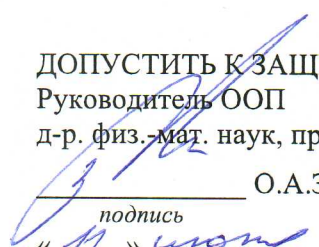


Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Научно-образовательный центр «Высшая ИТ школа»

ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК
Руководитель ООП
д-р. физ.-мат. наук, профессор


_____ О.А.Змеев
подпись
« 11 » июня 2022 г.

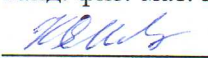
ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Разработка системы проверки документов с использованием блокчейна

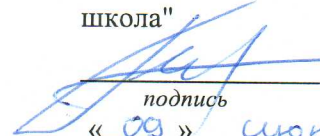
по направлению подготовки 09.03.04 Программная инженерия
направленность (профиль) «Программная инженерия»

Текин Онур

Руководитель ВКР
канд. физ.-мат. наук, ассистент


_____ К.С. Ким
подпись
« 10 » июня 2022 г.

Консультант ВКР
ассистент НОЦ "Высшая ИТ
школа"


_____ Д.С. Красавин
подпись
« 09 » июня 2022 г.

Автор работы
студент группы № 971810


_____ О. Текин
подпись
« 08 » июня 2022 г.

Министерство науки и высшего образования Российской Федерации.
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
НОЦ «Высшая ИТ школа»

УТВЕРЖДАЮ
руководитель ООП
д-р. физ.-мат. наук, профессор
О.А.Змеев
« 27 » февраля 20 22 г.

ЗАДАНИЕ

по выполнению выпускной квалификационной работы бакалавра обучающемуся
Текин Онур

(Ф.И.О. обучающегося)

по направлению подготовки Программная инженерия, направленность «Программная инженерия»

1. Тема выпускной квалификационной работы бакалавра
Разработка системы проверки документов с использованием блокчейна

2. Срок сдачи обучающимся выполненной выпускной квалификационной работы:

а) в учебный офис – « 29 » июня 20 22 г.

б) в ГЭК – « 11 » июля 20 22 г.

3. Исходные данные к работе:

Цель работы: разработка системы проверки документов с использованием блокчейна. Задачи:

цели и задачи ВКР, ожидаемые результаты

Исследование алгоритмов хеширования; Исследования по хранению и проверке данных в

блокчейне; Разработка контракта на проверку документов и контракта на управление проверкой

документов; Тестирование контрактов. Результат: система проверки документов разработана

Организация, по тематике которой выполняется работа

Общество с ограниченной ответственностью «КРИПТОН СТУДИО»

Руководитель выпускной квалификационной работы

канд. физ.-мат. наук, ассистент

(должность, место работы)


(подпись)

К.С. Ким

(И.О. Фамилия)

Консультант выпускной квалификационной работы

ассистент НОЦ "Высшая ИТ школа"

(должность, место работы)

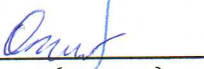

(подпись)

Д.С. Красавин

(И.О. Фамилия)

Задание принял к исполнению

« 18 » 02 2022 г.
(дата)


(подпись)

О. Текин

(И.О. Фамилия)

ОГЛАВЛЕНИЕ

Введение.....	3
1. Справочная информация.....	5
1. Блокчейн	6
1. Криптовалюты и смарт-контракты.....	6
2. Разрешенный и неразрешенный блокчейн.....	10
3. Риски безопасности блокчейна	14
4. Блокчейн-трилемма	15
5. Примеры использования блокчейна.....	16
2. Ethereum	19
2. Анализ и фиксация требований.....	24
1. Функциональные требования	24
2. Нефункциональные требования.....	28
3. Сценарии вариантов использования.....	29
3. Проектирование	42
1. Диаграмма классов	43
4. Реализация	45
1. Язык смарт-контрактов.....	45
2. Настройка контроллера и создателя документа.....	48
3. Поместить документ на проверку	51
4. Подписать документ	54
5. Проверить легитимность документа	56
6. Взаимодействие с блокчейном и развертывание смарт-контрактов.....	59
7. Тестирование.....	61
Заключение	63
СПИСОК ЛИТЕРАТУРЫ	64

ВВЕДЕНИЕ

Документы о присвоении степени выдаются после завершения программ обучения и считаются доказательством завершения соответствующих программ обучения. Таким образом, работодатели используют документы об образовании для подтверждения истории образования претендентов. Аналогичным образом, при поступлении в другие учебные заведения для получения высшего образования также требуется подтверждение документов об образовании. Эта ключевая роль документов об образовании привлекает мошенников и побуждает их пытаться получить работу или поступить в вуз на основании поддельных документов об образовании. Следовательно, работодателям или учебным заведениям необходимо проверять документы об образовании, прежде чем принимать их. Ежегодно университеты тратят миллионы долларов на обработку запросов о проверке документов об образовании. В результате возникает необходимость упрощения процесса проверки документов об образовании наряду с сокращением затрат.

Технологии изменили наш мир, и ожидаемое будущее будет зависеть от технологий. Недавно была разработана новая технология, названная технологией блокчейн (Blockchain). Технология блокчейн привлекла внимание исследователей и разработчиков, и они применили эту технологию в различных областях, таких как финансы, здравоохранение, банки и образование. Технология блокчейн набирает популярность с каждым днем, потому что все больше и больше областей принимают ее на вооружение. Технология блокчейн, известная как Distributed Ledger Technology (DLT), связана с базами данных и обеспечивает высокий уровень безопасности. Преимуществом технологии Blockchain является то, что история цифровых активов не может быть изменена, поскольку используется криптографическое хеширование.

Целью данного исследования является разработка системы проверки документов с использованием блокчейна.

Для достижения цели данного исследования были поставлены следующие задачи:

- Исследование алгоритмов хеширования
- Исследования по хранению и проверке данных в блокчейне
- Разработка контракта на проверку документов и контракта на управление проверкой документов
- Тестирование контрактов

1. СПРАВОЧНАЯ ИНФОРМАЦИЯ

Этот параграф начинается с представления темы блокчейна. Будут объяснены криптовалюты и смарт-контракты. Будет проведено сравнение двух различных технологий - блокчейн с разрешением и блокчейн без разрешения. Затем будут рассмотрены риски безопасности блокчейна. В заключении этого подраздела приводится несколько примеров использования блокчейна в различных сферах жизни общества. После этого будет представлена и объяснена открытая программная платформа на основе технологии блокчейн - Ethereum. В этом документе целый подраздел посвящен Ethereum, поскольку именно эта технология блокчейн использовалась в проекте.

1. Блокчейн

Блокчейн — это технология, которая позволяет множеству участников совместно вести базу данных, основанную на определенном типе консенсуса. Этот консенсус обусловлен необходимостью того, чтобы большинство участников учитывали вклад друг друга в расширение базы данных. Таким образом, никакое меньшинство участников не может изменить базу данных, испортить ее данные, внести незаконные изменения или прекратить ее обслуживание. Эта база данных/электронная книга не зависит от какого-либо центрального органа и используется совместно распределенными компьютерами в одноранговой сети. Каждый новый блок формируется набором предметов/транзакций, которые будут прикреплены к концу цепочки, блоки надежно связаны между собой с помощью криптографии [1].

1. Криптовалюты и смарт-контракты

По определению, криптовалюта — это цифровая валюта, предназначенная для работы в качестве средства обмена через компьютерную сеть, которая не зависит от какого-либо центрального органа, такого как правительство или банк, для ее поддержки или обслуживания [2]. Криптовалюты были первым применением технологии блокчейн, которая использовала публичную бухгалтерскую книгу для денежных транзакций. Биткойн был первым практическим решением, где блокчейн был реплицирован и доступен во всей сети.

Каждый раз, когда пользователь запрашивает новую транзакцию, она должна быть проверена другими компьютерами (майнерами) в сети. Если запрос подтверждается, майнеры объединяют его в новый блок. Каждый блок содержит хэш-указатель, который ссылается на предыдущий блок. Таким образом, невозможно вставить, удалить или изменить какой-либо блок в середине цепочки без пересчета всех хэшей последующих блоков.

Если кто-то произведет какое-либо изменение, хэш, хранящийся в следующем блоке, не будет совпадать с хэшем измененного блока.

Майнеры постоянно получают уведомления о запросах на транзакции и собирают их для создания нового блока. Это приводит к конкуренции между ними, поскольку тот, кто первым создаст действительный блок, получает деньги за свои услуги. Этот процесс называется вознаграждением за блок. По определению, вознаграждение за блок — это криптовалюта, которая начисляется майнеру, когда он успешно подтверждает новый блок [3]. В любом случае, остается возможность пересчитать хэши всех последующих блоков для майнера, который хочет вставить измененный блок в середину цепочки. Чтобы решить эту проблему, блок может быть согласован и присоединен только в том случае, если хэш-функция блока (proof-of-work) была решена, удовлетворяя математическим условиям [4]. Поэтому пересчет всех последующих блоков потребует гораздо больше времени и вычислительной мощности. Таким образом, неизменяемость является одним из основных принципов технологии блокчейн.

Существует несколько различных алгоритмов консенсуса блокчейна, каждый из которых имеет свои преимущества и недостатки. В настоящее время в Bitcoin и Ethereum принят алгоритм proof-of-work [5]. Этот алгоритм создает соревнование вычислительных мощностей, в котором участвующие узлы должны решить криптографическую головоломку или какую-то сложную математическую задачу, чтобы выиграть право на создание нового блока и получить соответствующее вознаграждение. Proof-of-stake - еще один популярный алгоритм консенсуса. Вместо того, чтобы зависеть от вычислительной мощности, этот алгоритм выбирает узел, который будет создавать новый блок по его соответствующей ставке. Это также энергосберегающий протокол, поскольку он поощряет приобретение внутренней валюты вместо того, чтобы потреблять большое количество вычислительной мощности для достижения консенсуса [5].

Таблица 1 — Различие между PoW и PoS

	Proof of Work	Proof of Stake
Майнинг/валидирование блока	Объем вычислительной работы определяет вероятность добычи блока.	Сумма ставки или количество монет определяет вероятность подтверждения нового блока.
Распределение вознаграждения	Тот, кто добывает блок первым, получает вознаграждение.	Валидатор не получает вознаграждение за блок, поскольку ему выплачивается сетевая комиссия.
Конкурс	Майнеры должны соревноваться в решении сложных головоломок, используя свою вычислительную мощность.	Алгоритм определяет победителя в зависимости от размера его ставки. Алгоритм определяет победителя, основываясь на размере его ставки.
Централизация	Решения POW все чаще назначаются для крупномасштабных операций, они централизованы по своей природе.	Алгоритм определяет победителя в зависимости от размера его ставки.

