МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
вычислительной техники	МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

Интернет-голосование с использованием Смарт-контрактов

Студент _____ Лавов Марк Алексеевич
(подпись, дата) (ФИО полностью)

Руководитель _____ Михалев А.Г.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____№_____)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____№____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____А.Ю. Козлов
«20» декабря 2024 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

БАКАЛАВРА

1. Студент____Лавов Марк Алексеевич_____гр._21ВП1_факультета___ВТ_____
<Фамилия, имя, отчество полностью>
2.Тема работы Интернет-голосование с использованием смарт-контрактов_____
_____
Тема утверждена приказом ПГУ №_____от "_____"_____20_____

2. Руководитель работы _Михалалёв А.Г._____

3. Задание на работу (назначение разработки, исходные данные и т.п.)_
1) Разработать информационную систему, обеспечивающее прозрачность Интернет голосования в сети блокчейн
2) Среда выполнения приложения: браузер
Среда выполнения сервера: операционная система Linux Ubuntu
Языковые средства разработки: Python, UML, SQL, Solidity
Инструментальные средства разработки: SQLite,
Microsoft Project, Visual Paradigm, Visual Studio Code
Технологии разработки: Hardhat, ООП, React, Web3
3) Основные функции информационной системы:
а) создание опросов;
б) голосование в опросах;
в) валидация голосования;
г) вознаграждение участников голосования.

4. Перечень подлежащих разработке вопросов_
1) Анализ предметной области и постановка задачи
2) Проектирование

3) Реализация

4) Контроль качества программного обеспечения

5) Оформление пояснительной записки

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2025 – 14.04.2025	
2	Проектирование программных средств	20%	15.04.2025 – 25.04.2025	
3	Реализация программных средств	30%	26.04.2025 – 12.05.2025	
4	Тестирование программных средств	20%	13.05.2025 – 22.05.2025	
5	Оформление пояснительной записки	20%	23.05.2025 – 03.06.2025	

Дата выдачи задания «20» декабря 2024г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 146 листов, 14 рисунков, 17 таблиц, 35 использованных источника.

ИНТЕРНЕТ-ГОЛОСОВАНИЕ С ИСПОЛЬЗОВАНИЕМ СМАРТ-КОНТРАКТОВ, LINUX, SQLITE, PYTHON, SOLIDITY, REACT, HARDHAT, UML

Объектом разработки: информационная система для обеспечения процесса прозрачного голосования в сети блокчейн.

Цель работы: разработка информационной системы для обеспечения прозрачного голосования в сети блокчейн, позволяющая создавать опросы, голосовать в опросах, валидировать данные о голосовании по средствам блокчейна.

Разработка серверной части производилась на языке программирования Python. В качестве среды разработки использовалась среда программирования Visual Studio Code. В качестве СУБД была использована СУБД SQLite. Разработка клиентской части производилось на языке программирования Type Script с использованием технологии React. В качестве среды разработки использовалась среда программирования Visual Studio Code. Приложение предназначено для использования в Браузере, а его серверная часть работает под управлением операционной системы Linux Ubuntu.

Содержание

Введение 7

1 Анализ предметной области и постановка задачи на разработку программного обеспечения для проведения опросов и голосований 8

1.1 Анализ предметной области. 8

1.2 Анализ аналогов программных решений для голосования 10

1.3 Анализ рынка программного обеспечения для реализации смарт-контрактов 13

1.4 Анализ функциональных требований 14

1.5 Анализ нефункциональных требований 19

1.6 Выбор программных средств 21

1.7 Постановка задачи 22

2. Проектирование интернет-голосования на основе контрактов 23

2.1 Проектирование архитектуры интернет-голосования 23

2.2 Проектирование пользовательского интерфейса 26

2.3 Проектирование БД 31

2.3.1 Разработка концептуальной модели базы данных 31

2.3.2 Разработка логической модели базы данных 33

2.3.3 Разработка физической модели базы данных 36

3. Реализация интернет-голосования на основе смарт-контракта 38

3.1 Реализация смарт-контрактов голосования 38

3.2 Реализация работы с базой данных 39

3.3 Реализация клиентской части для интернет-голосования 42

3.3.1 Получение доступа к голосованию 45

4 Тестирование 49

4.1 Функциональное ручное тестирование интернет-голосования..... 50

4.2 Метрики кода..... 55

Заключение 61

Список использованных источников..... 63

Приложение А (обязательное) Листинг приложения 66

Введение

В рамках цифровизации общества, которая происходит в наше время, когда скорость передачи и достоверность передаваемой информации становятся определяющими факторами для выбора того или иного программного продукта, резко возрастает потребность в том, чтобы создать прозрачные и надежные инструменты для сбора общественного мнения. Традиционные методы опросов и голосований часто сталкиваются с проблемами вокруг централизации, высокой вероятности непрозрачности и даже уязвимости к манипуляциям, что подрывает доверие к полученным результатам.

Целью данной работы является создание программного обеспечения для проведения опросов и голосований, которое позволит:

- эффективно организовывать процесс сбора мнений;
- обеспечивать прозрачность для пользователя;
- предоставлять инструменты для анализа данных.

Использование смарт-контрактов в системе гарантирует прозрачность и честность проведенного голосования.

Для разработки был выбран следующий технологический стек:

- блокчейн: Ethereum, Solidity, Hardhat, Web3.py;
- бэкенд: Python (FastAPI, SQLAlchemy), SQLite;
- фронтенд: React, TypeScript, Web3.js;
- тестирование: Pytest.

Использование технологии блокчейн и смарт-контрактов на платформе Ethereum должно обеспечить необходимый уровень прозрачности и защищенности системы, а применение современного технологического стека, включающего React, TypeScript, Python FastAPI и SQLAlchemy, позволит создать надежное и эффективное решение для сбора и анализа общественного мнения.

1 Анализ предметной области и постановка задачи на разработку программного обеспечения для проведения опросов и голосований

Данный раздел будет посвящен изучению и систематизации предметной области для системы, которую мы разрабатываем. Отдельно будут рассмотрены функциональные и нефункциональные требования, а также мы проведем сравнение нашего проекта с уже существующими системами аналогичного назначения и выявим их ключевые особенности, сходства и различия.

1.1 Анализ предметной области

В настоящее время цифровые платформы, позволяющие организовывать и проводить опросы и голосования, становятся важным компонентом информационных систем, направленных на сбор и обработку мнений пользователей. Такие системы применяются в самых различных сферах: в образовании, в общественных инициативах и на уровне организаций. Их основное назначение – упростить процедуру выражения мнения, обеспечить прозрачность голосования и автоматизировать обработку результатов. С развитием цифровых технологий растёт потребность в таких решениях, которые бы обеспечивали достоверность результатов, защиту от искажения данных и прозрачность процесса.

В рамках данной работы разрабатывается децентрализованная система, предназначенная для создания опросов и голосования с возможностью поощрения и вознаграждения пользователей. Основу системы составляет технология децентрализованного хранения данных, которая исключает возможность изменения информации. Это обеспечивает высокую степень доверия к результатам голосования. Ключевая особенность платформы заключается в том, что участие пользователей в опросах может быть вознаграждено начислением специальных цифровых единиц, которые хранятся в защищённой электронной среде.

Структура разрабатываемой системы предполагает участие двух основных

категорий пользователей. Каждая из этих ролей обладает собственным набором прав и обязанностей, соответствующих функционалу платформы.

Программный продукт ориентирован на взаимодействие двух категорий пользователей: обычных пользователей (участников системы) и администратора, выполняющего функции поддержки и распределения ресурсов.

К категории пользователей системы относятся все зарегистрированные участники, обладающие возможностью взаимодействия с системой. Пользователи имеют право как создавать новые опросы, так и участвовать в уже существующих. Для запуска опроса необходимо внести оплату в виде внутренней цифровой единицы, используемой в системе и комиссии для фиксации информации в блокчейне. Такой взнос выполняет функцию фильтра. Каждый участник самостоятельно формулирует тему голосования, возможные варианты ответа и временные рамки его проведения.

В процессе голосования пользователи подтверждают свой выбор с помощью электронной подписи на сформированном сообщении. Далее это сообщение передаётся в систему и записывается в защищённой среде хранения – блокчейне. После успешного участия в голосовании пользователю начисляется поощрение согласно внутренним правилам платформы.

Основные функции пользователей:

- создание тематических опросов с произвольным содержанием;
- установление вариантов ответа и продолжительности голосования;
- участие в голосованиях по мере доступности;
- подтверждение выбора с помощью цифровой подписи;
- получение вознаграждений за участие.

Администратор выполняет функции технического регулирования и служит доверенным участником, осуществляющим ключевые операции, связанные с устойчивым функционированием системы. Он не инициирует опросы и не принимает участия в голосовании. Основной задачей администратора является управление распределением цифровых единиц внутри системы, необходимых для создания опросов и осуществления вознаграждений

пользователей. Данные действия производятся через специальные средства внутри системы, не допускающие ручного вмешательства в голосования и их результаты.

Функции администратора реализуются ограниченным числом операций:

– выпуск дополнительных цифровых единиц при необходимости пополнения резерва;

– распределение внутреннего ресурса между пользователями платформы;

– обеспечение технической работоспособности системы;

– взаимодействие с базовой вычислительной инфраструктурой.

Таким образом, в рамках разрабатывемой системы заложено разделение на две основные роли. Первая – это обычные пользователи, формирующие и поддерживающие активность посредством создания и прохождения опросов. Вторая – администратор, фокусирующий своё внимание на обеспечении бесперебойной работы платформы и управлении экономикой внутренней поощрительной системы.

Предлагаемая архитектура позволяет обеспечить стабильность, высокую степень независимости участников и самостоятельность каждого пользователя при выражении и сборе мнений в цифровом формате.

1.2 Анализ аналогов программных решений для голосования

Проведем всесторонний анализ существующих решений в области децентрализованных систем голосования с целью выявления актуальных проблем, ограничения текущих подходов, а также определения потенциальных конкурентных преимуществ разрабатываемого проекта. На сегодняшний день заметен возрастающий интерес к применению блокчейн-технологий в сфере голосования – как в политических, так и в корпоративных системах принятия решений. Современные технологии, основанные на принципах децентрализации и криптографической защищенности, позволяют значительно повысить доверие к результатам голосования, обеспечивая при этом устойчивость к

фальсификациям и вмешательству со стороны третьих лиц [1].

Подходы, реализованные в существующих проектах, демонстрируют возможность обеспечения прозрачности всех этапов процесса голосования: от регистрации избирателей до подсчета голосов, пощрения за активность и хранения результатов в сети блокчейн.

Тем не менее, несмотря на очевидные преимущества, практическое внедрение таких решений сталкивается с рядом проблем, включая вопросы масштабируемости, этичности, защиты приватности участников, сложности идентификации и законодательной несогласованности в различных юрисдикциях.

Были приведены данные о технологиях, которые используются в разработке современных приложений на блокчейне. Эти данные позволят провести сравнительный анализ и определить, какие элементы и методы целесообразно заимствовать или доработать при создании новой, усовершенствованной платформы децентрализованного голосования в блокчейн сети Ethereum [4].

Сравнительный анализ с представленными аналогами наглядно демонстрирует, что разрабатываемая система SmartPoll обладает целым рядом значимых преимуществ, выгодно выделяющих её на фоне существующих решений. Одним из ключевых отличий является внедрение системы вознаграждения участников, стимулирующей активность пользователей и способствующей формированию устойчивого пользовательского сообщества.

Такой подход не только повышает вовлечённость, но и способствует более справедливому и прозрачному участию в голосовании и исключению возможности подделки, так как все данные валидируются блокчейном и подтверждаются цифровой подписью.

Ниже на таблице 1 представлены примеры наиболее заметных и распространённых систем голосования, основанных на технологии блокчейн, с краткой характеристикой их архитектуры, реализованных функций, а также указанием ключевых преимуществ и отличительных особенностей.

Таблица 1 – Обзор аналогов

Проект	**Преимущества аналога**	**Преимущества SmartPoll**
Snapshot	Поддержка DAO, простота интеграции, опросы без газа	Вознаграждение участникам в токенах, встроенная проверка на уникальное участие, поддержка подписи и relayer-механизма
Aragon	Децентрализованное управление, гибкая структура DAO, интерфейс	Более лёгкий и кастомный контракт, возможность интеграции с индивидуальной базой данных и API
Polys	Красивый веб-интерфейс, ориентирован на реальное образование, точная верификация	Поддержка Web3, криптографическая подпись, децентрализация и защита без регистрации или сбора PII
Simply Voting	Высокая стабильность, применяется университетами и ассоциациями.	Смарт-контракт с открытым кодом, проверяемость транзакций, возможность мотивации пользователей через токены
Google forms	Простота использования, доступность, знаком пользователям	Криптографическая подтверждаемость, исключается возможность фальсификации данных или повторной отправки
ElectionBuddy	Поддержка e-mail/SMS уведомлений, кастомные шаблоны бюллетеней	Публичность результатов в блокчейне, автоматическое вычисление вознаграждений, контроль газа осуществляет сервер.
Decidim	Подходит для гражданского участия, интеграция в порталы	Легковесная и простая архитектура, не требует сложной инфраструктуры, развертывается локально или в децентрализованной сети
BallotReady	Собирает данные по выборам в США, полезен для информирования	Прямое участие в голосовании, не только информирование – взаимодействие с токенами и контрактами
DVote	Подходит для масштабных выборов.	Простота реализации
Open Vote Network	Академическое обоснование, высокая безопасность	Больше ориентирована на пользовательский опыт и гибкость интеграции в реальные web-приложения

В совокупности электронная подпись, возможность получить вознаграждение за активность, геймификация и обеспечение безопасности через валидацию в блокчейн сети делают разрабатываемую платформу SmartPoll конкурентоспособной на рынке децентрализованных систем голосования и

открывают перспективы её масштабного внедрения как в общественно-политических, так и в корпоративных структурах [5].

1.3 Анализ рынка программного обеспечения для реализации смарт-контрактов

Для разработки и тестирования смарт-контрактов на сегодняшний день существует широкое разнообразие технологий, инструментов и фреймворков, каждый из которых обладает своими уникальными особенностями, преимуществами и областями применения. Рост популярности блокчейн-платформ, таких как Ethereum, Polygon, Binance Smart Chain и других, способствовал активному развитию специализированного программного обеспечения, ориентированного на упрощение создания, отладки, тестирования и развертывания смарт-контрактов.

Современные инструменты охватывают все этапы жизненного цикла смарт-контракта – начиная с разработки и локального тестирования, заканчивая безопасным и надежным деплоем в основную сеть. При этом важную роль играют средства для автоматического и интеграционного тестирования, статического анализа кода, эмуляции блокчейна, а также обеспечения безопасности и соответствия контрактов общепринятым стандартам (например, ERC-20, ERC-721 и др.). Эти инструменты позволяют не только выявлять ошибки на ранних этапах, но и оптимизировать производительность [7]. Важно отметить, что наличие качественных инструментов также способствует повышению доверия пользователей и блокчейн-приложениям в целом.

В таблице 2 представлен обзор актуального программного обеспечения и инструментов, используемых для разработки и тестирования смарт-контрактов. Для каждого из решений указаны его основные характеристики, преимущества, поддерживаемые блокчейн-платформы и особенности, делающие его подходящим для использования в рамках создания децентрализованных приложений. Например Solidity – это объектно-ориентированный язык программирования смарт-контрактов блокчейна Ethereum.

Таблица 2 – Обзор рынка программного обеспечения

Название	**Язык программирования**	**Особенности и преимущества**
Ethereum + Solidity	Solidity	Самая популярная экосистема, развитая инфраструктура
Solana + Rust	Rust	Высокая скорость и пропускная способность, низкие комиссии, сложнее в разработке
Binance Smart Chain (BSC)	Solidity	Совместим с Ethereum, дешёвые транзакции, популярен для DeFi
Avalanche	Solidity	Поддержка собственного виртуального машины
Algorand + PyTeal	Python	Простота интеграции, быстрая сеть, возможность писать логику почти как в обычном Python
Cardano + Plutus	Haskell	Академически обоснованная модель, высокая формальная верификация, узкая аудитория

Наиболее широко используемой технологией для разработки смарт контрактов остается Solidity в сети (блокчейне) Ethereum, что обусловлено надежностью экосистемы и обширной поддержкой сообщества [2].

1.4 Анализ функциональных требований

В ходе анализа задачи по разработке системы для проведения онлайн-опросов и голосований были сформулированы ключевые функциональные требования к системе.

Базовый функционал пользователя:

- регистрация и авторизация участников через кошелек Metamask;
- просмотр доступных опросов и голосований;
- участие в открытых опросах;
- просмотр результатов завершенных голосований;
- создание и настройка новых опросов.

Функционал администратора:

– создание и настройка новых опросов;
– управление списком участников;
– конфигурация параметров голосования;
– архивация завершенных голосований.

Функционал работы с опросами:

– создание различных типов вопросов;
– настройка логики опроса;
– установка временных рамок проведения.

Аналитика и отчетность:

– мониторинг активности участников в реальном времени;
– визуализация промежуточных и финальных результатов;
– формирование детализированных отчетов;
– сравнительный анализ результатов.

Для реализации указанных требований была разработана UML-диаграмма вариантов использования, отображающая взаимодействие с системой основных акторов в соответствии с их ролями (администратор, участник, модератор) в системе. Диаграмма демонстрирует полный цикл работы с опросами – от создания до анализа результатов, включая все ключевые сценарии взаимодействия пользователей с системой.

Разрабатываемая система SmartPoll призвана обеспечить комплексное решение для организации прозрачных и достоверных онлайн-опросов различной сложности, удовлетворяя потребности как организаторов, так и участников голосований и обеспечивающая прозрачность и честность опросов, так как любые записи валидируются и остаются в сети блокчейн без возможности подделки.

На рисунке 1 изображена UML диаграмма вариантов использования разрабатываемой системы для проведения опросов и голосований.

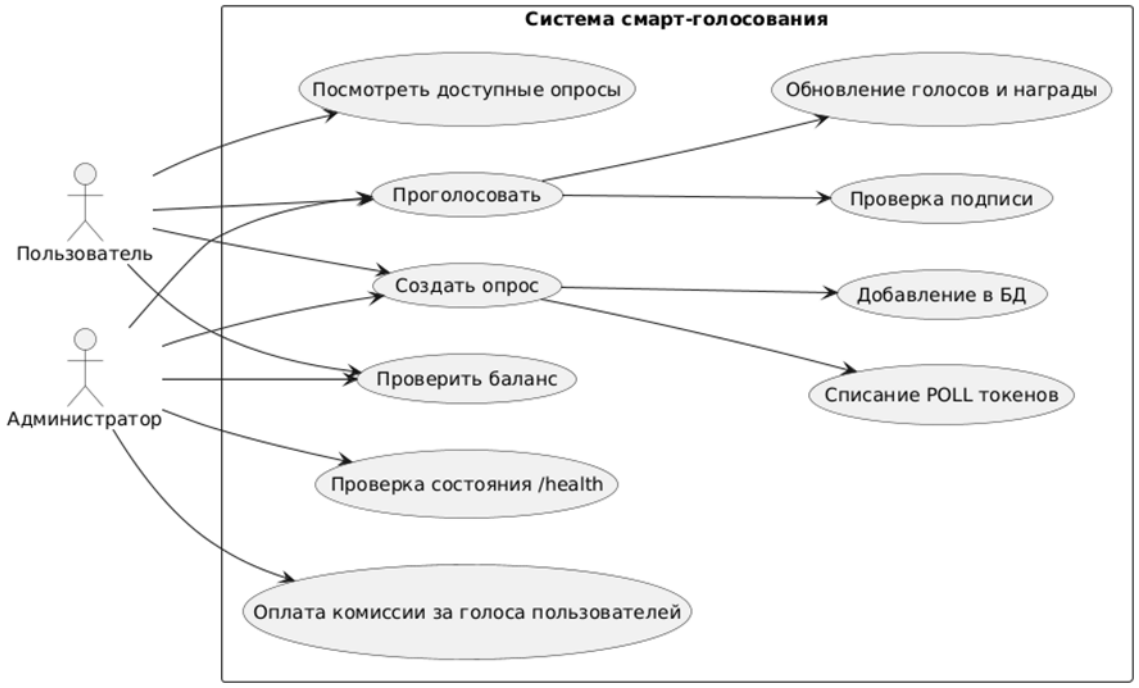

Рисунок 1 – Диаграмма вариантов использования

Далее приведены спецификации некоторых из вариантов использования.

На рисунке 2 представлено описание сценария «Создать опрос».

Наименование: Создать опрос
ID: 1
Краткое описание: Пользователь отправляет данные опроса, комиссия списывается во внутренней валюте POLL. Дальше идёт запись в блокчейн и БД
Действующие лица: Пользователь, Администратор
Предусловие: Достаточный баланс внутренней валюты (POLL)
Основной поток: 1. Заполняется форма с вопросом, вариантами ответа и временем окончания 2. Проверяется баланс и разрешение на списание 3. Отправляется транзакция создания опроса 4. Данные добавляются в БД.
Постусловие: Опрос успешно создан и доступен другим пользователям

Рисунок 2 – Описание сценария варианта использования «Создать опрос»

Создать опрос может любой пользователь, заплатив 100 денежных единиц во внутренне валюте POLL, заработанных через голосование в других опросах.

На рисунке 3 представлено описание сценария «Проголосовать в опросе».

Наименование: Проголосовать в опросе
ID: 2
Краткое описание: Пользователь подписывает данные голосования. Сервер отправляет транзакцию в блокчейн
Действующие лица: Пользователь, Администратор
Предусловие: Достаточный баланс внутренней валюты (POLL)
Основной поток: 1. Пользователь выбирает опрос и вариант ответа 2. Подписывает данные в кошельке (MetaMask) 3. Отправляет подпись на сервер 4. Сервер валидирует подпись, транслирует транзакцию 5. Начисляется награда, голос сохраняется в БД
Постусловие: Голос учтён в смарт-контракте и внести в базу данных, как подтверждённый

Рисунок 3 – Описание сценария варианта использования

«Проголосовать в опросе»

Голос проходит валидацию через цифровую подпись с помощью крипто-кошелька MetaMask. Пользователь подписывает транзакцию.

На рисунке 4 представлено описание сценария «Проверить баланс».

Наименование: Проверить баланс
ID: 3
Краткое описание: Отображение баланса по адресу пользователя
Действующие лица: Пользователь, Администратор
Предусловие: Пользователь указывает корректный адрес
Основной поток: 1. Отправляется запрос на API 2. Через Web3 вызывается balanceOf 3. Баланс отображается на клиенте
Постусловие: Пользователь получил информацию о текущем количестве внутренней валюты.

Рисунок 4 – Описание сценария варианта использования «Проверить баланс»

Проверка баланса позволяет узнать пользователю количество денежных еденицы POLL на своем аккаунте, заработанных через голосование в других опросах. Проверка происходит через обращение к смарт-контракту голосования. Смарт-контракт присылает количество денежных единиц на кошельке пользователя.

На рисунке 5 представлено описание сценария «Посомтреть доступные опросы».

Наименование: Посмотреть доступные опросы
ID: 4
Краткое описание: Отображаются все опросы
Действующие лица: Пользователь, Администратор
Предусловие: Доступ к сети и API
Основной поток: 1. Отправляется запрос /polls 2. Сервер получает все ID опросов из смарт-контракта 3. Вызывается getPoll для каждого ID 4. Данные передаются клиенту
Постусловие: Пользователь видит список всех опросов.

Рисунок 5 – Описание сценария варианта использования «Посмотреть доступные опросы»

Позволяет пользователям увидеть список всех опросов. Можно проголосовать, выбрав катрточку опроса, выбрать вариант ответа. После этого будет предложено подписать свою транзакцию через крипто кошелек MetaMask. После этого транзакция отправляется в блокчейн Ethereum. Только после этого голос будет засчитан за опросом. Такая система позволяет повысить ответственность и исключает возможность фальсификации и повторного голосования.

На рисунке 6 представлено описание сценария «Оплата комисси за голоса пользователей».

Наименование: Оплата комиссии за голоса пользователей
ID: 5
Краткое описание: Отображаются все опросы
Действующие лица: Администратор
Предусловие: Пользователь отправил валидную подпись
Основной поток: 1. Получение подписи от клиента 2. Валидация цифровой подписи 3. Отправка транзакции voteWithSignature 4. Оплата газ за счёт администратора
Постусловие: Голос успешно отправлен, комиссионные затраты покрыты.

Рисунок 6 – Описание сценария варианта использования «Оплата комиссии за голоса пользователей»

С учетом разделения функционала для разных пользователей было выделено несколько основных действий: работа с опросами и балансом. Для каждого варианта действия прописаны действующие лица, алгоритм и условия успешного выполнения.

1.5 Анализ нефункциональных требований

Нефункциональные требования включают в себя требования: по удобству использования, производительности, безопасности, доступности.

Требования по удобству использования (Usability) приложения для голосования:

– 95% новых пользователей должны успешно создать/пройти опрос без ошибок за ≤ 2 минуты;

– интерфейс для голосования должен быть интуитивно понятным: ≥ 90% пользователей должны освоить основные процессы без инструкций;

– совместимость с мобильными устройствами: все функции в полной мере доступны на экранах ≥ 320px.

Требования к производительности (Performance):

– система обслуживает до 500 одновременных пользователей (пиковая нагрузка – 200 пользователей/мин);

– страницы загружаются ≤ 3 сек при скорости интернета 10 мбит/сек;

– результаты опроса отображаются ≤ 2 сек после завершения голосования.

Требования к безопасности (Security):

– шифрование всех данных с помощью TLS 1.3 для веб-трафика, хеширование паролей (алгоритм Argon2);

– обязательная аутентификация (для голосования – MetaMask или WalletConnect, для админов – 2FA – Google Authenticator);

– разграничение прав – только админы создают и удаляют опросы, пользователи видят только свои голоса.

Требования к защите данных (Safety):

– прозрачность – все опросы хранятся в блокчейне, участники могут проверить свой голос через хеш-идентификатор;

– анонимность (опционально) – режим zk-SNARKs для конфиденциальных голосований.

Требования к доступности (Availability):

– доступность 99% в рабочие часы (08:00–23:00);

– плановое обслуживание ≤ 1 часа в месяц.

Требования к надежности (Reliability):

– при разрыве соединения незавершенные голоса сохраняются локально (30 мин);

– автоматическая повторная отправка при восстановлении сети.

Дополнительные требования СМР-1:

– совместимость – экспорт и удаление персональных данных по запросу;

– масштабируемость – поддержка до 10 тысяч опросов в базе SQLite.

Основываясь на сформулированых нефункциональных требованиях к системе умного голосования необходимо сделать вывод о допустимой архитектуре приложения, средствах разработки и средствах защиты.

1.6 Выбор программных средств

Для реализации системы умного голосования на основе смарт-контрактов были задействованы современные и проверенные инструменты разработки, которые обеспечивают стабильную работу.

В инструменты разработки входят: Solidity – стандарт разработки для Ethereum сети, OpenZeppelin – библиотека с шаблонами контрактов.

В таблице 3 приведен выбор средств разработки.

Таблица 3 – Выбор средств разработки

Компонент системы	**Инструменты, технологии**	**Обоснование выбора**
Смарт контракты	Solidity, OpenZeppelin	Solidity – стандарт разработки для Ethereum сети; OpenZeppelin – безопасность и шаблоны
Локальная сеть блокчейна	Hardhat	Быстрая отладка и отладка транзакций без затрат на газ
API и серверная часть	FastAPI (Python), Web3.py, SQLAlchemy	Удобный и быстрый фреймворк для REST, полная интеграция с Web3 через Python
База данных	SQLite	Простота разработки
Клиентская часть	TypeScript Web3.js	Взаимодействие с кошельком пользователя через браузер
Тестирование	Pytest	Быстрая проверка работоспособности и логики работы REST

В результате анализа и практического применения выбранных технологий можно заключить, что выбранные технологии полностью отвечают требованиям разработки современной системы умного интернет- голосования. Комбинация Solidity и OpenZeppelin для смарт-контрактов, FastAPI для серверной части, и TypeScript с Web3.js для клиентского взаимодействия обеспечивает надежную и безопасную работу системы, а использование Hardhat и SQLite существенно упрощает процесс разработки и тестирования. Выбранные инструменты не только соответствуют современным стандартам разработки, но и гарантируют необходимую масштабируемость и поддерживаемость системы в долгосрочной перспективе [6].

