Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий - РТФ

Базовая кафедра «Аналитика больших данных и методы видеоанализа»

ДОПУСТИТЬ К ЗАЩИТЕ ПЕРЕД ГЭК

/ Зав. кафедрой «АБД и МВ»

_____ М.А. Медведева

02.06.2023

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

«Разработка децентрализованного приложения для проведения лотерей»

Научный руководитель: Медведев А.Н.
доцент, к.-т.н.
Научный руководитель: Медведева М.А.
Доцент, к.ф.-м.н.

Студент группы РИМ-210981 Демин Д. Н.

Екатеринбург
2023

РЕФЕРАТ

Тема магистерской диссертации:
«Разработка децентрализованного приложения для проведения лотерей»

Магистерская диссертация выполнена на 85 страницах, содержит 3 таблицы, 55 рисунков, 65 использованных источников.

Децентрализованные сервисы становятся все более актуальными в современном мире, поскольку они предоставляют пользователям совершать финансовые транзакции без посредников. Благодаря децентрализации, такие сервисы более устойчивы к кибератакам и регулированию со стороны корпораций и государств.

Целью магистерской работы является разработка онлайн-сервиса для автоматического проведения лотерей в блокчейн-сети и приобретение навыков использования блокчейн-технологий.

Задачи работы:

1) провести литературно-аналитический обзор по следующим темам: блокчейн-технологии, криптовалюты, смарт-контракты, децентрализованные сервисы;
2) изучить применение блокчейн-технологий в разработке онлайн-сервисов, работу с криптовалютными кошельками, взаимодействие со смарт-контрактами;
3) разработать проект децентрализованного сервиса по проведению лотерей с помощью полученных навыков и знаний.

Объектом исследования являются блокчейн-технологии и децентрализованные сервисы, предметом - децентрализованный сервис для проведения лотерей.

Практическая значимость работы состоит в разработанном проекте децентрализованного приложения для проведения лотерей и разработке

материалов учебного курса и методического пособия по созданию децентрализованных сервисов для студентов вузов.

Экономическая эффективность заключается в экономии на инфраструктуре. Децентрализованный сервис не требует дорогостоящей инфраструктуры и обладает более гибкой масштабируемостью. Также, в рамках данной работы была представлена методика по созданию децентрализованных приложений, что позволяет сэкономить на покупке платных обучающих курсов.

Исследование проводилось с целью изучения блокчейн-технологий.

Для разработки децентрализованного приложения использовался блокчейн Ethereum, а также фронтенд-фреймворк React.

В результате был разработан смарт-контракт для проведения лотерей, а также веб-интерфейс для взаимодействия с ним, образующие децентрализованное приложение.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Децентрализованные финансовые сервисы (DeFi) — это новое поколение финансовых сервисов, ориентированных на потребителя, составленных в виде смарт-контрактов, развернутых на основе технологий блокчейна. Быстрый рост популярности этих сервисов привел к тому, что общая стоимость заблокированных активов (TVL), выросла с 675 млн долларов в начале 2020 года до более чем 40 млрд долларов к концу первого квартала следующего года [21]. График представлен на рисунке 1.

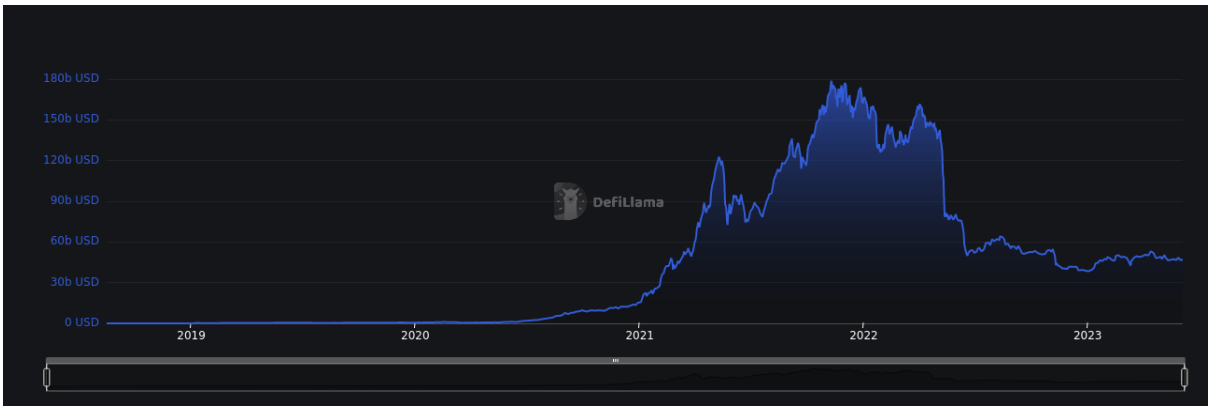

Рисунок 1 – TVL в DeFi

Децентрализованные приложения (DApps) являются одним из самых актуальных технологических трендов в настоящее время, а также основой децентрализованных финансов. Они представляют собой приложения, которые работают на блокчейне и не имеют единого централизованного управления. Одной из главных причин актуальности DApps является их высокий уровень безопасности. Благодаря технологии блокчейн, данные в DApps хранятся в распределенной системе, что делает их невозможными для взлома или изменения.

Также DApps позволяют создавать новые модели бизнеса и экономические системы, которые могут быть более эффективными и справедливыми, чем традиционные модели. Например, благодаря использованию токенов и умных контрактов, DApps могут автоматизировать процессы и упростить

взаимодействие между участниками. Кроме того, DApps могут применяться в различных сферах, от финансов до здравоохранения и образования. Они могут улучшить процессы и повысить качество предоставляемых услуг.

В целом, децентрализованные приложения имеют большой потенциал для изменения многих аспектов нашей жизни и являются одним из ключевых направлений развития технологий в ближайшее время. Децентрализованные финансовые сервисы по мере своего роста и развития, также создают новые инструменты, позволяющие найти новые способы заработка. Авторы Kaihua Qin, Liyi Zhou, Yaroslav Afonin, Ludovico Lazzaretti, Arthur Gervais в статье CeFi vs. DeFi -- Comparing Centralized to Decentralized Finance [22] сравнивают традиционную и децентрализованную системы.

Цель работы состоит в изучении блокчейн технологий, в частности, разработке смарт-контрактов, а также в создании децентрализованного приложения.

Также, были поставлены следующие задачи:

1) провести литературно-аналитический обзор по следующим темам: блокчейн-технологии, криптовалюты, смарт-контракты, децентрализованные сервисы;
2) изучить применение блокчейн-технологий в разработке онлайн-сервисов, работу с криптовалютными кошельками, взаимодействие со смарт-контрактами;
3) разработать проект децентрализованного сервиса по проведению лотерей с помощью полученных навыков и знаний.

Объектом исследования являются блокчейн-технологии и децентрализованные сервисы, предметом - децентрализованный сервис для проведения лотерей и пособие по блокчейн-технологиям.

Степень разработанности темы можно определить, проанализировав статьи российских и зарубежных ученых, а также их статистику. На рисунке 2 представлена статистика количества публикаций, содержащих ключевые слова “Decentralized application”, “Blockchain”, “Smart contract”. Можно заметить, что

количество публикаций растет с каждым годом, что может свидетельствовать о положительном тренде популярности данной темы.

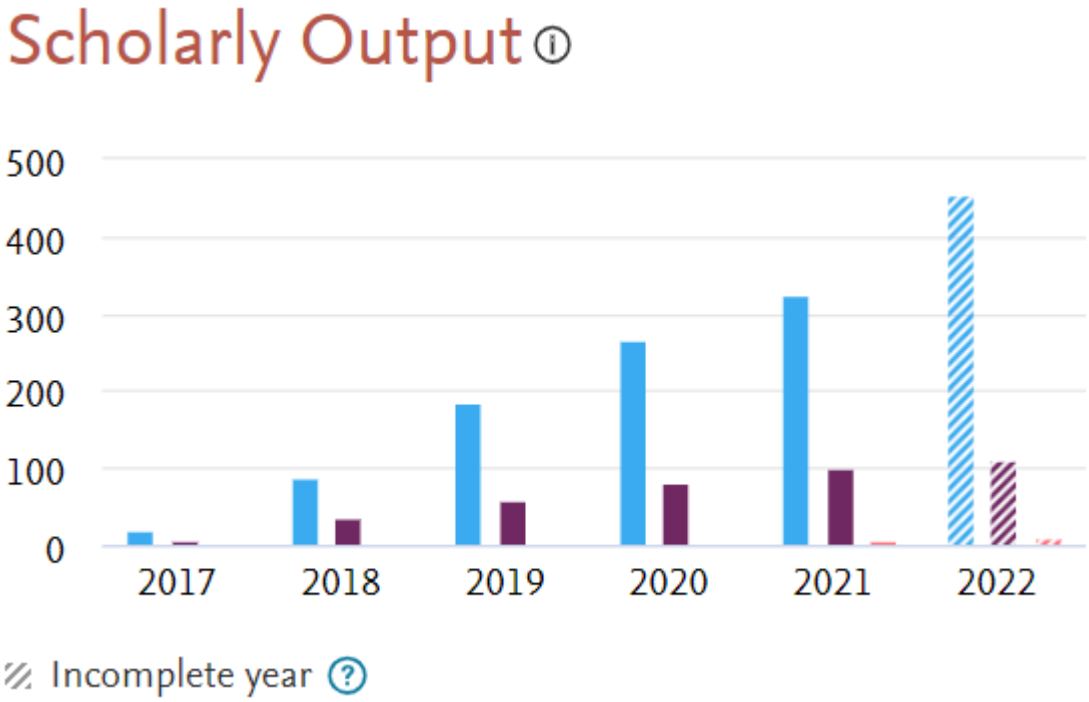

Рисунок 2 – Количество публикаций в SciVal

Научная новизна работы характеризуется методическим пособием по созданию децентрализованных сервисов, которое может использоваться в дальнейшем для интеграции блокчейн-технологий в сферу образования, а также вкладом в развитие блокчейн-технологий.

Практическая значимость работы состоит в разработанном проекте децентрализованного приложения для проведения лотерей. Данный проект может использоваться как пример для создания приложений в сфере DeFi. Кроме того, разработанный сервис возможно улучшить и сделать более доступным для пользователей, получив возможность зарабатывать комиссии на регистрации лотерей.

Основные положения, представленные на защиту:

1. Сравнительный обзор и анализ блокчейн-технологий и децентрализованных сервисов.

2. Авторская методология применения блокчейн-технологий в разработке децентрализованных сервисов.
3. Проект децентрализованного приложения для проведения лотерей.

Внедрение результатов исследования – научная статья.

Апробация результатов исследования – основные положения диссертационного исследования были представлены на I Международной научно-практической конференции молодых ученых и специалистов “Аналитический клуб: актуальные вызовы и решения”.

Структура диссертации определяется целью и задачами, состоит из введения, трех глав, заключения, списка использованной литературы, включающего 65 источников. Объем работы составляет 85 страниц основного текста, содержащего 3 таблицы и 55 рисунков.

Во введении обосновывается актуальность темы исследования, формулируются цель и задачи, степень разработанности темы, научная и практическая значимость работы.

В первой главе проводится обзор научной литературы по блокчейн-технологиям, изучаются принципы работы децентрализованных приложений, а также сравниваются различные блокчейн-сети.

Во второй главе приводится методология применения блокчейн-технологий в разработке децентрализованных сервисов, которая включает в себя использование криптовалютных кошельков и создание смарт-контрактов, а также публикация контрактов в блокчейне.

В третьей главе описываются основные этапы разработки децентрализованного приложения, в которые входят разработка смарт-контракта, фронтенда, а также их тестирование.

В заключении представлены основные результаты и выводы, полученные на основе диссертационного исследования.

1 ОБЗОР И АНАЛИЗ БЛОКЧЕЙН-ТЕХНОЛОГИЙ И ДЕЦЕНТРАЛИЗОВАННЫХ СЕРВИСОВ

1.1 Обзор практических работ и исследований по теме блокчейн-технологий

Ниже приводится обзор и анализ исследований и практических работ в теме блокчейн-технологий и децентрализованных сервисов.

Прежде чем рассматривать уже разработанные децентрализованные сервисы, необходимо изучить предметную область, а также исследования о проблемах в данной сфере.

В статье “CeFi vs. DeFi — Comparing Centralized to Decentralized Finance” [22] авторы сравнивают централизованные и децентрализованные финансовые сервисы, акцентируя внимание на трех аспектах: юридический, прикладной и экономический.

Одним из юридических аспектов финансовых систем является процедура KYC (Know Your Customer), которая требует предоставления документов для идентификации личности. Эта процедура является обязательной, если пользователь захочет воспользоваться, например, банковским продуктом. В DeFi пользователь с технической стороны может не подтверждать свою личность и пользоваться сервисами. К тому же KYC может занимать от нескольких часов до нескольких недель, в то время как создать криптовалютный кошелек может любой пользователь без каких-либо подтверждений с другой стороны.

В следующей главе авторы сравнивают CeFi и DeFi на уровне приложений и сервисов. Схема из статьи представлена на рисунке 3 [22].

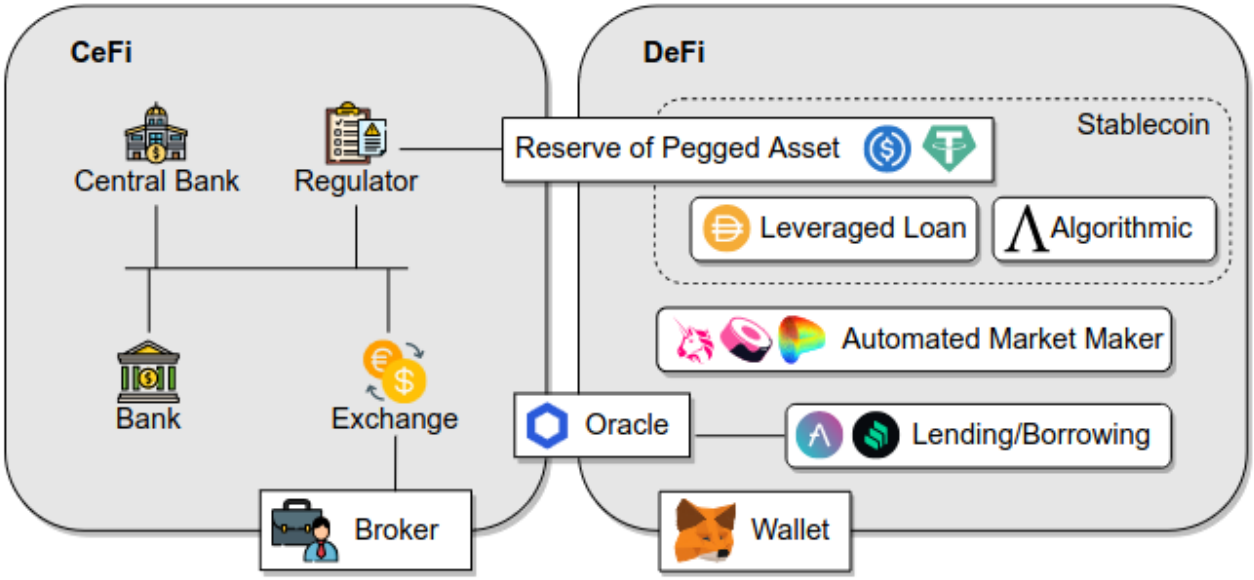

Рисунок 3 – CeFi и DeFi

Биржа — это торговая площадка, на которой торгуются финансовые инструменты. Исторически это могло происходить в физическом месте, где трейдеры встречаются для ведения бизнеса, например на Нью-Йоркской фондовой бирже. В последние десятилетия торговля перешла на централизованные электронные биржи [22]. Отличия представлены в таблице 1, где рассматриваются централизованные, гибридные и децентрализованные биржи, а также их характеристики.

Таблица 1 – Отличия централизованных, децентрализованных и гибридных бирж [22]

	Централизованная (CEX)		Гибридная	Децентрализованная(DEX)		
Название биржи	NASDAQ	Coinbase	IDEX	0x	Tesseract	Uniswap

Продолжение таблицы 1 – Отличия централизованных, децентрализованных и гибридных бирж [22]

	Централизованная (CEX)		Гибридная	Децентрализованная(DEX)		
Валюты	USD	Fiat +	Crypto	Crypto	Crypto -	Crypto
Управление	Централизованно	Централизованно	Централизованно	DAO	Централизованно	DAO + smart contract
Механизм определения цены	Централизованно	Централизованно	Централизованно	Децентрализованно	TEE	Смарт-контракт
Механизм сопоставления сделок	Централизованно	Централизованно	Централизованно	Децентрализованно	TEE	Смарт-контракт
Система клиринга	Централизованно	Централизованно	Блокчейн	Блокчейн	Блокчейн	Блокчейн
Возможность манипуляции транзакцией	Регулируется	Регулируется	Да	Да	Нет	Да (майнеры)

Окончание таблицы 1 – Отличия централизованных, децентрализованных и гибридных бирж [22]

	Централизованная (CEX)		Гибридная	Децентрализованная(DEX)		
Возможность отклонить транзакциию	Регулируется	Регулируется	Да	Да	Нет	Да (майнеры)

В централизованных финансовых сервисах (CeFi) существует система кредитов и займов, которая позволяет делать покупки, откладывая платеж на будущее. Кредитор несет риски, когда выдает кредит, так как заемщик может поступить недобросовестно и не возвращать деньги обратно. Кредитоспособность — это измерение или оценка платежеспособности заемщика. Обычно она рассчитывается, например, исходя из истории погашения и получаемого дохода, если речь идет о кредите. В CeFi как кредиторами, так и заемщиками могут быть физические лица, государственные или частные группы или финансовые учреждения.

В DeFi отсутствие системы проверки кредитоспособности и инструментов правоприменения в случае неисполнения обязательств приводит к необходимости чрезмерного обеспечения в большинстве протоколов кредитования и заимствования (например, Aave, Compound). Чрезмерное обеспечение означает, что заемщик обязан предоставить обеспечение, стоимость которого превышает непогашенные долги.

Также, в DeFi существует такое понятие, как Flash Loan (срочный кредит). Срочный кредит инициируется и погашается в рамках одной атомной блокчейн-транзакции, в ходе которой заемщик B выполняет следующие три действия:

1) B запрашивает активы из пула срочных займов.
2) B волен использовать заемные средства произвольно.
3) B выплачивает срочный заем плюс проценты кредитному пулу.

Такой инструмент как срочный кредит широко используется в DeFi, однако, чтобы им воспользоваться, необходимо разработать программное обеспечение, ведь транзакция занимает всего несколько секунд.

Третий аспект, который рассматривают авторы в статье — это экономический аспект. Одним из ключевых понятий, свойственным моделям CeFi и DeFi является инфляция-девальвация существующего предложения валюты за счет увеличения предложения. Официальной целью политики центральных банков на развитых рынках является поддержание инфляции на уровне или около 2.0%. Уровень инфляции Ethereum составляет около 4%.

Далее, авторы статьи приводят примеры различных атак на сервисы CeFi и DeFi, говоря о безопасности и приватности. Также, важно упомянуть о манипуляции рынками.

В последней главе авторы затрагивают тему синергии CeFi и DeFi и подводят итоги, говоря, что каждая из моделей имеет свои преимущества и недостатки. Авторы надеются, что DeFi и CeFi будут сосуществовать вместе, дополняя и улучшая друг друга.

Теперь, когда известны отличия между DeFi и CeFi, можно перейти к обзору статей, практическим результатом которых являются разработанные сервисы.

В статье “Decentralized application platform for private key management” [61] рассматривается создание децентрализованной платформы для управления

закрытыми ключами. Авторы используют блокчейн для создания децентрализованного приложения, однако сфера применения такого сервиса - информационная безопасность. Платформа используется для аутентификации пользователей, а в самой работе не описываются шаги по созданию такой платформы, показана лишь архитектура сервиса.

Следующая статья “SYSTEM, METHOD, AND DECENTRALIZED APPLICATION FOR BLOCKCHAIN - BASED GAMBLING” [62] более похож на тему данной работы тем, что совпадает как технология, так и сфера применения. Гемблинг - азартная игра, поэтому децентрализованное приложение для проведения лотерей также можно характеризовать как гемблинг. Схемы, представленные в данном патенте, описывают часть процессов проекта, полученного в результате третьей главы, однако, в работе не описывается методика по созданию такого приложения, а приводятся лишь схемы работы приложения и последовательность действий в различных условиях.

В статье “System and method for implementing a blockchain-based decentralized application” рассматривается методика для реализации децентрализованных приложений. Авторы представляют методику разработки приложений на сети блокчейн, однако некоторые важные аспекты, вроде создания и использования криптовалютного кошелька, упущены и предполагается, что читатель уже знаком с этими процессами.

По результатам проведенного анализ сделаны следующие основные выводы: децентрализованные сервисы имеют ряд преимуществ перед традиционной финансовой моделью, например, отсутствие идентификации пользователя с помощью KYC и AML проверок. Если сравнивать централизованные и децентрализованные биржи, различаются механизмы управления, определения цены актива и т.д. Также, сама технология блокчейн менее подвержена кибератакам, однако ее минусом является отсутствие приватности, так как все транзакции видны для всех пользователей, что позволяет отследить отправителя.

Рассматриваемые статьи с практическим результатом показывают возможные способы применения блокчейн-технологий в разработке децентрализованных приложений, а также отражают проблемы, с которыми столкнулись авторы в ходе исследования.

1.2 Блокчейн-технологии

По определению, блокчейн — это непрерывно растущая цепочка блоков, каждый из которых содержит криптографический хэш предыдущего блока, временную метку и передаваемые данные [19]. Благодаря существованию криптографического хэша данные, хранящиеся в блокчейне, по своей сути устойчивы к модификации: если один блок данных изменен, все последующие блоки должны быть восстановлены с новыми значениями хэша. Эта особенность неизменяемости является фундаментальной для блокчейн-приложений [18].

Блокчейн распределен, это означает, что его копии сохранены на множестве компьютеров, и все они должны совпадать, чтобы он был действительным. Блокчейн собирает информацию о транзакциях и вводит ее в блок, подобный ячейке в электронной таблице, содержащей информацию. Как только он заполняется, информация проходит через алгоритм шифрования, который создает шестнадцатеричное число, называемое хэшем.

Затем хэш вводится в заголовок следующего блока и шифруется вместе с другой информацией в блоке. Это создает серию блоков, которые соединены цепочкой вместе [9]. Этот процесс показан на рисунке 4.