Таблица 1 — (Продолжение)

Специализированное оборудование	Для добычи монет используются специализированные интегральные схемы (ASICS) и графические процессоры (GPU).	Для систем на базе PoS достаточно стандартного устройства серверного класса.
Добавление вредоносного блока	Чтобы внедрить вредоносный блок, хакерам потребуется 51 % вычислительной мощности.	Хакеры должны обладать 51% всей криптовалюты в сети.
Эффективность и надежность	Системы POW менее энергоэффективны и менее дороги, но они более надежны.	POS-системы гораздо более экономичны и энергоэффективны, хотя и менее надежны.
Безопасность	Чем больше хэш, тем более безопасной является сеть.	Стейкинг помогает заблокировать криптоактивы для обеспечения безопасности сети в обмен на вознаграждение.
Форкинг	Благодаря экономическому стимулу, системы POW естественным	POS-системы автоматически не препятствуют форкингу.

	образом предотвращают постоянные вилки.	
--	---	--

После дебюта биткойна стало ясно, что такая система будет полезнее, чем просто денежные транзакции. Люди начали размышлять, какие еще приложения могут работать на блокчейне. Майнеры уже запускали небольшие программы для подтверждения транзакций, это означает, что запуск более сложных программ был вполне возможен.

Второе поколение блокчейна было разработано на основе программируемой инфраструктуры, начиная с Ethereum. Ethereum заменил простые валютные транзакции на более универсальные автономные программы, работающие в сети блокчейн и называемые смарт-контрактами. Пользователи могут взаимодействовать с этими программами, отправляя транзакции с определенными инструкциями для обработки. Смарт-контракты — это самоисполняющиеся контракты, в которых условия соглашения между покупателем и продавцом записаны непосредственно в строках кода [6]. Код и содержащиеся в нем соглашения существуют в распределенной, децентрализованной сети блокчейн и могут использоваться для достижения договоренностей и решения общих проблем с минимальным доверием. В сети блокчейна Bitcoin можно создавать самоисполняющиеся контракты, однако эти контракты очень просты и ограничены из-за соответствующего скриптового языка, который не позволяет создавать сложные потоки управления.

2. Разрешенный и неразрешенный блокчейн

Блокчейн может быть как разрешенным, с закрытой ограниченной группой участников, так и неразрешенным, допускающим публичное использование.

Блокчейн без разрешений не имеет владельца, и каждый может внести свой вклад и сохранить копию бухгалтерской книги. Она децентрализована и открыта для общественности. Поэтому не существует ни привратников, ни цензуры. Целостность бухгалтерской книги гарантируется ее участниками путем достижения консенсуса между ними [7].

Любой пользователь может свободно начать взаимодействовать с сетью, поскольку нет никаких ограничений на то, какие пользователи могут отправлять транзакции или создавать новые блоки. Более того, каждый имеет возможность управлять узлом и работать с майнинговыми протоколами для проверки транзакций.

Самыми большими преимуществами блокчейн без разрешений являются следующие условия:

- Сети без разрешений должны быть децентрализованными, то есть ни один центральный орган не сможет изменить или переписать любую часть бухгалтерской книги.
- Анонимность - еще один важный атрибут, не требующий от пользователей предоставления личной информации при создании адреса или совершении транзакции. Однако, в некоторых случаях, по юридическим причинам может быть затребована .
- Прозрачность является обязательным условием для того, чтобы пользователи доверяли сети, поэтому необходимо предоставить пользователям доступ ко всей информации в блокчейне без ограничений.

Bitcoin и Ethereum - самые популярные примеры блокчейн-систем без права доступа, где майнеры (верификаторы) имеют решающее значение для предотвращения вставки или изменения базы данных участниками

транзакций, противоречащих правилам. Чтобы транзакция попала в базу данных, она должна быть предварительно проверена.

С ростом популярности технологии блокчейн без права доступа финансовые учреждения начали понимать, что открытая структура противоречит их потребностям. Пользователи, представленные только буквенно-цифровым публичным адресом без каких-либо доказательств их реальной личности, вызывали серьезную озабоченность у этих учреждений.

Разрешенные блокчейны были созданы для решения этих проблем путем разработки блокчейна, в котором обработка транзакций осуществляется четко определенным набором идентифицированных субъектов, а все данные в системе могут быть видны только ограниченной группе субъектов. Таким образом, блокчейн с разрешением является закрытым или имеет уровень контроля доступа. Этот дополнительный уровень безопасности позволяет участникам выполнять только те действия, на которые они уполномочены. В блокчейне с разрешением пользователь должен получить разрешение от владельца сети, чтобы стать частью этой сети [7].

Такая система предназначена для субъектов, которые уже имеют хотя бы небольшую степень доверия друг к другу, поэтому только заранее определенный набор субъектов может добавлять новые блоки в базу данных, что означает, что больше нет необходимости в proof-of-work майнинге или транзакционных токенах (криптовалютах).

Приняв систему блокчейн с правами доступа, финансовые учреждения могут предоставить своим клиентам ограниченный доступ на чтение к транзакциям и заголовкам блоков, обеспечивая прозрачный и надежный способ обеспечения безопасности средств своих клиентов. Полный доступ на чтение также может быть предоставлен регулирующим органам для обеспечения требуемого уровня соответствия.

В следующей таблице приведены основные различия между блокчейнами с разрешением и без разрешения.

Таблица 2 — Различия между блокчейнами без разрешений и с разрешениями

Без разрешения	Разрешенный
Публичный; не требует разрешения для присоединения к сети и участия в консенсусе	Приватный; требуется разрешение на присоединение к сети и участие в консенсусе
Полностью децентрализованная, без привратников	От постепенной децентрализации до полной централизации; орган управления выступает в качестве привратника
Транзакции прозрачны и доступны	Транзакции являются приватными
Низкая скорость транзакций	Высокая скорость и производительность
Сложность масштабирования	Масштабируемая сеть
Потребляет много энергии	Энергоэффективный
Разработка с открытым исходным кодом; больше информации, так как больше разработчиков	Разработка частными организациями; меньшая осведомленность
Без доверия; математика предоставляет доказательства	Руководящий орган обеспечивает определенный уровень доверия к системе
Достижение консенсуса занимает больше времени из-за размера сети и сложности вычислений	Консенсус достигается быстро, поскольку вычисления менее сложны из-за ограниченного числа пользователей

3. Риски безопасности блокчейна

В настоящее время блокчейн считается прорывной технологией, которая может быть применена в самых разных областях повседневной социальной и деловой деятельности. Однако, несмотря на то что технология блокчейн представляет собой инновацию и большой набор возможностей, ее использование все равно будет связано с определенными рисками.

Контроль над децентрализованной сетью является одной из основных проблем, связанных с использованием блокчейна. Атака Sybil — это тип атаки на службу компьютерной сети, при которой злоумышленник подрывает систему репутации службы путем создания большого количества псевдонимных идентификаторов и использует их для получения непропорционально большого влияния. Атака Sybil осуществляется при попытке контролировать одноранговую сеть путем создания множества поддельных идентификаторов. Эти поддельные пользователи кажутся уникальными. Однако, один субъект контролирует сразу множество идентификаторов и, следовательно, может влиять на сеть посредством дополнительного права голоса. Proof-of-work помогает снизить этот риск, поскольку без наличия вычислительной мощности узлы не могут добавлять блоки в структуру данных блокчейна.

Еще одной угрозой отсутствия контроля над децентрализованной сетью является атака 51%. Эта атака заключается в том, что группа узлов имеет 51% вычислительной мощности в сети. В результате эти узлы будут иметь возможность подтверждать транзакции и создавать новые блоки. Эта атака экономически нецелесообразна, поскольку для ее реализации потребовались бы огромные вычислительные мощности.

Кроме того, смарт-контракты обладают опасными уязвимостями, которыми могут воспользоваться злоумышленники или майнеры для получения прибыли. Реентерабельность функций смарт-контрактов может стать проблемой, поскольку функции этого типа могут быть прерваны во

время выполнения и повторно вызваны без полного выполнения предыдущего вызова.

Еще одной уязвимостью, присутствующей в смарт-контрактах, является зависимость от состояния транзакции [8]. В цепочке вызовов контрактов контракт должен знать только адрес контракта, из которого он был вызван. Однако, переменная состояния `tx.origin` может быть использована контрактом для получения информации о начальном контракте, с которого началась цепочка, что приводит к проблеме конфиденциальности.

Если контракт имеет в своей реализации зависимость от состояния блока, он зависит от переменных, которые определены в заголовке какого-либо блока [8]. Таким образом, майнер блока может подделать блок, изменив определенные значения, чтобы получить выгоду для себя на уязвимом контракте. Эта атака имеет различную степень успеха, поскольку майнер не может гарантировать свой выбор в качестве майнера конкретного блока.

Более того, контракт может иметь зависимость от порядка транзакций, если его функциональность зависит от правильной обработки параллелизма. Следовательно, майнер может повлиять на порядок транзакций в новом блоке, что приведет к неожиданным результатам в состоянии контракта, которые могут быть финансово выгодны для майнера.

4. Блокчейн-трилемма

Поскольку внедрение блокчейна находится на ранней стадии, естественно, оно сталкивается с трудностями. Трилемма блокчейна относится к широко распространенной проблеме, что децентрализованные сети могут обеспечить только два из трех преимуществ в любой момент времени в отношении децентрализации, безопасности и масштабируемости [9].

1. Децентрализация относится к распределению собственности, влияния и стоимости в блокчейне. Этот элемент также имеет идеологическую перспективу возвращения власти сообществу.
2. Безопасность представляет собой то, какой степени блокчейн может защитить себя от внутренних или внешних атак. Децентрализация и безопасность являются соответствующими элементами, поскольку чем больше узлов в сети, тем меньше риск возникновения центральной точки отказа.
3. Масштабируемость считается наиболее важной из трех, поскольку она устанавливает пределы размера сети. Скорость и объем транзакций играют важную роль в масштабируемости.

Хотя трилемма блокчейна представляет значительные трудности для внедрения технологии блокчейна, появляющиеся решения могут решить эту головоломку. Цель состоит в том, чтобы найти эффективный баланс между безопасностью сети, децентрализацией и масштабируемостью.

5. Примеры использования блокчейна

В связи с популярностью криптовалют блокчейн начал получать признание, и стали появляться различные предложения по применению этой технологии. Поскольку эта технология находится на ранней стадии внедрения, все еще часто предлагается большое количество нереалистичных и непрактичных решений. Необходимо прояснить и четко понять возможные преимущества, ограничения и реальное применение блокчейна.

Финансовые приложения составляют наибольшее количество попыток использования технологии блокчейн. С появлением биткоина и других криптовалют финансовый сектор быстро осознал возможности, связанные с внедрением этой технологии, используя преимущества прозрачности блокчейна, повышенной безопасности, повышенной

эффективности, улучшенной отслеживаемости, неизменяемости и снижения затрат. Несмотря на то, что финансовые приложения демонстрируют большой потенциал, все еще существует большое количество вопросов, которые необходимо решить, например, конфиденциальность данных и масштабируемость на финансовых рынках.