1.7 Постановка задачи

Цель данного проекта – разработать децентрализованную систему интернет-голосования, которая обеспечивает. Мною были сформулированы задачи, которые должны быть решены системной умного интернет-голосования.

Задачи:

– прозрачность и открытость результатов;

– подтверждение авторства голосов через криптографические подписи;

– возможность мотивировать участников с помощью вознаграждения во внутренней валюте;

– исключение повторного голосования;

– независимость от централизованных серверов и баз данных за счёт использования блокчейна.

Результатом проекта является комплексная платформа для проведения интернет-голосований, которая успешно решает ключевые проблемы традиционных систем электронного голосования. Разработанное решение обеспечивает высокий уровень удобства использования благодаря интуитивному интерфейсу и автоматизации процессов, при этом сохраняя полную прозрачность всех операций через публичный блокчейн Ethereum.

Безопасность системы достигается за счет применения криптографических механизмов защиты и смарт-контрактов, которые исключают возможность манипуляций с результатами голосования [8]. Внедренная умная автоматизация позволяет эффективно управлять процессами создания голосований, подсчета результатов и распределения вознаграждений участникам. Практическое применение платформы демонстрирует её способность успешно функционировать в реальных условиях, обеспечивая надежное и децентрализованное решение для проведения различных типов голосований и опросов.

2. Проектирование интернет-голосования на основе контрактов

В данном разделе рассматриваются ключевые аспекты разработки системы интернет-голосования, реализованной на основе блокчейн-технологий. Архитектура решения построена с учетом всех поставленных задач и обеспечивает необходимый уровень прозрачности, безопасности и децентрализации. Особое внимание уделено механизмам криптографической верификации голосов и предотвращения повторного голосования, что достигается за счет использования смарт-контрактов Ethereum. Система спроектирована таким образом, чтобы обеспечить эффективное взаимодействие между компонентами блокчейн-сети, серверной части и пользовательского интерфейса. Реализованная структура базы данных позволяет оптимально хранить и обрабатывать информацию о голосованиях, при этом сохраняя независимость от централизованных серверов благодаря распределенной природе блокчейна. Внедренная система мотивации участников через вознаграждения во внутренней валюте способствует повышению активности пользователей.

2.1 Проектирование архитектуры интернет-голосования

При выборе архитектуры необходимо учитывать особенности работы с

блокчейном.

Существуют современные платформы для проведения онлайн-голосований, которые позволяют проводить честное голосование без возможности махинаций.

PollDapp – это современная платформа для проведения онлайн-голосований, где вся информация надёжно защищена технологией блокчейн, а участники получают вознаграждение за свою активность. Ниже приведены основные компоненты системы.

Умные контракты (смарт-контракты):

– PollToken – отвечает за создание и распределение внутренней валюты платформы (токенов POLL);

– PollSystem – управляет процессом голосования: созданием опросов, подсчётом голосов и начислением вознаграждений.

Серверная часть приложения для голосования принимает и проверяет данные от пользователей, обновляет информацию в локальной базе данных и формулирует статистику. Серверная часть включает в себя работу с блокчейном через ABI – интерфейс определющий работу смарт-контракта.

Серверная часть приложения работет с базой данный SQLite по средствам технологии SQLAlchemy. В базе данных приложения хранится информация об участниках, опросах и результатах голосований, которая потребуется для быстрого доступа к информации.

Клиенская часть приложения отвечает за возможность создавать опросы, голосовать и следить за результатами.

Крипто-кошелек – это программа, через которую пользователь может взаимодействовать с блокчейнами. Кошелёк пользователя нужен только для электронной подписи при голосовании и не требует расходов на комиссии блокчейна. На рисунке 7 представлена диаграмма компонентов.

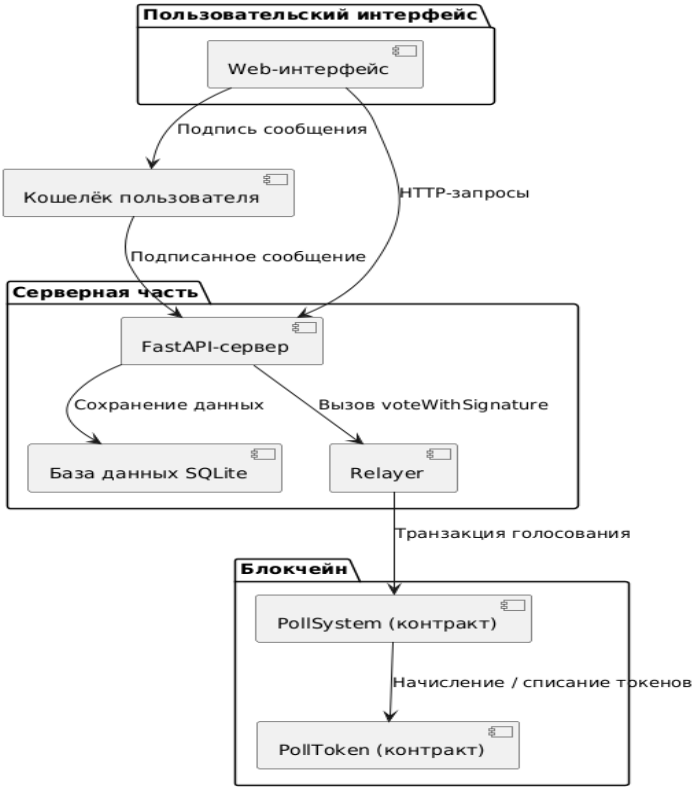

Рисунок 7 – Диаграмма компонентов

Для голосования пользователь заходит на сайт PollChain и выбирает интересующий его опрос. Система формирует специальное сообщение с информацией о выборе пользователя. Пользователь подтверждает свой выбор электронной подписью через кошелёк. Подписанные данные отправляются на сервер для проверки. После проверки информация записывается в блокчейн. Участник автоматически получает вознаграждение во внутренней валюте (токенах POLL). Данные дублируются в локальную базу для быстрого доступа.

Ниже на рисунке 8 представлена диаграмма взаимодействия.

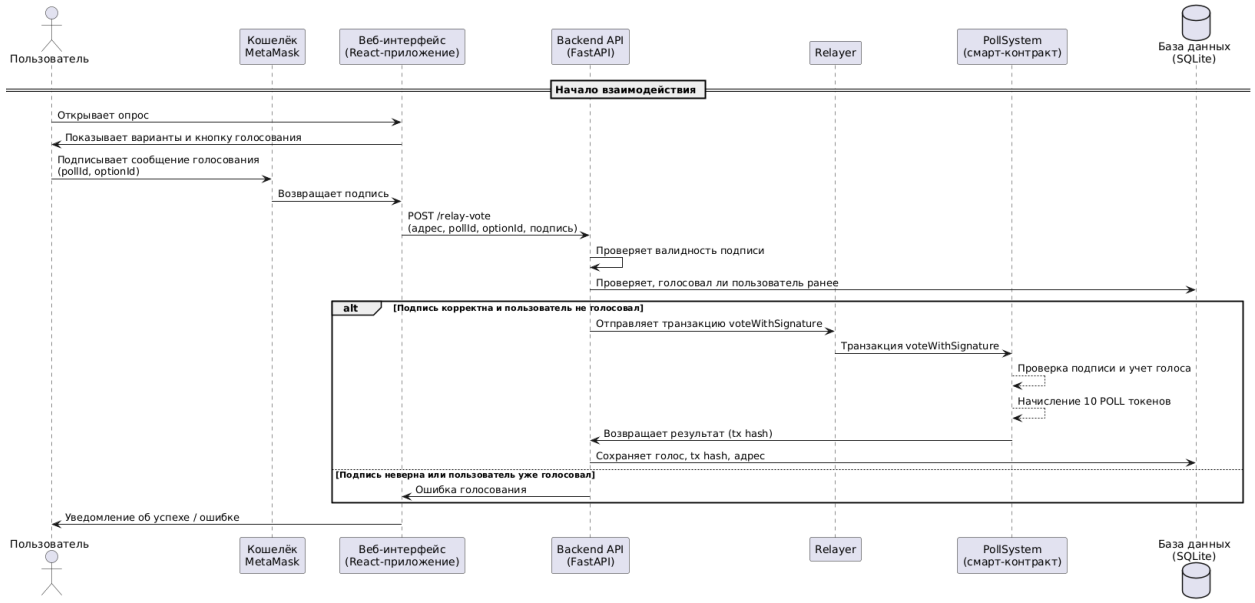

Рисунок 8 – Диаграмма взаимодействий

Пошаговый алгоритм обработки голоса:

– система проверяет подлинность электронной подписи;

– проверяется, не голосовал ли пользователь ранее в этом опросе;

– если всё в порядке, голос записывается в блокчейн;

– на счёт пользователя поступает вознаграждение;

– информация сохраняется в локальной базе данных.

Архитектура проекта была спроектирована, для решения поставленной задачи максимально оптимальным способом.

2.2 Проектирование пользовательского интерфейса

Интерфейс PollDapp создан с расчётом на интуитивную понятность для пользователей с разным уровнем технической подготовки. Ниже представлен список разделов пользовательского интерфейса платформы.

Главная страница представляет собой интуитивно понятный интерфейс, где пользователь может просматривать существующие опросы, осуществлять поиск и отслеживать свой баланс внутренней валюты. Реализованная система навигации позволяет легко перемещаться между различными разделами платформы. На рисунке 9 представлена главная страница.

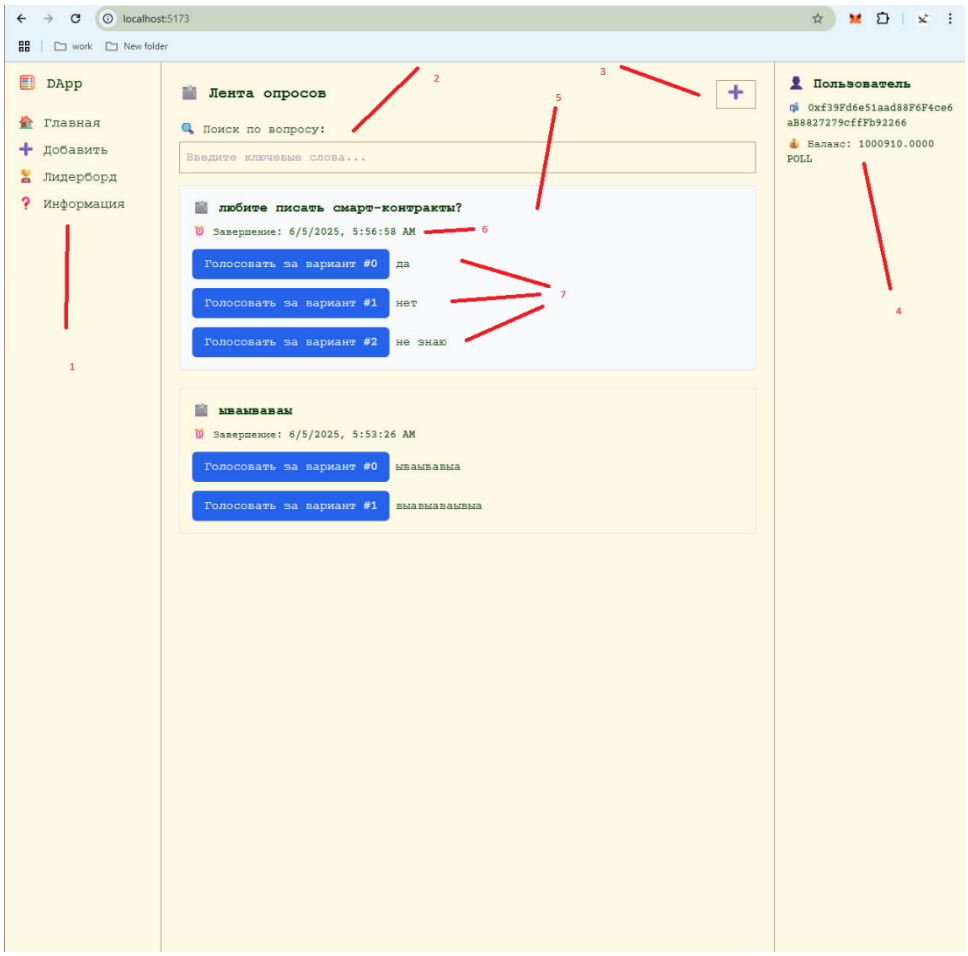

Рисунок 9 – Главная страница

Цифрами обозначены основные компоненты:

1 – панель навигации;

2 – поле поиска опросов;

3 – кнопка создания нового опроса;

4 – данные о кошельке (адресе) пользователя и его баланс;

5 – форма опросов;

6 – временные рамки опроса;

7 – варианты опроса.

Страница создания опроса обеспечивает удобный процесс формирования нового голосования, где пользователь может задать вопрос, добавить варианты ответов и установить временные рамки. Интерфейс включает все необходимые элементы для создания полноценного опроса с использованием смарт-контракта.

На рисунке 10 представлена страница создания опроса.

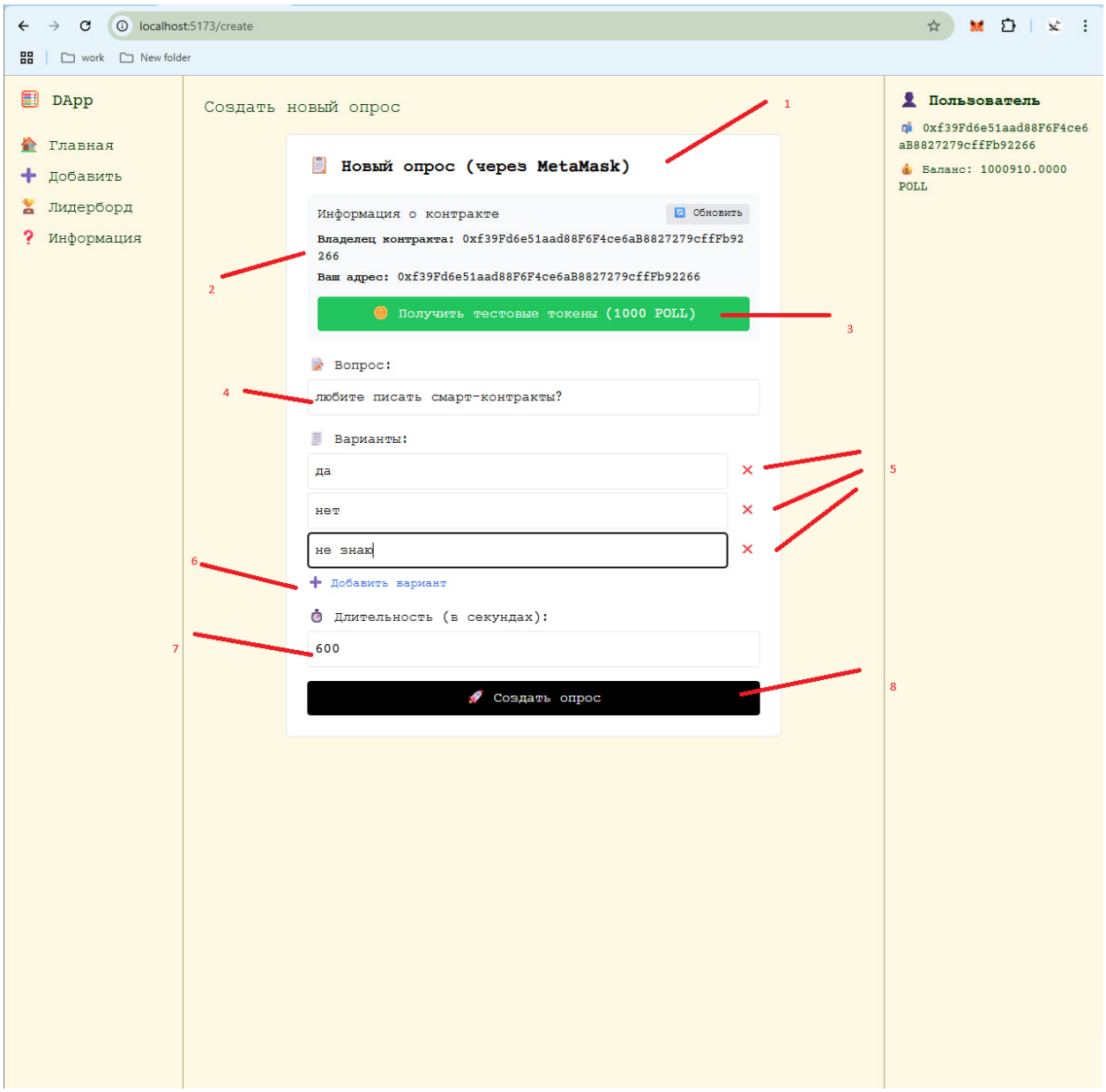

Рисунок 10 – Создание опроса

Цифрами обозначены основные элементы страницы:

1 – форма создания опроса;

2 – информация об адресе контракта;

3 – кнопка для тестового создания новой внутренней валюты;

4 – поле для ввода вопроса опроса;

5 – варианты ответов;

6 – кнопка добавления новых вариантов ответа;

7 – поле для задания длительности опроса;

8 – кнопка публикации опроса.

На рисунке 11 представлена страница таблицы активности пользователей.

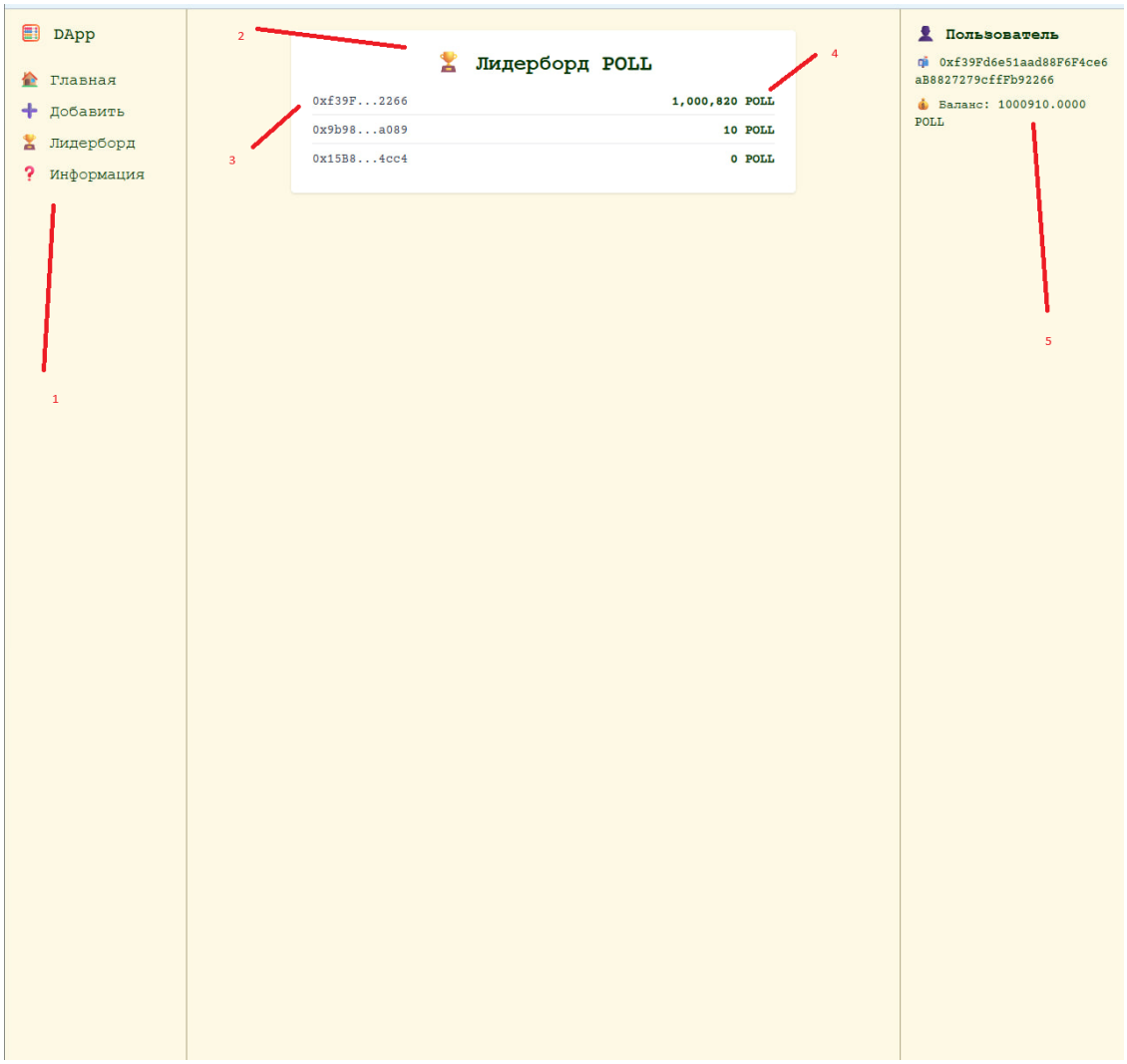

Рисунок 11 – Таблица активности пользователей

Цифрами обозначены основные элементы страницы:

1 – панель навигации;

2 – форма таблицы активности пользователей;

3 – адреса пользователей;

4 – количество внутренней валюты на балансе пользователя;

5 – данные о кошельке (адресе) пользователя и его баланс.

Таблица активности пользователей наглядно отображает информацию о балансах и транзакциях участников системы, что обеспечивает прозрачность всех операций с внутренней валютой POLL и позволяет отслеживать активность

пользователей. На рисунке 12 представлена страница с информацией.

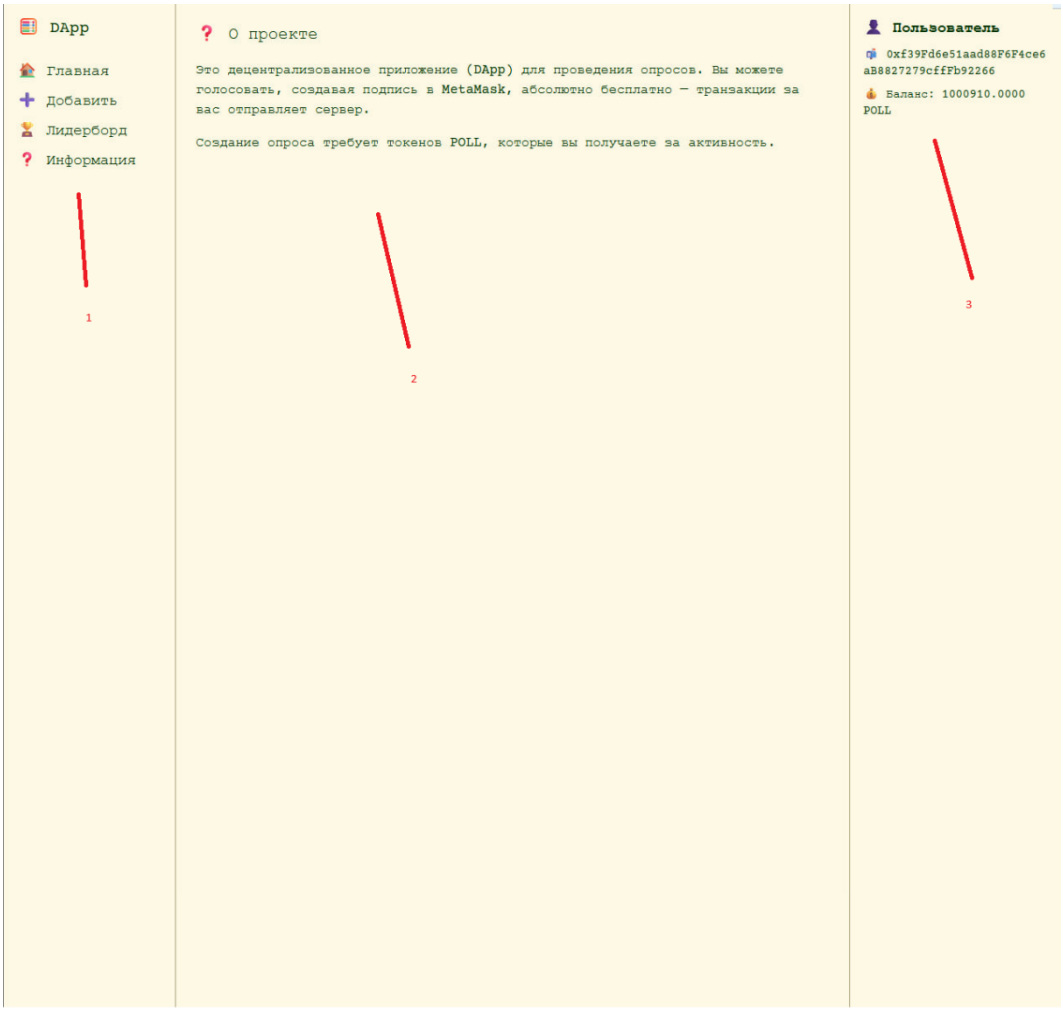

Рисунок 12 – Информация

Цифрами обозначены основные элементы страницы:

1 – панель навигации;

2 – текст с информацией о проекте;

3 – данные о кошельке (адресе) пользователя и его баланс.

Информационная страница предоставляет пользователям необходимые сведения о проекте и его функционале, что способствует лучшему пониманию принципов работы системы и повышает доверие к платформе.

Разработанный пользовательский интерфейс системы умного голосования представляет собой комплексное решение, сочетающее в себе простоту

использования и функциональность. Все компоненты интерфейса логически связаны между собой и обеспечивают полный цикл работы с платформой: от создания опросов до отслеживания результатов и управления внутренней валютой.

2.3 Проектирование БД

Для быстрой работы системы используется локальная база данных SQLite, которая хранит информацию, не требующую записи в блокчейн. Такой подход позволяет оптимизировать производительность системы и снизить затраты на транзакции в сети Ethereum, сохраняя при этом все преимущества децентрализованного хранения критически важных данных [9]. Развертывание базы данных в среде управления SQLite обеспечивает надежное хранение локальных данных, быстрый доступ к информации и возможность легкого масштабирования системы при увеличении нагрузки.

2.3.1 Разработка концептуальной модели базы данных

Концептуальная модель базы данных представляет собой высокоуровневое описание данных и их взаимосвязей, независимое от конкретной СУБД. На этом этапе определены основные сущности, их атрибуты и связи между ними. Основные сущности, выделенные в рамках платформы, включают:

– пользователи (Users) – хранит информацию о пользователях, зарегистрированных в системе.

– опросы (Polls) – хранит информацию об опросах, созданных пользователями.

– голоса (Votes) – Хранит информацию о голосах, оставленных пользователями на опросы.

Описание сущностей предметной области представлено в таблице 4.

Таблица 4 – Табличное описание сущностей для концептуальной модели базы данных

Сущность	**Описание**	**Атрибуты**
Users	Зарегистрированные пользователи системы	address, balance, polls_created, votes_cast
Polls	Опросы, созданные пользователями	id, question, options, end_time, created_at, creator_address
Votes	Ответы, поставленные пользователями на опросы	id, poll_id, user_address, option_id, tx_hash, timestamp

Связи между сущностями определены на основе функциональных требований. Спецификация связей между сущностями представлена в таблице 5.

