МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2023 г.

Разработка системы электронного голосования на основе технологии блокчейн

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2023.308-276.ВКР

Научный руководитель,
доцент кафедры СП, к.ф.-м.н.,
доцент
_____ Г.И. Радченко

Автор работы,
студент группы КЭ-403
_____ В.О. Богатырева

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2023 г.

Челябинск, 2023 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
06.02.2023 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студентке группы КЭ-403
Богатыревой Веронике Олеговне,
обучающейся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 25.04.2023 г. № 753-13/12) Разработка системы электронного голосования на основе технологии блокчейн.
2. **Срок сдачи студентом законченной работы:** 05.06.2023 г.
3. **Исходные данные к работе**

3.1. Фролов А.В. Создание смарт-контрактов Solidity для блокчейна Ethereum. Практическое руководство. // ЛитРес: Самиздат, 2019. – 240 с.
3.2. Прасти Н. Блокчейн. Разработка приложений. // СПб.: БВХ-Петербург, 2018. – 256 с.
3.3. Трубочкина Н. К., Поляков С. В. Система электронного голосования на основе технологии блокчейн с использованием смарт-контракта. // Информационные технологии, 2019. – Т. 25. – № 2. – С. 75–85.
3.4. Metamask. [Электронный ресурс] URL: https://metamask.io/ (дата обращения: 11.02.2023 г.).

4. **Перечень подлежащих разработке вопросов**

4.1. Выполнить обзор литературы и существующих аналогов.
4.2. Спроектировать смарт-контракты для электронного голосования на основе технологии блокчейн.
4.3. Спроектировать веб-приложение для электронного голосования на основе технологии блокчейн.

4.4. Реализовать смарт-контракты и веб-приложение.

4.5. Провести тестирование работы приложения.

5. Дата выдачи задания: 06.02.2023 г.

Научный руководитель,
доцент кафедры СП, к.ф.-м.н., доцент Г.И. Радченко

Задание принял к исполнению В.О. Богатырева

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 6
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 8
1.1. Блокчейн 8
1.2. Блокчейн Ethereum 9
1.3. Смарт-контракт 9
1.4. Децентрализованное приложение 10
1.5. Подходы к реализации методов голосования с использованием блокчейн 11
1.5.1. Делегированное голосование 11
1.5.2. Голосование с использованием токенов 12
1.5.3. Весовое голосование 13
1.6. Анализ аналогичных проектов 13
1.6.1. Система онлайн-голосований Polys 13
1.6.2. Платформа «Московское голосование» 15
1.6.3. Сервис блокчейн-голосований WE.Vote 15
1.7. Краткий обзор технологий для разработки веб-приложений 16
1.7.1. Angular 16
1.7.2. Vue.js 17
1.7.3. React.js 17
2. ПРОЕКТИРОВАНИЕ 18
2.1. Функциональные требования к системе EVoting 18
2.2. Нефункциональные требования к системе EVoting 19
2.3. Диаграмма вариантов использования системы EVoting 20
2.4. Компоненты системы EVoting 22
2.4.1. Компоненты смарт-контрактов системы EVoting 23
2.4.2. Веб-сервер системы EVoting 24
2.4.3. Компоненты веб-интерфейса системы EVoting 24
2.5. Диаграмма деятельности прецедента создания голосования 25
2.6. Макеты веб-интерфейса приложения EVoting 26

3. РЕАЛИЗАЦИЯ 29
3.1. Программные средства реализации системы EVoting 29
3.2. Реализация компонентов смарт-контрактов системы EVoting ... 30
3.2.1. Реализация смарт-контракта EVotingManager 30
3.2.2. Реализация смарт-контракта голосования EVoting 30
3.2.3. Реализация смарт-контракта токена для голосования EVotingToken 33
3.3. Реализация веб-сервера системы EVoting 34
3.4. Реализация компонентов веб-интерфейса системы EVoting 35
3.4.1. Реализация компонента отображения всех голосований 35
3.4.2. Реализация компонента подключения веб3-провайдера MetaMask 35
3.4.3. Реализация компонента создания голосования 36
3.4.4. Реализация компонента проведения голосования 37
4. ТЕСТИРОВАНИЕ 39
4.1. Функциональное тестирование смарт-контрактов 39
4.2. Функциональное тестирование веб-приложения 39
ЗАКЛЮЧЕНИЕ 40
ЛИТЕРАТУРА 41
ПРИЛОЖЕНИЯ 45
Приложение А. Листинги исходного кода 45
Приложение Б. Тестирование приложения 49

ВВЕДЕНИЕ

Актуальность

В современном мире выборы являются важной задачей. Традиционные системы голосования имеют недостаточную прозрачность и безопасность. В существующих системах зачастую происходят манипуляции результатами голосования. Онлайн-голосование на блокчейне может избавить выборы от этих проблем.

Электронное голосование решает проблему явки избирателей: участники избирательного процесса могут голосовать из любой точки мира, где есть Интернет. Также данное решение позволяет уменьшить расходы на организацию выборов, распечатывание бюллетеней и открытие избирательных участков.

Постановка задачи

Целью выпускной квалификационной работы является разработка системы электронного голосования на основе технологии блокчейн. Для достижения поставленной цели необходимо решить следующие задачи:

1) выполнить обзор литературы и существующих аналогов;
2) спроектировать смарт-контракты для электронного голосования на основе технологии блокчейн;
3) спроектировать веб-приложение для электронного голосования на основе технологии блокчейн;
4) реализовать смарт-контракты и веб-приложение;
5) провести тестирование работы приложения.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения, списка литературы и двух приложений. Объем работы составляет 50 страниц, объем списка литературы – 48 источников.

В первой главе описываются предметная область, подходы к реализации методов голосования, аналогичные проекты по созданию электронного голосования и технологии для разработки веб-приложений.

Вторая глава посвящена определению функциональных и нефункциональных требований к системе и проектированию ее архитектуры.

Третья глава содержит в себе подробности и особенности реализации смарт-контракта и веб-приложения для электронного голосования.

В четвертой главе описывается процесс тестирования работы смарт-контракта и веб-приложения.

В приложениях содержатся листинги исходного кода системы электронного голосования, а также таблицы с тестированием смарт-контрактов и веб-приложения.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Блокчейн

Блокчейн – это особая структура данных, применяемая для создания децентрализованного регистра. Блокчейн состоит из блоков (block), особым образом соединенных в цепочку (chain). В каждом блоке содержатся транзакции, хеш предыдущего блока и время создания блока. Все блоки связаны в цепочку по хешу предыдущего блока. Каждый узел сети хранит полную копию блокчейна [1].

Вместо того чтобы обращаться к третьим лицам, например, финансово-кредитным организациям, в качестве посредников при проведении транзакций, узлы блокчейн-сети используют специальный протокол консенсуса для согласования содержимого реестра, а также криптографические алгоритмы хеширования и электронно-цифровые подписи для обеспечения целостности транзакции и передачи ее параметров [2].

В настоящее время блокчейн предлагает новые возможности для разработки приложений благодаря ключевым особенностям этой технологии, таким как прозрачность и защищенность процесса передачи данных.

Проблемы традиционных избирательных систем рассматриваются в работах [3–5]. Авторы утверждают, что существующие методы не могут обеспечить достаточный уровень прозрачности и надежности голосования.

В последнее время электронные системы голосования стали использоваться во многих странах. Эстония первой в мире внедрила электронную систему голосования на национальных выборах. Вскоре после этого электронное голосование было принято Швейцарией для выборов в масштабах штата и Норвегией для выборов в совет. Но данные электронные системы голосования требуют серьезных доработок в области безопасности и анонимности [4].

1.2. Блокчейн Ethereum

Блокчейн Ethereum представляет собой платформу, на базе которой можно создавать распределенные приложения DApp – программы, работа которых поддерживается распределенной сетью узлов. В отличие от других платформ, Ethereum позволяет использовать так называемые умные контракты (смарт-контракты, smart contracts), написанные на языке программирования Solidity [6]. Структура блокчейн Ethereum представлена на рисунке 1.

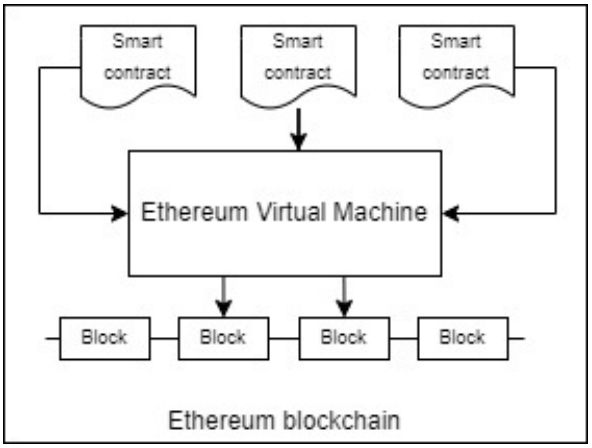

Рисунок 1 – Структура блокчейн Ethereum

Эта платформа была создана в 2013 году Виталиком Бутериным, основателем журнала Bitcoin Magazine, и запущена в 2015 году.

1.3. Смарт-контракт

Смарт-контракт представляет собой программный код, работающий в среде виртуальной машины Ethereum. Он запускается на всех узлах сети, и результаты его работы также реплицируются на все узлы.

С помощью смарт-контрактов очень удобно отслеживать выполнение транзакций. Если смарт-контракт получил тем или иным способом подтверждение выполнения условий сделки, то он может сам, автоматически, пере-

вести средства поставщику. Если условия сделки были выполнены не полностью или не выполнены вовсе, смарт-контракт может вернуть средства покупателю или перевести сумму штрафа на счет пострадавшей стороны.

Смарт-контракт может хранить данные, например, значения баланса, флаги, строки и числа, идентификаторы документов. Одной особенностью смарт-контракта является то, что его нельзя оспорить. Если логика смарт-контракта сработает таким образом, что средства будут переведены, эти средства уже невозможно будет вернуть. Достоинством смарт-контрактов является то, что на их работу требуется очень мало средств [2].

Как описано в статье [7], преимуществами применения смарт-контрактов в сочетании с технологией блокчейн в процессе голосования являются:

1) прозрачность процесса голосования – любой человек получит возможность контролировать ход голосования;

2) анонимность голоса – любой из избирателей генерирует индивадуальный приватный и публичный ключ, который он имеет право не разглашать другим участникам голосования;

3) подлинность и надежность результатов – результаты голосования невозможно сфальсифицировать, так как любой участник голосования может проверить сколько токенов-голосов было выпущено в начале голосования и как они распределялись по кошелькам после;

4) экономическая целесообразность и скорость обработки данных.

Перед публикацией смарт-контракта в сети Ethereum его необходимо компилировать в байт-код. Далее этот код сохраняется в сети с помощью транзакции.

1.4. Децентрализованное приложение

DApp, или децентрализованное приложение – приложение, которое базируются на технологии блокчейн совместно с механизмом

распределенного выполнения необходимых инструкций. В децентрализованных приложениях используются главные преимущества блокчейна: прозрачность, надежность и неизменность данных [8].

У децентрализованного приложения есть бэкенд-код, который работает в децентрализованной одноранговой сети. Децентрализованное приложение может иметь фронтенд-код и пользовательский интерфейс на любом языке (как и обычное приложение) для запросов к бэкенду. Также, фронтенд может быть размещен в децентрализованном хранилище, таком как IPFS [9].

DApp могут стать важным компонентом будущего без цензуры, однако и они не лишены недостатков. Децентрализованные приложения находятся на ранних стадиях развития, и им еще предстоит решить проблемы масштабируемости, модификации кода и небольшой базы пользователей [10].