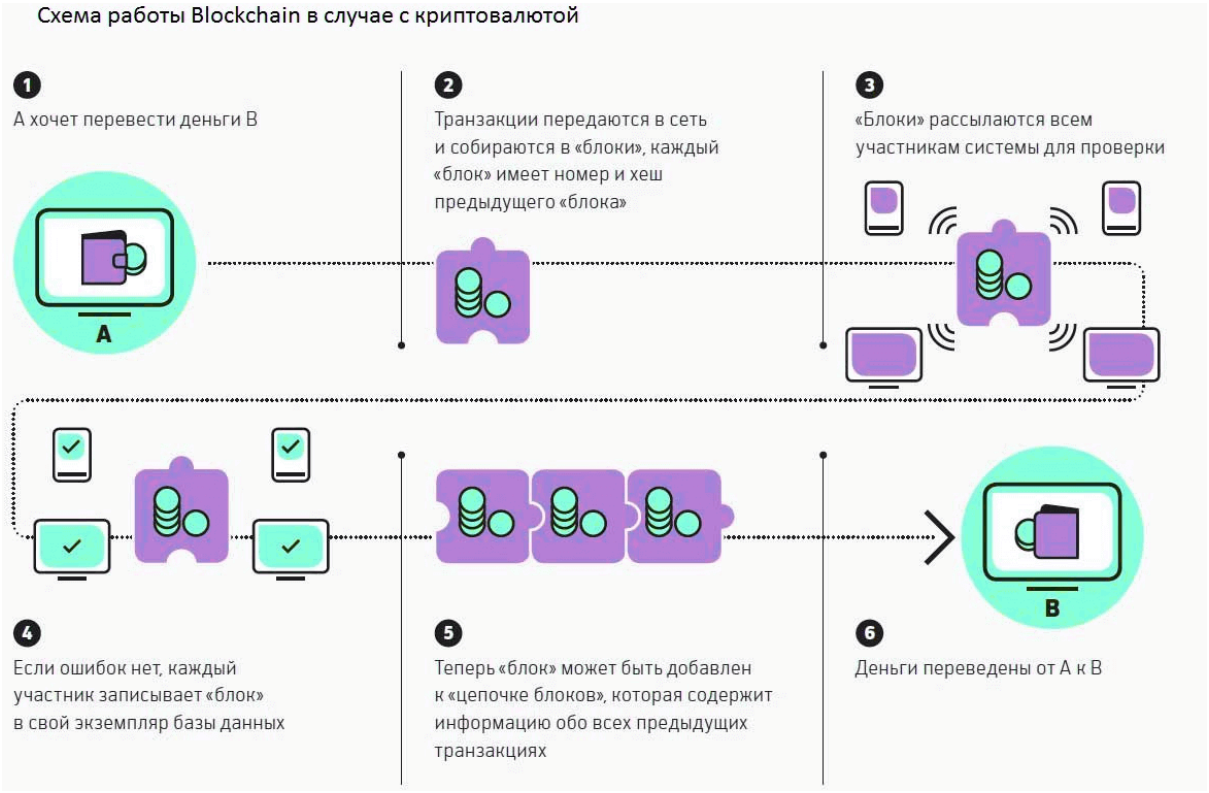

Рисунок 4 – Схема работы блокчейна [17, с.5]

В статье “Технология блокчейн и ее практическое применение” [23]. Авторы рассматривают положительные аспекты и возможности применения технологии блокчейн:

Платежи. Банки и платежные сервисы берут комиссию за все переводы или платежи, хоть рядовому пользователю может и казаться, что это бесплатно. К тому же есть возможность хищения средств злоумышленниками или посредником. Блокчейн позволяет избежать таких случаев, так как все транзакции фиксируются у других участников цепи. Также, блокчейн представляет возможность условных платежей, что позволяет разрабатывать сервисы с запрограммированными сценариями.

Бонусы. Здесь авторы выделяют преимущество в том, что накопления не тратятся без ведома пользователя, однако, стоит добавить, что владельцы таких бонусов могут иметь преимущества и привилегии, в зависимости от сети

блокчейн. Некоторые сети, например Uniswap, предоставляют право голоса за владение токенами.

Аутентификация. Технология блокчейн подходит для идентификации пользователя, так как обладает высокой надежностью и защищенностью.

Реестр имущества. Авторы акцентируют внимание на доступности и актуальности реестра в сети блокчейн, а также достоверности данных. Стоит отметить, что при неправильной архитектуре реестра, стоимость его использования и хранения может быть довольно высокой по сравнению с централизованной архитектурой.

Таким образом, технология блокчейн имеет широкий спектр возможностей для применения в современном мире. Чтобы понять, как работает блокчейн изнутри, необходимо разобраться в алгоритмах консенсуса и криптографии.

1.1.1 Алгоритмы консенсуса

В каждом блокчейне есть алгоритм, который позволяет подтвердить новую транзакцию и признать ее правильной. Такой алгоритм называется консенсусом.

Алгоритм консенсуса блокчейна — это способ, благодаря которому децентрализованные ноды сети достигают согласия о текущем состоянии данных во всех блоках. Согласованные алгоритмы позволяют набору машин работать как слаженная группа, которая может пережить сбои некоторых из своих участников [24]. Нода — это любой компьютер, подключенный к блокчейну, который проверяет и подтверждает транзакции, и хранит копию блокчейна.

Есть несколько наиболее популярных алгоритмов консенсуса в сетях блокчейн: Proof-of-Work, Proof-of-Stake и гибридные варианты. Рассмотрим их подробнее.

Proof-of-Work (доказательство выполнения работы, PoW) — алгоритм консенсуса, который впервые был представлен в сети первой криптовалюты Bitcoin. Суть PoW заключается в следующем: ноды сети (майнеры), чтобы подтвердить транзакции и не позволить другим участникам расходовать одни и те же монеты дважды, должны решать сложные математические задачи (хэш-функции). Нода, которая первой нашла решение, получает вознаграждение — новые монеты сети. При этом сложность майнинга защищает сеть от возможных угроз в виде DDoS-атак, атаки 51% (когда злоумышленники получают контроль над подтверждением транзакций и созданием блоков) и других видов атак. Если бы задачи были слишком легкими, злоумышленники могли бы легко взломать сеть.

PoW стал прорывом для своего времени и позволил запустить первые криптовалюты. Он делает сеть децентрализованной и устойчивой ко взломам. Так, Bitcoin практически невозможно взломать — понадобился бы квантовый компьютер и гигантские ресурсы для получения контроля над сетью.

Но по мере популяризации криптовалют и их массового внедрения недостатки этого алгоритма начали становиться все заметнее. Более того, сейчас эти уязвимости мешают и ограничивают развитие ранних проектов криптовалют.

Proof-of-Stake – алгоритм консенсуса для подтверждения новых транзакций в сети блокчейн, который выбирает валидатора в зависимости от количества его криптовалюты. Простыми словами, Proof-of-Stake — это возможность получать вознаграждение за счет владения определенной монетой. Самый известный пример блокчейна на алгоритме PoS – Ethereum. Плюсом этого алгоритма является энергоэффективность, ведь PoS, в отличие от PoW, не требует мощного оборудования для майнинга. Транзакции

подтверждают валидаторы – ноды, которые заблокировали определенное количество монет для получения права поддерживать работоспособность сети и получать за это вознаграждение. Сам процесс блокирования некоторого количества монет называется стейкинг (от англ. stake — «доля») — способ получения пассивного дохода от криптовалют, работающих на алгоритме консенсуса Proof-of-Stake (PoS) и его разновидностях. Минусом данного алгоритма можно назвать привилегии для высоких стейков. PoS основывается на размере стейка валидатора. Высокие стейки увеличивают шансы валидатора быть выбранным по сравнению с менее «богатыми» нодами. Так появляется проблема привилегий.

Delegated Proof-of-Stake схож по работе с PoS, за исключением того, что он включает в себя механизм голосования и делегирования, чтобы стимулировать пользователей защищать сеть и проверять блоки с помощью монет, оставленных в качестве залога. Благодаря данному залогу пользователи могут участвовать в голосовании, где выбирают делегатов, которые будут ответственны за все аспекты проведения транзакций. Когда делегаты наконец избраны, важно, чтобы они смогли договориться о том, какие транзакции следует отклонить, а какие — одобрить. Применяется в Cosmos, Tron, EOS.

Делегаты — это пользователи в системе DPoS, которые контролируют управление в блокчейне. За делегатов голосуют другие пользователи. Делегат может предложить изменить размер отдельного блока или изменить количество свидетелей, которым платят за проверку блоков. Когда изменения предлагаются делегатом, пользователи блокчейна голосуют за принятие этих изменений.

Валидаторы — это узлы, которые могут проверять, что блоки, созданные разными свидетелями, соответствуют правилам консенсуса. Если пользователь становится валидатором блоков, все, что от него требуется, — это запустить валидатор и проверить сеть.

Свидетели — пользователи, которые несут ответственность за безопасность и проверку блокчейна. Чтобы стать свидетелем, пользователь должен получить достаточное количество голосов. Общее количество свидетелей на одном сервере может варьироваться от 21 до 101. Несмотря на то, что свидетель может удерживать определенные транзакции вне блока, он не может изменить информацию о транзакции. Среди свидетелей очень большая конкуренция, если свидетель теряет доверие пользователей, его место занимает другой.

Данный протокол отличается особой надежностью. Алгоритмы консенсуса лежат в основе каждой сети блокчейнов. Использование правильного алгоритма обеспечивает быструю и безопасную проверку транзакций, наряду с меньшим потреблением энергии, чем системы PoW, DPoS также требует меньше времени, чем системы PoS [2].

В исследовании “A Survey about Consensus Algorithms Used in Blockchain” авторы сравнивают алгоритмы консенсуса по нескольким критериям, которые представлены в таблице 2.

Таблица 2 – Сравнение алгоритмов консенсуса

Критерий	PoW	PoS	Гибридная форма PoW и PoS
Энергоэффективность	Нет	Да	Нет
Современное оборудование	Очень важно	Не требуется	Важно

Продолжение таблицы 2 – Сравнение алгоритмов консенсуса

Разветвление	Когда два узла находят подходящий одноразовый номер одновременно	Очень сложно	Возможно
Атака двойной траты	Да	Сложно	Да, но менее серьезно, чем в PoW
Скорость создания блока	Низкий, зависит от случая	Быстро	Низкий, зависит от случая
Pool Mining	Да, но может быть предотвращен	Да, тяжело предотвратить	Да
Пример	Bitcoin	Nextcoin	PPcoin, Blackcoin

Таким образом, можно отметить, что у каждого алгоритма консенсуса есть свои плюсы и минусы, а их разработка не стоит на месте, и появляются новые способы подтверждения транзакций в сети. Также, некоторые блокчейны переходят с одного консенсуса на другой, как, например, Ethereum, который разработчики перевели с PoW на PoS.

Проанализировав и сравнив алгоритмы консенсуса, можно сделать вывод, Proof-of-Stake больше подходит для разработки децентрализованных приложений, чем другие алгоритмы за счет своей энергоэффективности и меньших затрат.

1.1.2 Криптография в блокчейне

Криптография — это наука о безопасном общении в присутствии враждебного поведения. Хотя большинство людей понимает роль криптографии в сохранении конфиденциальности коммуникации (например, приложения для обмена сообщениями с шифрованием), она также используется для аутентификации происхождения, проверки целостности и установления безответности коммуникации; т. е. для определения того, что сообщение пришло от конкретного человека и не было подделано, и что эти свойства не могут быть опровергнуты отправителем.

Криптография используется для преобразования оригинального сообщения в неразборчивое сообщение, которое может интерпретировать только тот, кто в курсе. Шифрование — это двусторонняя функция, которая шифрует исходное сообщение (т. е. открытый текст) в неразборчивое сообщение (т. е. шифротекст), которое может расшифровать только получатель, чтобы извлечь открытый текст. Другие криптографические функции являются односторонними и направлены только на доказательство какого-либо знания или свойства данных, а не на то, чтобы сделать исходные данные известными.

Алгоритм шифрования называется шифром, а ключ — это секрет, обычно строка символов, которая позволяет кому-то понять шифр. В шифрах используются два основных типа техник: подстановка и перестановка (также называемая транспортировкой). Шифры подстановки заменяют буквы, цифры и символы открытого текста другими буквами, цифрами или символами. Шифры перестановки используют буквы открытого текста, но меняют их порядок. Для создания сложности эти методы часто комбинируются, наслаиваются друг на друга, подвергаются влиянию внешних данных и изменяются с течением времени.

Хэш-функции преобразуют данные любой длины в фиксированную битовую длину, называемую хэшем. Хэш — это уникальный идентификатор

данных, сродни отпечатку пальца, по сути, проверяющий исходный набор данных. Хеширование данных широко используется для индексации данных и эффективного извлечения их из базы данных. Оно также используется для безопасного хранения данных, например, в случае веб-сайтов, которые хранят только соленый хэш паролей пользователей, чтобы предотвратить утечку паролей в случае взлома базы данных.

Хеш-функции не используют ключи и основаны на односторонних функциях, что обеспечивает их устойчивость к предварительным образам - практически невозможно, чтобы кто-то, кто не знает исходного сообщения, смог вычислить его по одному лишь хешу. Единственным реальным методом получения оригинального сообщения является перебор, т. е. угадывание всех возможных входных данных, что в случае хэшей представляет собой неограниченный набор возможных комбинаций символов.

Самым популярным безопасным алгоритмом хэширования на сегодняшний день является SHA-256, который создает строки длиной 256 бит (т. е. 256 последовательных 1 и 0). 256-битные хэши часто представляются в шестнадцатеричном формате строк, поэтому их длина может составлять 32 или 64 символа. SHA-256 существует в рамках набора хэш-функций SHA-2, пришедшего на смену взломанному SHA-1. SHA-3 также был представлен в 2015 году на основе криптографического примитива Keccak.

Симметричное шифрование (оно же шифрование с использованием секретных ключей) предполагает использование общего секретного ключа, который отправитель и получатель используют для шифрования и расшифровки сообщений. Симметричное шифрование является быстрым и эффективным, но при этом возникает проблема безопасного обмена секретным ключом через Интернет. Хотя обмен можно осуществлять лично, этот метод не является глобально масштабируемым, поскольку требует формальных отношений с каждым контрагентом. Другая проблема заключается в том, что если секретный ключ скомпрометирован, то любой, кто его использует, рискует расшифровать

все предыдущие и будущие сообщения. Именно поэтому многие интернет-протоколы, использующие симметричное шифрование, такие как TLS, также используют протоколы обмена ключами для безопасного создания общего секретного ключа без его передачи через интернет.

Наиболее популярным симметричным шифром является Advanced Encryption System (AES). AES предлагает ключи длиной 128, 192 и 256 бит - существенное улучшение по сравнению с 56-битными ключами, используемыми в предыдущем и ныне небезопасном стандарте шифрования данных (DES). Для сравнения, 56-битный ключ имеет 256, или 72 квадриллиона, возможных ключей, которые можно взломать перебором менее чем за 24 часа. В противном случае, для того чтобы перебрать 2 в степени 128 возможных ключей, даже при объединении всех компьютеров мира, потребуется триллионы и триллионы лет.

Асимметричное шифрование (оно же криптография с открытым ключом) дает каждому пользователю пару открытого и закрытого ключей. Открытый ключ виден всем, в то время как закрытый ключ известен только его владельцу. Пользователи могут шифровать сообщения с помощью своего закрытого ключа, который может расшифровать любой человек с открытым ключом. Это известно как цифровая подпись, поскольку она доказывает знание секрета без раскрытия самого секрета (т. е. у пользователя есть закрытый ключ к адресу открытого ключа). Пользователи также могут шифровать сообщения с помощью чужого открытого ключа, который может расшифровать только человек с закрытым ключом (т. е. отправка конфиденциальной информации).

Наиболее популярными алгоритмами асимметричного шифрования являются RSA (названный в честь его изобретателей Рамиса, Шамира и Адлемана), ECC (Elliptic Curve Cryptography), Diffie Hellman (популярный протокол обмена ключами) и DSS (Digital Signature Standard). Некоторые алгоритмы асимметричного шифрования устойчивы к квантованию, а другие - нет, и в будущем их, возможно, придется модернизировать или отказаться от них [3].

В статье Research on the Application of Cryptography on the Blockchain [49] рассматриваются некоторые возможные проблемы с безопасностью в блокчейне. Например, во многих блокчейнах транзакции можно отследить, тем самым, идентифицировать пользователя криптовалюты, нарушая конфиденциальность. Однако, некоторые монеты, такие как Monero и Dash, при помощи криптографических алгоритмов затрудняют алгоритм идентификации пользователя или делают его практически невозможным.

Авторы статьи Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts [50] представляют модель блокчейна, которая решает проблемы с конфиденциальностью пользователей. Hawk - децентрализованная система смарт-контрактов, которая не хранит финансовые транзакции в открытом виде в блокчейне, тем самым сохраняя конфиденциальность транзакций от посторонних глаз. Программист Hawk может написать частный смарт-контракт интуитивно понятным способом без необходимости внедрения криптографии, а компилятор автоматически генерирует эффективный криптографический протокол, в котором стороны договора взаимодействуют с блокчейном, используя криптографические примитивы, такие как доказательства с нулевым разглашением [50].

В статье Blockchain Education [51] авторы показывают следующие способы применения криптографии в блокчейне:

1) шифрование данных;
2) предотвращение внесения изменений;
3) вставка новых блоков;
4) проверка данных в блоках.

Таким образом, криптография в блокчейне играет важную роль, выполняя множество задач, таких как шифрование данных и предотвращение изменений, однако существуют и проблемы, которые решены не во всех блокчейнах, например, конфиденциальность данных.

1.3 Типы криптовалют и блокчейн-сетей

Криптовалюта — это децентрализованная цифровая валюта, которая обеспечивает безопасность с помощью криптографии. Она может работать без каких-либо посредников вроде банков и платежных систем.

В статье Cryptocurrencies, Blockchain and Regulation: A Review [26] авторы отмечают бурный рост популярности криптовалюты, показывая рост количества транзакций в блокчейне Bitcoin. Данный рост показан на рисунке 5.

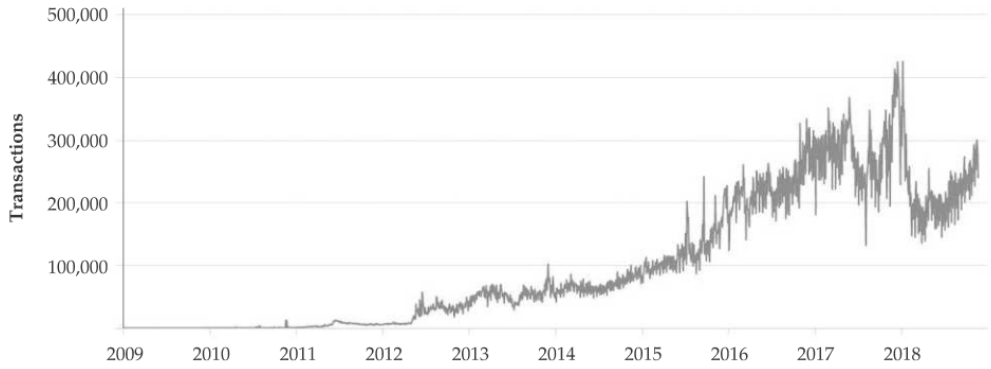

Рисунок 5 – График количества транзакций Bitcoin

Децентрализованная природа криптовалют позволяет осуществлять одноранговые (P2P) транзакции между пользователями. Однако вместо физических кошельков или банковских счетов владельцы криптовалют получают доступ к своим активам через криптокошельки или криптобиржи [12].

Принципы работы криптовалют:

1. Децентрализация - все действия выполняются участниками сети без привлечения единого центрального эмитента. Информация о средствах хранится во всех нодах, поэтому в случае проблем доступ к данным сохраняется.
2. Дефляция - при выпуске большинства криптовалют известна конечная эмиссия монет, что гарантирует ценность актива и отсутствие инфляции.

3. Защищенность - использование блокчейна и специальных алгоритмов позволяет защитить сеть от хакерских атак. Чем больше майнеров и нод у криптовалюты, тем сложнее осуществить атаку.
4. Анонимность - каждый пользователь получает номер кошелька, который никак не связан с его личностью. Но зная публичный ключ, можно посмотреть информацию о количестве денег на счете [13].

Все криптовалютные активы можно разделить по разным категориям. Bitcoin является первой криптовалютой, поэтому он имеет статус отдельного актива вне категорий. Остальные блокчейн-проекты можно разделить на несколько видов:

1. Альткоины (альтернативные коины) – любые криптовалюты со своим блокчейном (кроме биткоина). Некоторые из них похожи на биткоин. Другие ориентированы на внедрение и использование новых инструментов, а также расширение возможностей. При помощи изменения открытого исходного кода BTC разработчики альткоинов могут ускорять транзакции, оптимизировать процесс майнинга, создавать различные автоматизированные контракты, формировать базу для работы с криптоприложениями и т. д.
2. Токены – выполняют функцию цифровых активов, но не имеют собственного блокчейна, как стандартная криптовалюта. Вместо майнинга токены сразу выпускаются в полной эмиссии. Эти активы создают различные компании, с целью привлечения средств на развитие своих проектов или обеспечения работоспособности продуктов. Инвесторы, в свою очередь, получают гарантии того, что компания выполнит перед ними свои обязательства.
3. Стейблкоины – цифровые монеты, цена которых привязана к материальным активам – доллару, золоту, нефти и др. Курс биткоина и других подобных криптовалют изменяется в течение дня, недели и месяца. Стоимость стейблокинов напротив – предельно стабильна. Колебания

цены, конечно, могут быть, но они намного меньше, чем у криптовалют. Стейблкоины подходят тем, кто намерен перевести свои сбережения в цифровые активы с минимальными рисками.

4. NFT – так обозначаются невзаимозаменяемые токены (non-fungible token). Они были созданы для переноса в блокчейн прав на владение уникальным активами. Например, антиквариатом, произведениями искусства, 3-D моделями, игровыми предметами и прочим. Каждый токен является уникальным и его нельзя заменить другим. Особой популярности NFT заслуживают в сообществе коллекционеров [14].

Помимо представленных выше преимуществ криптовалют, существуют и риски, связанные с отсутствием регулирования. Например, авторы Ayesha Afzal and Aiman Asif в статье “Cryptocurrencies, Blockchain and Regulation: A Review.“ [26] выделяют следующие риски:

1. Черные рынки. Пользователь может не осознавать, что участвует в преступных схемах, используя криптовалюту.
2. Технологические рынки. Блокчейн требует использования большого количества графических процессоров, поэтому на технологическом рынке происходит дефицит оборудования.
3. Потребление энергии. Технология блокчейн очень энергозатратна, например, по данным Digiconomist, вся сеть Bitcoin потребляет 73 тераватт-часа (73 миллиона мегаватт-часов) электроэнергии [27].
4. Налогообложение и уклонение от уплаты налогов. Правительство вынуждено контролировать оборот криптовалюты.
5. Взломы. Сам блокчейн невосприимчив ко взлому благодаря алгоритмам консенсуса, однако, кибератаки возможны из-за развивающейся инфраструктуры вокруг блокчейна.
6. Пузырь спекуляций.
7. Волатильность.

1.4 Смарт-контракты

Смарт-контракт — это приложение, работающее на блокчейне. Оно выступает в качестве цифрового соглашения, подкрепляемого набором правил. Эти правила определяются компьютерным кодом, который копируют и обрабатывают все ноды сети.

Смарт-контракты позволяют создавать протоколы, не требующие доверия. Это означает, что обе стороны могут взаимодействовать через блокчейн без необходимости доверять друг другу. Участники процесса могут быть уверены, что несоблюдение условий контракта приведет к его аннулированию. Также использование смарт-контрактов избавляет от необходимости в посредниках, значительно снижая расходы на операции [10].

Умные контракты полностью цифровые и написаны на языке программирования. В дополнение к установлению обязательств и последствий таким же образом, как и в обычном физическом документе, код может выполняться автоматически. Следовательно, он может получать и обрабатывать информацию, касающуюся переговоров, уже принимая меры в соответствии с правилами договора [11].

Таблица 3 – Преимущества и недостатки смарт-контрактов

Преимущества	Недостатки
● Автономность технологии. ● Надежность проводимых операций. ● Высокая безопасность данных. ● Быстрота и экономичность. ● Точность работы контрактов	● В коде смарт-контрактов могут быть ошибки, допускающие махинации. ● Неразвит инструментарий, например программы-оракулы, связывающие цифровую экосистему с реальным миром.

Продолжение таблицы 3 – Преимущества и недостатки смарт-контрактов

Преимущества	Недостатки
	• Высокие требования к точности данных (их нельзя изменить после попадания в блокчейн). • Существуют проблемы масштабирования систем и скорости обработки транзакций.

В статье “An Overview on Smart Contracts: Challenges, Advances and Platforms” [28] авторы описывают множество вариантов применения смарт-контрактов, а также прикладывают схему для наглядности, которая представлена на рисунке 6.