Таблица 5 – Спецификации связей между сущностями для концептуальной модели базы данных

Связь	**Сущность 1**	**Сущность 2**	Тип связи	Описание связи
Users-Polls	User	Polls	Один ко многим	Один пользователь может создать несколько опросов
User-Votes	User	Polls	Один ко многим	Один пользователь может голосовать несколько раз
Polls-Votes	Polls	Votes	Один ко многим	Один опрос может содержать несколько голосов

Концептуальная модель данных представлена на рисунке 13.

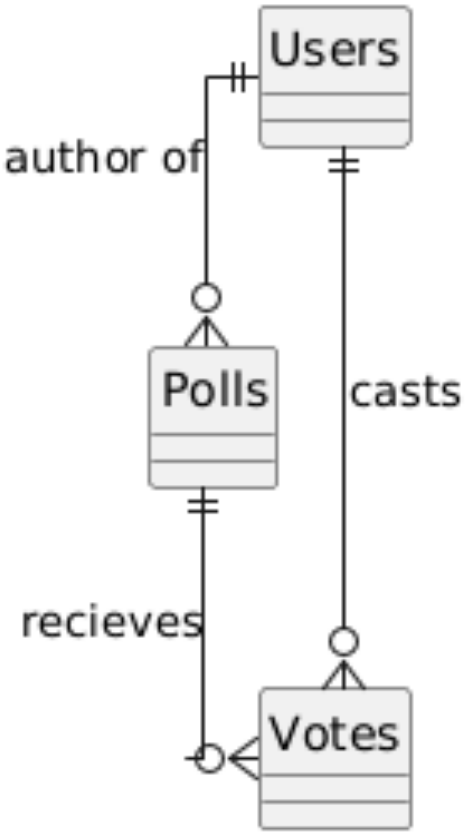

Рисунок 13 – Концептуальная модель данных системы PollSystem

Таким образом, на данном этапе была разработана концептуальная модель базы данных, включающая основные объекты предметной области и связи между ними.

2.3.2 Разработка логической модели базы данных

Логическая модель базы данных детализирует концептуальную модель, определяя таблицы, их атрибуты, типы данных и связи, но оставаясь независимой от конкретной СУБД. На этом этапе данные нормализованы для устранения избыточности и обеспечения целостности.

Таблица users содержит информацию о пользователях системы (в том числе выступающих в роли участников голосования и авторов опросов). Поля таблицы включают: address (строка, длиной до 42 символов) – уникальный идентификатор пользователя, представляет собой адрес кошелька, используется как первичный ключ, balance (целое число) – текущее количество внутренней

валюты на счете пользователя. polls_created (целое число) – количество созданных пользователем опросов, votes_cast (целое число) – количество голосов, которые отдал пользователь.

Таблица polls представляет собой список опросов. Каждая запись содержит следующую информацию: id (целое число, автоинкремент) – уникальный идентификатор опроса, первичный ключ, question (текст) – формулировка вопроса в опросе, options (тип JSON) – список возможных вариантов ответа, end_time (дата и время) – момент завершения голосования, created_at (дата и время) – дата и время создания опроса, creator_address (строка, длиной до 42 символов) – внешний ключ, ссылается на поле address в таблице users, определяет автора опроса.

Таблица votes хранит информацию о голосах, отданных пользователями в рамках различных опросов. Каждое голосование представлено следующими параметрами: id (целое число, автоинкремент) – уникальный идентификатор голоса, первичный ключ, pollid (целое число) – внешний ключ, ссылается на поле id в таблице polls, указывает, к какому опросу относится голос, useraddress (строка, длиной до 42 символов) – внешний ключ, ссылается на поле address в таблице users, указывает, кто проголосовал, optionid (целое число) – индекс выбранного варианта ответа из набора options, txhash (строка, длиной до 66 символов) – хэш транзакции в блокчейне, подтверждающей голос, timestamp (дата и время) – момент отправки голоса, по умолчанию – текущая дата и время. Эта структура позволяет эффективно отслеживать и управлять голосами, обеспечивая целостность данных через внешние ключи и возможность быстрого доступа к информации о том, кто и за что проголосовал. Кроме того, наличие поля tx_hash обеспечивает прозрачность и проверяемость голосования, позволяя в любой момент подтвердить факт участия пользователя в опросе через блокчейн. В таблице 6 представлена спецификация логической модели отношений базы данных.

Таблица 6 – Спецификация отношений логической модели базы данных

Таблица	Связанная таблица	Тип связи	Внешний ключ	Описание связи
Polls	Users	Один ко многим	Creator_address	Ссылка на пользователя, который является автором опроса
Votes	Users	Один ко многим	User_address	Ссылка на пользователя, который проголосовал
Votes	Polls	Один ко многим	Poll_id	Ссылка на опрос, на который был оставлен голос

Диаграмма отношений логической модели базы данных представлена на рисунке 14.

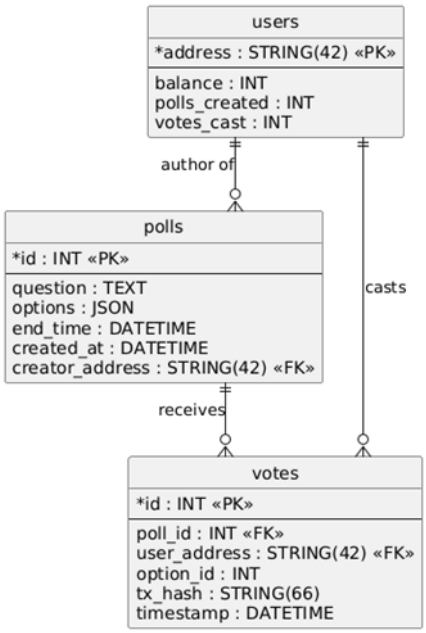

Рисунок 14 – Диаграмма отношений логической модели базы данных

Данные отношения находятся в второй нормальной форме, так как все неключевые атрибуты зависят от первичного ключа. И все значения атомарны, то есть находятся уже в первой нормальной форме. Следующий этап проектирования уточнит типы и размер полей для хранения данных.

2.3.3 Разработка физической модели базы данных

Физическая модель базы данных определяет реализацию логической модели в конкретной СУБД (SQLite), включая типы данных, индексы, ограничения и оптимизацию производительности.

Пример описания физической реализации основных таблиц приведен далее.

Таблица users состоит из:

– address имеет тип VARCHAR(42) и является первичным ключом.

– balance имеет тип INTEGER со значением по умолчанию 0.

– polls_created имеет тип INTEGER со значением по умолчанию 0.

– votes_cast имеет тип INTEGER со значением по умолчанию 0.

Таблица Votes состоит из:

– id имеет тип SERIAL и является первичным ключом;

– poll_id имеет тип INTEGER, внешний ключ, ссылающийся на поле id из таблицы polls;

– user_address имеет тип VARCHAR(42), внешний ключ, ссылающийся на поле address из таблицы users;

– option_id имеет тип INTEGER;

– tx_hash имеет тип VARCHAR(66);

– timestamp имеет тип TIMESTAMP со значением по умолчанию NOW().

Таблица Polls состоит из:

– id имеет тип SERIAL и является первичным ключом;

– question имеет тип TEXT, не содержит ограничений;

– options имеет тип JSON;

– end_time имеет тип TIMESTAMP;

– created_at имеет тип TIMESTAMP со значением по умолчанию NOW();

– creator_address имеет тип VARCHAR(42), внешний ключ, ссылающийся на поле address в таблице users.

На Рисунке 15 представлена диаграмма физической модели базы даны в терминах и типах СУБД SQLlite.

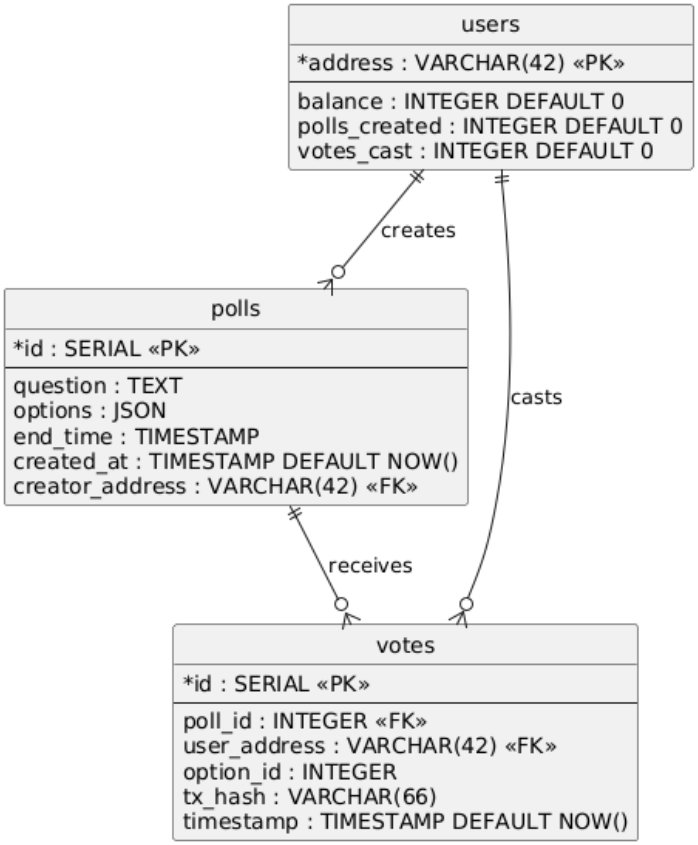

Рисунок 15 – Диаграмма физической модели базы данных системы PollSystem

В процессе физического проектирования базы данных была получена база данных в представлении в СУБД SQLite. Скрип генерации базы данных приведен в приложении А.

3. Реализация интернет-голосования на основе смарт-контракта

Реализация проекта PollSystem была реализована с помощью смарт контрактов на платформе Ethereum на языке программирования solidity, которая позволяет обеспечить децентрализацию и прозрачность голосования. Реализация взаимодействия смарт контрактов с традиционной серверной и клиентской частями была обеспечена через Web3 интерфейс, Rest API, СУБД SQLite и React JS.

3.1 Реализация смарт-контрактов голосования

Для системы было создано два смарт контракта: PollSystem, PollToken. PollSystem написан для взаимодействия с серверной частью, PollToken описывает внутреннюю валюту системы.

PollSystem состоит из функций:

– создание опросов с оплатой комиссии в виде внутренней валюты (poll);

– голосование строго через цифровую подпись;

– защита от двойного голосования;

– начисление внутренней валюты (POLL) в качестве поощрения за участие.

Контракт PollToken был разработан отдельно и реализует стандарт ERC20 с возможностью выпуска новых внутренних денежных единиц организатором.

Контракт написан на языке Solidity и использует библиотеку OpenZeppelin для реализации стандартов ERC20, а также модулей безопасности (Ownable, ECDSA).

Взаимодействие серверной части с контрактом реализовано с помощью библиотеки Web3.py. Сервисы FastAPI создают, читают и отправляют транзакции при помощи аккаунта администратора. Электронная подпись проверяется сначала на клиенте, затем транслируется через релейер сервер в блокчейн с минимизацией затрат со стороны пользователя.

Важнейшие функции контракта: createPoll(string question, string[] options, uint256 duration), voteWithSignature(uint pollId, uint optionId, address voter, bytes signature), getPoll(...), getVotes(...) – функции чтения данных. Они необходимы для получения информации об активных опросах, сформированных вариантах ответов, времени завершения голосования, а также для отображения текущих результатов голосования. Эти функции являются «view», то есть не изменяют состояние блокчейна и могут быть вызваны без комиссии, что делает их удобными для отображения данных на клиентской стороне в реальном времени. Функция createPoll инициирует создание нового опроса, автоматически увеличивает счетчик опросов, записывает вопрос, варианты ответов и время действия голосования, при этом взимая комиссию во внутренней валюте POLL с инициатора. В свою очередь, voteWithSignature позволяет зафиксировать голос, подписанный участником вне сети (off-chain), и проверить его подлинность перед записью результата в блокчейн. Такая схема позволяет эффективно разгрузить сеть от лишних транзакций, обеспечивая при этом неизменяемость и достоверность данных. Дополнительное использование сигнатуры в формате EIP-191 (через ECDSA) предоставляет возможность защищенной авторизации без необходимости отправлять транзакции напрямую с пользовательского кошелька, снижая барьер входа для участников. Таким образом, архитектура смарт-контрактов и серверного взаимодействия была построена с приоритетом на децентрализацию, безопасность, прозрачность и удобство для конечного пользователя [10].

3.2 Реализация работы с базой данных

Для хранения информации, связанной с пользователями, опросами и голосами, используется СУБД SQLite. Работа с базой реализована при помощи SQLAlchemy.

Таблица user: хранит адреса кошельков, количество созданных опросов и голосов. Таблица polls: сохраняются данные об опросах, созданных через смарт-контракт. Таблица votes: содержит информацию о голосах, привязанных к хешу

транзакции и опции голосования. Для доступа к данным таблиц используется строка кода session = SessionLocal(), которая открывает новую сессию взаимодействия с базой данных. Примером использования базы данных может служить добавление нового пользователя в таблицу users при первом голосовании. Сначала сессия работы с базой данных инициализируется с помощью строки session = SessionLocal(). Затем происходит попытка получить пользователя по его адресу: если запись отсутствует, создаётся новый экземпляр модели User с начальными значениями – адрес кошелька, количество созданных опросов и количество голосов. Этот объект добавляется в базу с помощью session.add(user), после чего изменения фиксируются через session.commit(). Такой подход позволяет автоматически регистрировать новых участников по мере их активности в системе. Код работы с таблицей User представлен в листинге 1.

Листинг 1 – Код взаимодействия с таблицей User

```
from models import SessionLocal, User, Poll, Vote
session = SessionLocal()
user = session.get(User, req.voter)
if not user:
user=User(address=req.voter,polls_created=0,votes_cast=0)
    session.add(user)
```

Сервер интернет-голосования с использованием смарт-контрактов реализует сопряжение между блокчейн-сетью и внутренней инфраструктурой платформы через набор REST API-эндпоинтов. Одной из ключевых функций сервера является прослушивание и обработка событий, поступающих из опубликованных смарт-контрактов – например, событие PollCreated, возникающее при успешной регистрации нового опроса в блокчейне. После получения такого события сервер извлекает с помощью Web3 необходимую информацию из контракта (вопрос, варианты ответов, идентификатор опроса и т.д.), а затем дополняет ее метаданными, которые не сохраняются в блокчейне по соображениям экономии ресурсов. К таким данным относятся, например, точное время создания опроса в формате datetime, адрес инициатора (создателя)

опроса и другие параметры, важные для пользовательской аналитики.

На рисунке 16 представлена диаграмма последовательности.

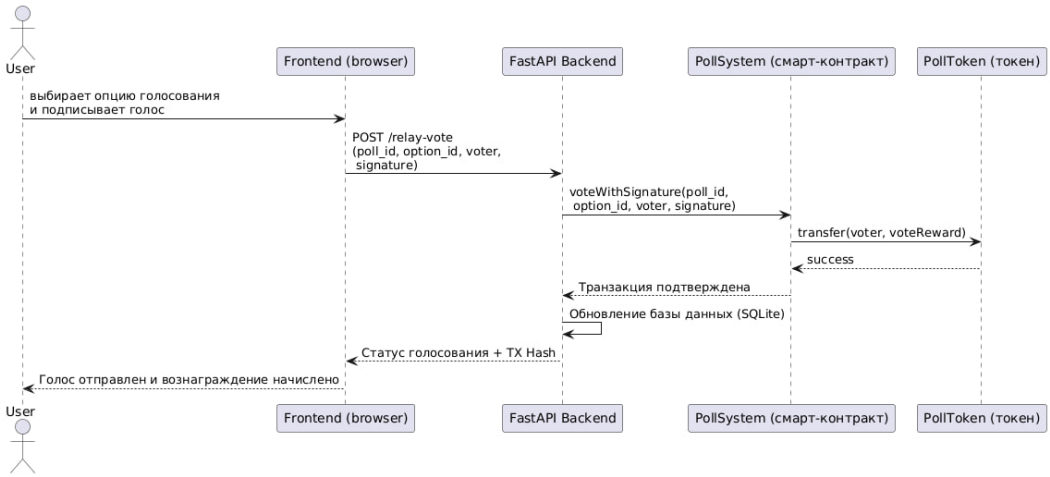

Рисунок 16 – Диаграмма последовательности

Дополненная информация сохраняется в локальную базу данных посредством ORM-средств (например, SQLAlchemy), после чего становится доступной для клиентских приложений через специально разработанные REST API-эндпоинты: /polls – получение списка всех доступных опросов, /create-poll – создание нового опроса с передачей данных на блокчейн, /leaderboard – отображение таблицы активности пользователей по балансу внутренней валюты, /balance/{address} – получение баланса внутренней валюты POLL для конкретного адреса [11].

При создании нового опроса информация о нём извлекается из смарт-контракта после вызова функции createPoll, дополняется метаинформацией, такой как точное время создания и адрес создателя, и сохраняется в базу данных средствами SQLAlchemy. Код сохранения метаинформации об опросе в локальную базу данных представлен на листинге 2.

Листинг 2 – Код сохранения метаинформации об опросе в локальную базу данных

```
db_poll = Poll(
    question=poll_data['question'],
    options=poll_data['options'],
    end_time=datetime.fromtimestamp(poll_data['end_time'],
tz=timezone.utc),
```

```
    creator_address=ADMIN,
    created_at=datetime.now(timezone.utc)
)
session.add(db_poll)
```

Основной листинг программных средств приведен в приложении А.

Такой подход обеспечивает гибкость и масштабируемость системы: данные, требующие прозрачности и неизменяемости, хранятся в блокчейне, тогда как вспомогательная информация кэшируется на сервере для оптимизации быстродействия и снижения комиссии блокчейна (газа). В результате достигается высокоэффективная архитектура, объединяющая преимущества децентрализации и удобства современного подхода к архитектуре информационных систем.

3.3 Реализация клиентской части для интернет-голосования

Клиентская часть проекта была реализована с использованием языка программирования TypeScript и библиотеки React, что обеспечило современный, типизированный и реактивный интерфейс пользователя. React используется для построения компонентов, отвечающих за отображение опросов, управление выбором, подключение цифрового кошелька и взаимодействие с сервером через REST API. Связь с блокчейн-сетью Ethereum осуществляется посредством криптовалютного кошелька MetaMask, который интегрируется через библиотеку web3.js.

MetaMask представляет собой расширение браузера, которое предоставляет доступ к аккаунту пользователя в Ethereum-сети, хранилищу ключей и функциональности подписи сообщений и транзакций. Приложение использует API MetaMask (window.ethereum) для аутентификации, извлечения адреса пользователя и создания цифровых подписей.

У процесса голосования на клиентской стороне можно выделить несколько этапов: создание криптографической подписи сообщения, отправка подписи на сервер, обработка подписи и транзакции на сервере.

Когда пользователь выбирает вариант ответа в опросе, запускается процесс

создания подписи. Подписывается не вся транзакция, а лишь сообщение, включающее в себя следующие параметры: pollId – идентификатор опроса, optionId – выбранный вариант ответа, user_address – адрес пользователя в сети Ethereum.

На клиентской стороне вызывается функция signMessage из API MetaMask, которая запускает окно подтверждения, и пользователь подписывает сообщение приватным ключом, не раскрывая его. Код создания подписи на клиентской стороне представлен на листинге 3.

Листинг 3 – Код создания подписи на клиентской стороне

```
const ethereum = (window as any).ethereum;
const signer = ethereum.selectedAddress;
const message = Web3.utils.soliditySha3(pollId, optionId, signer);
const signature = await ethereum.request({
  method: "personal_sign",
  params: [message, signer],
});
```

Основной листинг программных средств приведен в приложении А.

В результате получается строка-подпись, которая подтверждает, что именно владелец указанного адреса выбрал конкретный вариант в опросе.

После получения подписи, клиент формирует тело запроса и отправляет его на backend посредством POST-запроса к API/relay-vote. Отправляется следующая информация: poll_id – ID опроса, option_id – выбранный вариант, voter – адрес пользователя, signature – криптографическая подпись. Код отправки подписи на серверную часть представлен на листинге 4.

Листинг 4 – Код отправки подписи на серверную часть

```
await fetch("/relay-vote", {
  method: "POST",
  headers: {
    "Content-Type": "application/json"
  },
  body: JSON.stringify({
    poll_id: pollId,
    option_id: optionId,
    voter: signer,
    signature: signature  }));
```

На этом этапе за счёт передачи заранее сформированной цифровой подписи на сервер достигается упрощение взаимодействия пользователя с системой. Вместо необходимости самостоятельно отправлять транзакцию в блокчейн, пользователь лишь подтверждает своё волеизъявление подписью, что делает процесс голосования быстрым, удобным и безопасным.

На сервере запрос с подписью анализируется, проверяется её подлинность (восстанавливается адрес на основе хэша сообщения и подписи), и только после валидации вызывается функция voteWithSignature() смарт-контракта от имени администратора. Таким образом, транзакция осуществляется от сервера, а не клиента, и пользователь не платит комиссии (газ). Это упрощает участие в голосованиях и снижает барьер входа для пользователей без ETH на счету.

Передача подписи на сервер позволяет централизованно управлять транзакционной частью голосования. Сервер проверяет подлинность подписи, связывая её с адресом пользователя, и выполняет транзакцию в смарт-контракт от имени администратора. Это снимает с пользователя необходимость оплачивать комиссию за газ и делает участие в голосовании доступным даже при нулевом балансе на кошельке.

В таком подходе есть достаточно преимуществ:

– безопасность: закрытый ключ остаётся только у пользователя, транзакция подписывается безопасным образом через MetaMask;

– простота: пользователь голосует без взаимодействия с блокчейном напрямую;

– экономичность: комиссии оплачивает сервер или организатор, а не конечный пользователь;

– масштабируемость: все голосования происходят с верифицированной подписью, а транзакции отправляются централизованно одной сущностью – сервером через аккаунт администратора.

Таким образом, клиентская часть выступает интерфейсом для взаимодействия с кошельком и отправки криптографически подписанных сообщений. Благодаря использованию MetaMask и Web3, пользователи могут

голосовать безопасно, без расходов на газ, при сохранении всех преимуществ децентрализованного подтверждения подписи и прозрачности голосования [12].

3.3.1 Получение доступа к голосованию

Пользователь получает доступ к системе голосования при наличии внутренней валюты POLL, которая выступает не только средством оплаты за создание опросов, но и инструментом поощрения активности участников. В качестве метода аутентификации используется цифровой кошелёк, например MetaMask, через который осуществляется безопасное и децентрализованное подтверждение личности. Процесс голосования начинается с того, что пользователь на веб-интерфейсе выбирает актуальный опрос и указывает интересующий его вариант ответа. После этого инициируется создание подписи – в кошельке автоматически формируется криптографическая подпись сообщения, включающего уникальный идентификатор опроса, номер выбранного варианта и адрес кошелька пользователя. Эта подпись гарантирует, что голос действительно исходит от владельца указанного адреса, и она используется в качестве необратимого доказательства выбора.

Сформированная подпись передаётся на сервер с помощью POST-запроса к специально разработанному REST API /relay-vote, а сервер, в свою очередь, выполняет её валидацию, восстанавливая адрес отправителя из подписи и сравнивая его с переданным значением. Это позволяет убедиться в подлинности подписи и предотвратить попытки подделки или повторного голосования. После успешной проверки сервер инициирует транзакцию голосования, вызывая функцию voteWithSignature в смарт-контракте от имени администратора. Таким образом, пользователь не взаимодействует с блокчейном напрямую и не несёт затрат на оплату газа, а все действия происходят в фоновом режиме от имени доверенной серверной стороны.

На Рисунке 17 представлена диаграмма деятельности получения доступа к голосованию.

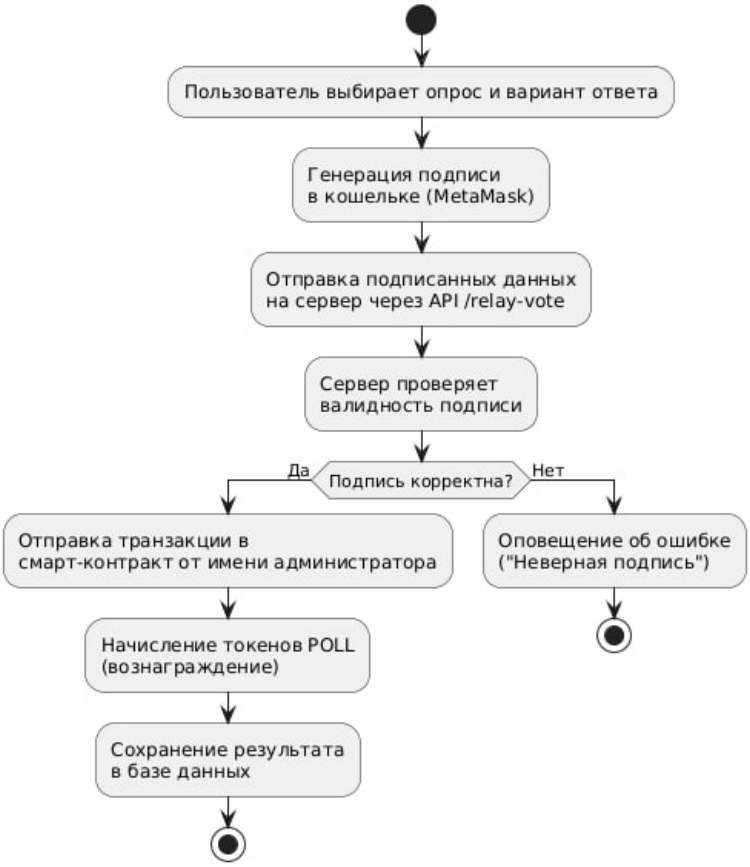

Рисунок 17 – диаграмма деятельности доступа к голосованию

После завершения транзакции смарт-контракт фиксирует голос, а пользователю автоматически начисляется вознаграждение в токенах POLL за участие в голосовании. Это стимулирует пользователей активно вовлекаться в опросы и участвовать в жизни децентрализованного приложения. Дополнительно, сервер сохраняет все приобретённые данные – такие как идентификатор опроса, выбранный вариант, хеш транзакции и адрес пользователя – в локальную базу данных. Эта информация позже используется для отображения истории голосований, генерации таблиц активности и анализа пользовательского участия. Таким образом, реализованный процесс сочетает в себе прозрачность, безопасность, удобство и мотивацию, обеспечивая надёжную и эффективную архитектуру децентрализованного интернет-голосования.

Таким образом, представленный алгоритм голосования демонстрирует прозрачную и безопасную модель взаимодействия между пользователем, сервером и блокчейн-сетью. Использование цифровой подписи позволяет

подтверждать действия пользователя без необходимости прямого взаимодействия с контрактом и затрат на комиссии, обеспечивая при этом высокую достоверность данных. Централизованная обработка транзакций сервером в связке с децентрализованной фиксацией результатов в смарт-контракте делает систему удобной для пользователя и при этом сохраняет принципы прозрачности и неизменяемости, присущие блокчейн-технологиям.

На рисунке 18 представлена подпись голоса через кошелек Metamask.