1.5. Подходы к реализации методов голосования с использованием блокчейн

Блокчейн-голосования реализуются с помощью смарт-контрактов различными алгоритмами и методами. Рассмотрим делегированное голосование, голосование с использованием токенов и весовое голосование.

1.5.1. Делегированное голосование

В алгоритме делегированного голосования [11] лица, находящие в списке избирателей, могут либо голосовать сами, либо делегировать свой голос человеку, которому они доверяют.

Для этого подхода требуется указать адрес, на который будет начислен голос, а также проверить, голосовал ли избиратель и не совпадает ли его адрес с адресом делегата. После вызова метода делегирования, доверенное лицо имеет право голоса в блокчейн-голосовании.

Таким образом, в блокчейне содержатся данные о передаче права голоса другому лицу, а также транзакции о голосовании за выбранного кандидата.

1.5.2. Голосование с использованием токенов

Данный метод описывается в работах [7,12] на конкретном примере голосования. Для начала избиратель выбирает кандидата, на адрес которого будет переведен токен (голос). Эта транзакция пересылается в состоящую из компьютеров сеть равноправных узлов, называемых «нодами». Сеть нода подтверждает транзакцию, используя алгоритмы консенсуса. После подтверждения транзакция объединяется с другими подтвержденными транзакциями, формируя новый блок цифрового реестра. Затем данный блок добавляется в блокчейн с использованием хэша предыдущего блока.

Авторы утверждают, что транзакция, записанная в блокчейн, гарантирует ее достоверность и защищенность, а выбранный кандидат получает «голос», что автоматически отображается для всех наблюдателей (рисунок 2) [7].

Рисунок 2 – Алгоритм работы голосования с использованием токенов

1.5.3. Весовое голосование

Метод голосования с весами предоставляет возможность назначать и учитывать веса участников пропорционально их доле в уставном капитале общества. Чтобы голосование было легитимным, кворум должен составлять 50% + 1 участник организации [13]. Данный подход может применяться, к примеру, при голосовании акционеров.

В работе [11] рассматривается случай, когда наибольший вес голоса имеет председатель голосования, т.е. создатель смарт-контракта. Остальные участники голосования не имеют права решающего голоса.

В работах [14, 15] применяется метод весового голосования. В сервисе существует возможность проведения данных голосований. Также авторы указывают на возможность указать требования по кворуму, чтобы оно стало легитимным.

1.6. Анализ аналогичных проектов

1.6.1. Система онлайн-голосований Polys

Polys – система онлайн-голосований на блокчейне, созданная на базе «Лаборатории Касперского» [16]. За два года существования проекта на платформе проголосовали российские и зарубежные вузы, прошли голосования на выборах мэра Москвы, в Саратовский Молодежный Парламент.

Решение Polys представляет различные виды бюллетеней: с единичным и множественным выбором, с распределением кумулятивных голосов, бюллетени для референдума. В одних случаях не требуется анонимность, в других каждый голос участника голосования имеет определенный вес. Для этого авторы Polys обеспечили модульность системы [15].

Ключевые компоненты проекта:

1) блокчейн-платформа, которая предоставляет среду для выполнения смарт-контрактов, определяющих логику процесса голосования;

2) сервисный слой, отвечающий за аутентификацию пользователя, подпись зашифрованных сообщений и отправку уведомлений;

3) библиотека polys-protocol, которая обеспечивает слой абстракции над протоколом взаимодействия с блокчейном.

Блокчейн-платформа построена на базе фреймворка Exonum, написанного на языке программирования Rust.

Система работает следующим образом: зашифровываются сами голоса и шифруется личность избирателя. Каждый избиратель может проверить, что его голос принят и учтен в интерфейсе веб-приложения для голосования. Подсчет результатов происходит автоматически. Polys расшифровывает общий результат, а не каждый голос по отдельности. Это сделано для того, чтобы сохранить промежуточные результаты в тайне до конца голосования.

Рассмотрим панель создания голосования в системе. Пользователь может ввести название или основной вопрос голосования, задать ему фоновое изображение, ввести название организации. Существует возможность выбрать тип бюллетеня, создать список избирателей и период голосования. Затем необходимо добавить варианты ответов. Панель организатора при создании голосования изображена на рисунке 3 [16].

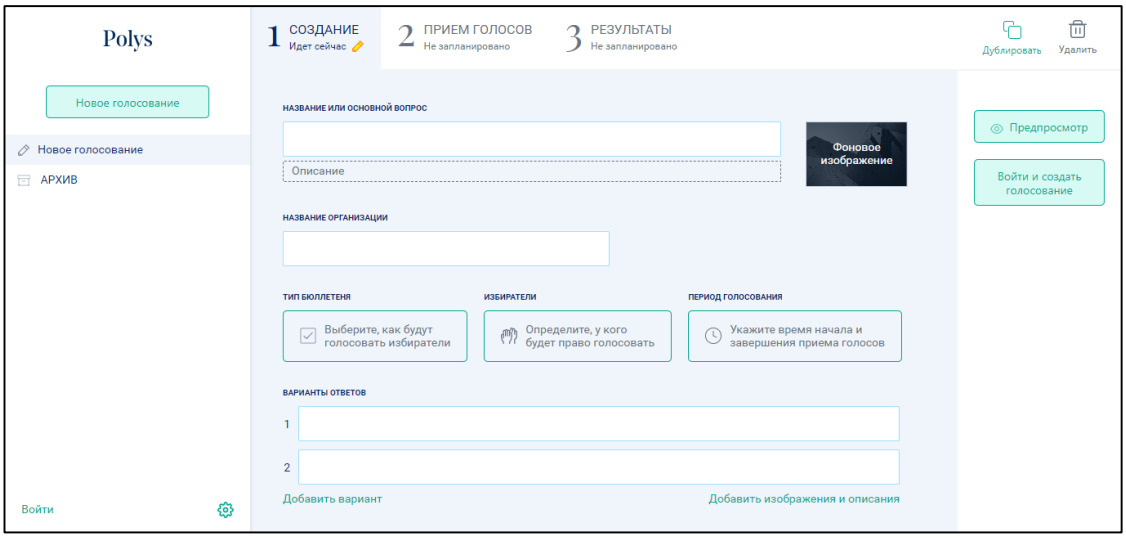

Рисунок 3 – Панель организатора при создании голосования

1.6.2. Платформа «Московское голосование»

Московское голосование – электронное голосование для жителей Москвы, которое проводится на портале мэра и правительства столицы [17]. На основе этого сервиса были проведены выборы в Мосгордуму и выборы муниципальных депутатов.

Для проведения онлайн-голосования используется блокчейн Ethereum с алгоритмом консенсуса PoA (Proof of Authority). В данной системе любой желающий может отслеживать все происходящее в блокчейне, но создавать новые блоки могут только узлы-валидаторы. За основу было взято программное обеспечение портала mos.ru. Анонимность избирателя гарантирует прокси-сервер, который генерирует уникальные ссылки на бюллетень, формируемые случайным образом [18].

Для участия в голосовании нужно было оставить заявку на включение в реестр избирателей на сайте mos.ru. Во время обработки заявки все необходимые данные, которые указал избиратель, проходят проверку. В день голосования избиратель переходит в раздел «Услуги» электронных выборов и получает доступ к форме голосования. Далее избиратель проставляет галочки напротив выбранных вариантов ответа и нажимает «Проголосовать». Анонимайзер генерирует ссылку и доступ к анонимному бюллетеню через личный кабинет. Блокчейн обеспечивает сохранение данных в ходе голосования и их интерпретацию по завершении голосования. Он размещается в отдельном защищенном сегменте сети внутри центра обработки данных [19].

1.6.3. Сервис блокчейн-голосований WE.Vote

WE.Vote – сервис блокчейн-голосований, разработанный российской компанией Waves Enterprise [20, 21]. Сервис WE.Vote наиболее полно реализует концепцию безопасной системы онлайн-голосования. Сервис основан на блокчейн-сети Waves Enterprise Mainnet.

Со стороны WE.Vote выглядит как обычный сервис дистанционных голосований. Организатор голосования заходит в веб-интерфейс сервиса, создает новое голосование, настраивает бюллетени и добавляет электронные адреса участников. Далее бюллетени рассылаются участникам, система подсчитывает голоса и выдает готовый отчет с итогами голосования. Порядок не отличается от привычного для традиционных голосований.

На рисунке 4 изображена главная страница, на которой отображаются все голосования пользователя [20].

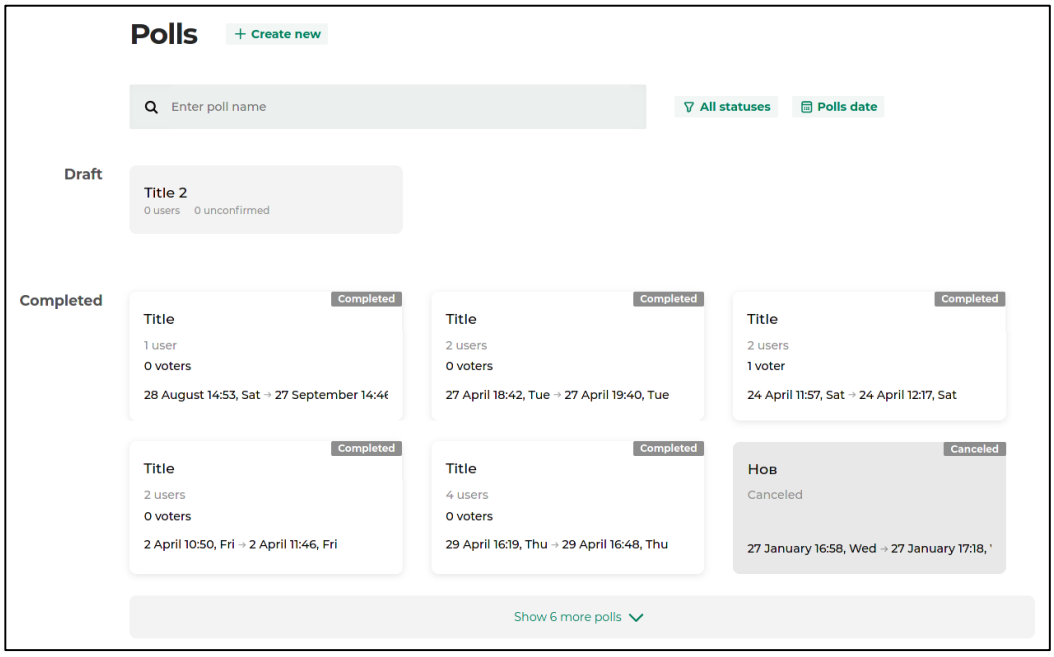

Рисунок 4 – Главная страница WE.Vote

1.7. Краткий обзор технологий для разработки веб-приложений

Рассмотрим основные технологии для разработки веб-приложений. Данные технологии повышают производительность приложений, так как содержат в себе шаблоны и наборы функций.

1.7.1. Angular

Angular – это JavaScript-фреймворк от Google [22, 23], совместимый с большинством распространенных редакторов кода. Angular предназначен для создания динамических одностраничных веб-приложений (SPA – Single

Page Applications). Фреймворк использует компонентный подход, а также преобразовывает документы на основе HTML [24] в динамический контент.

1.7.2. Vue.js

Vue.js – это фреймворк с открытым исходным кодом для одностраничных приложений, который требует знания HTML и CSS [25, 26]. Он использует модель разработки на основе компонентов и позволяет присоединять компоненты к проекту. Vue известен небольшим размером документов и синтаксисом на основе HTML. Так как фреймворк является самым «молодым», размер сообщества разработчиков небольшой.

1.7.3. React.js

React.js – это библиотека для создания динамических пользовательских интерфейсов, которая была разработана компанией Facebook [27, 28]. React основан на JavaScript [29] и JSX и позволяет создавать HTML-элементы для многократного использования. По сравнению со своими аналогами React достаточно быстро справляется с прогрузкой и ререндерингом страницы. Данная библиотека является наиболее популярной технологией для разработки децентрализованных приложений.