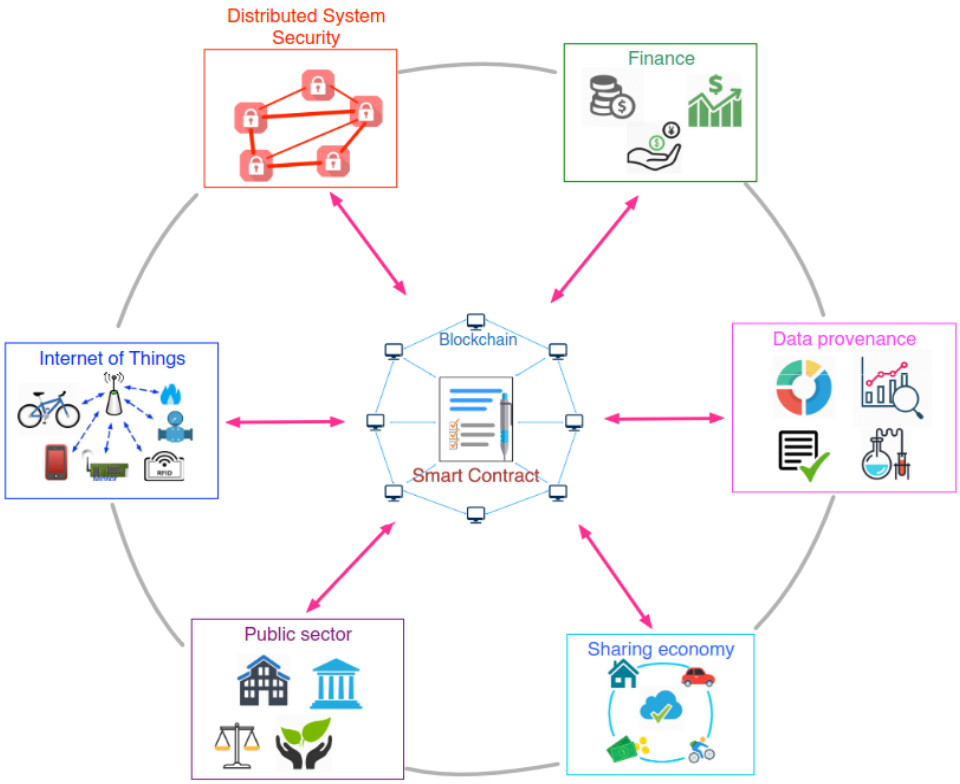

Рисунок 6 – Варианты использования смарт-контрактов

Рассмотрим каждый из вариантов по порядку:

1. Интернет вещей. В качестве примера авторы приводят встроенное ПО, которое для обновления обращается в централизованному серверу. При использовании смарт-контрактов ПО могло бы обращаться к блокчейну для получения данных об обновлении.
2. Защита распределенных систем. Авторы предлагают защиту от DDOS (Distributed Denial-of-Service) путем размещения в смарт-контракте IP-адреса атакующего, информируя всю сеть об атаке.
3. Финансы. Смарт-контракты позволяют реализовать различные банковские инструменты, а также, например, страховку.
4. Источник данных. Смарт-контракты могут помочь в идентификации и гарантии качества данных, расположенных в блокчейне, что может использоваться в научных исследованиях или медицине.
5. Экономика совместного использования. Смарт-контракты могут снизить стоимость комиссий в различных сценариях шеринг-экономики, например, при сдаче в аренду различных вещей.
6. Государственный сектор. Так как блокчейн обеспечивает прозрачность данных, смарт-контракты можно использовать для электронного голосования.

В исследовании “Smart contract development: Challenges and opportunities” [30] был проведен опрос среди разработчиков смарт-контрактов, который показал некоторые трудности разработки:

1. Например, 88,8% респондентов согласились с тем, что отлаживать приложения для смарт-контрактов сложно. Существует нехватка мощных интерактивных дебаггеров, а сообщения об ошибках не такие информативные, как хотелось бы.

2. Ограничения языка программирования. Большинство смарт-контрактов написаны на Solidity, в то время как в нем есть некоторые недостатки, такие как нехватка важных библиотек, нехватка стандартов/правил, отсутствие проверок безопасности для типов данных. В ходе опроса авторы обнаружили, что сами языки программирования являются основным препятствием при разработке смарт-контрактов. 39,7% респондентов, участвовавших в этом опросе, согласились с тем, что это одна из трех основных проблем.
3. Виртуальная машина Эфириума (EVM). Отсутствие поддержки популярных языков программирования, неэффективное выполнение байт-кода, ограничение размера стэка. 35.3% респондентов опроса согласились с тем, что недостатки EVM не позволяют эффективно разрабатывать смарт-контракты.
4. Газ. Газ — это комиссия, которую необходимо заплатить для взаимодействия со смарт-контрактом. При отклонении транзакции газ не возвращается, 35,3% респондентов часто сталкивались со сбоями транзакций, вызванными нехваткой газа.
5. Поддержка сообщества. 22,8% респондентов опроса проголосовали за то, что им не хватает достаточного количества ресурсов онлайн-обучения и поддерживающего сообщества.

В статье “Recent Advances in Smart Contracts: A Technical Overview and State of the Art” [33] приводится схема разработки и использования смарт-контракта, представленная на рисунке 7.

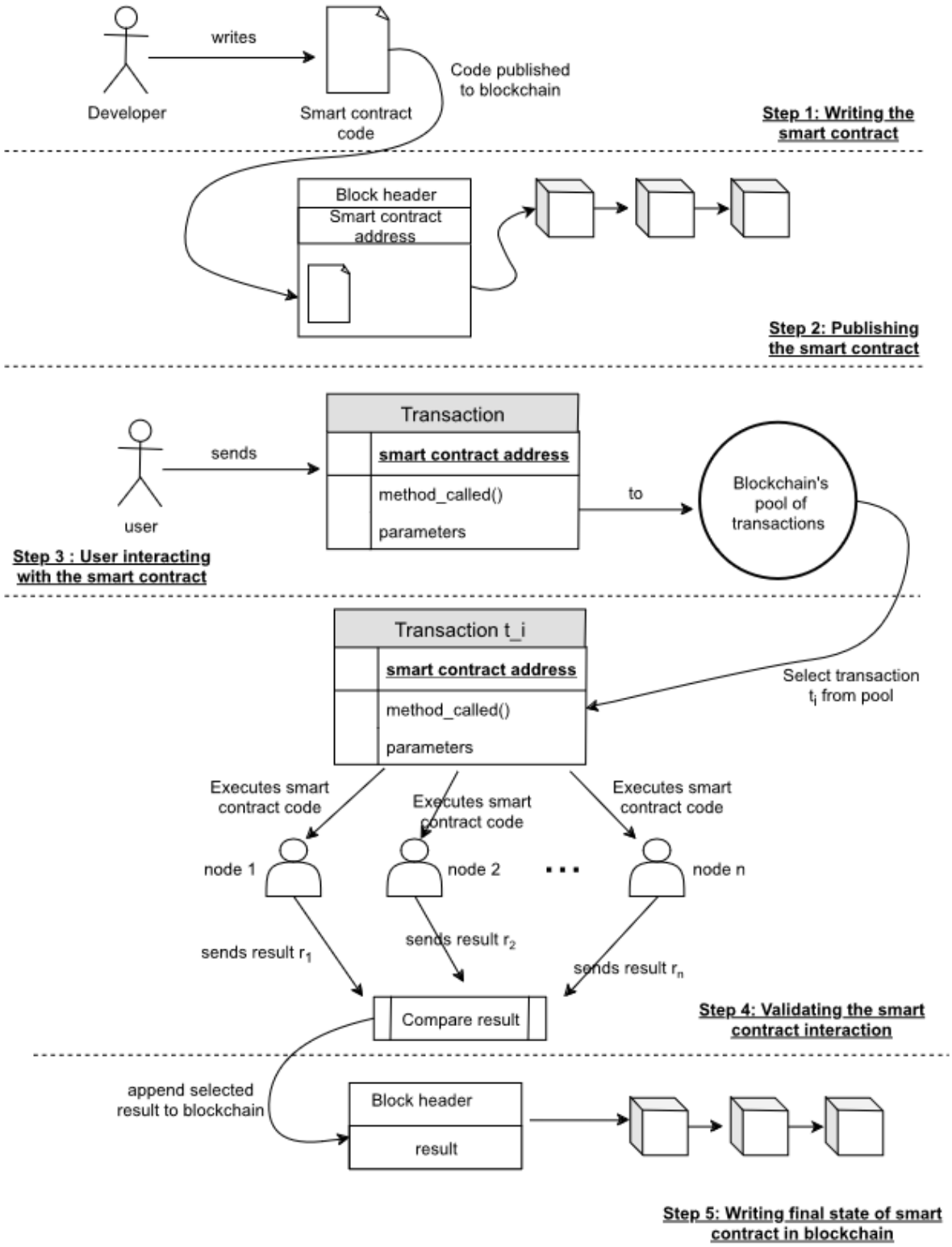

Рисунок 7 – Схема разработки и использования смарт-контракта

На схеме представлен пользователь, который взаимодействует со смарт-контрактом, однако стоит учесть, что пользователям привычнее и удобнее

работать с сервисами через веб-интерфейс. Децентрализованные приложения, состоящие из смарт-контракта и веб-интерфейса решают эту проблему, что будет рассмотрено в следующем параграфе.

1.5 Децентрализованные приложения

Децентрализованное приложение (dapp) — это приложение, построенное в децентрализованной сети, которое сочетает в себе смарт-контракт и внешний пользовательский интерфейс. В Ethereum смарт–контракты доступны и прозрачны – как открытые API, - поэтому dapp может даже включать смарт-контракт, написанный кем-то другим.

Приложение dapp может иметь внешний код и пользовательские интерфейсы, написанные на любом языке (точно так же, как приложение), для выполнения вызовов к своему внутреннему интерфейсу. Кроме того, его интерфейс может быть размещен в децентрализованном хранилище, таком как IPFS [6].

IPFS (Межпланетная файловая система) — это децентрализованная файловая система. IPFS использует
структуры данных DHT (распределенная хэш-таблица) и Merkle DAG (направленный ациклический граф). Он использует
протокол, аналогичный BitTorrent, чтобы решить, как перемещать данные по сети. Одной из
расширенных функций IPFS является то, что она поддерживает управление версиями файлов. Для обеспечения управления версиями файлов он
использует структуры данных, аналогичные Git [5].

В статье “Decentralized Applications: The Blockchain-Empowered Software System” [34] рассматриваются следующие характеристики децентрализованных приложений:

1. Производительность. Длительная задержка транзакций была критической проблемой с момента появления Bitcoin. Поскольку среднее время, затрачиваемое биткойн-узлами на майнинг блока, составляет 10 минут, среднее время завершения транзакции составляет около часа (поскольку пользователь обычно ожидает 6 блоков). Несмотря на то, что задержка ответа в Ethereum была значительно снижена примерно до 15 секунд, еще предстоит достичь достаточно малой задержки для поддержки взаимодействия обычных приложений. На самом деле, более длительные задержки расстраивают пользователей и делают приложения менее конкурентоспособными по сравнению с существующими альтернативами, не основанными на блокчейне. Например, обычный пользователь в социальной сети, основанной на блокчейне, веб-сайт обычно требует, чтобы система отреагировала на его / ее действие "Нравится" или "поделиться" публикацией в течение 2–3 секунд [34].
2. Оффлайн-платежи. Приложение DApp можно спроектировать таким образом, чтобы оно могло принимать автономные транзакции, которые подписываются для оплаты товаров. Таким образом, вместо интернет-соединения, можно использовать Bluetooth.
3. Стоимость. В нынешней блокчейн-экосистеме разработчики приложений борются с высокими комиссиями за транзакции, которые им приходится платить при развертывании и выполнении своих смарт-контрактов.
4. Идентификация. Многие системы блокчейн-приложений борются с проблемами, связанными с идентификацией. Некоторые системы, такие как ZCash и Monero, пытаются скрыть личность пользователей и транзакции. С другой стороны, многие блокчейны делают ставку на однозначную идентификацию пользователя.

1.4.1 Характеристики децентрализованных приложений:

1. Децентрализованные - dapps работают на Ethereum, открытой публичной децентрализованной платформе, где ни один человек или группа не имеет контроля.
2. Детерминированные - dapps выполняют одну и ту же функцию независимо от среды, в которой они выполняются.
3. Полные по Тьюрингу - dapps может выполнять любые действия при наличии необходимых ресурсов.
4. Изолированные - dapps выполняются в виртуальной среде, известной как виртуальная машина Ethereum, так что, если в смарт-контракте есть ошибка, это не помешает нормальному функционированию блокчейн-сети.

1.4.2 Преимущества децентрализованных приложений.

1. Нулевое время простоя – как только смарт-контракт будет развернут на блокчейне, сеть в целом всегда сможет обслуживать клиентов, желающих взаимодействовать с контрактом. Таким образом, злоумышленники не могут запускать атаки типа "отказ в обслуживании", нацеленные на отдельные децентрализованные приложения.
2. Конфиденциальность – вам не нужно предоставлять реальные идентификационные данные для развертывания dapp или взаимодействия с ним.
3. Устойчивость к цензуре – ни одна организация в сети не может запретить пользователям отправлять транзакции, развертывать dapps или считывать данные из блокчейна.
4. Полная целостность данных – данные, хранящиеся в блокчейне, являются неизменяемыми и неоспоримыми благодаря криптографическим примитивам. Злоумышленники не могут подделывать транзакции или другие данные, которые уже были обнародованы.

5. Вычисления без доверия / проверяемое поведение – смарт-контракты могут быть проанализированы и гарантированно будут выполняться предсказуемым образом без необходимости доверять центральному органу власти. Это неверно в традиционных моделях; например, когда пользователи используют системы онлайн-банкинга, они должны быть уверены, что финансовые учреждения не будут злоупотреблять их финансовыми данными, подделывать записи или подвергаться взлому.

1.4.3 Недостатки децентрализованных приложений

1. Обслуживание – Dapps может быть сложнее поддерживать, поскольку код и данные, публикуемые в блокчейне, сложнее модифицировать. Разработчикам трудно вносить обновления в свои dapps (или базовые данные, хранящиеся в dapp) после их развертывания, даже если в старой версии обнаружены ошибки или угрозы безопасности.
2. Накладные расходы на производительность – это огромные накладные расходы на производительность, и масштабирование действительно сложное. Чтобы достичь уровня безопасности, целостности, прозрачности и надежности, к которому стремится Ethereum, каждый узел запускает и хранит каждую транзакцию. Вдобавок ко всему, достижение консенсуса по доказательству доли участия также требует времени.
3. Перегрузка сети – когда одно приложение dapp использует слишком много вычислительных ресурсов, выполняется резервное копирование всей сети. В настоящее время сеть может обрабатывать только около 10–15 транзакций в секунду; если транзакции отправляются быстрее, чем это, пул неподтвержденных транзакций может быстро увеличиться.
4. Пользовательский интерфейс – возможно, будет сложнее создать удобный интерфейс, потому что среднестатистическому конечному пользователю может оказаться слишком сложно настроить набор инструментов,

необходимый для взаимодействия с блокчейном по-настоящему безопасным способом.

5. Централизация – удобные для пользователя и разработчика решения, построенные поверх базового уровня Ethereum, в любом случае могут в итоге выглядеть как централизованные сервисы. Например, такие сервисы могут хранить ключи или другую конфиденциальную информацию на стороне сервера, обслуживать интерфейс с использованием централизованного сервера или запускать важную бизнес-логику на централизованном сервере перед записью в блокчейн. Централизация устраняет многие (если не все) преимущества блокчейна по сравнению с традиционной моделью [6].

Результаты и выводы первой главы

В результате анализа предметной области было определено, что децентрализованные приложения на основе блокчейн-технологий имеют большой потенциал для решения многих проблем, связанных с централизованными системами. Такие приложения позволяют достичь более высокого уровня безопасности, прозрачности и открытости, а также могут снизить затраты и повысить эффективность бизнес-процессов.

Однако, разработка децентрализованных приложений также связана с рядом технических и организационных проблем, которые нужно учитывать и решать в процессе разработки. Например, необходимо обеспечить достаточную масштабируемость, производительность и безопасность системы, а также учитывать особенности работы с криптовалютой и различными блокчейн-платформами.

В целом, разработка децентрализованного приложения может быть сложным и трудоемким процессом, однако она может принести значительные выгоды и стать конкурентным преимуществом в условиях современного рынка.

2 МЕТОДОЛОГИЯ ПРИМЕНЕНИЯ БЛОКЧЕЙН-ТЕХНОЛОГИЙ В РАЗРАБОТКЕ ДЕЦЕНТРАЛИЗОВАННЫХ СЕРВИСОВ

2.1 Создание смарт-контракта в среде разработки Remix IDE

Solidity — это объектно-ориентированный язык высокого уровня для реализации смарт-контрактов [41]. Solidity является наиболее популярным языком для разработки смарт-контрактов для Ethereum. Также, командой Ethereum разрабатывается браузерный компилятор и среда разработки с открытыми исходным кодом Remix. Remix Project — это богатый набор инструментов, который может быть использован на протяжении всего процесса разработки контрактов пользователями с любым уровнем знаний, а также в качестве учебной лаборатории для обучения и экспериментов с Ethereum [42].

Remix IDE доступна по адресу https://remix.ethereum.org/ и при подключении создает рабочее пространство с некоторыми файлами. Структура воркспейса представлена на рисунке 8.

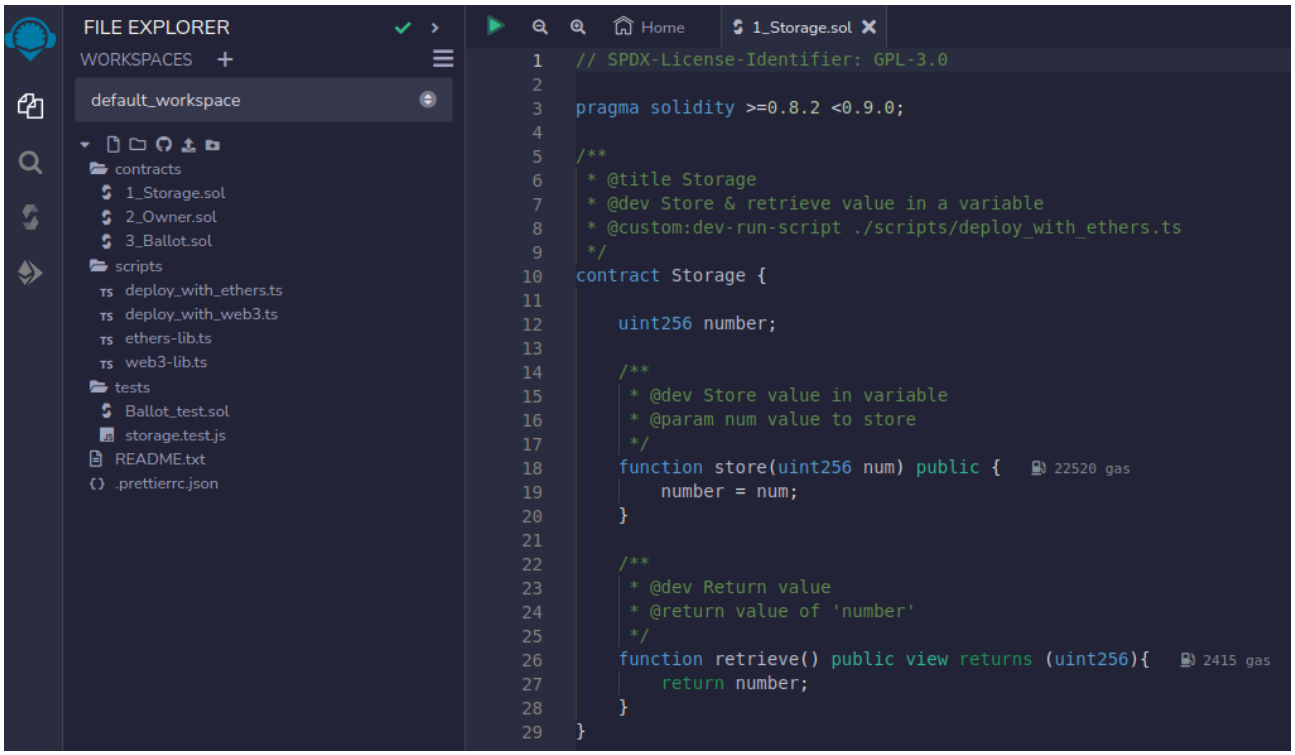

Рисунок 8 – Рабочее пространство Remix IDE

В папке contracts/ находятся 3 примера смарт-контракта: 1_Storage.sol для сохранения и получения числа, 2_Owner.sol для определения и передачи прав на смарт-контракт, а также 3_Ballot.sol, в котором запрограммирован процесс голосования.

Папка scripts/ содержит скрипты для разворачивания смарт-контракта в сети, а папка tests/ включает примеры тестирования контрактов.

Рассмотрим структуру смарт-контракта на примере файла 1_Storage.sol, код которого представлен на рисунке 8.

Доверие к смарт-контрактам может быть лучше установлено, если доступен их исходный код. Поскольку предоставление доступа к исходному коду всегда связано с юридическими проблемами, связанными с авторским правом, компилятор Solidity поощряет использование машиночитаемых идентификаторов лицензий SPDX. Каждый исходный файл должен начинаться с комментария, указывающего на его лицензию: // SPDX-License-Identifier: MIT [56].

Ключевое слово pragma используется для включения определенных функций компилятора или проверок. В данном примере определяется требование к версии компилятора Solidity >= 0.8.2 и < 0.9.0.

Далее идут комментарии, которые расположены между символами /* и */. Эти комментарии делаются для разработчиков с целью улучшения понимания кода.

Код смарт-контракта начинается с ключевого слова contract, после которого следует его название. Функции контракта определяются в фигурных скобках.

Solidity — это язык со статической типизацией, что означает, что необходимо указать тип каждой переменной (state и local). Solidity предоставляет

несколько элементарных типов, которые могут быть объединены для формирования сложных типов [57]. В языке программирования Solidity существуют типы данных, встречающиеся и в других языках, например, boolean и int/uint, однако, есть и уникальные типы, требующиеся при разработке смарт-контрактов, например, address. В данном примере объявляется переменная number типа uint256, который может хранить целые числа от 0 до 2^256-1.

Далее объявляется функция store, которая принимает на вход параметр num типа uint256 и присваивает переменной смарт-контракта number это число.

Ниже объявлена функция retrieve, которая возвращает число, сохраненное в смарт-контракте.

Чтобы скомпилировать смарт-контракт, необходимо в левой панели перейти во вкладку “Solidity Compiler”, после чего выбрать компилятор необходимой версии и нажать кнопку “Compile 1_Storage.sol”. Интерфейс вкладки представлен на рисунке 9.

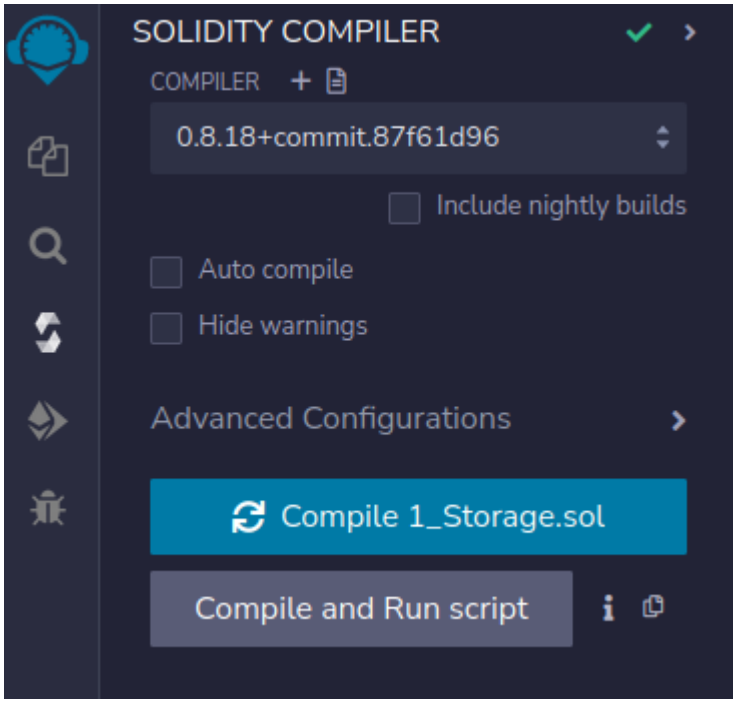

Рисунок 9 – Компиляция смарт-контракта

После компиляции контракта появляется возможность его развертывания в сети блокчейн. Для этого необходимо перейти на вкладку ниже, которая называется “Deploy & Run Transactions”. Веб-форма для развертывания контракта представлена на рисунке 10.

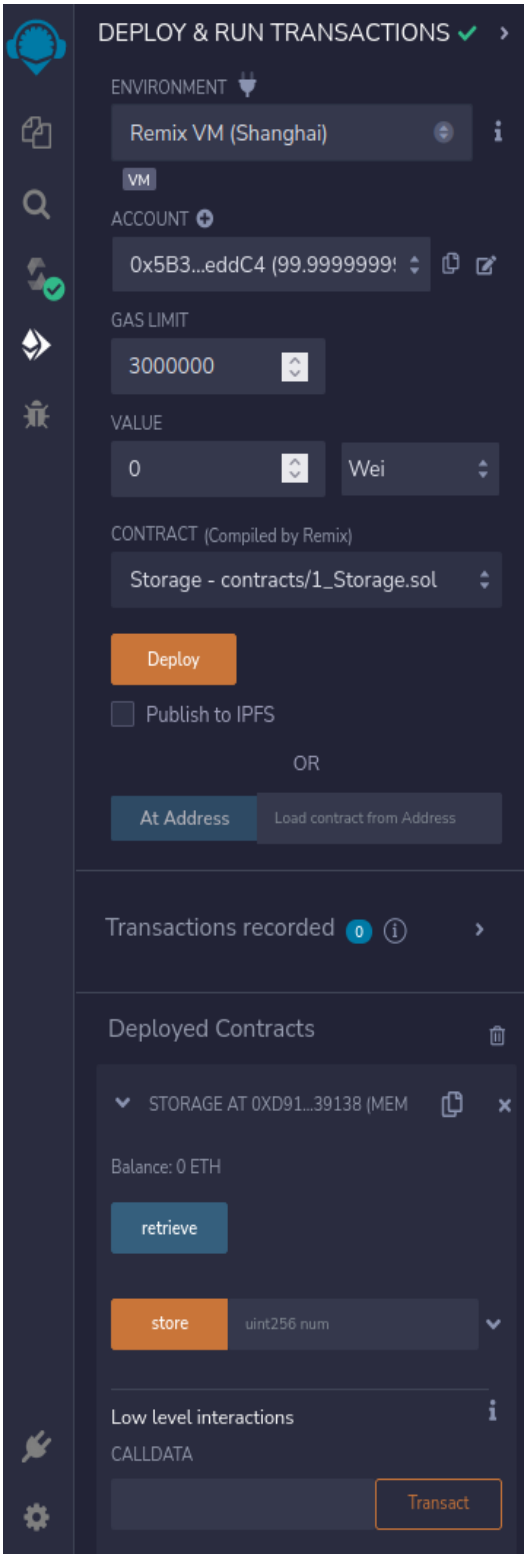

Рисунок 10 – Веб-форма для развертывания смарт-контракта

Рассмотрим подробнее поля для заполнения.

Поле Environment позволяет выбрать провайдера для взаимодействия с блокчейном. Для тестирования можно выбрать встроенную в IDE имитацию блокчейна, которая работает локально. Для этого в форме выбора Environment нужно выбрать “Remix VM (Shanghai)”.

Account - адрес кошелька, с которого будет происходить развертывание смарт-контракта. Так как в примере используется тестовое окружение, можно выбрать любой аккаунт, на котором уже находится эфир.

Газ — это плата, необходимая для успешного проведения транзакции или исполнения контракта на блокчейн-платформе Ethereum. Gas Limit — это максимальный объем работы, который, по вашим оценкам, валидатор выполнит по конкретной транзакции [48]. В тестовом окружении нет смысла менять предел используемого газа, так как транзакции в локальном блокчейне обрабатываются почти мгновенно.

Далее, можно выбрать количество эфира, которое будет приложено к транзакции при развертывании контракта, однако это не требуется в данном примере.

Ниже есть поле для выбора контракта для регистрации в сети, где нужно выбрать файл “1_Storage.sol”, после чего нажать “Deploy”. Результат развертывания контракта виден в консоли и представлен на рисунке 11.

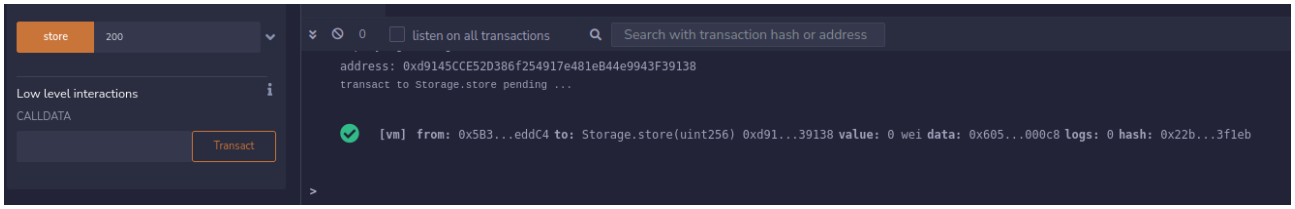

Рисунок 11 – Результат развертывания контракта

После успешной регистрации контракта в сети слева в панели появляется возможность взаимодействия со смарт-контрактом путем вызова функций. На рисунке 12 слева отображена форма с двумя функциями - retrieve и store для получения и сохранения числа соответственно. В консоль выводится информация о вызове функций, где можно заметить, что использование функции получения числа является бесплатным.

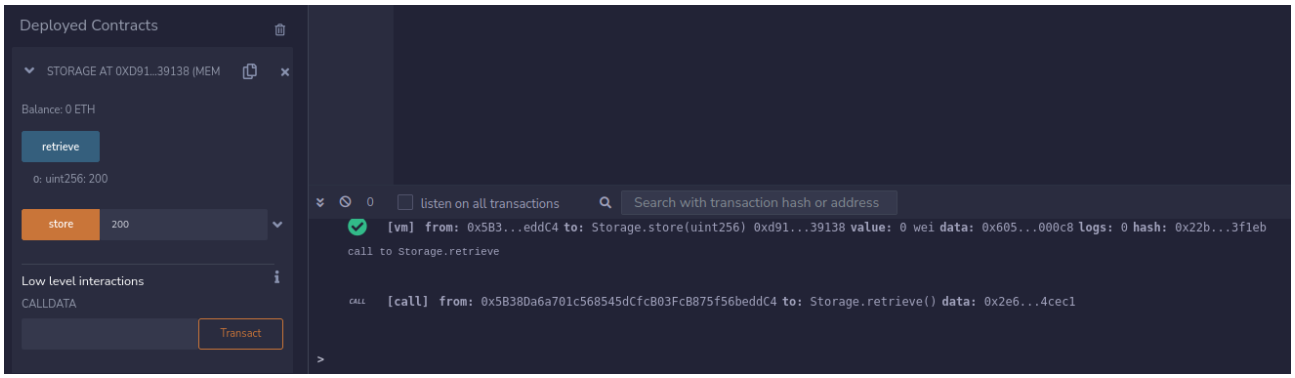

Рисунок 12 – Использование функций смарт-контракта

Таким образом, был рассмотрен базовый пример смарт-контракта, а также структура контракта в языке программирования Solidity. Данный смарт-контракт был развернут в тестовом локальном блокчейне, а также протестированы его функции.

2.2 Создание криптовалютного кошелька и получение монет в тестовой сети блокчейна

MetaMask - популярный криптовалютный кошелек, который поддерживает широкий спектр токенов на основе Ethereum и не взаимозаменяемых токенов (NFT) в поддерживаемых блокчейнах [45]. Так как этот кошелек позволяет хранить и пересылать эфир как в основной сети, так и тестовых, он подойдет в рамках данной работы для взаимодействия со смарт-контрактами.

MetaMask поддерживается на различных ОС, а также доступен в виде браузерного расширения или мобильного приложения. Установщик доступен на сайте разработчика: https://metamask.io/download/.

После установки расширения необходимо создать или импортировать кошелек. Интерфейс расширения MetaMask представлен на рисунке 13. При создании кошелька необходимо придумать мастер-пароль и записать seed-фразу, чтобы иметь возможность восстановить аккаунт. Seed-фраза (мнемоническая) — это набор из 12 или более слов, который используется для восстановления доступа к кошельку [60].

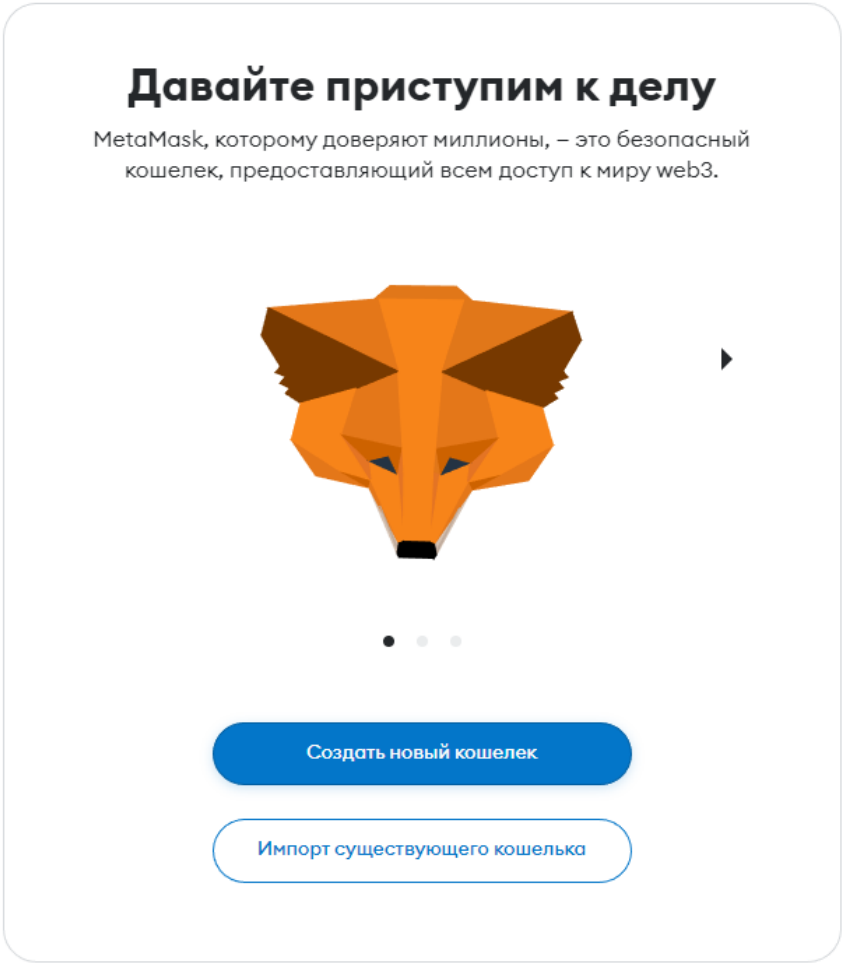

Рисунок 13 – Интерфейс расширения MetaMask

После создания кошелька в меню MetaMask необходимо выбрать тестовую сеть, чтобы оперировать эфиром, который не стоит денег - SepoliaETH. Для этого

в расширении необходимо добавить тестовые сети с помощью кнопки “Добавить сеть” на рисунке 14.

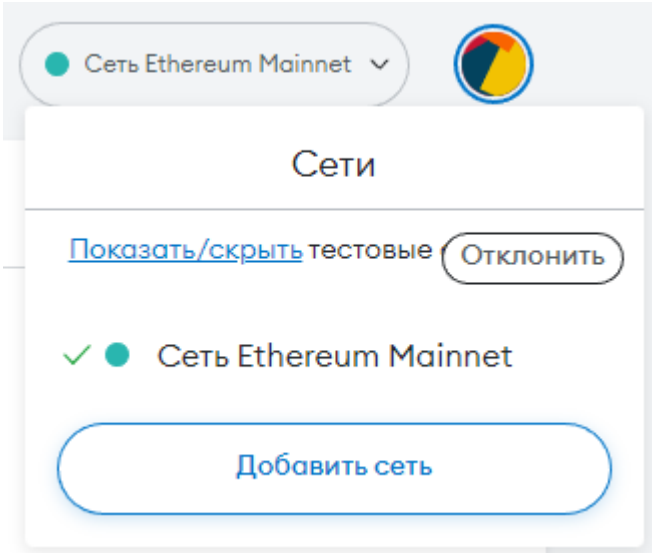

Рисунок 14 – Добавление сети в MetaMask

Далее, необходимо добавить сеть вручную с помощью кнопки “Добавить вручную” на рисунке 15.

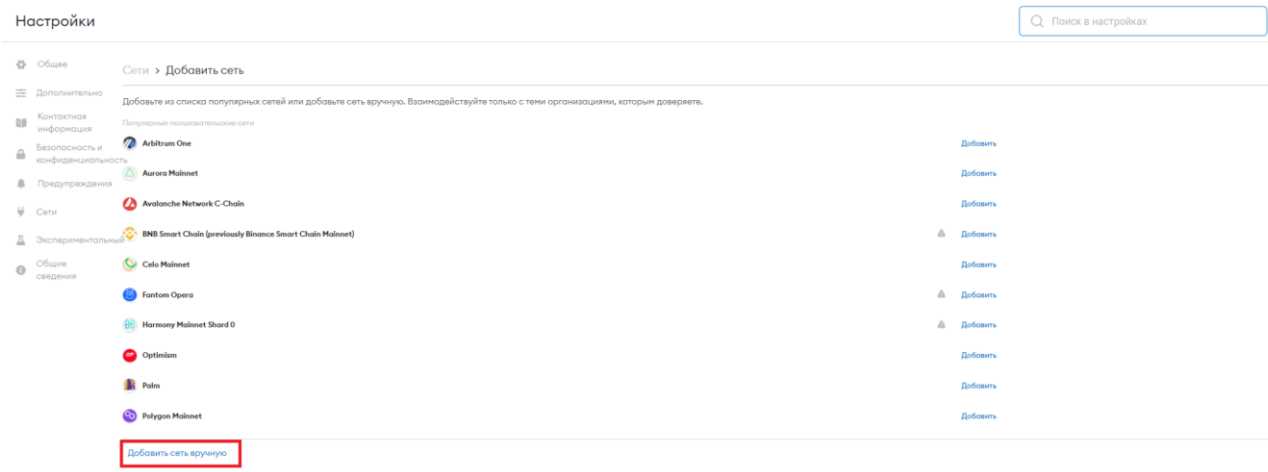

Рисунок 15 – Добавление сети вручную

Далее, необходимо перейти в свой аккаунт Alchemy [58], чтобы получить ссылки на тестовые сети Ethereum.

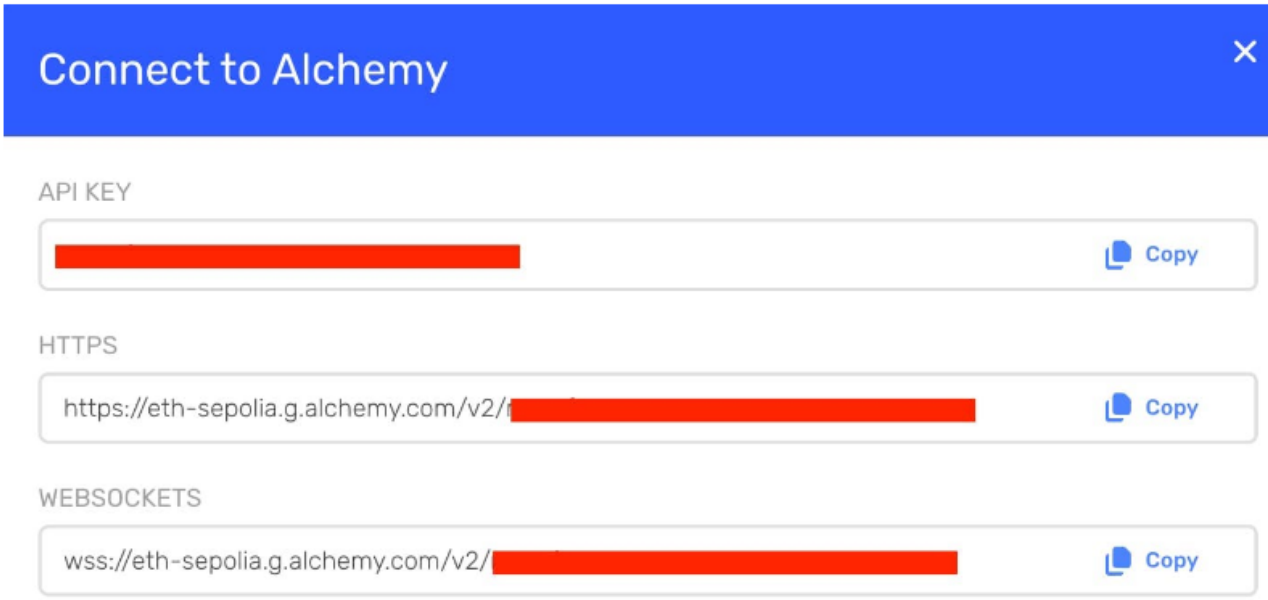

Рисунок 16 – Ссылки для подключения к тестовым сетям Ethereum

Данные необходимо ввести на странице расширения MetaMask, чтобы кошелек подключился к тестовым сетям блокчейна. После этого появится возможность выбора сети Sepolia, представленной на рисунке 17.

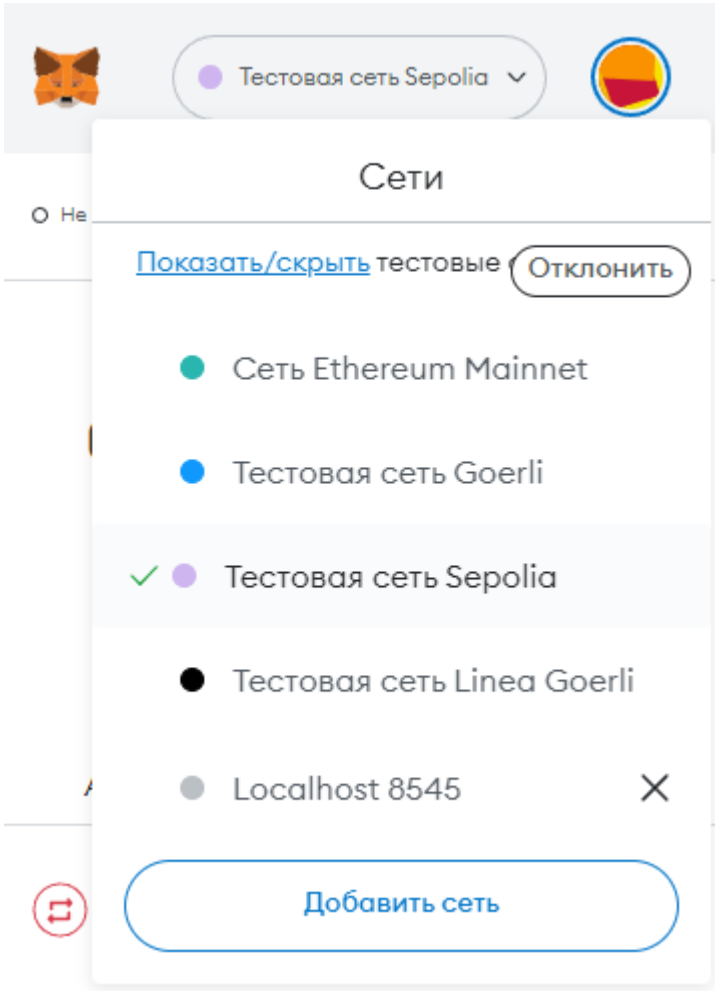

Рисунок 17 – Выбор тестовой сети в MetaMask

Для работы со смарт-контрактами требуется эфир, однако в основной сети его стоимость высока для учебных целей. Помимо основной сети, разработчики поддерживают тестовые сети, в которых эфир не имеет никакой цены. Разработчики могут использовать различные краны, чтобы получить небольшое количество эфира и использовать его в своих экспериментах.

Ethereum faucet – это интернет-ресурс, выдающий пользователям установленную правилами сайта, премию в криптовалюте. Выплата производится через определенные интервалы, от минут до часов [46]. В данной работе рассматривается создание смарт-контрактов в тестовой сети Ethereum Sepolia. Alchemy [58] представляет бесплатный кран SepoliaETH, который позволяет получить 0.5 эфира в день и использовать его в своих целях, например, в разработке.

Для получения Sepolia ETH необходимо перейти на Sepolia Faucet [59] и ввести адрес кошелька в тестовой сети, который можно узнать в MetaMask. Интерфейс крана представлен на рисунке 18.

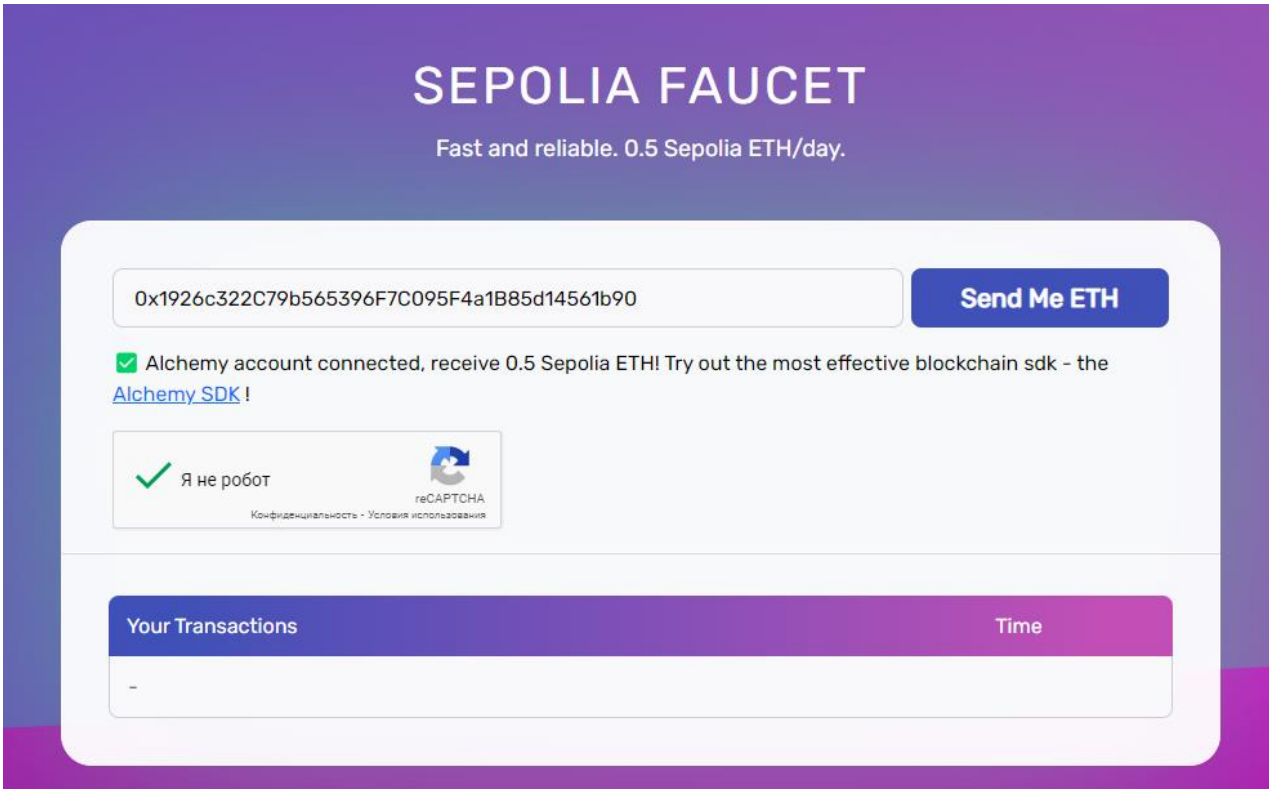

Рисунок 18 – Sepolia Faucet

Транзакцию получения эфира можно отследить на ресурсе Etherscan, скриншот представлен на рисунке 19.

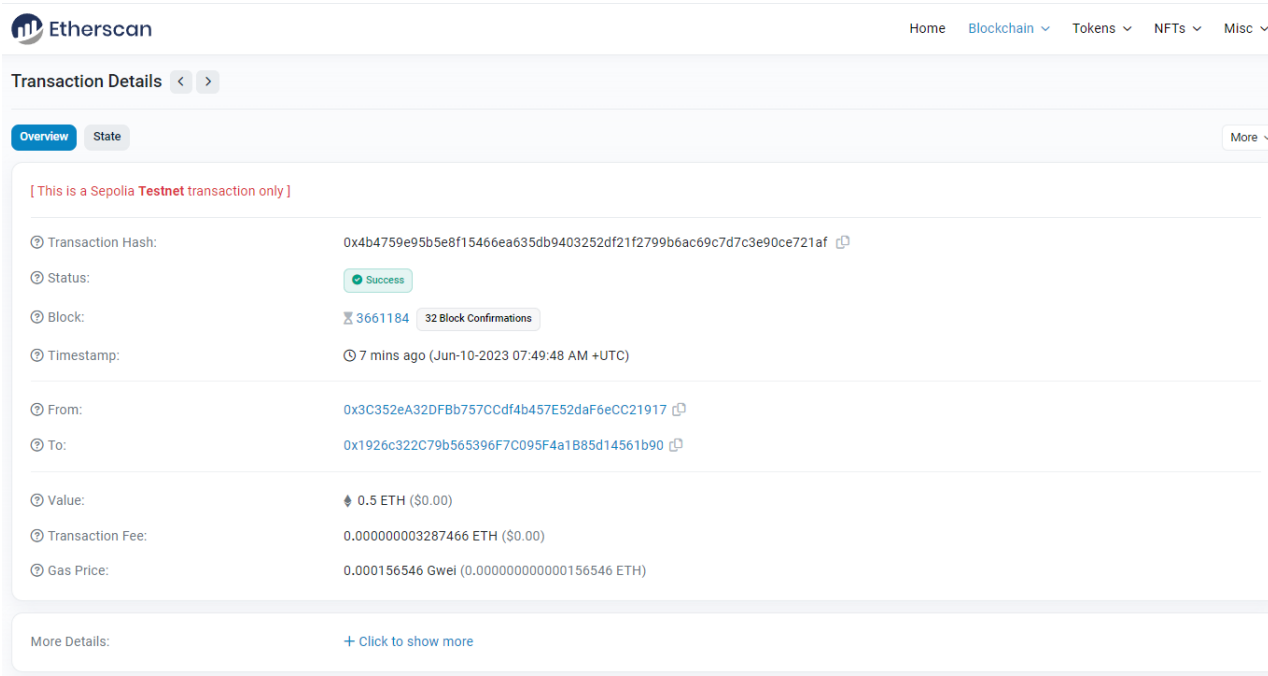

Рисунок 19 – Транзакция получения SepoliaETH

Полученный бесплатный эфир в тестовой сети Sepolia можно использовать для разработки и тестирования смарт-контрактов и децентрализованных приложений.

2.3 Публикация и верификация смарт-контракта в блокчейн-сети

Теперь, когда выполнены шаги по созданию криптовалютного кошелька, получению тестового эфира и рассмотрен пример создания смарт-контракта, появилась возможность опубликовать контракт не только для личного пользования, но и в открытый доступ для других пользователей сети.

Вернемся к параграфу 2.1, где описывается развертывание смарт-контракта в сети. В поле Environment появилась возможность выбора другого провайдера сети: Injected Provider - MetaMask, что представлено на рисунке 20.

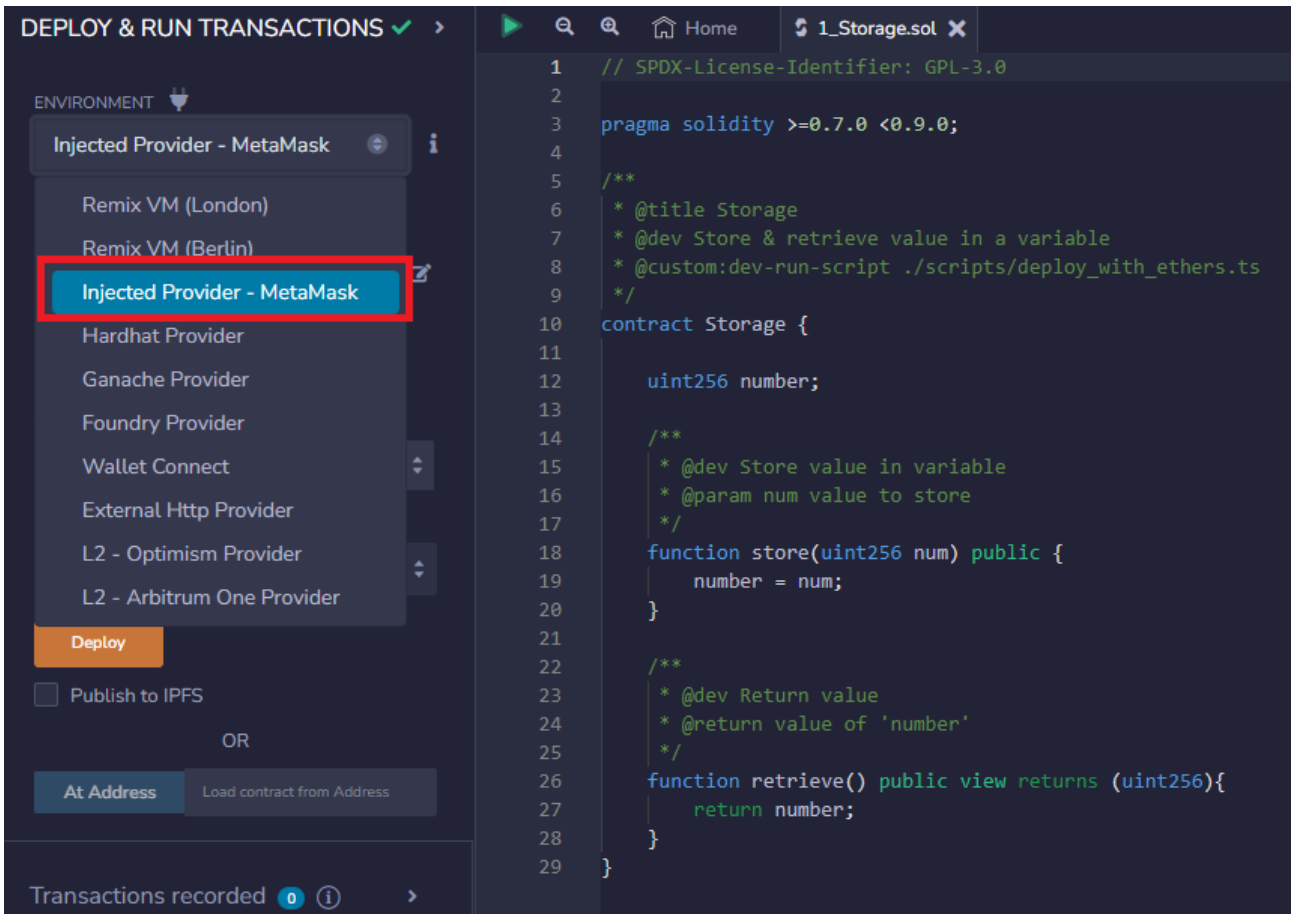

Рисунок 20 – Новый провайдер - MetaMask

При выборе данного провайдера в списке аккаунтов для развертывания смарт-контракта появляется созданный в параграфе 2.2 адрес кошелька MetaMask, представленный на рисунке 21.

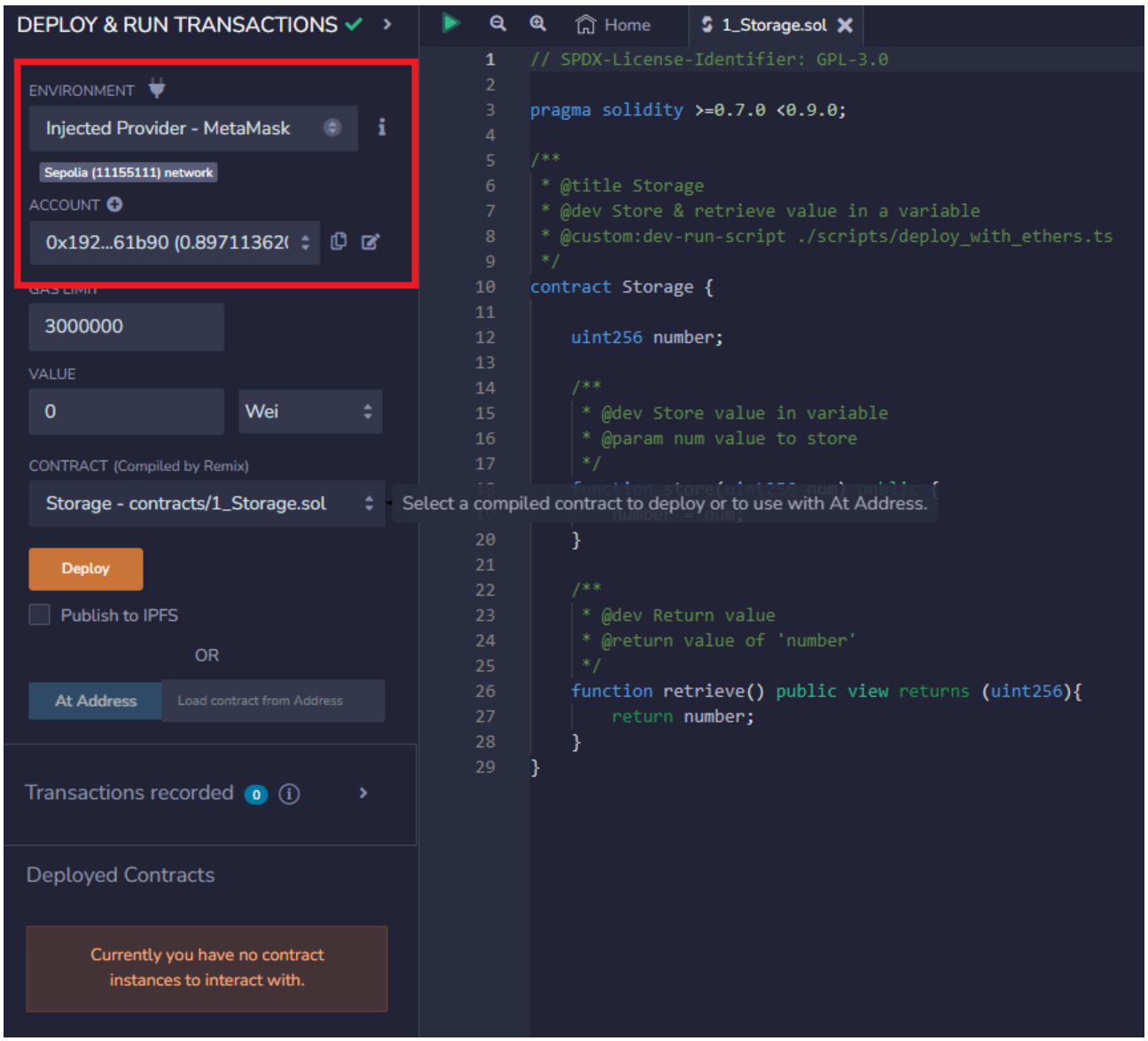

Рисунок 21 –Выбор аккаунта MetaMask для развертывания

При нажатии кнопки “Deploy” появится окно расширения MetaMask для подтверждения транзакции развертывания смарт-контракта, представленное на рисунке 22.

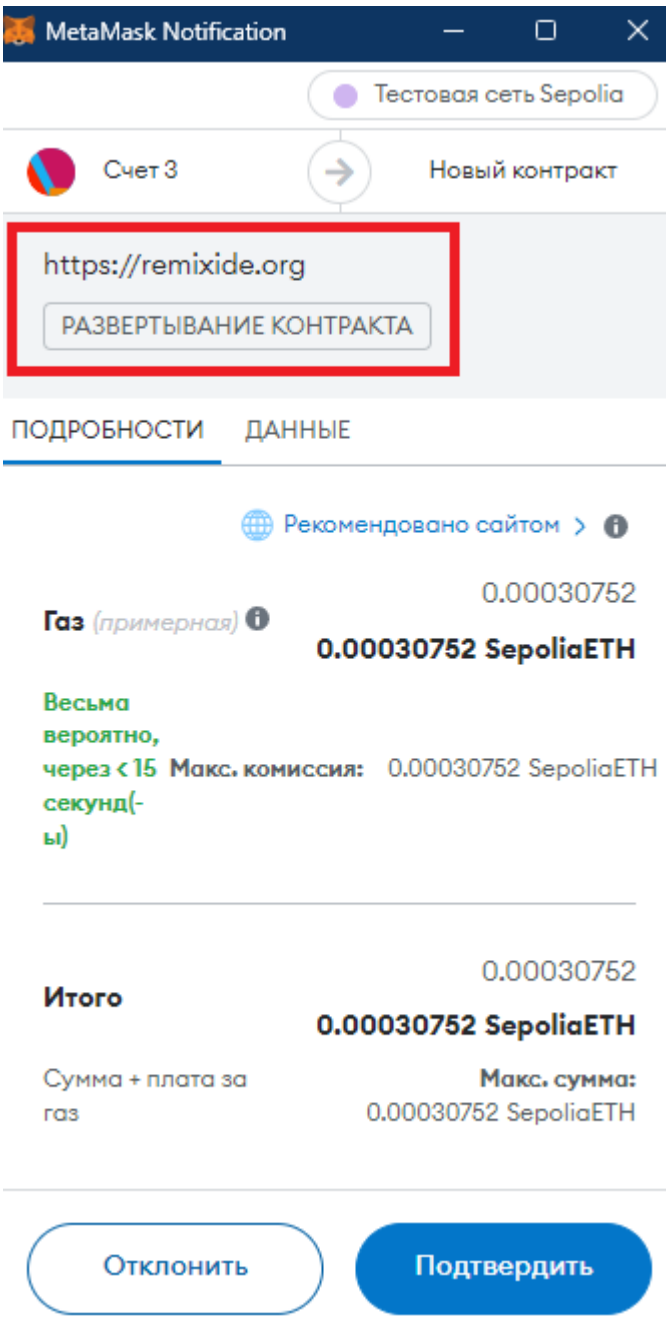

Рисунок 22 – Окно подтверждения транзакции развертывания

После успешного развертывания смарт-контракта появляется возможность его использования в интерфейсе Remix IDE. Отличие данного контракта от того, который был развернут в локальном окружении состоит в том, что этим контрактом могут пользоваться люди по всему миру, подключенные к тестовой сети Sepolia Ethereum. Подтверждение успешного развертывания контракта представлено на рисунке 23.

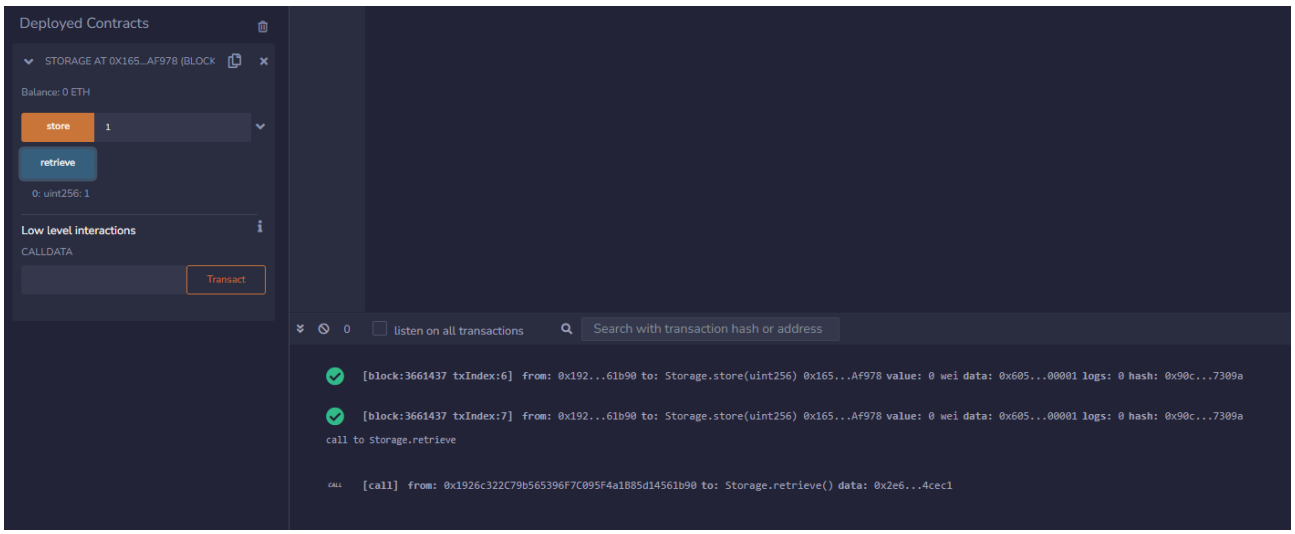

Рисунок 23 – Успешное развертывание смарт-контракта

Однако, для того чтобы другие разработчики могли использовать этот смарт-контракт в своих проектах, желательно подтвердить его прозрачность и безопасность, верифицировав код контракта.

Для этого необходимо перейти на блокчейн-эксплорер Etherscan в сети Sepolia по адресу смарт-контракта и перейти во вкладку Contract, а затем перейти по ссылке “Verify and Publish”. Этот шаг представлен на рисунке 24.

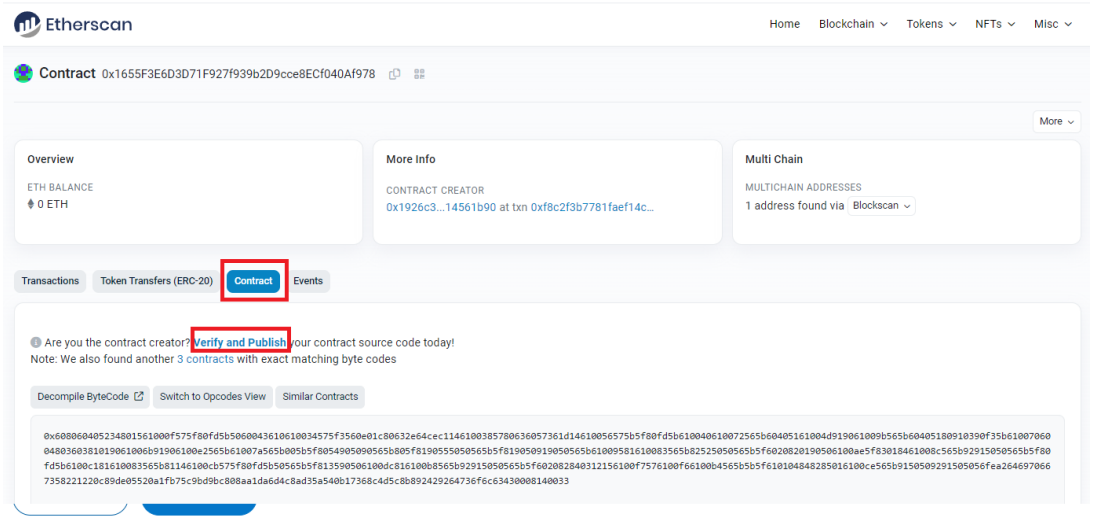

Рисунок 24 – Смарт-контракт в Etherscan

Для верификации смарт-контракта необходимо выбрать версию компилятора и тип лицензии MIT, что представлено на рисунке 25.

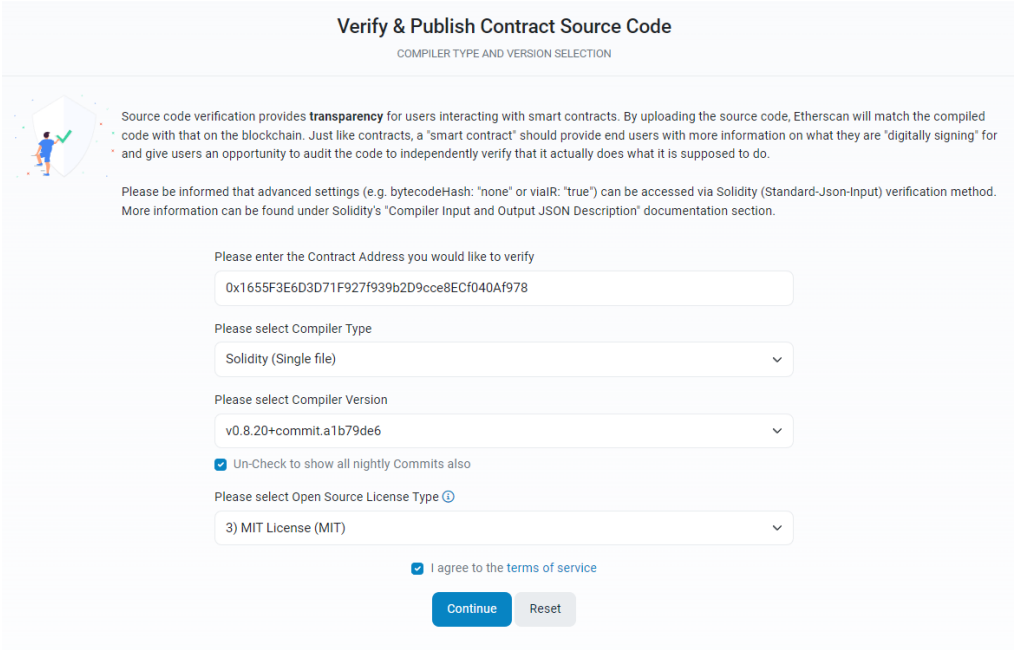

Рисунок 25 – Верификация смарт-контракта

После этого необходимо вставить код контракта, чтобы верифицировать его, а затем нажать кнопку “Verify & Publish”. Этот шаг представлен на рисунке 26.

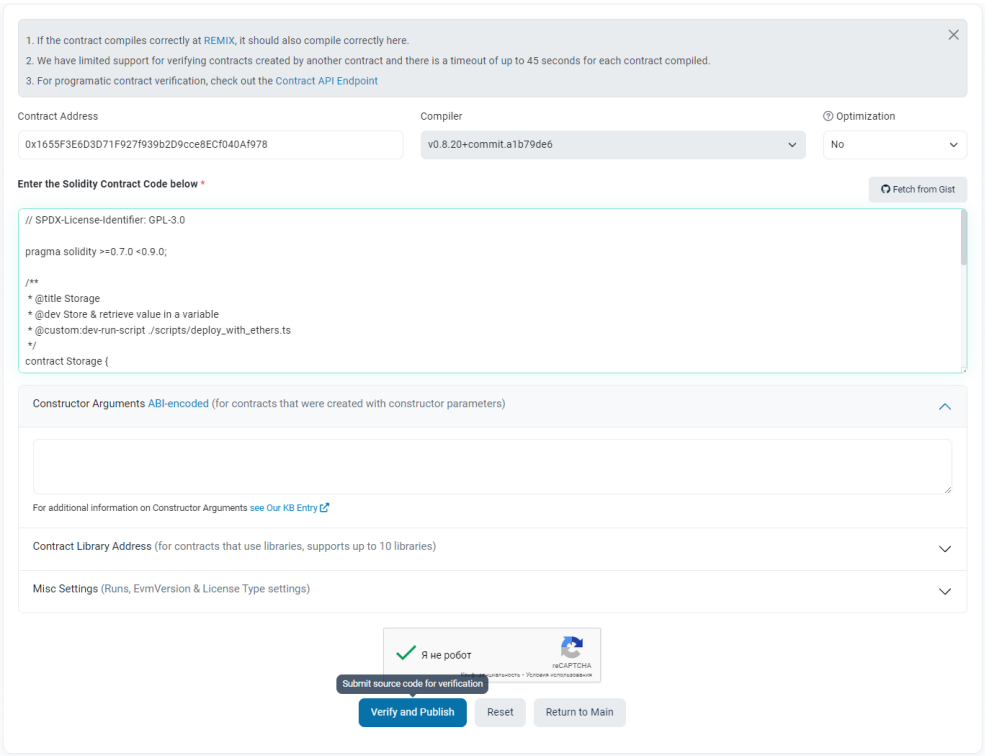

Рисунок 26 – Верификация кода смарт-контракта

Теперь смарт-контракт опубликован и верифицирован, что позволит другим разработчикам использовать его в своих проектах, удостоверившись в его безопасности. Результат успешной верификации отображен на рисунке 27.

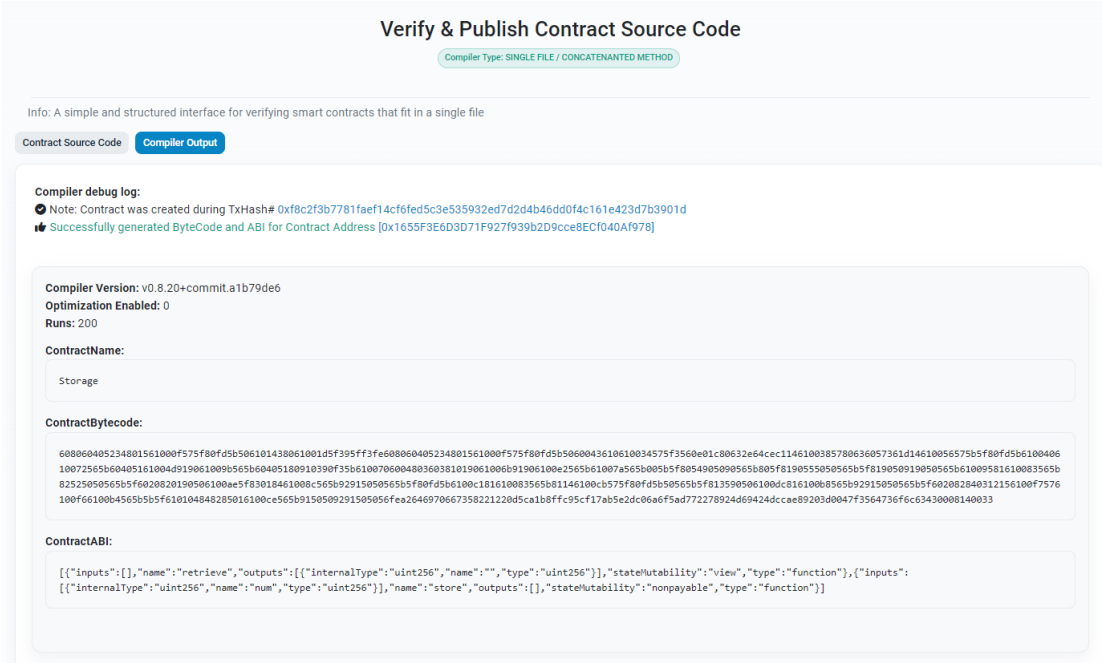

Рисунок 27 – Успешная верификация смарт-контракта

Результаты и выводы второй главы

В ходе второй главы была рассмотрена методология применения блокчейн-технологий в разработке децентрализованных сервисов. В частности, продемонстрирована методика разработки смарт-контрактов, создание и использование криптовалютного кошелька, получение бесплатного эфира для взаимодействия со смарт-контрактами, а также публикация и верификация смарт-контракта в сети.

3 РАЗРАБОТКА ДЕЦЕНТРАЛИЗОВАННОГО СЕРВИСА ДЛЯ ПРОВЕДЕНИЯ ЛОТЕРЕЙ

3.1 Проектирование и разработка смарт-контракта

Смарт-контракт для проведения лотерей должен содержать следующие функции:

Lottery() - конструктор, т.е., функция, вызываемая при инициализации контракта. Внутри этой функции менеджером игры становится пользователь, который развернул смарт-контракт в сети.

Enter() - если пользователь отправил вместе со смарт-контрактом больше 0.1 ETH, то его нужно добавить в список участников лотереи.

Random() - функция, которая генерирует случайное число.

PickWinner() - функция, которая выбирает случайного игрока и отправляет ему призовой фонд.

Restricted() - функция, которая определяет, что отправитель ETH - менеджер контракта.

GetPlayers() - функция, которая возвращает список участников лотереи.

Код этого смарт-контракта на языке программирования Solidity с функциями, описанными выше, представлен на рисунке 28.

```
pragma solidity ^0.4.17;

contract Lottery {
    address public manager;
    address[] public players;

    function Lottery() public {
        manager = msg.sender;
    }

    function enter() public payable {
        require(msg.value > .01 ether);
        players.push(msg.sender);
    }

    function random() public view returns (uint) {
        return uint(keccak256(block.difficulty, now, players.length));
    }

    function pickWinner() public restricted {
        uint index = random() % players.length;
        players[index].transfer(this.balance);
        players = new address[](0);
    }

    modifier restricted() {
        require(msg.sender == manager);
        _;
    }

    function getPlayers() public view returns (address[]) {
        return players;
    }
}
```

Рисунок 28 – Код смарт-контракта для проведения лотерей

Чтобы развернуть смарт-контракт в сети блокчейн, нужно повторить шаги из пункта 2.3 этой работы. Для этого необходимо перейти в Remix IDE, создать файл lottery.sol и скомпилировать его, как показано на рисунке 29.

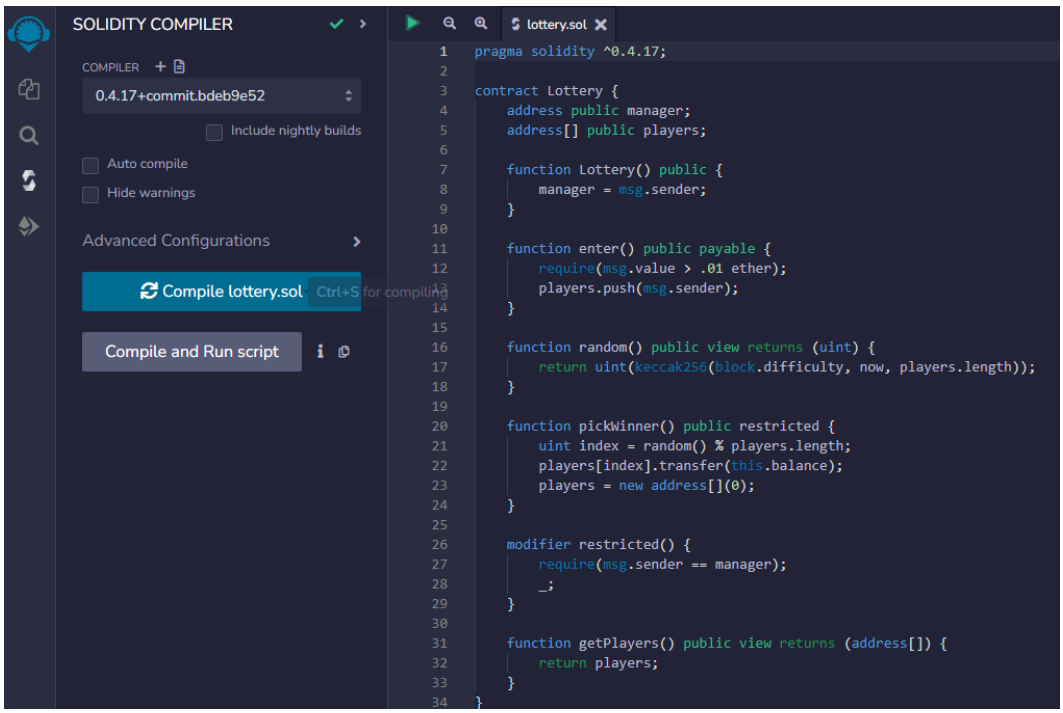

Рисунок 29 – Компиляция смарт-контракта в Remix IDE

Успешно скомпилированный контракт необходимо развернуть в одной из сетей блокчейна. Так как в рамках данной работы ставилась цель разработать учебный проект, то данный скрипт был развернут в тестовой сети Ethereum - Sepolia.

На рисунке 30 представлен интерфейс развертывания смарт-контракта в сети Sepolia. Был выбран кошелек Metamask и аккаунт, где находилось чуть более 0.39 тестового эфира. Также, был установлен лимит газа для этой транзакции.

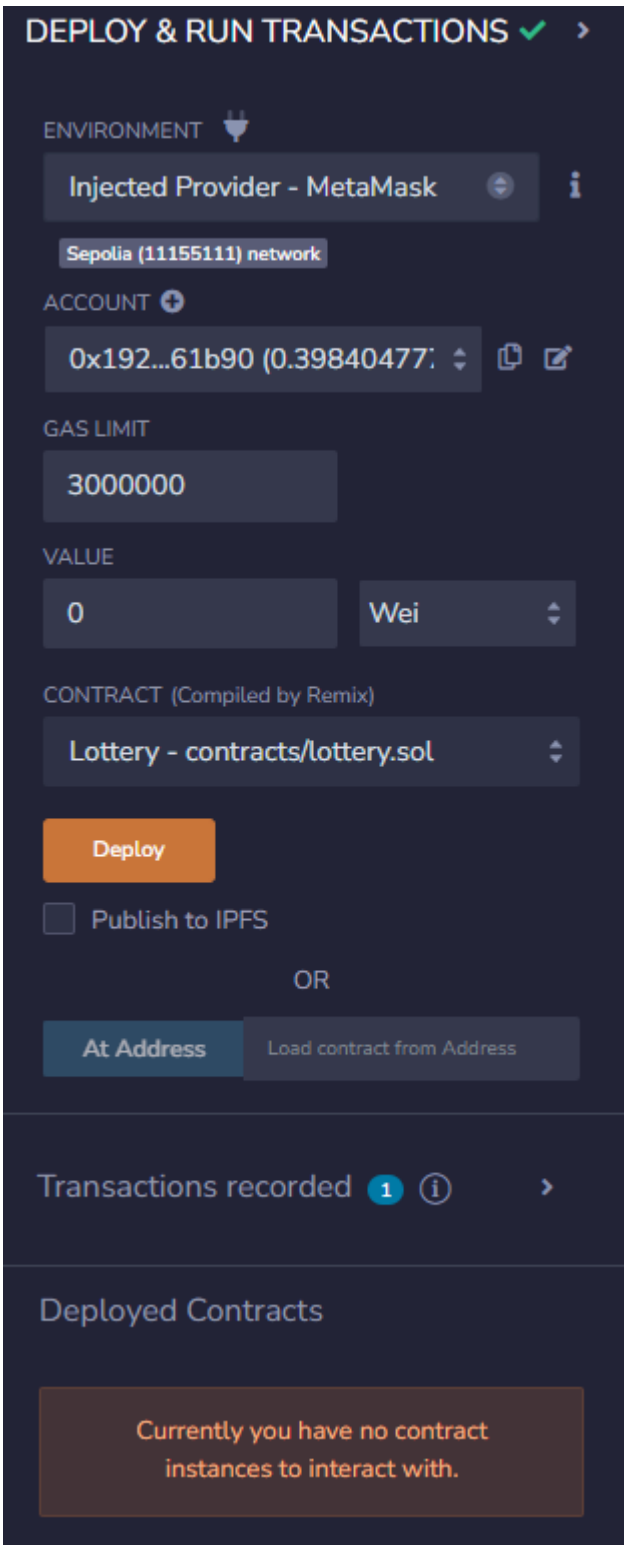

Рисунок 30 – Развертывание контракта в сети Sepolia

Далее, на рисунке 31 показано, что предлагаемое количество газа для развертывания - 0.00104307 SepoliaETH, совсем небольшая сумма относительно

того, что тестовый эфир можно получать в количестве 0.5 SepoliaETH на специальных ресурсах, как было рассмотрено в пункте 2.2.

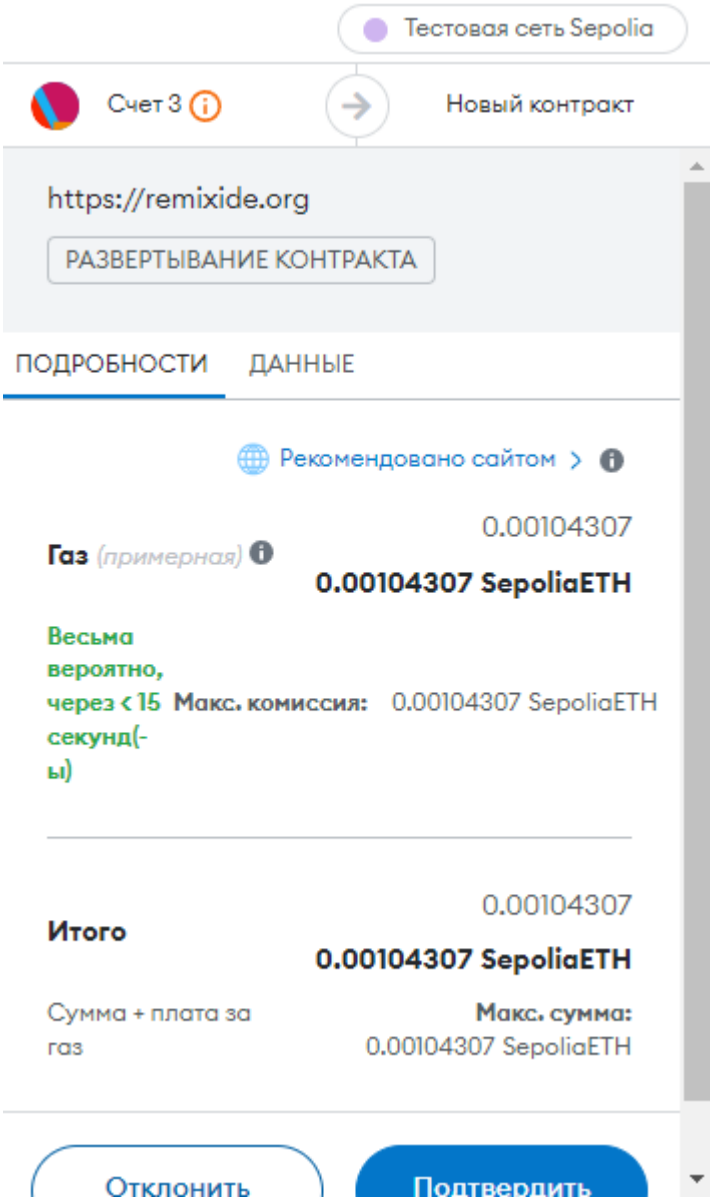

Рисунок 31 – Подтверждение транзакции для развертывания смарт-контракта

После нескольких подтверждений транзакций смарт-контракт для проведения лотерей успешно развернут в сети, а интерфейс Remix IDE позволяет взаимодействовать с ним при помощи интерфейса, представленного на рисунке 32.

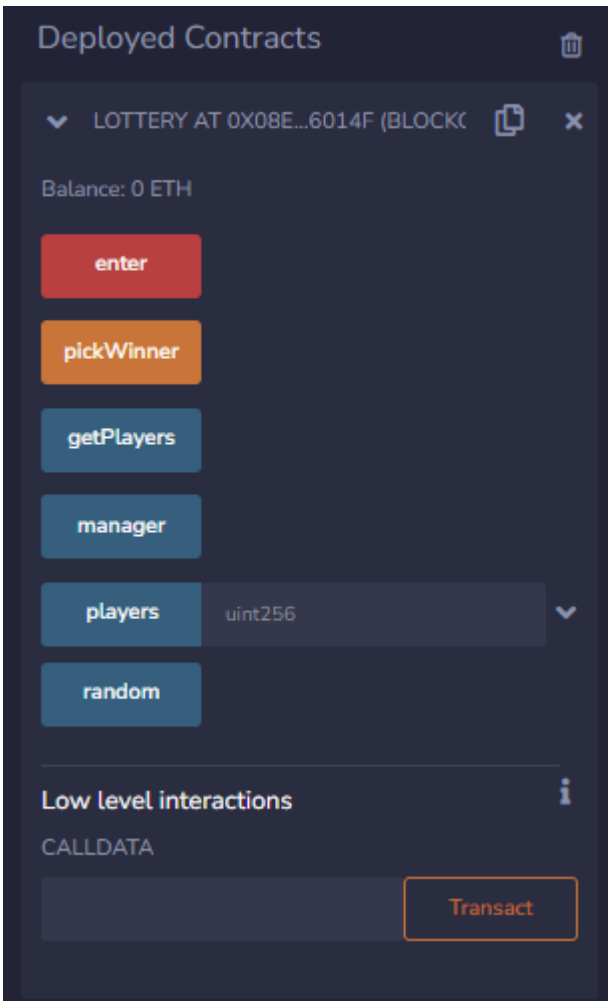

Рисунок 32 – Интерфейс для взаимодействия со смарт-контрактом

Функция enter выделена красным цветом, так как имеет свойство payable - для ее использования вместе с вызовом функции, необходимо передать > 0.1 SepoliaETH, иначе транзакция будет отклонена.

Для тестирования используем функцию enter, передав вместе с ней 0.1 эфира, чтобы участвовать в лотерее, затем вызовем функцию getPlayers, чтобы удостовериться в том, что аккаунт стал участником. На рисунке 33 показано, что баланс контракта увеличился на 0.1 SepoliaETH, а в списке участников находится аккаунт-отправитель эфира.

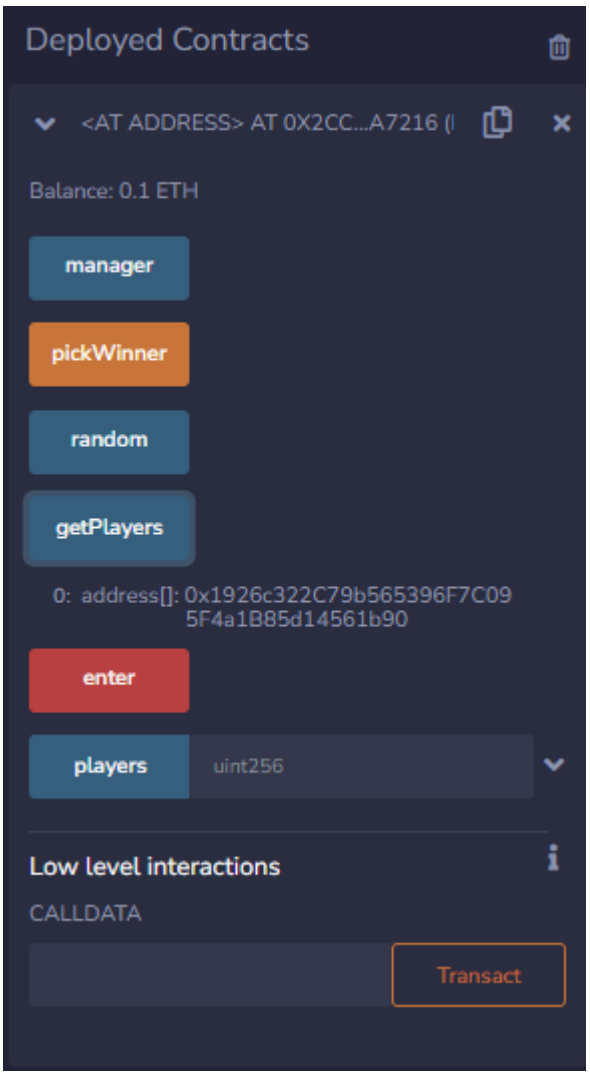

Рисунок 33 – Баланс контракта и список участников

Синим цветом выделены функции, которые возвращают данные, поэтому их использование бесплатно. Функция manager возвращает создателя смарт-контракта - адрес 0x1926c322C79b565396F7C095F4a1B85d14561b90. Функция random возвращает случайное число, а в функцию players можно передать индекс для списка участников, чтобы узнать адрес игрока под конкретным номером. Все эти функции представлены на рисунке 34.

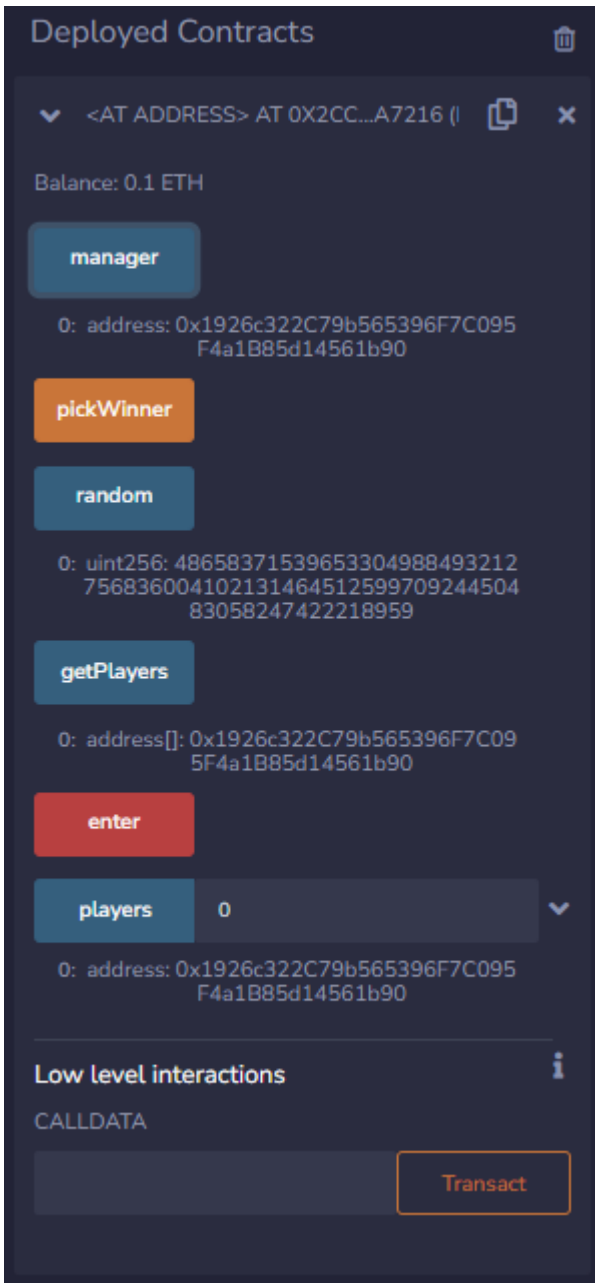

Рисунок 34 – Get-функции смарт-контракта

Последняя из оставшихся функций является pickWinner, которую может вызвать только менеджер игры. Она выделена желтым, так как при использовании нет необходимости отправлять эфир, а плата идет только за газ.

Если перейти на Etherscan и отследить транзакции с развернутым смарт-контрактом, можно увидеть транзакции enter и pickWinner, совершенные ранее. Транзакции представлены на рисунке 35.

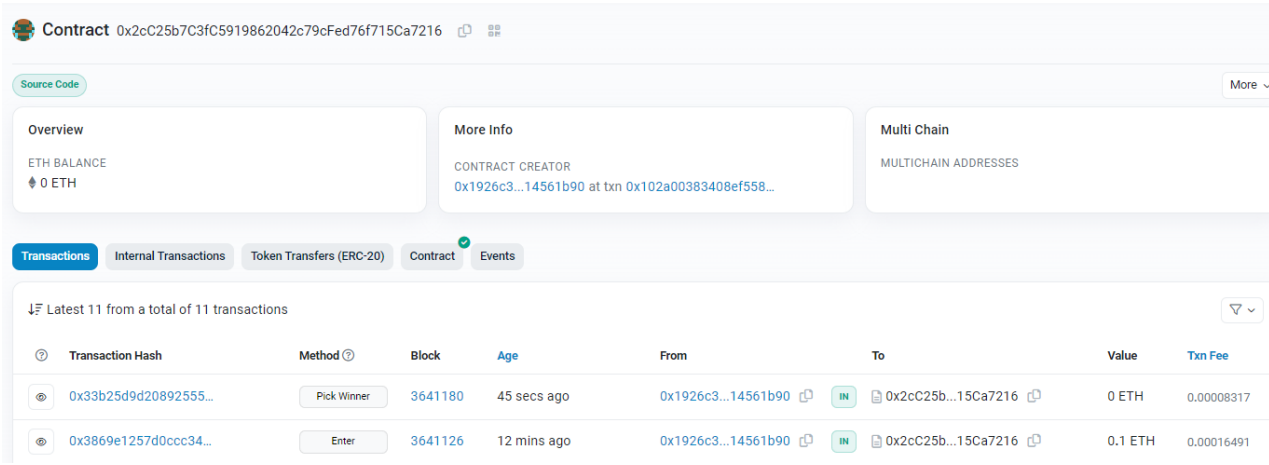

Рисунок 35 – Транзакции смарт-контракта на Etherscan

Если перейти во вкладку Internal Transactions, то можно увидеть передачу призового фонда (0.1 SepoliaETH) адресу-победителю лотереи - 0x1926c322C79b565396F7C095F4a1B85d14561b90, что представлено на рисунке 36.

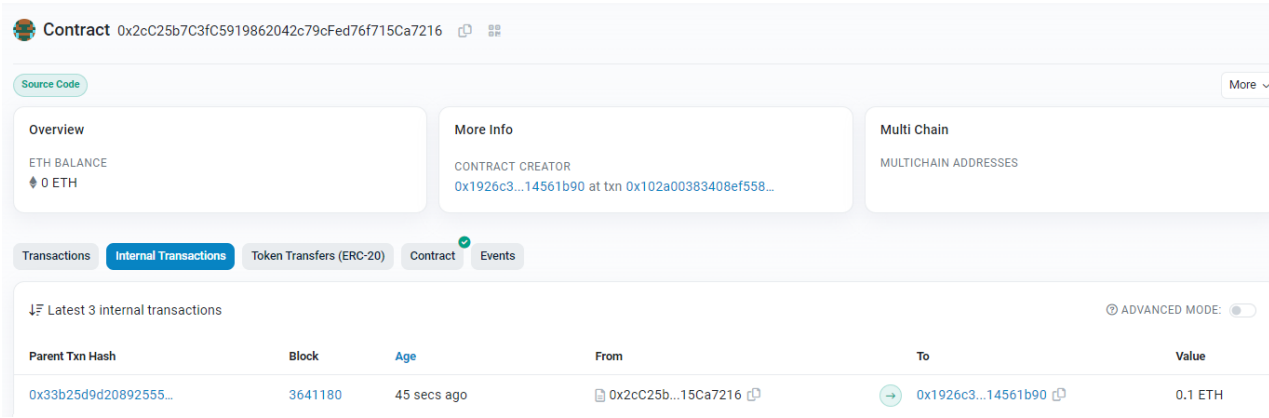

Рисунок 36 – Передача призового фонда адресу-победителю

На основании проведенного ручного тестирования можно сделать вывод, что смарт-контракт для проведения лотерей был успешно скомпилирован и развернут в сети SepoliaETH, а также, есть возможность использования его функций. Следующим этапом разработки децентрализованного приложения будет создание веб-интерфейса для более удобного взаимодействия со смарт-контрактом.

3.2 Разработка фронтенда децентрализованного сервиса

Для разработки фронтенда децентрализованного веб-приложения для проведения лотерей был выбран следующий стек технологий: React, JavaScript, web3.js, HTML, CSS, node.js версии 9.6.4, а также express.js.

На рисунке 37 представлен макет интерфейса. В блоке слева указано описание и различные ссылки, например, на проверку транзакций. В блоке справа находится форма для отправки эфира на адрес смарт-контракта для участия в лотерее. Также, указаны требования к лотерее, участники и призовой фонд. Внизу расположена кнопка для владельца контракта, чтобы определить победителя.

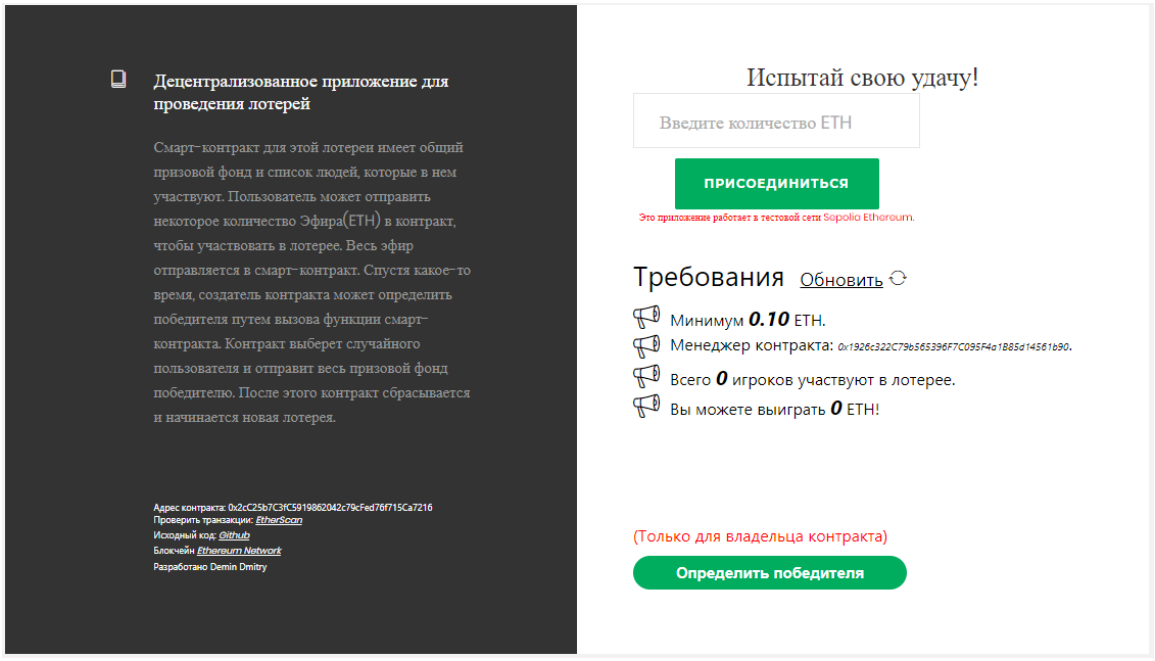

Рисунок 37 – Макет веб-интерфейса

Для фронтенда была создана следующая структура папок, представленная на рисунке 38.

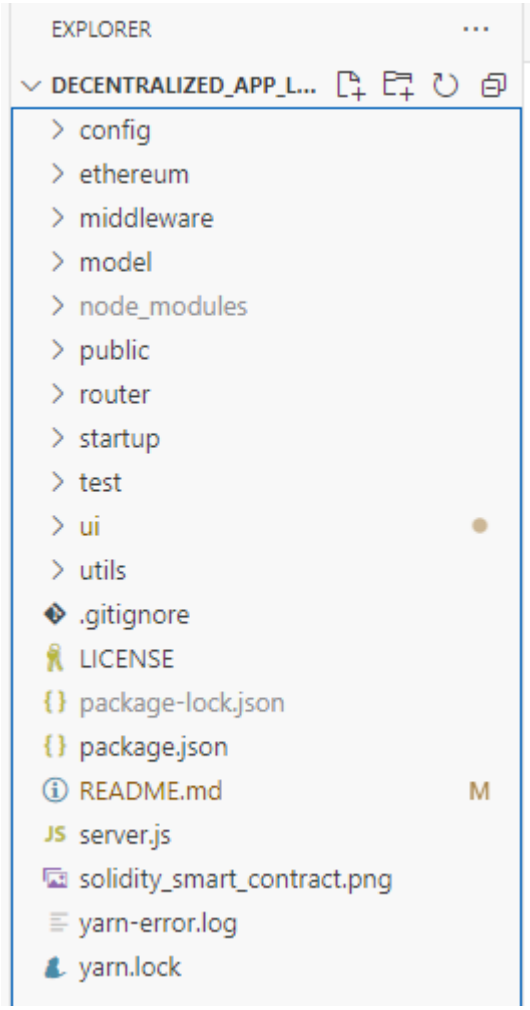

Рисунок 38 – Структура проекта для фронтенда

Папка config/ содержит переменные окружения, такие как адрес смарт-контракта.

Папка ethereum/ содержит ABI-интерфейс смарт-контракта для взаимодействия. Application Binary Interface (ABI) представляет собой набор соглашений, необходимых для доступа к низкоуровневым сервисам [40].

Папка public/ содержит HTML, CSS и другие статичные файлы.

Папка test/ содержит файлы для автоматического тестирования.

Папка ui/ содержит файлы React, такие как App.js.

Папка utils/ содержит вспомогательные скрипты для взаимодействия со смарт-контрактом.

Файл server.js необходим для запуска сервера на node.js, который требуется для корректной работы веб-интерфейса.

Основным файлом для работы фронтенда является App.js. Это компонент фреймворка React, в котором описана логика веб-приложения, а также отрисовываются и обновляются компоненты страницы.

componentDidMount() - это метод жизненного цикла, который рассматривается как при монтировании и упорядочивании раздела [41]. В децентрализованном приложении в этом методе прописывается проверка на наличие у пользователя установленного клиента MetaMask, который необходим для взаимодействия со смарт-контрактом. Код метода представлен на рисунке 39.

```
async componentDidMount() {
      const isMetaMaskPluginAvailable = web3 && lottery;
      this.setState({ isMetaMaskPluginAvailable });
      this.updateContractInfo();
  }
```

Рисунок 39 – Код метода componentDidMount()

onSumbit - метод, который вызывается при нажатии кнопки “Присоединиться”. Этот метод формирует транзакцию в MetaMask для взаимодействия со смарт-контрактом, используя введенное пользователем количество эфира для участия в лотерее. Код метода onSumbit представлен на рисунке 40.

```
onSubmit = async event => {
      event.preventDefault();
      this.setState({
      errorMessage: ''
      });
      const { isMetaMaskPluginAvailable } = this.state;
      if (!isMetaMaskPluginAvailable) {
      return this.metaMaskNotAvailable();
      }
```

```
      await window.ethereum.enable();
      const accounts = await web3.eth.getAccounts();
      this.setState({
      message: 'Транзакция в процессе. Это может занять некоторое время.',
      isTransactionIsRunning: true
      });
      try {
      await lottery.methods.enter().send({
      from: accounts[0],
      value: web3.utils.toWei(this.state.value, 'ether')
      });
      this.updateContractInfo();
      this.setState({
      message: 'Вы участвуете в лотерее!',
      value: ''
      });
      } catch (err) {
      this.setState({
      errorMessage: err.message
      });
      }
      this.setState({ isTransactionIsRunning: false });
  };
```

Рисунок 40 – Код метода onSubmit

onClickPickWinner() - метод для менеджера контракта, который позволяет выбрать победителя автоматически. Этот метод привязан к кнопке “Определить победителя” и представлен на рисунке 41.

```
onClickPickWinner = async event => {
      event.preventDefault();
      const { isMetaMaskPluginAvailable } = this.state;
      if (!isMetaMaskPluginAvailable) {
      return this.metaMaskNotAvailable();
      }

      await window.ethereum.enable();
      const accounts = await web3.eth.getAccounts();
```

```
    this.setState({
    message: 'Транзакция в процессе. Это может занять некоторое время.',
    isTransactionIsRunning: true
    });

    try {
    await lottery.methods.pickWinner().send({
    from: accounts[0]
    });
    } catch (ex) {}

    this.setState({
    message: 'Winner picked'
    });

    this.setState({ isTransactionIsRunning: false });
};
```

Рисунок 41 – Код метода onClickPickWinner

3.3 Тестирование приложения

Для тестирования смарт-контракта для проведения лотерей был разработан модуль на Lottery.test.js на языке JavaScript. Тестирование веб-интерфейса было проведено вручную.

3.3.1 Юнит-тестирование смарт-контракта

Для тестирования смарт-контракта использовались библиотеки web3, ganache-cli и assert. Библиотека ganache-cli использует локальный блокчейн для мгновенного тестирования кейсов.

Перед каждым юнит-тестом вызывается функция, которая разворачивает смарт-контракт в локальном блокчейне. Код функции представлен на рисунке 42.

```
beforeEach(async () => {
  accounts = await web3.eth.getAccounts();
```

```
  lottery =  await new web3.eth.Contract(JSON.parse(interface))
      .deploy({ data: bytecode })
      .send({
      from: accounts[0],
      gas: '1000000'
      });
});
```

Рисунок 42 – Код функции вызова перед каждым тестом

На рисунке 43 представлен код теста, который проверяет, что контракт был успешно развернут.

```
it('deploys a contract', () => {
      assert.ok(lottery.options.address);
  });
```

Рисунок 43 – Тест развертывания контракта

На рисунке 44 представлен код теста, в котором проверяется возможность участия в лотерее одного пользователя.

```
it('allows one account to enter', async () => {
      await lottery.methods.enter().send({
      from: accounts[0],
      value: web3.utils.toWei('0.02', 'ether')
      });

      const players = await lottery.methods.getPlayers().call({
      from: accounts[0]
      });

      assert.equal(accounts[0], players[0]);
      assert.equal(1, players.length);
  });
```

Рисунок 44 – Тест возможности участия в лотерее

Далее, проверяется возможность участия нескольких пользователей в лотерее. Код представлен на рисунке 45.

```
it('allows multiple accounts to enter', async () => {
      await lottery.methods.enter().send({
      from: accounts[0],
      value: web3.utils.toWei('0.02', 'ether')
      });

      await lottery.methods.enter().send({
      from: accounts[1],
      value: web3.utils.toWei('0.02', 'ether')
      });

      await lottery.methods.enter().send({
      from: accounts[2],
      value: web3.utils.toWei('0.02', 'ether')
      });

      const players = await lottery.methods.getPlayers().call({
      from: accounts[0]
      });

      assert.equal(accounts[0], players[0]);
      assert.equal(accounts[1], players[1]);
      assert.equal(accounts[2], players[2]);
      assert.equal(3, players.length);
  });
```

Рисунок 45 – Тест участия нескольких пользователей

В следующем юнит-тесте проверяется отсутствие возможности участия при отправке меньше 0.1 ETH. Код представлен на рисунке 46.

```
it('requires a minimum amount of ether to enter', async () => {
      try {
      await lottery.methods.enter().send({
      from: accounts[0],
      value: 0
      });
      assert(false);
```

```
      } catch (err) {
      assert.ok(err);
      }
  });
```

Рисунок 46 – Тест наличия > 0.1 ETH

Следующий тест проверяет, что только менеджер контракта может вызвать функцию для определения победителя. Код представлен на рисунке 47.

```
it('only manager can call pickWinner', async () => {
      try {
      await lottery.methods.pickWinner().send({
      from: accounts[1]
      });
      assert(false);
      } catch (err) {
      assert.ok(err);
      }
  });
```

Рисунок 47 – Тест определения победителя менеджером контракта

Последний тест проверяет, что при определении победителя призовой фонд отправляется адресу-победителю, а список участников лотереи обнуляется. Код представлен на рисунке 48.

```
it ('sends money to the winner and resets the players array', async () => {
      await lottery.methods.enter().send({
      from: accounts[0],
      value: web3.utils.toWei('2', 'ether')
      });

      const initialBalance = await web3.eth.getBalance(accounts[0]);
      await lottery.methods.pickWinner().send({ from: accounts[0] });
      const finalBalance = await web3.eth.getBalance(accounts[0]);
      const difference = finalBalance - initialBalance;
      assert(difference > web3.utils.toWei('1.8', 'ether'));
  });
```

Рисунок 48 – Тест отправления призового фонда победителю

3.3.2 Ручное тестирование веб-интерфейса

Для проверки работы веб-интерфейса перейдем в папку проекта и запустим фронтенд с помощью команды *npm start*. Это запустит фронтенд-сервер по адресу *localhost:3000*.

При открытии адреса в браузере откроется веб-интерфейс децентрализованного приложения для проведения лотерей, который представлен на рисунке 49.

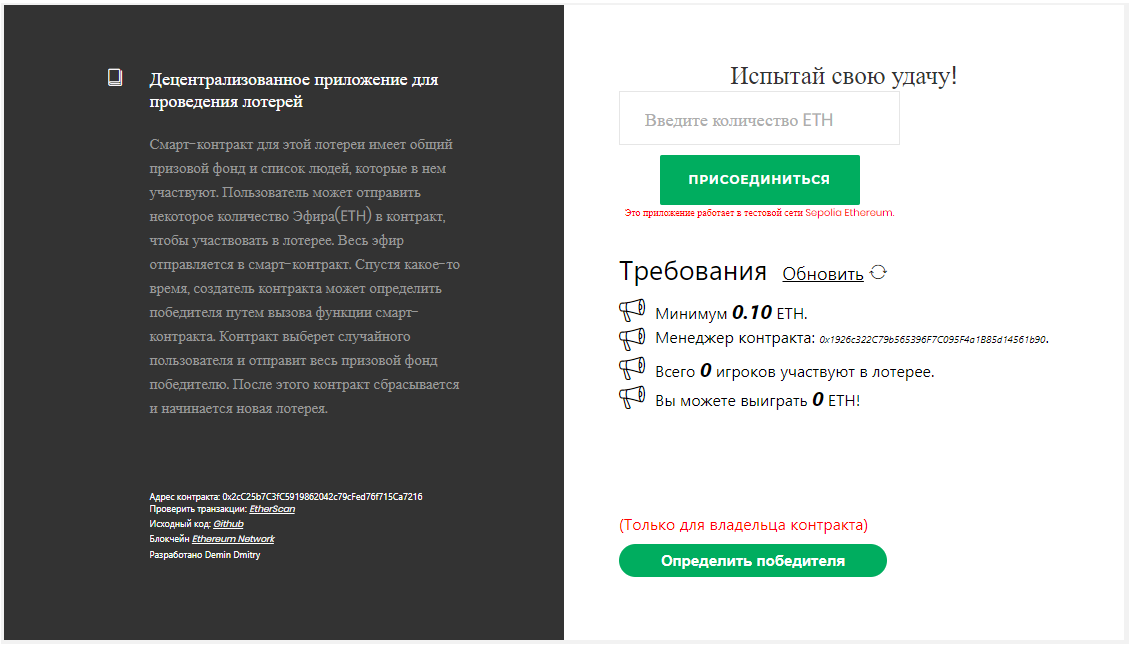

Рисунок 49 – Фронтенд приложения

Для проверки возможности участия в лотерее введем 0.1 в поле для ввода количества эфира и нажмем “Присоединиться”. После этого появляется надпись о том, что транзакция в прогрессе, которая представлена на рисунке 50 и открывается окно расширения MetaMask для отправки эфира по адресу смарт-контракт, представленное на рисунке 51.

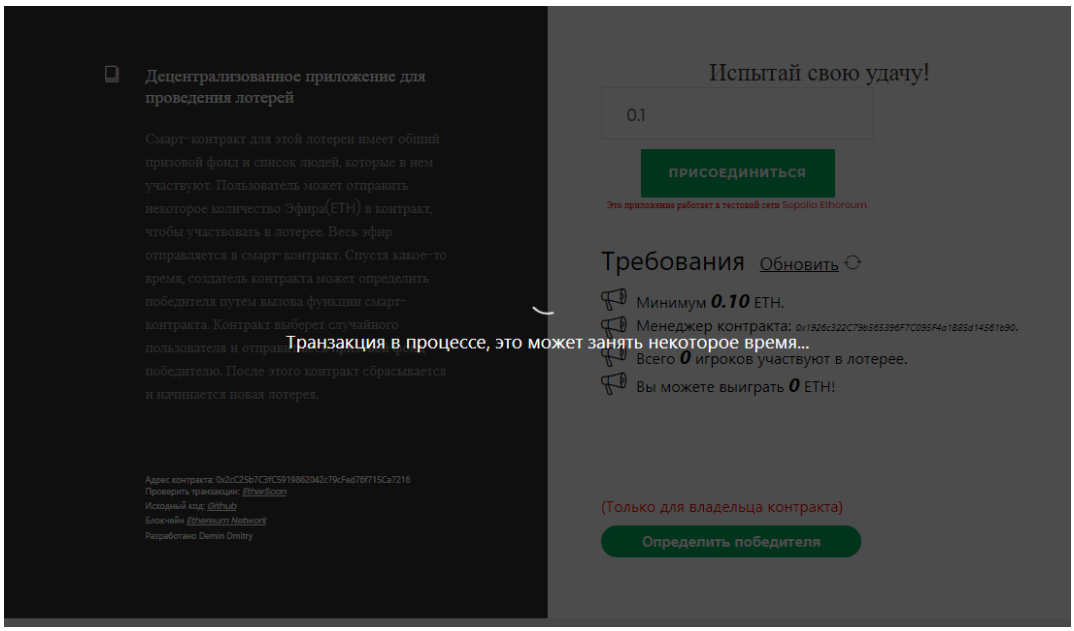

Рисунок 50 – Транзакция в прогрессе

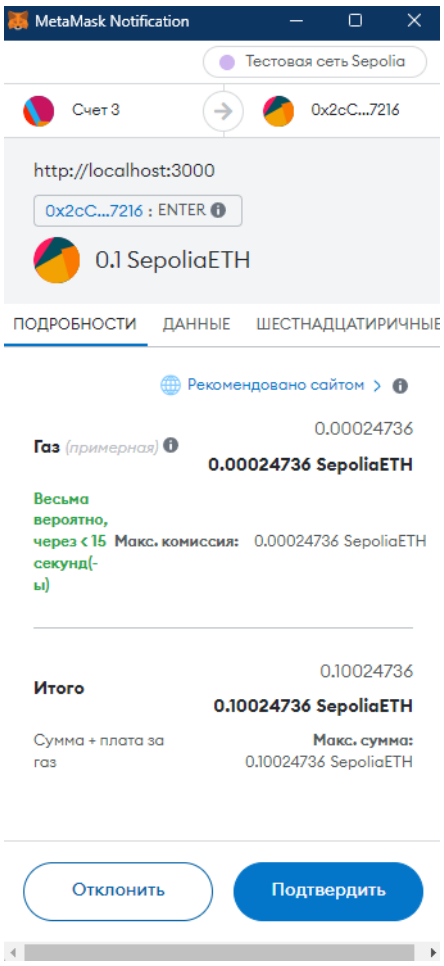

Рисунок 51 – Подтверждение транзакции для участия в лотерее

После формирования блока в сети Sepolia, информация в веб-интерфейсе обновляется, показывая, что 1 пользователь участвует в лотерее, а призовой фонд становится 0.1 ETH. Это представлено на рисунке 52.

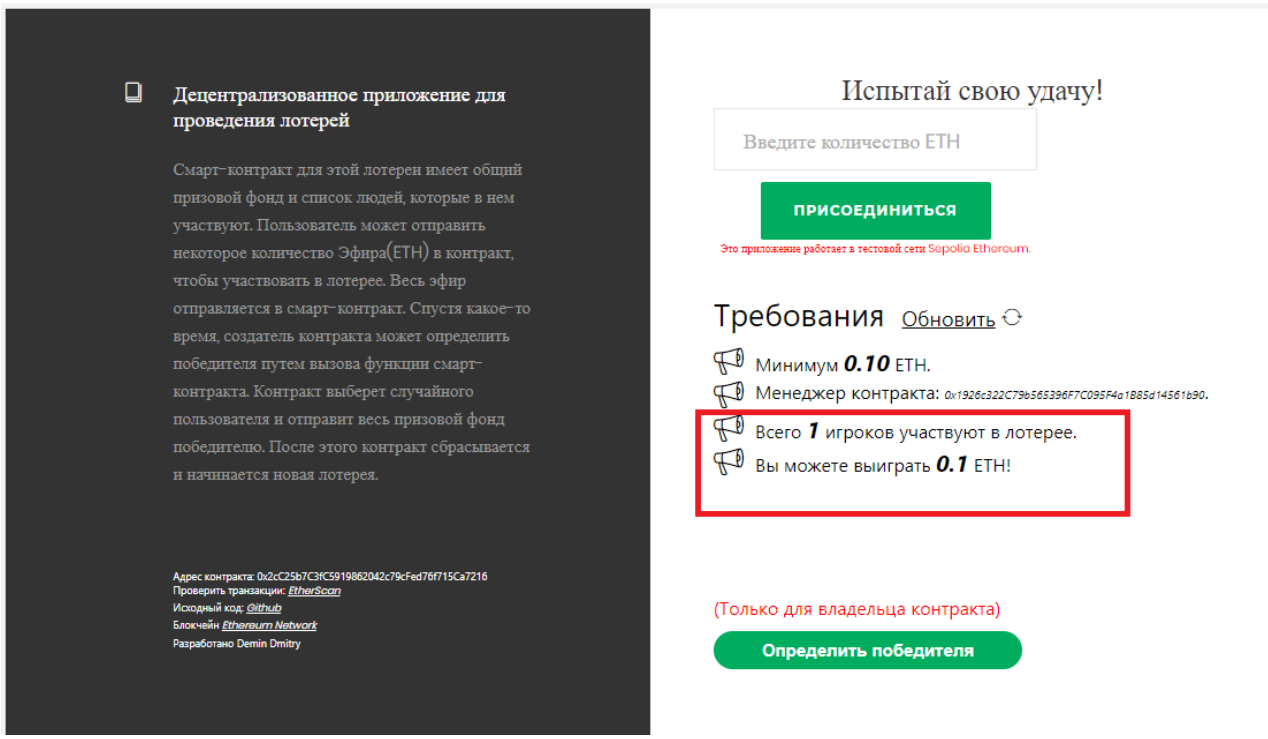

Рисунок 52 – Участники и призовой фонд лотереи

Для тестирования примем участие в лотерее от трех аккаунтов. Если перейти по ссылке “Проверить транзакции”, можно подтвердить и удостоверится в честности приложения, проверив транзакции смарт-контракта.

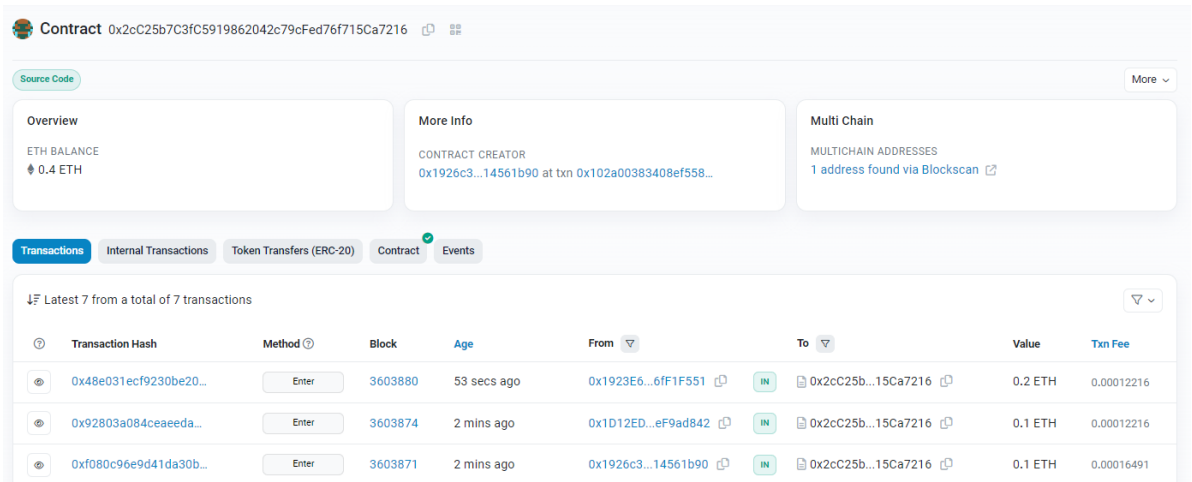

Рисунок 53 – Транзакции смарт-контракта

Также, в рамках тестирования проверим, что определить победителя может только менеджер контракта. Как видно на рисунке 54, транзакция с попыткой определить победителя не от владельца контракта была отклонена, а транзакция менеджера подтверждена.

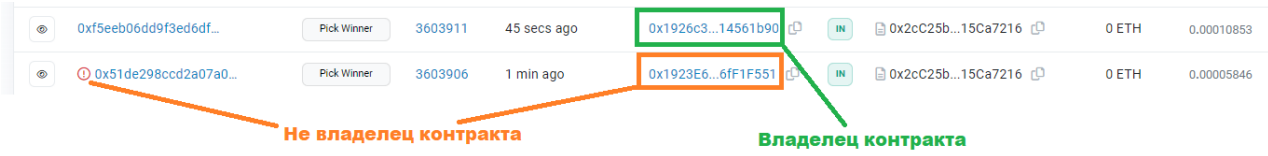

Рисунок 54 – Вызов функции Pick Winner

Также, удостоверимся в том, что победитель лотереи получил свой выигрыш, проверив транзакции смарт-контракта. На рисунке 55 видно, что один из участников получил весь призовой фонд 0.4 SepoliaETH.

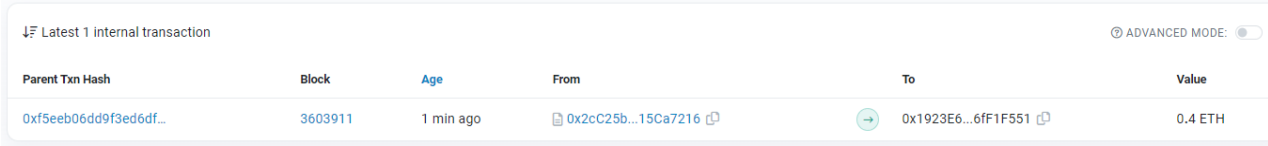

Рисунок 55 – Транзакция получения победителем призового фонда

Результаты и выводы третьей главы

В этой главе был спроектирован и разработан смарт-контракт для проведения лотерей, разработан веб-интерфейс для взаимодействия со смарт-контрактом, а также разработаны автоматические юнит тесты и проведено ручное тестирование веб-приложения. Подводя итоги, можно сказать, что в рамках третьей главы было разработано децентрализованное приложение для проведения лотерей, которое выполняет все необходимые функции.

ЗАКЛЮЧЕНИЕ

В заключении данной работы можно отметить, что поставленная цель была достигнута, а задачи, поставленные в работе, были решены. Исследование проведено с целью изучения блокчейн-технологий и их применения в разработке децентрализованных сервисов, а также создания децентрализованного сервиса для проведения лотерей.

В результате анализа блокчейн-технологий были выявлены и описаны основные принципы и преимущества децентрализованных финансовых сервисов, рассмотрены научные статьи по данной теме, проведено сравнение централизованной финансовой системы и децентрализованной. Были рассмотрены блокчейн-технологии, такие как смарт-контракты и криптовалюты, а также схема работы блокчейна и его основы - алгоритмы консенсуса и криптография. В ходе обзора типов криптовалют и блокчейнов, был выбран блокчейн Ethereum для разработки смарт-контракта в учебных целях. В научных статьях были найдены способы применения смарт-контрактов, а также их преимущества и недостатки. В конце первой главы был сделан обзор на децентрализованные приложения, определены их характеристики.

Во второй главе были рассмотрены основные этапы создания смарт-контракта, синтаксис языка программирования Solidity, проведен обзор среды разработки смарт-контрактов для Ethereum - Remix IDE. Также, были описаны шаги для создания криптовалютного кошелька в MetaMask и получение бесплатного тестового эфира. В конце второй главы были рассмотрены шаги для публикации и верификации смарт-контракта в сети для повышения его прозрачности и безопасности.

В третьей главе был разработан децентрализованный сервис для проведения лотерей. Был разработан смарт-контракт, который позволяет проводить лотереи среди пользователей, а также был создан фронтенд приложения для удобного управления лотереями. Было проведено как

автоматическое, так и ручное тестирование приложения, которое показало его работоспособность и эффективность.

Основным выводом работы является то, что блокчейн-технологии имеют большой потенциал для применения в различных областях, в том числе в финансовой сфере и разработке децентрализованных приложений. В частности, создание децентрализованных сервисов для проведения лотерей может быть эффективным способом привлечения пользователей и увеличения доходов. Децентрализованная структура вызывает доверие пользователей за счет своей прозрачности и безопасности, а в некоторых случаях помогает сэкономить за счет отсутствия необходимости создания собственной инфраструктуры.

Рекомендуется использовать полученные результаты исследования для дальнейшей разработки децентрализованных сервисов и приложений на базе блокчейн-технологий. Также рекомендуется проводить дополнительные исследования в области применения блокчейн-технологий для решения конкретных задач в различных отраслях экономики.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. What is blockchain technology? [Сайт]. – URL: https://www.ibm.com/topics/blockchain.
2. Азбука блокчейна: протоколы и алгоритмы консенсуса [Сайт]. – URL: https://habr.com/ru/articles/657413/
3. Криптография и будущее децентрализованных вычислений [Сайт]. – URL: https://habr.com/ru/articles/680650/
4. Что такое смарт-контракты на блокчейне? [Сайт]. – URL: https://vc.ru/crypto/373518-chto-takoe-smart-kontrakty-na-blokcheyne
5. Narayan Prusty. 2017. Building Blockchain Projects. Packt Publishing, p. 16. https://raw.githubusercontent.com/clojurians-org/blockchain-ebook/master/Building%20Blockchain%20Projects%20(2017).pdf
6. Introduction to dapps [Сайт]. – URL: https://ethereum.org/en/developers/docs/dapps/
7. Криптовалюта, токен, монета: в чем разница? Максимально понятно [Сайт]. – URL: https://myfin.by/stati/view/kriptovaluta-token-moneta-v-cem-raznica-maksimalno-ponatno
8. Что такое токен и чем он отличается от монеты. Простыми словами [Сайт]. – URL: https://www.rbc.ru/crypto/news/63dbefd29a79477dedd4b62c
9. Blockchain facts: what is it, how it works, and how it can be used [Сайт]. – URL: https://www.investopedia.com/terms/b/blockchain.asp
10. Что такое смарт-контракты? [Сайт]. – URL: https://academy.binance.com/ru/articles/what-are-smart-contracts
11. Автономность и безопасность. Как работают смарт-контракты [Сайт]. – URL: https://www.rbc.ru/crypto/news/600bd6409a79473b23a6d3c4
12. Что такое криптовалюта [Сайт]. – URL: https://academy.binance.com/ru/articles/what-is-a-cryptocurrency

13. Криптовалюта (крипта): что это такое и как на ней заработать [Сайт]. – URL: https://www.finam.ru/publications/item/kriptovalyuta-kripta-chto-eto-takoe-i-kak-na-neiy-zarabotat-20190605-15210/
14. Виды криптовалют - какие существуют и чем отличаются [Сайт]. – URL: https://www.binance.com/ru/blog/all/%D0%B2%D0%B8%D0%B4%D1%8B-%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B2%D0%B0%D0%BB%D1%8E%D1%82--%D0%BA%D0%B0%D0%BA%D0%B8%D0%B5-%D1%81%D1%83%D1%89%D0%B5%D1%81%D1%82%D0%B2%D1%83%D1%8E%D1%82-%D0%B8-%D1%87%D0%B5%D0%BC-%D0%BE%D1%82%D0%BB%D0%B8%D1%87%D0%B0%D1%8E%D1%82%D1%81%D1%8F-421499824684902939
15. Введение в токены ERC-20 [Сайт]. – URL: https://academy.binance.com/ru/articles/an-introduction-to-erc-20-tokens
16. ERC-20 TOKEN STANDARD [Сайт]. – URL: https://ethereum.org/en/developers/docs/standards/tokens/erc-20/
17. Sinenko, Sergey & Doroshin, Ivan. (2020). Use of Modern Means and Methods in the Organization and Management in Construction. IOP Conference Series: Materials Science and Engineering. 753. 042017. 10.1088/1757-899X/753/4/042017. [Сайт] – URL: https://www.researchgate.net/publication/339776218_Use_of_Modern_Means_and_Methods_in_the_Organization_and_Management_in_Construction
18. W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng and V. C. M. Leung, "Decentralized Applications: The Blockchain-Empowered Software System," in IEEE Access, vol. 6, pp. 53019-53033, 2018, doi: 10.1109/ACCESS.2018.2870644.
19. M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, ‘‘Blockchain,’’ Bus. Inf. Syst. Eng., vol. 59, no. 3, pp. 183–187, Mar. 2017
20. Dirk A. Zetzsche‡ , Douglas W. Arner , and Ross P. Buckley. Decentralized Finance. Journal of Financial Regulation, 2020, 6, 172–203. [Сайт]. – URL:

https://deliverypdf.ssrn.com/delivery.php?ID=319121006065081095003108066104096104056074007011089060073092003025097003014072001125123036060024038014002024014101119091099016103054002035031024113104113112125029064028082072068026096084030020105106091003086072066108094102014116072094031115009002008&EXT=pdf&INDEX=TRUE

21. Johannes Rude Jensen, Victor von Wachter, and Omri Ross. An Introduction to Decentralized Finance (DeFi). Department of Computer Science, University of Copenhagen, Copenhagen, Denmark. [Сайт]. – URL: https://csimq-journals.rtu.lv/article/view/csimq.2021-26.03/2570
22. Kaihua Qin, Liyi Zhou, Yaroslav Afonin, Ludovico Lazzaretti, Arthur Gervais. CeFi vs. DeFi -- Comparing Centralized to Decentralized Finance. [Сайт]. – URL: https://arxiv.org/abs/2106.08157
23. Технология блокчейн и ее практическое применение. Мащенко П.Л., Пилипенко М.О. [Сайт]. – URL: https://cyberleninka.ru/article/n/tehnologiya-blokcheyn-i-ee-prakticheskoe-primenenie/viewer
24. Diego Ongaro and John Ousterhout. Stanford University. In Search of an Understandable Consensus Algorithm. [Сайт]. – URL: https://www.usenix.org/system/files/conference/atc14/atc14-paper-ongaro.pdf
25. Giang-Truong Nguyen and Kyungbaek Kim. A Survey about Consensus Algorithms Used in Blockchain. J Inf Process Syst, Vol.14, No.1, pp.101~128, February 2018. [Сайт]. – URL: https://volunteerscience.com/media/research_teams/Luke's%20Sandbox/consent_forms/revised_proposal.2.pdf
26. Ayesha Afzal and Aiman Asif. Cryptocurrencies, Blockchain and Regulation: A Review. The Lahore Journal of Economics. 24 : 1 (Summer 2019): pp. 103–130. [Сайт]. – URL: https://d1wqtxts1xzle7.cloudfront.net/86730094/05_20LJE01_20Afzal_20ED_20AAC_20ttc-libre.pdf?1653939191=&response-content-disposition=inline%3B+filename%3DCryptocurrencies_Blockchain_and_Regulati.pdf&Expires=1686036247&Signature=JyUYomFCZn~3moyYfkleunK1yb

F7Cygw1-BHqhTEEA6WGp1WsLo3q~DegKUx1-27JfjNKOgdmil~pW6zTiwkcWFvfWpQqRy9PkQi0-WPLkJL4IGkob0z9PRmjA22xdHb2gB~cjyAbBjSXYYzgPQDVFvlGH1PES3e57ErA8f~f7gtTLHD0ZLkNIf3i5a4W~Bf~3b93vE0RsuZPpsegdQmlrgDMnw4ihQd-LQsfEDj68cnDh04xCJqwqJoHGDMnrp9NTFVUr7akNRKD9xzR~rvyiADPsHaJ0viK8UfED73aqEtRWWtOHqtxJ8opA8ijj-JjkkdAR1P4wb7YB5ksW9dBg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

27. Bitcoin Energy Consumption Index. Digiconomist. [Сайт]. – URL: https://digiconomist.net/bitcoin-energy-consumption
28. Zibin Zheng,Shaoan Xie, Hong-Ning Dai, Weili Chen, Xiangping Chen, Jian Weng, Muhammad Imran. An Overview on Smart Contracts: Challenges, Advances and Platforms. [Сайт]. – URL: https://arxiv.org/pdf/1912.10370.pdf
29. Merit Ko˜lvart, Margus Poola, and Addi Rull. Smart Contracts. [Сайт]. – URL: https://www.researchgate.net/profile/Tanel-Kerikmaee/publication/314538429_The_Future_of_Law_and_Technologies/links/58c32c8445851538eb809e8c/The-Future-of-Law-and-Technologies.pdf#page=141
30. Xin XIA, Xuan-Bach D. LE, Pavneet Singh KOCHHAR, David LO, Weiqin ZOU. Smart contract development: Challenges and opportunities. [Сайт]. – URL: https://ink.library.smu.edu.sg/cgi/viewcontent.cgi?article=5499&context=sis_research
31. Max Raskin. THE LAW AND LEGALITY OF SMART CONTRACTS. [Сайт]. – URL: https://moodle.epfl.ch/pluginfile.php/2861851/mod_resource/content/1/Raskin-1-GEO.-L.-TECH.-REV.-305-.pdf
32. KONSTANTINOS CHRISTIDIS, (Graduate Student Member, IEEE), AND MICHAEL DEVETSIKIOTIS, (Fellow, IEEE). Blockchains and Smart

Contracts for the Internet of Things. [Сайт]. – URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7467408

33. VICTOR YOUDOM KEMMOE, WILLIAM STONE, JEEHYEONG KIM, DAEYOUNG KIM, AND JUNGGAB SON, (Member, IEEE). Recent Advances in Smart Contracts: A Technical Overview and State of the Art. [Сайт]. – URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9125932
34. WEI CAI 1,2, (Member, IEEE), ZEHUA WANG2,3, (Member, IEEE), JASON B. ERNST3, (Member, IEEE), ZHEN HONG2, (Student Member, IEEE), CHEN FENG 4, (Member, IEEE), AND VICTOR C. M. LEUNG 2, (Fellow, IEEE). Decentralized Applications: The Blockchain-Empowered Software System. [Сайт]. – URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8466786
35. Kaidong Wu. Key Lab of High-Confidence Software Technology, MoE (Peking University), Beijing, China. An Empirical Study of Blockchain-based Decentralized Applications. [Сайт]. – URL: https://arxiv.org/pdf/1902.04969.pdf
36. Vitalik Buterin. A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM. [Сайт]. – URL: https://finpedia.vn/wp-content/uploads/2022/02/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf
37. Kaidong Wu, Yun Ma, Gang Huang, Xuanzhe Liu. Key Lab of High-Confidence Software Technology, MoE (Peking University), Beijing, China. A First Look at Blockchain-based Decentralized Applications. [Сайт]. – URL: https://arxiv.org/pdf/1909.00939.pdf
38. Nicolas Six, Nicolas Herbaut, Camille Salinesi. Blockchain software patterns for the design of decentralized applications: A systematic literature review. [Сайт]. – URL: https://www.sciencedirect.com/science/article/pii/S209672092200001X/pdfft?

md5=2ce94848f3b614ed4465566b5f21b040&pid=1-s2.0-S209672092200001X-main.pdf

39. Claudia Pop, Tudor Cioara, Ionut Anghel, Marcel Antal and Ioan Salomie. Blockchain based Decentralized Applications: Technology Review and Development Guidelines. [Сайт]. – URL: https://arxiv.org/pdf/2003.07131.pdf
40. Алиев Иса Адамович. Уязвимости смарт-контрактов блокчейн-платформы Ethereum. [Сайт]. – URL: https://cyberleninka.ru/article/n/uyazvimosti-smart-kontraktov-blokcheyn-platformy-ethereum/viewer
41. Solidity. [Сайт]. – URL: https://docs.soliditylang.org/en/v0.8.20/
42. Remix Project. [Сайт]. – URL: https://remix-project.org/
43. Solidity, the Smart Contract Programming Language [Сайт]. – URL: https://github.com/ethereum/solidity
44. MetaMask. [Сайт]. – URL: https://metamask.io/
45. MetaMask Cryptocurrency Wallet Review. [Сайт]. – URL: https://www.investopedia.com/metamask-cryptocurrency-wallet-review-5235562
46. Мамаева Наталья Владимировна. Эфириум. [Сайт]. – URL: https://cyberleninka.ru/article/n/efirium/viewer
47. Sepolia Faucet. [Сайт]. – URL: https://sepoliafaucet.com/
48. Gas (Ethereum): How Gas Fees Work on the Ethereum Blockchain. [Сайт]. – URL: https://www.investopedia.com/terms/g/gas-ethereum.asp
49. Sheping Zhai et al 2019 J. Phys.: Conf. Ser. 1168 032077. Research on the Application of Cryptography on the Blockchain. [Сайт]. – URL: https://iopscience.iop.org/article/10.1088/1742-6596/1168/3/032077/pdf
50. Ahmed Kosba , Andrew Miller , Elaine Shi , Zikai Wen , Charalampos Papamanthou. University of Maryland and Cornell University. Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. [Сайт]. – URL: https://eprint.iacr.org/2015/675.pdf

51. MAYANK RAIKWAR , DANILO GLIGOROSKI , AND KATINA KRALEVSKA. SoK of Used Cryptography in Blockchain. [Сайт]. – URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8865045
52. Christian Cachin. IBM Research – Zurich. Blockchain, cryptography, and consensus. [Сайт]. – URL: https://crypto.unibe.ch/talks/20170622-blockchain-ice.pdf
53. Gautam Srivastava, Shalini Dhar, Ashutosh Dhar Dwivedi, Jorge Crichigno. Blockchain Education. [Сайт]. – URL: http://ce.sc.edu/cyberinfra/docs/publications/PID5824833.pdf
54. Harry Halpin, Marta Piekarska. Introduction to Security and Privacy on the Blockchain. [Сайт]. – URL: https://inria.hal.science/hal-01673293/document
55. Structure of a Contract - Solidity. [Сайт]. – URL: https://docs.soliditylang.org/en/latest/structure-of-a-contract.html
56. Layout of a Solidity Source File. [Сайт]. – URL: https://docs.soliditylang.org/en/latest/layout-of-source-files.html
57. Types - Solidity. [Сайт]. – URL: https://docs.soliditylang.org/en/latest/types.html
58. Alchemy - The web3 development platform. [Сайт]. – URL: https://www.alchemy.com/
59. Sepolia Faucet. [Сайт]. – URL: https://sepoliafaucet.com/
60. Основы криптобезопасности. [Сайт]. – URL: https://habr.com/ru/companies/first/articles/696452/
61. Decentralized application platform for private key management. [Сайт]. – URL: https://patents.google.com/patent/US20200374113A1/en?q=(decentralized+application)&oq=decentralized+application
62. System, method, and decentralized application for blockchain-based gambling. [Сайт]. – URL: https://patents.google.com/patent/US20200027315A1/en?q=(decentralized+application)&oq=decentralized+application

63. System and method for implementing a blockchain-based decentralized application. [Сайт]. – URL: https://patents.google.com/patent/US11163775B2/en?q=(decentralized+application)&oq=decentralized+application
64. Decentralized application deployment method for web application middleware, system and computer program thereof. [Сайт]. – URL: https://patents.google.com/patent/JP5254547B2/en?q=(decentralized+application)&oq=decentralized+application
65. System and method for blockchain-based decentralized application development. [Сайт]. – URL: https://patents.google.com/patent/US11086621B2/en?q=(decentralized+application)&oq=decentralized+application