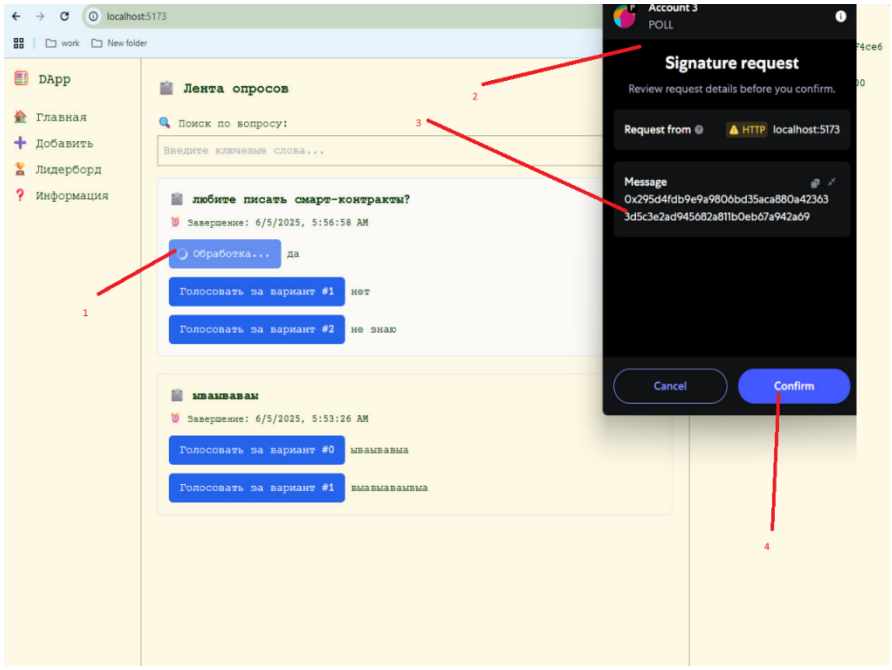

Рисунок 18 – Форма подписания ответа через кошелек MetaMask

Элементы страницы обозначены цифрами:

– выбранный вариант ответа в опросе;
– аккаунт кошелька пользователя;
– хэш транзакции подписи;
– кнопка подтверждения транзакции.

На представленном рисунке демонстрируется процесс подтверждения выбора пользователя через цифровую подпись, создаваемую в кошельке MetaMask. Такая реализация обеспечивает высокий уровень безопасности, поскольку доступ к голосованию возможен только после подтверждения транзакции самим пользователем. Использование криптографической подписи

как метода аутентификации исключает возможность подделки голосов и гарантирует их подлинность. Интерфейс интуитивен: пользователь выбирает вариант, подписывает транзакцию и получает подтверждение. Таким образом, система реализует надежный, децентрализованный и прозрачный механизм получения доступа к голосованию.

4 Тестирование

В рамках тестирования и анализа качества программного обеспечения системы децентрализованного голосования были использованы следующие методы:

– модульное тестирование;
– функциональное тестирование;
– метрики Холстеда.

Дальнейшее тестирование проводилось с целью обеспечения надежной работы системы, выявления логических и структурных ошибок, а также проверки корректности взаимодействия всех компонентов приложения – от пользовательского интерфейса до смарт-контрактов.

Модульное тестирование применялось для изолированной проверки отдельных компонентов серверной части, включая валидацию входных данных, генерацию подписей, формирование и отправку транзакций в блокчейн. Это позволило оперативно локализовать возможные дефекты и упростить отладку кода.

Функциональное тестирование проводилось вручную и охватывало весь пользовательский сценарий участия в голосовании: подключение кошелька, выбор варианта ответа, подписание сообщения в MetaMask, отправка запроса на сервер, обработка подписи и выполнение транзакции в смарт-контракте от имени администратора. Такой подход обеспечил проверку всех ключевых функций системы в условиях, максимально приближенных к реальному использованию.

Для оценки сложности логики и качества кода серверной части применялась метрика Холстеда. С её помощью удалось количественно оценить трудоёмкость реализованных решений, а также потенциальный объём информации, проходящий через систему. Эти данные позволили повысить обоснованность проектных решений и провести незначительную рефакторинговую оптимизацию критичных участков.

Комплексное использование данных методов позволило убедиться в стабильности работы системы, а также повысить её готовность к эксплуатации и масштабированию.

4.1 Функциональное ручное тестирование интернет-голосования

процессе разработки и отладки системы децентрализованного интернет-голосования было выполнено ручное функциональное тестирование ключевых пользовательских сценариев. Основное внимание уделялось проверке корректности работы интерфейса голосования, внутренней логики обработки подписи на серверной части, а также фиксации результатов в смарт-контракте на блокчейне. Целью испытаний являлось подтверждение соответствия реализованной системы заявленным требованиям, а также обеспечение надежности функционирования в условиях, приближенных к реальной эксплуатации.

Тестирование проводилось по методике «черного ящика», при которой внимание сосредоточено преимущественно на входных и выходных данных, без анализа внутреннего исходного кода. Такой подход позволил протестировать систему с точки зрения конечного пользователя: начиная с момента открытия интерфейса голосования и заканчивая подтверждением успешной отправки голоса и получением вознаграждения [13]. На этапе моделирования входных условий были установлены ключевые причины – наборы начальных состояний и действий со стороны пользователя (например, доступность кошелька, выбор опроса, формирование цифровой подписи), по результату которых система должна была породить определенные следствия (например, сообщение об ошибке, корректная обработка транзакции, начисление токенов и пр.).

Для того чтобы структурировать зависимости между причинами и следствиями, была построена таблица причинно-следственных связей, в которой отражены все возможные комбинации входных условий (димесячные бинарные значения) и соответствующие им выходные состояния системы. Такая таблица

позволяет охватить не только идеальный успешный сценарий выполнения голосования, но также протестировать поведение при неполных или некорректных условиях: отсутствии доступа к кошельку MetaMask, невыбранном опросе, неправильной подписи и других ошибках пользователя. Причинно-следственные связи представлены на таблице 7.

Таблица 7 – Таблица причин и следствий

Причины	
П1	MetaMask доступен пользователю
П2	Пользователь авторизован в кошельке
П3	Пользователь выбрал валидный опрос
П4	Сформирована цифровая подпись
П5	Подпись отправлена на сервер
Следствия	
С1	MetaMask не найден
С2	Пользователь не авторизован
С3	Не выбран опрос
С4	Подпись недействительна
С5	Сервер отклонил голос
С6	Голос принят, POLL начислены

При использовании данной таблицы была составлена функциональная диаграмма, представленная на рисунке 19.

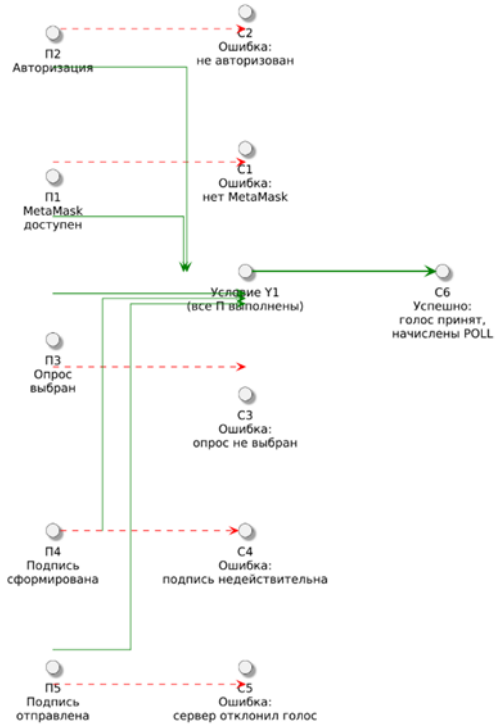

Рисунок 19 – Функциональная диаграмма

На основе диаграммы составляем схему решений. В верхней части – комбинация причин, в нижней – соответствующее следствие на каждую комбинацию причин. Схема решений представлена в таблице 8.

Таблица 8 – Таблица решений

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
П1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1
П2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1
П3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1
П4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
П5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
С1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0
С2	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0
С3	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	0
С4	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
С5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
С6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Из таблицы решений удалены несуществующую комбинации причин, затем были построены наборы тестов, расположенные в таблице 9.

Таблица 9 – Таблица тест-кейсов

№	Входные данные	Ожидаемый результат
1	MetaMask: Авторизация: Опрос: Подпись: Отправка:	Выводится сообщение: ТаблицаMetaMask не найден»
2	MetaMask: Включен Авторизация: Опрос: Подпись: Отправка:	Выводится сообщение: «Пользователь не авторизован»
3	MetaMask: Включен Авторизация: Выполнена Опрос: Подпись: Отправка:	Выводится сообщение: «Не выбран опрос»
4	MetaMask: Включен Авторизация: Выполнена Опрос: Выбран Подпись: Отправка:	Выводится сообщение: «Подпись недействительна»
5	MetaMask: Включен Авторизация: Выполнена Опрос: Выбран Подпись: Сформирована	Выводится сообщение: «Сервер отклонил голос»
6	MetaMask: Включен Авторизация: Выполнена Опрос: Выбран Подпись: Сформирована Отправка: Выполнена	Голос успешно зафиксирован, токены POLL начислены

Продолжение таблицы 9

№	Входные данные	Ожидаемый результат
7	MetaMask: Включен Авторизация: Выполнена Повторное голосование в том же опросе	Голос отклонён, выведено сообщение: «Вы уже голосовали»
8	MetaMask: Включен Авторизация: Выполнена Опрос: Выбран Подпись: Повреждена Отправка: Выполнена	Голос отклонён, выведено сообщение: «Подпись невалидна»

В результате проведённого ручного функционального тестирования системы децентрализованного интернет-голосования были успешно проверены основные пользовательские сценарии, включая процесс получения доступа к голосованию, формирование цифровой подписи, валидацию на сервере и запись результатов в смарт-контракт. Функциональное тестирование по методу «черного ящика» позволило выявить возможные отклонения при нарушении входных условий: отсутствие MetaMask, неавторизованный пользователь, отсутствие выбранного опроса или некорректная подпись.

На основе функциональной модели была составлена таблица причин и следствий, построена функциональная диаграмма и сформированы тест-кейсы, отражающие возможные критические и корректные сценарии поведения системы. Все тесты соответствовали ожидаемым результатам, что подтверждает корректность реализованной логики приложения и его готовность к дальнейшим этапам использования и автоматизации тестирования. Ручная проверка обеспечила высокую степень уверенности в надёжности системы при типовых пользовательских взаимодействиях.

4.2 Метрики кода

Для определения метрик кода используются метрики Холстеда. Была выбрана функция get_balance, реализованная в серверной части системы интернет-голосования.

Данная функция позволяет проверить, голосовал ли пользователь ранее в конкретном опросе, и является частью логики валидации повторных голосов. Код функции get_balance приведён в листинге 5. Основной листинг программных средств приведен в приложении А.

Листинг 5 – Код функции get_balance

```
@app.get("/balance/{address}")
def get_balance(address: str):
    try:
        if not Web3.is_address(address):
raise HTTPException(status_code=400, detail="Неверный адрес
кошелька")

        checksum = Web3.to_checksum_address(address)
   balance = poll_token.functions.balanceOf(checksum).call()
        decimals = poll_token.functions.decimals().call()
        symbol = poll_token.functions.symbol().call()

        return {
            "address": checksum,
            "balance": str(balance),
            "symbol": symbol,
            "decimals": decimals
        }

    except Exception as e:
        raise HTTPException(status_code=500, detail="Ошибка
получения баланса")
```

Для вычисления метрик Холстеда были определены: количество уникальных операторов (n1), количество уникальных операндов (n2), общее количество операторов (N1) и операндов (N2)

Операторы функции get_balance и число их вхождений приведены в таблице 10.

Таблица 10 – Операторы функции get_balance

№	Оператор	Количество вхождений
1	def	1
2	try/except	2
3	if	1
4	not	1
5	raise	2
6	= (присваивание)	5
7	return	1
8	. (атрибуты)	8
9	() (вызов функции)	8
10	{} (словарь)	1
	Итого N1	30

Операнды функции get_balance и число их вхождений в функцию приведены в таблице 11.

Таблица 11 – Операнды функции get_balance

№	Операнд	Количество вхождений
1	address	3
2	Web3	2
3	poll_token	3
4	functions	3
5	balanceOf, decimals, symbol	3
6	checksum	2
7	balance	2
8	str	1
9	symbol	1
10	HTTPException	2
11	«Неверный адрес кошелька»	1
12	«Ошибка получения баланса»	1
13	status_code=400	1
14	status_code=500	1

Продолжение таблицы 11

№	**Операнд**	**Количество вхождений**
15	detail=...	2
	Итого N2	28

По таблицам операторов и операндов были проведены вычисления метрик Холстеда. Результаты приведены на таблице 12.

Таблица 12 – Результаты вычисления метрик Холстеда

Показатель	Результат
Длина программы (N = N1 + N2)	58
Словарь программы (n = n1+ n2)	25
Объём (V = N × log2 n)	269,12
Сложность (D = n1 / 2 × N2 / n2)	9,35
Усилия (E = D × V)	2516,47
Время разработки (T = E / 18)	139,8 сек.
Прогноз ошибок (B = E^(2/3) / 3000)	0,0063

Анализ функции get_balance по метрике Холстеда показывает умеренную структурную сложность (n=25), достаточный объём программы и низкий прогноз дефектов. Это свидетельствует о грамотной реализации кода для взаимодействия с контрактом и эффективной работе с исключениями. Низкий уровень потенциальных ошибок и высокая читаемость функции делают её устойчивой в эксплуатации. Данные, полученные в ходе анализа, могут быть использованы для дальнейшего сравнения сложности компонентов в составе всей системы.

4.3 Автоматизированное модульное тестирование интернет-голосования

Для автоматизации процесса проверки корректности работы основных компонентов системы была выбрана библиотека Pytest, предоставляющая удобные инструменты для модульного и функционального тестирования. Pytest

позволяет формировать независимые тестовые сценарии, выполнять их пакетно, а также автоматически отслеживать и фиксировать ошибки.

Формируемое автоматическое тестирование охватывает все ключевые сценарии использования системы, включая базовые операции (создание и вывод опросов, обработка подписей, начисление токенов) и потенциально необязательно-проходящие ветви логики (например, ошибки авторизации, повторное голосование, отсутствие опроса и др.). Каждое тестовое покрытие ориентировано на один из пользовательских или системных кейсов и описывает как входные данные, так и ожидаемый вывод. Нагрузочные проверки выполнялись на серверной части приложения и охватывали такие участки, как Web3-интеграция, валидация данных и работа с базой данных через SQLAlchemy.

Результаты выполнения автоматизированного тестирования были обобщены и представлены в таблице 13. Для каждого теста зафиксированы параметры и соответствующие результаты выполнения, что позволяет объективно оценить стабильность поведения функциональных блоков системы.

Таблица 13 – Тестирование

Тест	**Описание**	**Результат**
test_health	Проверка доступности API и подключения к смарт-контракту	Тест пройден
test_create_poll_invalid	Проверка отклонения некорректных параметров при создании опроса	Тест пройден
test_invalid_signature	Проверка отказа в голосовании при недействительной подписи	Тест пройден
test_token_reward_ on_vote_multiple_polls	Проверка выдачи вознаграждения при голосовании	Тест пройден

Продолжение таблицы 13

Тест	**Описание**	**Результат**
test_create_user_manually	Тест создания пользователя вручную в БД	Тест пройден
test_get_balance	Получение и валидация баланса пользователя по адресу	Тест пройден

На рисунке 20 приведен результат выполнения автотестов.

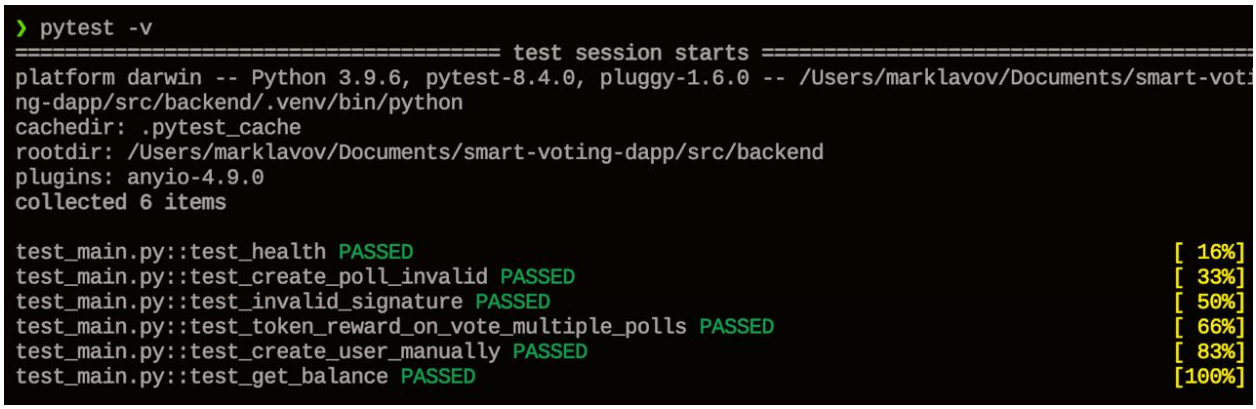

Рисунок 20 – Результат выполнения тестов

Результаты тестирования показали, что программная система интернет-голосования на основе блокчейна демонстрирует стабильную и предсказуемую работу при различных условиях эксплуатации. Были успешно протестированы как типичные пользовательские сценарии, соответствующие основному функционалу системы, так и граничные случаи, включающие нештатные ситуации, обработку ошибок, некорректные подписи и отклонения от стандартного пользовательского поведения.

По итогу тестирования подтвержден корректность выполнения всех ключевых функций системы, включая проверку авторизации, обработку входящих данных, взаимодействие со смарт-контрактом и возврат ожидаемых ответов через API. Большинство тестов завершились успешно, без выявления критических сбоев, что свидетельствует о высокой степени работоспособности

логики приложения. Выявленные незначительные отклонения были учтены и устранены в процессе отладки. Таким образом, проведённое тестирование позволяет сделать вывод о готовности системы к эксплуатации в условиях реальной пользовательской загрузки.

Заключение

В рамках дипломного проектирования была спроектирована, реализована и протестирована система интернет-голосования с использованием технологии смарт-контрактов в блокчейн-сети Ethereum. Разработка охватывала полный цикл создания программного обеспечения – от анализа предметной области до тестирования готового продукта.

На начальном этапе был проведён детальный анализ предметной области, в результате чего выделены основные классы пользователей и их роли в процессе голосования. Далее был выполнен анализ существующих программных решений и аналогов, что позволило оценить их преимущества и ограничения, чтобы учесть это при построении собственной системы. Особое внимание было уделено исследованию современного рынка приложений на основе смарт-контрактов и рассмотрению перспектив их применения в электронном голосовании. На этой основе были сформулированы как функциональные, так и нефункциональные требования к системе: они включали требования к безопасности, доступности, отказоустойчивости, совместимости и удобству использования.

После установления требований был произведён обоснованный выбор технологий и инструментов разработки. В качестве ядра системы выбрана сеть Ethereum со смарт-контрактами на языке Solidity, интеграция с которой реализована посредством библиотеки Web3.py на серверной части, написанной на языке Python. Клиентская часть была реализована с использованием фреймворка React и языка TypeScript, обеспечивающих кроссплатформенный доступ и удобный интерфейс пользователя, в том числе через криптокошелёк MetaMask. Также была составлена формализованная постановка задачи, в которой были определены основные цели проекта, технические ограничения и ожидаемые результаты.

На этапе проектирования было разработано логическое описание архитектуры системы, а также спроектирован удобный пользовательский

интерфейс. Особое внимание уделено формированию базы данных: были последовательно спроектированы концептуальная, логическая и физическая модели, обеспечивающие хранение дополнительной информации, не критичной к размещению в блокчейне, – таких как данные о пользователях, кастомные настройки и структура опросов.

В процессе реализации проекта были разработаны смарт-контракты для создания и проведения голосований, реализующие механизмы защиты от повторного голосования и начисления вознаграждения за участие. Одновременно была построена серверная часть, взаимодействующая как с блокчейном, так и с локальной базой данных. На стороне клиента реализован интерфейс с авторизацией через MetaMask и полной поддержкой подписанных голосований с формированием безопасной транзакции.

Завершающим этапом разработки стало комплексное тестирование системы. Выполнено ручное функциональное тестирование методом «черного ящика», проверяющее корректность реакции системы в стандартных и граничных ситуациях. Дополнительно, с целью стабильности и отказоустойчивости, были написаны автотесты с использованием библиотеки Pytest, что позволило выявить и устранить логические ошибки ещё до этапа эксплуатации.

Таким образом, в рамках выполнения дипломной работы была разработана и реализована полнофункциональная система интернет-голосования, отвечающая современным требованиям безопасности, прозрачности и децентрализации. Система сочетает в себе архитектурное соответствие современным стандартам и пользовательскую простоту. Результаты проектирования и тестирования демонстрируют её технологическую готовность к использованию в реальных условиях и возможность масштабируемости под более сложные кейсы – от систем электронного принятия решений до внедрения в концепции DAO и электронного самоуправления.

Список использованных источников

1. Рассказов, С.В. СОЦИОЛОГИЧЕСКИЙ ВЗГЛЯД НА УЧАСТИЕ ГРАЖДАН В СИСТЕМЕ ЭЛЕКТРОННОГО ГОЛОСОВАНИЯ С БЛОКЧЕЙН // С.В. Рассказов, А.С. Изюмова, А.Н. Рассказова // УПРАВЛЕНИЕ РАЗВИТИЕМ КРУПНОМАСШТАБНЫХ СИСТЕМ MLSD'2020 ТРУДЫ ТРИНАДЦАТОЙ МЕЖДУНАРОДНОЙ КОНФЕРЕНЦИИ. № 13. - 2020. - 1950-1960. url: https://mlsd2020.ipu.ru/proceedings/1950-1960.pdf (дата обращения: 10.05.2025)

2. Buterin V. «Ethereum Whitepaper» [электронный ресурс]. URL: https://ethereum.org/en/whitepaper (дата обращения: 10.05.2025)

3. Spanos, A. A Blockchain-based Electronic Voting System: EtherVote / A. Spanos, I. Kantzavelou // Proceedings of ACM Gender Equality in Computing ('GEC 2023). – New York, 2023. – P. 2.

4. Рассказов, С. В. Социологический взгляд на участие граждан в системе электронного голосования с блокчейн / С. В. Рассказов, А. С. Изюмова, А. Н. Рассказова // Полис. Политические исследования. – 2018. – № 5. – С. 88-99.

5. Баракова Алия Шаризатовна, Усатова Ольга Александровна СРАВНЕНИЕ МЕТОДОВ АУТЕНТИФИКАЦИИ НА ВЕБ-РЕСУРСАХ // Труды Международной Азиатской школы-семинара «Проблемы оптимизации сложных систем». 2023. №. URL: https://cyberleninka.ru/article/n/sravnenie-metodov-autentifikatsii-na-veb-resursah (дата обращения: 10.05.2025).

6. Функциональное тестирование программного обеспечения [электронный ресурс] https://daglab.ru/funkcionalnoe-testirovanie-programmnogo-obespechenija/ (дата обращения: 14.05.2025)

7. Guo, Y. Zk-SNARKs-Based Anonymous Payment Channel in Blockchain / Y. Guo, H. Liang, L. Zhu, K. Gai // Blockchains. – 2024. – Vol. 2, Is. 1. – P. 20-39. – DOI: 10.3390/blockchains2010002.

8. Rodrigues, B. Blockchain and Smart Contracts: From Theory to Practice / B. Rodrigues, R. Blum, T. Bocek, B. Stiller // 14th International Conference on Network and Service Management (CNSM). – Rome, 2018. – P. 1-31.

9. Балашова, И. Ю. Современные информационные технологии в проектировании программных систем и комплексов: учеб. пособие / И. Ю. Балашова, Д. В. Такташкин; под ред. П. П. Макарычева. – Пенза : Изд-во ПГУ, 2019. – 106 с.

10. Антонопулос, А. М. Mastering Ethereum: Building Smart Contracts and DApps / А. М. Антонопулос, Г. Вуд. – Sebastopol, CA : O'Reilly Media, 2018. – 352 с.

11. Dannen, C. Introducing Ethereum and Solidity / C. Dannen. – Berkeley, CA : Apress, 2017. – 195 p. – DOI: 10.1007/978-1-4842-2535-6.

12. Secure Smart Contract Development Patterns [Электронный ресурс] // OpenZeppelin Docs. – 2024. – URL: https://docs.openzeppelin.com/contracts/5.x (дата обращения: 14.05.2025).

13. Wood, G. Ethereum Yellow Paper [Электронный ресурс] / G. Wood. – 2024 ed. – URL: https://ethereum.github.io/yellowpaper (дата обращения: 14.05.2025).

14. Decentralized Identifiers (DIDs) v2.0 [Электронный ресурс] // W3C. – 2023. – URL: https://www.w3.org/TR/did-core (дата обращения: 15.05.2025).

15. Signers and Transactions [Электронный ресурс] // Ethers.js Documentation. – Version 6.0. – URL: https://docs.ethers.org/v6 (дата обращения: 15.05.2025).

16. Style Guide and Best Practices [Электронный ресурс] // Solidity Lang. – 2024. – URL: https://docs.soliditylang.org (дата обращения: 15.05.2025).

17. UML 2.0: проектирование программных систем / Д. Фаулер. – СПб.: Питер, 2023. – 480 с.

18. Мугаяр В. Блокчейн для бизнеса: Обещания, практика и применение технологий следующего поколения / В. Мугаяр. – Хобокен, Нью-Джерси : Wiley, 2016. – 208 с.

19. Аллин Д. У. Э., Лейн А. М., Берг К. Блокчейн и теория общественного выбора: Экономика, управление и политика / Д. У. Э. Аллин, А. М. Лейн, К. Б. Берг. – Челтнем : Edward Elgar Publishing, 2020. – 224 с.

20. Гоусман Д., Котарски Р. Блокчейн-революция: как технология биткойна меняет деньги, бизнес и мир / Д. Гоусман, Р. Котарски. – Москва : Эксмо, 2017. – 430 с.

