Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра информационных и управляющих систем
Направление подготовки / специальность 09.04.04 Программная инженерия
Направленность (профиль) / специализация Управление разработкой программного обеспечения

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____ А.В. Бушманов
«___» _____2024 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему: Информационная система онлайн голосования на основе технологии цепочки блоков

Исполнитель
студент группы 2105-ом _____ (подпись, дата) Т.Е. Матеишен

Руководитель
доцент, канд. техн. наук _____ (подпись, дата) Л.В. Никифорова

Руководитель научного
содержания программы
магистратуры
профессор, доктор техн. наук _____ (подпись, дата) И.Е. Ерёмин

Нормоконтроль
доцент, канд. техн. наук _____ (подпись, дата) Т.А. Галаган

Рецензент
_____ (подпись, дата) И.С. Вирта

Благовещенск 2024

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра информационных и управляющих систем

УТВЕРЖДАЮ
Зав. кафедрой
_____ А.В. Бушманов
«_____» _____2024 г.

З А Д А Н И Е

К магистерской диссертации студента группы 2105-ом
Матеишена Тимофея Евгеньевича

1. Тема магистерской диссертации: Информационная система онлайн голосования на основе технологии цепочки блоков

(Утверждено приказом от 06.03.2024 № 632-уч)

2. Срок сдачи студентом законченной работы (проекта): 10.06.2024

3. Исходные данные к магистерской диссертации: Предметная область, проектная документация, отчёты по практической подготовке, перечень учебной литературы, интернет ресурсы.

4. Содержание магистерской диссертации: анализ предметной области; поиск существующих решений; моделирование объекта исследования

5. Перечень материалов приложения: (наличие чертежей, таблиц, графиков, схем, программных продуктов, иллюстративного материала и т.п.): UML схемы

6. Рецензент магистерской диссертации: И.С. Вирта

7. Дата выдачи задания 29.01.2024

Руководитель выпускной квалификационной работы: Никифорова Л.В. и. о. зав. кафедрой, доцент кафедры ИБ, канд. тех. наук, доцент

Задание принял к исполнению: _____

РЕФЕРАТ

Магистерская диссертация содержит 84 страницы, 14 рисунков, 2 приложения, 55 источников.

ИНФОРМАЦИОННАЯ СИСТЕМА, ПРОЕКТИРОВАНИЕ, АЛГОРИТМ, БЛОКЧЕЙН, ТЕСТИРОВАНИЕ, УМНЫЙ-КОНТРАКТ

Цель работы – создание информационной системы онлайн-голосования, которая использует возможности технологии блокчейн, для обеспечения безопасных, прозрачных и защищенных от несанкционированного доступа выборов.

В работе исследованы существующие решения данной задачи. Описаны основные аспекты проектирования и разработки системы и приложения. Была выбрана наиболее подходящая для реализации и соответствующая требованиям среда разработки. Также было проведено тестирование работоспособности проекта.

Задачи работы:

– изучение целесообразности и преимуществ использования технологии блокчейн в системах онлайн-голосования;

– определение функциональности информационной системы;

– формулирование требований к программному продукту;

– проектирование архитектуры системы и взаимодействия ее модулей;

– разработка децентрализованного приложения на блокчейне;

– интегрирование приложения MetaMask в качестве средства аутентификации и идентификации пользователей;

– разработка, отладка и тестирование системы, чтобы обеспечить ее надежность.

СОДЕРЖАНИЕ

Введение 6
1 Применение технологии блокчейн в системах онлайн-голосования 8
1.1 Предметная область исследования 8
1.1.1 Технология блокчейн 8
1.1.2 Подходы к разработке блокчейна 10
1.1.3 Актуальность темы исследования 11
1.2 Общая характеристика исследуемой задачи 13
1.2.1 Характеристика технологии «цепочка блоков» 13
1.2.2 Использование блокчейна в системах онлайн голосования 16
1.3 Обзор существующих методов решения задачи 22
2 Алгоритмическое и программное обеспечение решения задачи 27
2.1 Предлагаемый алгоритм решения задачи 27
2.1.1 Структура системы 27
2.1.2 Структура умного-контракта 30
2.2 Обзор возможностей профильного программного обеспечения 34
2.2.1 Подсистема WSL 35
2.2.2 Среда разработки Visual Studio 36
2.2.3 Библиотека React 37
2.2.4 Языки фронтенд разработки 38
2.2.5 Платформы Node.js и Express.js 39
2.2.6 Система управления базами данных MongoDB 40
2.2.7 Библиотека Ethers.js 40
2.2.8 Среда разработки Hardhat 41
2.2.9 Блокчейн Volta 42
2.2.10 Язык программирования Solidity 43
2.2.11 Криптовалютный кошелек Metamask 44
2.3 Обоснование выбора программно-технического обеспечения 46
2.3.1 Обоснование выбора блокчейна Volta 46

2.3.2 Обоснование выбора среды разработки Visual Studio 47
2.3.3 Обоснование выбора React, Node.js и Express.js 48
2.3.4 Обоснование выбора MongoDB 49
2.3.5 Обоснование выбора Hardhat и Ethers.js 50
2.3.6 Обоснование выбора Solidity 51
3 Программная реализация предлагаемого алгоритма решения задачи 53
3.1 Основные этапы практической разработки программного продукта 53
3.1.1 Программная реализация системы онлайн-голосования 54
3.1.2 Программная реализация умного контракта 59
3.1.2 Подключение к сети Volta 62
3.2 Результаты фактического тестирования программного продукта 64
3.3 Анализ достоверности и практической значимости результатов 69
Заключение 72
Библиографические ссылки 74
Библиографический список 76
Приложение А 81
Приложение Б 84

ВВЕДЕНИЕ

В современную цифровую эпоху технология блокчейн стала революционным решением для создания безопасных, прозрачных и децентрализованных систем. Одним из наиболее перспективных применений технологии блокчейн является онлайн-голосование. Традиционные системы голосования уже давно сталкиваются с такими проблемами, как фальсификация результатов голосования, отсутствие прозрачности и централизованного контроля. Однако появление систем онлайн-голосования на основе блокчейна может революционизировать методы проведения выборов и принятия коллективных решений.

Разработка систем онлайн-голосования, основанных на технологии блокчейн, направлена на устранение ограничений и уязвимостей традиционных методов голосования. Используя неотъемлемые характеристики блокчейна, такие как неизменность, прозрачность и децентрализация, эти системы могут обеспечить целостность и безопасность процесса голосования. Умные-контракты, которые представляют собой самоисполняющиеся контракты с условиями соглашения, непосредственно записанными в коде, играют решающую роль в автоматизации и обеспечении соблюдения правил процесса голосования.

Актуальность систем онлайн-голосования на основе блокчейна заключается в их способности повышать доверие, доступность и эффективность демократического процесса. В условиях растущей цифровизации различных аспектов общественной жизни важно изучить и внедрить безопасные и надежные методы проведения выборов и сбора общественного мнения. Использование технологии блокчейн в системах онлайн-голосования потенциально может повысить явку избирателей, снизить затраты, связанные с традиционными методами голосования, и обеспечить защищенную от несанкционированного доступа запись результатов голосования.

В данной работе рассматривается процесс разработки информационной системы онлайн-голосования с использованием технологии цепочки блоков. В ней рассматривается текущее состояние систем голосования, основанных на

блокчейне, проблемы и возможности, которые они представляют, а также ключевые соображения по проектированию и внедрению таких систем. В диссертации представлен обзор функциональных и архитектурных особенностей предлагаемой системы онлайн-голосования, включая использование смарт-контрактов, механизмов аутентификации пользователей и интеграцию с удобным интерфейсом.

Подробно описан процесс разработки системы онлайн-голосования, включая выбор подходящих блокчейн-платформ, языков программирования и инструментов разработки. В статье также представлены результаты тестирования и оценки производительности, безопасности и удобства использования системы.

Данная работа призвана внести вклад в расширение знаний в области технологии цепочки блоков и дать представление о практическом применении блокчейна для укрепления демократического процесса. Также выводы и рекомендации, описанные в работе, могут послужить основой для будущих исследований и разработок в области систем голосования на основе блокчейна.

1 ПРИМЕНЕНИЕ ТЕХНОЛОГИИ БЛОКЧЕЙН В СИСТЕМАХ ОНЛАЙН-ГОЛОСОВАНИЯ

1.1 Предметная область исследования

1.1.1 Технология блокчейн

Технология блокчейн – это децентрализованная распределенная система учета, которая обеспечивает безопасную и прозрачную запись транзакций через сеть компьютеров. Данная технология позволяет нескольким участникам (или узлам) вести и проверять общий цифровой реестр транзакций [1]. Принцип децентрализации означает отсутствие единого центрального органа, контролирующего данные. Каждый блок в блокчейне содержит запись транзакций с отметкой времени, которая криптографически связана с предыдущим блоком, образуя цепочку блоков. Данные распределяются по сети компьютеров, и каждый участник сохраняет копию всего блокчейна. Это обеспечивает целостность и неизменяемость данных, хранящихся в цепочке блоков, поскольку любая попытка изменить блок потребовала бы манипулирования последующими блоками, что упрощает обнаружение мошенничества.

Механизм блокчейна заключается в создании цепочки блоков, где каждый блок содержит набор транзакций. Блоки связаны друг с другом криптографическими хэшами [2], образуя неизменяемую хронологическую последовательность данных (рис. 1). Как только блок добавляется в блокчейн, изменение или подделка данных в нём чрезвычайно трудны.

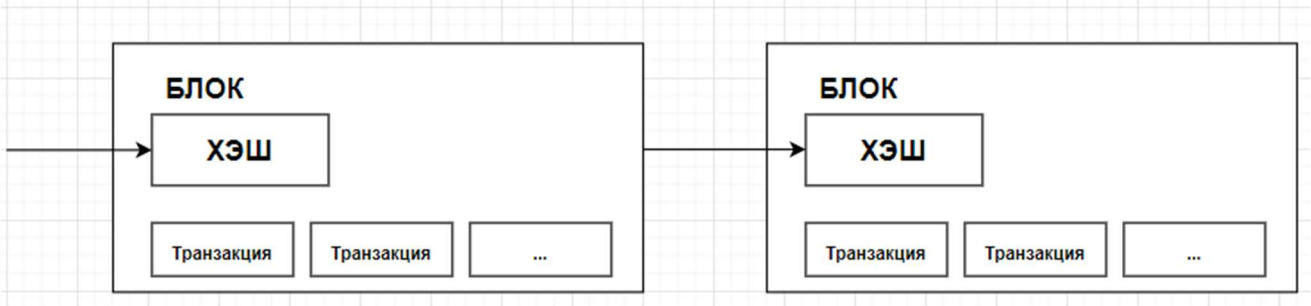

Рисунок 1 – Технология блокчейн

Использование технологии цепочки блоков имеет следующие преимущества:

– данные хранятся в сети компьютеров, что делает практически невозможным взлом или несанкционированное вмешательство (децентрализованное хранение);

– транзакции подписываются криптографически и проверяются, что обеспечивает их подлинность и целостность;

– после записи данные не могут быть изменены или удалены, что создает проверяемую и прозрачную историю;

– устраняет необходимость в сторонних посредниках. Прямое доверие между участниками устраняет зависимость от третьих сторон, снижая затраты и повышая эффективность;

– прозрачность, все участники имеют доступ к одним и тем же данным, что позволяет создавать четкую и проверяемую отчетность о транзакциях;

– движение активов и данных можно легко отслеживать и проверять, повышая подотчетность;

– умные-контракты позволяют автоматизировать сложные рабочие процессы, сокращая ручные усилия и количество ошибок;

– сокращает бумажную волокиту и дублирование записей. Все участники обмениваются унифицированными данными, удаляя копии;

– транзакции могут обрабатываться быстрее и эффективнее, снижая затраты.

Технология цепочки блоков также поддерживает умные-контракты, которые представляют собой самоисполняющиеся контракты с условиями соглашения, непосредственно записанными в код, что автоматизирует выполнение соглашения между двумя или более сторонами и устраняет необходимость в посредниках, снижая затраты и повышая доверие. Умные-контракты могут использоваться для автоматизации различных аспектов процесса голосования, таких как проверка правомочности избирателей, обеспечение анонимности и точный подсчет голосов.

Важно учитывать, что технология цепочки блоков все еще развивается, с определенными проблемами и ограничениями, требующими решения. Тем не менее, потенциальные преимущества обширны, и блокчейн готов оказать значительное влияние на различные отрасли в будущем.

1.1.2 Подходы к разработке блокчейна

Технология блокчейн быстро развивается, что приводит к появлению различных подходов, адаптированных к различным потребностям и вариантам использования. Поскольку внедрение блокчейна в различных отраслях продолжает расширяться, понимание различных подходов к разработке блокчейна становится критически важным. От публичных и частных блокчейнов до консорциумных моделей и инновационных решений, таких как побочные цепочки и платформы токенизации. Существует несколько подходов к разработке блокчейна, каждый из которых имеет свои особенности и варианты использования:

– публичные блокчейны: открытые и не требующие разрешения, позволяющие любому желающему присоединиться к сети и участвовать в ней. Децентрализованный механизм согласования, такой как Proof of Work (PoW) или Proof of Stake (PoS). Примеры: Bitcoin, Ethereum, Litecoin;

– частные блокчейны: разрешенные и ограниченные определенной группой участников. Централизованный контроль доступа к сети и механизм согласования. Подходит для использования на предприятиях и в консорциумах. Примеры: Hyperledger Fabric, R3 Corda;

– консорциумные блокчейны: частично разрешенные, управляемые группой организаций. Механизм консенсуса, управляемый предварительно выбранными узлами. Обеспечивает баланс между открытостью публичных блокчейнов и контролем частных блокчейнов. Примеры: Quorum, Corda Enterprise;

– гибридные блокчейны: объединяет элементы как публичных, так и частных блокчейнов. Обеспечивает избирательную прозрачность и конфиденциальность. Обеспечивает взаимодействие между различными блокчейн-сетями. Примеры: Dragonchain, XinFin;

– сайдчейны и решения уровня 2: отдельные блокчейны, которые взаимодействуют с основным блокчейном. Обеспечивает масштабируемость, конфиденциальность и индивидуальную настройку, одновременно повышая безопасность основного блокчейна. Примеры: Liquid Network (боковая цепочка Биткоина), Polygon (уровень 2 Ethereum);

– блокчейн как услуга (BaaS): облачные сервисы, которые предоставляют инфраструктуру и инструменты блокчейна. Позволяет разработчикам создавать и развертывать приложения блокчейна без управления базовой инфраструктурой. Примеры: Блокчейн Amazon Web Services (AWS), блокчейн-сервис Microsoft Azure;

– платформы токенизации: блокчейн-платформы, ориентированные на создание, выпуск и управление цифровыми токенами. Позволяет представлять реальные активы или утилиты на блокчейне. Примеры: Ethereum (токены ERC-20), Polymath (токены безопасности).

Выбор подхода к разработке блокчейна зависит от таких факторов, как желаемый уровень децентрализации, требования к конфиденциальности, масштабируемости и конкретный вариант использования. У каждого подхода есть свои компромиссы и соображения, и разработчики должны тщательно оценить свои требования, прежде чем выбрать тот или иной подход.

1.1.3 Актуальность темы исследования

На сегодняшний день традиционные системы голосования сталкиваются с многочисленными уязвимостями, включая мошенничество, манипуляции и ошибки. Растущий спрос на удобство и доступность для избирателей дополнительно требует альтернативных решений. Но в контексте информационной системы онлайн-голосования, основанной на технологии цепочки блоков, каждый поданный голос регистрируется как транзакция в блокчейне, обеспечивая безопасную и защищенную от несанкционированного доступа запись о голосовании. Это повышает прозрачность и доверие к процессу голосования, позволяя заинтересованным сторонам независимо проверять достоверность результатов.

Разработка и внедрение информационной системы онлайн-голосования,

основанной на технологии цепочки блоков, имеет большое значение в сфере современного управления и избирательных процессов. Обеспечение целостности, безопасности и прозрачности процедур голосования имеет первостепенное значение для поддержания демократических ценностей, и технология блокчейн представляет собой многообещающее решение для устранения уязвимостей, присутствующих в традиционных системах голосования.

Используя технологию блокчейн, система онлайн-голосования может предложить повышенные меры безопасности для защиты от несанкционированного доступа, мошенничества и киберугроз. Децентрализованная и неизменяемая природа блокчейн-реестра гарантирует, что благодаря криптографическим механизмам защиты, голоса защищены от подделки, манипулирования или несанкционированного доступа. Каждый голос точно регистрируется и остается неизменным, повышая доверие к процессу голосования.

Более того, прозрачность, обеспечиваемая этой технологией, позволяет повысить доверие среди избирателей, должностных лиц избирательных комиссий и других заинтересованных сторон. Возможность независимой проверки и аудита записей о голосовании способствует подотчетности и снижает вероятность возникновения споров или оспаривания результатов выборов.

Использование данной технологии в информационной системе онлайн-голосования также упрощает процесс голосования, потенциально повышая явку избирателей и их участие в голосовании. Благодаря принципу доступности и защищенным механизмам цифрового голосования отдельные лица могут получить больший доступ к осуществлению своих избирательных прав, преодолевая географические барьеры и логистические ограничения (например, лица с ограниченными возможностями передвижения), часто связанные с традиционными методами голосования.

Кроме того, интеграция умных-контрактов в систему голосования может автоматизировать определенные ручные процессы, повышая эффективность и точность регистрации избирателей, аутентификации, упростить подсчет голосов и подведение итогов, что приведет к более быстрым и эффективным выбо-

рам.

В целом, внедрение информационной системы онлайн-голосования, основанной на технологии цепочки блоков, представляет собой прогрессивный шаг на пути модернизации избирательной практики, укрепления демократии и вселения уверенности в целостность избирательной системы.

1.2 Общая характеристика исследуемой задачи

1.2.1 Характеристика технологии «цепочка блоков»

Технология «цепочка блоков» – это выстроенная по определённым правилам непрерывная последовательная цепочка блоков (связный список), содержащих информацию. Связь между блоками обеспечивается не только нумерацией, но и тем, что каждый блок содержит свою собственную хеш-сумму (результат обработки неких данных хеш-функцией, то есть функция, осуществляющая преобразование массива входных данных произвольной длины в выходную битовую строку установленной длины [3], выполняемое определённым алгоритмом) и хеш-сумму предыдущего блока. Изменение любой информации в блоке изменит его хеш-сумму. Чтобы соответствовать правилам построения цепочки, изменения хеш-суммы нужно будет записать в следующий блок, что вызовет изменения уже его собственной хеш-суммы. При этом предыдущие блоки не затрагиваются. Если изменяемый блок последний в цепочке, то внесение изменений может не потребовать существенных усилий. Но если после изменяемого блока уже сформировано продолжение, то изменение может оказаться крайне трудоёмким процессом. Дело в том, что обычно копии цепочек блоков хранятся на множестве разных компьютеров независимо друг от друга.

Блок транзакций – специальная структура для записи группы транзакций в системе. Транзакция считается завершённой и достоверной («подтверждённой»), когда проверены её формат и подписи, и когда сама транзакция объединена в группу с несколькими другими и записана в специальную структуру – блок. Содержимое блоков может быть проверено, так как каждый блок содержит информацию о предыдущем блоке. Все блоки выстроены в одну цепочку, которая содержит информацию обо всех совершённых когда-либо операциях в

базе. Самый первый блок в цепочке – первичный блок – рассматривается как отдельный случай, так как у него отсутствует родительский блок.

Блок состоит из заголовка и списка транзакций. Заголовок блока включает в себя свой хеш [2], хеш предыдущего блока, хеши транзакций и дополнительную служебную информацию. Для транзакций в блоке используется древовидное хеширование.

Созданный блок будет принят остальными пользователями, если числовое значение хеша заголовка равно или меньше определённого целевого числа, величина которого периодически корректируется. Так как результат хеширования функции SHA-256 (рис. 2) считается необратимым, на данный момент нет алгоритма получения желаемого результата, кроме случайного перебора. Если хеш не удовлетворяет условию, то в заголовке изменяется параметр nonce и хеш пересчитывается. Обычно (статистически) требуется большое количество пересчётов. Когда вариант найден, узел рассылает полученный блок другим подключенным узлам, которые проверяют блок. Если ошибок нет, то блок считается добавленным в цепочку и следующий блок должен включить в себя его хеш.

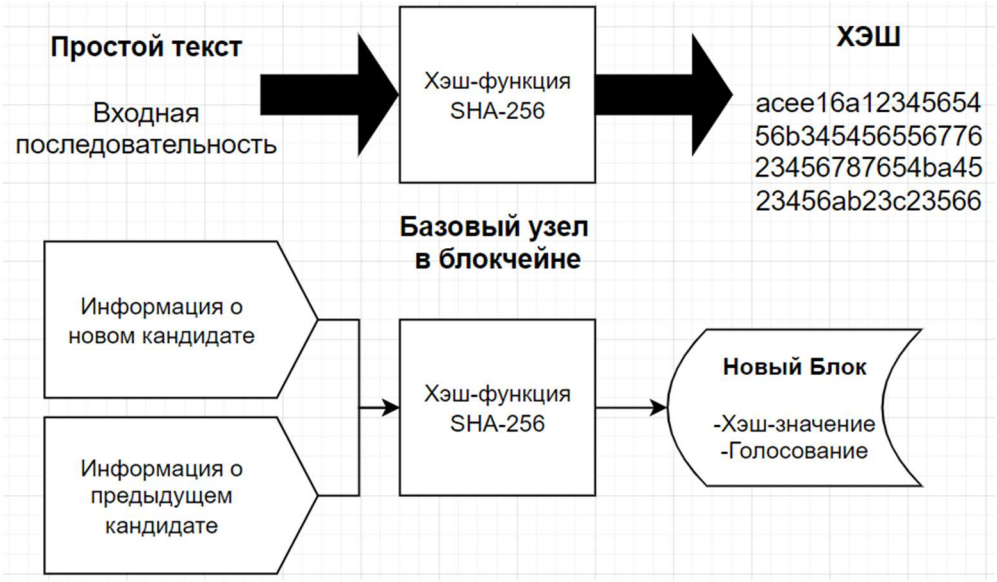

Рисунок 2 – Работа алгоритма SHA-256

Блоки одновременно формируются множеством «майнеров». Удовлетворяющие критериям блоки отправляются в сеть, включаясь во все репликации распределённой базы блоков. Регулярно возникают ситуации, когда несколько

новых блоков в разных частях распределённой сети называют предыдущим один и тот же блок, то есть цепочка блоков может ветвиться. Специально или случайно можно ограничить ретрансляцию информации о новых блоках (например, одна из цепочек может развиваться в рамках локальной сети). В этом случае возможно параллельное наращивание различных ветвей. В каждом из новых блоков могут встречаться как одинаковые транзакции, так и разные, вошедшие только в один из них. Когда ретрансляция блоков возобновляется, майнеры начинают считать главной цепочку с учётом уровня сложности хеша и длины цепочки. При равенстве сложности и длины предпочтение отдаётся той цепочке, конечный блок которой появился раньше. Транзакции, вошедшие только в отвергнутую ветку (в том числе по выплате вознаграждения), теряют статус подтверждённых. Транзакции получения вознаграждения за создание отсечённых блоков не дублируются в другой ветке.

Блокчейн формируется как непрерывно растущая цепочка блоков с записями обо всех транзакциях. Копии базы или её части одновременно хранятся на множестве компьютеров и синхронизируются согласно формальным правилам построения цепочки блоков. Информация в блоках не шифрована и доступна в открытом виде, но отсутствие изменений удостоверяется криптографически через хеш-цепочки (элемент цифровой подписи).

База публично хранит в незашифрованном виде информацию о всех транзакциях, подписываемых с помощью асимметричного шифрования. Для предотвращения многократной траты одной и той же суммы используются метки времени, реализованные путём разбиения базы данных (БД) на цепочку специальных блоков, каждый из которых, в числе прочего, содержит в себе хеш предыдущего блока и свой порядковый номер. Каждый новый блок осуществляет подтверждение транзакций, информацию о которых содержит и дополнительное подтверждение транзакций во всех предыдущих блоках цепочки. Изменять информацию в блоке, который уже находится в цепи, не практично, так как в таком случае пришлось бы редактировать информацию во всех последу-

ющих блоках. Благодаря этому успешная double-spending атака (повторная трата ранее израсходованных средств) на практике крайне маловероятна.

Чаще всего умышленное изменение информации в любой из копий базы или даже в достаточно большом количестве копий не будет признано истинным, так как не будет соответствовать правилам. Некоторые изменения могут быть приняты, если будут внесены во все копии базы (например, удаление нескольких последних блоков из-за ошибки в их формировании).

За требование к хешам блоков отвечает специальный параметр, называемый «сложность». Так как вычислительные мощности сети непостоянны, этот параметр пересчитывается клиентами сети через каждые 2016 блоков таким образом, чтобы поддерживать среднюю скорость формирования блокчейна на уровне 2016 блоков в две недели. Таким образом, 1 блок должен создаваться примерно раз в десять минут. На практике, когда вычислительная мощность сети растёт – соответствующие временные промежутки короче, а когда снижается – длиннее. Перерасчёт сложности с привязкой ко времени возможен благодаря наличию в заголовках блоков времени их создания. Оно записывается в Unix-формате по системным часам автора блока (если блок создается в пуле, то по системным часам сервера этого пула).

1.2.2 Использование блокчейна в системах онлайн голосования

Онлайн голосование – термин, определяющий различные виды голосования, охватывающий как электронные средства голосования (электронная демократия), так и технические электронные средства подсчёта голосов. Разновидностями электронного голосования являются Интернет-выборы, телефонный сервис телеголосование и использования блокчейн-технологии [4].

Технология электронного голосования позволяет ускорить процесс подсчёта голосов, а также упростить голосование людям с ограниченными возможностями.

Голосование, как процесс принятия решений по важнейшим для общества вопросам, известно человечеству тысячи лет, но население росло, общество развивалось и в процессе принятия решений стали принимать участие всё

больше людей – сотни тысяч и даже миллионы. Тогда на смену таким простым методам подсчёта голосов пришли более сложные: стали создаваться избирательные участки, амфоры заменили на урны для голосования, а шарики – на избирательные бюллетени. Хотя такой подход к голосованию позволил учитывать мнение огромных масс населения, он имеет ряд минусов.

В связи с этим электронные устройства для голосования и само электронное голосование не стали общемировой практикой. Более того, многие страны (Нидерланды, Великобритания, Германия), в которых изначально электронное голосование получило широкое распространение, в итоге ограничили их применение из-за несовершенства технологии и вернулись к более надёжному аналоговому методу голосования.

Тем не менее, развитие технологий и общества подталкивает человечество к поиску более дешевых и надёжных методов голосования. Применение технологии блокчейн в голосовании может решить большую часть проблем существующих избирательных систем.

В основе технологии блокчейн лежит транзакционная модель: у каждого пользователя есть кошелёк, с уникальными публичным и приватным ключами, которыми он подтверждает любое изменение данных. Вся информация о транзакциях хранится в последовательно записанных блоках, таким образом, хэш данных предыдущего блока входит в данные следующего. Так обеспечивается неизменность данных – изменение любого блока автоматически сделает невалидными все последующие. Блокчейн хранит всю информацию о всех транзакциях в полном объеме одновременно на всех узлах, и она не может быть изменена или удалена. Наиболее широкое применение блокчейн нашел именно в сфере регистрации данных о движении имущественных прав на некие цифровые объекты – на этой идее построены все современные криптовалюты. Такие объекты, называемые обычно «монетами» или «токенами», могут либо создаваться автоматически, по заранее оговоренному алгоритму, в процессе эксплуатации системы («майниться») либо выпускаться имеющим на это права участником системы.

Идея применения блокчейна в системах голосования напрашивается сама собой: блокчейн позволяет заменить древнюю технологию голосования путем передачи кому-то своего голоса, выраженного физическим объектом (шариком нужного цвета, бумажным бюллетенем и т.п.), на передачу цифрового токена. При этом, как и во многих других случаях перехода бизнес-процесса из физического мира в цифровой, резко сокращаются транзакционные издержки и повышается доступность системы.

В контексте разработки информационной системы онлайн-голосования, основанной на технологии блокчейн, общие характеристики предметной области можно изложить следующим образом:

– использование технологии блокчейн обеспечивает высокий уровень безопасности и целостности в процессе онлайн-голосования. Шифрование избирательных бюллетеней и информации об избирателях, наряду с децентрализованным характером блокчейна, защищает от несанкционированного доступа и подделки;

– благодаря прозрачной и неизменяемой природе записей блокчейна система онлайн-голосования повышает прозрачность, позволяя заинтересованным сторонам проверять целостность процесса голосования. Кроме того, технология блокчейн обеспечивает повышенный доступ к более широкому кругу избирателей, потенциально повышая участие избирателей;

– используя децентрализованную архитектуру блокчейна, система онлайн-голосования уменьшает зависимость от центрального органа власти, укрепляя доверие и подотчетность среди участников. Каждая транзакция, записанная в блокчейне, прозрачна и отслеживаема, обеспечивая подотчетность на протяжении всего жизненного цикла голосования;

– система уделяет приоритетное внимание конфиденциальности и анонимности избирателей, используя методы шифрования для защиты личных данных избирателей при сохранении конфиденциальности отдельных голосов. Это защищает от принуждения, манипуляций и внешнего вмешательства в процесс голосования;

– дизайн системы подчеркивает эффективность и проверяемость за счет оптимизации административных задач, автоматизации процессов с помощью умных-контрактов и обеспечения надежной основы для аудита и проверки результатов голосования. Использование смарт-контрактов повышает операционную эффективность и снижает вероятность человеческих ошибок;

– используя криптографические алгоритмы (SHA-256, SHA-3 и ECDSA) и механизмы консенсуса, система значительно снижает риски мошеннических действий, двойного голосования и несанкционированных модификаций. Неизменность записей блокчейна обеспечивает постоянство и действительность транзакций голосования;

– информационная система онлайн-голосования, построенная на технологии блокчейн, устойчива и масштабируема, способна принимать большое количество голосов при сохранении стандартов производительности и безопасности. Ее устойчивость к кибератакам и утечке данных способствует долгосрочной жизнеспособности системы.

Существующие проблемы при внедрении систем онлайн-голосования на блокчейне:

– децентрализованная проверка голосов в распределенной сети может быть дорогостоящей с точки зрения вычислений;

– интеграция систем голосования на основе блокчейна с устаревшей инфраструктурой голосования может быть сложной и дорогостоящей;

– отсутствие правовой базы: во многих странах нет законов, напрямую регулирующих использование технологии блокчейн на выборах, что создает неопределенность и риск;

– отсутствие согласованных стандартов для систем голосования, основанных на блокчейне, может препятствовать внедрению и функциональной совместимости;

– достижение общественного доверия к безопасности, надежности и добросовестности систем онлайн-голосования, независимо от используемой технологии, остается препятствием;

– ключевым вопросом остается обеспечение баланса между анонимностью избирателей и необходимостью поддающейся проверке идентификации;

– решение проблемы потенциального лишения избирательных прав избирателей, не имеющих доступа к технологиям или подключению к Интернету.

Несмотря на эти проблемы, потенциальные преимущества блокчейна для онлайн-голосования значительны. Дальнейшие исследования и разработки в сочетании с ответственным внедрением могут проложить путь к более безопасному, прозрачному и доступному процессу голосования для всех.

Разработка системы онлайн-голосования на блокчейне требует безопасного и эффективного способа достижения консенсуса, обеспечивающего защиту от несанкционированного доступа и поддающиеся проверке выборы. Вот разбивка четырёх популярных механизмов консенсуса и их пригодность для онлайн-голосования:

– доказательство работы (PoW) – требуется решение сложных математических головоломок, вознаграждение получает первый решивший. Это стимулирует участие и обеспечивает безопасность сети за счет вычислительных затрат [5]. Высокая безопасность, децентрализация;

– подтверждение доли (PoS) – валидаторы выбираются на основе количества криптовалюты, которой они владеют ("доля") [6]. Менее энергоемкий, чем PoW. Энергоэффективность, более быстрые транзакции, потенциал для децентрализации;

– подтверждение полномочий (PoA) – выбранные валидаторы с заранее определенными идентификационными данными и репутацией проверяют транзакции [7]. Централизованный, но более быстрый и масштабируемый. Скорость, масштабируемость, подходит для сетей с разрешениями (например, для правительственных выборов);

– подтверждение истории (PoH) – использует проверяемую случайную функцию для генерации временных меток внутри блоков, защищая сеть

без вычислительных затрат [8]. Энергоэффективность, потенциально более быстрая, чем PoW, требует особых соображений при реализации.

В таблице 1 показана пригодность механизмов консенсуса для системы онлайн-голосования.

Таблица 1 – Сравнения механизмов консенсуса

Наименование	Безопасность	Децентрализация	Энергоэффективность	Масштабируемость	Прозрачность
Proof of Work (PoW)	Очень высокая	Высокая	Низкая	Средняя	Низкая
Proof of Stake (PoS)	Высокая	Средняя	Высокая	Высокая	Средняя
Proof of Authority (PoA)	Высокая	Низкая	Высокая	Высокая	Высокая
Proof of History (PoH)	Средняя	Высокая	Очень высокая	Высокая	Средняя

На основе таблицы можно сделать такие выводы о механизмах консенсуса:

– доказательство работы (PoW) не идеален из-за значительно высокого энергопотребления и потенциальных проблем с масштабируемостью, которые могут препятствовать эффективному расширению;

– подтверждение доли (PoS) может подходить в зависимости от реализации. Требует тщательного проектирования для обеспечения безопасности и распределения долей участия;

– подтверждение полномочий (PoA) подходит для сетей разрешений с доверенными органами, может вызвать опасения по поводу цензуры;

– подтверждение истории (PoH) потенциал для будущих применений требует дальнейших исследований и разработок систем голосования;

Таким образом, интеграция технологии блокчейн в информационную систему онлайн-голосования революционизирует электоральный ландшафт, укрепляя доверие, прозрачность и эффективность проведения выборов, а иде-

альным выбором для реализации онлайн голосования на основе технологии цепочки блоков является механизм консенсуса подтверждение полномочий (PoA).

1.3 Обзор существующих методов решения задачи

Растущий спрос на удобство и доступность голосования привел к изучению систем онлайн-голосования. Однако проблемы, связанные с безопасностью, прозрачностью и анонимностью избирателей, ограничивают широкое распространение. Технология цепочки блоков, с присущими ей характеристиками неизменности, децентрализации и прозрачности, предлагает потенциальное решение этих проблем. В работе рассматриваются существующие подходы к онлайн-голосованию с использованием технологии блокчейн, выделяются их сильные стороны, ограничения и реальные реализации.

Системы, основанные на публичной технологии блокчейн:

– Voatz (Эстония) были проведены небольшие испытания для местных выборов в Эстонии. Поддерживает функции геймификации для поощрения участия. Результаты показываются в режиме реального времени. На рисунке 3 показана работа приложения.

– Agora (Швейцария) используется для внутрипартийных выборов внутри политической партии в Швейцарии. Является децентрализованной платформой, устойчивой к цензуре. Основана на технологии блокчейн для голосования, защищенного от несанкционированного доступа.

Сильными сторонами является: высокая прозрачность и возможность публичной проверки процесса голосования; высокая неизменяемость и устойчивость к вмешательству; потенциал для расширения участия избирателей, особенно географически распределенных лиц [9].

Ограничения таких систем заключается: в проблемах с масштабируемостью из-за ограничений по размеру блока и перегрузки сети; в потенциальных проблемах с конфиденциальностью, если голоса регистрируются непосредственно в публичном реестре; в уязвимостях к атакам Sybil, когда злоумышленники пытаются получить контроль над сетью.

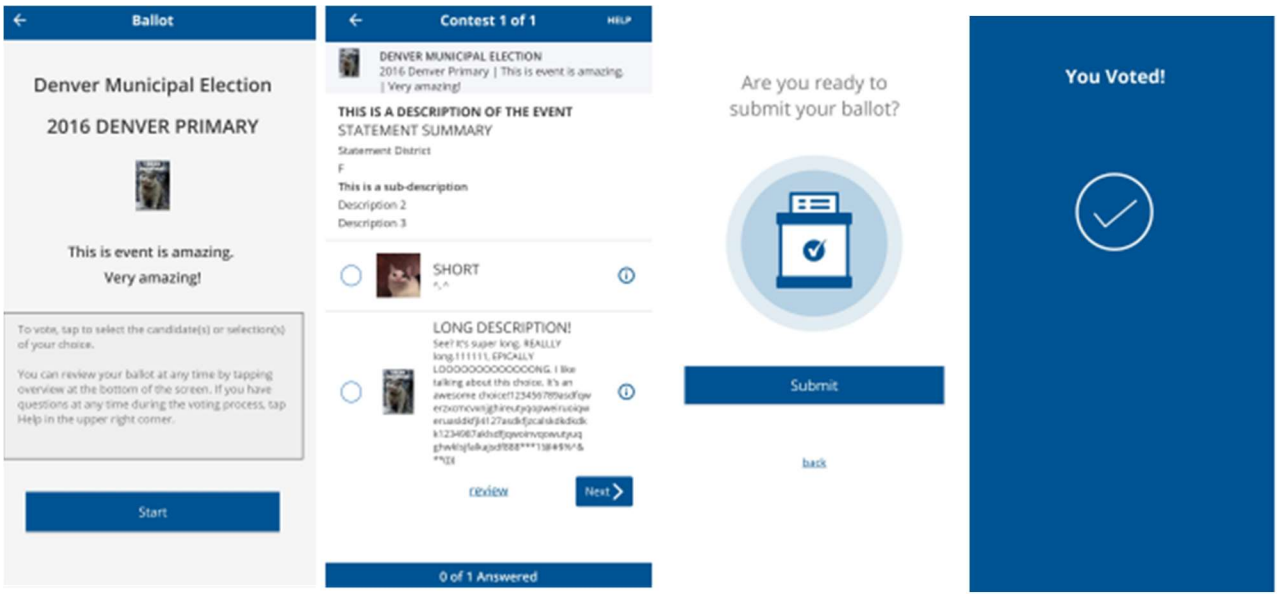

Рисунок 3 – Мобильная платформа Voatz

Системы, основанные на разрешенной (частной) технологии блокчейн:

– FollowMyVote (США). В первую очередь ориентирован на просвещение и верификацию избирателей с ограниченной функциональностью фактического голосования (образовательные ресурсы и поддержка). Интегрируется с существующими системами голосования.

– Блокчейн-сеть голосования (Индия). Представляет собой пилотный проект для внутрипартийных выборов.

Сильными сторонами является: повышенная масштабируемость и производительность по сравнению с общедоступными блокчейнами; больший контроль над доступом участников и управлением идентификационными данными; возможность внедрения механизмов сохранения конфиденциальности.

Ограничения таких систем: для управления сетью требуется доверенный орган, что потенциально влияет на децентрализацию; меньшая прозрачность и проверяемость по сравнению с публичными блокчейнами.

Гибридный подход:

– Фонд децентрализованного голосования (стадия концепции) предлагает гибридную модель с zk-SNARKs для голосования с сохранением конфиденциальности.

Сильными сторонами является: использование общедоступных блокчейнов для прозрачности и проверяемости ключевых этапов (например, регистрация избирателей, объявление результатов); использование разрешенных блокчейнов, обеспечивает конфиденциальность и масштабируемость.

Ограничения таких систем: повышенная сложность проектирования и внедрения системы; требуется тщательная настройка для обеспечения баланса между потребностями прозрачности и конфиденциальности.

В таблице 2 представлен SWOT-анализ существующих систем онлайн-голосования, основанных на технологии блокчейн.

Таблица 2 – SWOT-анализ существующих систем онлайн-голосования

Особенность	Voatz	Agora	FollowMyVote
Сильные стороны	- Удобный интерфейс - Широкий выбор типов вопросов - Анонимное голосование	- Особое внимание безопасности и конфиденциальности - Открытый исходный код (прозрачность)	- Фокус на ранжированном голосовании, способствующем более справедливым результатам
Слабые стороны	- Полагается на централизованные серверы, уязвимые для атак - Могут использоваться в недемократических целях	- Сложный интерфейс для некоторых пользователей - Малоизвестен за пределами технических кругов	- Полагается не на автономное решение - Может не подходить для крупномасштабных выборов
Возможности	- Партнерство с правительствами и организациями - Интеграция с платформами социальных сетей	- Расширение сферы корпоративного управления и принятия решений - Партнерство с НПО и активистами	- Интеграция с системами управления выборами - Ввод в образовательные учреждения и общественные группы
Угрозы	- Государственное регулирование онлайн-голосования - Нарушения безопасности и хакерские атаки	- Сложный нормативный ландшафт для технологии блокчейн - Проблемы масштабирования для крупномасштабных выборов	- Зависимость от существующих систем голосования, которые могут не измениться

Опираясь на SWOT-анализ Voatz, Agora и FollowMyVote, можно выделить ключевые аспекты, которые следует учитывать при разработке системы онлайн-голосования на основе блокчейна:

– прозрачность и проверяемость: использование общедоступных элементов блокчейна для ключевых этапов, таких как регистрация избирателей и объявление результатов, аналогично Voatz;

– масштабируемость и конфиденциальность: внедрение авторизованной системы голосования на блокчейне для обеспечения эффективной работы и конфиденциальности пользователей;

– ограниченные варианты использования: выход за рамки внутренних выборов, таких как Agora, изучая пилотные программы с избирательными органами, подобные FollowMyVote.

– уменьшение зависимости от централизованных функций, которые ставят под угрозу децентрализацию, как показано на примере FollowMyVote, путем изучения и внедрения альтернативных подходов, таких как гибридные модели или механизмы распределенного управления.

– разработка комплексных и масштабируемых программ обучения избирателей, как показано в FollowMyVote, но с учетом конкретной системы и целевой аудитории;

– исследование гибридных подходов, таких как «Децентрализованный фонд голосования», которые сочетают публичные и разрешенные блокчейны для обеспечения оптимальной прозрачности и конфиденциальности;

– активное взаимодействие с регулирующими органами, чтобы понять и устранить юридические препятствия, потенциально сотрудничая с ними в рамках пилотных программ;

– интеграция технологий, повышающие конфиденциальность, такие как zk-SNARKs, для решения проблем конфиденциальности при сохранении возможности проверки, вдохновленные концепцией децентрализованного фонда голосования.

Хотя системы онлайн-голосования, основанные на технологии блокчейн, предлагают многообещающие решения, практические проблемы остаются. Необходимы дальнейшие исследования и разработки для решения проблем масштабируемости, конфиденциальности и безопасности, прежде чем можно будет рассматривать возможность широкого внедрения. Кроме того, тщательное изучение законодательной и нормативно-правовой базы, потребностей пользователей и интеграции с существующими системами имеет решающее значение для успешного внедрения.

2 АЛГОРИТМИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ РЕШЕНИЯ ЗАДАЧИ

2.1 Предлагаемый алгоритм решения задачи

Проект включает в себя два взаимосвязанных компонента: информационную систему и блокчейн-сеть. Блокчейн работает как децентрализованный реестр, который облегчает безопасную и неизменяемую запись и извлечение данных. В этом качестве он служит базовой инфраструктурой для хранения и проверки данных. С другой стороны, информационная система функционирует как пользовательский интерфейс, который обрабатывает и представляет записанные данные из блокчейна. Ее роль включает визуализацию данных и обеспечение взаимодействия пользователя с системой посредством различных интерактивных функций. Плавная координация между блокчейном и информационной системой обеспечивает сплоченную экосистему, в которой целостность данных поддерживается блокчейном, в то время как информационная система облегчает взаимодействие с пользователем [10].

2.1.1 Структура системы

Прогресс в области технологий привел к разработке сложной системы онлайн-голосования, основанной на технологии цепочки блоков. Эта система обеспечивает безопасность, прозрачность и добросовестность избирательных процессов за счет использования криптографических принципов. В разрабатываемой системе каждый пользователь, допущенный к процессу голосования входит в систему со своим уникальным адресом кошелька и имеет на счёте токен для голосования. Каждый пользователь может отдать голос только за одного кандидата. Администратор системы входит в систему со своим уникальным адресом кошелька, наделенного привилегиями для инициирования голосования и добавления кандидатов.

Модули разрабатываемой информационной системы (ИС) представлены на рисунке 4.

Рисунок 4 – Диаграмма DFD2 разрабатываемой системы

Компонентами разрабатываемой информационной системы онлайн-голосования с использованием технологии цепочки блоков являются:

– модуль «Администратор» предназначен для администратора ИС, ответственного за управление системой голосования и надзора за ней. Он предоставляет функциональные возможности для настройки и мониторинга процесса голосования;

– модуль «Избиратель» предназначен для лиц, имеющих право участвовать в процессе голосования и отдавать свой токен голосования за понравившегося кандидата. Этот модуль позволяет избирателям безопасно удостоверять свою личность, используя учетные данные, усиливая меры безопасности системы и гарантируя, что в избирательном процессе принимают участие только законные пользователи. Он предоставляет функциональные возможности, связанные с процессом голосования, и обеспечивает целостность и сохранность результатов голосования.

В сложной структуре информационной системы онлайн-голосования, использующей технологию цепочки блоков, пользователи взаимодействуют с системой, используя адреса кошельков Metamask, каждый из которых символизирует уникальную личность, привязанную к конкретному пользователю. Эти адреса кошельков, обладающие своей индивидуальностью, служат шлюзом для

доступа пользователей к системе, при этом их соответствующие данные надежно хранятся в базе данных системы, обеспечивая усиленный уровень аутентификации и авторизации. Основа системы опирается на неизменяемый реестр технологии блокчейн, гарантирующий целостность и достоверность процесса голосования.

Чтобы начать этап голосования, системный администратор инициирует создание умного-контракта внутри системы, записывая его в блокчейн. Первостепенным для процесса инициализации умного-контракта является обязательство администратора оговорить срок действия этого контракта в блокчейне, определяя продолжительность, отведенную избирателям для беспрепятственного голосования. Одновременно администратор обладает прерогативой быстро составлять список кандидатов на этапе создания умного-контракта или откладывать эту задачу на более поздний срок, используя универсальную функциональность, предоставляемую страницей системного администратора. Примечательно, что системный администратор, отличающийся уникальным адресом, закрепленным в сети блокчейн, защищает неприкосновенность системы от потенциального вмешательства или брешей, обеспечивая неуязвимую системную архитектуру.

После создания умного-контракта инициализируется ключевое событие, контракт встраивается в блокчейн-структуру, защищенную от любых попыток изменения или манипулирования. Это придает системе атмосферу прозрачности и неизменности, создавая доверительную среду для беспрепятственного развития демократического процесса. Наделенный функциональностью смарт-контракта, администратор имеет возможность регистрировать новых кандидатов для участия в голосовании вплоть до истечения срока действия контракта в блокчейне, усиливая динамичный характер системы при одновременном соблюдении принципов подотчетности и честности. Процесс работы показан на рисунке А.1 приложения А.

При входе в систему, система на уровне уникальных адресов пользователя понимает, что если пользователь является избирателем, то для него полно-

стью скрыта страница администратора системы. Каждый пользователь имеет право отдать только один голос за одного кандидата, после подтверждения записи, отданный голос записывается в блокчейн и не может быть изменен. Процесс голосования избирателя в системе показан на рисунке А.2 приложения А.

2.1.2 Структура умного-контракта

В рамках этого проекта разрабатывается система, использующая умный-контракт на основе данных, передаваемых системным администратором. После развертывания этот контракт неизменно регистрируется в инфраструктуре блокчейна для организации процесса голосования и контроля за ним.

Умные-контракты, по своей сути самоисполняющиеся соглашения, поддерживаемые в цепочке блоков, служат программируемыми объектами, которые автоматически обеспечивают соблюдение условий соглашения при выполнении заранее определенных условий. Устраняя необходимость в посредниках, эти контракты упрощают выполнение соглашений, обеспечивая мгновенную проверку результатов для всех вовлеченных сторон. Это позволяет избежать временных задержек и уменьшает зависимость от внешних арбитров, позволяя участникам обладать полной властью над контрактными обязательствами.

Переменные состояния в умных-контрактах инкапсулируют ключевые состояния контракта, воплощая важные точки данных, которые определяют ход выполнения соглашения. Функции определяют допустимые действия, которые может предпринять контракт, реагируя на триггеры, такие как события или взаимодействия пользователя, одновременно манипулируя переменными состояния для определения операционного поведения контракта. Модификаторы, функционирующие как механизмы контроля доступа, устанавливают предварительные условия для выполнения функций, укрепляя протоколы безопасности и предотвращая несанкционированные изменения состояния контракта. События, с другой стороны, означают уведомления, отправляемые контрактом при выполнении определенных действий, обеспечивая канал связи в экосистеме блокчейна для уведомления соответствующих заинтересованных сторон.

Умные контракты в рамках блокчейн-инфраструктур предлагают много-

гранные преимущества. Они автоматизируют выполнение соглашений без учета межпартийного доверия, повышая уровень доверия между потенциально незнакомыми субъектами без участия центральных посредников. Примечательно, что прозрачность и неизменяемость, присущие технологии цепочки блоков, сохраняют целостность кода и последовательность выполнения, предотвращая несанкционированное вмешательство и обеспечивая нестираемую запись транзакций. Повышение эффективности достигается за счет автоматизации, способствующей упорядочению процессов и соразмерному сокращению эксплуатационных расходов и распределения времени.

Разработка смарт-контрактов требует использования определенных парадигм программирования и хорошо зарекомендовавших себя алгоритмов (хэш-функции: SHA-256 и SHA-3; цифровой подписи: ECDSA и RSA). Криптографические методы, такие как хеширование, создают уникальные отпечатки данных, гарантируя целостность данных и предотвращая несанкционированный доступ к данным. Цифровые подписи обеспечивают криптографическую валидацию, повышая возможность проверки подлинности и предотвращая незаконные взаимодействия с контрактами. Кроме того, механизмы консенсуса, поддерживаемые распределенными алгоритмами, обеспечивают единодушие всех участников блокчейн-сети в отношении действительности и исполнения контрактов, укрепляя доверие к соглашению и его функциональную совместимость в рамках децентрализованной экосистемы.

Алгоритм работы умного-контракта включает такие этапы:

– определить соглашение: несколько сторон определяют возможность сотрудничества и желаемые результаты, и соглашения могут включать бизнес-процессы, обмен активами и т.д.;

– установить условия: умные-контракты могут быть инициированы самими сторонами или при выполнении определенных условий, таких как индексы финансового рынка, события, такие как местоположение по GPS и т.д.;

– бизнес-логика кода: пишется компьютерная программа, которая будет выполняться автоматически при выполнении условных параметров;

– шифрование и технология блокчейна: шифрование обеспечивает безопасную аутентификацию и передачу сообщений между сторонами, относящимися к смарт-контрактам;

– выполнение и обработка: в итерации блокчейна всякий раз, когда между сторонами достигается консенсус относительно аутентификации и верификации, выполняется код, а результаты запоминаются для соответствия требованиям и проверки;

– сетевые обновления: после выполнения смарт-контрактов все узлы сети обновляют реестр, чтобы отразить новое состояние. Как только запись опубликована и проверена в сети блокчейн, ее нельзя изменить, она находится только в режиме добавления.

Общий алгоритм работы умного-контракта показан на рисунке 5.

Рисунок 5 – Работа смарт-контракта

Разрабатываемый умный контракт в данной работе, должен быть создан с использованием языка программирования Solidity, специально разработанного для Ethereum и подобных сетей [11]. На рисунке А.3 приложения А показано, какие ключевые моменты и функции для инициализации голосования должен содержать контракт.

В области структур данных контракт инкапсулирует жизненно важные объекты, такие как структура «Кандидат», воплощая в себе существенные атрибуты, такие как имена кандидатов и соответствующие подсчеты голосов. По-

следовательный массив «кандидаты» служит хранилищем, в котором хранятся экземпляры структуры «Кандидат», включающей всех кандидатов, участвующих в избирательном процессе. Более того, адрес «владельца» означает инициатора контракта, наделенного административными привилегиями, в то время как карта «избирателей» тщательно отслеживает активность голосования, сохраняя соответствующие адреса и их соответствующие статусы голосования. Кроме того, временные аспекты инкапсулируются с помощью временных меток «Начало голосования» и «Окончание голосования», определяющих начало и завершение периода голосования.

При развертывании контракта функция-конструктор управляет процессом инициализации, принимая параметры, обозначающие имена кандидатов и продолжительность интервала голосования. Эта функция тщательно организует создание экземпляров объектов-кандидатов для каждого указанного имени в предоставленном массиве, впоследствии заполняя массив «кандидатов». Кроме того, он четко определяет инициатора контракта как «владельца» и устанавливает временные метки «votingStart» и «votingEnd» в зависимости от преобладающей временной метки блока и указанной продолжительности.

Операционные директивы дополнительно усиливаются за счет включения модификатора «onlyOwner», ограничивающего конкретные функциональные возможности, доступные исключительно владельцу контракта. Заслуживающие внимания функции, такие как: «addCandidate» позволяющая владельцу добавлять новых кандидатов на выборы даже после развертывания, но только в том случае, если они еще не проголосовали; «vote» позволяющая избирателю отдать свой голос за конкретного кандидата, идентифицированного по его индексу в массиве «кандидаты» и выполняет проверки, чтобы убедиться, что пользователь не голосовал ранее (предоставленный индекс действителен); «getVotingStatus» проверяет текущую временную метку на соответствие «votingStart» и «votingEnd», чтобы определить, продолжается ли голосование; «getRemainingTime» вычисляет оставшееся время в периоде голосования, если оно продолжается в данный момент или выдает ошибку, если голосование еще

не началось.

Подводя итог, разрабатываемый умный-контракт воплощает фундаментальную архитектуру, направленную на создание бесшовной системы онлайн-голосования на основе парадигмы блокчейна Ethereum.

2.2 Обзор возможностей профильного программного обеспечения

Для реализации децентрализованного приложения голосования будут использоваться среда разработки Visual Studio и командный язык Bash для взаимодействия с подсистемой Linux (так как блокчейн-сети, такие как Ethereum и подобные, построены поверх систем на базе Linux). Технологии используемые для реализации системы показаны на рисунке 6.

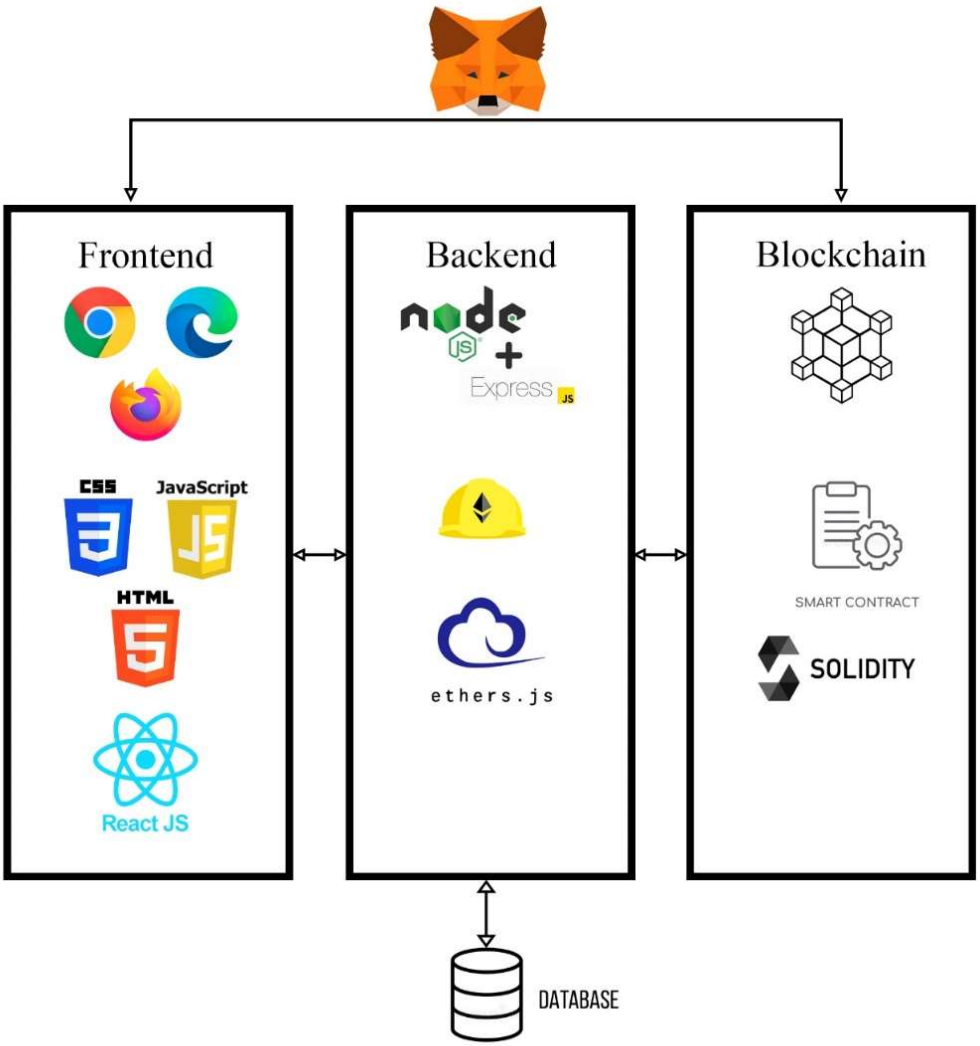

Рисунок 6 – Взаимосвязь используемого ПО в проекте

На рисунке 7 показано взаимодействие приложения, базы данных и сторонних удаленных сервисов.

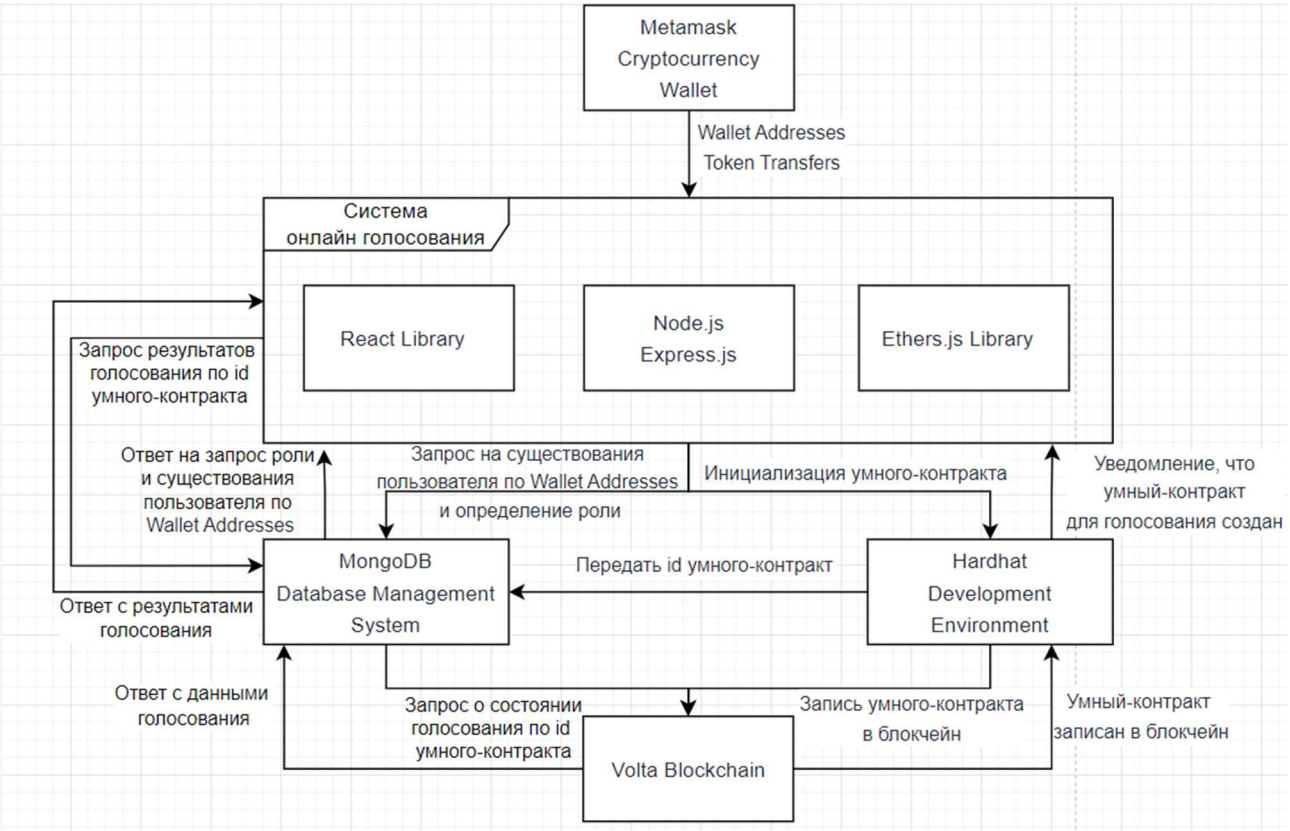

Рисунок 7 – Взаимодействие сервисов в проекте

2.2.1 Подсистема WSL

Для создания системы необходимо установить подсистему WSL (для пользователей Windows) или работать непосредственно на устройстве с ОС Linux. Подсистема WSL (Windows для Linux [12]) позволяет использовать преимущества Linux для развертывания смарт-контрактов и разработки приложений с использованием фреймворка React на компьютере под управлением Windows.

Развертывание смарт-контрактов заключается в том, что WSL предоставляет Linux-подобную среду в Windows, позволяя разработчикам устанавливать и использовать популярные инструменты разработки блокчейнов, такие как Truffle, Hardhat и web3.js, которые предназначены для Linux. Это позволяет разработчикам, знакомым с командной строкой Linux, легко взаимодействовать с этими инструментами для выполнения таких задач, как компиляция, развертывание и взаимодействие со смарт-контрактами.

WSL обеспечивает совместимость и помогает избежать проблем при развертывании контрактов в блокчейн-сетях, поскольку многие из них основаны

на инфраструктуре и инструментах Linux. Кроме того, WSL позволяет устанавливать Node.js и npm в среде Linux, что позволяет использовать широкий спектр библиотек и инструментов React наряду с другими средствами разработки Linux.

WSL предлагает гибкую рабочую среду, позволяющую разработчикам использовать свои любимые текстовые редакторы или IDE в Windows, одновременно используя среду Linux для выполнения конкретных задач. Это помогает уменьшить расхождение между Windows и Linux, предоставляя разработчикам, ориентированным на Linux, привычную среду для работы над блокчейн-проектами и приложениями React на своих ПК с Windows.

2.2.2 Среда разработки Visual Studio

Visual Studio от Microsoft – это интегрированная среда разработки, облегчающая создание приложений на различных платформах и устройствах [13]. Она поддерживает Windows Forms, UWP, веб-технологии и мобильные платформы, такие как.NET, включая разработку для iOS и Android посредством интеграции с Xamarin. Набор инструментов включает в себя редактирование кода с помощью IntelliSense, универсальные опции отладки, редакторы форм, веб-страниц, классов и схем баз данных. Сторонние расширения могут быть интегрированы для расширения функциональности и поддержки систем контроля версий.

Упрощена разработка приложений и игр для устройств Windows, Android, iOS и Интернета. Интеграция с Azure позволяет разрабатывать облачные приложения, а разнообразные инструменты с открытым исходным кодом помогают создавать современные веб-приложения. Платформа удовлетворяет потребности в разработке Office и облегчает разработку и развертывание баз данных SQL.

Используя несколько языков, таких как C++, JavaScript с Node.js и Python, Visual Studio обеспечивает производительность, масштабируемость и гибкость для различных устройств и типов приложений. Python особенно полезен для кроссплатформенного написания сценариев, веб-сервисов, приложений Интер-

нета вещей и исследования данных.

2.2.3 Библиотека React

React – это библиотека JavaScript для создания пользовательских интерфейсов с компонентами, позволяет создавать различные типы приложений, такие как одностраничные, мобильные или серверные приложения, с помощью таких фреймворков, как Next.js и Express.js.

React предоставляет такие возможности:

– способствует созданию пользовательских интерфейсов с повторно используемыми компонентами, улучшая организацию кода и обслуживание. Объединение более мелких компонентов позволяет эффективно создавать сложные интерфейсы;

– используя виртуальное представление DOM, React быстро сравнивает и обновляет только необходимые части реального DOM, обеспечивая более быстрое взаимодействие с пользователем;

– упрощает разработку пользовательского интерфейса с помощью декларативного подхода. Разработчики определяют состояние пользовательского интерфейса, а React управляет обновлениями и рендерингом, повышая ясность кода и скорость разработки;

– извлекает выгоду из обширного сообщества, предлагающего различные библиотеки и инструменты. Обширная документация облегчает разработку, эффективно удовлетворяя разнообразные потребности;

– используют SSR (Server Side Rendering) и SSG (Static Site Generation) для создания удобных для SEO (Search Engine Optimization) функций, улучшая видимость в поисковых системах и удобство начальной загрузки для пользователей;

– используя принципы React, React Native позволяет создавать собственные приложения для iOS и Android с использованием общих компонентов JavaScript и React. Такой кроссплатформенный подход повышает производительность разработчиков.

Таким образом, React позволяет разработчикам создавать динамичные,

интерактивные и масштабируемые веб-приложения с акцентом на возможность повторного использования и производительность. Надежная экосистема и поддержка сообщества укрепляют репутацию React как универсального решения для разработки современного пользовательского интерфейса.

2.2.4 Языки фронтенд разработки

Трио CSS (Cascading Style Sheets), HTML (HyperText Markup Language) и JavaScript играет жизненно важную роль в формировании веб-ландшафта системы. CSS определяет стили, HTML структурирует контент, а JavaScript обеспечивает динамическую интерактивность веб-сайтов, а React служит платформой, которая легко интегрирует эти технологии для современной веб-разработки.

CSS имеет решающее значение для определения стиля веб-документов, написанных в HTML или XML (eXtensible Markup Language), образуя основной элемент Всемирной паутины наряду с HTML и JavaScript.

HTML – это стандартный язык разметки веб-контента, структурирует и представляет информацию в браузерах, часто дополняемый функциональными возможностями CSS и JavaScript.

JavaScript – это язык веб-программирования, который обеспечивает динамическое поведение в Интернете и широко используется почти на всех веб-сайтах, обрабатывая интерактивные элементы и сторонние инструменты.

В React CSS стилизует элементы пользовательского интерфейса, использует библиотеки CSS-in-JS для повторно используемых компонентов и обеспечивает адаптивный дизайн. HTML структурирует содержимое пользовательского интерфейса, интегрирует JSX для бесшовного выравнивания компонентов React и фокусируется на семантических элементах для обеспечения доступности и SEO. JavaScript управляет динамическими операциями, обработкой данных и контролем состояния в React, используя логическую изоляцию компонентов и возможные дополнительные библиотеки, такие как Redux или MobX, для сложного управления состоянием.

В заключение отметим, что гармоничная интеграция CSS, HTML,

JavaScript и React позволяет разработчикам создавать визуально привлекательные, интерактивные и масштабируемые веб-приложения, которые улучшают пользовательский опыт и функциональность.

2.2.5 Платформы Node.js и Express.js

Сочетание Node.js, Express.js и React.js предлагает мощный и универсальный подход к созданию современных веб-приложений.

Node.js служит кроссплатформенной средой выполнения JavaScript, облегчая написание сценариев на стороне сервера и создание API.

Express.js – это платформа веб-сервера, упрощает создание приложений с функциями маршрутизации и промежуточного программного обеспечения. Он предназначен для создания веб-приложений и API. Его называют де-факто стандартной серверной платформой для Node.js.

Вот как Node.js и Express.js можно использовать в сочетании с React для создания системы голосования:

– Node.js выступает в качестве серверной среды выполнения для серверной логики в JavaScript, включая взаимодействие с базой данных (например, MySQL, MongoDB), создание API и рендеринг на стороне сервера.

– Express.js, основанный на Node.js, упрощает разработку веб-приложений с помощью маршрутизации, промежуточного программного обеспечения и создания шаблонов.

– React.js – это интерфейсная библиотека для разработки пользовательского интерфейса, включающая архитектуру компонентов, виртуальный DOM и управление состоянием.

Эта комбинация, часто называемая стеком MERN (MongoDB, Express, React и Node.js), является популярным выбором для создания динамических и интерактивных веб-приложений. Таким образом, Node.js, и Express.js являются ценными инструментами для создания систем с библиотекой React. Они обеспечивают четкое разделение задач, эффективную разработку интерфейса и мощную серверную платформу для вашего приложения.

2.2.6 Система управления базами данных MongoDB

MongoDB – это система управления базами данных документов NoSQL, она хранит данные в гибких документах, подобных JSON (JavaScript Object Notation), которые могут содержать различные типы данных и структуры. Эта гибкость делает ее хорошо подходящей для хранения и управления сложными, неструктурированными или часто изменяющимися данными.

MongoDB можно использовать в приложении React для:

– управления входом пользователя, сохраняя учетные данные для входа в MongoDB, защищая вход через сервер Node.js и предоставляя React возможность управлять пользовательским интерфейсом на основе ролей пользователей;

– использования уникальные адреса кошельков для входа в систему, добавив поле "walletAddress" в MongoDB и подтвердив вход с использованием адреса кошелька;

– реализации ролей пользователей и контроля доступа, сохранив роль в документе пользователя, применив управление доступом на основе ролей в React и ограничив компоненты пользовательского интерфейса для каждой роли.

2.2.7 Библиотека Ethers.js

Ethers.js – это библиотека JavaScript, специально разработанная для взаимодействия с блокчейном Ethereum и его экосистемой. Она позволяет разработчикам создавать различные приложения и инструменты, связанные блокчейном, включая:

– предоставляет разработчикам функциональные возможности для создания децентрализованных приложений (это приложения, которые работают в распределенной сети, устраняя необходимость в центральном органе) и взаимодействия с ними в блокчейне;

– предлагает инструменты и функциональные возможности, помогающие разработчикам писать, развертывать смарт-контракты и взаимодействовать с ними в сети;

– может использоваться для создания кошельков для хранения токе-

нов и управления ими. Популярные кошельки, такие как MetaMask и Tally, используют преимущества Ethers.js за их функциональность.

Ключевыми особенностями Ethers.js является:

– предоставляет простой и мощный API, который позволяет разработчикам легко взаимодействовать с Ethereum и подобными блокчейн-сетями;

– несмотря на свою функциональность, Ethers.js является относительно небольшой библиотекой, что делает ее эффективной и подходящей для различных сред разработки;

– поддерживает как Node.js, так и веб-браузеры, предлагая разработчикам гибкость в выборе среды разработки;

– обеспечивает приоритетность безопасности и позволяет разработчикам сохранять свои личные ключи в безопасности в своих клиентских приложениях.

В целом, Ethers.js является популярным и мощным инструментом для разработчиков, стремящихся создавать приложения и инструменты, взаимодействующие с блокчейном.

2.2.8 Среда разработки Hardhat

Hardhat – это среда разработки профессионального уровня для Ethereum и аналогичных блокчейн-сетей. Он предоставляет комплексную платформу для оптимизации процесса создания, тестирования, развертывания и отладки смарт-контрактов и децентрализованных приложений (dApps) в блокчейне.

Возможности Hardhat:

– автоматизирует повторяющиеся задачи, связанные с разработкой смарт-контрактов, такие как компиляция, развертывание, тестирование и отладка;

– поставляется со встроенной локальной сетью Ethereum, называемой Hardhat Network. Это позволяет разработчикам тестировать и отлаживать свои смарт-контракты и dApps в своей локальной среде разработки без необходимости развертывать их в реальной основной сети Ethereum или тестовых сетях;

– поддержка языка Solidity, который является основным языком про-

граммирования, используемым для написания смарт-контрактов на блокчейне. Hardhat предоставляет инструменты для эффективной компиляции, отладки и тестирования кода Solidity;

– позволяет разработчикам разветвлять состояние основной сети Ethereum или тестовых сетей, создавая локальную копию для целей тестирования. Это позволяет им взаимодействовать с реальными данными и сценариями, не влияя на сам блокчейн;

– совместимость с такими инструментами, как MetaMask и Truffle, для бесшовной интеграции.

Подводя итог, Hardhat служит удобной для разработчиков средой, которая упрощает и рационализирует весь процесс создания и развертывания блокчейн приложений. Его функциональные возможности помогают разработчикам экономить время, писать более надежные смарт-контракты и обеспечивать их функционирование по назначению перед развертыванием в общедоступном блокчейне.

2.2.9 Блокчейн Volta

Тестовая сеть Volta – это тестовая среда, подключенная к блокчейну Energy Web Chain (EWC), в первую очередь для тестирования и развертывания смарт-контрактов перед запуском в основной сети EWC [14]. Разработчики используют Volta аналогично "промежуточному серверу" при разработке программного обеспечения для обеспечения надлежащей функциональности. Он работает со своим собственным токеном, Volta Token (хранящийся в цифровом кошельке MetaMask), который покрывает транзакционные издержки в сети, но не имеет реальной ценности.

Ключевые возможности Volta:

– тестирования умных-контрактов перед их внедрением в производственную цепочку. Поскольку токены в тестовой сети (токены Volta) не имеют реальной ценности, то можно провести тщательное тестирование без каких-либо затрат;

– тестирование обновлений протокола перед их внедрением в произ-

водственную цепочку. Каждый раз, когда происходит обновление протокола Ethereum, он сначала тестируется в сети Volta.

Служебным токеном для тестовой сети Volta является токен Volta. Токены Volta не имеют реальной ценности, но они понадобятся для "оплаты" транзакционных издержек, связанных с взаимодействием со смарт-контрактами в сети Volta.

Блокчейн выполняет три ключевые функции в EW-DOS:

– предоставляет механизм смарт-контрактов для хранения децентрализованных идентификационных данных (DID);

– облегчает проверку внутри цепочки и транзакции между сторонами;

– выполняет смарт-контракты, которые используются децентрализованными приложениями, SDK и пакетами утилит EW-DOS.

Сеть хранит следующую информацию:

– умные-контракты для децентрализованных идентификационных данных (DID), которые создаются с помощью библиотеки управления идентификацией и доступом EW-DOS;

– умные-контракты, которые регулируют согласованное поведение валидатора и вознаграждение. Они известны как системные контракты;

– умные-контракты, реализующие другие сетевые протоколы Ethereum, такие как разрешение доступа и открытые клиентские протоколы Ethereum;

– умные-контракты, содержащие логику и функциональность, специфичные для приложений, развернутых в сети Energy Web, и пакетов утилит, которые подключают их и пользователей к сети Energy Web.

2.2.10 Язык программирования Solidity

Solidity – это специализированный язык высокого уровня для разработки самоисполняющихся умных-контрактов на блокчейнах, таких как Ethereum, обеспечивающий безопасность и прозрачность транзакций.

Структура создания смарт-контракта для системы онлайн-голосования с помощью Solidity состоит из:

– регистрация избирателей: избиратели могут зарегистрироваться, взаимодействуя с контрактом, предоставляя свою уникальную идентификационную информацию и контракт может надежно хранить эту информацию в блокчейне;

– подача голосов: избиратели могут проголосовать, отправив транзакцию в контракт с идентификатором выбранного ими кандидата и контракт может подтвердить регистрацию избирателя, предотвращая дублирование голосов;

– подсчет голосов и результаты: контракт может автоматически подсчитывать голоса по мере их поступления и выдавать окончательные результаты прозрачно хранятся в блокчейне и доступны любому.

В целом, Solidity предлагает инновационный подход к созданию умных-контрактов для безопасных и прозрачных систем онлайн-голосования. Несмотря на существующие проблемы, он обещает будущее безопасных и поддающихся проверке выборов.

2.2.11 Криптовалютный кошелек Metamask

MetaMask служит популярным криптовалютным кошельком для управления Ethereum и другими токенами.

MetaMask действует как мост между пользователем и миром децентрализованных приложений (dApps), которые представляют собой приложения, построенные на блокчейнах. Это позволяет пользователям взаимодействовать с этими приложениями, например, принимать участие в голосовании или участвовать в протоколах DeFi (децентрализованные финансы) [15].

В системе онлайн-голосования, основанной на блокчейне, использующей MetaMask для подписи транзакций, процесс подписания транзакции для передачи голосов за кандидата включает в себя такие шаги:

– пользователь получает доступ к системе онлайн-голосования и выбирает предпочтительного кандидата для голосования;

– онлайн-система подготавливает транзакцию для передачи голоса кандидату на блокчейне;

– интеграция с MetaMask для взаимодействия с блокчейном Volta;

– MetaMask отображает запрос подтверждения, содержащий подробную информацию о транзакции, включая получателя, количество голосов для перевода, плату за газ (комиссию сети) и другую соответствующую информацию;

– пользователь подписывает транзакцию, подтверждая данные паролем;

– MetaMask надежно подписывает транзакцию, используя закрытый ключ для авторизации;

– подписанная транзакция транслируется в сеть Volta с помощью MetaMask;

– майнеры проверяют транзакцию для подтверждения в блоке;

– транзакция подтверждается, когда она получает достаточное количество подтверждений блока;

– обратная связь с подтверждением предоставляется онлайн-системой после успешной обработки и записи в цепочку блоков.

На рисунке 8 показан кошелёк MetaMask с уникальным адресом в сети Volta, содержащим 50 токенов VT на балансе.

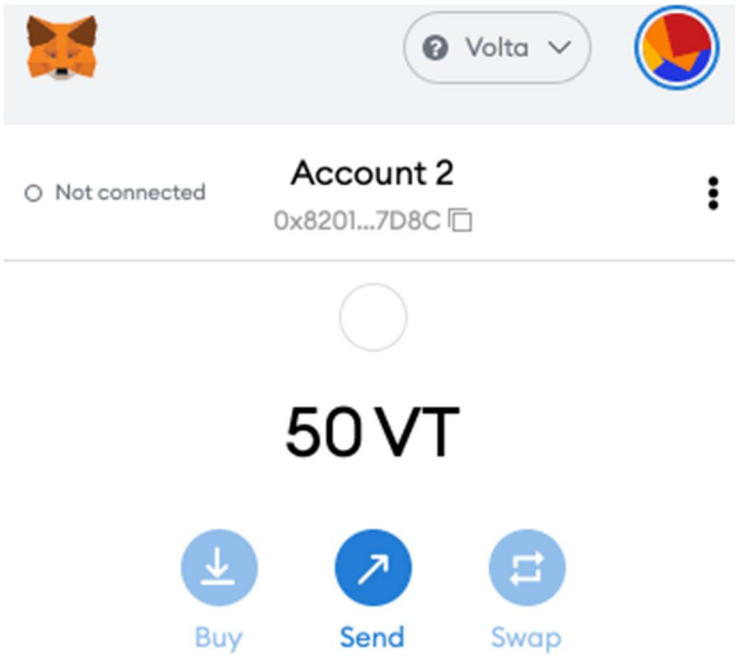

Рисунок 8 – Криптовалютный кошелек Metamask

2.3 Обоснование выбора программно-технического обеспечения

2.3.1 Обоснование выбора блокчейна Volta

Сеть Volta предлагает разработчикам безопасную и экономичную среду тестирования, ускоряющую процесс разработки при одновременном снижении рисков, связанных с развертыванием смарт-контрактов в основной сети.

Преимуществами использования сети Volta является:

– предоставляет безопасную и экономичную среду для тестирования смарт-контрактов и протоколов перед их развертыванием в основной сети Energy Web Chain. Тестирование на Volta позволяет разработчикам выявлять и исправлять ошибки, не неся реальных затрат, связанных с развертыванием в основной сети;

– обеспечивает более быстрые циклы разработки, позволяя разработчикам быстро повторять и тестировать изменения, не влияя на основную сеть. Это может значительно ускорить процесс разработки и увеличить время вывода приложений на рынок;

– предоставляет бесплатный доступ к таким ресурсам, как токены Volta, которые используются для покрытия комиссионных за транзакции в тестовой сети. Это устраняет финансовые барьеры и позволяет любому участвовать в тестировании и разработке в сети Energy Web.

Недостатками сети является:

– в качестве тестовой сети Volta может не обладать всеми функциями, доступными в основной сети EWC. Это потенциально может ограничить объем тестирования для некоторых приложений;

– результаты, наблюдаемые в тестовой сети Volta, не всегда могут идеально соответствовать основной сети EWC из-за потенциальных различий в конфигурации или условиях сети.

В целом, несмотря на ограничения, Volta предлагает значительные преимущества разработчикам, работающим в сети Energy Web. Возможность тестирования и итераций в безопасной и экономичной среде может значительно улучшить процесс разработки и обеспечить надежность приложений, разверну-

тых в основной сети. Однако крайне важно понимать ограничения Volta и тщательно учитывать потребности в тестировании конкретного проекта.

2.3.2 Обоснование выбора среды разработки Visual Studio

Visual Studio предоставляет комплексную интегрированную среду разработки (IDE), оснащенную мощными инструментами, надежными возможностями отладки и бесшовной интеграцией с широким спектром языков программирования, что делает ее оптимальным выбором для разработчиков, стремящихся к эффективности и продуктивности в своих задачах кодирования.

Преимуществами выбора Visual Studio для реализации проекта:

– легко интегрируется с подсистемой Windows для Linux, позволяя разработчикам использовать возможности как сред Windows, так и Linux в едином рабочем пространстве. Это обеспечивает доступ к более широкому спектру инструментов и функциональных возможностей, что особенно полезно для проектов, использующих специфичные для Linux инструменты или библиотеки;

– поддерживает широкий спектр языков программирования, включая HTML, CSS, JavaScript и Solidity. Такая универсальность позволяет разработчикам работать над различными проектами веб-разработки, включая интерфейсные и серверные компоненты, в одной среде, устраняя необходимость переключения между различными инструментами;

– предлагает функциональные возможности для распараллеливания кода, позволяющие разработчикам оптимизировать выполнение определенных разделов кода, распределяя задачи между несколькими ядрами или процессорами. Это может значительно повысить производительность при выполнении задач с высокой интенсивностью вычислений;

– может похвастаться полным набором функций, предназначенных для улучшения процесса разработки.

Недостатками Visual Studio является:

– может быть ресурсоемкой, особенно на старых или менее мощных машинах.

– благодаря широкому набору функций, Visual Studio может иметь более крутую кривую обучения по сравнению с некоторыми более простыми редакторами кода.

Visual Studio представляет собой мощную и универсальную среду разработки для проектов, использующих различные языки программирования и потенциально требующих распараллеливания кода. Бесшовная интеграция с WSL еще больше расширяет ее возможности, но пользователям следует учитывать требования к ресурсам, сложность и потенциальные затраты при выборе подходящей версии для своих проектов.

2.3.3 Обоснование выбора React, Node.js и Express.js

Преимуществами использования React, Node.js и Express.js связки в рамках проекта являются:

– четкое разделение задач. React обрабатывает пользовательский интерфейс, в то время как Node.js и Express.js обрабатывают внутреннюю логику и коммуникации. Это делает кодовую базу более поддерживаемой и масштабируемой;

– мощный серверный фреймворк. Express.js предоставляет надежную и эффективную платформу для создания ваших серверных функций;

– каждый уровень может быть развернут и обновлен независимо, что упрощает обслуживание и ускоряет циклы выпуска;

– компонентная система React позволяет создавать модульные и повторно используемые элементы пользовательского интерфейса, способствуя сопровождаемости кода;

– виртуальный DOM React оптимизирует производительность за счет минимизации количества требуемых фактических манипуляций с DOM, что приводит к более плавному и отзывчивому пользовательскому интерфейсу;

– синтаксис JSX упрощает создание макетов пользовательского интерфейса за счет сочетания HTML, CSS и JavaScript, делая разработку более интуитивной и эффективной;

– обширная экосистема React предлагает множество инструментов и библиотек для таких задач, как управление состоянием, маршрутизация, тестирование и отладка, оптимизация разработки пользовательского интерфейса и обеспечение качества кода;

– Express.js облегчает создание надежных и масштабируемых API-интерфейсов, предоставляя функции маршрутизации и промежуточное программное обеспечение для эффективной обработки запросов;

– Node.js управляемый событиями, неблокирующий характер делает его хорошо подходящим для создания приложений реального времени.

2.3.4 Обоснование выбора MongoDB

Выбор правильной системы управления базами данных имеет решающее значение для успеха любого проекта. Когда дело доходит до выбора базы данных, которая обеспечивает гибкость, масштабируемость и легкую интеграцию с современными приложениями, MongoDB выделяется как популярный выбор как среди разработчиков, так и среди предприятий.

Преимущества использования MongoDB:

– документно-ориентированная структура MongoDB обеспечивает гибкость для хранения разнообразных данных, позволяя организовать данные без схемы или полуструктурированные данные в соответствии с потребностями вашего приложения;

– эффективно обрабатывает большие объемы данных;

– предлагает разнообразные варианты выполнения запросов, включая объектно-ориентированные операторы запросов и платформы агрегации, для эффективного поиска данных и манипулирования ими;

– известен своей скоростью и масштабируемостью, что делает его подходящим выбором для обработки больших наборов данных и сложных прикладных требований;

– ориентированный на документы. Хорошо сочетается с объектно-ориентированными моделями программирования, такими как React.

2.3.5 Обоснование выбора Hardhat и Ethers.js

Hardhat и Ethers.js образуют мощную комбинацию для разработки систем на основе блокчейна, особенно в Ethereum и совместимых сетях. Популярность их использования заключается:

– Hardhat предлагает оптимизированную среду разработки с инструментами развертывания. Блокчейн локальной разработки, который позволяет тестировать и развертывать смарт-контракты на локальном узле блокчейна, экономя время и затраты по сравнению с использованием общедоступной сети;

– Hardhat упрощает процесс развертывания в тестовых и основных сетях, оптимизируя переход от разработки к производству;

– Hardhat легко настраивается, позволяя разработчикам адаптировать среду разработки;

– Hardhat хорошо интегрируется с различными библиотеками и инструментами блокчейна, позволяя разработчикам использовать предпочитаемые ими технологии в рамках рабочего процесса разработки;

– Ethers.js предоставляет библиотеку JavaScript для низкоуровневого взаимодействия с Ethereum и подобными сетями. Обеспечивая детальный контроль, разработчики имеют прямой доступ к функциональным возможностям сети;

– Ethers.js используется для развертывания умных-контрактов и взаимодействие с ними, считывания данных из блокчейна, отправка и получение транзакций, создание пользовательских инструментов и библиотек.

Преимущества использования Hardhat и Ethers.js вместе:

– эффективный процесс разработки. Среда разработки Hardhat в сочетании с низкоуровневыми функциональными возможностями Ethers.js способствует упрощению рабочего процесса;

– гибкость и кастомизация. Этот дуэт позволяет разработчикам создавать разнообразные блокчейн-приложения, адаптированные к их конкретным потребностям.

В целом, Hardhat и Ethers.js предлагают мощную комбинацию для разработки блокчейн- приложения, особенно на Ethereum и совместимых сетях.

2.3.6 Обоснование выбора Solidity

Solidity – это язык программирования высокого уровня, который в основном используется для написания смарт-контрактов на блокчейн-платформах. Он разработан, чтобы облегчить разработчикам создание безопасных и надежных децентрализованных приложений, предоставляя такие функции, как строгая типизация, наследование и математические операции.

Solidity является привлекательным выбором для разработки умных контрактов благодаря нескольким ключевым факторам:

– интеграция с виртуальной машиной Ethereum (EVM). Solidity была специально разработана для компиляции в байт-код. Эта встроенная совместимость упрощает разработку и обеспечивает интеграцию с блокчейн-экосистемой;

– широкое внедрение EVM на различных блокчейн-платформах еще больше расширяет охват и функциональную совместимость умных-контрактов на основе Solidity;

– Solidity в значительной степени заимствован из устоявшихся языков программирования, таких как JavaScript, Python и C++. Это знакомство облегчает разработчикам, имеющим опыт работы с этими языками;

– существуют готовые решения для различных функций, что экономит время разработки;

– Solidity прошел обширное тестирование и аудит, укрепив свои позиции в качестве зрелого и безопасного языка для разработки смарт-контрактов.

Преимущества использования языка Solidity:

– синтаксис и структура языка способствуют удобочитаемости кода и сопровождаемости, облегчая разработчикам понимание и модификацию существующих умных-контрактов;

– Solidity предлагает высокоуровневые абстракции для сложных

функциональных возможностей, позволяя разработчикам сосредоточиться на основной логике разрабатываемых умных-контрактов, не увязая в низкоуровневых деталях;

– Solidity применяет определенные методы безопасного кодирования, такие как проверка типов и механизмы контроля доступа, которые помогают снизить потенциальные уязвимости в умных-контрактах.

В целом, уникальное сочетание совместимости с EVM, удобства для разработчиков, большого сообщества и внимания к безопасности делает Solidity привлекательным выбором для создания умных-контрактов.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРЕДЛАГАЕМОГО АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ

3.1 Основные этапы практической разработки программного продукта

Предлагаемый алгоритм решения задачи разработки информационной системы онлайн-голосования на основе технологии цепочки блоков состоит из следующих шагов:

– определение требований и параметров системы голосования, включая регистрацию избирателей, аутентификацию, создание бюллетеней, процесс голосования, подсчет голосов и объявление результатов;

– сбор и обработка данных, связанных с процессом голосования, включая информацию об избирателях, данные о кандидатах, а также правила и положения о выборах;

– разработка архитектуры и модели системы голосования на основе блокчейна, включая выбор подходящей блокчейн-платформы, разработку умных-контрактов и интеграцию с веб-интерфейсами;

– разработка и внедрение умного-контракта на выбранной блокчейн-платформе для управления различными аспектами процесса голосования, такими как регистрация избирателей, создание бюллетеней, подача и подсчет голосов;

– создание удобного веб-интерфейса, который взаимодействует со слоем блокчейна, позволяя избирателям безопасно регистрироваться, проходить аутентификацию и отдавать свои голоса;

– внедрение мер безопасности для обеспечения целостности и конфиденциальности процесса голосования, включая шифрование, контроль доступа и защиту от потенциальных атак и уязвимостей;

– тестирование и аудит системы голосования для обеспечения ее функциональности, надежности и соответствия установленным требованиям и

регламентам.

Разработка информационной системы онлайн-голосования на основе технологии блокчейн требует комплексного подхода и последовательных этапов работы для достижения максимальной эффективности и удовлетворения потребностей пользователей.

3.1.1 Программная реализация системы онлайн-голосования

Для реализации системы используется несколько функций, которые работают вместе, обрабатывая подключение к Metamask, извлекая данные из умного контракта и управляя состоянием приложения на основе взаимодействий с пользователем. Основными из них является:

– основной компонент приложения App();

– функция, addVoting() для создания голосования;

– функция, когда необходимо добавить нового кандидата addCandidate();

– функция fetchCandidates() используется для получения списка кандидатов из конечной точки API;

– функция vote () вызывается, когда пользователь хочет проголосовать.;

– функция canVote(), эта функция проверяет, имеет ли текущий пользователь право голосовать;

– getCandidates(), эта функция извлекает список кандидатов из контракта;

– функция getCurrentStatus() извлекает текущий статус голосования из контракта;

– функция getRemainingTime() извлекает из контракта оставшееся время для периода голосования;

– функция handleAccountsChanged(accounts) вызывается при изменении учетной записи пользователя в Metamask;

– функция connectToMetamask() вызывается, когда пользователь нажимает кнопку «Подключиться к Metamask».

Разработанный код является частью приложения React, которое взаимодействует с умным-контрактом с использованием библиотеки Ethers.js. Он настраивает переменные состояния и функции для проверки подлинности пользователя, статуса голосования, оставшегося времени и управления кандидатами. Часть разработанного кода приложения представлена в Приложении Б. Вот подробное описание того, что делает код:

– код импортирует необходимые зависимости, включая React перехватчиков («useState» и «useEffect»), Ethers library.js, ABI договора и адрес, с постоянной файл, а также различные компоненты (Login, Finished, Connected, ListVotes, LoginError, AddVoting), которые являются веб-страницы системы. Также импортированы стили CSS («App.css» и «bootstrap/dist/css/bootstrap.min.css») и компоненты из React Router и React Bootstrap для навигации и компонентов пользовательского интерфейса;

– компонент приложения: функция «App()» является основным компонентом приложения. Она использует несколько переменных статуса для управления статусом приложения, таких как поставщик услуг, учетная запись пользователя, статус подключения, статус голосования, оставшееся время, кандидаты и имя нового кандидата;

– функция «useEffect» используется для выполнения побочных эффектов в жизненном цикле компонента. При монтаже компонента вызываются функции «getCandidates()», «getRemainingTime()» и «getCurrentStatus()». Также настраивает прослушиватель событий для события «accountsChanged()» от поставщика Ethereum (MetaMask) с помощью «window.ethereum.on». Функция очистки удаляет прослушиватель событий, когда компонент отключен;

– функция проверки подлинности: эта асинхронная функция позволяет добавить нового кандидата в смарт-контракт. Сначала проверяется, есть ли у текущей учетной записи разрешение на добавление кандидата (проверка происходит по уникальному адресу учетной записи администратора). Если у учетной записи есть разрешение, создается новая запись ethers.js поставщика и подписавшего, а также копию контракта с использованием предоставленных ABI и

адреса. В экземпляре контракта вызывается функция «addCandidate()», присваивающая кандидату новое имя. После добавления кандидата программа извлекает обновленный список кандидатов с помощью функции «fetchCandidates()» и обновляет переменную статуса «кандидаты». После она очищает переменную статуса «newCandidateName», чтобы сбросить поле ввода;

– функция выборки кандидатов: эта функция является заполнителем для получения списка кандидатов из смарт-контракта (используя API). В коде она имитирует отбор кандидатов, отправляя запрос на выбор в «/api/candidates» и возвращая ответ в формате JSON;

– функция «vote()»:эта функция позволяет пользователю проголосовать за кандидата. Она создает новый экземпляр провайдера Ethereum, используя «ethers.providers.Web3Provider» и запрашивает учетные записи пользователей. Он получает имя подписавшего (учетную запись пользователя) от провайдера. Он создает экземпляр контракта, используя «contractAddress» и «contractabi». Он вызывает функцию голосования из контракта, передавая «number-1» в качестве аргумента (при условии, что «number» представляет индекс кандидата). Он ожидает подтверждения транзакции с помощью «tx.wait()». Наконец, он вызывает функцию «canVote()», чтобы проверить статус голосования пользователя;

– функция «canVote()»:эта функция проверяет, имеет ли пользователь право голоса. Он создает новый экземпляр провайдера Ethereum, запрашивает учетные записи пользователей и получает имя подписавшего. Он создает экземпляр контракта, используя «contractAddress» и «contractabi». Она вызывает функцию «избиратели» из контракта, передавая адрес пользователя (полученный с помощью «signer.getAddress()») в качестве аргумента. Функция устанавливает переменную статуса «CanVote» на основе возвращенного статуса голосования;

– функция «getCandidates()»:эта функция извлекает список кандидатов и результаты подсчета их голосов из смарт-контракта. Она создает новый экземпляр провайдера Ethereum, запрашивает учетные записи пользователей и

получает имя подписавшего. Он создает экземпляр контракта, используя «contractAddress» и «contractabi». Он вызывает функцию «getAllVotesOfCandiates()» из контракта, чтобы получить список кандидатов. Он форматирует данные о кандидатах, сопоставляя каждый объект-кандидат с новым объектом со следующими свойствами: «индекс» - это индекс кандидата, «имя» - это имя кандидата, «Количество голосов» - количество голосов, полученных кандидатом (преобразовано из большого числа в обычное номер с помощью «ToNumber()»). Наконец, он обновляет переменную статуса «кандидаты» форматированными данными о кандидатах;

– функция «getCurrentStatus()»:эта функция извлекает текущий статус голосования из смарт-контракта. Она создает новый экземпляр провайдера Ethereum, запрашивает учетные записи пользователей и получает имя подписавшего. Она создает экземпляр контракта, используя «contractAddress» и «contractAbi». Она вызывает функцию «getVotingStatus()» из контракта, чтобы получить текущий статус голосования. Он обновляет переменную статуса «votingStatus» с помощью полученного статуса;

– функция «getRemainingTime()»: эта функция извлекает из смарт-контракта оставшееся время для процесса голосования. Она создает новый экземпляр провайдера Ethereum, запрашивает учетные записи пользователей и получает имя подписавшего. Она создает экземпляр контракта, используя «contractAddress» и «contractabi». Он вызывает функцию «getRemainingTime()» из контракта для вычисления оставшегося времени. Она обновляет переменную состояния «remainingTime», используя проанализированное оставшееся время (преобразованное из шестнадцатеричного в целое с помощью «parseInt()»);

– функция «handleAccountsChanged (accounts)»: эта функция вызывается при смене учетных записей пользователей в сети. Она проверяет, есть ли какие-либо доступные учетные записи и отличается ли текущая учетная запись от первой учетной записи в списке. Если обнаружена новая учетная запись, она обновляет переменную статуса «учетная запись» новым адресом учетной записи и вызывает функцию «canVote()» для проверки права пользователя голосо-

вать. Если учетные записи недоступны, параметру «isConnected» присваивается значение «false», а переменная статуса «account» очищается;

– функция «Преобразовать время (оставшееся время)»: эта функция преобразует оставшееся время (в секундах) в форматированную строку, представляющую часы и минуты. Она вычисляет время, деля оставшееся время на 3600 (количество секунд в часе) и округляя в меньшую сторону с помощью «Math.floor()». Наконец, он возвращает отформатированную строку времени;

– функция «connectToMetamask()»: эта функция вызывается, когда пользователь нажимает на кнопку для подключения своего кошелька Metamask. Она проверяет, доступен ли объект «window.ethereum», что указывает на то, что Metamask установлен в браузере. Он запрашивает учетные записи пользователей, используя «provider.send(«eth_requestAccounts», [])». Он извлекает имя подписавшего (учетную запись пользователя) у поставщика и получает адрес пользователя, используя «signer.getAddress()». Он обновляет переменную статуса «учетная запись», добавляя к ней адрес пользователя, устанавливает для параметра «isConnected» значение «true» и вызывает функцию «canVote()», чтобы проверить право пользователя голосовать. Если Metamask не обнаружен, отображается сообщение об ошибке;

– функция «handleNumberChange(e)»: эта функция вызывается, когда пользователь вводит число в поле ввода. Она обновляет переменную состояния «number» введенным значением;

– компонент JSX: компонент использует React Router для определения различных маршрутов и отображения соответствующих компонентов на основе текущего маршрута. Он отображает панель навигации («Navbar»), если пользователь подключен («isConnected» означает «true»). Панель навигации содержит ссылки на различные страницы в зависимости от адреса учетной записи пользователя. Компонент Routes определяет маршруты и соответствующие им компоненты. Корневой маршрут («/») отображает различные компоненты в зависимости от статуса голосования («votingStatus») и статуса подключения пользователя («isConnected»). Если голосование активно и пользователь подключен,

он отображает компонент «Подключен», передавая необходимые сведения. Если голосование активно, но пользователь не подключен, он выбирает компонент «Вход» с функцией «connectToMetamask» в качестве запроса. Если голосование завершено, он отображает компонент «Готово», передавая имя победителя и количество голосов. Маршруты «Добавить кандидата» и «Создать голосование» доступны только для определенной учетной записи (учетной записи, принадлежащей администратору системы). Если у пользователя есть доступ, он отображает компоненты «Добавить кандидата» и «Создать голосование» с необходимыми данными. Если у пользователя нет доступа, он отображает компонент «LoginError».

В целом, этот код устанавливает начальное состояние и функции для приложения React, которое взаимодействует со смарт-контрактом для целей голосования. Это позволяет пользователям подключаться к MetaMask, получать статус голосования и оставшееся время, управлять кандидатами (добавлять новых кандидатов при наличии разрешения).

3.1.2 Программная реализация умного контракта

Для реализации умного-контракта используется язык программирования Solidity. Основными функциями контракта являются:

– функция-конструктор constructor(), которая инициализирует контракт;

– функция addCandidate() позволяет администратору системы добавить нового кандидата в голосование;

– функция vote() позволяет избирателю отдать свой голос за определенного кандидата;

– функция getAllVotesOfCandiates() позволяет любому пользователю получить список кандидатов и количество их голосов;

– функция getVotingStatus() возвращает логическое значение, указывающее, активен ли период голосования в данный момент или нет;

– функция getRemainingTime() возвращает оставшееся время голосования.

Контракт также содержит модификатор под названием «onlyOwner», который ограничивает доступ к определенным функциям (например, «addCandidate()») только для администратора системы.

Описание кода умного контракта для разрабатываемой системы:

– инициализируеся идентификатор лицензии и директивы pragma (pragma solidity ^0.8.0), указывающей версию Solidity;

– определяется структура с именем «Candidate», которая представляет кандидата в процессе голосования. У каждого кандидата есть имя и количество голосов;

– в контракте объявлены переменные состояния: «candidates» - массив структур «Candidate» для хранения списка кандидатов, «владелец» - адрес владельца контракта, «голосующие» - отображение, которое отслеживает, проголосовал ли адрес уже или нет, «Начало голосования» - Отметка времени начала голосования, «Окончание голосования» - Отметка времени окончания голосования;

– функция конструктора выполняется при развертывании контракта. В качестве параметров она принимает массив имен кандидатов и продолжительность периода голосования в минутах. Инициализирует массив «кандидаты», создавая структуру «Кандидат» для каждого имени и устанавливая их начальное количество голосов равным 0. Также присваивает «владельцу» адрес, по которому выполняется развертывание контракта (адрес администратора), «votingStart» - текущую временную метку блока, а «votingEnd» - текущую временную метку блока плюс продолжительность в минутах;

– модификатор «onlyOwner» определен для ограничения доступа к определенным функциям только для владельца контракта. Он проверяет, совпадает ли адрес вызывающей функции («msg.sender») с адресом «owner», и выдает исключение, если условие не выполняется;

– функция «addCandidate» позволяет администратору добавить нового кандидата в массив «кандидаты». Она принимает имя кандидата («_name») в качестве параметра и помещает в массив новую структуру «Candidate» с задан-

ным именем и начальным количеством голосов, равным 0. Эта функция может быть вызвана только владельцем контракта из-за модификатора «onlyOwner»;

– функция «vote» позволяет избирателю отдать свой голос за определенного кандидата. В качестве параметра используется индекс кандидата («_candidateIndex») в массиве «кандидаты». Он проверяет, не голосовал ли избиратель ранее, используя отображение «избиратели», и подтверждает, что указанный индекс кандидата действителен. Если условия выполнены, количество голосов выбранного кандидата увеличивается на 1 (каждый избиратель в процессе голосования может отдать только один голос, если администратором системы не предусмотрено другого) и избиратель помечается как проголосовавший в отображении «избиратели»;

– функция «getAllVotesOfCandiates» – это функция просмотра, которая возвращает весь массив «кандидатов», позволяя любому пользователю получить список кандидатов и количество их голосов;

– функция «getVotingStatus» – это функция просмотра, которая возвращает логическое значение, указывающее, активен ли период голосования в данный момент или нет. Он проверяет, соответствует ли текущая временная метка блока периоду голосования (больше или равно «votingStart» и меньше «votingEnd»);

– функция «getRemainingTime» – это функция просмотра, которая возвращает оставшееся время (в секундах) до окончания периода голосования. Она проверяет, началось ли голосование, и выдает исключение, если нет. Если голосование уже завершилось, она возвращает значение 0. В противном случае, он вычисляет оставшееся время, вычитая текущую временную метку блока из «votingEnd»;

Этот контракт обеспечивает базовую структуру для проведения процесса голосования, позволяя владельцу контракта добавлять кандидатов, избирателям отдавать свои голоса и любому пользователю получать результаты голосования и статус голосования. Умный-контракт для инициализации голосования должен быть развернут в блокчейне, используя RPC-узел (вызов удалённых процедур) с

Ethers.js для связи с блокчейн-сетью, чтобы приложение работало без сбоев.

Внутри функции «main()» в выполняется развертывание контракта «Голосование» с использованием функции «deploy()». Функция «deploy()» принимает массив аргументов, которые передаются конструктору контракта «Голосование». В данном случае аргументами являются:

– Массив строк, представляющих имена кандидатов: «[]».

– Число, обозначающее продолжительность периода голосования в минутах: «90».

Экземпляр развернутого контракта хранится в переменной «Voting_».

Адрес развернутого контракта регистрируется в консоли с помощью «console.log()». Это позволяет увидеть адрес, по которому контракт развернут в блокчейн сети.

За вызовом функции «main()» следуют два обработчика promise:

– «.then(() => process.exit(0))»: Этот обработчик выполняется, если функция «main()» успешно завершается. Он вызывает «process.exit(0)» для завершения сценария с кодом состояния успеха, равным 0.

– «.catch(ошибка => { ... })»: Этот обработчик выполняется, если во время выполнения функции возникает ошибка. Регистрирует ошибку на консоли, используя «console.error(ошибка)», а затем вызывает «process.exit(1)», чтобы завершить работу скрипта с кодом состояния ошибки, равным 1.

Таким образом, этот код развертывает контракт «Голосование» в блокчейне, используя предоставленные имена кандидатов (если при инициализации контракта они были указаны) и продолжительность голосования (время существования контракта). Он извлекает фабрику контрактов, развертывает контракт и записывает адрес развернутого контракта. Затем скрипт завершает работу с кодом состояния success, если развертывание завершилось успешно, или с кодом состояния error, если во время процесса произошла ошибка.

3.1.2 Подключение к сети Volta

Чтобы развернуть умный-контракт в выбранной блокчейн-сети, такой как Volta, и обеспечить бесперебойное взаимодействие между децентрализованным

приложением (DApp) и блокчейном, необходимо выполнить несколько важных шагов. Первым обязательным условием является установка браузерного расширения Metamask, которое служит связующим звеном между DApp и блокчейн-сетью.

Metamask – это широко используемый криптовалютный кошелек, который позволяет пользователям безопасно хранить свои цифровые активы, управлять ими и взаимодействовать с ними. Он выступает в качестве шлюза для подключения dApps к различным блокчейн-сетям. Чтобы установить соединение между приложением и сетью Volta, сеть должна быть добавлена в Metamask с помощью функции «Пользовательский RPC» (рисунок 9).

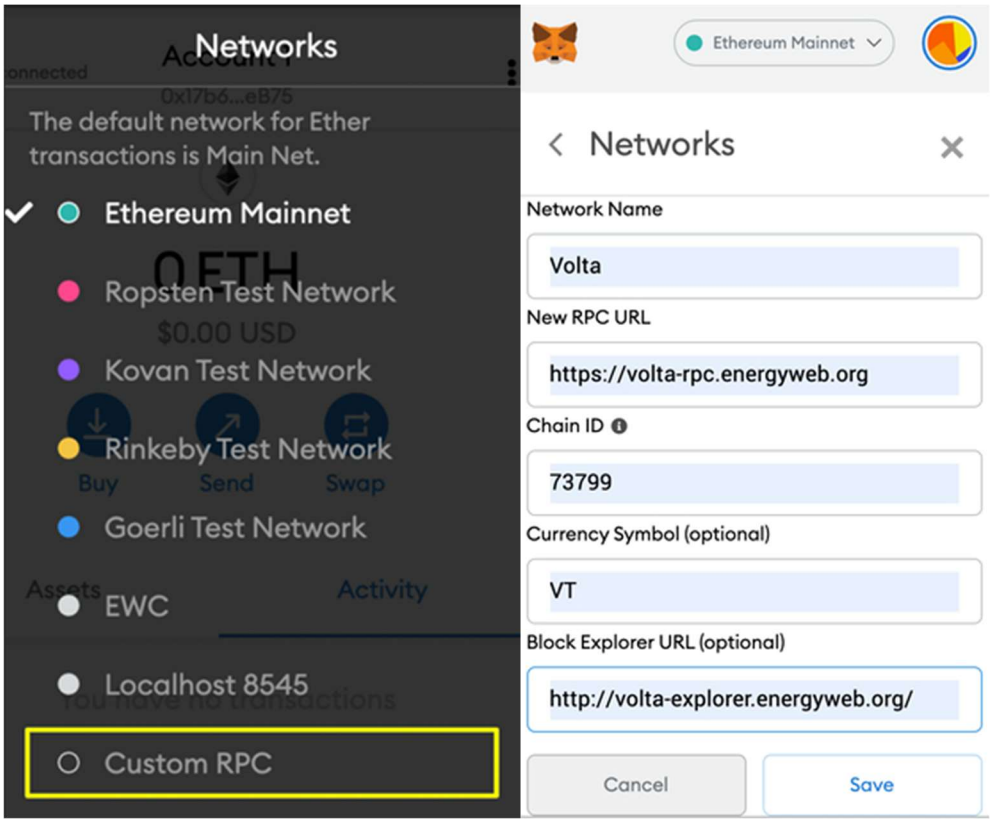

Рисунок 9 – Функция подключения учетной записи

Как только сеть Volta будет успешно добавлена в Metamask, следующим шагом будет получение закрытых ключей API. Эти ключи обеспечивают безопасное и аутентифицированное соединение между сетью Volta и уникальным адресом Metamask администратора системы. Ключи API имеют решающее значение для обеспечения взаимодействия DApp с сетью Volta и выполнения не-

обходимых операций, таких как развертывание умных контрактов и взаимодействие с ними.

Для обеспечения безопасности и конфиденциальности этих ключей API они обычно хранятся в файле с именем «.env». Этот файл служит в качестве файла конфигурации для децентрализованного приложения, позволяя разработчикам управлять конфиденциальной информацией и настройками приложения, которые не подходят для установки в качестве переменных среды. Сохраняя ключи API в файле «.env», Децентрализованное приложение может легко получить к ним доступ и использовать их, сохраняя при этом высокий уровень безопасности.

Благодаря надежному хранению ключей API, приложение теперь может инициализировать и изменять умные контракты в блокчейне Volta. Процесс инициализации и модификации смарт-контрактов включает в себя использование ключей API для аутентификации и авторизации взаимодействия приложения с сетью Volta. Приложение может развертывать новые умные контракты, вызывать функции в рамках существующих контрактов и выполнять различные операции на основе логики, определенной в коде смарт-контракта.

3.2 Результаты фактического тестирования программного продукта

Для обеспечения надежности разработанной системы онлайн-голосования с использованием технологии блокчейн необходимо всестороннее тестирование. Подход к тестированию должен охватывать все аспекты системы, включая основной компонент приложения, функции умных-контрактов и взаимодействие с пользователем.

Данное приложение может быть запущено на стороне пользователя используя браузер для доступа к самой системе, вход в систему обеспечивается при помощи приложения Metamask с авторизированным аккаунтом пользователя. Разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами показана на рисунке А.5 приложения А. Данная диаграмма помогает описать архитектуру разработанной системы, установив зависимости между программными компонентами.

При тестировании разработанной системы каждая функция была отдельно протестирована:

– «addVoting()», чтобы убедиться, что умный контракт может быть успешно создан;

– «addCandidate()», чтобы убедиться, что новые кандидаты могут быть добавлены в систему голосования;

– «fetchCandidates()», чтобы убедиться, что соответствие кандидатов корректно получено из конечной точки API;

– «vote()», чтобы убедиться, что пользователи могут успешно проголосовать;

– «canVote()», чтобы убедиться, что она правильно проверяет, имеет ли пользователь право голоса;

– «getCandidates()», чтобы убедиться, что она точно извлекает список кандидатов из смарт-контракта;

– «getCurrentStatus()», чтобы убедиться, что она правильно извлекает текущий статус голосования из смарт-контракта;

– «getRemainingTime()», чтобы убедиться, что она точно извлекает из смарт-контракта оставшееся время для периода голосования;

– «handleAccountsChanged()», чтобы убедиться, что она корректно обрабатывает изменения в учетной записи пользователя Metamask;

– «connectToMetamask()», чтобы убедиться, что она устанавливает соединение с Metamask, когда пользователь нажимает кнопку «Подключиться к Metamask».

Была проведена проверка всех возможных состояний объектов. Проверка проводилась во время процесса регрессионного тестирования системы [16]. Неисправностей обнаружено не было.

На рисунке 10 показано создание контракта с предопределённым списком кандидатов, участвующих в процессе голосования и временем существования умного-контракта (временем выделенным для голосования) в административной панели информационной системы. Также на этом рисунке показана запись

в технологию цепочки блоков данной транзакции на создание умного контракта для системы онлайн голосования.

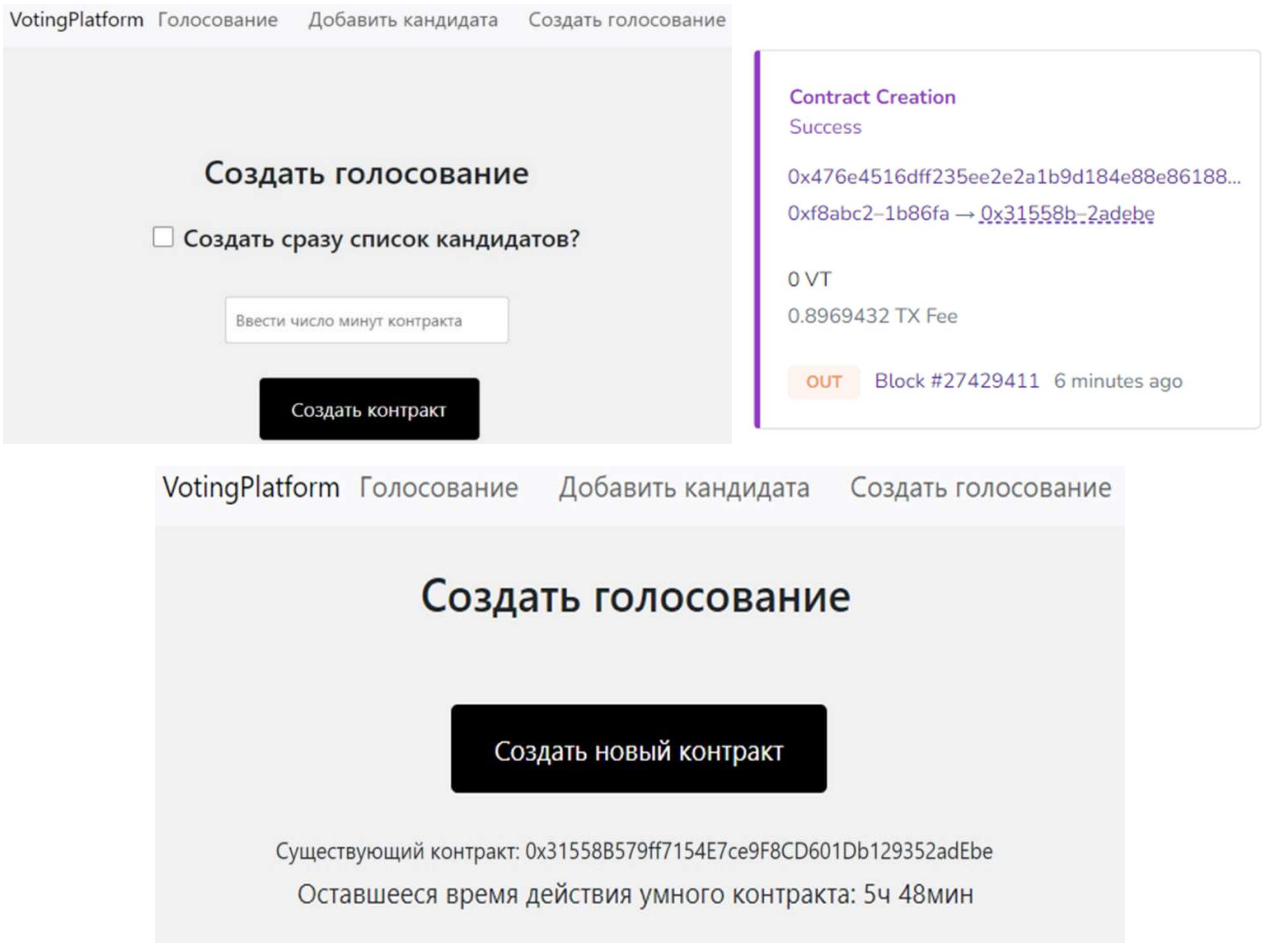

Рисунок 10 – Инициализация умного-контракта в блокчейне

На рисунке 11 показана работа функции «addCandidate()» для добавления кандидата (имя кандидата может быть строкой или цифрой) в голосование.

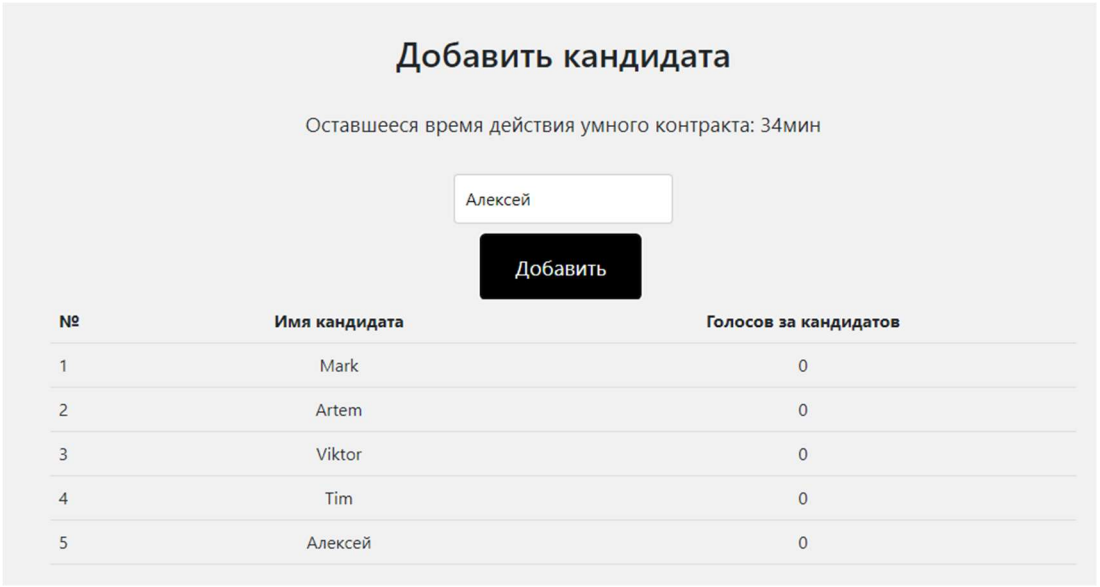

Рисунок 11 – Добавление нового кандидата

На рисунке 12 показана работа функций «vote()» и «canVote()», данные функции предоставляют возможность отдать голос за кандидата и проверяют имеет ли право избиратель на то, чтобы проголосовать.

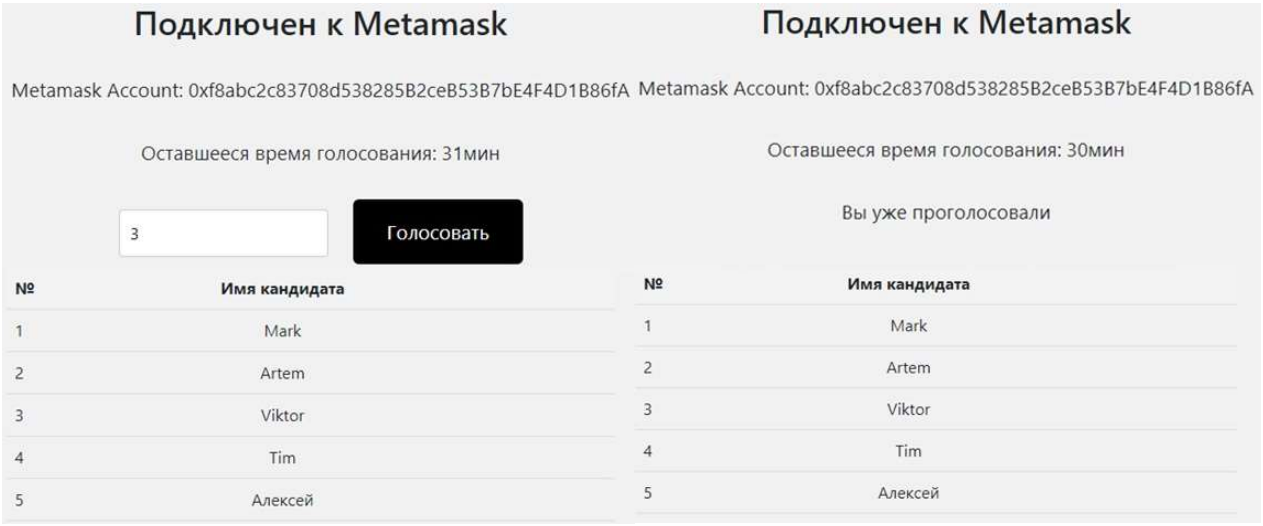

Рисунок 12 – Голос отдан за кандидата номер 3

На рисунке 13 показана работа функций «getCurrentStatus()», данная функция проверяет статус процесса голосования, если голосование завершено, то избиратели больше не могут голосовать, а только посмотреть результаты голосования.

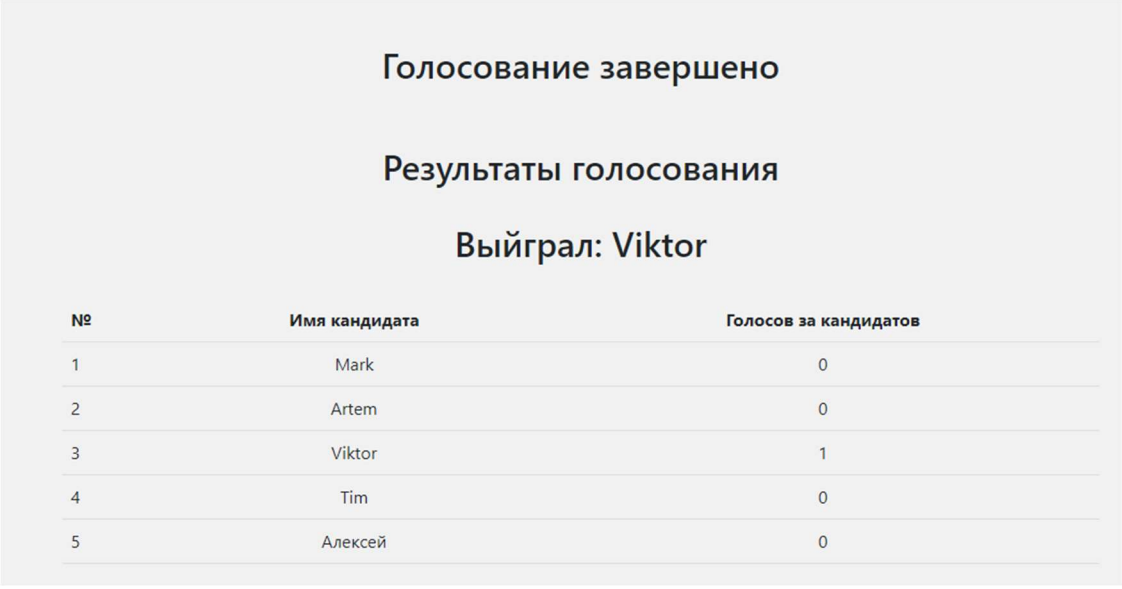

Рисунок 13 – Голосование завершено

Для проверки работоспособности системы в целом было проведено:

– тестирование пользовательского интерфейса, чтобы убедиться, что все элементы функционируют корректно и удобны для пользователя. Интегра-

ция между основным компонентом приложения и смарт-контрактом для обеспечения бесперебойной связи и обмена данными. Тестирование различных пользовательских сценариев, таких как подключение к Metamask, создание голосования, добавление кандидатов, подача голосов и получение результатов голосования. Тестирование реакции системы на различные запросы пользователей и нестандартные ситуации, чтобы обеспечить надежность и обработку ошибок;

– тестирование устойчивости системы к распространенным уязвимостям безопасности, таким как внедрение SQL, межсайтовый скриптинг (XSS) и подделка межсайтовых запросов (CSRF). Внедрение средств контроля доступа и механизмов аутентификации, чтобы гарантировать, что только авторизованные пользователи могут выполнять определенные действия. Тестирование умного контракта на наличие потенциальных уязвимостей и гарантия, что он защищен от распространенных атак, таких как повторный ввод данных и переполнение целых чисел;

– тестирование производительности системы при различных нагрузках, чтобы убедиться, что она может обрабатывать большое количество одновременных пользователей и транзакций (рисунок 14). Проверка времени отклика и системные задержки, чтобы обеспечить бесперебойную работу пользователей. Тестирование масштабируемости системы, чтобы убедиться, что она способна справляться с растущими требованиями и растущим числом пользователей.

Рисунок 14 – Тестирование производительности

На рисунке 14 показано тестирование производительности системы при помощи инструмента нагрузочного тестирования Grafana k6. Время отклика системы составило 322 миллисекунды при выполненных 13733 запросах записи голосов на странице голосования.

Проведя тщательное тестирование по этим различным аспектам, была проверена функциональность, безопасность и производительность разработанной системы онлайн-голосования, использующей технологию цепочки блоков. Важно постоянно отслеживать и тестировать систему даже после развертывания, чтобы выявлять и устранять любые проблемы, которые могут возникнуть со временем.

Тестирование системы заключается в полной проверке работоспособности приложения как единой системы. Как правило, тестирование совершается методом «чёрного ящика», то есть с помощью проверки работоспособности без исходного кода, или глазами обычного пользователя. Результаты тестирования совмещаются с результатами тестирования вышеописанных элементов, и на их основе делается окончательный вердикт.

На данный момент система работает стабильно. Критических неисправностей обнаружено не было.

3.3 Анализ достоверности и практической значимости результатов

Разработанная система онлайн-голосования с использованием технологии блокчейн может революционизировать способы проведения выборов и принятия решений. Надежность и практическая значимость результатов, полученных с использованием этой системы, могут быть проанализированы следующим образом [17]:

– использование технологии блокчейн обеспечивает неизменность и прозрачность записей о голосовании. Как только голос подан и зафиксирован в блокчейне, его невозможно изменить или удалить, что обеспечивает высокий уровень безопасности и целостности процесса голосования. Децентрализованный характер блокчейна устраняет необходимость в центральном органе для

управления процессом голосования, снижая риск фальсификации результатов голосования или манипуляций с ними. Криптографические алгоритмы, используемые в блокчейне, обеспечивают конфиденциальность и анонимность избирателей, защищая их частную жизнь и предотвращая любой несанкционированный доступ к их информации о голосовании;

– основанная на блокчейне система голосования предоставляет прозрачную и проверяемую информацию обо всех действиях при голосовании. Каждый голос записывается в блокчейн вместе с отметкой времени и другими соответствующими метаданными, что упрощает проверку и аудит процесса голосования. Прозрачность системы позволяет заинтересованным сторонам, таким как наблюдатели за выборами и общественность, независимо проверять точность результатов голосования, повышая доверие к избирательному процессу;

– онлайн-характер системы голосования позволяет избирателям участвовать в избирательном процессе из любой точки мира, используя свои устройства и подключение к Интернету. Это повышает доступность и удобство для избирателей, особенно для тех, кто может столкнуться с трудностями при физическом посещении избирательных участков. Удобный интерфейс системы голосования позволяет избирателям легко ориентироваться и отдавать свои голоса без каких-либо технических сложностей, что делает процесс более открытым и способствует более высокой явке избирателей;

– внедрение системы может значительно снизить затраты, связанные с традиционными методами голосования, такими как печать бюллетеней, строительство избирательных участков и наем персонала для управления процессом голосования. Автоматизированный характер системы упрощает процесс голосования, сокращая время и усилия, необходимые для подсчета голосов и подведения итогов. Это приводит к более быстрому и эффективному получению результатов выборов, сводя к минимуму вероятность человеческих ошибок и несоответствий;

– систему голосования можно масштабировать для проведения выборов различного масштаба, от небольших местных выборов до крупномасштабных национальных или международных выборов. Модульная архитектура системы обеспечивает легкую интеграцию с существующей инфраструктурой и процессами голосования, что позволяет адаптировать ее к различным избирательным условиям и требованиям.

Основанная на блокчейне система, созданная для онлайн-голосования, может быть применена в различных отраслях промышленности. Ее можно использовать для управления цепочками поставок, обеспечивая целостность и прослеживаемость товаров. В здравоохранении она может безопасно хранить данные о пациентах и обмениваться ими, улучшая качество обслуживания и сокращая количество ошибок. Токенизация активов позволяет осуществлять частичное владение и торговлю реальными активами, такими как недвижимость или произведения искусства. Система также может использоваться для управления идентификацией, предоставляя частным лицам контроль над своими личными данными. При управлении правами интеллектуальной собственности это может привести к созданию неизменных записей о владении цифровыми активами. Технология блокчейн может повысить доверие и подотчетность в благотворительной деятельности и инициативах, оказывающих социальное воздействие. Ее также можно применять для обеспечения экологической устойчивости, стимулирования и отслеживания экологически чистых практик.

Таким образом, разработанная система онлайн-голосования на основе блокчейна может послужить основой для множества других децентрализованных приложений в различных секторах. Присущие технологии блокчейн характеристики делают ее универсальным инструментом, адаптируемым для решения различных реальных задач и возможностей.

ЗАКЛЮЧЕНИЕ

В данной работе были изучены возможности использования технологии блокчейн в системах онлайн-голосования, а также проведен анализ существующих подходов к использованию смарт-контрактов в приложениях для голосования в качестве системы для безопасной и прозрачной обработки голосов. Выяснилось, что на данный момент существуют различные реализации таких систем, и в большинстве случаев используются смарт-контракты на основе Ethereum.

Далее был определен функционал системы онлайн-голосования, на основе которого были сформулированы требования к программному продукту, реализация которых позволила создать полноценный и работоспособный продукт. Затем было проведено проектирование программного продукта с модульной архитектурой и продумано взаимодействие модулей.

Была разработана структура смарт-контракта и рассмотрено его применение в процессе голосования. Были реализованы необходимые функции для создания сессии голосования, добавления кандидатов, подачи голосов и получения результатов голосования. Были выбраны типы данных для входных и выходных параметров и определена логика обработки различных сценариев. Затем смарт-контракт был реализован с использованием языка программирования Solidity.

Выбор блокчейн-платформы Volta для внедрения системы голосования был обоснован ее масштабируемостью, быстрой обработкой транзакций и низкими транзакционными издержками.

Использование React, Node.js и Express.js для разработки интерфейсов и серверной части было продиктовано их популярностью, надежностью и широкой поддержкой сообщества. MongoDB была выбрана в качестве решения для работы с базами данных благодаря своей гибкости, масштабируемости и способности эффективно обрабатывать неструктурированные данные.

Hardhat и Ethers.js были выбраны в качестве инструментов разработки для развертывания смарт-контрактов и взаимодействия с ними. Среда разработки и платформа тестирования Hardhat упростили процесс написания, компиляции и развертывания смарт-контрактов и Ethers.js предоставили удобную библиотеку для взаимодействия с блокчейном и выполнения транзакций.

В результате приложение состоит из нескольких модулей, одним из которых является модуль взаимодействия с блокчейном. В систему передаётся пользовательский ввод, и система отображает интерфейс и результаты голосования.

Информационная система была разработана, отлажена и протестирована. Была проверена работоспособность как самого веб-приложения, так и созданного умного-контракта.

На данный момент система работает с некоторыми ограничениями. В будущем возможно улучшение этого приложения путем расширения функционала, усиления мер безопасности и добавления новых функций для улучшения пользовательского опыта и доступности процесса голосования. Кроме того, для обеспечения точности и достоверности результатов голосования могут быть проведены дополнительные проверки и аудит.

БИБЛИОГРАФИЧЕСКИЕ ССЫЛКИ

1 Блокчейн // Wikipedia : офиц. сайт. 2024. URL: https://ru.wikipedia.org/wiki/Блокчейн (дата обращения: 30.01.2024).

2 Хеш-сумма // Wikipedia : офиц. сайт. 2024. URL: https://ru.wikipedia.org/wiki/Хеш-сумма (дата обращения: 30.01.2024).

3 Хеш-функция // Wikipedia : офиц. сайт. 2024. URL: https://ru.wikipedia.org/wiki/Хеш-функция (дата обращения: 30.01.2024).

4 Электронное голосование // Wikipedia : офиц. сайт. 2024. URL: https://ru.wikipedia.org/wiki/Электронное_голосование (дата обращения: 30.01.2024).

5 What Is Proof of Work (PoW)? // BinanceAcademy : офиц. сайт. 2024. URL: https://academy.binance.com/en/articles/proof-of-work-explained (дата обращения: 30.01.2024).

6 What Is Proof of Stake (PoS)? // BinanceAcademy : офиц. сайт. 2024. URL: https://academy.binance.com/en/articles/proof-of-stake-explained (дата обращения: 30.01.2024).

7 Proof of Authority Explained // BinanceAcademy : офиц. сайт. 2024. URL: https://academy.binance.com/en/articles/proof-of-authority-explained (дата обращения: 30.01.2024).

8 Solana: A new architecture for a high performance blockchain v0.8.13 // Solana : офиц. сайт. 2018. URL: https://solana.com/solana-whitepaper.pdf (дата обращения: 30.01.2024).

9 Difficulty control for blockchain-based consensus systems, peer-to-peer networking and applications // Researchgate : офиц. сайт. 2015. URL: https://www.researchgate.net/publication/276104952_Difficulty_control_for_blockchain-based_consensus_systems (дата обращения: 14.02.2024).

10 Коцюба И.Ю., Чунаев А.В., Шиков А.Н. Основы проектирования информационных систем. СПб. Университет ИТМО, 2015. 206 с.

11 A smart contract for boardroom voting with maximum voter privacy ap-

plications // Researchgate : офиц. сайт. 2017. URL: https://www.researchgate.net/publication/322000346_A_Smart_Contract_for_Boardroom_Voting_with_Maximum_Voter_Privacy (дата обращения: 14.02.2024).

12 Windows Subsystem for Linux // Wikipedia : офиц. сайт. 2024. URL: https://en.wikipedia.org/wiki/Windows_Subsystem_for_Linux (дата обращения: 10.03.2024).

13 Официальный сайт Visual Studio // Microsoft Visual Studio : офиц. сайт. 2024. URL: доступа: https://code.visualstudio.com/docs (дата обращения: 10.03.2024).

14 Volta Token (VT) // Energy Web : офиц. сайт. 2022. URL: доступа: https://energy-web-foundation.gitbook.io/energy-web/community-resources/using-the-energy-web-chain/volta-testnet-token (дата обращения: 18.04.2024).

15 MetaMask // Wikipedia : офиц. сайт. 2024. URL: https://en.wikipedia.org/wiki/MetaMask (дата обращения: 10.03.2024).

16 Что такое системное тестирование? Глубокое погружение в подходы, типы // Zaptest : офиц. сайт. 2024. URL: https://www.zaptest.com/ru/что-такое-системное-тестирование-глу (дата обращения: 18.04.2024).

17 Киндеев Е.А. Надежность технических систем. Владимир. ВлГУ, 2016. 97 с.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Белов, В.В. Проектирование информационных систем : учебник для студ. учреждений высш. проф. образования / В. В. Белов, В. И. Чистякова. – М. : Издательский центр «Академия», 2013. – 352 с.

2 Белый, серый и черный ящик [Электронный ресурс]. – Режим доступа: https://www.careerist.com/ru-insights/belyy-seryy-i-chernyy-yashchik – 27.03.2024.

3 Берг, Д.Б. Модели жизненного цикла : учебное пособие / Д. Б. Берг, Е. А. Ульянова, П. В. Добряк. – Екатеринбург : Изд-во Урал. ун-та, 2014. – 74 с.

4 Блокчейн [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Блокчейн – 30.01.2024.

5 Брауде, Э. Технология разработки программного обеспечения : пер. с англ. / Э. Брауде. – СПб. : ПИТЕР, 2004. – 655 с.

6 Буч, Г. Язык UML. Руководство пользователя : пер. с англ. Мухин Н. / Г. Буч, Д. Рамбо, И. Якобсон. – 2-е изд. – М. : ДМК Пресс, 2006. – 496 с.

7 Буч, Г. UML. Классика CS / Г. Буч, Д. Рамбо, И. Якобсон. – 2-е изд. – СПб. : Питер, 2006. – 736 с.

8 Васильев, Ф.П. Методы оптимизации. / Ф. П. Васильев. – М. : Факториал Пресс, 2002. – 824 с.

9 Вигерс, К. Разработка требований к программному обеспечению : пер. с англ. / К. Вигерс. – М. : Русская Редакция, 2004. – 576 с.

10 Децентрализованное приложение [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Децентрализованное_приложение – 27.02.2024.

11 Заяц, А. М. Блокчейн-системы и технологиии : учебное пособие для СПО / А. М. Заяц. – СПб. : Лань, 2024. – 112 с.

12 Киндеев, Е. А. Надежность технических систем / Е. А. Киндеев. – Владимир : ВлГУ, 2016. – 97 с.

13 Коцюба, И.Ю. Основы проектирования информационных систем. /

И. Ю. Коцюба, А. В. Чунаев, А. Н. Шиков. – СПб. : Университет ИТМО, 2015. – 206 с.

14 Леоненков, А. Самоучитель UML / А. Леоненков. – 2-е изд., доп. – СПб. : БХВ-Петербург, 2004. – 432 с.

15 Липаев, В. В. Надежность программных средств / В. В. Липаев. – М.: СИНТЕГ, 1998. – 358 с.

16 Макконнелл, С. Совершенный код. Мастер-класс : пер. с англ. / С. Макконнелл. – М. : Русская редакция, 2010. – 896 с.

17 Матеишен, Т.Е. Особенности разработки информационной системы онлайн-голосования с использованием технологии цепочки блоков. // Инновационный потенциал развития науки в современном мире: технологии, инновации, достижения / Сборник научных статей по материалам XIII Международной научно-практической конференции (6 октября 2023 г., г. Уфа) – Уфа : Изд. НИЦ Вестник науки, 2023. – С. 43-46.

18 Матеишен, Т.Е. Практическая реализация информационной системы онлайн-голосования с использованием технологии цепочки блоков. // Проблемы и перспективы разработки и внедрения передовых технологий / Сборник статей Международной научно-практической (25 октября 2023 г., г. Волгоград) – Уфа : Аэтерна, 2023. – С. 36-39.

19 Орлов, С.А. Технологии разработки программного обеспечения / С. А. Орлов. – СПб. : ПИТЕР, 2002. – 464 с.

20 Официальный сайт Visual Studio [Электронный ресурс]. – Режим доступа: https://code.visualstudio.com/docs – 25.02.2024.

21 Технология Blockchain простыми словами [Электронный ресурс]. – Режим доступа: https://club.dns-shop.ru/blog/t-57-tehnologii/30931-tehnologiya-blockchain-prostyimi-slovami/ – 30.01.2024.

22 О Стратегии развития информационного общества в Российской Федерации на 2017 – 2030 годы [Электронный ресурс] : Указ Президента РФ от 9.05.2017 № 203. Документ опубликован не был. Доступ из справ.- правовой системы «КонсультантПлюс».

23 Хеш-сумма [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Хеш-сумма – 30.01.2024.

24 Хеш-функция [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Хеш-функция – 30.01.2024.

25 Чекал, Е. Г. Надежность информационных систем / Е. Г. Чекал, А. А. Чичев – 2-е изд. – Ульяновск : Ульяновский гос. ун-т, 2012. – 118 с.

26 Что такое системное тестирование? Глубокое погружение в подходы, типы [Электронный ресурс]. – Режим доступа: https://www.zaptest.com/ru/что-такое-системное-тестирование-глу – 27.03.2024

27 Электронное голосование [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Электронное_голосование – 30.01.2024.

28 A smart contract for boardroom voting with maximum voter privacy [Электронный ресурс]. – Режим доступа: https://www.researchgate.net/publication/322000346_A_Smart_Contract_for_Boardroom_Voting_with_Maximum_Voter_Privacy – 30.01.2024.

29 A Survey of Blockchain Based on E-voting Systems [Электронный ресурс]. – Режим доступа: https://www.researchgate.net/publication/340081625_A_Survey_of_Blockchain_Based_on_E-voting_Systems – 25.04.2024.

30 Blanchard, B. S. Systems engineering and analysis / B. S. Blanchard, Wolter J. Fabrycky. – 4-е изд. – Хобокен : Prentice Hall, 2006 – 824 с.

31 Blockchain and Distributed Ledger Technology Use Cases: Applications and Lessons Learned [Электронный ресурс]. – Режим доступа: https://www.researchgate.net/publication/341958649_Blockchain_and_Distributed_Ledger_Technology_Use_Cases_Applications_and_Lessons_Learned – 30.01.2024.

32 China Academy of Information and Communication Technology Trusted Blockchain Initiatives Blockchain White Paper [Электронный ресурс]. – Режим доступа: http://www.caict.ac.cn/english/yjcg/bps/201901 – 27.03.2024.

33 CSS [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/CSS – 27.02.2024.

34 Difficulty control for blockchain-based consensus systems, peer-to-peer networking and applications [Электронный ресурс]. – Режим доступа: https://www.researchgate.net/publication/276104952_Difficulty_control_for_blockchain-based_consensus_systems – 30.01.2024.

35 Documentation is for Ethers v5.7 [Электронный ресурс]. – Режим доступа: https://docs.ethers.org/v5/ – 27.02.2024.

36 Documentation is for Hardhat [Электронный ресурс]. – Режим доступа: https://hardhat.org/docs – 27.02.2024.

37 Enigma: Decentralized Computation Platform with Guaranteed Privacy [Электронный ресурс]. – Режим доступа: http://livinglab.mit.edu/wp-content/uploads/2016/01/enigma_full.pdf – 25.04.2024.

38 Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. [Электронный ресурс]. – Режим доступа: https://ethereum.org/669c9e2e2027310 b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf – 25.04.2024.

39 Express [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Express– 27.02.2024.

40 HTML [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/HTML – 27.02.2024.

41 JavaScript [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/ JavaScript – 27.02.2024.

42 Managing the Development of Large Software Systems [Электронный ресурс]. – Режим доступа: http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/ – 28.02.2024.

43 MetaMask [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/MetaMask – 27.02.2024.

44 Microsoft Visual Studio [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio – 27.02.2024.

45 MongoDB [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/MongoDB – 27.02.2024.

46 Node.js [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Node.js – 27.02.2024.

47 OMG Unified Modeling Language (OMG UML), Infrastructure Version 2.2 [Электронный ресурс] : офиц. сайт. – 2009 – Режим доступа: https://www.omg.org/spec/UML/2.2/Superstructure/PDF – 20.02.2024.

48 React [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/React – 27.02.2024.

49 Selby, R.W. Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research / R. W. Selby. – Нью-Йорк: John Wiley & Sons, 2007. – 834 с.

50 Self-tallying Elections and Perfect Ballot Secrecy [Электронный ресурс]. – Режим доступа: https://www.researchgate.net/publication/221010610_Selftallying_Elections_and_Perfect_Ballot_Secrecy – 30.01.2024.

51 Set up MetaMask to interact with Energy Web Chain [Электронный ресурс]. – Режим доступа: https://energy-web-foundation.gitbook.io/energy-web/ew-dos-technology-components-2023/trust-layer-energy-web-chain/ewc-guides-and-tutorials/connect-to-energy-web-chain-main-network-with-metamash – 27.02.2024.

52 Solidity [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Solidity – 27.02.2024.

53 Trustworthy Electronic Voting Using Adjusted Blockchain Technology [Электронный ресурс]. – Режим доступа: https://ieeexplore.ieee.org/document/8651451 – 05.04.2024.

54 Volta Token (VT) [Электронный ресурс]. – Режим доступа: https://energy-web-foundation.gitbook.io/energy-web/community-resources/using-the-energy-web-chain/volta-testnet-token – 27.02.2024.

55 Windows Subsystem for Linux [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Windows_Subsystem_for_Linux – 27.02.2024

ПРИЛОЖЕНИЕ А

Диаграммы приложения

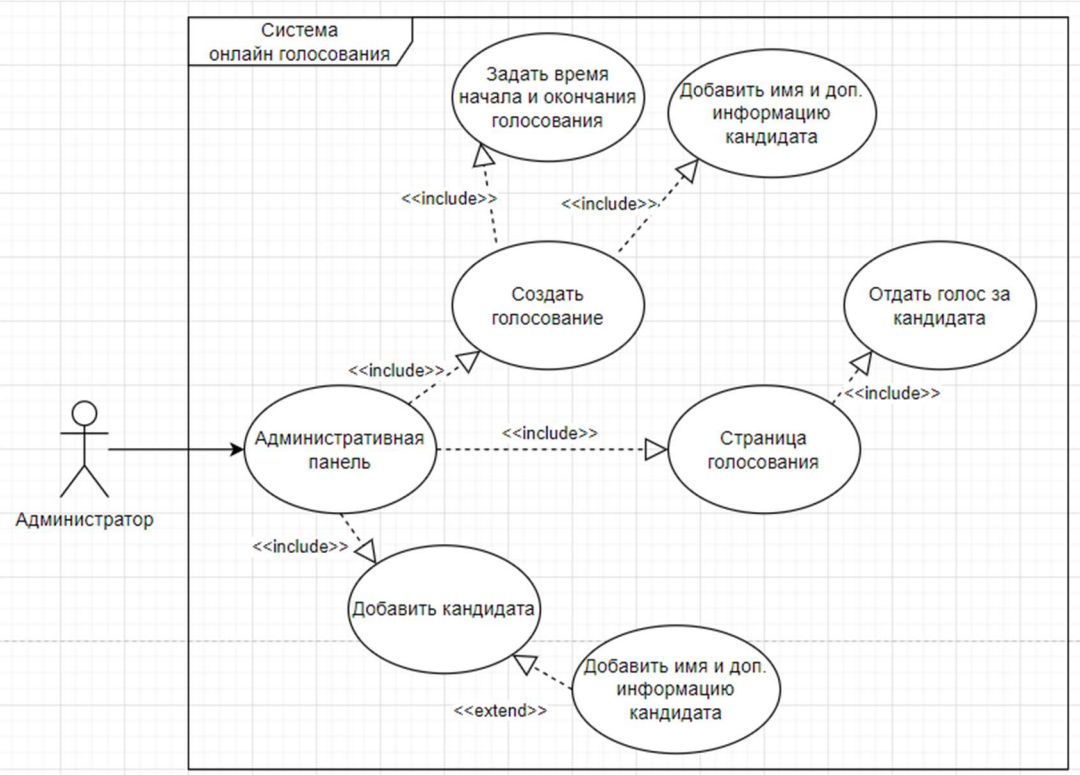

Рисунок А.1 – Диаграмма вариантов использования

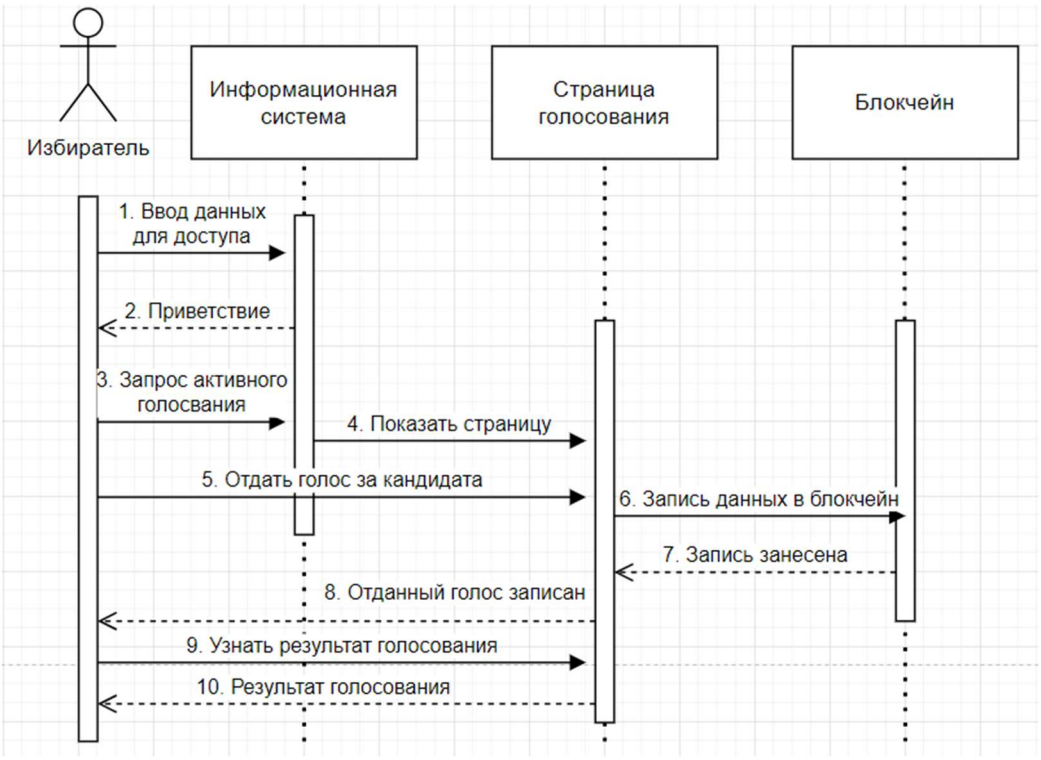

Рисунок А.2 – Диаграмма последовательности

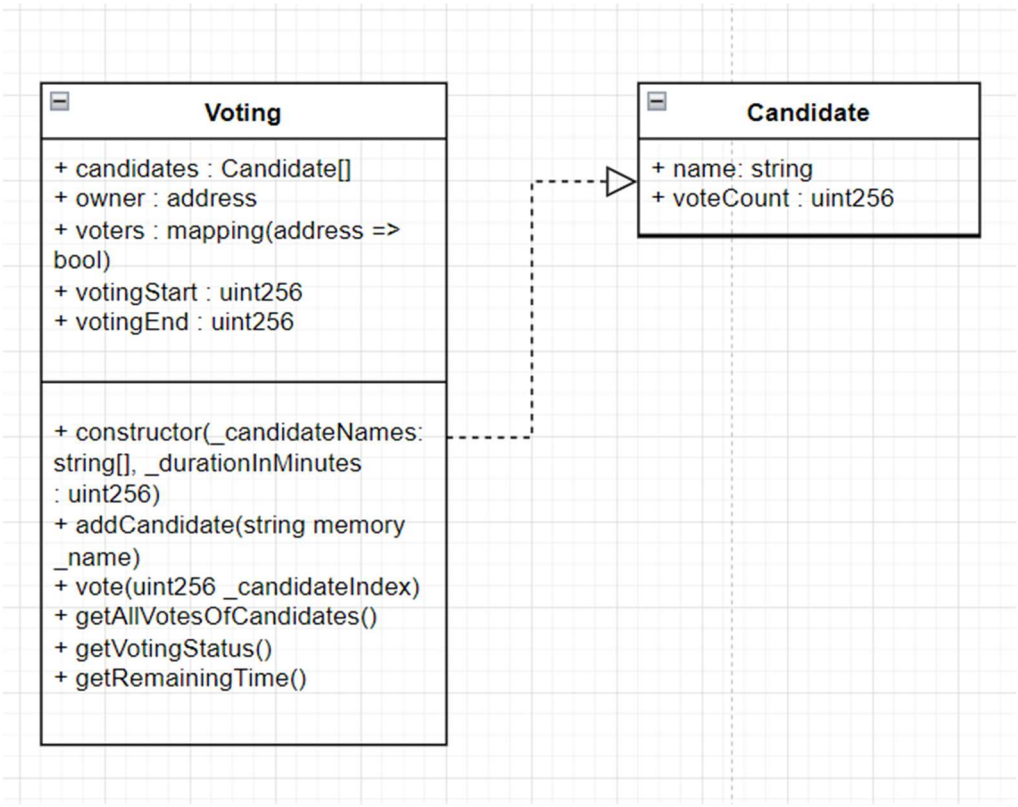

Рисунок А.3 – Диаграмма классов

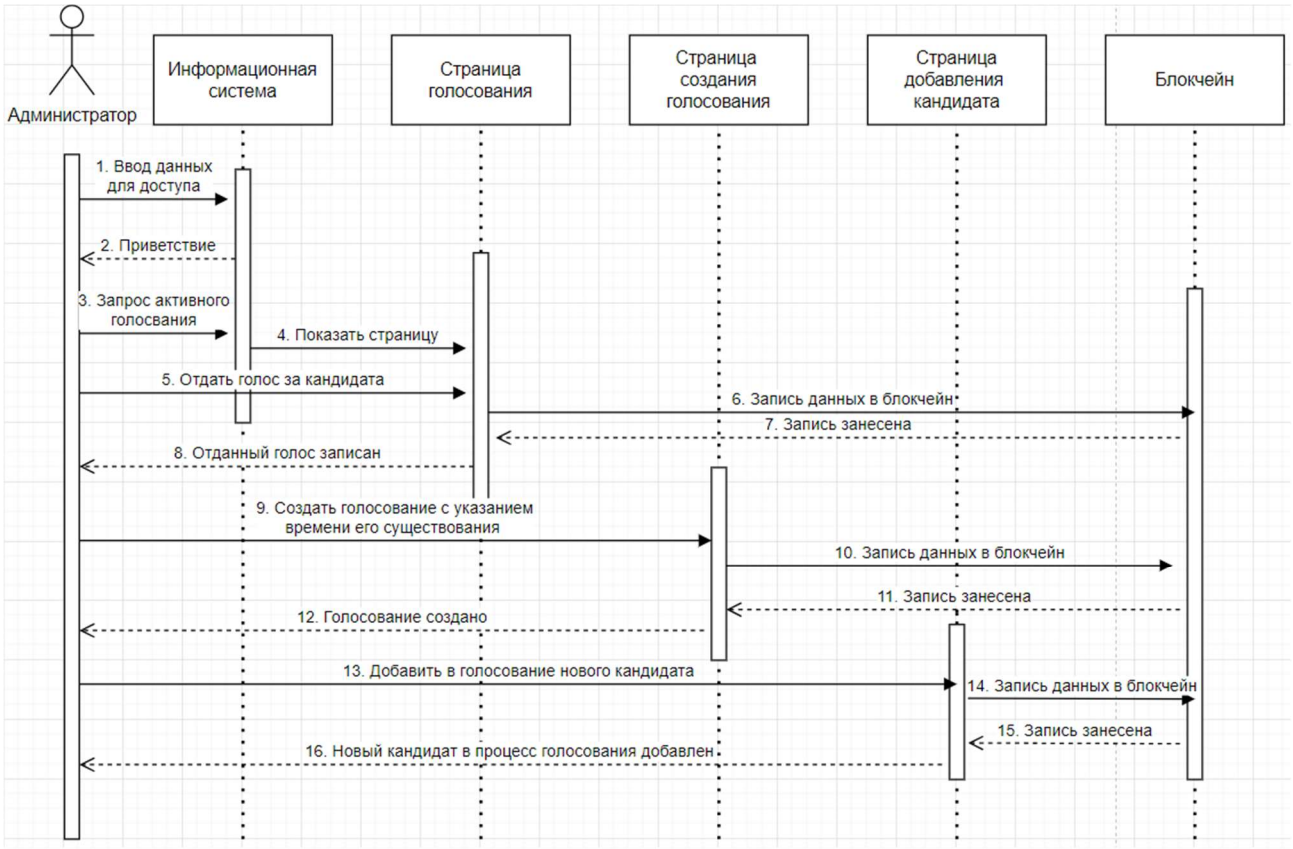

Рисунок А.4 – Диаграмма последовательности администратора

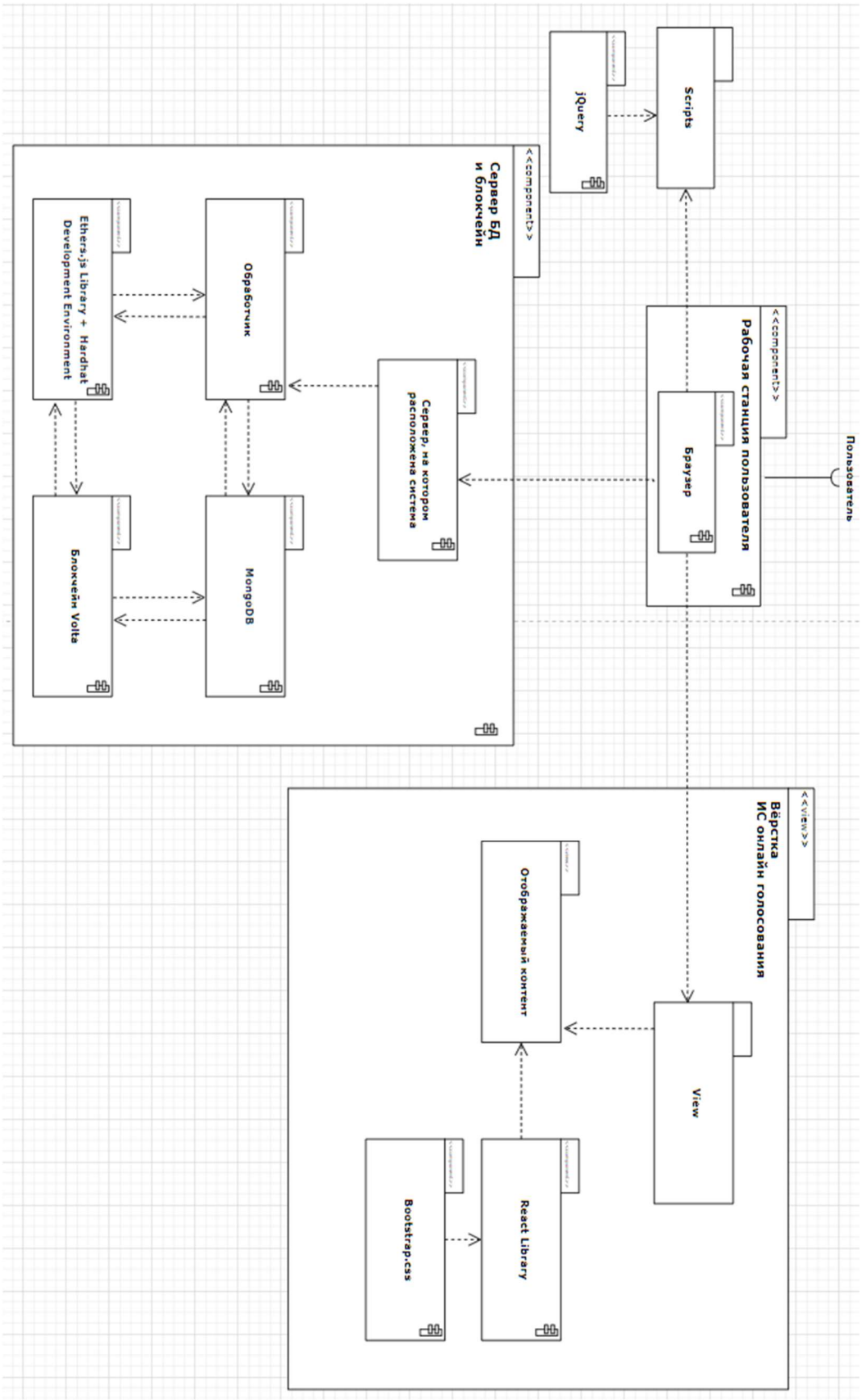

Рисунок А.5 – Диаграмма компонентов

ПРИЛОЖЕНИЕ Б

Листинг приложения

```
async function addCandidate() {
   try {
     if(account=='0xf8ab***fA'){
     const provider = new ethers.providers.Web3Provider(window.ethereum);
     await provider.send("eth_requestAccounts", []);
     const signer = provider.getSigner();
     const contractInstance = new ethers.Contract(
       contractAddress,
       contractAbi,
       signer      );
     const tx = await contractInstance.addCandidate(newCandidateName);
     await tx.wait();
     const updatedCandidates = await fetchCandidates();
     setCandidates(updatedCandidates);
     setNewCandidateName('');       }
     else{
       console.log("You don't have permissions");      }
   } catch (error) {
     console.error("Error adding candidate:", error);    }  }
  async function fetchCandidates() {
   const response = await fetch('/api/candidates');
   const candidates = await response.json();
   return candidates;  }
async function connectToMetamask() {
   if (window.ethereum) {
     try {
       const provider = new ethers.providers.Web3Provider(window.ethereum);
       setProvider(provider);
       await provider.send("eth_requestAccounts", []);
       const signer = provider.getSigner();
       const address = await signer.getAddress();
       setAccount(address);
       console.log("Metamask Connected : " + address);
       setIsConnected(true);
       canVote();
     } catch (err) {
       console.error(err);      }
   } else {
     console.error("Metamask is not detected in the browser")    }  }
```