Выводы по первой главе

В этой главе были рассмотрены предметная область проекта, подходы к реализации методов голосования, аналогичные проекты и технологии для разработки веб-приложений.

Таким образом, было принято решение реализовать систему электронного голосования с использованием токенов на основе технологии блокчейн. В качестве инструментов разработки смарт-контракта был выбран блокчейн Ethereum и язык программирования Solidity. Для разработки веб-приложения была выбрана библиотека React.

2. ПРОЕКТИРОВАНИЕ

Целью данной работы является разработка системы EVoting, которая представляет собой веб-приложение для проведения электронных голосований с использованием технологии блокчейн.

С помощью веб-приложения EVoting любому гостю сайта доступен просмотр списка всех голосований и подробных данных о каждом голосовании: название или основной вопрос, сроки и статус голосования (не началось, идет и завершено), варианты ответа и, если голосование завершено, его результаты. Для каждого созданного голосования отображается индивидуальная ссылка в блокчейн-обозреватель Etherscan [30]. Гость сайта может найти голосование по названию или основному вопросу, а также отфильтровать голосования по их статусу. Чтобы авторизоваться в системе, гость должен подключить веб3 провайдер MetaMask [31] для работы с блокчейном.

Авторизованный пользователь может создать новое голосование, заполнив поля формы и подписав транзакцию для добавления голосования в блокчейн в MetaMask. Данный вид пользователя может проголосовать, если он приглашен в определенное голосование, голосование еще не завершено и, если пользователь подписал транзакцию перевода токена (голоса) выбранному варианту ответа в MetaMask. В случае, если авторизованный пользователь проголосовал, при просмотре подробных данных о голосовании отображается ссылка на транзакцию «голоса» в блокчейн-обозреватель Etherscan.

2.1. Функциональные требования к системе EVoting

Можно выделить следующий набор функциональных требований к системе EVoting.

1. Система EVoting должна предоставлять гостю веб-приложения возможность авторизоваться с помощью веб3 провайдера MetaMask.

2. Система EVoting должна предоставлять гостю и авторизованному пользователю веб-приложения возможность просмотреть список всех голосований.

3. Система EVoting должна предоставлять гостю и авторизованному пользователю веб-приложения возможность просмотреть подробные данные о каждом голосовании.

4. Система EVoting должна предоставлять гостю и авторизованному пользователю веб-приложения возможность найти голосование по названию или основному вопросу.

5. Система EVoting должна предоставлять гостю и авторизованному пользователю веб-приложения возможность отфильтровать голосования по их статусу.

6. Система EVoting должна предоставлять авторизованному пользователю веб-приложения возможность создать голосование.

7. Система EVoting должна предоставлять авторизованному пользователю веб-приложения возможность проголосовать и просмотреть транзакцию «голоса» в блокчейн-обозревателе Etherscan.

2.2. Нефункциональные требования к системе EVoting

Можно выделить следующие нефункциональные требования к системе.

1. Система EVoting должна обеспечивать возможность проверки процесса голосования в режиме реального времени.

2. Смарт-контракты должны быть написаны на языке программирования Solidity.

3. Веб-приложение должно быть доступно в браузерах Google Chrome, FireFox, Яндекс Браузер текущих актуальных версий.

4. Веб-приложение должно быть разработано с использованием таких инструментов, как: React, TypeScript, MUI, Redux Toolkit.

2.3. Диаграмма вариантов использования системы EVoting

Язык объектного моделирования UML использовался для разработки системы EVoting. Была создана диаграмма вариантов использования (рисунок 5).

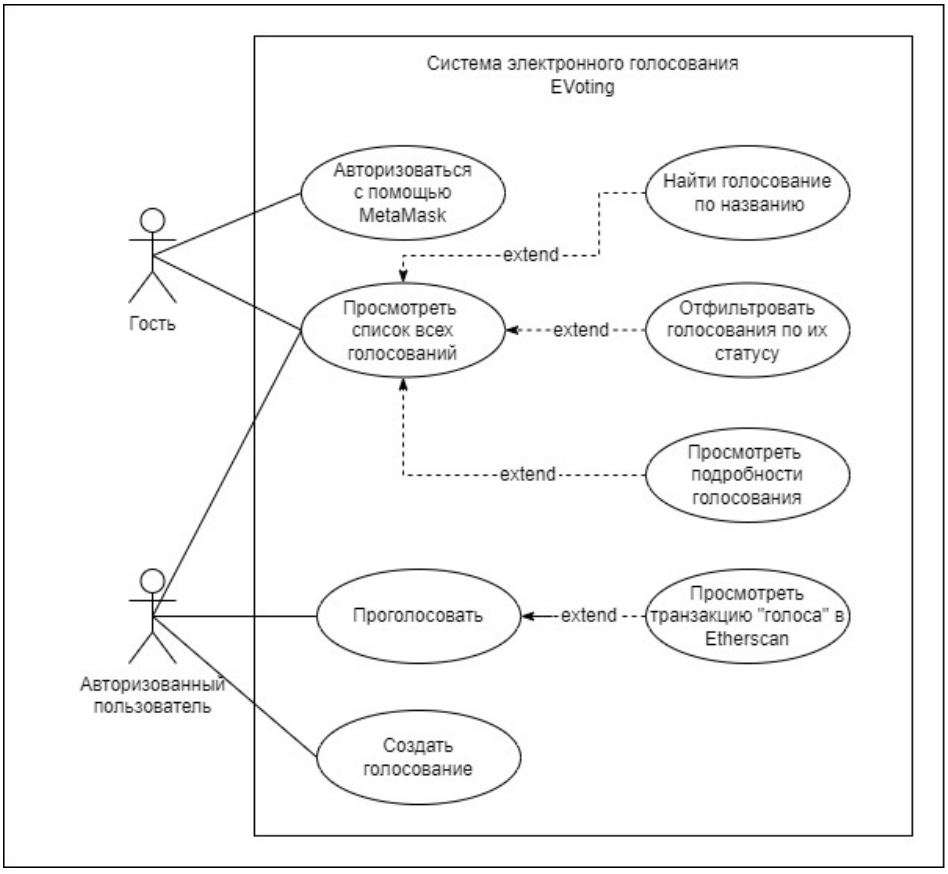

Рисунок 5 – Диаграмма вариантов использования системы электронного голосования EVoting

В системе определены следующие виды акторов.

1. *Гость* – это посетитель веб-приложения, который может авторизоваться с помощью MetaMask и просматривать голосования.

2. *Авторизованный пользователь* – это авторизованный посетитель веб-приложения, который имеет доступ к просмотру голосований, их созданию и к участию в голосовании.

Актору «Гость» доступны следующие варианты использования системы.

1. Гость может авторизоваться в системе EVoting с помощью web3 провайдера MetaMask.

2. Гость может просмотреть список всех голосований в веб-приложении.

3. Гость может просмотреть подробные данные о голосовании, выбрав определенное голосование из списка. Подробными данными о голосовании являются название или основной вопрос, сроки голосования (дата и время начала и окончания), статус голосования (не началось, идет и завершено), варианты ответа, ссылку на смарт-контракт голосования в блокчейн-обозреватель Etherscan и, если голосование завершено, его результаты.

4. Гость может найти голосование по его названию или основному вопросу.

5. Гость может отфильтровать голосования по их статусу (не началось, идет и завершено).

Авторизованному пользователь может выполнять действия, описанные выше для актора «Гость», кроме авторизации в системе EVoting с помощью web3 провайдера MetaMask. Также ему доступны следующие варианты использования системы.

1. Авторизованный пользователь может создать голосование. Для этого необходимо ввести название, сроки голосования, публичные идентификаторы избирателей, созданные в MetaMask, электронные почты избирателей и варианты ответов, а затем подписать транзакцию создания смарт-контракта в MetaMask.

2. Авторизованный пользователь может проголосовать, подписав транзакцию перевода токена (голоса) выбранному варианту ответа в MetaMask.

2.4. Компоненты системы EVoting

Архитектура системы EVoting изображена на рисунке 6. Данная архитектура состоит из следующих блоков.

1. Блокчейн Ethereum – глобально доступная детерминированная машина состояний, поддерживаемая одноранговой сетью узлов. Смарт-контракты хранятся и работают на блокчейне Ethereum. Виртуальная машина Ethereum выполняет выполняет логику, определенную в смарт-контрактах, и обрабатывает изменения состояния.

2. Веб-интерфейс определяет логику пользовательского интерфейса и взаимодействует с блокчейном с помощью веб3 провайдера MetaMask. В веб-интерфейсе системы авторизация пользователя осуществляется через MetaMask, так как MetaMask является браузерным расширением.

3. Веб3 провайдер MetaMask – узел сети, к которому подключается веб-интерфейс для авторизации и взаимодействия с блокчейном.

4. Веб-сервер обрабатывает логику приложения, определенную в смарт-контрактах. Также на веб-сервере собирает данные о голосованиях из блокчейна и записывает в него данные о созданых голосованиях. Для данного взаимодействия необходим «серверный» веб3 провайдер Alchemy.

5. Веб3 провайдер Alchemy – узел сети, к которому подключается веб-сервер для взаимодействия с блокчейном и смарт-контрактами.

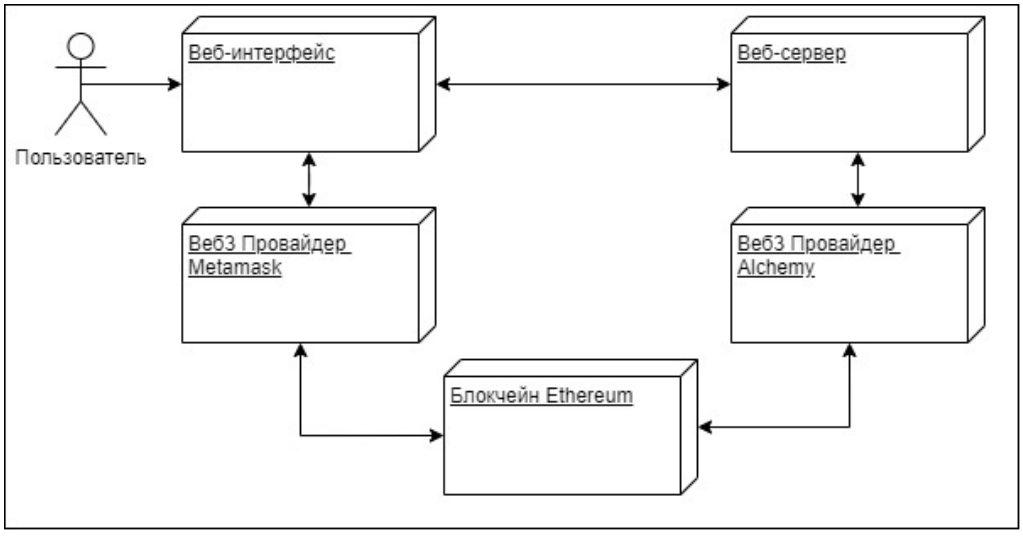

Рисунок 6 – Архитектура системы EVoting

2.4.1. Компоненты смарт-контрактов системы EVoting

Диаграмма смарт-контрактов (рисунок 7) была построена для визуального представления архитектуры смарт-контрактов, используемых в системе EVoting.

В управляющем смарт-контракте EVotingManager хранятся публичные ключи всех созданных смарт-контрактов голосований.

Программная логика каждого голосования находится в смарт-контракте для голосования EVoting. Данный смарт-контракт содержит поля, определяющие голосование: его название, дату и время начала и окончания, список публичных ключей избирателей, список вариантов ответов. За начисление голоса конкретному выбору избирателя отвечает метод vote. Для подсчета голосов представлен метод totalVotesFor.

В голосованиях используется токен EVotingToken, который отвечает за начисление токена избирателю на адрес кошелька и перевод этого токена на адрес кошелька кандидата. Данный токен использует стандарт токенов в блокчейне Ethereum ERC20. Все данные о голосованиях и действия с ними записываются в блокчейн Ethereum.

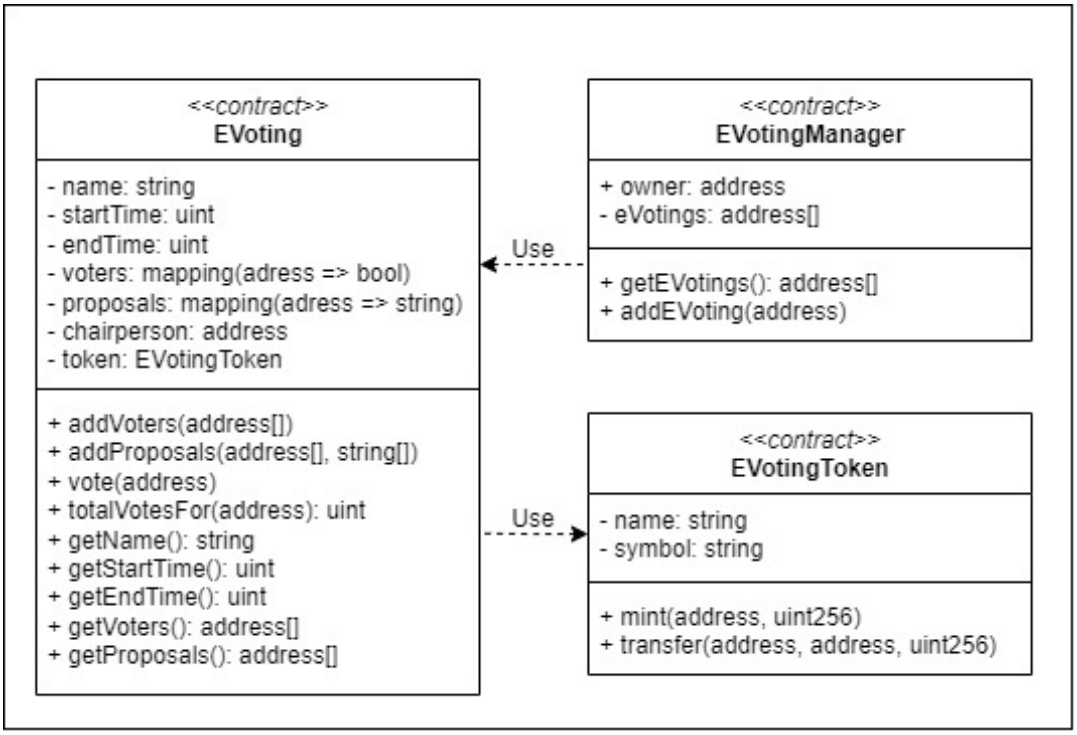

Рисунок 7 – Диаграмма смарт-контрактов системы EVoting

2.4.2. Веб-сервер системы EVoting

Веб-сервер позволяет взаимодействовать со смарт-контрактами с помощью REST API. Рассмотрим следующие запросы, доступные пользователю.

1. GET /get-voting-list – получение списка всех голосований системы, используя данные, хранящиеся в управляющем смарт-контракте. Каждое голосование в списке представлено в виде объекта со свойствами: адрес смарт-контракта голосования, название, дата начала и дата окончания, список участников-избирателей, а также список вариантов ответов.

2. POST /create-voting – добавление нового голосования в систему. В теле запроса пользователю нужно указать данные голосования: название, дату начала и окончания, список участников и вариантов ответов. Переданные параметры конвертируются в байт-код, который затем используется для развертывания смарт-контракта голосования в блокчейн. В результате вызова запроса пользователю возвращается адрес созданного голосования в сети блокчейна. Полученный адрес голосования добавляется в управляющий смарт-контракт.

2.4.3. Компоненты веб-интерфейса системы EVoting

Архитектура веб-интерфейса EVoting состоит из следующих компонентов.

1. Компонент отображения голосований, который получает данные о голосованиях с помощью веб-сервера и представляет их на главной странице веб-приложения.

2. Компонент подключения веб3-провайдера MeaMask необходимый для авторизации пользователя в системе.

3. Компонент голосования, в котором отображается информация о голосовании, его результаты, а также предоставляется возможность

пользователю проголосовать, если данный пользователь учавствует в голосовании.

4. Компонент создания голосования, в котором пользователь может заполнить необходимые поля и создать голосование в блокчейне.

Данная архитектура изображена на рисунке 8.

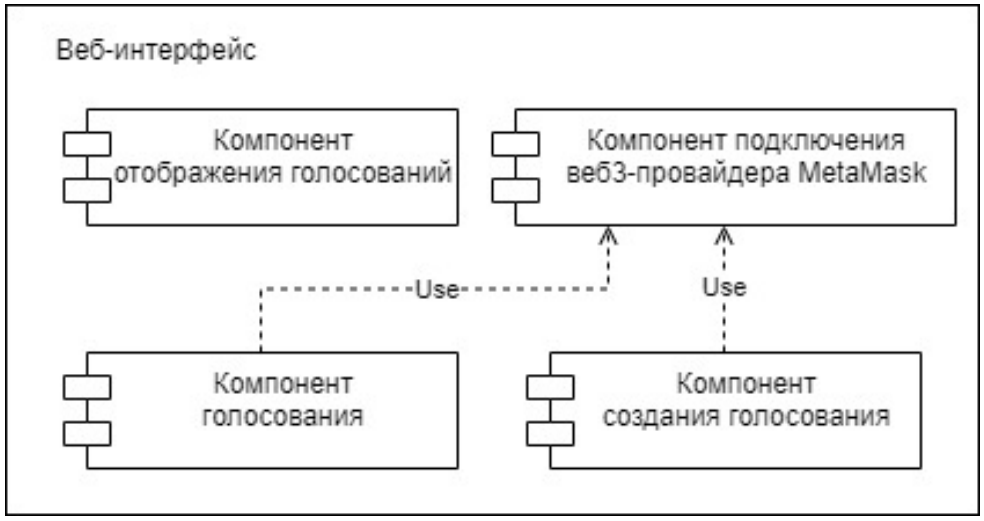

Рисунок 8 – Диаграмма компонентов веб-интерфейса EVoting

2.5. Диаграмма деятельности прецедента создания голосования

В результате анализа требований реализована диаграмма деятельности, которая показывает, как происходит процесс взаимодействия актеров с системой. Рассмотрим прецедент «Создать голосование». Предусловием является то, что пользователь находится в разделе «Создать голосование».

В представленной диаграмме пользователь заполняет поля формы создания голосования: название, сроки голосования, варианты ответов и публичные идентификаторы избирателей, созданные в MetaMask. Затем пользователь нажимает кнопку «Создать голосование». После этого веб-интерфейс формирует запрос для развертывания смарт-контракта голосования. Пользователю нужно подписать транзакцию создания смарт-контракта голосования с помощью MetaMask.

На веб-сервере системы выполняется развертывание смарт-контракта голосования в блокчейне. После развертывания смарт-контракт производит

начисление токенов (голосов) на публичные идентификаторы участников голосования. После этого веб-сервер выполняет транзакцию добавления нового адреса только что созданного смарт-контракта голосования в блокчейн. В свою очередь, веб-интерфейс переводит пользователя на страницу со всеми голосованиями. Данная диаграмма деятельности представлена на рисунке 9.

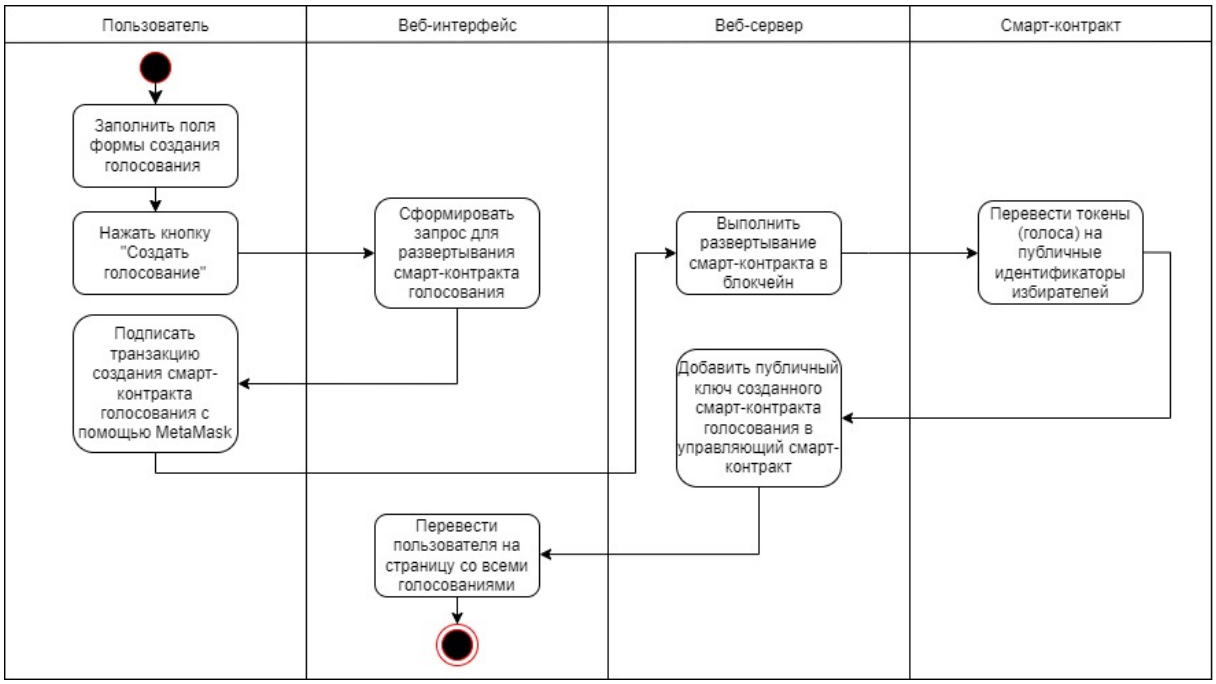

Рисунок 9 – Диаграмма деятельности прецедента создания голосования

2.6. Макеты веб-интерфейса приложения EVoting

Макет – это эскиз, который используется для проектирования, а также определяет внешний вид сайта и то, как пользовтель будет с ним взаимодействовать.

На главной странице приложения отображаются все голосования системы. В правом верхнем углу макета сайта изображена кнопка для авторизации с помощью MetaMask. Для авторизованного пользователя в интерфейсе есть кнопка для создания голосования. Пользователю веб-приложения доступны строка для поиска голосований и их фильтрации на

основе статуса голосования. Макет главной страницы веб-интерфейса приложения EVoting изображен на рисунке 10.