Приложение А

КОД ПРОГРАММЫ

(обязательное)

Листинг А1 - PollToken.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

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

contract PollToken is ERC20, Ownable {
    constructor() ERC20("Poll Token", "POLL") {
        _mint(msg.sender, 1000000 * 1e18);
    }

    /**
     * @dev Функция для выпуска новых токенов (доступна только владельцу контракта)
     * @param to Адрес получателя новых токенов
     * @param amount Количество токенов для выпуска (в наименьших единицах)
     */
    function mint(address to, uint256 amount) public onlyOwner {
        _mint(to, amount);
    }
}
```

Листинг А2 - PollSystem.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/utils/cryptography/ECDSA.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract PollSystem is Ownable {
    using ECDSA for bytes32;

    IERC20 public pollToken;
    address public relayer;

    uint256 public pollCreationFee = 100 ether;
    uint256 public voteReward = 10 ether;

    struct Poll {
        string question;
        string[] options;
        uint256 endTime;
        mapping(address => bool) voters;
        mapping(uint256 => uint256) votes;
```

```
    }

    uint256 public pollCount;
    mapping(uint256 => Poll) private polls;

    event PollCreated(uint256 indexed pollId, string question);
    event Voted(uint256 indexed pollId, address indexed voter,
uint256 optionId);
    event Rewarded(address indexed voter, uint256 amount);
    event RelayerChanged(address oldRelayer, address newRelayer);

    constructor(address _token, address _relayer) Ownable() {
        pollToken = IERC20(_token);
        relayer = _relayer;
    }

    function setRelayer(address newRelayer) external onlyOwner {
        emit RelayerChanged(relayer, newRelayer);
        relayer = newRelayer;
    }

    function createPoll(
        string memory _question,
        string[] memory _options,
        uint256 _duration
    ) external {
        require(bytes(_question).length > 0, "Empty question");
        require(_options.length >= 2, "Need at least 2 options");
        require(_duration > 0, "Duration must be > 0");

        bool success = pollToken.transferFrom(msg.sender,
address(this), pollCreationFee);
        require(success, "Poll creation fee failed");

        Poll storage newPoll = polls[pollCount];
        newPoll.question = _question;
        newPoll.options = _options;
        newPoll.endTime = block.timestamp + _duration;

        emit PollCreated(pollCount, _question);
        pollCount++;
    }

    function voteWithSignature(
        uint256 pollId,
        uint256 optionId,
        address voter,
        bytes memory signature
    ) external {
        require(msg.sender == relayer, "Only relayer can relay");

        Poll storage poll = polls[pollId];
```

```
        require(!poll.voters[voter], "Already voted");
        require(optionId < poll.options.length, "Invalid option");
        require(block.timestamp <= poll.endTime, "Poll ended");

        bytes32 dataHash = keccak256(abi.encodePacked(pollId,
optionId, voter));
        bytes32 ethSignedHash = dataHash.toEthSignedMessageHash();
        address recovered = ethSignedHash.recover(signature);
        require(recovered == voter, "Invalid signature");

        poll.votes[optionId]++;
        poll.voters[voter] = true;

        emit Voted(pollId, voter, optionId);

        bool success = pollToken.transfer(voter, voteReward);
        require(success, "Reward transfer failed");
        emit Rewarded(voter, voteReward);
    }

    function getPoll(uint256 pollId)
        external
        view
        returns (
            string memory question,
            string[] memory options,
            uint256 endTime
        )
    {
        Poll storage p = polls[pollId];
        return (p.question, p.options, p.endTime);
    }

    function getVotes(uint256 pollId) external view returns
(uint256[] memory results) {
        Poll storage p = polls[pollId];
        uint256 length = p.options.length;
        results = new uint256[](length);
        for (uint256 i = 0; i < length; i++) {
            results[i] = p.votes[i];
        }
    }

    function getAllPollIds() external view returns (uint256[] memory
ids) {
        ids = new uint256[](pollCount);
        for (uint256 i = 0; i < pollCount; i++) {
            ids[i] = i;
        }
    }

    function hasVoted(uint256 pollId, address user) external view
returns (bool) {
```

```
        return polls[pollId].voters[user];
    }
}
```

Листинг А3 - schemas.py

```
from pydantic import BaseModel, Field, field_validator
from typing import List

class CreatePoll(BaseModel):
    question: str = Field(..., min_length=1)
    options: List[str]
    duration: int = Field(..., gt=0)

    @field_validator("options")
    @classmethod
    def validate_options(cls, v):
        if len(v) < 2:
            raise ValueError("At least two options required")
        return v

class RelayVoteRequest(BaseModel):
    poll_id: int
    option_id: int
    voter: str
    signature: str
```

Листинг А4 - test_main.py

```
import pytest
from fastapi.testclient import TestClient
from web3 import Web3
from eth_account import Account
from eth_account.messages import encode_defunct
from models import SessionLocal, User, Poll, Vote
from web3_setup import ADMIN
from main import app, poll_token, poll_system, w3


client = TestClient(app)


def test_health():
    response = client.get("/health")
    assert response.status_code == 200
    assert response.json()["status"] == "ok"


def test_create_poll_invalid():
    response = client.post("/create-poll", json={
        "question": "",
```

```
        "options": ["Yes"],
        "duration": 0
    })
    assert response.status_code == 422  # validation error


def test_invalid_signature():
    payload = {
        "poll_id": 0,
        "option_id": 1,
        "voter": "0x0000000000000000000000000000000000000000",
        "signature": "0xInvalidSignature"
    }
    response = client.post("/relay-vote", json=payload)
    assert response.status_code == 400


@pytest.fixture
def new_voter():
    return Account.create()


def test_token_reward_on_vote_multiple_polls(new_voter):
    created_polls = []
    for i in range(3): # упрощено: не надо 10, главное — хотя бы
несколько
        question = f"Test poll #{i + 1}?"
        response = client.post("/create-poll", json={
            "question": question,
            "options": ["Option A", "Option B"],
            "duration": 300
        })
        if response.status_code != 200:
            print(f"[‼] Ошибка создания опроса: {response.json()}")
            assert False, "Failed to create poll"

        created_polls.append(response.json())

    for i, poll_data in enumerate(created_polls):
        poll_id = poll_data["poll_id"]
        voter_account = new_voter()

        message       =       encode_defunct(text=f"{poll_id}-{0}-
{voter_account.address}")
        signed_message    =    w3.eth.account.sign_message(message,
private_key=voter_account.key)
        signature = signed_message.signature.hex()

        payload = {
            "poll_id": poll_id,
            "option_id": 0,
            "voter": voter_account.address,
            "signature": signature
```

```
        }

        response = client.post("/relay-vote", json=payload)

        if response.status_code != 200:
            print(f"[‼] Ошибка голосования за опрос #{poll_id}:
{response.json()}")
            assert False, f"Failed to vote for poll {poll_id}"

        print(f"✅ Голос за опрос #{poll_id} от
{voter_account.address} успешно передан.")

        # Проверяем, что токен был начислен
        balance_response =
client.get(f"/balance/{voter_account.address}")
        assert balance_response.status_code == 200
        balance_data = balance_response.json()

        expected_balance = 10 * (i + 1) # 10 за каждый голос
        assert int(balance_data["balance"]) ==
w3.to_wei(expected_balance, "ether"), f"Expected {expected_balance}
POLL, got {balance_data['balance']}"
        print(f"✅ Баланс {voter_account.address}:
{balance_data['balance']} POLL")

        # Проверим, что пользователь не может голосовать дважды
        response = client.post("/relay-vote", json=payload)
        assert response.status_code == 400
        assert "Already voted" in response.json()["detail"]


def test_poll_end_time():
    question = "Test poll end time?"
    response = client.post("/create-poll", json={
        "question": question,
        "options": ["Option A", "Option B"],
        "duration": 5
    })
    assert response.status_code == 200
    created = response.json()
    poll_id = created["poll_id"]
    end_time = created["end_time"]

    # Перемотка времени вперед
    w3.provider.make_request("evm_setNextBlockTimestamp", [end_time
+ 10])
    w3.provider.make_request("evm_mine", [])

    current_time = w3.eth.get_block("latest")["timestamp"]
    print(f"[DEBUG] Current block time: {current_time}, poll end
time: {end_time}")
    assert current_time > end_time
```

```
    # Пробуем проголосовать после окончания опроса
    voter_account = Account.create()
    message = encode_defunct(text=f"{poll_id}-{0}-
{voter_account.address}")
    signed_message = w3.eth.account.sign_message(message,
private_key=voter_account.key)
    signature = signed_message.signature.hex()

    payload = {
        "poll_id": poll_id,
        "option_id": 0,
        "voter": voter_account.address,
        "signature": signature
    }
    response = client.post("/relay-vote", json=payload)
    assert response.status_code == 400
    assert "Poll ended" in response.json()["detail"]

    # Сбрасываем время для других тестов
    w3.provider.make_request("evm_setNextBlockTimestamp",
[int(datetime.now(timezone.utc).timestamp())])
    w3.provider.make_request("evm_mine", [])


def test_vote_before_end_time():
    question = "Test vote before end time?"
    response = client.post("/create-poll", json={
        "question": question,
        "options": ["Option A", "Option B"],
        "duration": 300
    })
    assert response.status_code == 200
    created = response.json()
    poll_id = created["poll_id"]
    end_time = created["end_time"]

    voter_account = Account.create()

    message = encode_defunct(text=f"{poll_id}-{0}-
{voter_account.address}")
    signed_message = w3.eth.account.sign_message(message,
private_key=voter_account.key)
    signature = signed_message.signature.hex()

    payload = {
        "poll_id": poll_id,
        "option_id": 0,
        "voter": voter_account.address,
        "signature": signature
    }

    # Убедимся, что текущее время меньше времени окончания опроса
```

```
    selected = client.get(f"/polls").json()[0]
    poll_id = selected["id"]
    end_time = selected["end_time"]

    vote_time = end_time - 10
    w3.provider.make_request("evm_setNextBlockTimestamp",
[vote_time])
    w3.provider.make_request("evm_mine", [])

    current_time = w3.eth.get_block("latest")["timestamp"]
    print(f"[DEBUG] Current block time: {current_time}, poll end
time: {end_time}")
    assert current_time < end_time


def test_create_user_manually():
    session = SessionLocal()
    test_wallet = "0xAbcdef1234567890000000000000000000dEaD"

    session.query(User).filter_by(address=test_wallet).delete()
    session.commit()

    new_user = User(address=test_wallet)
    session.add(new_user)
    session.commit()

    u = session.query(User).filter_by(address=test_wallet).first()
    assert u is not None
    assert u.polls_created == 0
    assert u.votes_cast == 0
    session.close()


def test_get_balance():
    response = client.get(f"/balance/{ADMIN}")

    if response.status_code != 200:
        print(f"[‼️] Ошибка получения баланса для {ADMIN}:
{response.json()}")
        assert False, "Failed to get admin balance"

    balance_data = response.json()
    print(f"✅ Баланс ADMIN ({ADMIN}): {balance_data['balance']}
{balance_data['symbol']}")
    assert int(balance_data["balance"]) > 0, "Admin balance should
be greater than 0"
```

Листинг А5 - web3_setup.py

```
import os
from dotenv import load_dotenv
```

```
from web3 import Web3
from eth_account import Account

from contract_abi import POLL_SYSTEM_ABI, ERC20_ABI, DEPLOY_INFO

load_dotenv()

# Настройки из .env
RPC_URL = os.getenv("RPC_URL", "http://127.0.0.1:8545")
PRIVATE_KEY = os.getenv("PRIVATE_KEY")
SYSTEM_ADDRESS = os.getenv("CONTRACT_ADDRESS") or
DEPLOY_INFO["pollSystemAddress"]
TOKEN_ADDRESS = os.getenv("POLL_TOKEN_ADDRESS") or
DEPLOY_INFO["pollTokenAddress"]


w3 = Web3(Web3.HTTPProvider(RPC_URL))
account = Account.from_key(PRIVATE_KEY)
ADMIN = account.address


poll_system = w3.eth.contract(address=SYSTEM_ADDRESS,
abi=POLL_SYSTEM_ABI)
poll_token = w3.eth.contract(address=TOKEN_ADDRESS, abi=ERC20_ABI)

print(f"✅ Connected as: {ADMIN}")
print(f"📦 System Contract: {SYSTEM_ADDRESS}")
print(f"💰 Token Contract: {TOKEN_ADDRESS}")
```

Листинг А6 - contract_abi.py

```
import os
import json

BASE_DIR = os.path.dirname(__file__)
DEPLOYMENTS_DIR = os.path.abspath(os.path.join(BASE_DIR,
"../deployments"))

with open(os.path.join(DEPLOYMENTS_DIR, "poll_system_abi.json")) as
f:
    POLL_SYSTEM_ABI = json.load(f)

with open(os.path.join(DEPLOYMENTS_DIR, "poll_token_abi.json")) as
f:
    ERC20_ABI = json.load(f)

with open(os.path.join(DEPLOYMENTS_DIR, "deploy.json")) as f:
    DEPLOY_INFO = json.load(f)
```

Листинг А7 - main.py

```
from fastapi import FastAPI, HTTPException
from datetime import datetime, timezone
from web3 import Web3
from eth_account.messages import encode_defunct
from fastapi.middleware.cors import CORSMiddleware
from  web3_setup  import  w3,  poll_system,  poll_token,  ADMIN,
PRIVATE_KEY
from models import SessionLocal, User, Poll, Vote
from schemas import CreatePoll, RelayVoteRequest
from fastapi import APIRouter, Body
from sqlalchemy.orm import Session
from fastapi import Depends
from sqlalchemy.exc import SQLAlchemyError
from eth_utils import from_wei

app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

def get_contract_poll(poll_id: int):
    try:
        question,          options,          end_time_raw          =
poll_system.functions.getPoll(poll_id).call()
        end_time = int(end_time_raw)

        try:
            votes = poll_system.functions.getVotes(poll_id).call()
        except Exception as e:
            print(f"⚠️ Голоса не получены для poll #{poll_id}: {e}")
            votes = None

        return {
            "id": poll_id,
            "question": question,
            "options": options,
            "end_time": end_time,
            "votes": votes,
        }
    except Exception as e:
        print(f"❌ Ошибка при получении poll #{poll_id}: {e}")
        return None


@app.get("/health")
def health_check():
```

```
    return {
        "status": "ok",
        "block": w3.eth.block_number,
        "admin": ADMIN
    }



@app.get("/leaderboard")
def get_leaderboard():
    session = SessionLocal()
    try:
        users = session.query(User).all()
        result = []

        for user in users:
            try:
                checksum = Web3.to_checksum_address(user.address)
                balance_raw                                          =
poll_token.functions.balanceOf(checksum).call()
                decimals = poll_token.functions.decimals().call()
                balance = balance_raw // (10 ** decimals)

                result.append({
                    "address": checksum,
                    "balance": balance
                })
            except Exception as e:
                print(f"⚠️   Не   удалось   получить   баланс   для
{user.address}: {e}")
                continue

        sorted_result = sorted(result, key=lambda x: x["balance"],
reverse=True)
        return sorted_result[:10]
    finally:
        session.close()


@app.get("/balance/{address}")
def get_balance(address: str):
    try:
        if not Web3.is_address(address):
            raise  HTTPException(status_code=400,  detail="Неверный
адрес кошелька")

        checksum = Web3.to_checksum_address(address)
        print("👉 Чекаем токен для адреса:", checksum)

        balance = poll_token.functions.balanceOf(checksum).call()
        decimals = poll_token.functions.decimals().call()
        symbol = poll_token.functions.symbol().call()
```

```
        print("✅ Баланс:", balance, symbol, "Decimals:", decimals)

        return {
            "address": checksum,
            "balance": str(balance),
            "symbol": symbol,
            "decimals": decimals
        }
    except Exception as e:
        import traceback
        traceback.print_exc()
        raise     HTTPException(status_code=500,     detail="Ошибка
получения баланса")


@app.get("/polls")
def get_polls():
    try:
        ids = poll_system.functions.getAllPollIds().call()
        return [get_contract_poll(pid) for pid in ids]
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))


@app.post("/create-poll")
def create_poll(req: CreatePoll):
    try:
        # Проверяем баланс ADMIN для оплаты комиссии
        fee = w3.to_wei(100, "ether") # 100 POLL
        balance = poll_token.functions.balanceOf(ADMIN).call()
        if balance < fee:
            raise HTTPException(status_code=400, detail="Not enough
POLL tokens")

        # Проверяем, достаточно ли approve для PollSystem
        allowance      =      poll_token.functions.allowance(ADMIN,
poll_system.address).call()
        if allowance < fee:
            # Если недостаточно, отправляем транзакцию approve
            nonce = w3.eth.get_transaction_count(ADMIN)
            approve_tx                                          =
poll_token.functions.approve(poll_system.address,
fee).build_transaction({
                "from": ADMIN,
                "nonce": nonce,
                "gasPrice": w3.to_wei("20", "gwei")
            })
            signed_approve_tx                                   =
w3.eth.account.sign_transaction(approve_tx, PRIVATE_KEY)
            approve_tx_hash                                     =
w3.eth.send_raw_transaction(signed_approve_tx.rawTransaction)
            w3.eth.wait_for_transaction_receipt(approve_tx_hash)
```

```
            print(f"✅ Approved {from_wei(fee, 'ether')} POLL for
PollSystem contract.")


        # Создание опроса через смарт-контракт
        nonce = w3.eth.get_transaction_count(ADMIN)
        tx = poll_system.functions.createPoll(
            req.question,
            req.options,
            req.duration
        ).build_transaction({
            "from": ADMIN,
            "nonce": nonce,
            "gasPrice": w3.to_wei("20", "gwei")
        })

        signed_tx       =       w3.eth.account.sign_transaction(tx,
PRIVATE_KEY)
        tx_hash                                                    =
w3.eth.send_raw_transaction(signed_tx.rawTransaction)
        receipt = w3.eth.wait_for_transaction_receipt(tx_hash)

        poll_id = None
        for log in receipt.logs:
            try:
                #   Попробуем   декодировать   логи,   если   они
соответствуют событию PollCreated
                decoded_log                                        =
poll_system.events.PollCreated().process_receipt({"logs": [log]})
                if decoded_log and len(decoded_log) > 0:
                    poll_id = decoded_log[0]["args"]["pollId"]
                    break
            except Exception as e:
                # Если лог не соответствует событию PollCreated,
просто пропускаем его
                continue

        if poll_id is None:
            # Fallback для получения poll_id, если событие не было
декодировано (например, в dev-сетях)
            # В реальной сети лучше полагаться на события
            current_poll_count                                     =
poll_system.functions.pollCount().call()
            # Предполагаем, что новый опрос имеет ID = текущему
pollCount до его инкремента
            poll_id = current_poll_count -1 if current_poll_count >
0 else 0


        session = SessionLocal()
        user = session.get(User, ADMIN)
        if not user:
```

```
            user = User(address=ADMIN, polls_created=0,
votes_cast=0)
            session.add(user)
        user.polls_created = (user.polls_created or 0) + 1 #
инкремент счетчика

        new_poll = Poll(
            id=poll_id,
            question=req.question,
            options=req.options,

end_time=datetime.fromtimestamp(w3.eth.get_block(receipt.blockNumb
er).timestamp + req.duration, timezone.utc),
            creator_address=ADMIN
        )
        session.add(new_poll)
        session.commit()
        session.close()

        return {
            "tx_hash": tx_hash.hex(),
            "poll_id": poll_id,
            "question": req.question,
            "options": req.options,
            "duration": req.duration,
            "end_time": new_poll.end_time.timestamp()
        }
    except SQLAlchemyError as e:
        raise HTTPException(status_code=500, detail=f"Database
error: {e}")
    except Exception as e:
        import traceback
        traceback.print_exc()
        raise HTTPException(status_code=500, detail=str(e))


@app.post("/relay-vote")
def relay_vote(req: RelayVoteRequest):
    try:
        # 1. Проверить, что опрос существует и не завершен
        poll_info = get_contract_poll(req.poll_id)
        if not poll_info:
            raise HTTPException(status_code=404, detail="Poll not
found or invalid")

        if datetime.now(timezone.utc).timestamp() >
poll_info["end_time"]:
            raise HTTPException(status_code=400, detail="Poll
ended")

        # 2. Проверить, что пользователь еще не голосовал
        has_voted = poll_system.functions.hasVoted(req.poll_id,
Web3.to_checksum_address(req.voter)).call()
```

```
        if has_voted:
            raise  HTTPException(status_code=400,  detail="Already
voted")

        # 3. Отправить транзакцию через relayer
        nonce = w3.eth.get_transaction_count(ADMIN)
        tx = poll_system.functions.voteWithSignature(
            req.poll_id,
            req.option_id,
            Web3.to_checksum_address(req.voter),
            bytes.fromhex(req.signature[2:]) # Убираем "0x" префикс
        ).build_transaction({
            "from": ADMIN,
            "nonce": nonce,
            "gasPrice": w3.to_wei("20", "gwei")
        })

        signed_tx      =      w3.eth.account.sign_transaction(tx,
PRIVATE_KEY)
        tx_hash                                                    =
w3.eth.send_raw_transaction(signed_tx.rawTransaction)
        receipt = w3.eth.wait_for_transaction_receipt(tx_hash)

        tx_info = w3.eth.get_transaction(tx_hash)

        gas_used = receipt["gasUsed"]
        gas_price = tx_info["gasPrice"]
        total_fee = gas_used * gas_price
        tx_fee = gas_used * gas_price

        print(f"""
            ✅ Транзакция подтверждена:
            🔗 TxHash: {tx_hash.hex()}
            📦 Gas Used: {gas_used}
            💵 Gas Price: {from_wei(gas_price, 'gwei')} Gwei
            💸 Total Fee: {from_wei(tx_fee, 'ether')} ETH
            ⏰ Time: {datetime.utcnow()}
        """)

        session = SessionLocal()
        user = session.get(User, req.voter)

        if not user:
            user    =    User(address=req.voter,    polls_created=0,
votes_cast=0)
            session.add(user)

        user.votes_cast = (user.votes_cast or 0) + 1

        vote = Vote(
            poll_id=req.poll_id,
            option_id=req.option_id,
```

```
            user_address=req.voter,
            tx_hash=tx_hash.hex()
        )

        session.add(vote)
        session.commit()
        session.close()

        return {
            "tx_hash": tx_hash.hex(),
            "gas_used": gas_used,
            "total_fee_eth": from_wei(tx_fee, 'ether')
        }

    except HTTPException as e:
        raise e
    except Exception as e:
        import traceback
        traceback.print_exc()
        raise HTTPException(status_code=500, detail=str(e))
```

Листинг А8 - models.py

```
from sqlalchemy import (
    create_engine,
    Column,
    Integer,
    String,
    DateTime,
    ForeignKey,
    JSON,
    Text
)
from sqlalchemy.orm import declarative_base, relationship,
sessionmaker
from datetime import datetime, timezone

Base = declarative_base()

#(user = wallet)
class User(Base):
    __tablename__ = "users"

    address = Column(String(42), primary_key=True)
    balance = Column(Integer, default=0)
    polls_created = Column(Integer, default=0)
    votes_cast = Column(Integer, default=0)
    votes = relationship("Vote", back_populates="user")
    polls = relationship("Poll", back_populates="author")

class Poll(Base):
    __tablename__ = "polls"
```

```
    id = Column(Integer, primary_key=True)
    question = Column(Text)
    options = Column(JSON)
    end_time = Column(DateTime)
    created_at       =       Column(DateTime,       default=lambda:
datetime.now(timezone.utc))

    creator_address              =              Column(String(42),
ForeignKey("users.address"))
    author = relationship("User", back_populates="polls")
    votes = relationship("Vote", back_populates="poll")

class Vote(Base):
    __tablename__ = "votes"

    id = Column(Integer, primary_key=True)
    poll_id = Column(Integer, ForeignKey("polls.id"))
    user_address = Column(String(42), ForeignKey("users.address"))
    option_id = Column(Integer)
    tx_hash = Column(String(66), unique=True, nullable=False) # Хэш
транзакции, 66 символов (0x + 64)

    created_at       =       Column(DateTime,       default=lambda:
datetime.now(timezone.utc))

    user = relationship("User", back_populates="votes")
    poll = relationship("Poll", back_populates="votes")

# Создание базы данных и таблиц
#SQLALCHEMY_DATABASE_URL = "sqlite:///./sql_app.db" # Для локальной
разработки
SQLALCHEMY_DATABASE_URL  =  "sqlite:///./data/database.db"  #  Для
Docker контейнера

engine = create_engine(
    SQLALCHEMY_DATABASE_URL,     connect_args={"check_same_thread":
False}
)
Base.metadata.create_all(bind=engine)

SessionLocal  =  sessionmaker(autocommit=False,  autoflush=False,
bind=engine)
```

Листинг А9 - index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" href="/vite.svg" />
```

```
    <meta  name="viewport"  content="width=device-width,  initial-
scale=1.0" />
    <title>Smart Voting DApp</title>

    <!-- 🎩 Retro шрифт -->
    <link

href="https://fonts.googleapis.com/css2?family=Courier+Prime&displ
ay=swap"
      rel="stylesheet"
    />
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.tsx"></script>
  </body>
</html>
```

Листинг А10 - main.tsx

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import "./index.css";

ReactDOM.createRoot(document.getElementById("root")!).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

declare global {
  interface ImportMetaEnv {
    readonly VITE_BACKEND_API_URL: string;
    readonly VITE_POLL_SYSTEM_ADDRESS: string;
    readonly VITE_POLL_TOKEN_ADDRESS: string;
  }

  interface ImportMeta {
    readonly env: ImportMetaEnv;
  }
}
```

Листинг А11 - App.tsx

```
import { BrowserRouter as Router, Routes, Route, Link } from "react-
router-dom";
import HomePage from "./pages/Home";
import CreatePage from "./pages/Create";
import LeaderboardPage from "./components/Leaderboard";
import InfoPage from "./pages/Info";
import UserPanel from "./components/UserPanel"; // 👈 Новая панель
пользователя
```

```
export default function App() {
  return (
    <Router>
      <div   className="flex   h-screen   bg-typewriter-bg   text-
typewriter-text font-mono select-none">
        <aside className="w-52 border-r border-typewriter-border p-
4">
          <h1 className="text-xl mb-6">🧱 DApp</h1>
          <nav className="space-y-2 text-lg">
            <Link   to="/"   className="block   hover:underline">🏠
Главная</Link>
            <Link to="/create" className="block hover:underline">➕
Добавить</Link>
            <Link        to="/leaderboard"        className="block
hover:underline">🏆 Лидерборд</Link>
            <Link  to="/info"  className="block  hover:underline">❓
Информация</Link>
          </nav>
        </aside>

        <main className="flex-1 p-6 overflow-y-auto">
          <Routes>
            <Route path="/" element={<HomePage />} />
            <Route path="/create" element={<CreatePage />} />
            <Route  path="/leaderboard"  element={<LeaderboardPage
/>} />
            <Route path="/info" element={<InfoPage />} />
          </Routes>
        </main>

        <aside className="w-64 border-l border-typewriter-border p-
4">
          <UserPanel />
        </aside>
      </div>
    </Router>
  );
}
```

Листинг А12 - CreatePollForm.tsx

```
import React, { useState, useEffect } from 'react';
import { ethers } from 'ethers';

import                      PollSystemABI                      from
'../../../deployments/poll_system_abi.json';
import                      PollTokenJson                      from
'../../../deployments/poll_token_abi.json';
```

```
const PollTokenABI: any[] = (PollTokenJson as { abi?: any[] }).abi ??
(PollTokenJson as any[]);
const                      POLL_SYSTEM_ADDRESS                      =
import.meta.env.VITE_POLL_SYSTEM_ADDRESS!;
const                       POLL_TOKEN_ADDRESS                      =
import.meta.env.VITE_POLL_TOKEN_ADDRESS!;
const  POLL_CREATION_FEE  =  ethers.utils.parseEther("100");  //  100
POLL

export default function CreatePollFormMetaMask() {
  const [question, setQuestion] = useState("");
  const [options, setOptions] = useState<string[]>(["", ""]);
  const [duration, setDuration] = useState(600);
  const [status, setStatus] = useState<string | null>(null);
  const [loading, setLoading] = useState(false);
  const [faucetLoading, setFaucetLoading] = useState(false);
  const  [contractOwner,  setContractOwner]  =  useState<string  |
null>(null);
  const   [currentUser,   setCurrentUser]   =   useState<string   |
null>(null);

  const initUser = async () => {
  if (!window.ethereum) return;
  const                provider                =                new
ethers.providers.Web3Provider(window.ethereum);
  const signer = provider.getSigner();
  const address = await signer.getAddress();
  setCurrentUser(address);
};

  useEffect(() => {
    checkContractOwner();
  }, []);

  useEffect(() => {
  initUser();
}, []);

  useEffect(() => {
  if (!window.ethereum) return;


  const handleAccountsChanged = async (accounts: string[]) => {
    if (accounts.length === 0) {
      console.warn("🦊 MetaMask аккаунт не выбран");
      setCurrentUser(null);
    } else {
      const newAddress = accounts[0];
      setCurrentUser(newAddress);

      try {
```

```
        const provider = new
ethers.providers.Web3Provider(window.ethereum);
        const pollToken = new ethers.Contract(POLL_TOKEN_ADDRESS,
PollTokenABI, provider);
        const balance: ethers.BigNumber = await
pollToken.balanceOf(newAddress);
        console.log("👉 Новый баланс:",
ethers.utils.formatEther(balance));
      } catch (err) {
        console.error("Ошибка обновления баланса:", err);
      }
    }
  };

  window.ethereum.on('accountsChanged', handleAccountsChanged);

  return () => {
    window.ethereum?.removeListener('accountsChanged',
handleAccountsChanged);
  };
}, []);

  const checkContractOwner = async () => {
    try {
      if (!window.ethereum) return;

      const provider = new
ethers.providers.Web3Provider(window.ethereum);
      const contract = new ethers.Contract(
        POLL_TOKEN_ADDRESS,
        [...PollTokenABI, "function owner() external view returns
(address)"],
        provider
      );

      const owner = await contract.owner();
      setContractOwner(owner);
      return owner;
    } catch (err) {
      console.error("Ошибка при проверке владельца:", err);
      setStatus("⚠️ Контракт не имеет функции owner()");
      return null;
    }
  };

  const handleFaucet = async () => {
    setFaucetLoading(true);
    setStatus("⏳ Запрос тестовых токенов...");

    try {
      if (!window.ethereum) throw new Error("🦊 Установите
MetaMask");
```

```
      const provider = new
ethers.providers.Web3Provider(window.ethereum);
      await provider.send("eth_requestAccounts", []);
      const signer = provider.getSigner();
      const userAddress = await signer.getAddress();
      setCurrentUser(userAddress);

      const pollToken = new ethers.Contract(POLL_TOKEN_ADDRESS,
PollTokenABI, signer);

      setStatus("✍️ Подтвердите транзакцию в MetaMask...");
      const tx = await pollToken.mint(userAddress,
ethers.utils.parseEther("1000"));
      setStatus(`⛓️ Ожидание подтверждения... (${tx.hash})`);
      await tx.wait();

      setStatus("✅ Получено 1000 тестовых POLL токенов!");
    } catch (err: any) {
      console.error("💥 Ошибка:", err);
      setStatus(`❌ Ошибка: ${err.reason || err.message || "Не
удалось получить токены"}`);
    } finally {
      setFaucetLoading(false);
    }
  };

  const addOption = () => setOptions([...options, ""]);
  const handleOptionChange = (i: number, value: string) => {
    const updated = [...options];
    updated[i] = value;
    setOptions(updated);
  };
  const removeOption = (index: number) => {
    if (options.length <= 2) return;
    setOptions(options.filter((_, i) => i !== index));
  };

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    setStatus("⏳ Проверка MetaMask...");
    setLoading(true);

    try {
      if (!window.ethereum) throw new Error("🦊 Установите
MetaMask");

      const provider = new
ethers.providers.Web3Provider(window.ethereum);
      await provider.send("eth_requestAccounts", []);
      const signer = provider.getSigner();
```

```
      const userAddress = await signer.getAddress();
      setCurrentUser(userAddress);

      const pollToken = new ethers.Contract(POLL_TOKEN_ADDRESS,
PollTokenABI, signer);
      const pollSystem = new ethers.Contract(POLL_SYSTEM_ADDRESS,
PollSystemABI, signer);

      // Проверка баланса
      const balance: ethers.BigNumber = await
pollToken.balanceOf(userAddress);
      if (balance.lt(POLL_CREATION_FEE)) {
        throw new Error(`❌ Недостаточно POLL. Нужно 100, у вас
${ethers.utils.formatEther(balance)}`);
      }

      // Проверка allowance
      const allowance: ethers.BigNumber = await
pollToken.allowance(userAddress, POLL_SYSTEM_ADDRESS);
      if (allowance.lt(POLL_CREATION_FEE)) {
        setStatus("✍️ Подтвердите разрешение (approve) в
MetaMask...");
        const tx = await pollToken.approve(POLL_SYSTEM_ADDRESS,
POLL_CREATION_FEE);
        await tx.wait();
        setStatus("✅ Разрешение успешно выдано");
      }

      // Создание опроса
      setStatus("📦 Подтвердите создание опроса в MetaMask...");
      const tx = await pollSystem.createPoll(question, options,
duration);
      setStatus(`⛓️ Ожидание подтверждения транзакции...
(${tx.hash})`);
      await tx.wait();
      setStatus(`✅ Опрос создан! Tx: ${tx.hash}`);

      // Очистка формы
      setQuestion("");
      setOptions(["", ""]);
      setDuration(600);
    } catch (err: any) {
      console.error("💥 Ошибка:", err);
      setStatus(`❌ Ошибка: ${err.reason || err.message || "Что-то
пошло не так"}`);
    } finally {
      setLoading(false);
    }
  };

  const hasMintFunction = PollTokenABI.some(x => x.name === "mint");
```

```
  return (
    <form onSubmit={handleSubmit} className="space-y-4 border p-6
rounded-md max-w-xl mx-auto bg-white text-black">
      <h2 className="text-xl font-bold">📋 Новый опрос (через
MetaMask)</h2>

      <div className="bg-gray-50 p-3 rounded-md mb-4">
        <div className="flex justify-between items-center mb-2">
          <h3 className="font-medium">Информация о контракте</h3>
          <button
            type="button"
            onClick={checkContractOwner}
            className="text-xs bg-gray-200 px-2 py-1 rounded"
          >
            🔄 Обновить
          </button>
        </div>

        {contractOwner && (
          <p className="text-sm break-all">
            <span                 className="font-semibold">Владелец
контракта:</span> {contractOwner}
          </p>
        )}

        {currentUser && (
          <p className="text-sm break-all mt-1">
            <span    className="font-semibold">Ваш    адрес:</span>
{currentUser}
          </p>
        )}

        {hasMintFunction && (
          <button
            type="button"
            onClick={handleFaucet}
            disabled={faucetLoading}
            className="mt-3 w-full px-3 py-2 bg-green-500 text-white
rounded disabled:opacity-40"
          >
            {faucetLoading  ?  "⏳  Обработка..."  :  "🌑  Получить
тестовые токены (1000 POLL)"}
          </button>
        )}
      </div>

      <div>
        <label className="block mb-1">📝 Вопрос:</label>
        <input
          className="w-full border p-2 rounded"
          type="text"
```

```
          value={question}
          onChange={(e) => setQuestion(e.target.value)}
          required
        />
      </div>

      <div>
        <label className="block mb-1">📝 Варианты:</label>
        {options.map((opt, i) => (
          <div key={i} className="flex gap-2 mb-1">
            <input
              className="flex-1 border p-2 rounded"
              value={opt}
              onChange={(e)        =>        handleOptionChange(i,
e.target.value)}
              required
            />
            {options.length > 2 && (
              <button
                type="button"
                onClick={() => removeOption(i)}
                className="text-red-500 px-2"
              >
                ✕
              </button>
            )}
          </div>
        ))}
        <button
          type="button"
          onClick={addOption}
          className="text-blue-600 text-sm mt-1"
        >
          ➕ Добавить вариант
        </button>
      </div>

      <div>
        <label    className="block    mb-1">⏱️    Длительность    (в
секундах):</label>
        <input
          type="number"
          className="border p-2 w-full rounded"
          value={duration}
          onChange={(e) => setDuration(Number(e.target.value))}
          required
          min={1}
        />
      </div>

      <button
        type="submit"
```

```
        disabled={loading}
        className="w-full px-4 py-2 bg-black text-white rounded
disabled:opacity-40"
      >
        {loading ? "⏳ Обработка..." : "🚀 Создать опрос"}
      </button>

      {status && (
        <div className="mt-4 p-3 bg-gray-50 rounded border border-
gray-200">
          <p className="whitespace-pre-wrap text-sm">{status}</p>
        </div>
      )}
    </form>
  );
}
```

Листинг A13 - PollList.tsx

```
import React, { useEffect, useState } from "react";
import VoteButton from "./VoteButton";
import { ethers } from "ethers";

type Poll = {
  id: number;
  question: string;
  options: string[];
  end_time: number;
  votes?: number[];
};

export default function PollList({ search = "" }: { search?:
string }) {
  const [polls, setPolls] = useState<Poll[]>([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);

  const [votedPolls, setVotedPolls] = useState<Set<number>>(new
Set());
  const [currentAddress, setCurrentAddress] = useState<string |
null>(null);

  useEffect(() => {
    const fetchAddress = async () => {
      if (window.ethereum) {
        const provider = new
ethers.providers.Web3Provider(window.ethereum);
        const accounts = await provider.send("eth_requestAccounts",
[]);
        setCurrentAddress(accounts[0]);
```

```
        window.ethereum.on("accountsChanged", (accounts: string[])
=> {
          setCurrentAddress(accounts[0]);
          window.dispatchEvent(new Event("vote-recorded"));
        });
      }
    };

    fetchAddress();

    return () => {
      window.ethereum?.removeAllListeners("accountsChanged");
    };
  }, []);

  // Загружаем опросы
  useEffect(() => {
    const fetchPolls = async () => {
      try {
        const res = await
fetch(`${import.meta.env.VITE_BACKEND_API_URL}/polls`);
        const data = await res.json();
        const validPolls = Array.isArray(data) ? data.filter((p) =>
p?.id !== undefined) : [];
        setPolls(validPolls);
      } catch (err) {
        setError("❌ Не удалось загрузить опросы");
        console.error(err);
      } finally {
        setLoading(false);
      }
    };

    fetchPolls();
  }, []);

  // Проверяем, за какие опросы голосовал текущий адрес
  useEffect(() => {
    const checkVotes = () => {
      const set = new Set<number>();
      if (!currentAddress) return;

      for (let i = 0; i < localStorage.length; i++) {
        const key = localStorage.key(i);
        if (!key) continue; // Пропускаем, если ключ отсутствует

        const match = key.match(/^voted_(\d+)_/);
        if (match && key.includes(currentAddress)) {
          set.add(parseInt(match[1]));
        }
      }
```

```
      setVotedPolls(set);
    };

    checkVotes();
    window.addEventListener("vote-recorded", checkVotes);
    return () => window.removeEventListener("vote-recorded",
checkVotes);
  }, [currentAddress]);

  const filtered = polls.filter((poll) =>
    poll.question.toLowerCase().includes(search.toLowerCase())
  );

  const sorted = [...filtered].sort((a, b) => b.id - a.id);

  if (loading) return <p>⏳ Загрузка опросов...</p>;
  if (error) return <p className="text-red-600">{error}</p>;
  if (filtered.length === 0) return <p>😕 Опросы не найдены</p>;

  return (
    <div className="space-y-6">
      {sorted.map((poll) => {
        const isFinished = Date.now() / 1000 > poll.end_time;
        const hasVoted = votedPolls.has(poll.id);

        return (
          <div
            key={poll.id}
            className={`border p-4 rounded transition-all ${
              isFinished || hasVoted ? "bg-gray-100 text-gray-500" :
"hover:bg-gray-50"
            }`}
          >
            <h3 className="text-lg font-bold mb-2">🗳️
{poll.question}</h3>

            <p className="text-sm mb-3">
              ⏰ Завершение:{" "}
              {new Date(poll.end_time * 1000).toLocaleString()}
            </p>

            {isFinished ? (
              <div>
                <p className="font-semibold text-green-600 mb-1">📊
Результаты:</p>
                <ul className="space-y-1">
                  {poll.options.map((opt, idx) => (
                    <li key={idx}>
                      ✅ {opt}: <strong>{poll.votes?.[idx] ??
0}</strong> голосов
                    </li>
```

```
                  ))}
                </ul>
              </div>
            ) : hasVoted ? (
              <p className="text-blue-600 font-semibold">
                ✅ Вы уже голосовали в этом опросе.
              </p>
            ) : (
              <ul className="space-y-3">
                {poll.options.map((opt, idx) => (
                  <li key={idx} className="flex items-center gap-
2">
                    <VoteButton pollId={poll.id} optionId={idx}
currentAddress={currentAddress} />
                    <span>{opt}</span>
                  </li>
                ))}
              </ul>
            )}
          </div>
        );
      })}
    </div>
  );
}
```

Листинг А12 - Leaderboard.tsx

```
import React, { useEffect, useState } from "react";

type Entry = {
  address: string;
  balance: number;
};

export default function Leaderboard() {
  const [topUsers, setTopUsers] = useState<Entry[]>([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const fetchLeaderboard = async () => {
      try {
        const res = await
fetch(`${import.meta.env.VITE_BACKEND_API_URL}/leaderboard`);
        const data = await res.json();
        setTopUsers(data);
      } catch (err) {
        console.error("Ошибка при получении лидерборда:", err);
      } finally {
        setLoading(false);
      }
    };
```

```
    fetchLeaderboard();
  }, []);

  return (
    <div className="max-w-xl mx-auto p-6 bg-white shadow rounded
space-y-4">
      <h2 className="text-2xl font-bold text-center">🏆 Лидерборд
POLL</h2>

      {loading ? (
        <p className="text-center">⏳ Загрузка...</p>
      ) : topUsers.length === 0 ? (
        <p className="text-center text-gray-500">😢 Пока никто не
попал в таблицу лидеров</p>
      ) : (
        <ol className="space-y-2 list-decimal list-inside">
          {topUsers.map((user, index) => (
            <li
              key={user.address}
              className="flex justify-between text-sm border-b pb-1
last:border-none"
            >
              <span className="font-mono text-gray-700">
                {user.address.slice(0,  6)}...{user.address.slice(-
4)}
              </span>
              <span                                    className="font-
semibold">{user.balance.toLocaleString()} POLL</span>
            </li>
          ))}
        </ol>
      )}
    </div>
  );
}
```

Листинг А14 - VoteButton.tsx

```
import React, { useEffect, useState } from "react";
import { ethers } from "ethers";

type VoteButtonProps = {
  pollId: number;
  optionId: number;
  currentAddress: string | null;
};

declare global {
  interface Window {
    ethereum?: any;
  }
```

```
}

const VoteButton: React.FC<VoteButtonProps> = ({ pollId, optionId,
currentAddress }) => {
  const [isLoading, setIsLoading] = useState(false);
  const [hasVoted, setHasVoted] = useState(false);

  useEffect(() => {
    if (!currentAddress) return;
    const key = `voted_${pollId}_${currentAddress}`;
    setHasVoted(!!localStorage.getItem(key));
  }, [pollId, currentAddress]);

  const showError = (message: string) => alert(`❌ ${message}`);
  const showSuccess = (message: string) => alert(`✅ ${message}`);

  const vote = async () => {
    if (hasVoted) {
      showError("Вы уже голосовали в этом опросе");
      return;
    }

    setIsLoading(true);

    try {
      if (!window.ethereum) {
        showError("🦊 Установите MetaMask для голосования");
        return;
      }

      const provider = new
ethers.providers.Web3Provider(window.ethereum);
      await provider.send("eth_requestAccounts", []);
      const signer = provider.getSigner();
      const voter = await signer.getAddress();

      const messageHash = ethers.utils.solidityKeccak256(
        ["uint256", "uint256", "address"],
        [pollId, optionId, voter]
      );

      const arrayifiedHash = ethers.utils.arrayify(messageHash);
      const signature = await signer.signMessage(arrayifiedHash);

      const response = await
fetch(`${import.meta.env.VITE_BACKEND_API_URL}/relay-vote`, {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({
          poll_id: pollId,
          option_id: optionId,
          voter,
```

```
          signature,
        }),
      });

      const data = await response.json();

      if (!response.ok) {
        const errorString = JSON.stringify(data);
        if (errorString.includes("Already voted")) {
          showError("Вы уже голосовали в этом опросе");
        } else {
          showError(data.detail || "Ошибка при отправке голоса");
        }
        return;
      }

      showSuccess(`Ваш голос за вариант #${optionId} принят!`);
      localStorage.setItem(`voted_${pollId}_${voter}`, "true");
      setHasVoted(true);
      window.dispatchEvent(new Event("vote-recorded"));
    } catch (error: any) {
      console.error("Ошибка голосования:", error);
      const msg = error?.message || "";
      if (msg.includes("Already voted")) {
        showError("Вы уже голосовали в этом опросе");
      } else if (msg.includes("User denied")) {
        showError("Вы отклонили MetaMask запрос");
      } else {
        showError("Произошла неизвестная ошибка");
      }
    } finally {
      setIsLoading(false);
    }
  };

  return (
    <button
      onClick={vote}
      disabled={isLoading || hasVoted}
      className={`px-4 py-2 rounded-md font-medium ${
        hasVoted
          ? "bg-gray-300 text-gray-600 cursor-not-allowed"
          : "bg-blue-600 text-white hover:bg-blue-700"
      } ${isLoading ? "opacity-70 cursor-wait" : ""} transition-
colors duration-200`}
    >
      {isLoading ? (
        <span className="flex items-center">
          <svg
            className="animate-spin -ml-1 mr-2 h-4 w-4 text-white"
            xmlns="http://www.w3.org/2000/svg"
            fill="none"
            viewBox="0 0 24 24"
```

```
          >
            <circle  className="opacity-25"  cx="12"  cy="12"  r="10"
stroke="currentColor" strokeWidth="4" />
            <path
              className="opacity-75"
              fill="currentColor"
              d="M4  12a8  8  0  018-8V0C5.373  0  0  5.373  0  12h4zm2
5.291A7.962 7.962 0 014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"
            />
          </svg>
          Обработка...
        </span>
      ) : hasVoted ? (
        "✓ Вы уже голосовали"
      ) : (
        `Голосовать за вариант #${optionId}`
      )}
    </button>
  );
};

export default VoteButton;
```

Листинг А15 - UserPanel.tsx

```
import React, { useEffect, useState } from "react";
import { ethers } from "ethers";

declare global {
  interface Window {
    ethereum?: any;
  }
}

export default function UserPanel() {
  const [address, setAddress] = useState<string | null>(null);
  const [balance, setBalance] = useState<number | null>(null);
  const [symbol, setSymbol] = useState<string>("TOKEN");

  useEffect(() => {
    const connectAndFetch = async () => {
      try {
        if (!window.ethereum) {
          console.warn("MetaMask is not installed");
          return;
        }

        const              provider              =              new
ethers.providers.Web3Provider(window.ethereum);
```

```
        await provider.send("eth_requestAccounts", []);
        const signer = provider.getSigner();
        const userAddress = await signer.getAddress();
        setAddress(userAddress);

        // Получение баланса с бэкенда
        const res = await
fetch(`${import.meta.env.VITE_BACKEND_API_URL}/balance/${userAddre
ss}`);
        if (!res.ok) {
          throw new Error("Ошибка получения баланса");
        }

        const data = await res.json();
        const divisor = Math.pow(10, data.decimals);
        setBalance(data.balance / divisor);
        setSymbol(data.symbol);
      } catch (err) {
        console.error("Ошибка подключения:", err);
      }
    };

    connectAndFetch();
  }, []);

  return (
    <div className="text-sm space-y-2">
      <h2 className="text-lg font-bold mb-2">👤 Пользователь</h2>
      {address ? (
        <>
          <p className="break-all">📫 {address}</p>
          <p>💰 Баланс: {balance?.toFixed(4)} {symbol}</p>
        </>
      ) : (
        <p>⏳ Ожидание MetaMask...</p>
      )}
    </div>
  );
}
```

Листинг А16 – Create.tsx Home.tsx Info.tsx

```
import CreatePollForm from "../components/CreatePollForm";

export default function Create() {
  return (
    <div>
      <h2 className="text-xl mb-4">Создать новый опрос</h2>
      <CreatePollForm />
    </div>
  );
}
```

```
import { useNavigate } from "react-router-dom";
import { useState } from "react";
import PollList from "../components/PollList";

export default function Home() {
  const [search, setSearch] = useState("");
  const navigate = useNavigate();

  return (
    <div>
      <div className="flex justify-between items-center mb-4">
        <h2 className="text-xl font-bold">📦 Лента опросов</h2>
        <button
          onClick={() => navigate("/create")}
          className="text-xl border border-typewriter-border px-3
py-1 hover:bg-typewriter-border/25"
        >
          ➕
        </button>
      </div>

      <div className="mb-4">
        <label className="block mb-1">🔍 Поиск по вопросу:</label>
        <input
          className="w-full border border-typewriter-border p-2 bg-
transparent"
          type="text"
          value={search}
          onChange={(e) => setSearch(e.target.value)}
          placeholder="Введите ключевые слова..."
        />
      </div>

      <PollList search={search} />
    </div>
  );
}

export default function Info() {
  return (
    <div>
      <h2 className="text-xl mb-4">❓ О проекте</h2>
      <p>
        Это децентрализованное приложение (DApp) для проведения
опросов.
        Вы можете голосовать, создавая подпись в MetaMask, абсолютно
бесплатно —
        транзакции за вас отправляет сервер.
      </p>
      <p className="mt-4">
```

```
        Создание опроса требует токенов POLL, которые вы получаете
за активность.
      </p>
    </div>
  );
}
```

Листинг А17 – run_project.sh

```
#!/bin/bash

set -e

echo "-----------------------------------------"
echo "1. Настройка бэкенда (Hardhat)"
echo "-----------------------------------------"

echo "🔧 Настройка бэкенда (Hardhat)..."

# Установка Node.js 20 через NVM
if ! command -v node &> /dev/null || [[ $(node -v) != v20* ]]; then
    echo "Установка Node.js 20..."
    curl         -o-         https://raw.githubusercontent.com/nvm-
sh/nvm/v0.39.5/install.sh | bash
    export NVM_DIR="$HOME/.nvm"
    [ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
    nvm install 20
    nvm use 20
fi

# Установка Hardhat и зависимостей
if [ ! -d "node_modules" ]; then
    echo "Установка зависимостей Hardhat..."
    npm install --save-dev hardhat @nomicfoundation/hardhat-toolbox
fi

# Запуск локальной ноды Hardhat в фоне
echo "🚀 Запуск Hardhat ноды..."
npx hardhat node &
HARDHAT_PID=$!
sleep 5  # Даем время ноде запуститься

# Деплой контрактов
echo "📦 Деплой контрактов..."
npx hardhat run scripts/deploy.js --network localhost

echo "-----------------------------------------"
echo "2. Настройка Python бэкенда (FastAPI)"
echo "-----------------------------------------"

cd backend || { echo "❌ Папка backend не найдена"; exit 1; }
```

```
# Проверка наличия Python 3
if ! command -v python3 &> /dev/null; then
    echo "❌ Python 3 не установлен!"
    exit 1
fi

# Создание виртуального окружения, если не существует
if [ ! -d ".venv" ]; then
    echo "📦 Создание виртуального окружения..."
    python3 -m venv .venv
fi

# Активация виртуального окружения
source .venv/bin/activate

# Установка / обновление pip
echo "⚙️ Обновление pip..."
python -m pip install --upgrade pip > /dev/null

# Установка зависимостей (или из requirements.txt при наличии)
if [ -f "requirements.txt" ]; then
    echo "⬇️ Установка зависимостей из requirements.txt..."
    pip install -r requirements.txt > /dev/null
else
    echo "⬇️ Установка Uvicorn и FastAPI..."
    pip install uvicorn fastapi > /dev/null
fi
# Запуск Python-бэкенда
if [ -f "main.py" ]; then
    echo "🌐 Запуск Python бэкенда (Uvicorn)..."
    uvicorn main:app --host 0.0.0.0 --port 8000 &
    UVICORN_PID=$!
else
    echo "⚠️ Файл main.py не найден! Пропускаем запуск Python
бэкенда."
    UVICORN_PID=""
fi

cd ..

echo "-------------------------------------------"
echo "3. Настройка и запуск фронтенда"
echo "-------------------------------------------"

if [ -d "frontend" ]; then
    cd frontend || exit

    # Установка зависимостей
    if [ ! -d "node_modules" ]; then
        npm install
    fi
```

```
    # Запуск фронтенда
    echo "⚡ Запуск фронтенда..."
    npm run dev &
    FRONTEND_PID=$!
    cd ..
else
    echo "⚠️ Папка frontend не найдена, пропуск..."
    FRONTEND_PID=""
fi

echo "-----------------------------------------"
echo "✅ Все компоненты запущены!"
echo "-----------------------------------------"
echo "Hardhat нода:    http://localhost:8545"
echo "Python бэкенд:   http://localhost:8000"
echo "Фронтенд:        http://localhost:5173"
echo "-----------------------------------------"
echo "Для остановки: нажмите Ctrl+C"

# Завершение по Ctrl+C
trap "echo '■ Остановка процессов...'; kill $HARDHAT_PID
$UVICORN_PID $FRONTEND_PID 2>/dev/null; exit 0" SIGINT

# Ожидание завершения
wait
```

Листинг А18 – deploy.js

```
const { ethers, artifacts, network } = require("hardhat");
const fs = require("fs");
const path = require("path");

async function main() {
  const [deployer] = await ethers.getSigners();
  console.log(`👤 Deployer: ${deployer.address}`);
  console.log(`💰     Balance:     ${ethers.utils.formatEther(await
deployer.getBalance())} ETH`);

  // Deploy token
  const PollToken = await ethers.getContractFactory("PollToken");
  const token = await PollToken.deploy();
  await token.deployed();
  console.log(`✅ PollToken deployed at: ${token.address}`);

  // Deploy system
  const PollSystem = await ethers.getContractFactory("PollSystem");
  const   pollSystem   =   await   PollSystem.deploy(token.address,
deployer.address);
  await pollSystem.deployed();
  console.log(`✅ PollSystem deployed at: ${pollSystem.address}`);
```

```
  // Send tokens to PollSystem
  await          (await          token.transfer(pollSystem.address,
ethers.utils.parseEther("1000"))).wait();
  console.log(`💸 Sent 1000 POLL to PollSystem`);

  // Prepare paths
  const deploymentsDir = path.join(__dirname, "../deployments");
  const frontendEnvPath = path.join(__dirname, "../frontend/.env");
  const backendEnvPath = path.join(__dirname, "../backend/.env");

  fs.mkdirSync(deploymentsDir, { recursive: true });

  // Save deploy.json
  fs.writeFileSync(
    path.join(deploymentsDir, "deploy.json"),
    JSON.stringify({
      pollTokenAddress: token.address,
      pollSystemAddress: pollSystem.address,
      relayer: deployer.address,
      network: network.name,
      deployedAt: new Date().toISOString()
    }, null, 2)
  );

  // Save ABI as pure array
  const            pollTokenArtifact            =            await
artifacts.readArtifact("PollToken");
  fs.writeFileSync(
    path.join(deploymentsDir, "poll_token_abi.json"),
    JSON.stringify(pollTokenArtifact.abi, null, 2)
  );

  const            pollSystemArtifact            =            await
artifacts.readArtifact("PollSystem");
  fs.writeFileSync(
    path.join(deploymentsDir, "poll_system_abi.json"),
    JSON.stringify(pollSystemArtifact.abi, null, 2)
  );

  // Write backend .env
  fs.writeFileSync(backendEnvPath, `# Auto-generated by deploy.js
RPC_URL=http://127.0.0.1:8545
CONTRACT_ADDRESS=${pollSystem.address}
POLL_TOKEN_ADDRESS=${token.address}
PRIVATE_KEY=0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae78
4d7bf4f2ff80
`.trim());

  // Write frontend .env ✨
  fs.writeFileSync(frontendEnvPath, `# Auto-generated by deploy.js
VITE_POLL_SYSTEM_ADDRESS=${pollSystem.address}
VITE_POLL_TOKEN_ADDRESS=${token.address}
```

```
VITE_BACKEND_API_URL=http://localhost:8000
`.trim());

  console.log("✅ Deployment complete! All .env and ABI files are in
sync 🧩");
}

main().catch((err) => {
  console.error("❌ Deploy failed:", err);
  process.exit(1);
});
```

Листинг А19 - package-lock.json

```
{
  "name": "frontend",
  "version": "0.0.0",
  "lockfileVersion": 3,
  "requires": true,
  "packages": {
    "": {
      "name": "frontend",
      "version": "0.0.0",
      "dependencies": {
        "buffer": "^6.0.3",
        "ethers": "^5.8.0",
        "react": "^19.1.0",
        "react-dom": "^19.1.0",
        "react-router-dom": "^6.30.1",
        "react-toastify": "^11.0.5"
      },
      "devDependencies": {
        "@eslint/js": "^9.25.0",
        "@metamask/providers": "^22.1.0",
        "@tailwindcss/postcss": "^4.1.8",
        "@types/react": "^19.1.2",
        "@types/react-dom": "^19.1.2",
        "@vitejs/plugin-react": "^4.4.1",
        "autoprefixer": "^10.4.21",
        "eslint": "^9.25.0",
        "eslint-plugin-react-hooks": "^5.2.0",
        "eslint-plugin-react-refresh": "^0.4.19",
        "globals": "^16.0.0",
        "postcss": "^8.5.4",
        "tailwindcss": "^3.4.3",
        "typescript": "~5.8.3",
        "typescript-eslint": "^8.30.1",
        "vite": "^6.3.5"
      }
    },
    "node_modules/@alloc/quick-lru": {
```

```
      "version": "5.2.0",
      "resolved":    "https://registry.npmjs.org/@alloc/quick-lru/-
/quick-lru-5.2.0.tgz",
      "integrity":                                              "sha512-
UrcABB+4bUrFABwbluTIBErXwvbsU/V7TZWfmbgJfbkwiBuziS9gxdODUyuiecfdGQ
85jglMW6juS3+z5TsKLw==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=10"
      },
      "funding": {
        "url": "https://github.com/sponsors/sindresorhus"
      }
    },
    "node_modules/@ampproject/remapping": {
      "version": "2.3.0",
      "resolved":
"https://registry.npmjs.org/@ampproject/remapping/-/remapping-
2.3.0.tgz",
      "integrity":                                              "sha512-
30iZtAPgz+LTIYoeivqYo853f02jBYSd5uGnGpkFV0M3xOt9aN73erkgYAmZU43x4V
fqcnLxW9Kpg3R5LC4YYw==",
      "dev": true,
      "license": "Apache-2.0",
      "dependencies": {
        "@jridgewell/gen-mapping": "^0.3.5",
        "@jridgewell/trace-mapping": "^0.3.24"
      },
      "engines": {
        "node": ">=6.0.0"
      }
    },
    "node_modules/@babel/code-frame": {
      "version": "7.27.1",
      "resolved":   "https://registry.npmjs.org/@babel/code-frame/-
/code-frame-7.27.1.tgz",
      "integrity":                                              "sha512-
cjQ7ZlQ0Mv3b47hABuTevyTuYN4i+loJKGeV9flcCgIK37cCXRh+L1bd3iBHlynerh
Q7BhCkn2BPbQUL+rGqFg==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/helper-validator-identifier": "^7.27.1",
        "js-tokens": "^4.0.0",
        "picocolors": "^1.1.1"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/compat-data": {
      "version": "7.27.3",
```

```
      "resolved":  "https://registry.npmjs.org/@babel/compat-data/-
/compat-data-7.27.3.tgz",
      "integrity":                                          "sha512-
V42wFfx1ymFte+ecf6iXghnnP8kWTO+ZLXIyZq+1LAXHHvTZdVxicn4yiVYdYMGaCO
3tmqub11AorKkv+iodqw==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/core": {
      "version": "7.27.4",
      "resolved":   "https://registry.npmjs.org/@babel/core/-/core-
7.27.4.tgz",
      "integrity":                                          "sha512-
bXYxrXFubeYdvB0NhD/NBB3Qi6aZeV20GOWVI47t2dkecCEoneR4NPVcb7abpXDEve
jgrUfFtG6vG/zxAKmg+g==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@ampproject/remapping": "^2.2.0",
        "@babel/code-frame": "^7.27.1",
        "@babel/generator": "^7.27.3",
        "@babel/helper-compilation-targets": "^7.27.2",
        "@babel/helper-module-transforms": "^7.27.3",
        "@babel/helpers": "^7.27.4",
        "@babel/parser": "^7.27.4",
        "@babel/template": "^7.27.2",
        "@babel/traverse": "^7.27.4",
        "@babel/types": "^7.27.3",
        "convert-source-map": "^2.0.0",
        "debug": "^4.1.0",
        "gensync": "^1.0.0-beta.2",
        "json5": "^2.2.3",
        "semver": "^6.3.1"
      },
      "engines": {
        "node": ">=6.9.0"
      },
      "funding": {
        "type": "opencollective",
        "url": "https://opencollective.com/babel"
      }
    },
    "node_modules/@babel/core/node_modules/semver": {
      "version": "6.3.1",
      "resolved":      "https://registry.npmjs.org/semver/-/semver-
6.3.1.tgz",
      "integrity":                                          "sha512-
BR7VvDCVHO+q2xBEWskxS6DJE1qRnb7DxzUrogb71CWoSficBxYsiAGd+Kl0mmq/Mp
rG9yArRkyrQxTO6XjMzA==",
      "dev": true,
```

```
      "license": "ISC",
      "bin": {
        "semver": "bin/semver.js"
      }
    },
    "node_modules/@babel/generator": {
      "version": "7.27.3",
      "resolved":    "https://registry.npmjs.org/@babel/generator/-
/generator-7.27.3.tgz",
      "integrity":                                                "sha512-
xnlJYj5zepml8NXtjkG0WquFUv8RskFqyFcVgTBp5k+NaA/8uw/K+OSVf8AMGw5e9H
KP2ETd5xpK5MLZQD6b4Q==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/parser": "^7.27.3",
        "@babel/types": "^7.27.3",
        "@jridgewell/gen-mapping": "^0.3.5",
        "@jridgewell/trace-mapping": "^0.3.25",
        "jsesc": "^3.0.2"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/helper-compilation-targets": {
      "version": "7.27.2",
      "resolved":        "https://registry.npmjs.org/@babel/helper-
compilation-targets/-/helper-compilation-targets-7.27.2.tgz",
      "integrity":                                                "sha512-
2+1thGUUWWjLTYTHZWK1n8Yga0ijBz1XAhUXcKy81rd5g6yh7hGqMp45v7cadSbEHc
9G3OTv45SyneRN3ps4DQ==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/compat-data": "^7.27.2",
        "@babel/helper-validator-option": "^7.27.1",
        "browserslist": "^4.24.0",
        "lru-cache": "^5.1.1",
        "semver": "^6.3.1"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/helper-compilation-
targets/node_modules/semver": {
      "version": "6.3.1",
      "resolved":      "https://registry.npmjs.org/semver/-/semver-
6.3.1.tgz",
      "integrity":                                                "sha512-
BR7VvDCVHO+q2xBEWskxS6DJE1qRnb7DxzUrogb71CWoSficBxYsiAGd+Kl0mmq/Mp
rG9yArRkyrQxTO6XjMzA==",
```

```
      "dev": true,
      "license": "ISC",
      "bin": {
        "semver": "bin/semver.js"
      }
    },
    "node_modules/@babel/helper-module-imports": {
      "version": "7.27.1",
      "resolved": "https://registry.npmjs.org/@babel/helper-module-
imports/-/helper-module-imports-7.27.1.tgz",
      "integrity":                                          "sha512-
0gSFWUPNXNopqtIPQvlD5WgXYI5GY2kP2cCvoT8kczjbfcfuIljTbcWrulD1CIPIX2
gt1wghbDy08yE1p+/r3w==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/traverse": "^7.27.1",
        "@babel/types": "^7.27.1"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/helper-module-transforms": {
      "version": "7.27.3",
      "resolved": "https://registry.npmjs.org/@babel/helper-module-
transforms/-/helper-module-transforms-7.27.3.tgz",
      "integrity":                                          "sha512-
dSOvYwvyLsWBeIRyOeHXp5vPj5l1I011r52FM1+r1jCERv+aFXYk4whgQccYEGYxK2
H3ZAIA8nuPkQ0HaUo3qg==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/helper-module-imports": "^7.27.1",
        "@babel/helper-validator-identifier": "^7.27.1",
        "@babel/traverse": "^7.27.3"
      },
      "engines": {
        "node": ">=6.9.0"
      },
      "peerDependencies": {
        "@babel/core": "^7.0.0"
      }
    },
    "node_modules/@babel/helper-plugin-utils": {
      "version": "7.27.1",
      "resolved": "https://registry.npmjs.org/@babel/helper-plugin-
utils/-/helper-plugin-utils-7.27.1.tgz",
      "integrity":                                          "sha512-
1gn1Up5YXka3YYAHGKpbideQ5Yjf1tDa9qYcgysz+cNCXukyLl6DjPXhD3VRwSb8c0
J9tA4b2+rHEZtc6R0tlw==",
      "dev": true,
      "license": "MIT",
```

```
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/helper-string-parser": {
      "version": "7.27.1",
      "resolved": "https://registry.npmjs.org/@babel/helper-string-
parser/-/helper-string-parser-7.27.1.tgz",
      "integrity":                                          "sha512-
qMlSxKbpRlAridDExk92nSobyDdpPijUq2DW6oDnUqd0iOGxmQjyqhMIihI9+zv4LP
yZdRje2cavWPbCbWm3eA==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/helper-validator-identifier": {
      "version": "7.27.1",
      "resolved":        "https://registry.npmjs.org/@babel/helper-
validator-identifier/-/helper-validator-identifier-7.27.1.tgz",
      "integrity":                                          "sha512-
D2hP9eA+Sqx1kBZgzxZh0y1trbuU+JoDkiEwqhQ36nodYqJwyEIhPSdMNd7lOm/4io
72luTPWH20Yda0xOuUow==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/helper-validator-option": {
      "version": "7.27.1",
      "resolved":        "https://registry.npmjs.org/@babel/helper-
validator-option/-/helper-validator-option-7.27.1.tgz",
      "integrity":                                          "sha512-
YvjJow9FxbhFFKDSuFnVCe2WxXk1zWc22fFePVNEaWJEu8IrZVlda6N0uHwzZrUM1i
l7NC9Mlp4MaJYbYd9JSg==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/helpers": {
      "version": "7.27.4",
      "resolved":      "https://registry.npmjs.org/@babel/helpers/-
/helpers-7.27.4.tgz",
      "integrity":                                          "sha512-
Y+bO6U+I7ZKaM5G5rDUZiYfUvQPUibYmAFe7EnKdnKBbVXDZxvp+MWOH5gYciY0EPk
4EScsuFMQBbEfpdRKSCQ==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
```

```
        "@babel/template": "^7.27.2",
        "@babel/types": "^7.27.3"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/parser": {
      "version": "7.27.4",
      "resolved":       "https://registry.npmjs.org/@babel/parser/-
/parser-7.27.4.tgz",
      "integrity":                                              "sha512-
BRmLHGwpUqLFR2jzx9orBuX/ABDkj2jLKOXrHDTN2aOKL+jFDDKaRNo9nyYsIl9h/U
E/7lMKdDjKQQyxKKDZ7g==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/types": "^7.27.3"
      },
      "bin": {
        "parser": "bin/babel-parser.js"
      },
      "engines": {
        "node": ">=6.0.0"
      }
    },
    "node_modules/@babel/plugin-transform-react-jsx-self": {
      "version": "7.27.1",
      "resolved":        "https://registry.npmjs.org/@babel/plugin-
transform-react-jsx-self/-/plugin-transform-react-jsx-self-
7.27.1.tgz",
      "integrity":                                              "sha512-
6UzkCs+ejGdZ5mFFC/OCUrv028ab2fp1znZmCZjAOBKiBK2jXD1O+BPSfX8X2qjJ75
fZBMSnQn3Rq2mrBJK2mw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/helper-plugin-utils": "^7.27.1"
      },
      "engines": {
        "node": ">=6.9.0"
      },
      "peerDependencies": {
        "@babel/core": "^7.0.0-0"
      }
    },
    "node_modules/@babel/plugin-transform-react-jsx-source": {
      "version": "7.27.1",
      "resolved":        "https://registry.npmjs.org/@babel/plugin-
transform-react-jsx-source/-/plugin-transform-react-jsx-source-
7.27.1.tgz",
```

```
      "integrity": "sha512-
zbwoTsBruTeKB9hSq73ha66iFeJHuaFkUbwvqElnygoNbj/jHRsSeokowZFN3CZ64I
vEqcmmkVe89OPXc7ldAw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/helper-plugin-utils": "^7.27.1"
      },
      "engines": {
        "node": ">=6.9.0"
      },
      "peerDependencies": {
        "@babel/core": "^7.0.0-0"
      }
    },
    "node_modules/@babel/template": {
      "version": "7.27.2",
      "resolved": "https://registry.npmjs.org/@babel/template/-
/template-7.27.2.tgz",
      "integrity": "sha512-
LPDZ85aEJyYSd18/DkjNh4/y1ntkE5KwUHWTiqgRxruuZL2F1yuHligVHLvcHY2vMH
XttKFpJn6LwfI7cw7ODw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/code-frame": "^7.27.1",
        "@babel/parser": "^7.27.2",
        "@babel/types": "^7.27.1"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/traverse": {
      "version": "7.27.4",
      "resolved": "https://registry.npmjs.org/@babel/traverse/-
/traverse-7.27.4.tgz",
      "integrity": "sha512-
oNcu2QbHqts9BtOWJosOVJapWjBDSxGCpFvikNR5TGDYDQf3JwpIoMzIKrvfoti93c
LfPJEG4tH9SPVeyCGgdA==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/code-frame": "^7.27.1",
        "@babel/generator": "^7.27.3",
        "@babel/parser": "^7.27.4",
        "@babel/template": "^7.27.2",
        "@babel/types": "^7.27.3",
        "debug": "^4.3.1",
        "globals": "^11.1.0"
      },
      "engines": {
        "node": ">=6.9.0"
```

```
      }
    },
    "node_modules/@babel/traverse/node_modules/globals": {
      "version": "11.12.0",
      "resolved":    "https://registry.npmjs.org/globals/-/globals-
11.12.0.tgz",
      "integrity":                                              "sha512-
WOBp/EEGUiIsJSp7wcv/y6MO+lV9UoncWqxuFfm8eBwzWNgyfBd6Gz+IeKQ9jCmyho
H99g15M3T+QaVHFjizVA==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=4"
      }
    },
    "node_modules/@babel/types": {
      "version": "7.27.3",
      "resolved": "https://registry.npmjs.org/@babel/types/-/types-
7.27.3.tgz",
      "integrity":                                              "sha512-
Y1GkI4ktrtvmawoSq+4FCVHNryea6uR+qUQy0AGxLSsjCX0nVmkYQMBLHDkXZuo5hG
x7eYdnIaslsdBFm7zbUw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/helper-string-parser": "^7.27.1",
        "@babel/helper-validator-identifier": "^7.27.1"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@esbuild/aix-ppc64": {
      "version": "0.25.5",
      "resolved":  "https://registry.npmjs.org/@esbuild/aix-ppc64/-
/aix-ppc64-0.25.5.tgz",
      "integrity":                                              "sha512-
9o3TMmpmftaCMepOdA5k/yDw8SfInyzWWTjYTFCX3kPSDJMROQTb8jg+h9Cnwnmm1v
OzvxN7gIfB5V2ewpjtGA==",
      "cpu": [
        "ppc64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "aix"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/android-arm": {
```

```
      "version": "0.25.5",
      "resolved":     "https://registry.npmjs.org/@esbuild/android-
arm/-/android-arm-0.25.5.tgz",
      "integrity":                                              "sha512-
AdJKSPeEHgi7/ZhuIPtcQKr5RQdo6OO2IL87JkianiMYMPbCtot9fxPbrMiBADOWWm
3T2si9stAiVsGbTQFkbA==",
      "cpu": [
        "arm"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "android"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/android-arm64": {
      "version": "0.25.5",
      "resolved":     "https://registry.npmjs.org/@esbuild/android-
arm64/-/android-arm64-0.25.5.tgz",
      "integrity":                                              "sha512-
VGzGhj4lJO+TVGV1v8ntCZWJktV7SGCs3Pn1GRWI1SBFtRALoomm8k5E9Pmwg3HOAa
l2VDc2F9+PM/rEY6oIDg==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "android"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/android-x64": {
      "version": "0.25.5",
      "resolved":     "https://registry.npmjs.org/@esbuild/android-
x64/-/android-x64-0.25.5.tgz",
      "integrity":                                              "sha512-
D2GyJT1kjvO//drbRT3Hib9XPwQeWd9vZoBJn+bu/lVsOZ13cqNdDeqIF/xQ5/VmWv
MduP6AmXvylO/PIc2isw==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
```

```
        "android"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/darwin-arm64": {
      "version": "0.25.5",
      "resolved":      "https://registry.npmjs.org/@esbuild/darwin-
arm64/-/darwin-arm64-0.25.5.tgz",
      "integrity":                                              "sha512-
GtaBgammVvdF7aPIgH2jxMDdivezgFu6iKpmT+48+F8Hhg5J/sfnDieg0aeG/jfSvk
YQU2/pceFPDKlqZzwnfQ==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "darwin"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/darwin-x64": {
      "version": "0.25.5",
      "resolved": "https://registry.npmjs.org/@esbuild/darwin-x64/-
/darwin-x64-0.25.5.tgz",
      "integrity":                                              "sha512-
1iT4FVL0dJ76/q1wd7XDsXrSW+oLoquptvh4CLR4kITDtqi2e/xwXwdCVH8hVHU43w
gJdsq7Gxuzcs6Iq/7bxQ==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "darwin"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/freebsd-arm64": {
      "version": "0.25.5",
      "resolved":     "https://registry.npmjs.org/@esbuild/freebsd-
arm64/-/freebsd-arm64-0.25.5.tgz",
      "integrity":                                              "sha512-
nk4tGP3JThz4La38Uy/gzyXtpkPW8zSAmoUhK9xKKXdBCzKODMc2adkB2+8om9BDYu
gz+uGV7sLmpTYzvmz6Sw==",
```

```
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "freebsd"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/freebsd-x64": {
      "version": "0.25.5",
      "resolved":     "https://registry.npmjs.org/@esbuild/freebsd-
x64/-/freebsd-x64-0.25.5.tgz",
      "integrity":                                          "sha512-
PrikaNjiXdR2laW6OIjlbeuCPrPaAl0IwPIaRv+SMV8CiM8i2LqVUHFC1+8eORgWyY
7yhQY+2U2fA55mBzReaw==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "freebsd"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/linux-arm": {
      "version": "0.25.5",
      "resolved":  "https://registry.npmjs.org/@esbuild/linux-arm/-
/linux-arm-0.25.5.tgz",
      "integrity":                                          "sha512-
cPzojwW2okgh7ZlRpcBEtsX7WBuqbLrNXqLU89GxWbNt6uIg78ET82qifUy3W6OVww
6ZWobWub5oqZOVtwolfw==",
      "cpu": [
        "arm"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ],
      "engines": {
        "node": ">=18"
      }
    },
```

```
    "node_modules/@esbuild/linux-arm64": {
      "version": "0.25.5",
      "resolved":       "https://registry.npmjs.org/@esbuild/linux-
arm64/-/linux-arm64-0.25.5.tgz",
      "integrity":                                                "sha512-
Z9kfb1v6ZlGbWj8EJk9T6czVEjjq2ntSYLY2cw6pAZl4oKtfgQuS4HOq41M/BcoLPz
rUbNd+R4BXFyH//nHxVg==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/linux-ia32": {
      "version": "0.25.5",
      "resolved": "https://registry.npmjs.org/@esbuild/linux-ia32/-
/linux-ia32-0.25.5.tgz",
      "integrity":                                                "sha512-
sQ7l00M8bSv36GLV95BVAdhJ2QsIbCuCjh/uYrWiMQSUuV+LpXwIqhgJDcvMTj+VsQ
mqAHL2yYaasENvJ7CDKA==",
      "cpu": [
        "ia32"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/linux-loong64": {
      "version": "0.25.5",
      "resolved":       "https://registry.npmjs.org/@esbuild/linux-
loong64/-/linux-loong64-0.25.5.tgz",
      "integrity":                                                "sha512-
0ur7ae16hDUC4OL5iEnDb0tZHDxYmuQyhKhsPBV8f99f6Z9KQM02g33f93rNH5A30a
gMS46u2HP6qTdEt6Q1kg==",
      "cpu": [
        "loong64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
```

```
      "os": [
        "linux"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/linux-mips64el": {
      "version": "0.25.5",
      "resolved":       "https://registry.npmjs.org/@esbuild/linux-
mips64el/-/linux-mips64el-0.25.5.tgz",
      "integrity":                                              "sha512-
kB/66P1OsHO5zLz0i6X0RxlQ+3cu0mkxS3TKFvkb5lin6uwZ/ttOkP3Z8lfR9mJOBk
14ZwZ9182SIIWFGNmqmg==",
      "cpu": [
        "mips64el"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/linux-ppc64": {
      "version": "0.25.5",
      "resolved":       "https://registry.npmjs.org/@esbuild/linux-
ppc64/-/linux-ppc64-0.25.5.tgz",
      "integrity":                                              "sha512-
UZCmJ7r9X2fe2D6jBmkLBMQetXPXIsZjQJCjgwpVDz+YMcS6oFR27alkgGv3Oqkv07
bxdvw7fyB71/olceJhkQ==",
      "cpu": [
        "ppc64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/linux-riscv64": {
      "version": "0.25.5",
      "resolved":       "https://registry.npmjs.org/@esbuild/linux-
riscv64/-/linux-riscv64-0.25.5.tgz",
```

```
      "integrity":                                          "sha512-
kTxwu4mLyeOlsVIFPfQo+fQJAV9mh24xL+y+Bm6ej067sYANjyEw1dNHmvoqxJUCMn
kBdKpvOn0Ahql6+4VyeA==",
      "cpu": [
        "riscv64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/linux-s390x": {
      "version": "0.25.5",
      "resolved":       "https://registry.npmjs.org/@esbuild/linux-
s390x/-/linux-s390x-0.25.5.tgz",
      "integrity":                                          "sha512-
K2dSKTKfmdh78uJ3NcWFiqyRrimfdinS5ErLSn3vluHNeHVnBAFWC8a4X5N+7FgVE1
EjXS1QDZbpqZBjfrqMTQ==",
      "cpu": [
        "s390x"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/linux-x64": {
      "version": "0.25.5",
      "resolved":  "https://registry.npmjs.org/@esbuild/linux-x64/-
/linux-x64-0.25.5.tgz",
      "integrity":                                          "sha512-
uhj8N2obKTE6pSZ+aMUbqq+1nXxNjZIIjCjGLfsWvVpy7gKCOL6rsY1MhRh9zLtUtA
I7vpgLMK6DxjO8Qm9lJw==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ],
      "engines": {
```

```
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/netbsd-arm64": {
      "version": "0.25.5",
      "resolved":      "https://registry.npmjs.org/@esbuild/netbsd-
arm64/-/netbsd-arm64-0.25.5.tgz",
      "integrity":                                              "sha512-
pwHtMP9viAy1oHPvgxtOv+OkduK5ugofNTVDilIzBLpoWAM16r7b/mxBvfpuQDpRQF
MfuVr5aLcn4yveGvBZvw==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "netbsd"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/netbsd-x64": {
      "version": "0.25.5",
      "resolved": "https://registry.npmjs.org/@esbuild/netbsd-x64/-
/netbsd-x64-0.25.5.tgz",
      "integrity":                                              "sha512-
WOb5fKrvVTRMfWFNCroYWWklbnXH0Q5rZppjq0vQIdlsQKuw6mdSihwSo4RV/YdQ5U
CKKvBy7/0ZZYLBZKIbwQ==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "netbsd"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/openbsd-arm64": {
      "version": "0.25.5",
      "resolved":     "https://registry.npmjs.org/@esbuild/openbsd-
arm64/-/openbsd-arm64-0.25.5.tgz",
      "integrity":                                              "sha512-
7A208+uQKgTxHd0G0uqZO8UjK2R0DDb4fDmERtARjSHWxqMTye4Erz4zZafx7Di9Cv
+lNHYuncAkiGFySoD+Mw==",
      "cpu": [
        "arm64"
      ],
```

```
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "openbsd"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/openbsd-x64": {
      "version": "0.25.5",
      "resolved":     "https://registry.npmjs.org/@esbuild/openbsd-
x64/-/openbsd-x64-0.25.5.tgz",
      "integrity":                                              "sha512-
G4hE405ErTWraiZ8UiSoesH8DaCsMm0Cay4fsFWOOUcz8b8rC6uCvnagr+gnioEjWn
0wC+o1/TAHt+It+MpIMg==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "openbsd"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/sunos-x64": {
      "version": "0.25.5",
      "resolved":  "https://registry.npmjs.org/@esbuild/sunos-x64/-
/sunos-x64-0.25.5.tgz",
      "integrity":                                              "sha512-
l+azKShMy7FxzY0Rj4RCt5VD/q8mG/e+mDivgspo+yL8zW7qEwctQ6YqKX34DTEleF
AvCIUviCFX1SDZRSyMQA==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "sunos"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/win32-arm64": {
      "version": "0.25.5",
```

```
      "resolved":       "https://registry.npmjs.org/@esbuild/win32-
arm64/-/win32-arm64-0.25.5.tgz",
      "integrity":                                              "sha512-
O2S7SNZzdcFG7eFKgvwUEZ2VG9D/sn/eIiz8XRZ1Q/DO5a3s76Xv0mdBzVM5j5R639
lXQmPmSo0iRpHqUUrsxw==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "win32"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/win32-ia32": {
      "version": "0.25.5",
      "resolved": "https://registry.npmjs.org/@esbuild/win32-ia32/-
/win32-ia32-0.25.5.tgz",
      "integrity":                                              "sha512-
onOJ02pqs9h1iMJ1PQphR+VZv8qBMQ77Klcsqv9CNW2w6yLqoURLcgERAIurY6QE63
bbLuqgP9ATqajFLK5AMQ==",
      "cpu": [
        "ia32"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "win32"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/win32-x64": {
      "version": "0.25.5",
      "resolved":  "https://registry.npmjs.org/@esbuild/win32-x64/-
/win32-x64-0.25.5.tgz",
      "integrity":                                              "sha512-
TXv6YnJ8ZMVdX+SXWVBo/0p8LTcrUYngpWjvm91TMjjBQii7Oz11Lw5lbDV5Y0Tzuh
SJHwiH4hEtC1I42mMS0g==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "win32"
```

```
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@eslint-community/eslint-utils": {
      "version": "4.7.0",
      "resolved":              "https://registry.npmjs.org/@eslint-
community/eslint-utils/-/eslint-utils-4.7.0.tgz",
      "integrity":                                              "sha512-
dyybb3AcajC7uha6CvhdVRJqaKyn7w2YKqKyAN37NKYgZT36w+iRb0Dymmc5qEJ549
c/S31cMMSFd75bteCpCw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "eslint-visitor-keys": "^3.4.3"
      },
      "engines": {
        "node": "^12.22.0 || ^14.17.0 || >=16.0.0"
      },
      "funding": {
        "url": "https://opencollective.com/eslint"
      },
      "peerDependencies": {
        "eslint": "^6.0.0 || ^7.0.0 || >=8.0.0"
      }
    },
    "node_modules/@eslint-community/eslint-
utils/node_modules/eslint-visitor-keys": {
      "version": "3.4.3",
      "resolved": "https://registry.npmjs.org/eslint-visitor-keys/-
/eslint-visitor-keys-3.4.3.tgz",
      "integrity":                                              "sha512-
wpc+LXeiyiisxPlEkUzU6svyS1frIO3Mgxj1fdy7Pm8Ygzguax2N3Fa/D/ag1WqbOp
rdI+uY6wMUl8/a2G+iag==",
      "dev": true,
      "license": "Apache-2.0",
      "engines": {
        "node": "^12.22.0 || ^14.17.0 || >=16.0.0"
      },
      "funding": {
        "url": "https://opencollective.com/eslint"
      }
    },
    "node_modules/@eslint-community/regexpp": {
      "version": "4.12.1",
      "resolved":              "https://registry.npmjs.org/@eslint-
community/regexpp/-/regexpp-4.12.1.tgz",
      "integrity":                                              "sha512-
CCZCDJuduB9OUkFkY2IgppNZMi2lBQgD2qzwXkEia16cge2pijY/aXi96CJMquDMn3
nJdlPV1A5KrJEXwfLNzQ==",
      "dev": true,
      "license": "MIT",
```

```
      "engines": {
        "node": "^12.0.0 || ^14.0.0 || >=16.0.0"
      }
    },
    "node_modules/@eslint/config-array": {
      "version": "0.20.0",
      "resolved":       "https://registry.npmjs.org/@eslint/config-
array/-/config-array-0.20.0.tgz",
      "integrity":                                          "sha512-
fxlS1kkIjx8+vy2SjuCB94q3htSNrufYTXubwiBFeaQHbH6Ipi43gFJq2zCMt6PHhI
mH3Xmr0NksKDvchWlpQQ==",
      "dev": true,
      "license": "Apache-2.0",
      "dependencies": {
        "@eslint/object-schema": "^2.1.6",