Управление цепочками поставок одна из областей, где блокчейн может быть применен с большим успехом. Прозрачность — это огромная проблема, с которой сталкиваются компании при управлении цепочкой поставок. Клиентам очень сложно определить стоимость продукции, поскольку в цепочке поставок отсутствует прозрачность. Кроме того, чрезвычайно сложно проводить расследования цепей поставок в связи с подозрениями в незаконной или нечестной практике. Учитывая неизменность, надежность и гарантии целостности, предоставляемые технологией блокчейн, ее использование может значительно повысить уверенность и удовлетворенность клиентов при покупке товара.

Децентрализация энергетического рынка еще одна область, в которой возможно применение блокчейна. В настоящее время крупные корпорации контролируют все распределение энергии на каждом рынке. Технология блокчейн обладает потенциалом для внедрения инноваций в одноранговую торговлю энергией и децентрализованное производство энергии. Энергия, произведенная отдельным человеком с помощью солнечной батареи, может быть оценена количественно, продана и записана в бухгалтерскую книгу, что позволит ему быть одновременно потребителем и производителем энергии, создавая децентрализованный рынок, где цены не будут определяться крупными корпорациями. Это может значительно повысить эффективность и снизить затраты.

Здравоохранение — это область, где блокчейн может полностью изменить способ хранения и защиты критически важных данных. В современном обществе растет потребность в мгновенном доступе к важной

медицинской информации. Однако, данные пациентов хранятся непоследовательно в различных медицинских учреждениях, что приводит к затруднению доступа к стандартизированным данным пациента. С помощью технологии блокчейн можно создать постоянно обновляемую децентрализованную базу данных, регулирующую доступ к медицинским данным. В каждом медицинском процессе участвует большое количество различных сторон. Это приводит к тому, что идентификация и аутентификация различных участников медицинского процесса занимает много времени. Технология блокчейн может предложить более эффективную и ускоренную регистрацию заинтересованных сторон в каждой медицинской процедуре, что приведет к улучшению и повышению эффективности системы здравоохранения. Блокчейн также обеспечивает безопасность медицинской информации пациента путем шифрования данных, позволяя установить систему условного контроля доступа, при которой только уполномоченные заинтересованные стороны могут просматривать медицинские записи пациента. Каждый доступ к этим данным будет регистрироваться в бухгалтерской книге, предоставляя пациентам больше контроля над их собственной медицинской информацией.

Кроме этих, блокчейн постоянно используется и в других областях. Еще одна из них - проверка документов, о чем и свидетельствует данная работа.

2. Ethereum

Биткойн был признан революционным явлением в области денег и валюты, став первым примером цифрового ресурса, не имеющего внутренней стоимости и централизованного контролера. Однако, быстро возник большой интерес к лежащей в его основе технологии (блокчейн).

Существовало три основных подхода к созданию продвинутых и сложных приложений на основе криптовалюты. Первый заключался в создании новой блокчейн-системы, которая обеспечивала неограниченную свободу в создании набора функций ценой времени, усилий и безопасности разработки. Второй подход заключался в использовании скриптов поверх уже созданного блокчейна Bitcoin, что было легко реализовать, но очень ограничено в возможностях и масштабируемости. Третий и последний подход заключался в создании мета-протокола поверх технологии Bitcoin, который, хотя и был прост в реализации, страдал от недостатков масштабируемости.

Ethereum был создан с намерением стать решением этой проблемы, предлагая альтернативный протокол для создания децентрализованных приложений, с быстрой и доступной разработкой, эффективным взаимодействием между различными приложениями и даже гарантиями безопасности для небольших приложений.

Ethereum использует технологию блокчейн для обеспечения встроенного полноценного языка программирования, завершенного по Тьюрингу, который можно использовать для создания контрактов, которые могут быть использованы любым пользователем для создания любой системы, просто применив логику в нескольких строках кода.

В то время как в Bitcoin транзакции идентифицируют денежные платежи, транзакции Ethereum могут хранить денежную стоимость, код и/или параметры и вызовы функций. Эта транзакция будет подписана закрытым ключом отправителя, гарантируя аутентификацию и

авторизацию для перевода денег, создания контракта или передачи параметров данных, связанных с транзакциями. Эта транзакция должна быть действительной и правильно сформированной, чтобы содержать всю информацию, необходимую для выполнения.

Состояние блокчейна Ethereum состоит из объектов, называемых счетами. Переходы между состояниями — это прямая передача стоимости и информации между этими счетами. Каждый счет содержит поппе, который действует как счетчик для обеспечения криптографической свежести и используется для того, чтобы убедиться, что каждая транзакция может быть обработана только один раз. Он также содержит текущий баланс Эфира на счете, код контракта (если он есть) и хранилище.

Ether - это базовый токен, питающий блокчейн Ethereum, который служит топливом для Ethereum и используется для оплаты транзакций.

В Ethereum существует два типа счетов:

- **Счета, принадлежащие внешним пользователям,** контролируются закрытыми ключами и не имеют кода. Любой человек может отправить сообщение с такого типа счета, создав и подписав транзакцию с помощью своего закрытого ключа.
- **Счета контрактов,** управляются соответствующим кодом контракта, поэтому, если счет получает сообщение, этот код активируется, позволяя выполнять операции чтения и записи во внутреннее хранилище. Он также разрешает учетной записи отправлять другие сообщения или создавать контракты.

Существуют некоторые ключевые различия между счетами, принадлежащими внешнему владельцу, и счетами по контракту.

Счета, принадлежащие внешним пользователям

- Создание счета не стоит ничего
- Может инициировать транзакции

- Транзакции между аккаунтами, принадлежащими внешним владельцам, могут быть только переводами ETH/токенов

Счета контрактов

- Создание контракта требует затрат, поскольку вы используете сетевое хранилище.
- Можно отправлять транзакции только в ответ на получение транзакции
- Транзакции с внешнего счета на счет контракта могут запускать код, который может выполнять множество различных действий, таких как перевод токенов или даже создание нового контракта

Аккаунты Ethereum имеют четыре поля [10]:

1. `nonce` - Счетчик, который указывает на количество транзакций, отправленных со счета. Это гарантирует, что транзакции обрабатываются только один раз. В контрактном счете это число представляет собой количество контрактов, созданных счетом
2. `balance` - Количество `wei`, принадлежащих данному адресу. `Wei` — это деноминация ETH, и на один ETH приходится $1e+18$ `wei`
3. `codeHash` - Этот хэш относится к коду счета на виртуальной машине Ethereum (EVM). В счетах контрактов запрограммированы фрагменты кода, которые могут выполнять различные операции. Этот код EVM выполняется, если счет получает вызов сообщения. Его нельзя изменить, в отличие от других полей счета. Все такие фрагменты кода содержатся в базе данных состояния под соответствующими хэшами для последующего поиска. Это хэш-значение известно как `codeHash`. Для счетов, принадлежащих внешним

пользователям, поле `codeHash` представляет собой хэш пустой строки

4. `storageRoot` - Иногда известен как хэш хранилища. 256-битный хэш корневого узла тройки Merkle Patricia, которая кодирует содержимое хранилища данного аккаунта (отображение между 256-битными целочисленными значениями), закодированное в тройке как отображение 256-битного хэша Кессак 256-битных целочисленных ключей на 256-битные целочисленные значения в RLP-кодировке. Эта тройка кодирует хэш содержимого хранилища данного счета и по умолчанию пуста

Транзакции — это криптографически подписанные инструкции от аккаунтов. Счет инициирует транзакцию для обновления состояния сети Ethereum. Транзакция Ethereum — это действие, инициированное внешним счетом, то есть счетом, управляемым человеком, а не контрактом.

Отправленная транзакция включает следующую информацию [11]:

- `recipient` - адрес получения (если счет принадлежит внешнему владельцу, транзакция переводит стоимость. Если счет контрактный, то транзакция исполняет код контракта)
- `signature` - идентификатор отправителя. Он генерируется, когда закрытый ключ отправителя подписывает транзакцию и подтверждает, что отправитель санкционировал эту транзакцию
- `value` - сумма ETH для перевода от отправителя к получателю (в WEI, деноминация ETH)
- `data` - необязательное поле для включения произвольных данных
- `gasLimit` - максимальное количество единиц Gas, которое может быть потреблено транзакцией. Единицы Gas представляют собой шаги вычислений

- `maxPriorityFeePerGas` - максимальное количество Gas, которое будет включено в качестве чаевых майнеру
- `maxFeePerGas` - максимальное количество Gas, которое можно заплатить за транзакцию (включает в себя `baseFeePerGas` и `maxPriorityFeePerGas`)

Gas — это ссылка на вычисления, необходимые для обработки транзакции майнером. Пользователи платят за эти вычисления. `GasLimit` и `maxPriorityFeePerGas` определяют максимальную плату за транзакцию, выплачиваемую майнеру.

Если бы злоумышленник совершил атаку типа "отказ в обслуживании", ему пришлось бы платить пропорционально за каждый потребленный ресурс, что привело бы к значительному увеличению платы за Gas.

Ethereum реализует среду исполнения на блокчейне, называемую виртуальной машиной Ethereum, которая работает на узлах, участвующих в сети. Код виртуальной машины Ethereum, основанный на стеке язык байткода, записан в каждом контракте, где каждый байт представляет собой операцию, которая должна быть выполнена.

Чтобы проверить блок, каждый узел в сети просматривает транзакции, перечисленные в новом блоке, и запускает код, который был вызван каждой транзакцией в EVM. Затем каждый узел выполняет одни и те же вычисления и сохраняет одни и те же значения. Это позволяет сети достичь консенсуса относительно состояния системы более эффективным способом.

2. АНАЛИЗ И ФИКСАЦИЯ ТРЕБОВАНИЙ

1. Функциональные требования

В системе выделяются следующие роли:

- **Администратор** - роль, которая позволяет создателем контракта на управление
- **Контролер** - роль, которая позволяет контролировать создателей документов в рамках одного контракта на проверку документов
- **Создатель документа** - роль, которая позволяет поместить документ на проверку
- **Подписант документа** - роль, которая позволяет подписывать определенный документ
- **Пользователь** - роль, которая не имеет конкретного значения в системе (любой пользователь)

Диаграмма вариантов использования роли Администратора показана на следующем рисунке.

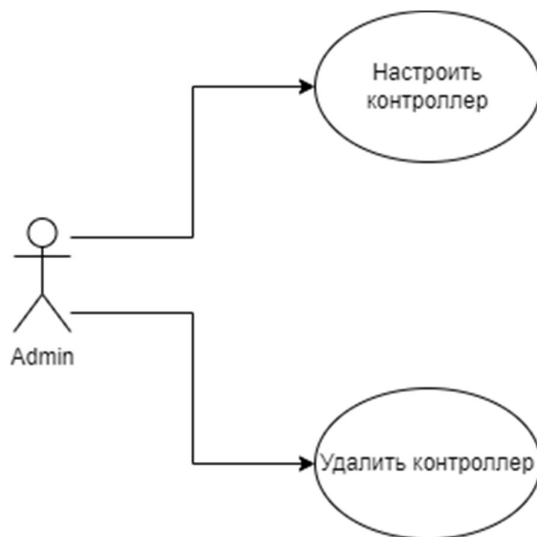


Рисунок 1 — Диаграмма ВИ Администратор

Диаграмма вариантов использования роли Контролера показана на следующем рисунке.