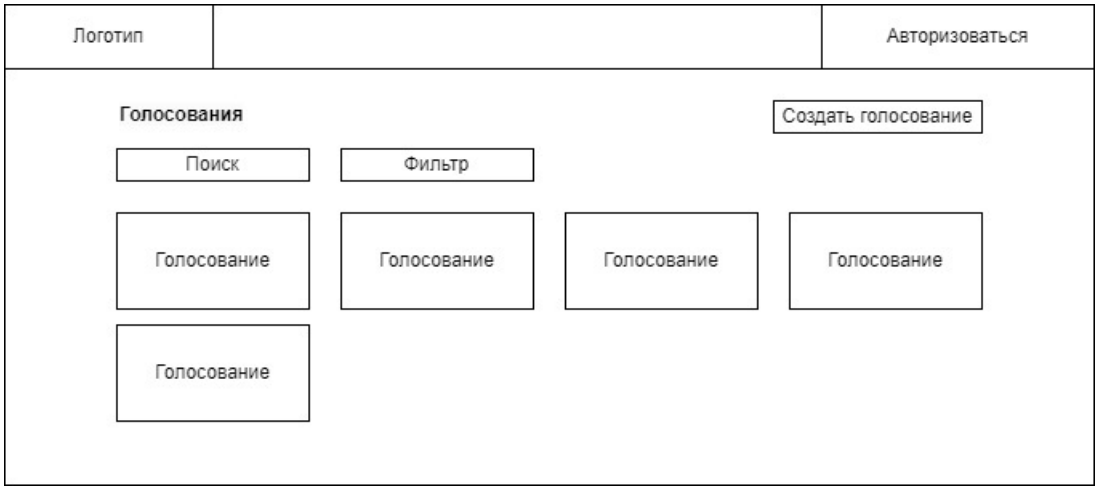

Рисунок 10 – Макет главной страницы

На рисунке 11 изображен макет страницы голосования. На странице голосования отображаются название, данные о голосовании (сроки, ссылка на смарт-контракт голосования в обозревателе блокчейна) и варианты ответов. Также есть кнопка «Проголосовать» и кнопка «Закрыть» для перехода на главную страницу сайта.

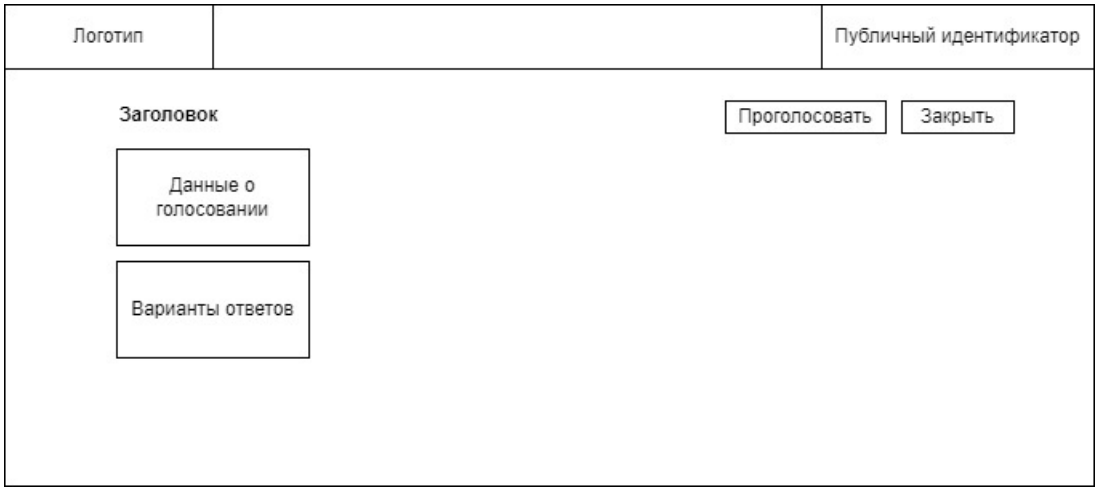

Рисунок 11 – Макет страницы голосования

На странице создания голосования изображены поля для ввода данных о голосовании и кнопка для создания голосования. Кнопка

«Закрыть» предназначена для перехода на главную страницу сайта. Макет страницы создания голосования представлен на рисунке 12.

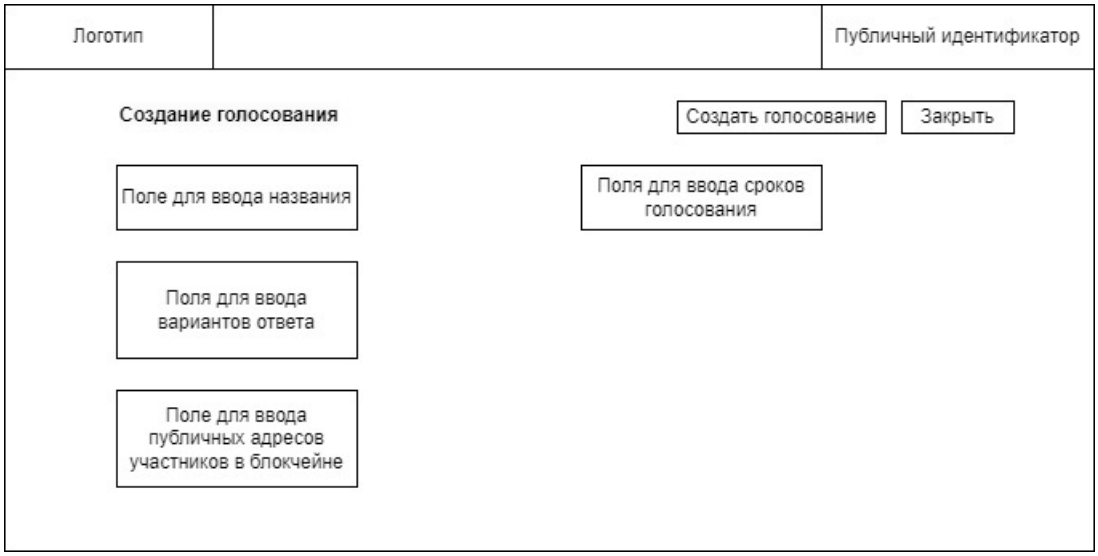

Рисунок 12 – Макет страницы создания голосования

В соответствии с данными макетами был разработан дизайн сайта [32]. Для создания макетов использовался онлайн редактор Figma и дизайн-система для создания интерфейсов Material Design [33, 34]. На рисунке 13 представлен фирменный стиль веб-приложения.

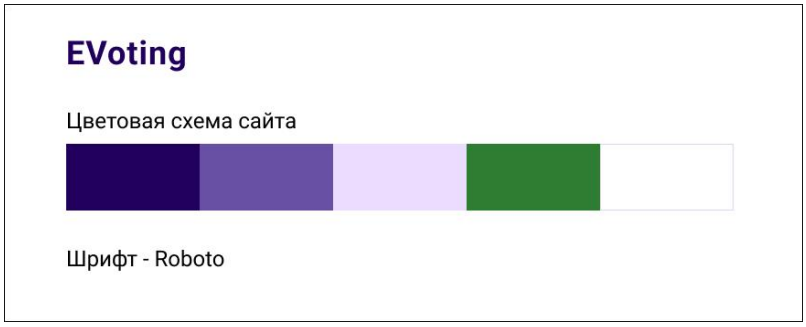

Рисунок 13 – Фирменный стиль веб-приложения

Выводы по второй главе

В процессе анализа требований были определены функциональные и нефункциональные требования. Кроме этого были составлены диаграмма вариантов использования, диаграммы компонентов и диаграмма деятельности системы, макеты и дизайн веб-приложения.

3. РЕАЛИЗАЦИЯ

3.1. Программные средства реализации системы EVoting

Для реализации смарт-контрактов был выбран язык программирования Solidity с использованием библиотек и модулей, описанных ниже [35].

1. Библиотека OpenZeppelin – библиотека, предназначенная для разработки смарт-контрактов, которая реализует стандарты в блокчейне Ethereum [36], в том числе стандарт ERC20 (набор правил для создания стандартного токена [37]).

2. Etherscan – обозреватель блоков, предоставляющий доступ к публичной информации о транзакциях, смарт-контрактах, адресах и других данных сети Ethereum [30].

Разработка смарт-контрактов велась в среде разработки Remix IDE [38].

Серверная часть системы EVoting была реализована с использованием веб-фреймворка Express.js и среды Node.js [39, 40]. Также были использованы Ethers.js и Alchemy API для взаимодействия с блокчейном Ethereum [41, 42].

Для реализации клиентской части приложения был использован язык программирования JavaScript и библиотека React [27, 29]. Управление состоянием данных и пользовательским интерфейсом приложения реализовано при помощи Redux и Redux Toolkit [43, 44]. Компоненты были стилизованы с использованием библиотеки MUI и Emotion.js [45, 46]. Взаимодействие с блокчейном происходит через MetaMask и библиотеку web3.js [31, 47].

Разработка серверной и клиентской части приложения велась в редакторе исходного кода Visual Studio Code [48].

3.2. Реализация компонентов смарт-контрактов системы EVoting

3.2.1. Реализация смарт-контракта EVotingManager

В смарт-контракте `EVotingManager` реализована логика хранения, добавления и получения адресов смарт-контрактов голосований `EVoting`.

В рассматриваемом смарт-контракте есть переменная адреса владельца, для которой задан адрес того, кто развертывает контракт, в функции конструктора. Контракт также содержит массив адресов `eVotings`, в котором хранятся адреса всех контрактов голосований, добавленных в систему.

Функция `getEVotings` – это функция, которая возвращает массив адресов `eVoting`. Функция `addEVoting` – это общедоступная функция, которую может вызвать кто угодно. Однако для этого требуется, чтобы вызывающая сторона была владельцем контракта. Эта функция добавляет новый адрес контракта в массив `eVotings`.

В целом, этот контракт позволяет управлять несколькими контрактами электронного голосования и отслеживать их централизованно. В листинге 1 приложения А представлена реализация смарт-контракта `EVotingManager`.

3.2.2. Реализация смарт-контракта голосования EVoting

Смарт-контракт `EVoting` реализует логику проведения голосования в системе. Рассмотрим конструктор смарт-контракта, вызываемый при развертывании контракта голосования в блокчейне (листинг 1). В качестве входных параметров конструктор принимает название голосования, время начала и конца голосования, массив адресов участников голосования, массив адресов и названий вариантов ответов.

Конструктор инициализирует несколько переменных, включая адрес председателя, экземпляр контракта `EVotingToken` и несколько сопоставле-

ний для отслеживания избирателей, вариантов ответов и голосов. Конструктор также вызывает две функции, `addVoters` и `addProposals`, для добавления избирателей и вариантов ответов в контракт.

Листинг 2 – Конструктор смарт-контракта голосования

```
constructor(string memory _name, uint256 _startTime, uint256 _endTime,
    address[] memory _voters, address[] memory _proposalsAddr,
    string[] memory _proposalsNames) {
    chairperson = msg.sender;
    name = _name;
    startTime = _startTime;
    endTime = _endTime;

    votersAddresses = _voters;
    proposalsAddresses = _proposalsAddr;
    proposalsNames = _proposalsNames;
    addVoters(_voters);
    addProposals(_proposalsAddr, _proposalsNames);
}
```

Модификатор – это особый тип функции, который можно использовать для изменения поведения других функций. Модификатор `onlyChairperson` проверяет, является ли вызывающий функцию создателем контракта голосования. Если вызывающий абонент не является создателем, модификатор выдает сообщение об ошибке.

Модификатор `voteIsOn` используется для проверки того, находится ли текущее время в пределах времени начала и окончания процесса голосования. Если текущее время находится за пределами периода голосования, модификатор выдает сообщение об ошибке. Реализация модификаторов представлена в листинге 2.

Листинг 2 – Реализация модификаторов onlyChairperson и voteIsOn

```
modifier onlyChairperson() {
    require(msg.sender == chairperson, "This isn't chairperson.");
    _;
    }

modifier voteIsOn() {
    require(
      block.timestamp >= startTime && block.timestamp <= endTime,
      "Voting completed."
    );
    _;
}
```

В методе `addVoters` реализовано добавление публичных идентификаторов участников голосования в смарт-контракт и начисление токенов для голосования. В листинге 3 представлена реализация данного метода. Метод использует модификатор `onlyChairperson`. В качестве аргумента метод принимает массив адресов участников голосования.