        "debug": "^4.3.1",
        "minimatch": "^3.1.2"
      },
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      }
    },
    "node_modules/@eslint/config-helpers": {
      "version": "0.2.2",
      "resolved":       "https://registry.npmjs.org/@eslint/config-
helpers/-/config-helpers-0.2.2.tgz",
      "integrity":                                          "sha512-
+GPzk8PlG0sPpzdU5ZvIRMPidzAnZDl/s9L+y13iodqvb8leL53bTannOrQ/Im7Ukp
smFU5Ily5U60LWixnmLg==",
      "dev": true,
      "license": "Apache-2.0",
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      }
    },
    "node_modules/@eslint/core": {
      "version": "0.14.0",
      "resolved":  "https://registry.npmjs.org/@eslint/core/-/core-
0.14.0.tgz",
      "integrity":                                          "sha512-
qIbV0/JZr7iSDjqAc60IqbLdsj9GDt16xQtWD+B78d/HAlvysGdZZ6rpJHGAc2T0FQ
x1X6thsSPdnoiGKdNtdg==",
      "dev": true,
      "license": "Apache-2.0",
      "dependencies": {
        "@types/json-schema": "^7.0.15"
      },
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      }
    },
    "node_modules/@eslint/eslintrc": {
      "version": "3.3.1",
```

```
      "resolved":    "https://registry.npmjs.org/@eslint/eslintrc/-
/eslintrc-3.3.1.tgz",
      "integrity":                                              "sha512-
gtF186CXhIl1p4pJNGZw8Yc6RlshoePRvE0X91oPGb3vZ8pM3qOS9W9NGPat9LziaB
V7XrJWGylNQXkGcnM3IQ==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "ajv": "^6.12.4",
        "debug": "^4.3.2",
        "espree": "^10.0.1",
        "globals": "^14.0.0",
        "ignore": "^5.2.0",
        "import-fresh": "^3.2.1",
        "js-yaml": "^4.1.0",
        "minimatch": "^3.1.2",
        "strip-json-comments": "^3.1.1"
      },
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      },
      "funding": {
        "url": "https://opencollective.com/eslint"
      }
    },
    "node_modules/@eslint/eslintrc/node_modules/globals": {
      "version": "14.0.0",
      "resolved":    "https://registry.npmjs.org/globals/-/globals-
14.0.0.tgz",
      "integrity":                                              "sha512-
oahGvuMGQlPw/ivIYBjVSrWAfWLBeku5tpPE2fOPLi+WHffIWbuh2tCjhyQhTBPMf5
E9jDEH4FOmTYgYwbKwtQ==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=18"
      },
      "funding": {
        "url": "https://github.com/sponsors/sindresorhus"
      }
    },
    "node_modules/@eslint/js": {
      "version": "9.28.0",
      "resolved":      "https://registry.npmjs.org/@eslint/js/-/js-
9.28.0.tgz",
      "integrity":                                              "sha512-
fnqSjGWd/CoIp4EXIxWVK/sHA6DOHN4+8Ix2cX5ycOY7LG0UY8nHCU5pIp2eaE1Mc7
Qd8kHspYNzYXT2ojPLzg==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      },
```

```
      "funding": {
        "url": "https://eslint.org/donate"
      }
    },
    "node_modules/@eslint/object-schema": {
      "version": "2.1.6",
      "resolved":       "https://registry.npmjs.org/@eslint/object-
schema/-/object-schema-2.1.6.tgz",
      "integrity":                                          "sha512-
RBMg5FRL0I0gs51M/guSAj5/e14VQ4tpZnQNWwuDT66P14I43ItmPfIZRhO9fUVIPO
AQXU47atlywZ/czoqFPA==",
      "dev": true,
      "license": "Apache-2.0",
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      }
    },
    "node_modules/@eslint/plugin-kit": {
      "version": "0.3.1",
      "resolved":  "https://registry.npmjs.org/@eslint/plugin-kit/-
/plugin-kit-0.3.1.tgz",
      "integrity":                                          "sha512-
0J+zgWxHN+xXONWIyPWKFMgVuJoZuGiIFu8yxk7RJjxkzpGmyja5wRFqZIVtjDVOQp
V+Rw0iOAjYPE2eQyjr0w==",
      "dev": true,
      "license": "Apache-2.0",
      "dependencies": {
        "@eslint/core": "^0.14.0",
        "levn": "^0.4.1"
      },
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      }
    },
    "node_modules/@ethereumjs/common": {
      "version": "3.2.0",
      "resolved":  "https://registry.npmjs.org/@ethereumjs/common/-
/common-3.2.0.tgz",
      "integrity":                                          "sha512-
pksvzI0VyLgmuEF2FA/JR/4/y6hcPq8OUail3/AvycBaW1d5VSauOZzqGvJ3RTmR4M
U35lWE8KseKOsEhrFRBA==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@ethereumjs/util": "^8.1.0",
        "crc-32": "^1.2.0"
      }
    },
    "node_modules/@ethereumjs/rlp": {
      "version": "4.0.1",
      "resolved":     "https://registry.npmjs.org/@ethereumjs/rlp/-
/rlp-4.0.1.tgz",
```

```
      "integrity":                                          "sha512-
tqsQiBQDQdmPWE1xkkBq4rlSW5QZpLOUJ5RJh2/9fug+q9tnUhuZoVLk7s0scUIKTO
zEtR72DFBXI4WiZcMpvw==",
      "dev": true,
      "license": "MPL-2.0",
      "bin": {
        "rlp": "bin/rlp"
      },
      "engines": {
        "node": ">=14"
      }
    },
    "node_modules/@ethereumjs/tx": {
      "version": "4.2.0",
      "resolved":  "https://registry.npmjs.org/@ethereumjs/tx/-/tx-
4.2.0.tgz",
      "integrity":                                          "sha512-
1nc6VO4jtFd172BbSnTnDQVr9IYBFl1y4xPzZdtkrkKIncBCkdbgfdRV+MiTkJYAtT
xvV12GRZLqBFT1PNK6Yw==",
      "dev": true,
      "license": "MPL-2.0",
      "dependencies": {
        "@ethereumjs/common": "^3.2.0",
        "@ethereumjs/rlp": "^4.0.1",
        "@ethereumjs/util": "^8.1.0",
        "ethereum-cryptography": "^2.0.0"
      },
      "engines": {
        "node": ">=14"
      }
    },
    "node_modules/@ethereumjs/util": {
      "version": "8.1.0",
      "resolved":    "https://registry.npmjs.org/@ethereumjs/util/-
/util-8.1.0.tgz",
      "integrity":                                          "sha512-
zQ0IqbdX8FZ9aw11vP+dZkKDkS+kgIvQPHnSAXzP9pLu+Rfu3D3XEeLbicvoXJTYnh
ZiPmsZUxgdzXwNKxRPbA==",
      "dev": true,
      "license": "MPL-2.0",
      "dependencies": {
        "@ethereumjs/rlp": "^4.0.1",
        "ethereum-cryptography": "^2.0.0",
        "micro-ftch": "^0.3.1"
      },
      "engines": {
        "node": ">=14"
      }
    },
    "node_modules/@ethersproject/abi": {
      "version": "5.8.0",
      "resolved":  "https://registry.npmjs.org/@ethersproject/abi/-
/abi-5.8.0.tgz",
```

```
      "integrity":                                              "sha512-
b9YS/43ObplgyV6SlyQsG53/vkSal0MNA1fskSC4mbnCMi8R+NkcH8K9FPYNESf6jU
efBUniE4SOKms0E/KK1Q==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/address": "^5.8.0",
        "@ethersproject/bignumber": "^5.8.0",
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/constants": "^5.8.0",
        "@ethersproject/hash": "^5.8.0",
        "@ethersproject/keccak256": "^5.8.0",
        "@ethersproject/logger": "^5.8.0",
        "@ethersproject/properties": "^5.8.0",
        "@ethersproject/strings": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/abstract-provider": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/abstract-provider/-
/abstract-provider-5.8.0.tgz",
      "integrity":                                              "sha512-
wC9SFcmh4UK0oKuLJQItoQdzS/qZ51EJegK6EmAWlh+OptpQ/npECOR3QqECd8iGHC
0RJb4WKbVdSfif4ammrg==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/bignumber": "^5.8.0",
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/logger": "^5.8.0",
        "@ethersproject/networks": "^5.8.0",
        "@ethersproject/properties": "^5.8.0",
```

```
        "@ethersproject/transactions": "^5.8.0",
        "@ethersproject/web": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/abstract-signer": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/abstract-signer/-
/abstract-signer-5.8.0.tgz",
      "integrity":                                          "sha512-
N0XhZTswXcmIZQdYtUnd79VJzvEwXQw6PK0dTl9VoYrEBxxCPXqS0Eod7q5TNKRxe1
/5WUMuR0u0nqTF/avdCA==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/abstract-provider": "^5.8.0",
        "@ethersproject/bignumber": "^5.8.0",
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/logger": "^5.8.0",
        "@ethersproject/properties": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/address": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/address/-/address-
5.8.0.tgz",
      "integrity":                                          "sha512-
GhH/abcC46LJwshoN+uBNoKVFPxUuZm6dA257z0vZkKmU1+t8xTn8oK7B9qrj8W2rF
RMch4gbJl6PmVxjxBEBA==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
```

```
        "@ethersproject/bignumber": "^5.8.0",
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/keccak256": "^5.8.0",
        "@ethersproject/logger": "^5.8.0",
        "@ethersproject/rlp": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/base64": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/base64/-/base64-
5.8.0.tgz",
      "integrity":                                          "sha512-
lN0oIwfkYj9LbPx4xEkie6rAMJtySbpOAFXSDVQaBnAzYfB4X2Qr+FXJGxMoc3Bxp2
Sm8OwvzMrywxyw0gLjIQ==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/bytes": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/basex": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/basex/-/basex-
5.8.0.tgz",
      "integrity":                                          "sha512-
PIgTszMlDRmNwW9nhS6iqtVfdTAKosA7llYXNmGPw4YAI1PUyMv28988wAb41/gHF/
WqGdoLv0erHaRcHRKW2Q==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/bytes": "^5.8.0",
```

```
        "@ethersproject/properties": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/bignumber": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/bignumber/-/bignumber-
5.8.0.tgz",
      "integrity":                                          "sha512-
ZyaT24bHaSeJon2tGPKIiHszWjD/54Sz8t57Toch475lCLljC6MgPmxk7Gtzz+ddNN
5LuHea9qhAe0x3D+uYPA==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/logger": "^5.8.0",
        "bn.js": "^5.2.1"
      }
    },
    "node_modules/@ethersproject/bytes": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/bytes/-/bytes-
5.8.0.tgz",
      "integrity":                                          "sha512-
vTkeohgJVCPVHu5c25XWaWQOZ4v+DkGoC42/TS2ond+PARCxTJvgTFUNDZovyQ/uAQ
4EcpqqowKydcdmRKjg7A==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/logger": "^5.8.0"
      }
    },
```

```
    "node_modules/@ethersproject/constants": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/constants/-/constants-
5.8.0.tgz",
      "integrity":                                          "sha512-
wigX4lrf5Vu+axVTIvNsuL6YrV4O5AXl5ubcURKMEME5TnWBouUh0CDTWxZ2GpnRn1
kcCgE7l8O5+VbV9QTTcg==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/bignumber": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/contracts": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/contracts/-/contracts-
5.8.0.tgz",
      "integrity":                                          "sha512-
0eFjGz9GtuAi6MZwhb4uvUM216F38xiuR0yYCjKJpNfSEy4HUM8hvqqBj9Jmm0IUz8
l0xKEhWwLIhPgxNY0yvQ==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/abi": "^5.8.0",
        "@ethersproject/abstract-provider": "^5.8.0",
        "@ethersproject/abstract-signer": "^5.8.0",
        "@ethersproject/address": "^5.8.0",
        "@ethersproject/bignumber": "^5.8.0",
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/constants": "^5.8.0",
        "@ethersproject/logger": "^5.8.0",
```

```
        "@ethersproject/properties": "^5.8.0",
        "@ethersproject/transactions": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/hash": {
      "version": "5.8.0",
      "resolved": "https://registry.npmjs.org/@ethersproject/hash/-
/hash-5.8.0.tgz",
      "integrity":                                          "sha512-
ac/lBcTbEWW/VGJij0CNSw/wPcw9bSRgCB0AIBz8CvED/jfvDoV9hsIIiWfvWmFEi8
RcXtlNwp2jv6ozWOsooA==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/abstract-signer": "^5.8.0",
        "@ethersproject/address": "^5.8.0",
        "@ethersproject/base64": "^5.8.0",
        "@ethersproject/bignumber": "^5.8.0",
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/keccak256": "^5.8.0",
        "@ethersproject/logger": "^5.8.0",
        "@ethersproject/properties": "^5.8.0",
        "@ethersproject/strings": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/hdnode": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/hdnode/-/hdnode-
5.8.0.tgz",
      "integrity":                                          "sha512-
4bK1VF6E83/3/Im0ERnnUeWOY3P1BZml4ZD3wcH8Ys0/d1h1xaFt6Zc+Dh9zXf9Tap
Gro0T4wvO71UTCp3/uoA==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
```

```
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/abstract-signer": "^5.8.0",
        "@ethersproject/basex": "^5.8.0",
        "@ethersproject/bignumber": "^5.8.0",
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/logger": "^5.8.0",
        "@ethersproject/pbkdf2": "^5.8.0",
        "@ethersproject/properties": "^5.8.0",
        "@ethersproject/sha2": "^5.8.0",
        "@ethersproject/signing-key": "^5.8.0",
        "@ethersproject/strings": "^5.8.0",
        "@ethersproject/transactions": "^5.8.0",
        "@ethersproject/wordlists": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/json-wallets": {
      "version": "5.8.0",
      "resolved":  "https://registry.npmjs.org/@ethersproject/json-
wallets/-/json-wallets-5.8.0.tgz",
      "integrity":                                          "sha512-
HxblNck8FVUtNxS3VTEYJAcwiKYsBIF77W15HufqlBF9gGfhmYOJtYZp8fSDZtn9y5
EaXTE87zDwzxRoTFk11w==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/abstract-signer": "^5.8.0",
        "@ethersproject/address": "^5.8.0",
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/hdnode": "^5.8.0",
        "@ethersproject/keccak256": "^5.8.0",
        "@ethersproject/logger": "^5.8.0",
        "@ethersproject/pbkdf2": "^5.8.0",
        "@ethersproject/properties": "^5.8.0",
        "@ethersproject/random": "^5.8.0",
        "@ethersproject/strings": "^5.8.0",
        "@ethersproject/transactions": "^5.8.0",
        "aes-js": "3.0.0",
        "scrypt-js": "3.0.1"
      }
    },
    "node_modules/@ethersproject/keccak256": {
```

```
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/keccak256/-/keccak256-
5.8.0.tgz",
      "integrity":                                          "sha512-
A1pkKLZSz8pDaQ1ftutZoaN46I6+jvuqugx5KYNeQOPqq+JZ0Txm7dlWesCHB5cndJ
Su5vP2VKptKf7cksERng==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/bytes": "^5.8.0",
        "js-sha3": "0.8.0"
      }
    },
    "node_modules/@ethersproject/logger": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/logger/-/logger-
5.8.0.tgz",
      "integrity":                                          "sha512-
Qe6knGmY+zPPWTC+wQrpitodgBfH7XoceCGL5bJVejmH+yCS3R8jJm8iiWuvWbG76R
UmyEG53oqv6GMVWqunjA==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT"
    },
    "node_modules/@ethersproject/networks": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/networks/-/networks-
5.8.0.tgz",
      "integrity":                                          "sha512-
egPJh3aPVAzbHwq8DD7Po53J4OUSsA1MjQp8Vf/OZPav5rlmWUaFLiq8cvQiGK0Z5K
6LYzm29+VA/p4RL1FzNg==",
```

```
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/logger": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/pbkdf2": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/pbkdf2/-/pbkdf2-
5.8.0.tgz",
      "integrity":                                          "sha512-
wuHiv97BrzCmfEaPbUFpMjlVg/IDkZThp9Ri88BpjRleg4iePJaj2SW8AIyE8cXn5V
1tuAaMj6lzvsGJkGWskg==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/sha2": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/properties": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/properties/-
/properties-5.8.0.tgz",
      "integrity":                                          "sha512-
PYuiEoQ+FMaZZNGrStmN7+lWjlsoufGIHdww7454FIaGdbe/p5rnaCXTr5MtBYl3Nk
eoVhHZuyzChPeGeKIpQw==",
      "funding": [
        {
          "type": "individual",
```

```
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/logger": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/providers": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/providers/-/providers-
5.8.0.tgz",
      "integrity":                                          "sha512-
3Il3oTzEx3o6kzcg9ZzbE+oCZYyY+3Zh83sKkn4s1DZfTUjIegHnN2Cm0kbn9YFy45
FDVcuCLLONhU7ny0SsCw==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/abstract-provider": "^5.8.0",
        "@ethersproject/abstract-signer": "^5.8.0",
        "@ethersproject/address": "^5.8.0",
        "@ethersproject/base64": "^5.8.0",
        "@ethersproject/basex": "^5.8.0",
        "@ethersproject/bignumber": "^5.8.0",
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/constants": "^5.8.0",
        "@ethersproject/hash": "^5.8.0",
        "@ethersproject/logger": "^5.8.0",
        "@ethersproject/networks": "^5.8.0",
        "@ethersproject/properties": "^5.8.0",
        "@ethersproject/random": "^5.8.0",
        "@ethersproject/rlp": "^5.8.0",
        "@ethersproject/sha2": "^5.8.0",
        "@ethersproject/strings": "^5.8.0",
        "@ethersproject/transactions": "^5.8.0",
        "@ethersproject/web": "^5.8.0",
        "bech32": "1.1.4",
```

```
        "ws": "8.18.0"
      }
    },
    "node_modules/@ethersproject/random": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/random/-/random-
5.8.0.tgz",
      "integrity":                                          "sha512-
E4I5TDl7SVqyg4/kkA/qTfuLWAQGXmSOgYyO01So8hLfwgKvYK5snIlzxJMk72IFdG
/7oh8yuSqY2KX7MMwg+A==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/logger": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/rlp": {
      "version": "5.8.0",
      "resolved":  "https://registry.npmjs.org/@ethersproject/rlp/-
/rlp-5.8.0.tgz",
      "integrity":                                          "sha512-
LqZgAznqDbiEunaUvykH2JAoXTT9NV0Atqk8rQN9nx9SEgThA/WMx5DnW8a9FOufo/
/6FZOCHZ+XiClzgbqV9Q==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/logger": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/sha2": {
```

```
      "version": "5.8.0",
      "resolved": "https://registry.npmjs.org/@ethersproject/sha2/-
/sha2-5.8.0.tgz",
      "integrity":                                          "sha512-
dDOUrXr9wF/YFltgTBYS0tKslPEKr6AekjqDW2dbn1L1xmjGR+9GiKu4ajxovnrDbw
xAKdHjW8jNcwfz8PAz4A==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/logger": "^5.8.0",
        "hash.js": "1.1.7"
      }
    },
    "node_modules/@ethersproject/signing-key": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/signing-key/-/signing-
key-5.8.0.tgz",
      "integrity":                                          "sha512-
LrPW2ZxoigFi6U6aVkFN/fa9Yx/+4AtIUe4/HACTvKJdhm0eeb107EVCIQcrLZkxaS
Igc/eCrX8Q1GtbH+9n3w==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/logger": "^5.8.0",
        "@ethersproject/properties": "^5.8.0",
        "bn.js": "^5.2.1",
        "elliptic": "6.6.1",
        "hash.js": "1.1.7"
      }
    },
```

```
    "node_modules/@ethersproject/solidity": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/solidity/-/solidity-
5.8.0.tgz",
      "integrity":                                          "sha512-
4CxFeCgmIWamOHwYN9d+QWGxye9qQLilpgTU0XhYs1OahkclF+ewO+3V1U0mvpiuQx
m5EHHmv8f7ClVII8EHsA==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/bignumber": "^5.8.0",
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/keccak256": "^5.8.0",
        "@ethersproject/logger": "^5.8.0",
        "@ethersproject/sha2": "^5.8.0",
        "@ethersproject/strings": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/strings": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/strings/-/strings-
5.8.0.tgz",
      "integrity":                                          "sha512-
qWEAk0MAvl0LszjdfnZ2uC8xbR2wdv4cDabyHiBh3Cldq/T8dPH3V4BbBsAYJUeonw
D+8afVXld274Ls+Y1xXg==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/constants": "^5.8.0",
        "@ethersproject/logger": "^5.8.0"
```

```
      }
    },
    "node_modules/@ethersproject/transactions": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/transactions/-
/transactions-5.8.0.tgz",
      "integrity":                                          "sha512-
UglxSDjByHG0TuU17bDfCemZ3AnKO2vYrL5/2n2oXvKzvb7Cz+W9gOWXKARjp2URVw
cWlQlPOEQyAviKwT4AHg==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/address": "^5.8.0",
        "@ethersproject/bignumber": "^5.8.0",
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/constants": "^5.8.0",
        "@ethersproject/keccak256": "^5.8.0",
        "@ethersproject/logger": "^5.8.0",
        "@ethersproject/properties": "^5.8.0",
        "@ethersproject/rlp": "^5.8.0",
        "@ethersproject/signing-key": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/units": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/units/-/units-
5.8.0.tgz",
      "integrity":                                          "sha512-
lxq0CAnc5kMGIiWW4Mr041VT8IhNM+Pn5T3haO74XZWFulk7wH1Gv64HqE96hT4a7i
iNMdOCFEBgaxWuk8ETKQ==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
```

```
      "license": "MIT",
      "dependencies": {
        "@ethersproject/bignumber": "^5.8.0",
        "@ethersproject/constants": "^5.8.0",
        "@ethersproject/logger": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/wallet": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/wallet/-/wallet-
5.8.0.tgz",
      "integrity":                                          "sha512-
G+jnzmgg6UxurVKRKvw27h0kvG75YKXZKdlLYmAHeF32TGUzHkOFd7Zn6QHOTYRFWn
fjtSSFjBowKo7vfrXzPA==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/abstract-provider": "^5.8.0",
        "@ethersproject/abstract-signer": "^5.8.0",
        "@ethersproject/address": "^5.8.0",
        "@ethersproject/bignumber": "^5.8.0",
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/hash": "^5.8.0",
        "@ethersproject/hdnode": "^5.8.0",
        "@ethersproject/json-wallets": "^5.8.0",
        "@ethersproject/keccak256": "^5.8.0",
        "@ethersproject/logger": "^5.8.0",
        "@ethersproject/properties": "^5.8.0",
        "@ethersproject/random": "^5.8.0",
        "@ethersproject/signing-key": "^5.8.0",
        "@ethersproject/transactions": "^5.8.0",
        "@ethersproject/wordlists": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/web": {
      "version": "5.8.0",
      "resolved":  "https://registry.npmjs.org/@ethersproject/web/-
/web-5.8.0.tgz",
      "integrity":                                          "sha512-
j7+Ksi/9KfGviws6Qtf9Q7KCqRhpwrYKQPs+JBA/rKVFF/yaWLHJEH3zfVP2plVu+e
ys0d2DlFmhoQJayFewcw==",
      "funding": [
```

```
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/base64": "^5.8.0",
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/logger": "^5.8.0",
        "@ethersproject/properties": "^5.8.0",
        "@ethersproject/strings": "^5.8.0"
      }
    },
    "node_modules/@ethersproject/wordlists": {
      "version": "5.8.0",
      "resolved":
"https://registry.npmjs.org/@ethersproject/wordlists/-/wordlists-
5.8.0.tgz",
      "integrity":                                          "sha512-
2df9bbXicZws2Sb5S6ET493uJ0Z84Fjr3pC4tu/qlnZERibZCeUVuqdtt+7Tv9xxhU
xHoIekIA7avrKUWHrezg==",
      "funding": [
        {
          "type": "individual",
          "url":   "https://gitcoin.co/grants/13/ethersjs-complete-
simple-and-tiny-2"
        },
        {
          "type": "individual",
          "url": "https://www.buymeacoffee.com/ricmoo"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "@ethersproject/bytes": "^5.8.0",
        "@ethersproject/hash": "^5.8.0",
        "@ethersproject/logger": "^5.8.0",
        "@ethersproject/properties": "^5.8.0",
        "@ethersproject/strings": "^5.8.0"
      }
    },
    "node_modules/@humanfs/core": {
      "version": "0.19.1",
      "resolved": "https://registry.npmjs.org/@humanfs/core/-/core-
0.19.1.tgz",
```

```
      "integrity":                                          "sha512-
5DyQ4+1JEUzejeK1JGICcideyfUbGixgS9jNgex5nqkW+cY7WZhxBigmieN5Qnw9Zo
sSNVC9KQKyb+GUaGyKUA==",
      "dev": true,
      "license": "Apache-2.0",
      "engines": {
        "node": ">=18.18.0"
      }
    },
    "node_modules/@humanfs/node": {
      "version": "0.16.6",
      "resolved": "https://registry.npmjs.org/@humanfs/node/-/node-
0.16.6.tgz",
      "integrity":                                          "sha512-
YuI2ZHQL78Q5HbhDiBA1X4LmYdXCKCMQIfw0pw7piHJwyREFebJUvrQN4cMssyES6x
+vfUbx1CIpaQUKYdQZOw==",
      "dev": true,
      "license": "Apache-2.0",
      "dependencies": {
        "@humanfs/core": "^0.19.1",
        "@humanwhocodes/retry": "^0.3.0"
      },
      "engines": {
        "node": ">=18.18.0"
      }
    },

"node_modules/@humanfs/node/node_modules/@humanwhocodes/retry": {
      "version": "0.3.1",
      "resolved":
"https://registry.npmjs.org/@humanwhocodes/retry/-/retry-
0.3.1.tgz",
      "integrity":                                          "sha512-
JBxkERygn7Bv/GbN5Rv8Ul6LVknS+5Bp6RgDC/O8gEBU/yeH5Ui5C/OlWrTb6qct7L
jjfT6Re2NxB0ln0yYybA==",
      "dev": true,
      "license": "Apache-2.0",
      "engines": {
        "node": ">=18.18"
      },
      "funding": {
        "type": "github",
        "url": "https://github.com/sponsors/nzakas"
      }
    },
    "node_modules/@humanwhocodes/module-importer": {
      "version": "1.0.1",
      "resolved":
"https://registry.npmjs.org/@humanwhocodes/module-importer/-
/module-importer-1.0.1.tgz",
      "integrity":                                          "sha512-
bxveV4V8v5Yb4ncFTT3rPSgZBOpCkjfK0y4oVVVJwIuDVBRMDXrPyXRL988i5ap9m9
bnyEEjWfm5WkBmtffLfA==",
```

```
      "dev": true,
      "license": "Apache-2.0",
      "engines": {
        "node": ">=12.22"
      },
      "funding": {
        "type": "github",
        "url": "https://github.com/sponsors/nzakas"
      }
    },
    "node_modules/@humanwhocodes/retry": {
      "version": "0.4.3",
      "resolved":
"https://registry.npmjs.org/@humanwhocodes/retry/-/retry-
0.4.3.tgz",
      "integrity":                                    "sha512-
bV0Tgo9K4hfPCek+aMAn81RppFKv2ySDQeMoSZuvTASywNTnVJCArCZE2FWqpvIatK
u7VMRLWlR1EazvVhDyhQ==",
      "dev": true,
      "license": "Apache-2.0",
      "engines": {
        "node": ">=18.18"
      },
      "funding": {
        "type": "github",
        "url": "https://github.com/sponsors/nzakas"
      }
    }
```