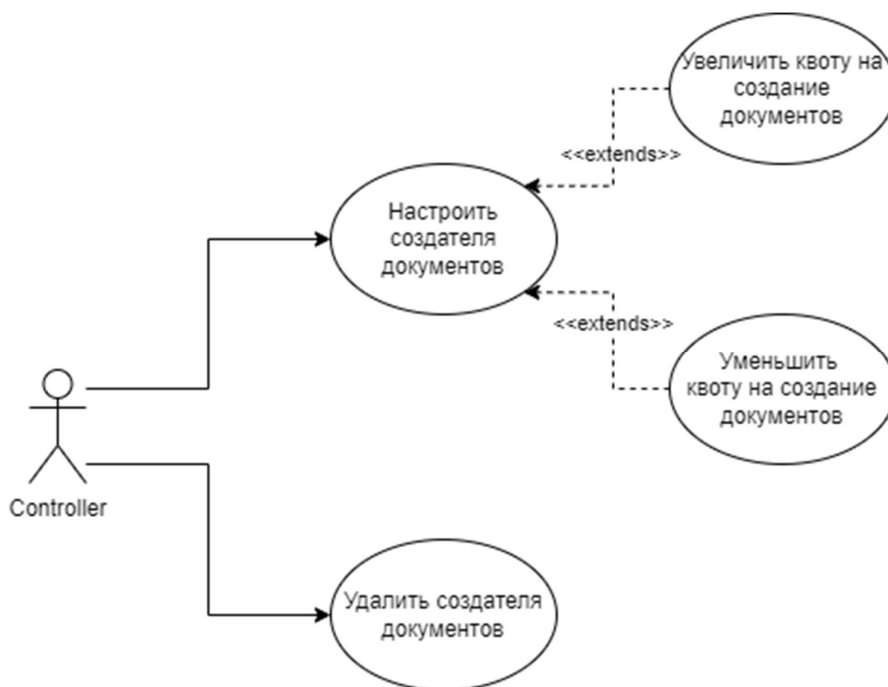


Рисунок 2 — Диаграмма ВИ Контролер

Диаграмма вариантов использования роли Создателя документа показана на следующем рисунке.

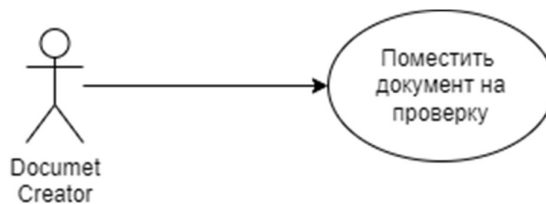


Рисунок 3 — Диаграмма ВИ Создатель документа

Диаграмма вариантов использования роли Подписанта документа показана на следующем рисунке.

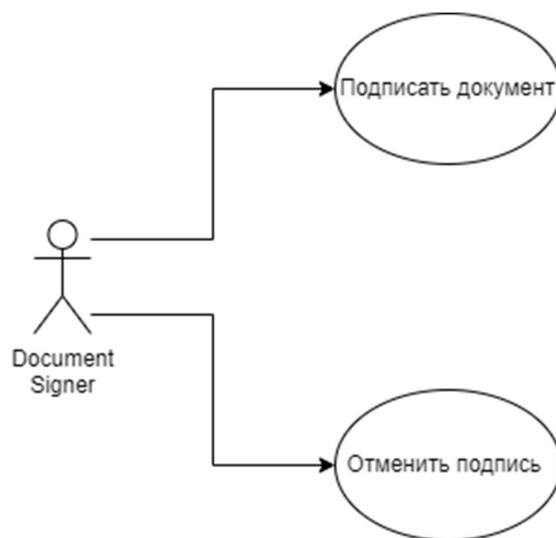


Рисунок 4 — Диаграмма ВИ Подписант документа

Диаграмма вариантов использования роли Пользователя показана на следующем рисунке.

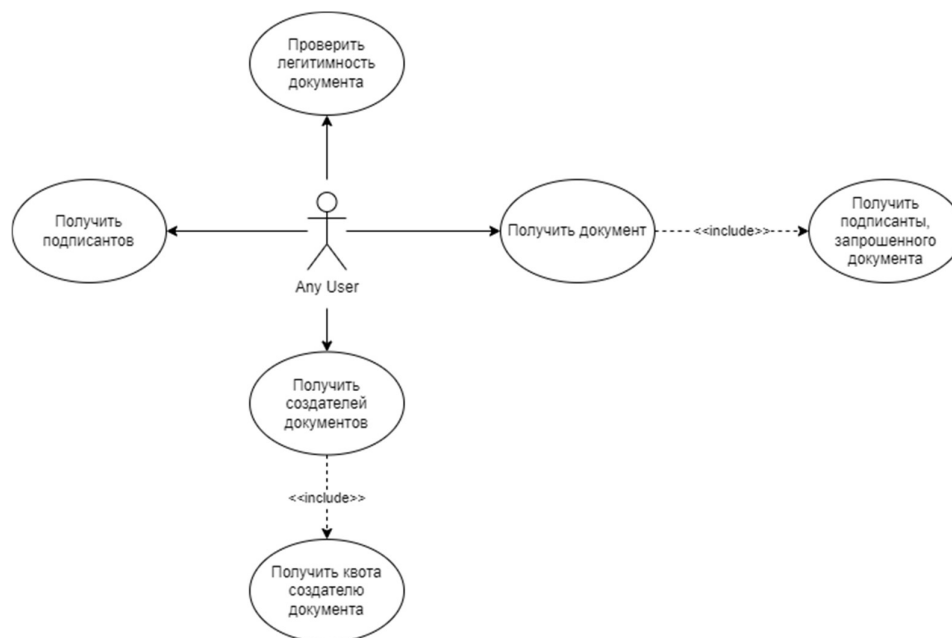


Рисунок 5 — Диаграмма ВИ Пользователь

Любой адрес может иметь более одной роли. Однако это не рекомендуется. Потому что это накладывает слишком большую ответственность на один адрес и, следовательно, создает риск безопасности.

Кроме того, поскольку роли присутствуют в адресах, любая роль может быть как человеком, так и смарт-контрактом. Использование смарт-контракта приводит к еще большей децентрализации проекта. Например, контроллер может быть настроен с использованием голосования или любой другой логики.

2. Нефункциональные требования

- Контракт должен быть развертываемым на любом блокчейне, совместимом с EVM
- Solidity как основной язык для контрактов
- TypeScript как основной язык для тестирования и взаимодействия с контрактами
- Покрытие тестов должно быть 100%

Solidity является наиболее логичным выбором для написания смарт-контрактов, которые будут работать в блокчейн на основе EVM. Потому что сообщество окружает Solidity, и среда разработки данной работы построена с учетом Solidity.

TypeScript также тесно взаимодействует со средой разработки данной работы. Многие библиотеки также используют TypeScript.

Поскольку безопасность является одним из приоритетов при разработке данного проекта, качественное тестирование и полное покрытие тестами - обязательное условие.

3. Сценарии вариантов использования

Таблица 3 — Сценарий варианта использования “Настроить контроллер”

Название ВИ	Настроить контроллер
Цель	Привлечение контролера к контракту на проверку документов
Актер	Администратор
Предусловия	Среда для взаимодействия с контрактом, адрес с достаточным количеством Gas
Сценарий	1. Пользователь вводит адрес контроллера 2. Пользователь вводит адрес контракта на проверку документов 3. Пользователь подтверждает транзакцию 4. Система отправляет транзакцию в блокчейн 5. Блокчейн сохраняет адрес контракта на проверку документов для данного контроллера 6. Блокчейн испускает событие "ControllerConfigured" 7. Система возвращает успешное получение транзакции
Альтернативы	3.1. Пользователь отклоняет транзакцию 3.1.1. Система показывает, что транзакция отклонена

Таблица 4 — Сценарий варианта использования “Удалить контроллер”

Название ВИ	Удалить контроллер
Цель	Удаление контролера из контракта на проверку документов
Актер	Администратор

Таблица 4 — (Продолжение)

Предусловия	Среда для взаимодействия с контрактом, адрес с достаточным количеством Gas
Сценарий	1. Пользователь вводит адрес контроллера 2. Пользователь подтверждает транзакцию 3. Система отправляет транзакцию в блокчейн 4. Блокчейн признает недействительным (присваивает адрес 0) адрес контракта на проверку документов для данного контроллера 5. Блокчейн испускает событие "ControllerRemoved" 6. Система возвращает успешное получение транзакции
Альтернативы	2.1. Пользователь отклоняет транзакцию 2.1.1. Система показывает, что транзакция отклонена

Таблица 5 — Сценарий варианта использования "Настроить создателя документов"

Название ВИ	Настроить создателя документов
Цель	Привлечение создателя документа к контракту на проверку документов
Актер	Контролер
Предусловия	Среда для взаимодействия с контрактом, адрес с достаточным количеством Gas

Таблица 5 — (Продолжение)

Сценарий	1. Пользователь вводит адрес создателя документа 2. Пользователь вводит квоту создателя документа 3. Пользователь подтверждает транзакцию 4. Система отправляет транзакцию в блокчейн 5. Блокчейн сохраняет адрес создателя документа и предоставленную ему квоту 6. Блокчейн испускает событие "DocumentCreatorConfigured" 7. Система возвращает успешное получение транзакции
Альтернативы	3.1. Пользователь отклоняет транзакцию 3.1.1. Система показывает, что транзакция отклонена 5.1. Блокчейн отменяет транзакцию с ошибкой "CallerIsNotManagement" 5.1.1. Система показывает, что транзакция отменена с ошибкой

Таблица 6 — Сценарий варианта использования "Увеличить квоту на создание документов"

Название ВИ	Увеличить квоту на создание документов
Цель	Увеличение квоты создателей документов для создания документов
Актер	Контролер
Предусловия	Среда для взаимодействия с контрактом, адрес с достаточным количеством Gas

Таблица 6 — (Продолжение)

Сценарий	1. Пользователь вводит адрес создателя документа 2. Пользователь вводит сумму приращения 3. Пользователь подтверждает транзакцию 4. Система отправляет транзакцию в блокчейн 5. Блокчейн сохраняет увеличенную квоту для создателя документа 6. Блокчейн испускает событие "DocumentCreatorConfigured" 7. Система возвращает успешное получение транзакции
Альтернативы	3.1. Пользователь отклоняет транзакцию 3.1.1. Система показывает, что транзакция отклонена 5.1. Блокчейн отменяет транзакцию с ошибкой "DocumentCreatorNotFound" 5.1.1. Система показывает, что транзакция отменена с ошибкой 5.2. Блокчейн отменяет транзакцию с ошибкой "CallerIsNotManagement" 5.2.1. Система показывает, что транзакция отменена с ошибкой

Таблица 7 — Сценарий варианта использования "Уменьшить квоту на создание документов"

Название ВИ	Уменьшить квоту на создание документов
Цель	Уменьшение квоты создателей документов для создания документов

Таблица 7 — (Продолжение)

Актер	Контролер
Предусловия	Среда для взаимодействия с контрактом, адрес с достаточным количеством Gas
Сценарий	1. Пользователь вводит адрес создателя документа 2. Пользователь вводит сумму уменьшения 3. Пользователь подтверждает транзакцию 4. Система отправляет транзакцию в блокчейн 5. Блокчейн сохраняет уменьшенную квоту для создателя документа 6. Блокчейн испускает событие "DocumentCreatorConfigured" 7. Система возвращает успешное получение транзакции
Альтернативы	3.1.Пользователь отклоняет транзакцию 3.1.1. Система показывает, что транзакция отклонена 5.1.Блокчейн отменяет транзакцию с ошибкой "DocumentCreatorNotFound" 5.1.1. Система показывает, что транзакция отменена с ошибкой 5.2.Блокчейн отменяет транзакцию с ошибкой "DecrementAmountExceedsAllowance" 5.2.1. Система показывает, что транзакция отменена с ошибкой 5.3.Блокчейн отменяет транзакцию с ошибкой "CallerIsNotManagement"

	5.3.1. Система показывает, что транзакция отменена с ошибкой
--	--

Таблица 8 — Сценарий варианта использования "Удалить создателя документов"

Название ВИ	Удалить создателя документов
Цель	Удаление создателя документа из контракта на проверку документов
Актер	Контролер
Предусловия	Среда для взаимодействия с контрактом, адрес с достаточным количеством Gas
Сценарий	1. Пользователь вводит адрес создателя документа 2. Пользователь подтверждает транзакцию 3. Система отправляет транзакцию в блокчейн 4. Блокчейн удаляет адрес создателя документа и устанавливает его квоту на 0 5. Блокчейн испускает событие "DocumentCreatorRemoved" 6. Система возвращает успешное получение транзакции
Альтернативы	2.1. Пользователь отклоняет транзакцию 2.1.1. Система показывает, что транзакция отклонена 4.1. Блокчейн отменяет транзакцию с ошибкой "CallerIsNotManagement" 4.1.1. Система показывает, что транзакция отменена с ошибкой

Таблица 9 — Сценарий варианта использования "Поместить документ на проверку"

Название ВИ	Поместить документ на проверку
Цель	Генерация хэша документа и помещение его в блокчейн для проверки
Актер	Создатель документа
Предусловия	Среда для взаимодействия с контрактом, адрес с достаточным количеством Gas
Сценарий	1. Пользователь получает хэш документа, используя любой алгоритм хэширования, который возвращает 32-байтовую хэш-строку 2. Пользователь вводит хэш документа 3. Пользователь вводит крайний срок для процесса проверки 4. Пользователь вводит крайний срок для документа (чтобы документ считался легитимным) 5. Пользователь вводит тип проверки для документа (тип алгоритма, который проверяет легитимность документов) 6. Пользователь вводит запрашиваемые лица, подписавшие документ (список адресов, которым разрешено подписывать документ) 7. Пользователь подтверждает транзакцию 8. Система отправляет транзакцию в блокчейн 9. Блокчейн сохраняет реквизиты документа с заданными входными данными

	10.Блокчейн снижает квоту для создателей документов 11.Блокчейн испускает событие "DocumentPutOnVerification" 12.Система возвращает успешное получение транзакции
--	--

Таблица 9 — (Продолжение)

Альтернативы	7.1.Пользователь отклоняет транзакцию 7.1.1. Система показывает, что транзакция отклонена 9.1.Блокчейн отменяет транзакцию с ошибкой "DocumentIsAlreadyOnVerification" 9.1.1. Система показывает, что транзакция отменена с ошибкой 9.2.Блокчейн отменяет транзакцию с ошибкой "RequestedSignersAreNotEnough" 9.2.1. Система показывает, что транзакция отменена с ошибкой 9.3.Блокчейн отменяет транзакцию с ошибкой "DocumentCreatorAllowanceNotEnough" 9.3.1. Система показывает, что транзакция отменена с ошибкой
--------------	--

Таблица 10 — Сценарий варианта использования "Подписать документ"

Название ВИ	Подписать документ
Цель	Подписание документа
Актер	Подписант документа

Таблица 10 — (Продолжение)

Предусловия	Среда для взаимодействия с контрактом, адрес с достаточным количеством Gas
Сценарий	1. Пользователь вводит хэш документа 2. Пользователь подтверждает транзакцию 3. Система отправляет транзакцию в блокчейн 4. Блокчейн сохраняет адрес подписанта документа и временную метку для данного хэша документа 5. Блокчейн испускает событие "DocumentSigned" 6. Система возвращает успешное получение транзакции
Альтернативы	2.1. Пользователь отклоняет транзакцию 2.1.1. Система показывает, что транзакция отклонена 4.1. Блокчейн отменяет транзакцию с ошибкой "InvalidDocument" 4.1.1. Система показывает, что транзакция отменена с ошибкой 4.2. Блокчейн отменяет транзакцию с ошибкой "LateToExecute" 4.2.1. Система показывает, что транзакция отменена с ошибкой 4.3. Блокчейн отменяет транзакцию с ошибкой "SignerIsNotRequested" 4.3.1. Система показывает, что транзакция отменена с ошибкой

	4.4.Блокчейн отменяет транзакцию с ошибкой "SignerAlreadySigned" 4.4.1. Система показывает, что транзакция отменена с ошибкой
--	---

Таблица 11 — Сценарий варианта использования "Отменить подпись"

Название ВИ	Отменить подпись
Цель	Отмена подписи в хэше документа
Актер	Подписант документа
Предусловия	Среда для взаимодействия с контрактом, адрес с достаточным количеством Gas
Сценарий	1. Пользователь вводит хэш документа 2. Пользователь подтверждает транзакцию 3. Система отправляет транзакцию в блокчейн 4. Блокчейн удаляет адрес подписанта документа для данного хэша документа 5. Блокчейн испускает событие "DocumentSignRevoked" 6. Система возвращает успешное получение транзакции
Альтернативы	2.1.Пользователь отклоняет транзакцию 2.1.1. Система показывает, что транзакция отклонена 4.1.Блокчейн отменяет транзакцию с ошибкой "InvalidDocument" 4.1.1. Система показывает, что транзакция отменена с ошибкой

	4.2.Блокчейн отменяет транзакцию с ошибкой "LateToExecute" 4.2.1. Система показывает, что транзакция отменена с ошибкой 4.3.Блокчейн отменяет транзакцию с ошибкой "SignerDidNotSigned" 4.3.1. Система показывает, что транзакция отменена с ошибкой
--	--

Таблица 12 — Сценарий варианта использования "Проверить легитимность документа"

Название ВИ	Проверить легитимность документа
Цель	Проверка легитимности документа по заданному хэшу документа
Актер	Пользователь
Предусловия	Среда для взаимодействия с контрактом
Сценарий	1. Пользователь вводит хэш документа 2. Пользователь вызывает функцию, используя среду для взаимодействия с блокчейном 3. Система вызывает функцию из блокчейна 4. Блокчейн проверяет легитимность документа 5. Блокчейн возвращает результат проверки 6. Система показывает результат проверки
Альтернативы	

Таблица 13 — Сценарий варианта использования "Получить создателей документов"

Название ВИ	Получить создателей документов
Цель	Получение адресов создателей документов и их квот

Таблица 13 — (Продолжение)

Актер	Пользователь
Предусловия	Среда для взаимодействия с контрактом
Сценарий	1. Пользователь вводит адрес 2. Пользователь вызывает функцию, используя среду для взаимодействия с блокчейном 3. Система вызывает функцию из блокчейна 4. Блокчейн проверяет, является ли адрес создателем документа и его квоту 5. Блокчейн возвращает результат проверки 6. Система показывает результат проверки
Альтернативы	

Таблица 14 — Сценарий варианта использования "Получить подписантов"

Название ВИ	Получить подписантов
Цель	Получение информации о подписантах определенного документа
Актер	Пользователь
Предусловия	Среда для взаимодействия с контрактом
Сценарий	1. Пользователь вводит хэш документа 2. Пользователь вызывает функцию, используя среду для взаимодействия с блокчейном 3. Система вызывает функцию из блокчейна 4. Блокчейн перебирает все адреса подписантов и временные метки подписей 5. Блокчейн возвращает результат 6. Система показывает результат
Альтернативы	

Таблица 15 — Сценарий варианта использования "Получить документ"

Название ВИ	Получить документ
Цель	Получение информации об определенном документе
Актер	Пользователь
Предусловия	Среда для взаимодействия с контрактом
Сценарий	1. Пользователь вводит хэш документа 2. Пользователь вызывает функцию, используя среду для взаимодействия с блокчейном 3. Система вызывает функцию из блокчейна 4. Блокчейн проверяет всю информацию о документе (тип проверки, срок и т.д.) 5. Блокчейн возвращает результат проверки 6. Система показывает результат проверки
Альтернативы	

3. ПРОЕКТИРОВАНИЕ

Перед реализацией проекта важно определить роль папки в разработке. Этот проект состоит не только из кода смарт-контракта, но и среды для тестирования смарт-контракта и взаимодействия с блокчейном.

Хотя это не все папки в проекте, ниже представлены наиболее важные из них:

- `contracts` - содержит все файлы разработки смарт-контракта
- `docs` - содержит файлы документации только о смарт-контрактах
- `scripts` - содержит скрипты для смарт-контрактов (например, скрипты развертывания)
- `tasks` - содержит задачи для блокчейна (может быть взаимодействие с конкретной функцией смарт-контракта или проверка контракта)
- `test` - содержит тестовые файлы для смарт-контракта

Также для хранения информации об `api` ключах для связи с блокчейном и информации о частном адресе используется `“.env”` (файл среды).

1. Диаграмма классов

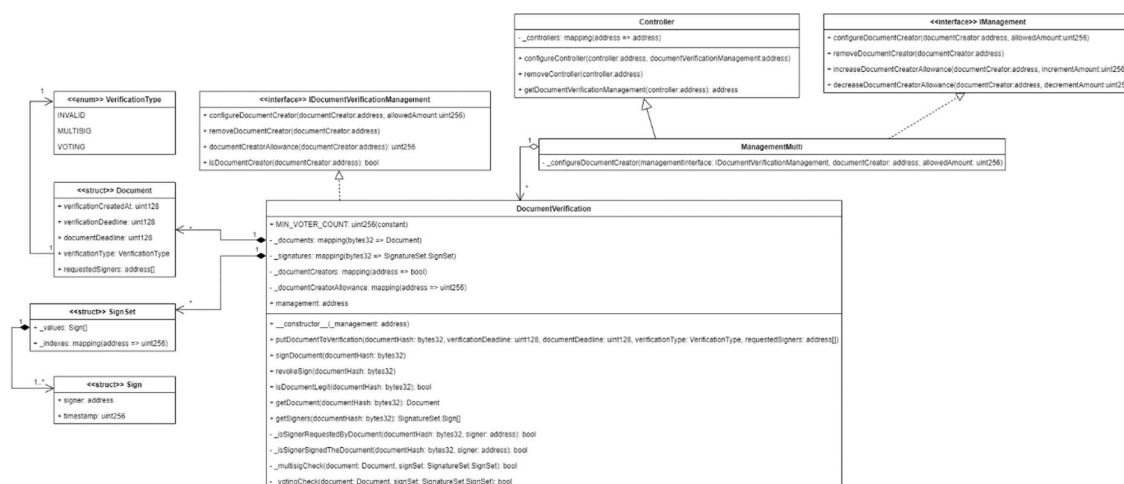


Рисунок 6 — Диаграмма классов

Controller - сохраняет информацию об адресах контроллеров и соответствующих им адресах договоров на проверку документов. Он также содержит функции для управления контроллерами.

ManagementMulti - содержит функции для управления создателями документов. Он может управлять создателями документов для нескольких различных контрактов "DocumentVerification". Он наследует контракт "Controller" и реализует интерфейс "IManagement".

Sign - сохраняет информацию о подписи, например, адрес подписанта и метку времени.

SignSet - хранит структуры "Sign" и их индексы для более быстрого доступа к данным. Она должна хранить по крайней мере одну структуру "Sign".

Document - сохраняет информацию о реквизитах документа. Например, срок исполнения, подписи, которые должны утвердить легитимность документа и т.д. Он также содержит перечисление "VerificationType", которое определяет тип проверки документа.

DocumentVerification - контракт, лежащий в основе основной логики проекта. Он содержит структуры "Document" и "SignSet" для каждого хэша

документа. В нем также хранятся создатели документов и их квоты. Он управляется одним контрактом "ManagementMulti". Контракт "ManagementMulti" управляет контрактом "DocumentVerification" с интерфейсом "IDocumentVerificationManagement", поэтому он реализует этот интерфейс. Он также содержит функции, связанные с документами, такие как, передача документа на проверку, подписание документа, проверка легитимности документов и т.д.

4. РЕАЛИЗАЦИЯ

1. Язык смарт-контрактов

Смарт-контракты для проекта написаны на языке Solidity. Solidity — это объектно-ориентированный, статически типизированный язык программирования, предназначенный для разработки смарт-контрактов, которые работают на виртуальной машине Ethereum (EVM) [12].

Следующие типы являются типами значений в Solidity:

- `boolean`
- `integer`
- `uint` (unsigned integer)
- `address`
- `bytes` (байтовые массивы фиксированного размера, от `bytes1` до `bytes32`)
- `enum`

Большинство из этих типов значений могут быть знакомы по другим языкам программирования. Однако, когда дело доходит до "address", ситуация становится более интересной. Как уже объяснялось ранее, существует два типа адресов ЕОА и контрактные адреса. Для обоих типов адресов адрес имеет формат 42-символьного шестнадцатеричного адреса. Адрес формируется из первых 20 байт открытого ключа, контролирующего аккаунт (если это ЕОА) или развертывающего контракт (если это смарт-контракт). Таким образом, каждый адрес занимает 20 байт памяти. Адрес может быть объявлен следующим образом:

Листинг 1 — Декларация адреса

```
address public immutable management;
```

Примечание: "public" указывает на видимость типа значения, "immutable" указывает на то, что значение переменной не может быть изменено после присвоения ей значения.

Каждый раз, когда в коде solidity создается переменная, evm сохраняет ее в слоте памяти размером 32 байта (256 бит). Поэтому при работе со Struct (пользовательскими объектами данных) следует учитывать упаковку данных. Пример, следующий:

Листинг 2 — Демонстрация структуры документа

```
struct Document {  
    uint128 verificationCreatedAt;  
    uint128 verificationDeadline;  
}
```

Поскольку uint128 — это 16 байт, соответственно, два объекта uint128 в сумме занимают 32 байта, то есть 1 слот для хранения. Это приводит к экономии газа при отправке транзакции. Поскольку хранение данных является одним из наиболее газопотребляющих операторов для EVM, то при написании контракта разработчику следует по возможности избегать хранения данных.

Чтобы сообщить клиентскому приложению или внешнему веб-сайту о том, что что-то произошло в блокчейне, используется "event".

Чтобы объявить новое событие необходимо:

Листинг 3 — Объявление нового события

```
event DocumentPutOnVerification(bytes32 indexed documentHash, address  
indexed documentCreator);
```

А затем испустить событие:

Листинг 4 — Событие испускания

```
function putDocumentToVerification(// Function args) external  
onlyDocumentCreator {  
    // Function code  
  
    emit DocumentPutOnVerification(documentHash, msg.sender);  
}
```

В данном примере событие "DocumentPutOnVerification" возникает, когда создатель документа ставит документ на проверку с помощью функции "putDocumentToVerification".

Когда некоторые условия не выполняются, транзакции должны быть возвращены. В Solidity транзакции могут быть отменены с помощью пользовательских ошибок, используя синтаксис "revert":

Листинг 5 — Пример revert

```
if (msg.sender != management) revert CallerIsNotManagement();
```

В этом примере, если вызывающая функция (msg.sender) не является менеджментом (адрес контракта менеджмента), то транзакция будет возвращена с ошибкой "CallerIsNotManagement".

Перед выполнением функции может потребоваться проверка некоторых условий. Именно тогда модификаторы функций становятся полезными в Solidity. Модификаторы функций используются для изменения или ограничения поведения функции в смарт-контракте.

Чтобы объявить новый модификатор, необходимо произвести следующие действия:

Листинг 6 — Пример modifier

```
modifier onlyManagement() {  
    if (msg.sender != management) revert CallerIsNotManagement();  
    -;  
}
```

А затем использовать его:

Листинг 7 — Использование модификатора

```
function removeDocumentCreator(// Function args) external override  
onlyManagement { // Function code}
```

В этом примере проверяется условие, описанное в Листинг 5 — Пример revert. В случае, если условие выполнено, то выполняется функция "removeDocumentCreator".

2. Настройка контроллера и создателя документа

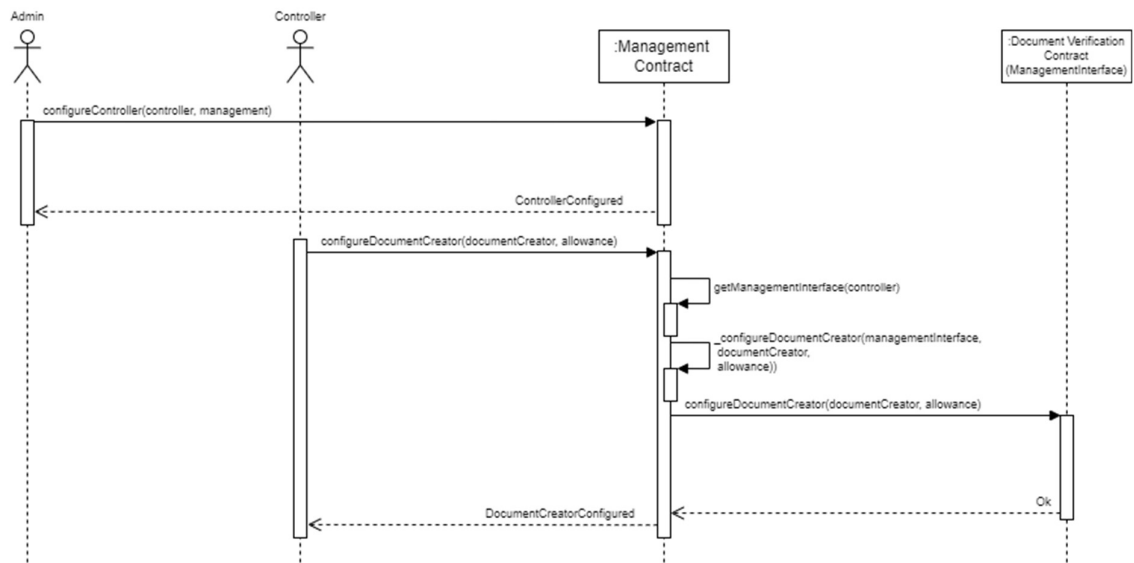


Рисунок 7 — Диаграмма последовательности для ВИ “Настроить контроллер” и “Настроить документ создатель”

На диаграмме первая половина диаграммы, содержащая вызов функций между администратором и контрактом управления, предназначена для конфигурирования контроллера.

Функция для конфигурирования контроллера, написанная на контракте "Controller" (который наследуется контрактом "ManagementMulti"). Сама функция выглядит следующим образом:

Листинг 8 – Функция configureController

```

mapping(address => address) private _controllers;

function configureController(address controller, address
documentVerificationManagement) external onlyOwner {
    _controllers[controller] = documentVerificationManagement;

    emit ControllerConfigured(controller,
documentVerificationManagement);
}
  
```

Адреса управления проверкой документов хранятся для каждого контроллера в отображении, называемом "_controllers". Функция "configureController" вызывается с параметрами:

- controller - адрес контроллера

- documentVerificationManagement - адрес documentVerificationManagement (который является интерфейсом для контракта DocumentVerification)

В самой функции сначала адрес "documentVerificationManagement" связывается с адресом "controller", затем выдается событие "ControllerConfigured".

Вторая половина диаграммы, содержащая вызов функций между контроллером и контрактом проверки документов, предназначена для настройки создателя документа.

Функция для настройки создателя документов, написанная на контракте "ManagementMulti". Сама функция выглядит следующим образом:

Листинг 9 — Функция configureDocumentCreator(ManagementMulti)

```
function configureDocumentCreator(address documentCreator, uint256
allowedAmount) external override onlyController {
    address managementAddress =
getDocumentVerificationManagement(msg.sender);
    IDocumentVerificationManagement managementInterface =
IDocumentVerificationManagement(managementAddress);

    _configureDocumentCreator(managementInterface, documentCreator,
allowedAmount);
}

function _configureDocumentCreator(
    IDocumentVerificationManagement managementInterface,
    address documentCreator,
    uint256 allowedAmount
) private {
    managementInterface.configureDocumentCreator(documentCreator,
allowedAmount);

    emit DocumentCreatorConfigured(documentCreator, allowedAmount);
}
```

Функция "configureDocumentCreator" вызывается с параметрами:

- documentCreator - адрес создателя документа

- `allowedAmount` - количество (квота) документов, которые создатель документа может создать

В самой функции сначала запрашивается адрес управления с помощью адреса `controller(msg.sender)`, затем вызывается приватная функция `"_configureDocumentCreator"`. В этой функции `"configureDocumentCreator"` вызывается с использованием интерфейса управления (который является контрактом создателя документа), затем испускается событие `"DocumentCreatorConfigured"`.

В самом контракте `"DocumentVerification"` параметры сохраняются следующим образом:

Листинг 10 — Функция

`configureDocumentCreator(DocumentVerification)`

```
mapping(address => bool) private _documentCreators;
mapping(address => uint256) private _documentCreatorAllowance;

function configureDocumentCreator(address documentCreator, uint256
allowedAmount) external override onlyManagement {
    _documentCreators[documentCreator] = true;
    _documentCreatorAllowance[documentCreator] = allowedAmount;
}
```

Адреса создателей документов хранятся в маппинге под названием `"_documentCreators"`, а их квоты хранятся в маппинге под названием `"_documentCreatorAllowance"`. В самой функции сначала сохраняется создатель документа, а затем его разрешенное количество (квота).

После того как создатели документов настроены, документы готовы к помещению в блокчейн.

3. Поместить документ на проверку

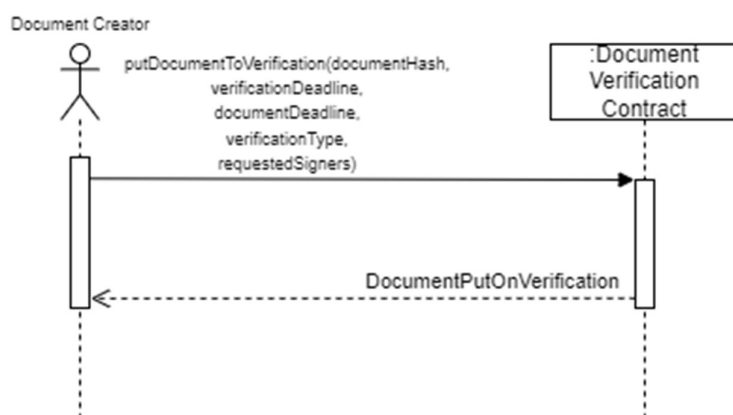


Рисунок 8 — Диаграмма последовательности для ВИ “Поместить документ на проверку”

Это решающий шаг для данного проекта. Но прежде, чем поместить документ на проверку, сначала необходимо сгенерировать хэш документа. Информация об алгоритмах хэширования выходит за рамки данной диссертации. Но любой алгоритм хэширования, который возвращает 32-байтовую хэш-строку, подходит (например, sha256, keccak256 и т.д.).

Функция для помещения документа на проверку написана на контракте "DocumentVerification". Сама функция выглядит следующим образом:

Листинг 11 — Функция putDocumentToVerification

```
mapping(bytes32 => Document) private _documents;

function putDocumentToVerification(
    bytes32 documentHash,
    uint128 verificationDeadline,
    uint128 documentDeadline,
    VerificationType verificationType,
    address[] calldata requestedSigners
) external onlyDocumentCreator {
    if (_documents[documentHash].verificationCreatedAt != 0) revert
DocumentIsAlreadyOnVerification();
    if (requestedSigners.length < MIN_VOTER_COUNT)
        revert RequestedSignersAreNotEnough({
            sentLength: requestedSigners.length,
            requiredLength: MIN_VOTER_COUNT
        });
    if (_documentCreatorAllowance[msg.sender] == 0) revert
DocumentCreatorAllowanceNotEnough();

    Document storage document = _documents[documentHash];

    document.verificationCreatedAt = uint128(block.timestamp);
    document.verificationDeadline = verificationDeadline;
    document.documentDeadline = documentDeadline;
    document.verificationType = verificationType;
    document.requestedSigners = requestedSigners;

    _documentCreatorAllowance[msg.sender]--;

    emit DocumentPutOnVerification(documentHash, msg.sender);
}
```

Документы хранятся для каждого хэша документа в маппинге, называемом "_documents". Функция "putDocumentToVerification" вызывается с параметрами:

- documentHash - хэш документа, который нужно поместить на проверку
- verificationDeadline - крайний срок подписного периода (в секундах)
- documentDeadline - крайний срок действия документа (в секундах)
- verificationType - тип алгоритма проверки для документа

- requestedSigners - список адресов, которым разрешено подписывать данный документ

В самой функции сначала выполняются некоторые проверки, например, проверяется, не находится ли документ уже на проверке, проверяется, достаточно ли запрашиваемых подписчиков (по умолчанию минимум 1), и, наконец, проверяется, достаточно ли квоты у создателя документа. После этих проверок объект документа сохраняется с заданными параметрами. Затем квота создателя документа уменьшается на единицу. И, наконец, выдается событие "DocumentPutOnVerification".

4. Подписать документ

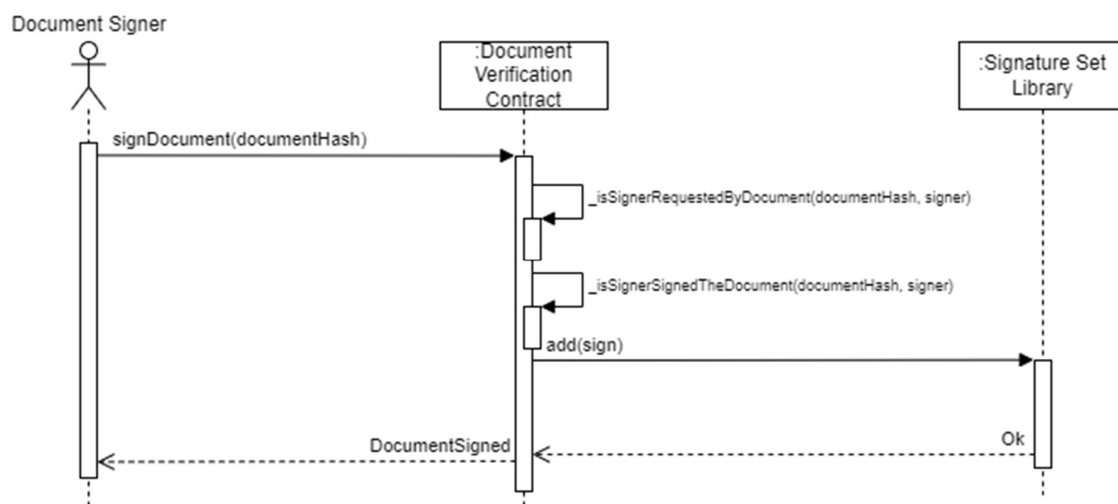


Рисунок 9 — Диаграмма последовательности для ВИ “Подписать документ”

При помещении документа на проверку лицо, подписывающее документ, обращается к определенному документу. Подписывающему документ также необходим хэш документа при подписании документа, а хэш документа может быть сгенерирован тем же способом, как это объясняется в заголовке "Поместить документ на проверку".

Функция для подписания документа, написанного на контракте "DocumentVerification". Сама функция выглядит следующим образом:

Листинг 12 — Функция signDocument

```
mapping(bytes32 => SignatureSet.SignSet) private _signatures;

function signDocument(bytes32 documentHash) external
validDocument(documentHash) {
    Document memory document = _documents[documentHash];
    if (block.timestamp > document.verificationDeadline)
        revert LateToExecute({ executeTime:
document.verificationDeadline });
    if (!_isSignerRequestedByDocument(documentHash, msg.sender)) revert
SignerIsNotRequested();
    if (_isSignerSignedTheDocument(documentHash, msg.sender)) revert
SignerAlreadySigned();

    // add signature to document
    SignatureSet.Sign memory sign = SignatureSet.Sign({ signer:
msg.sender, timestamp: block.timestamp });
    _signatures[documentHash].add(sign);

    emit DocumentSigned(documentHash, msg.sender);
}
```

Подписи хранятся для каждого хэша документа в связке, называемой "_signatures". Функция "signDocument" вызывается с параметрами:

- documentHash - хэш документа для подписи

В самой функции сначала выполняются некоторые проверки, такие как: проверка того, не прошел ли срок проверки, проверка того, запрашивается ли документ подписантом, проверка того, подписан документ. После этих проверок подпись сохраняется с текущей меткой времени и адресом подписанта. И, наконец, выдается событие "DocumentSigned".

5. Проверить легитимность документа

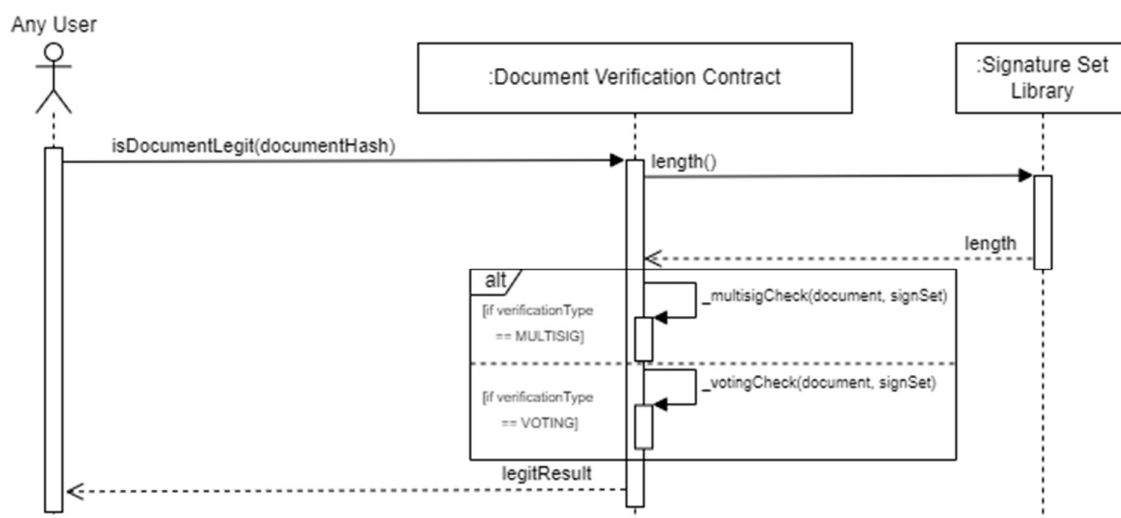


Рисунок 10 — Диаграмма последовательности для ВИ “Проверить легитимность документа”

После выполнения всех шагов от настройки контроллеров до подписания документа, остается одна часть для завершения проекта - проверка легитимности документов.

В отличие от других шагов, проверка легитимности документов не требует транзакции в блокчейн, поэтому не требуется адрес. Эта функциональность общедоступна для всех. Причина этого в том, что при чтении документа система не производит никаких изменений в хранении блокчейна, она только считывает данные из него и производит некоторые проверки.

Перед проверкой легитимности документов необходимо сначала сгенерировать хэш документа. Хэш документа должен быть сгенерирован с помощью точно такого же алгоритма хэширования, когда документ будет помещен на проверку. Хэш документа также может храниться в блокчейне для поддержания согласованности. Однако при создании проекта этот путь не был выбран по следующим причинам: различные хэши документов могут храниться в блокчейне с использованием различных алгоритмов

хэширования, а некоторые документы могут быть конфиденциальными, поэтому их легитимность должна проверяться только частными организациями или людьми. Поэтому алгоритм хэширования документов не хранится в блокчейне.

Функция проверки легитимности документов, написанная на контракте "DocumentVerification". Сама функция выглядит следующим образом:

Листинг 13 — Функция isDocumentLegit

```
function isDocumentLegit(bytes32 documentHash) external view returns (bool legit) {
    Document memory document = _documents[documentHash];
    SignatureSet.SignSet storage signSet = _signatures[documentHash];

    if (block.timestamp < document.documentDeadline) {
        if (signSet.length() >= MIN_VOTER_COUNT) {
            if (document.verificationType == VerificationType.MULTISIG)
            {
                legit = _multisigCheck(document, signSet);
            }
            if (document.verificationType == VerificationType.VOTING) {
                legit = _votingCheck(document, signSet);
            }
        }
    }
}

function _multisigCheck(Document memory document, SignatureSet.SignSet
storage signSet)
private
view
returns (bool legit)
{
    legit = document.requestedSigners.length == signSet.length();
}

function _votingCheck(Document memory document, SignatureSet.SignSet
storage signSet)
private
view
returns (bool legit)
{
    legit = (signSet.length() * 10e18) / document.requestedSigners.length >
5 * 10e17;
}
```

Функция "isDocumentLegit" вызывается с параметрами:

- documentHash - хэш документа для проверки его легитимности

В самой функции, во-первых, проверяется крайний срок документов. Если срок документа истек, то документ считается не легитимным. Затем проверяется количество подписей. Если количество подписей меньше минимального количества избирателей (по умолчанию 1), то документ снова считается не легитимным. Последняя проверка осуществляется по типу проверки документов, существует 2 различных типа проверки:

- multisig (мультиподпись)
- voting (голосование)

Функция “_multisigCheck” проверяет только, совпадает ли общее количество подписей с общим количеством запрошенных подписчиков. То есть, по сути, она проверяет, подписал ли документ каждый запрошенный подписант или нет. Если да, то документ считается легитимным, если нет, то документ считается не легитимным.

Функция "_votingCheck" проверяет, превышает ли общее количество подписей 50% от общего количества запрошенных подписчиков. То есть, по сути, проверяется, согласилось ли большее количество подписантов подписать документ или нет. Если да, то документ считается легитимным, если нет, то документ считается не легитимным.

6. Взаимодействие с блокчейном и развертывание смарт-контрактов

В проекте при взаимодействии со смарт-контрактами и, соответственно, блокчейном используется язык TypeScript. Также для взаимодействия необходимо использовать некоторые библиотеки, например ethers и typechain.

TypeScript — это язык программирования, разработанный и поддерживаемый компанией Microsoft. Он является строгим синтаксическим надмножеством JavaScript и добавляет в язык необязательную статическую типизацию. Он предназначен для разработки больших приложений и транспонируется на JavaScript [13].

ethers — это библиотека для взаимодействия с блокчейнами на основе EVM и их экосистемой.

typechain — это библиотека для генерации кода из смарт-контракта, код генерируется на TypeScript. Это упрощает вызов функций смарт-контракта.

Взаимодействие со смарт-контрактами в основном происходит следующим образом: поскольку код смарт-контракта написан на Solidity, его функции или переменные недоступны языку TypeScript. Библиотека typechain решает эту проблему, создавая типы из смарт-контракта для использования TypeScript. С помощью библиотеки ethers и типов, созданных библиотекой typechain, осуществляется взаимодействие смарт-контрактов.

В следующем фрагменте кода показывается скрипт развертывания:

Листинг 14 — Скрипт развертывания контракта

```
async function main() {  
  const [owner] = await ethers.getSigners();  
  const args = contractArgs();  
  const contract = await getContract(owner, args);  
  console.log("DocumentVerification deployed to: ", contract.address);  
}  
  
async function getContract(owner: SignerWithAddress, args: any[]) {  
  const factory = new DocumentVerification__factory(owner);  
  const contract = await factory.deploy(args[0]);  
  await contract.deployed();  
  
  return contract;  
}
```

Поскольку отправка транзакции также требует сначала подписать ее, необходимы подписывающие лица. К счастью, библиотека ethers также занимается этим, с помощью библиотеки ethers разработчик может получить подписывающих лиц с их адресами. Затем с помощью этого подписанта в качестве владельца разворачивается контракт. Однако, как видно из фрагмента кода, при выполнении этого действия не требуется никаких дополнительных шагов, транзакция подписывается автоматически с помощью библиотеки ethers и типа, созданного библиотекой typechain.

7. Тестирование

В отличие от программ на традиционных языках программирования, которые можно отлаживать, в контрактах Solidity ошибки нельзя редактировать или исправлять; транзакции нельзя отменить. Solidity придерживается мантры "Код — это закон", что означает, что любой смарт-контракт должен быть безупречно закодирован, когда он вступает в силу. Вот почему тестирование действительно очень важно в блокчейне.

Покрытие тестов должно быть 100%, а качество тестов должно быть высоким. Кроме того, тест должен быть читабельным и понятным.

Mocha и Chai - два фреймворка, которые обычно используются вместе для модульного тестирования.

Mocha — это основа для тестирования, которая предоставляет функции, выполняемые в определенном порядке, и записывает их результаты в окно терминала.

Chai — это библиотека утверждений, которая часто используется вместе с Mocha. Она предоставляет функции и методы, которые помогают сравнить результат определенного теста с его ожидаемым значением. Chai предоставляет чистый синтаксис, который читается почти как английский.

В следующем фрагменте кода показан пример теста, написанного с использованием библиотек Mocha и Chai:

Листинг 15 — Пример теста

```
describe("Sign document check", function () {
  it("Should not sign document, if document is invalid", async function
  () {
    const documentHash = ethers.utils.id("DOCUMENT-INVALID");

    await
    expect(this.documentVerification.signDocument(documentHash)).to.revertedWith(
      utils.errorInvalidDocument(),
    );
  });
});
```

Тестовые наборы объединяются под ключевым словом "describe", а тестовые случаи - под ключевым словом "it".

В этом примере кода библиотека Chai утверждает, что транзакция должна быть отменена с ошибкой "errorInvalidDocument".

После завершения тестов можно проверить покрытие тестов с помощью библиотеки "solidity-coverage". На следующем рисунке видно, что смарт-контракты в проекте имеют 100% покрытие кода:

File ▴		Statements ▾	Branches ▾	Functions ▾	Lines ▾				
Controller.sol		100%	6/6	100%	2/2	100%	4/4	100%	7/7
IManagement.sol		100%	0/0	100%	0/0	100%	0/0	100%	0/0
ManagementMulti.sol		100%	22/22	100%	6/6	100%	5/5	100%	22/22
ManagementSingle.sol		100%	15/15	100%	6/6	100%	6/6	100%	15/15

Рисунок 11 — Тестовое покрытие смарт-контрактов

Однако 100% покрытие тестами НЕ гарантирует, что контракт безопасен. Перед развертыванием контрактов в mainnet, контракты должны быть развернуты в testnet и должны быть протестированы таким же образом.

ЗАКЛЮЧЕНИЕ

В данной квалификационной работе представлена работающая и протестированная система проверки документов с использованием блокчейна.

Цели и задачи были полностью выполнены.

- Исследование алгоритма хеширования проведено, хэширование документов получено
- Исследование блокчейна (особенно на основе EVM) и хранения данных в блокчейне проведено, информация о легитимности документов сохранилась на блокчейне
- Разработаны смарт-контракт проверки документов и смарт-контракт управления
- Разработанные смарт-контракты протестированы

СПИСОК ЛИТЕРАТУРЫ

1. Blockchain [Электронный ресурс] // URL:
<https://en.wikipedia.org/wiki/Blockchain>
2. Cryptocurrency [Электронный ресурс] // URL:
<https://en.wikipedia.org/wiki/Cryptocurrency>
3. Block reward [Электронный ресурс] // URL:
<https://academy.binance.com/en/glossary/block-reward>
4. Proof of work [Электронный ресурс] // URL:
https://en.wikipedia.org/wiki/Proof_of_work
5. Proof of work or proof of stake [Электронный ресурс] // URL:
<https://www.coinbase.com/tr/learn/crypto-basics/what-is-proof-of-work-or-proof-of-stake>
6. Smart contracts [Электронный ресурс] // URL:
<https://www.investopedia.com/terms/s/smart-contracts.asp>
7. Permissioned vs permissionless blockchain key differences [Электронный ресурс] // URL: <https://cointelegraph.com/blockchain-for-beginners/permissioned-blockchain-vs-permissionless-blockchain-key-differences>
8. ZEUS: Analyzing Safety of Smart Contracts [Электронный ресурс] // URL: https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018_09-1_Kalra_paper.pdf
9. Blockchain trilemma decentralization scalability definition [Электронный ресурс] // URL:
<https://www.gemini.com/cryptopedia/blockchain-trilemma-decentralization-scalability-definition>
10. Ethereum: accounts [Электронный ресурс] // URL:
<https://ethereum.org/en/developers/docs/accounts/>
11. Ethereum: transactions [Электронный ресурс] // URL:
<https://ethereum.org/en/developers/docs/transactions/>

12. Solidity [Электронный ресурс] // URL:
<https://en.wikipedia.org/wiki/Solidity>
13. TypeScript [Электронный ресурс] // URL:
<https://en.wikipedia.org/wiki/TypeScript>

Отчет о проверке на заимствования №1



Автор: Текин Онур

Проверяющий: (kralonur1998@gmail.com / ID: 9902683)

Отчет предоставлен сервисом «Антиплагиат» - <http://users.antiplagiat.ru>

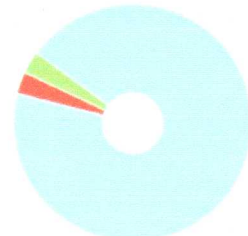
Кисел К.С.
10.06.22

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 3
Начало загрузки: 07.06.2022 21:17:41
Длительность загрузки: 00:00:06
Корректировка от 07.06.2022 21:25:18
Имя исходного файла: Diplom.pdf
Название документа: Разработка системы проверки документов с использованием блокчейна
Размер текста: 1 кБ
Тип документа: Дипломная работа
Символов в тексте: 72282
Слов в тексте: 8034
Число предложений: 400

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Последний готовый отчет (ред.)
Начало проверки: 07.06.2022 21:17:47
Длительность проверки: 00:02:36
Комментарии: не указано
Поиск с учетом редактирования: да
Модули поиска: ИПС Адилет, Библиография, Сводная коллекция ЭБС, Интернет Плюс, Сводная коллекция РГБ, Цитирование, Переводные заимствования (RuEn), Переводные заимствования по eLIBRARY.RU (EnRu), Переводные заимствования по eLIBRARY.RU (KkRu), Переводные заимствования по eLIBRARY.RU (KyRu), