту голосования, или нет.

```
function doesPollExist (string pollName) private view returns (bool) {
    if (polls[pollName].exists) {
        return true;
    } else {
        return false;
    }
}
```

Рисунок 3.4 – Функция проверки наличия опроса

```
function hasAlreadyVoted (string pollName) private view returns (bool) {
    if (polls[pollName].voted[msg.sender]) {
        return false;
    } else {
        return true;
    }
}
```

Рисунок 3.5 – Функция проверки статуса избирателя

Функция просмотра счетчика голосов объекта голосования реализована на рисунке 3.6.

```
function getPollCounts (string pollName) public view returns (uint[3]) {
    require(doesPollExist(pollName));

    return [polls[pollName].count1, polls[pollName].count2, polls[pollName].count3];
}
```

Рисунок 3.6 – Функция, выводящая число голосов за объекты голосования

Также, в смарт-контракте реализованы функции, позволяющие посмотреть варианты голосования в том или ином опросе.

```
function getOption1 (string pollName) public view returns (string) {
    require(doesPollExist(pollName));

    return polls[pollName].option1;
}

function getOption2 (string pollName) public view returns (string) {
    require(doesPollExist(pollName));

    return polls[pollName].option2;
}

function getOption3 (string pollName) public view returns (string) {
    require(doesPollExist(pollName));

    return polls[pollName].option3;
}
```

Рисунок 3.7 – Функция для вывода названия объектов голосования в созданных опросах

### 3.2.3 Промежуточное тестирование смарт-контракта

Для того, чтобы осуществить промежуточную проверку работы смарт контракта применяется инструмент Ganache-CLI, который запускается из bash-консоли и создает локальный тестовый блокчейн с десятью аккаунтами. Вызывается он из консоли nodejs, командой ganache-cli, после этого нам показывается 10 тестовых аккаунтов. Тестовая сеть располагается по адресу localhost:8545.

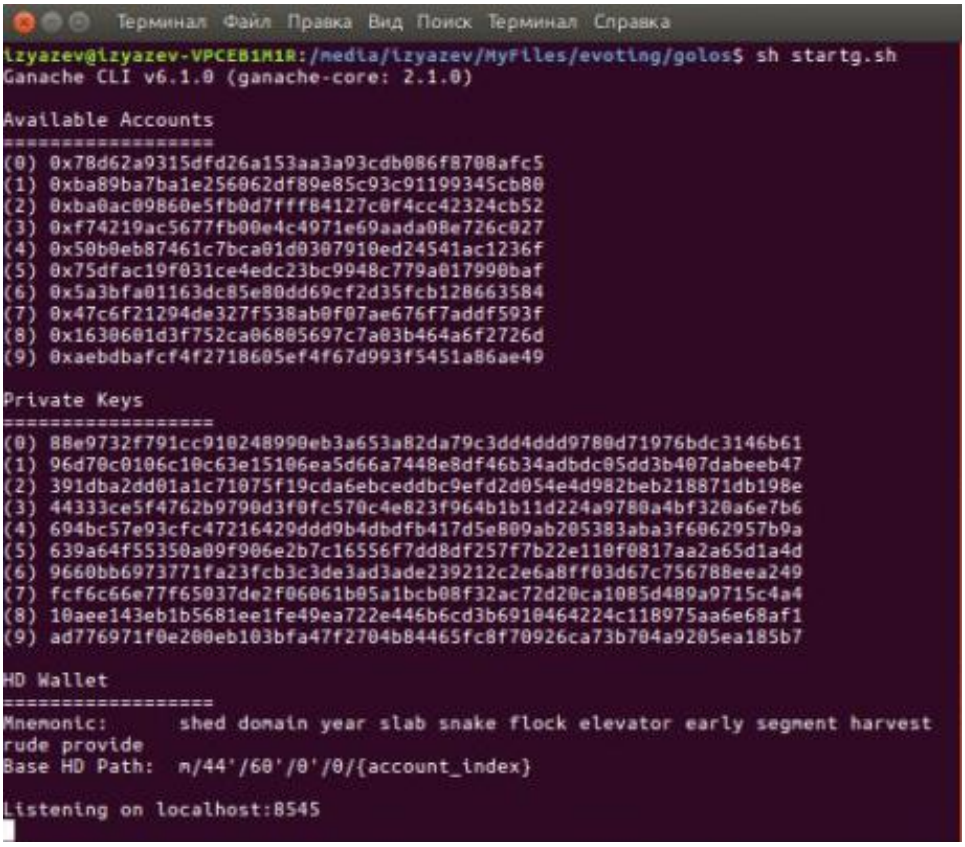


Рисунок 3.8 – тестовая blockchain сеть для проверки работы смарт контракта

После того, как тестовая блокчейн сеть поднята, выполняется компиляция смарт-контракта с расширением «.sol» в интегрированной среде разработки Remix. Далее, необходимо перейти во вкладку «Run» в среде разработки и в поле «Environment» указать адрес созданного с помощью ganache-cli web3-провайдера, то есть http://localhost:8545. После этого, в поле «Account» отобразятся существующие в сети тестовые аккаунтов, как на рисунке 3.9.

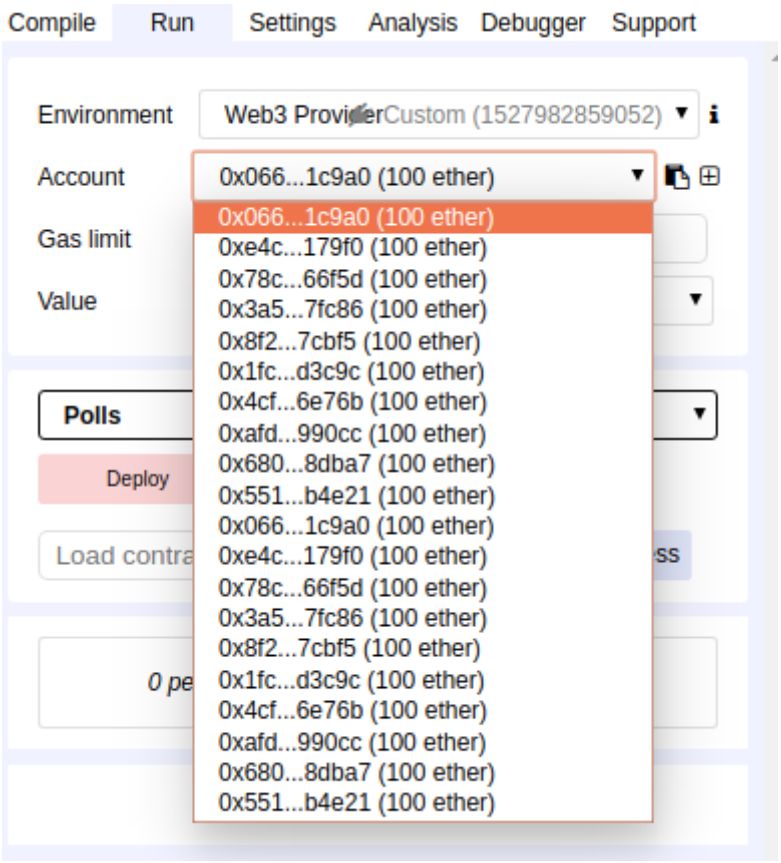


Рисунок 3.9 – Подключение Remix к тестовой сети ganache-CLI

После этих манипуляций нужно развернуть контракт с помощью кнопки «Deploy» и протестировать его работу, начиная с наполнения конструктора информацией о создающемся опросе.

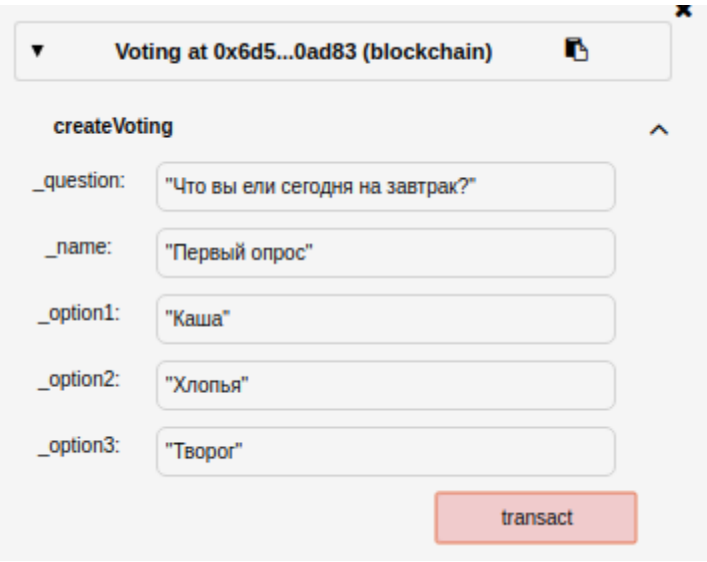


Рисунок 3.10 – Наполнения конструктора входными данными

Далее нужно протестировать функцию голосования, на рисунке 3.11 выбран вариант «Творог».

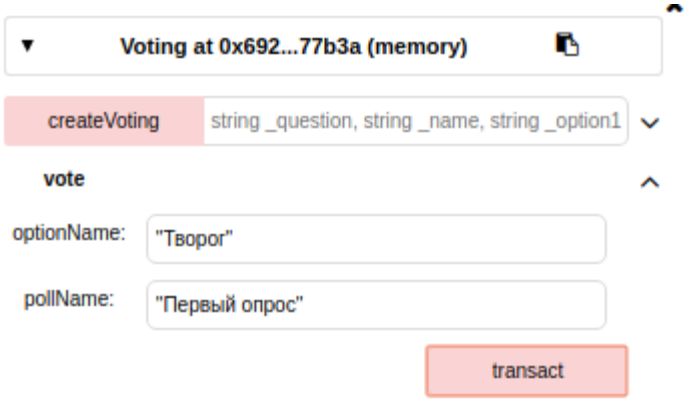


Рисунок 3.11 – Вызов функции голосования

Посмотреть, успешно ли сработала функция можно в ganache-cli. На рисунке 3.12 показано, что после того, как мы задействовали функцию голосования, успешно создалась транзакция и сформировалась в блок №3(блок №1 сформировался при развертывании транзакции, а блок №2 при создании опроса).

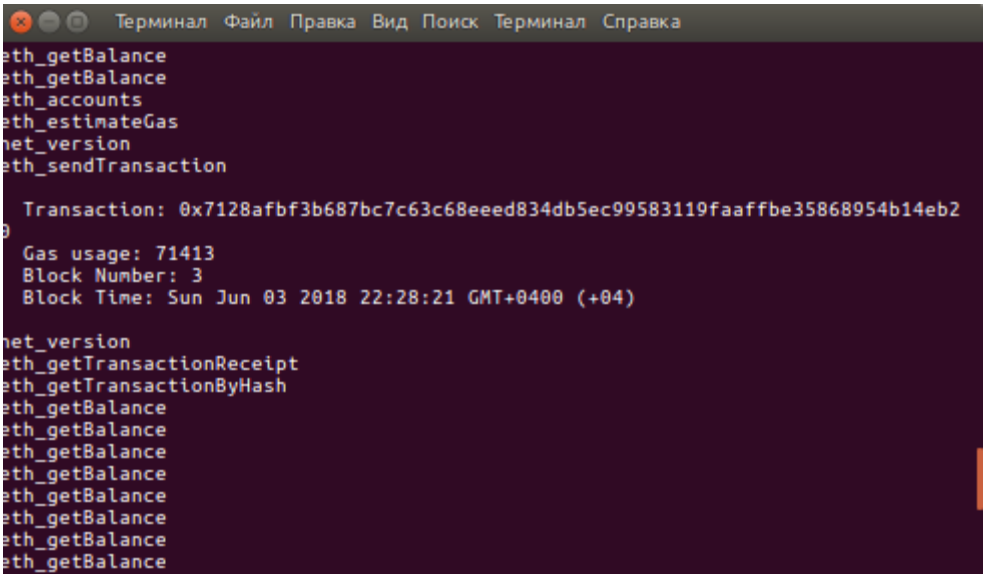


Рисунок 3.12 – Сформированный блок подтверждает успешное выполнение функции

После этого, можно проверить все остальные функции, на рисунке 3.13 показано, что работают функции вывода всех вариантов голосования, вывод

результатов голосования и вывод обсуждаемого вопроса в созданном голосовании.

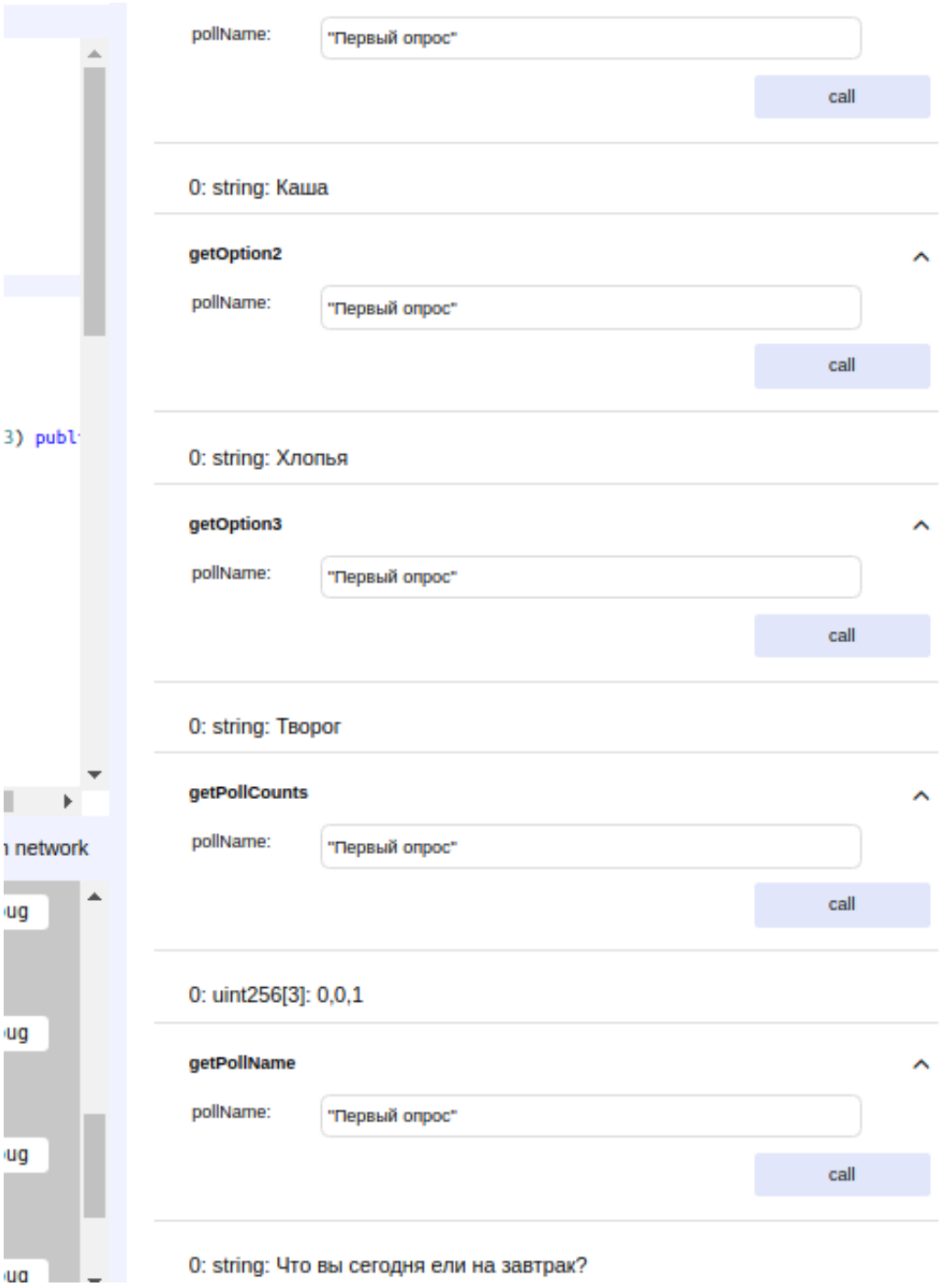


Рисунок 3.13 – успешное тестирование всех функций

### 3.2.4 Реализация регистрации

После тестирования основных функций следует добавить некоторые ограничения на голосование. Так, на данный момент к такой системе может присоединиться любой участник и проголосовать. Чтобы этого избежать, необходимо продумать как решить данную проблему.

Регистрация предусмотрена белым списком. В данном смарт контракте реализована возможность создавать неограниченное количество голосований. Это означает, что такой системой могут пользоваться разные организации, разные люди могут создавать свое голосование, свой обсуждаемый опрос и варианты ответов. В разрабатываемой системе голосования было решено назначить каждому опросу администратора, который будет являться создателем опроса. После того, как пользователь создает опрос, ему присваивается статус creator, пользуясь которым он может вызвать функцию создания белого списка и добавить туда необходимые аккаунты в сети блокчейн, которым будет позволено голосовать.

```
function createVoting (string _question, string _name, string _option1, st

    polls[_name].title = _name;
    polls[_name].question = _question;
    polls[_name].option1 = _option1;
    polls[_name].option2 = _option2;
    polls[_name].option3 = _option3;
    polls[_name].count1 = 0;
    polls[_name].count2 = 0;
    polls[_name].count3 = 0;
    polls[_name].exists = true;
    creator = msg.sender;
}
```

Рисунок 3.14 – Присваивание создателю опроса статуса creator

```
function addWhitelistAddress (address _address) public {
    if (msg.sender != creator) return;
    whitelist[_address] = true;
}

function addManyWhitelistAddress (address[] _addresses) public {
    if (msg.sender != creator) return;
    for (uint256 i = 0; i < _addresses.length; i++) {
    whitelist[_addresses[i]] = true;
   }
}
```

Рисунок 3.15 – Функция белого списка

При выполнении функция белого листа сначала происходит проверка наличия у пользователя статуса creator.

Теперь, для того чтобы участвовать в голосовании, пользователю требуется иметь аккаунт в частной или публичной сети blockchain, в которой

развернут данный смарт-контракт и разрешение от создателя, которое дается при внесении пользователя в белый список.

Разработка смарт-контракта закончена.

### 3.3 Разворачивание смарт-контракта в реальную blockchain-сеть

После того, как разработаны смарт контракт и web-интерфейс, чтобы система голосования работала как требуется, необходимо развернуть смарт-контракт в реальной blockchain-сети. Т.к. в blockchain-симуляции ganache-CLI смарт-контракт был успешно протестирован, то для завершения проекта осталось выполнить всего несколько манипуляций.

Для начала нужно установить клиент Go Ethereum из репозитория Ethereum. Делается это с помощью таких команд, как:

- apt-get install software-properties-common.
- add-apt-repository -y ppa:ethereum/ethereum.
- apt-get install ethereum.

Развернуть смарт-контракт можно либо на приватной сети Ethereum-сети, либо на публичной.

Частный Ethereum-blockchain можно создать самому средствами клиента geth, установленного раннее. Такая сеть будет закрыта от всего остального мира и предоставлена только разработчику, который ее запустил и тем, кому он дал в неё доступ.

Преимуществом частной сети является быстрая скорость транзакций, т.к. такая сеть не нагружена. Однако у такой сети будет ряд минусов. Так, например, в случае данного проекта электронного голосования, ввиду того, что из-за маленького количества пользователей, а на данный момент участников разрабатываемой системы голосования вообще нет, некому будет обрабатывать транзакции и верефицировать блоки. К тому же, при небольшом количестве пользователей существенно увеличивается риск «Атаки 51», при которым больше 50% майнеров могут объединиться и манипулировать данными в блокчейне по своему усмотрению.

Публичной сетью являются, например, основная сеть Ethereum, которая запустилась в 2015 году, в день презентации платформы. Также, имеется большое количество публичных сетей Ethereum, но не являющиеся официальными сетями проекта. В таких сетях «Атака 51» практически невозможна из-за большого количества майнеров и участников сети. В публичной сети на такую атаку придется затратить больше средств, чем можно получить.

Разворачивать смарт-контракт планируется в Ethereum-сети с названием Rinkeby. Выбор был сделан в сторону неофициальной блокчейн-сети Ethereum, т.к. при разворачивании контракта нужно скачать всю базу данных сети, а это очень ресурсоёмкая задача. Понадобится значительное количество времени и памяти на диске, т.к. официальная blockchain Ethereum по состоянию на 2018 год имеет в своей базе данных около 500 Гб.

Для того, чтобы запустить на своем устройстве узел Ethereum Rinkeby, необходимо выполнить команду^

geth --rinkeby --syncmode "fast" --rpc --rpcapi db,eth,net,web3,personal --cache=1024 --rpcport 8545 --rpcaddr 127.0.0.1 --rpccorsdomain "*".

После этого начнется скачивание блокчейна, в среднем данная операция занимает около 1 часа.

После того, как блокчейн загрузился, для развертывания смарт контракта понадобится фреймворк Truffle, который позволит развернуть контракт в публичной сети.

Далее необходимо распаковать Truffle в директории со всеми файлами, которые были созданы в процессе разработки системы голосования, упорядочить файлы требуемым фреймворком образом.

После того, как смарт контракт и средства для взаимодействия с системой готовы, нужно создать аккаунт с помощью команды в консоли

node.js web3.personal.newAccount('xxx'), где xxx – пароль от создаваемого аккаунта.

Затем необходимо, чтобы у аккаунта, который будет голосовать на балансе было некоторое количество токенов, чтобы с помощью них совершать голосование, которое происходит в виде транзакции. Для этого следует либо купить их за реальные деньги, либо воспользоваться сервисами, которые раз в какое-то количество времени переводят на нужный аккаунт символическую сумму в знак благодарности пользования их проектом.

После пополнения баланса аккаунта избирателя нужно разблокировать аккаунт, т.к. по умолчанию он заблокирован, делается это с помощью команды
web3.personal.unlockAccount('0xds651g6s5g6d654684654x3g354352srg654g', 'xxx', 13782), где после команды в скобках идет хэш аккаунта, пароль и количество «топлива» для выполнения операции.

После всех этих манипуляций выполняется компиляция и развёртывание смарт-контракта, которая совершается также, как и в тестовой сети, и после небольшого количества времени, его статус сменяется на состояние готовности к работе.

Для дальнейшего взаимодействия пользователя с контрактом, сейчас необходимо в IDE Remix перейти в раздел «Details» и скопировать всю информацию из поля «ABI», которая показана на рисунке 3.16:

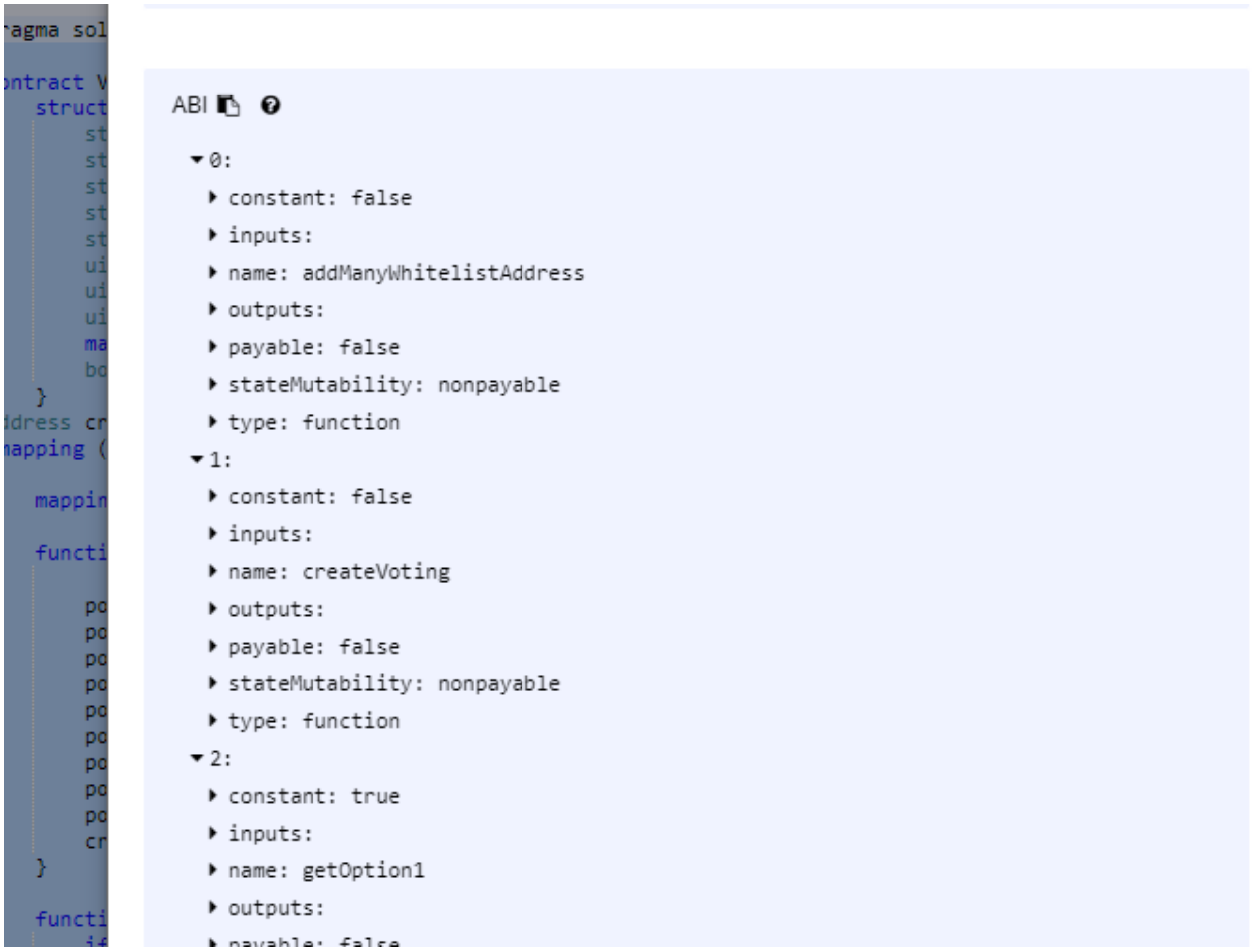


Рисунок 3.16 – ABI интерфейс

Данная манипуляция необходима для того, чтобы в будущем пользователь мог использовать графический интерфейс для работы со смарт-контрактом.

Помимо этого, в консоли Truffle необходимо ввести deployedContract.adress, то есть адрес смарт-контракта в сети blockchain, это также необходимо для дальнейшего взаимодействие через графический интерфейс.

## 3.4 Графический интерфейс для взаимодействия с децентрализованным приложением

Взаимодействие с системой голосования будет осуществляться посредством ПО Mist от разработчиков проекта Ethereum. Данное ПО является браузером и позволяет взаимодействовать пользователю со смарт-контрактом посредством своего графического интерфейса.

После того, как раннее, в сети rinkeby был развернут смарт-контракт, пользователю необходимо установить себе на компьютер ПО Mist с официального репозитория, который находится по адресу https://github.com/ethereum/mist/releases, данное ПО поддерживается практически на всех операционных системах.

После этого, ему необходимо создать аккаунт, сделать это можно нажав кнопку «Добавить аккаунт».

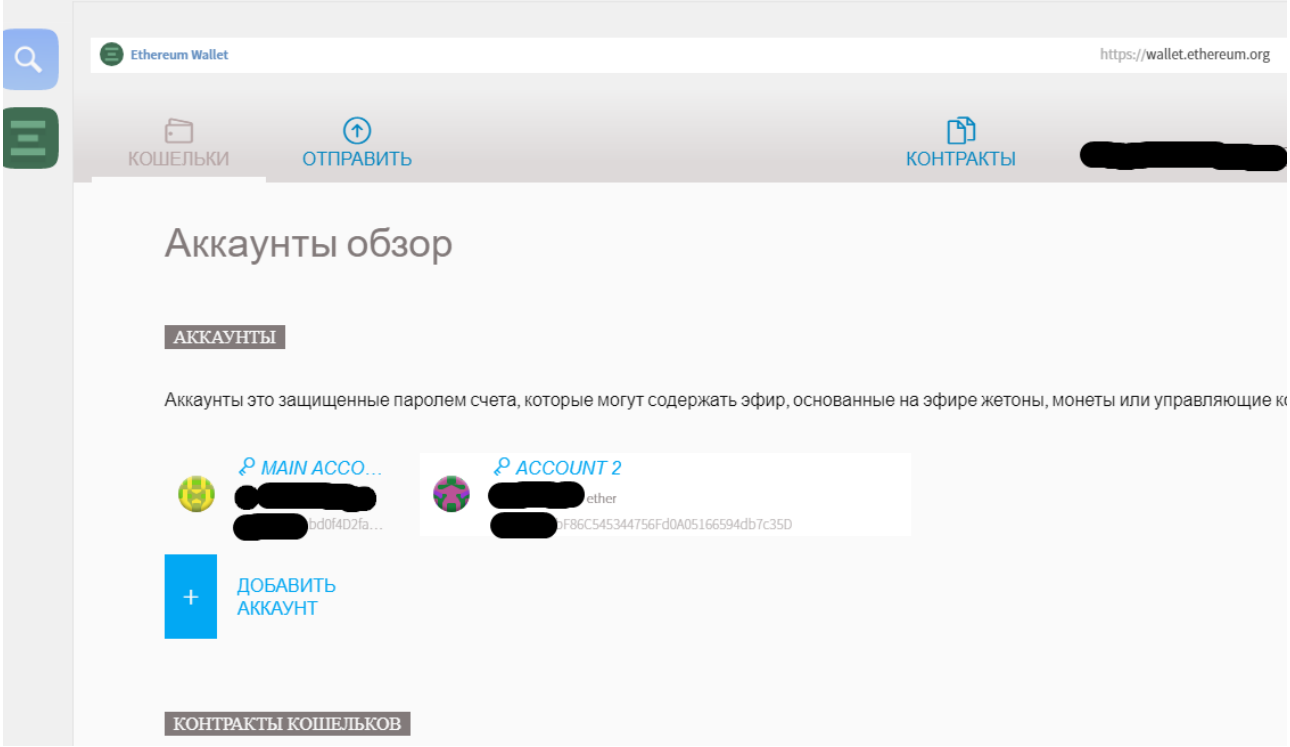


Рисунок 3.17 – создание аккаунта для дальнейшего взаимодействия со смарт-контрактом

После этого пользователю необходимо перейти во вкладку «Контракты», и нажать на кнопку «Наблюдать за контрактом». Откроется окно с полями для ввода «Адрес контракта», «Название контракта» и «JSON интерфейс». Пользователю необходимо заполнить эти поля. В нашем случае, адрес развернутого контракта это «0x4638D059097Ee13d97aC51919519549E844cC473», название контракта «Voting», а JSON интерфейс это ABI, которое было сохранено из среды разработки Remix.

После этого, у пользователя появляется возможность использовать все функции контракта, которые были разработаны.

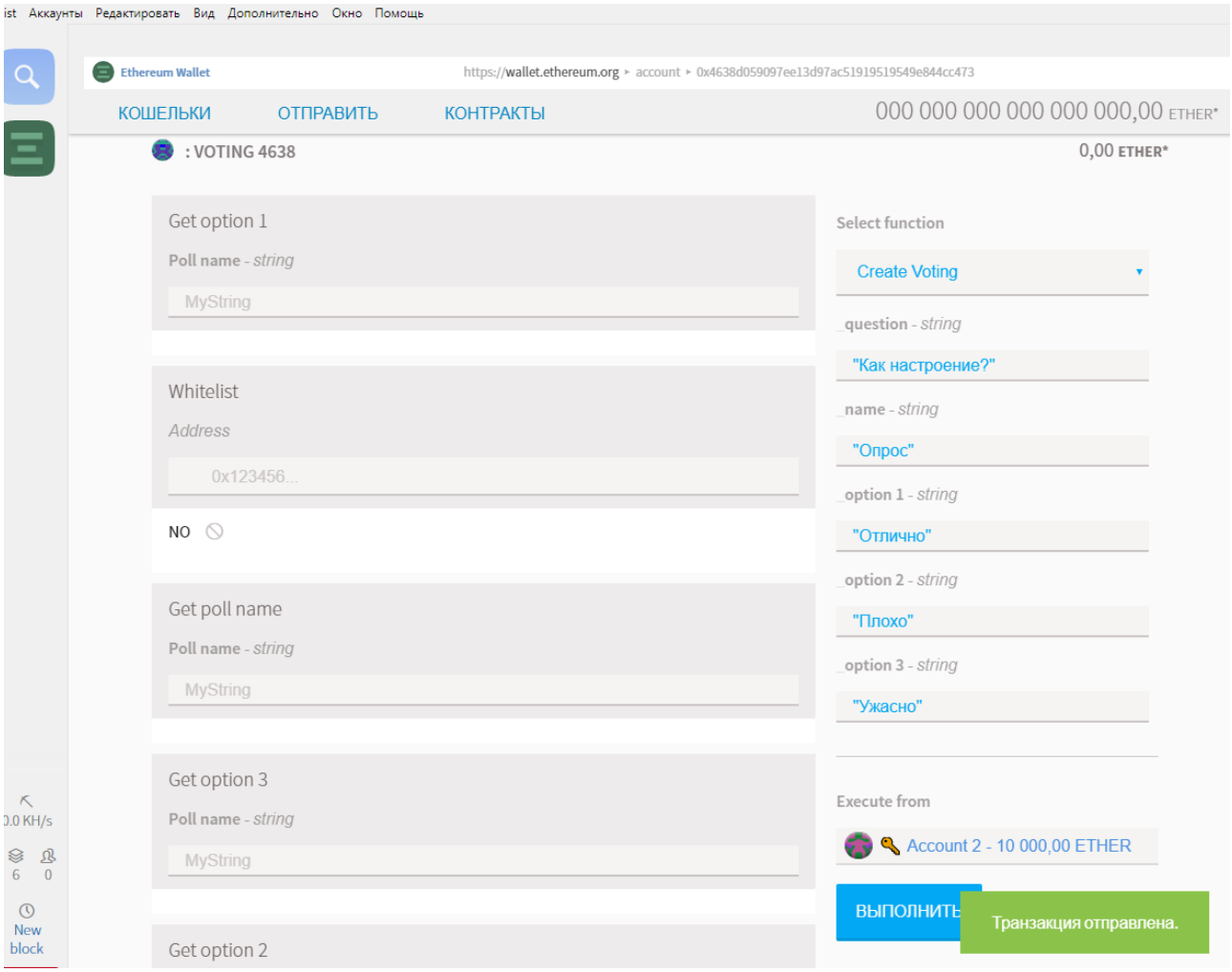


Рисунок 3.18 – Взаимодействие со смарт-контрактом через браузер Mist

## 3.5 Разработанное децентрализованное приложение

Итак, разработанный смарт-контракт позволяет совершать следующие манипуляции:

- При создании опроса, пользователь получает статус creator и имеет право дать избирателям доступ к голосованию в своем опросе.
- При создании опроса, в систему передаются данные о названии голосования, предмете спора и варианты ответа.
- Пользователь имеет возможность проголосовать только в том случае, если он добавлен в список избирателей и еще не голосовал.

- Все участники системы имеют право просмотреть результаты голосования.
- Объект голосования шифруется в момент передачи ему голоса по протоколу Secure Hash Algorithm-3 (Keccak), средствами языка solidity, с помощью указания функции «keccak256».

Алгоритм консенсуса, который будет использоваться в системе зависит от блокчейн сети, в которой развернут контракт. Так, в данный момент контракт развернут в сети rinkeby. В данной сети реализован алгоритм консенсуса «proof of authority». Данный алгоритм подразумевает возможность проверки транзакций и формирование блоков авторитетными узлами сети, т.е. майнерами. То есть, обрабатывать транзакции могут только доверенные лица. Данный алгоритм не получил широкого распространения и используется в частных и тестовых сетях, например, таких, как Rinkeby и Kovan. Данный алгоритм позволяет, не затрачивая много времени протестировать разработанный смарт-контракт.

Также, смарт-контракт можно развернуть в любой другой Ethereum blockchain-сети с другими алгоритмами, так, например, разворачивая контракт в основной сети от разработчиков Ethereum project, в системе будет использоваться алгоритм Proof-of-Work, который упоминался в первой и второй главе бакалаврской работы.

# ЗАКЛЮЧЕНИЕ

Целью данной выпускной квалификационной работы являлась разработка системы электронного голосования на основе технологии blockchain. В разработанной системе реализованы функции голосования, регистрация в системе реализована путём создания аккаунта в сети blockchain, в которой развернут смарт-контракт и добавлением пользователя в белый список того или иного созданного опроса.

В ходе выполнения работы был изучен теоретический базис по технологии blockchain, платформы для разработки децентрализованных приложений Ethereum, язык программирования Solidity, на котором происходит разработка смарт-контрактов, регулирующих условия для манипулирования данными в сети blockchain.

Проводился анализ существующих решений систем электронного голосования, выявлены их недостатки в сравнении с аналогичной системой, реализованной с помощью технологии blockchain. Кроме того, были проанализированы различные средства разработки децентрализованного приложения на платформе Ethereum, интегрированная среда разработки Remix, программная платформа node.js, специальные библиотеки node.js, протокол Ethereum go, позволяющий запустить приватную сеть блокчейн или подключиться к уже существующей.

Разработка и тестирование децентрализованного приложения осуществлялось на двух устройствах с операционными системами Ubuntu 16.04 и Windows 10.

Разработанное децентрализованное приложение позволяет запускать голосование с желаемыми объектами голосования, регистрировать аккаунты избирателям для пользования системой. Приложение имеет простой и понятный пользователю интерфейс, реализованный ПО Mist, позволяющий удобно работать с функциями смарт-контракта.

Задачи, поставленные в начале работы были выполнены, цель выпускной квалификационной работы была достигнута.

## СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ


1. Ожегов, С.И. Толковый словарь русского языка // С.И Ожегов, Н.Ю. Шведова. - 3 изд., стер. - Москва: Азъ, 1996. - 907 с.
2. Прасти Н. Блокчейн. Разработка приложений, // Н. Прасти, В.С. Яценков. — СПб.: БХВ-Петербург, 2018. – 256 с.
3. Равал С. Децентрализованные приложения. Технология Blockchain в действии, // С. Равал. — СПб.: Питер, - 2017. - 192 с.
4. Тапскотт Д., Тапскотт А. Технология блокчейн - то, что движет финансовой революцией сегодня, // Д. Тапскотт, А. Тапскотт. — М.: Эксмо, 2017. – 448 с.
5. Насколько надежно электронное голосование [Электронный ресурс]. – Режим доступа: https://www.svoboda.org/a/269300.html, свободный.
6. Норвегия официально отказалась от электронного голосования на выборах: оно контрпродуктивно [Электронный ресурс]. – Режим доступа: http://www.mk.ru/politics/world/2014/06/30/norvegiya-otkazalas-v-politike-ot-elektronnogo-golosovaniya.html, свободный.
7. Block The Vote: Could Blockchain Technology Cybersecure Elections? [Электронный ресурс]. – Режим доступа: http://www.forbes.com/sites/realspin/2016/08/30/block-the-vote-couldblockchain-technology-cybersecure-elections, свободный.
8. California: The Top to Bottom Review [Электронный ресурс]. – Режим доступа: http://www.votetrustusa.org/index.php?option=com_content&task=view&id=2554&Itemid=113, свободный.
9. IGS Votomatic Prototype Goes to the Smithsonian [Электронный ресурс]. – Режим доступа: https://web.archive.org/web/20070713201451/http://www.igs.berkeley.edu/publications/par/winter2001/votomatic.htm, свободный.
10. Kiwi. Bitcoin testnet sandbox. [Электронный ресурс]. – Режим доступа: https://testnet.manu.backend.hamburg/faucet, свободный.

11. NSW election result could be challenged over iVote security flaw [Электронный ресурс]. – Режим доступа: https://www.theguardian.com/australia-news/2015/mar/23/nsw-election-result-could-be-challenged-over-ivote-security-flaw, свободный.
12. Peer-to-peer [Электронный ресурс]. – Режим доступа: https://bitcoin.org/bitcoin.pdf, свободный.
13. Russian Hackers Acted to Aid Trump in Election, U.S. Says [Электронный ресурс]. – Режим доступа: https://www.nytimes.com/2016/12/09/us/obama-russia-election-hack.html, свободный.
14. Slim. Middeware-slim. [Электронный ресурс]. – Режим доступа: https://www.slimframework.com/docs/v3/concepts/middleware.html, свободный.
15. State bans electronic balloting in 4 counties / Touch-screen firm accused of 'reprehensible,' illegal conduct [Электронный ресурс]. – Режим доступа: https://www.sfgate.com/politics/article/State-bans-electronic-balloting-in-4-counties-2784975.php, свободный.
16. Top 100 Cryptocurrencies by Market Capitalization [Электронный ресурс]. – Режим доступа: https://coinmarketcap.com/, свободный.
17. Voting Machine Company Submits to Inquiry [Электронный ресурс]. – Режим доступа: https://www.nytimes.com/2006/10/31/us/politics/31vote.html?_r=1&oref=slogin, свободный.
18. Why machines are bad at counting votes [Электронный ресурс]. – Режим доступа: https://www.theguardian.com/technology/2009/apr/30/e-voting-electronic-polling-systems, свободный.
19. Baudron, O. Practical multi-candidate election system. In proceedings of the twentieth annual ACM symposium on Principles of distributed computing, // Baudron, O., Fouque, P.A., Pointcheval, D., Stern, J., Poupard, G. — ACM, 2001. -pp. 274–283.

20. Benaloh, J. Receipt-free secret-ballot elections. In Proceedings of the twenty-sixth annual ACM symposium on Theory of computing, // Benaloh, J., and Tuinstra, D. — ACM, - 1994. - pp. 544–553.
21. Card, D. Does voting technology affect election outcomes? touch-screen voting and the 2004 presidential election // Card, D., Moretti, E.- he Review of Economics and Statistics.,-2007.-pp. 660-673.
22. Chaum, D. L. Untraceable electronic mail, return addresses, and digital pseudonyms, // Chaum, D. L. — Communications of the ACM 24, - 1981. - pp. 84-90.
23. Christian Schaupp, L. E-voting: from apathy to adoption, // Christian Schaupp, L., Carter, L.— Journal of Enterprise Information Management 18,- 2005.- pp. 586-601.
24. Cohen, J. D. A robust and verifiable cryptographically secure election scheme // Cohen, J. D., Fischer, M. J.- Yale University. Department of Computer Science, -1985.
25. Czepluch, J. S. The use of block chain technology in different application domains, // Czepluch, J. S., Lollike, N. Z., Malone, S. O. — The IT University of Copenhagen, Copenhagen, - 2015.
26. DeMillo, R. A. Proceedings of the fourteenth annual ACM symposium on Theory of computing // DeMillo, R. A., Lynch, N. A. - ACM,-1982.- pp. 383–400.
27. Fraunholz, B. E-governance: enabling the french web 2.0 revolution? In Foundations of e-government, // Fraunholz, B., Unnithan, C.— [International Conference on E-Governance] Academic Publishing, - 2007. - pp. 344–359.
28. Fujioka, A. A practical secret voting scheme for large scale elections. In International Workshop on the Theory and Application of Cryptographic Techniques, // Fujioka, A., Okamoto, T., and Ohta, K. — Springer, - 1992. – pp. 244-251.

29. Hirt, M. Efficient receipt-free voting based on homomorphic encryption. In Advances in CryptologyEUROCRYPT 2000, // Hirt, M., and Sako, K. — Springer, - 2000. - pp. 539–556.
30. Jason, P. C. E-voting system based on the bitcoin protocol and blind signatures, // Jason, P. C., Yuichi, K.— TOM 10, - 2017.- pp. 14-22.
31. Jonker, H. Privacy and verifiability in voting systems: Methods, developments and trends, // Jonker, H., Mauw, S., and Pang, J. — Computer Science Review 10, - 2013. - pp. 1-30.
32. Juels, A. Coercion-resistant electronic elections, // Juels, A., Catalano, D., Jakobsson, M.— Towards Trustworthy Elections 6000, - 2010.- pp. 37–63.
33. Luo, F. Design and analysis of coercion-resistant electronic voting scheme. // Luo, F. - Fujian Normal University, 2005.
34. M. D. Pierro: What Is the Blockchain? [Text] / M. D. Pierro // Computing in Science & Engineering. – 2017. – PP. 92-95.
35. Macintosh, A. Characterizing e-participation in policy-making. In System Sciences, 2004, // Macintosh, A.— Proceedings of the 37th Annual Hawaii International Conference on,- 2004.- pp. 10-10.
36. Niemi, V. How to prevent buying of votes in computer elections. In International Conference on the Theory and Application of Cryptology, // Niemi, V., and Renvall, A. — Springer, - 1994. - pp. 164–170.
37. Ohkubo, M. An improvement on a practical secret voting scheme, // Ohkubo, M., Miura, F., Abe, M., Fujioka, A., Okamoto, T. — Information Security, - 1999. – pp. 771-771.
38. Okamoto, T. An electronic voting scheme. In Advanced IT Tools, // Okamoto, T. — Springer, - 1996. - pp. 21–30.
39. Peters, G. W. Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart contracts on the internet of money. In Banking Beyond Banks and Money, // Peters, G. W., Panayi, E.— Springer, -2016. - pp. 239–278.

40. Sako, K. Receipt-free mix-type voting scheme. In Advances in CryptologyEUROCRYPT95, // Sako, K., and Kilian, J. — Springer, - 1995.- pp. 393–403.
41. Zhao, Z. How to vote privately using bitcoin. In International Conference on Information and Communications Security, // Zhao, Z., and Chan, T.-H. H.— Springer, - 2015. - pp. 82–96.

## ПРИЛОЖЕНИЕ А

### Программный код смарт-контракта

```
pragma solidity ^0.4.24;
contract Voting {
struct Start {
 string title;
string question;
string option1;
string option2;
string option3;
uint count1;
uint count2;
uint count3;
mapping (address => bool) voted;
bool exists;
}
address creator;
mapping (address => bool) public whitelist;
mapping (string => Start) polls;
function createVoting (string _question, string _name, string _option1, string
_option2, string _option3) public {
polls[_name].title  = _name;
polls[_name].question = _question;
polls[_name].option1 = _option1;
polls[_name].option2 = _option2;
polls[_name].option3 = _option3;
polls[_name].count1 = 0;
polls[_name].count2 = 0;
polls[_name].count3 = 0;
polls[_name].exists = true;
```

```
creator = msg.sender;
}
function addWhitelistAddress (address _address) public {
if (msg.sender != creator) return;
whitelist[_address] = true;
}
function addManyWhitelistAddress (address[] _addresses) public {
if (msg.sender != creator) return;
for (uint256 i = 0; i < _addresses.length; i++) {
whitelist[_addresses[i]] = true;
}
}
function vote (string optionName, string pollName) public {
require(doesPollExist(pollName));
require(whitelist[msg.sender] == true);
require(hasAlreadyVoted(pollName));
polls[pollName].voted[msg.sender] = true;
if (keccak256(polls[pollName].option1) == keccak256(optionName)) {
polls[pollName].count1 += 1; }
if (keccak256(polls[pollName].option2) == keccak256(optionName)) {
polls[pollName].count2 += 1; }
if (keccak256(polls[pollName].option3) == keccak256(optionName)) {
polls[pollName].count3 += 1; }
}
function getPollName (string pollName) public view returns (string) {
require(doesPollExist(pollName));
return polls[pollName].question;
}
function getOption1 (string pollName) public view returns (string) {
require(doesPollExist(pollName));
```

```
return polls[pollName].option1;
}
function getOption2 (string pollName) public view returns (string) {
require(doesPollExist(pollName));
return polls[pollName].option2;
}
function getOption3 (string pollName) public view returns (string) {
require(doesPollExist(pollName));
return polls[pollName].option3;
}
function getPollCounts (string pollName) public view returns (uint[3]) {
require(doesPollExist(pollName));
return [polls[pollName].count1, polls[pollName].count2,
polls[pollName].count3];
}
function doesPollExist (string pollName) private view returns (bool) {
if (polls[pollName].exists) {
return true;
} else {
return false;
}
}
function hasAlreadyVoted (string pollName) private view returns (bool) {
if (polls[pollName].voted[msg.sender]) {
return false;
} else {
return true;
}}
}
```

## ПРИЛОЖЕНИЕ Б

### Информация для взаимодействия с разработанным смарт-контрактом

JSON Interface: [ { "constant": false, "inputs": [ { "name": "_addresses", "type": "address[]" } ], "name": "addManyWhitelistAddress", "outputs": [], "payable": false, "stateMutability": "nonpayable", "type": "function" }, { "constant": false, "inputs": [ { "name": "_question", "type": "string" }, { "name": "_name", "type": "string" }, { "name": "_option1", "type": "string" }, { "name": "_option2", "type": "string" }, { "name": "_option3", "type": "string" } ], "name": "createVoting", "outputs": [], "payable": false, "stateMutability": "nonpayable", "type": "function" }, { "constant": true, "inputs": [ { "name": "pollName", "type": "string" } ], "name": "getOption1", "outputs": [ { "name": "", "type": "string" } ], "payable": false, "stateMutability": "view", "type": "function" }, { "constant": false, "inputs": [ { "name": "_address", "type": "address" } ], "name": "addWhitelistAddress", "outputs": [], "payable": false, "stateMutability": "nonpayable", "type": "function" }, { "constant": true, "inputs": [ { "name": "", "type": "address" } ], "name": "whitelist", "outputs": [ { "name": "", "type": "bool", "value": false } ], "payable": false, "stateMutability": "view", "type": "function" }, { "constant": true, "inputs": [ { "name": "pollName", "type": "string" } ], "name": "getPollName", "outputs": [ { "name": "", "type": "string" } ], "payable": false, "stateMutability": "view", "type": "function" }, { "constant": true, "inputs": [ { "name": "pollName", "type": "string" } ], "name": "getOption3", "outputs": [ { "name": "", "type": "string" } ], "payable": false, "stateMutability": "view", "type": "function" }, { "constant": true, "inputs": [ { "name": "pollName", "type": "string" } ], "name": "getOption2", "outputs": [ { "name": "", "type": "string" } ], "payable": false, "stateMutability": "view", "type": "function" }, { "constant": false, "inputs": [ { "name": "optionName", "type": "string" }, { "name": "pollName", "type": "string" } ], "name": "vote", "outputs": [], "payable": false, "stateMutability": "nonpayable", "type": "function" }, { "constant": true, "inputs": [ { "name": "pollName", "type": "string" } ], "name": "getPollCounts", "outputs": [ { "name": "", "type": "uint256[3]", "value": [ "0", "0", "0" ] } ], "payable": false, "stateMutability": "view", "type": "function" } ];

contractName: Voting

deployedContract.adress:0x4638D059097Ee13d97aC51919519549E844cC473