Листинг 3 – Реализация метода добавления голосования

```
function addVoters(address[] memory _voters) public onlyChairperson {
    for (uint256 i = 0; i < _voters.length; i++) {

      require(!proposalsExist[_voters[i]], "Proposal can't be voter.");
      require(!votersExist[_voters[i]], "Voter exists.");

      voters[_voters[i]] = false;
      votersExist[_voters[i]] = true;
      token.mint( voters[i], 100000000000000000000);
    }
}
```

Добавление адресов вариантов ответов и их названий в смарт-контракт голосования реализовано в методе `addProposals`. Функция принимает два массива, один из которых содержит адреса вариантов ответов, а другой — их названия. Реализация метода представлена в листинге 4.

Листинг 4 – Реализация метода добавления вариантов ответов

```
function addProposals(
    address[] memory _proposalsAddr,
    string[] memory _proposalsNames
    ) public onlyChairperson {
    require(
      _proposalsAddr.length == _proposalsNames.length,
      "Length is different."
    );

    for (uint256 i = 0; i < _proposalsAddr.length; i++) {
      proposals[_proposalsAddr[i]] = _proposalsNames[i];
      proposalsExist[_proposalsAddr[i]] = true;
    }
}
```

Метод `vote` позволяет избирателю отдать свой голос за определенный вариант ответа. Функция принимает адрес варианта ответа, за которое избиратель хочет проголосовать. Сначала функция проверяет, существует ли голосующий, достаточно ли у него токенов для голосования и существует ли

такой кандидат в голосовании. Если все условия соблюдены, функция переводит токен от голосующего на адрес кандидата (листинг 5).

Листинг 5 – Метод голосования за определенный вариант ответа

```
function vote(address proposal) public voteIsOn {
    require(votersExist[msg.sender], "Voter doesn't exist.");
    require(token.balanceOf(msg.sender) > 0, "Token doesn't exist.");
    require(proposalsExist[proposal], "Proposal doesn't exist.");
    require(!voters[msg.sender], "Voter already voted.");

    token.transfer(proposal, 100000000000000000000);
    voters[msg.sender] = true;
    votesFor[msg.sender] = proposal;
}
```

В методе `totalVotesFor()` реализован подсчет количества голосов после окончания голосования (листинг 6). Аргументом метода является адрес варианта ответа в блокчейне.

Листинг 6 – Метод подсчета количества голосов

```
function totalVotesFor(address proposal) public view returns (uint256) {
    require(proposalsExist[proposal], "Proposal doesn't exist.");
    return token.balanceOf(proposal);
}
```

3.2.3. Реализация смарт-контракта токена для голосования EVotingToken

Смарт-контракт `EVotingToken` используется для создания пользовательского токена для системы электронного голосования (листинг 2 приложения А). Контракт наследуется от двух других контрактов, `ERC20` и `Ownable`. `ERC20` – это стандартный интерфейс для токенов в блокчейне Ethereum, а `Ownable` — это контракт, обеспечивающий контроль доступа только создателю голосования. Данные стандарты реализованы библиотекой OpenZeppelin [36].

Конструктор устанавливает имя и символ токена в «EVotingToken» и «EVT» соответственно. Контракт также определяет две функции: `mint` и `transfer`. Функция `mint` используется для создания новых токенов и их

присвоения определенному адресу. Функция `transfer` используется для перевода токенов с одного адреса на другой.

3.3. Реализация веб-сервера системы EVoting

Веб-сервер представляет собой REST API.

В методе `getVotingList` реализуется извлечение списка голосований из блокчейна Ethereum с использованием веб3-провайдера Alchemy (листинг 3 приложения А).

Для того, чтобы получить данные о каждом голосовании в тестовой сети «Sepolia» необходимо знать адреса смарт-контрактов голосований в блокчейне. Эти данные были взяты из смарт-контракта `EVotingManager`.

Для каждого адреса контракта голосования создается экземпляр контракта, используя его ABI, а затем извлекается информация о голосовании, такая как его название, время начала и окончания, список участников и список вариантов ответов. Если голосование закончилось, дополнительно извлекается результат (общее количество голосов в каждом варианте ответа).

Метод `createVoting` создает новый экземпляр смарт-контракта электронного голосования `EVoting` в блокчейне Ethereum, а также добавляет адрес контрата в `EVotingManager`. Функция принимает параметры: название голосования, дату и время начала и окончания, список избирателей и список вариантов ответов. Данный метод представлен в листинге 4 приложения А.

Рассматриваемая функция сначала генерирует список случайных публичных ключей для каждого варианта ответа, а затем кодирует входные параметры с помощью `AbiCoder`. Затем создается новый экземпляр класса `ContractFactory`, используя байт-код и ABI смарт-контракта `EVoting`.

Используя метод `deploy`, развертывается смарт-контракт голосования в блокчейне. После этого адрес развернутого контракта добавляется в смарт-контракт `EVotingManager` с помощью метода `addEVoting`. Функция возвращает адрес развернутого смарт-контракта.

3.4. Реализация компонентов веб-интерфейса системы EVoting

3.4.1. Реализация компонента отображения всех голосований

Компонент `MainPage` отображает главную страницу приложения для голосования. Он показывает список карточек голосований, которые можно отфильтровать по названию и статусу (листинг 5 приложения А). Компонент получает список данных всех голосований с помощью функции `getVotingList`. Компонент также позволяет авторизованному пользователю создать новое голосование, нажав кнопку для перехода на страницу создания голосования. Во время загрузки списка голосования отображается индикатор загрузки.

На рисунке 14 изображен скриншот главной страницы приложения.

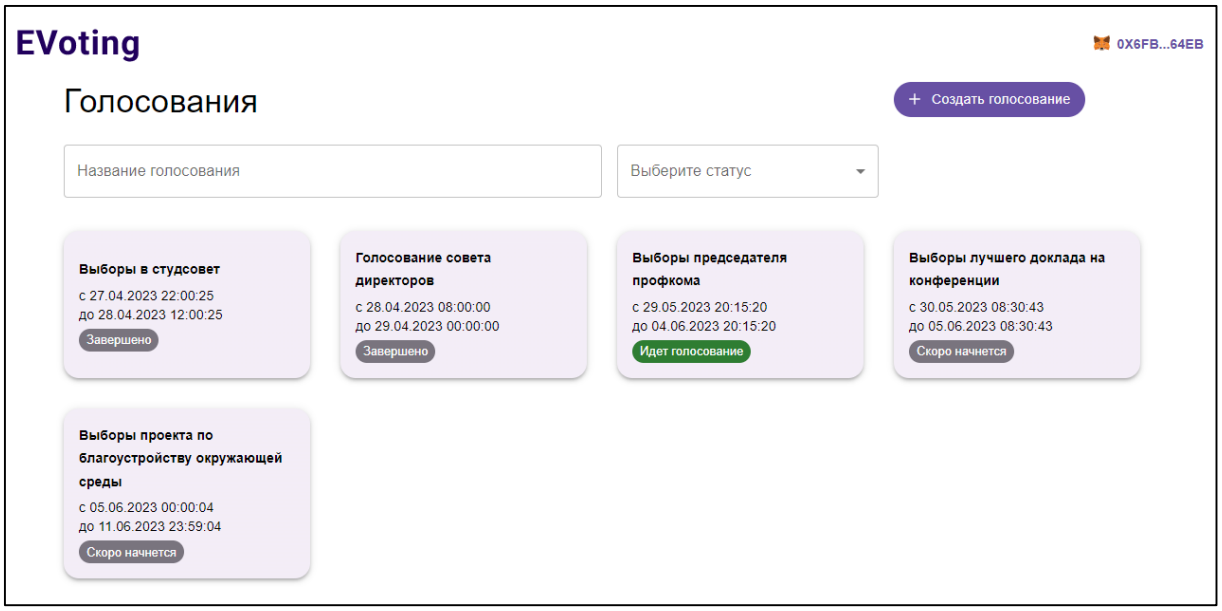

Рисунок 14 – Скриншот главной страницы приложения

3.4.2. Реализация компонента подключения веб3-провайдера MetaMask

В качестве веб3-провайдера в веб-интерфейсе используется MetaMask [31]. Чтобы подключиться к данному провайдеру, пользователю необходимо нажать на кнопку «Авторизоваться». Обработчик нажатия на данную кнопку вызывает метод `login` (листинг 6 приложения А).

В методе происходит вызов MetaMask, в котором пользователю нужно выбрать аккаунт, через который он подключается, а затем нажать кнопку «Подключиться». После подключения веб3 провайдера в веб-интерфейсе приложения вместо кнопки авторизации отображается выбранный пользователем аккаунт MetaMask. На рисунке 15 изображен процесс подключения веб3-провайдера MetaMask.

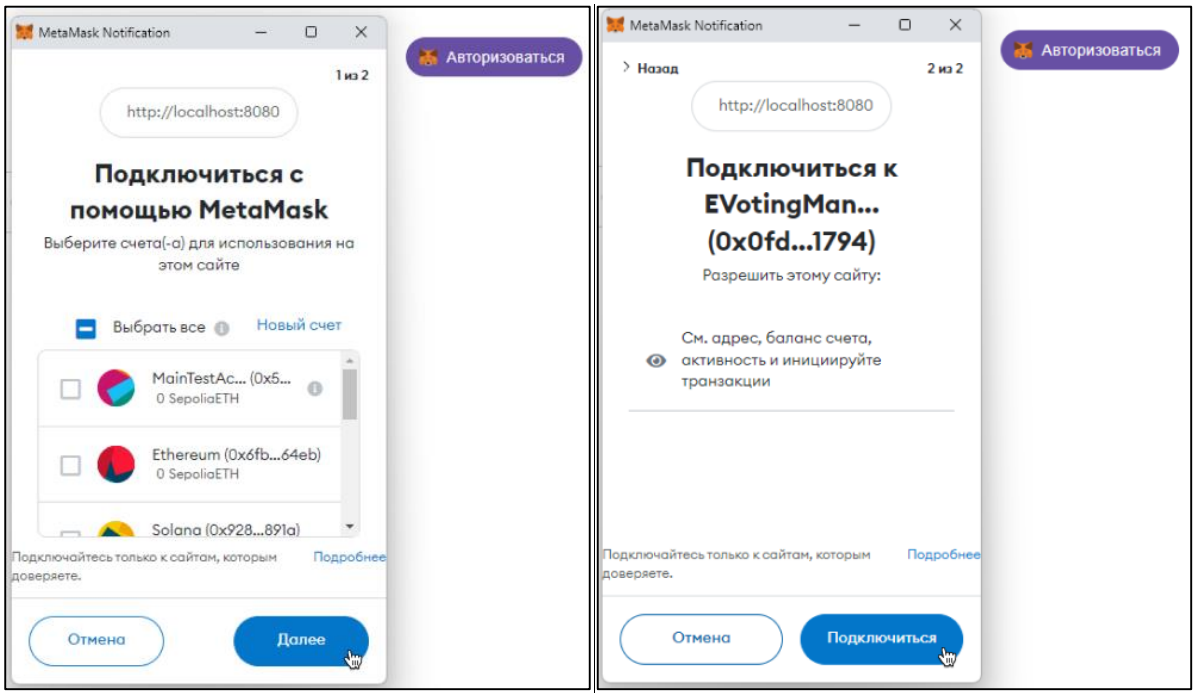

Рисунок 15 – Процесс подключения веб3 провайдера MetaMask

3.4.3. Реализация компонента создания голосования

После подключения веб3-провайдера MetaMask, пользователю становится доступным страница для создания голосования.

Форма на странице «Создание голосования» включает поля ввода для названия голосования, даты и времени начала и окончания голосования, списка участников и списка вариантов ответов. На странице используются компоненты Material-UI для стилей и полей ввода.

Если пользователь заполнил не все поля формы, кнопка для создания голосования будет для него недоступна.

Когда пользователь нажимает кнопку «Создать голосование», компонент отправляет запрос на создание нового голосования с использованием предоставленной информации. Компонент также включает в себя индикатор загрузки, который отображается во время обработки запроса. Скриншот страницы сайта «Создание голосования» представлено на рисунке 16.

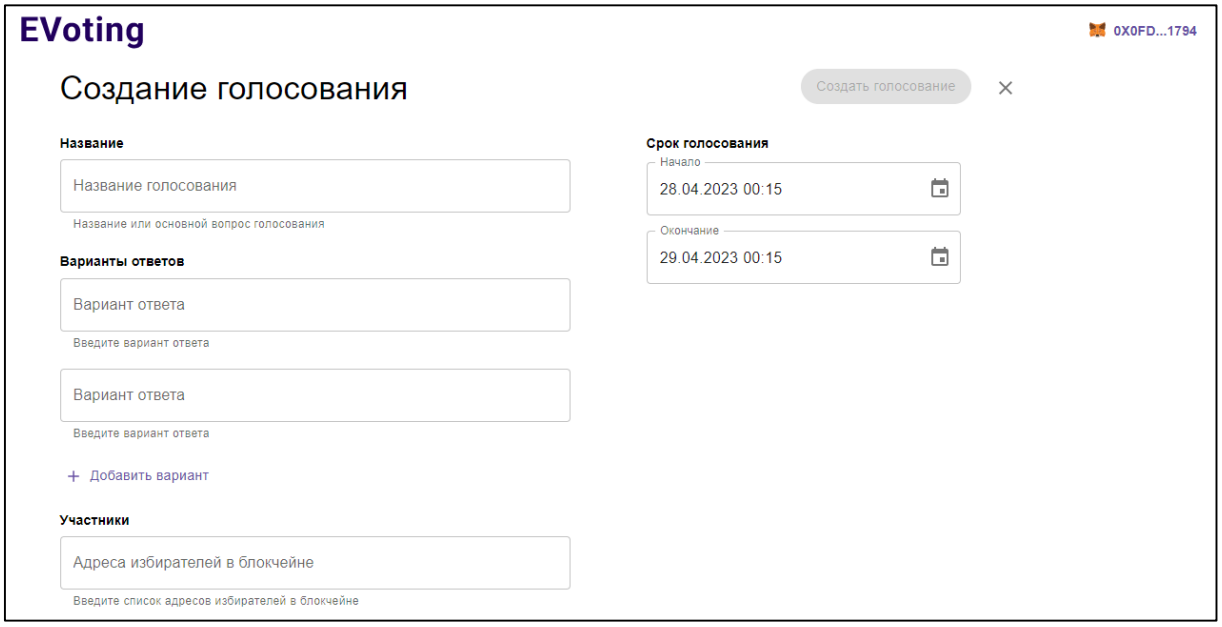

Рисунок 16 – Скриншот страницы сайта «Создание голосования»

3.4.4. Реализация компонента проведения голосования

Компонент проведения голосования отображает данные о голосовании: его название, сроки и статус, варианты ответа и позволяет пользователю голосовать. В данном компоненте также отслеживается, проголосовал ли пользователь или нет, и отключает возможность нажать на кнопку голосования, если пользователь уже проголосовал или если голосование в данный момент не началось.

Когда пользователь выбирает вариант ответа и нажимает кнопку голосования, создается запрос перевода голоса от избирателя на адрес кандидата. После подтверждения о переводе голоса блокчейном на сайте отобра-

жается ссылка на транзакцию на Etherscan. Реализация метода `vote` представлена в листинге 7 приложения А. Скриншот страницы голосования изображен на рисунке 17.

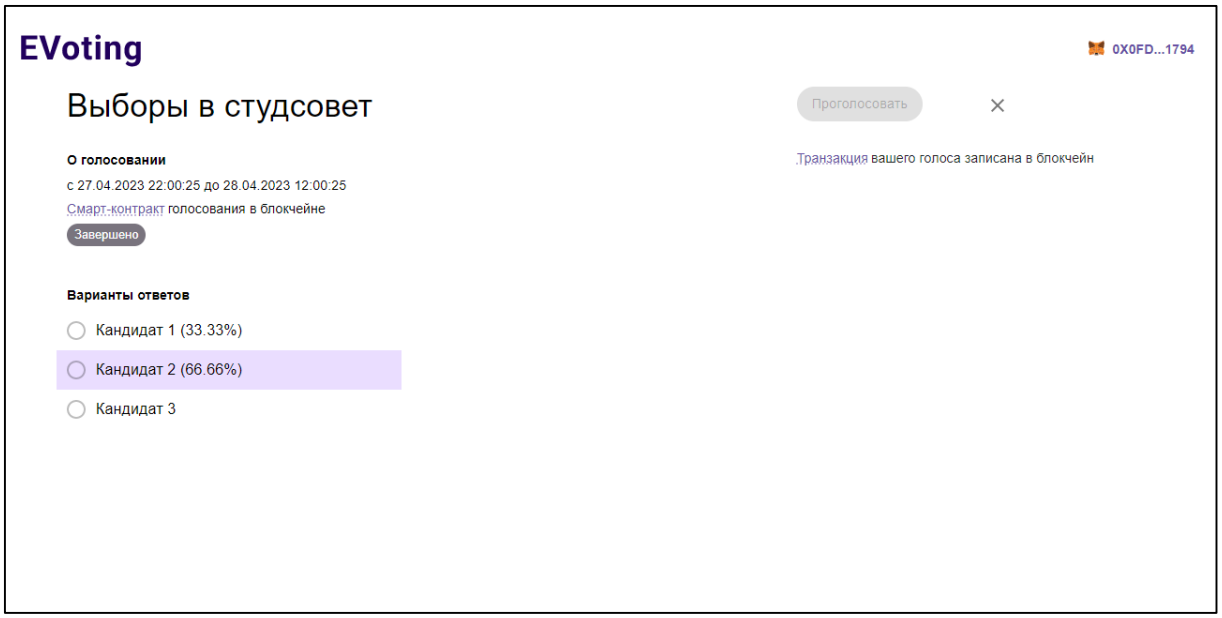

Рисунок 17 – Скриншот страницы голосования

Выводы по третьей главе

В третьей главе были рассмотрены программные средства реализации системы EVoting, а также были реализованы компоненты смарт-контрактов, веб-сервера и веб-интерфейса системы.

4. ТЕСТИРОВАНИЕ

4.1. Функциональное тестирование смарт-контрактов

Каждый смарт-контракт проверялся при помощи набора функциональных тестов. Результаты тестирования отображены в таблицах 1-3 приложения Б. Таким образом, все тесты были пройдены успешно.

4.2. Функциональное тестирование веб-приложения

Для проверки работоспособности описанных функциональных требований проводилось функциональное тестирование. В таблице 4 приложения Б представлены наборы тестов и результаты данного вида тестирования. Все полученные результаты соответствуют ожидаемым.

Выводы по четвертой главе

В данной главе было проведено тестирование смарт-контрактов и функциональное тестирование системы EVoting. Также были описаны результаты тестирования приложения.

ЗАКЛЮЧЕНИЕ

В данной работе была разработана система для электронного голосования на основе технологии блокчейн. В ходе работы были решены следующие задачи.

1. Выполнен обзор литературы и существующих аналогов.
2. Спроектированы смарт-контракты для электронного голосования на основе технологии блокчейн.
3. Спроектировано веб-приложение для электронного голосования на основе технологии блокчейн.
4. Реализованы смарт-контракты и веб-приложение.
5. Проведено тестирование работы приложения.

Исходный код разработанной системы для электронного голосования на основе технологии блокчейн доступен по URL-адресу: https://github.com/wbogatyrewa/EVoting. Разработанное приложение размещено на хостинге и доступно по URL-адресу: http://veronijo.beget.tech/.

ЛИТЕРАТУРА

1. Прасти Н. Блокчейн. Разработка приложений. // СПб.: БВХ-Петербург, 2018. – 256 с.

2. Фролов А.В. Создание смарт-контрактов Solidity для блокчейна Ethereum. Практическое руководство. // ЛитРес: Самиздат, 2019. – 240 с.

3. Digital Voting with use of Blockchain Technology. [Электронный ресурс] URL: https://www.economist.com/sites/default/files/plymouth.pdf (дата обращения: 11.02.2023 г.).

4. Ben Ayed A. A conceptual Secure Blockchain – Based Electronic Voting System. // International Journal of Network Security and Its Application (IJNSA), 2017. Vol.9, no. 3. – 93–101 pp.

5. Boucher P. What if blockchain technology revolutionized voting? // European Union, 2016. – 18–19 pp.

6. Главная | ethereum.org. [Электронный ресурс] URL: https://ethereum.org/ru/ (дата обращения: 11.02.2023 г.).

7. Трубочкина Н. К., Поляков С. В. Система электронного голосования на основе технологии блокчейн с использованием смарт-контракта. // Информационные технологии. 2019. – Т. 25. – № 2. – С. 75–85.

8. Введение в децентрализованное приложение. [Электронный ресурс] URL: https://ethereum.org/ru/developers/docs/dapps/#definition-of-a-dapp (дата обращения: 11.02.2023 г.).

9. IPFS. [Электронный ресурс] URL: https://github.com/ipfs/ipfs (дата обращения: 11.02.2023 г.).

10. Что такое децентрализованные приложения. [Электронный ресурс] URL: https://academy.binance.com/ru/articles/what-are-decentralized-applications-dapps (дата обращения: 11.02.2023 г).

11. Solidity by Example: Voting. [Электронный ресурс] URL: https://docs.soliditylang.org/en/v0.8.18/solidity-by-example.html (дата обращения: 11.02.2023 г.).

12. Abuidris Y., Kumar R., Yang T., Onginjo J. Secure large-scale E-voting system based on blockchain contract using a hybrid consensus model combined with sharding. // ETRI Journal, 2020. – 357-370 pp.

13. 14-ФЗ «Об обществах с ограниченной ответственностью». [Электронный ресурс] URL: http://pravo.gov.ru/proxy/ips/?docbody=&nd=102051516&intelsearch=08.02.1998+14 (дата обращения: 11.02.2023 г.).

14. WE.Vote – общие собрания в ООО. [Электронный ресурс] URL: https://we.vote/kogda-primenyaetsya-blockchain-golosovanie/golosovanie-v-ooo (дата обращения: 11.02.2023 г.).

15. Polys Whitepaper. [Электронный ресурс] URL: https://polysdocs.website.yandexcloud.net/Whitepaper/7262_WP_Polys_Ru_W/EB_4.pdf (дата обращения: 11.02.2023 г.).

16. Polys. [Электронный ресурс] URL: https://org.polys.me/ (дата обращения: 11.02.2023 г.).

17. Официальный сайт Мэра Москвы. [Электронный ресурс] URL: https://www.mos.ru/ (дата обращения: 11.02.2023 г.).

18. Выборы-2022. [Электронный ресурс] URL: https://www.mos.ru/city/projects/vote2022/ (дата обращения: 11.02.2023 г.).

19. Кибервыборы v1.0: как создавалась система блокчейн-голосования в Москве. [Электронный ресурс] URL: https://habr.com/ru/article/480152/ (дата обращения: 11.02.2023 г).

20. WE.Vote – Дистанционное электронное голосование на блок-чейне. [Электронный ресурс] URL: https://we.vote/ (дата обращения: 11.02.2023 г.).

21. Waves Enterprise. [Электронный ресурс] URL: https://wavesenterprise.com/ (дата обращения: 11.02.2023 г.).

22. Angular. [Электронный ресурс] URL: https://angular.io/ (дата обращения: 11.02.2023 г.).

23. Google. [Электронный ресурс] URL: http://google.com/ (дата обращения: 11.02.2023 г.).

24. HTML. [Электронный ресурс] URL: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics (дата обращения: 11.02.2023 г.).

25. Vue.js. [Электронный ресурс] URL: https://vuejs.org/ (дата обращения: 11.02.2023 г.).

26. CSS. [Электронный ресурс] URL: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/CSS_basic (дата обращения: 11.02.2023 г.).

27. React.js. [Электронный ресурс] URL: https://reactjs.org/ (дата обращения: 11.02.2023 г.).

28. Facebook. [Электронный ресурс] URL: https://ru-ru.face-book.com/ (дата обращения: 11.02.2023 г.).

29. JavaScript [Электронный ресурс] URL: https://learn.javascript.ru/ (дата обращения: 11.02.2023 г.).

30. Etherscan. [Электронный ресурс] URL: https://etherscan.io/ (дата обращения: 11.02.2023 г.).

31. MetaMask. [Электронный ресурс] URL: https://metamask.io/ (дата обращения: 11.02.2023 г.).

32. Дизайн веб-приложения. [Электронный ресурс] URL: https://www.figma.com/file/VojP3gQ6OhDoXFEXe2RKBe/EVoting?node-id=53097%3A27272&t=9FGhzLfubAYA3St5-1 (дата обращения: 11.02.2023 г).

33. Figma.com. [Электронный ресурс] URL: https://www.figma.com/ (дата обращения: 11.02.2023 г).

34. Material Design. [Электронный ресурс] URL: https://m3.material.io/ (дата обращения: 11.02.2023 г).

35. Язык программирования Solidity. [Электронный ресурс] URL: https://docs.soliditylang.org/en/v0.8.19/ (дата обращения: 11.02.2023 г).

36. Библиотека OpenZeppelin. [Электронный ресурс] URL: https://docs.openzeppelin.com/contracts/4.x/ (дата обращения: 11.02.2023 г.).

37. Стандарт ERC20. [Электронный ресурс] URL: https://docs.openzeppelin.com/contracts/2.x/erc20 (дата обращения: 11.02.2023 г.).

38. Remix IDE. [Электронный ресурс] URL: https://remix.ethereum.org/ (дата обращения: 11.02.2023 г.).

39. Express.js. [Электронный ресурс] URL: https://expressjs.com/ (дата обращения: 11.02.2023 г.).

40. Node.js. [Электронный ресурс] URL: https://nodejs.org/en (дата обращения: 11.02.2023 г.).

41. Документация Ethers.js. [Электронный ресурс] URL: https://docs.ethers.org/v5/ (дата обращения: 11.02.2023 г.).

42. Alchemy API [Электронный ресурс] URL: https://www.alchemy.com/ (дата обращения: 11.02.2023 г.).

43. Redux. [Электронный ресурс] URL: https://redux.js.org/ (дата обращения: 11.02.2023 г.).

44. Redux Toolkit. [Электронный ресурс] URL: https://redux-toolkit.js.org/ (дата обращения: 11.02.2023 г.).

45. MUI. [Электронный ресурс] URL: https://mui.com/ дата обращения: 11.02.2023 г.).

46. Документация Emotion.js. [Электронный ресурс] URL: https://emotion.sh/docs/introduction (дата обращения: 11.02.2023 г.).

47. Документация web3.js. [Электронный ресурс] URL: https://web3js.readthedocs.io/en/v1.8.2/ (дата обращения: 11.02.2023 г.).

48. Visual Studio Code. [Электронный ресурс] URL: https://code.visualstudio.com/ (дата обращения: 11.02.2023 г.).

ПРИЛОЖЕНИЯ

Приложение А. Листинги исходного кода

Листинг 1 – Смарт-контракт EVotingManager

```
contract EVotingManager {
    address public owner;
    address[] eVotings;

    constructor() {
        owner = msg.sender;
    }

    function getEVotings() view public returns (address[] memory) {
        return eVotings;
    }

    function addEVoting(address _eVoting) public {
        require(
            msg.sender == owner,
            "Only owner can add voting."
        );
        eVotings.push(_eVoting);
    }
}
```

Листинг 2 – Смарт-контракт EVotingToken

```
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract EVotingToken is ERC20, Ownable {
    constructor() ERC20("EVotingToken", "EVT") {}

    function mint(address to, uint256 amount) public onlyOwner {
        _mint(to, amount);
    }

    function transfer(address from, address to, uint256 amount) public {
        _transfer(from, to, amount);
    }
}
```

Листинг 3 – Метод получения списка голосований getVotingList

```
export const getVotingList = async () => {
  const EVotingManager = await new Contract(
    EVotingManagerAddress,
    EVotingManagerABI,
    signer
  );

  const votingAddresses = await EVotingManager.getEVotings();

  const votingList = votingAddresses.map(async (address) => {
    let contract = new ethers.Contract(address, ABI, signer);

    let name = "";
```

Окончание листинга 3 приложения А

```
    let startDateTime = new Date();
    let endDateTime = new Date();
    let voters = [];
    let proposalsAddr = [];
    let proposalsNames = [];

    name = await contract.getName();
    startDateTime = moment.unix(Number(await contract.getStartTime())).to-
Date();
    endDateTime = moment.unix(Number(await contract.getEndTime())).to-
Date();
    voters = await contract.getVoters();
    proposalsAddr = await contract.getPropposalsAddresses();
    proposalsNames = await contract.getPropposalsNames();

    let answers = [];
    for (let i = 0; i < proposalsAddr.length; i++) {
      answers.push({
        label: proposalsNames[i],
        result:
          moment() > endDateTime
            ? Number(await contract.totalVotesFor(proposalsAddr[i])) /
              Math.pow(10, 21)
            : null,
        address: proposalsAddr[i],
      });
    }
    let voting = {
      name: name,
      startDateTime: startDateTime,
      endDateTime: endDateTime,
      address: address,
      voters: voters,
      answers: answers,
    };
    return voting;
  });

  return Promise.all(await votingList);
};
```

Листинг 4 – Метод создания голосования createVoting

```
export const createVoting = async (
  name,
  startDateTime,
  endDateTime,
  voters,
  proposalsNames
) => {
  const proposalsAddress = proposalsNames.map(
    (name) => Wallet.createRandom().address
  );

  const encodedParameters = new AbiCoder().encode(
    ["string", "uint256", "uint256", "address[]", "address[]", "string[]"],
    [name, startDateTime, endDateTime, voters, proposalsAddress, proposals-
Names]
  );
```

```
  const bytecodeWithEncoded = `${bytecode}${encodedParameters.slice(2)}`;

  const EVoting = await new ContractFactory(ABI, bytecodeWithEncoded,
signer);

  const eVoting = await EVoting.deploy(
    name,
    startDateTime,
    endDateTime,
    voters,
    proposalsAddress,
    proposalsNames
  );

  const address = eVoting.getAddress();

  const EVotingManager = await new Contract(
    EVotingManagerAddress,
    EVotingManagerABI,
    signer
  );

  const tx = await EVotingManager.addEVoting(address);

  return address;
};
```

Листинг 5 – Метод фильтрации голосований по названию и статусу

```
const filteredVotingList = useMemo(() =>
    name.length === 0 && (status.length === 0 || status === "Все") ? vot-
ingList :
    name.length === 0 && (status.length !== 0 && status !== "Все") ?
    votingList.filter(voting => {
      let now = new Date().getTime();
      let votingStatus = now >= new Date(voting.startDateTime).getTime() ?
now <= new Date(voting.endDateTime).getTime() ?
        Status.Active : Status.Finished : Status.Before;
      return votingStatus === status;
    }) :
    name.length !== 0 && (status.length === 0 || status === "Все") ?
    votingList.filter(voting => voting.name.includes(name)) :
    name.length !== 0 && (status.length !== 0 && status !== "Все") ?
    votingList.filter(voting => {
      let now = new Date().getTime();
      let votingStatus = now >= new Date(voting.startDateTime).getTime() ?
now <= new Date(voting.endDateTime).getTime() ?
        Status.Active : Status.Finished : Status.Before;
      return voting.name.includes(name) && votingStatus === status;
    }) : votingList,[name, status, votingList]);
```

Листинг 6 – Метод подключения веб3-провайдера MetaMask

```
export const login = async () => {
  let account = "";
  if ((window as any).ethereum) {
    await (window as any).ethereum
    .request({method: 'eth_requestAccounts'})
```

```
    .then((res: string[]) => {
      account = res[0];
    });
  } else {
    alert("Please, install MetaMask");
  }
  return account;
}
```

Листинг 7 – Метод голосования vote

```
export const vote = async (votingAddress: string, answerAddress: string)
=> {
  let res;
  if ((window as any).ethereum) {
    var web3 = new Web3(Web3.givenProvider);
    var contract = await new web3.eth.Contract(ABI as AbiItem[],
votingAddress);
    const fromAddress = (await getAccounts());

    try {
      await contract.methods.vote(answerAddress).send({from: fromAddress},
function(error : any, transactionHash : any) {
        if (error != null) {
          res = error;
        } else {
          res = `https://sepolia.etherscan.io/tx/${transactionHash}`;
        }
      });
    } catch (error) {
      res = error;
    }
  }
  return res;
};
```

Приложение Б. Тестирование приложения

Таблица 1 – Тестирование смарт-контракта EVotingManager

№	Тест	Ожидаемый результат	Тест пройден?
1	Проверка получения списка адресов всех голосований	Массив адресов всех голосований возвращается корректно	Да
2	Проверка добавления адреса голосования	Новый адрес голосования добавляется в массив голосований корректно	Да

Таблица 2 – Тестирование смарт-контракта EVoting

№	Тест	Ожидаемый результат	Тест пройден?
1	Проверка создания голосования	Голосование создается в блокчейне	Да
2	Проверка добавления избирателей	Новые избиратели добавляются в смарт-контракт, токены зачисляются на адреса избирателей корректно	Да
3	Проверка добавления вариантов ответов	Новые варианты ответов добавляются корректно	Да
4	Проверка возможности проголосовать	Токен избирателя переводится на адрес выбранного варианта ответа, транзакция записывается в блокчейн	Да
5	Проверка ограничений на возможность проголосовать	Попытка проголосовать без токенов, за несуществующий вариант ответа приводит к ошибке	Да
6	Проверка получения информации о голосовании	Информация о голосовании: название, сроки, адреса избирателей, адреса вариантов ответов и результаты возвращается корректно	Да

Таблица 3 – Тестирование смарт-контракта EVotingToken

№	Тест	Ожидаемый результат	Тест пройден?
1	Проверка создания токена	Токены создаются корректно	Да
2	Проверка передачи токена	Токены переводятся с одного адреса на другой корректно	Да

Таблица 4 – Функциональное тестирование приложения

№	Тест	Ожидаемый результат	Тест пройден?
1	Проверка авторизации гостя с помощью MetaMask	Гость авторизуется с помощью MetaMask	Да

Окончание таблицы 4 приложения Б

2	Проверка просмотра списка всех голосований	Список всех голосований отображается корректно	Да
3	Проверка просмотра подробных данных о голосовании	Подробные данные о голосовании: название, сроки, адреса избирателей, адреса вариантов ответов и результаты отображаются корректно	Да
4	Проверка поиска голосования по названию или основному вопросу	Найденное по названию или основному вопросу голосование отображается корректно	Да
5	Проверка возможности отфильтровать голосования по их статусу	Голосования фильтруются по статусу корректно	Да
6	Проверка создания голосования	Голосование создается в блокчейне	Да
7	Проверка возможности проголосовать и просмотреть транзакцию «голоса» в блокчейн-обозревателе Etherscan	Авторизованный пользователь голосует и может просмотреть транзакцию «голоса» в блокчейн-обозревателе Etherscan	Да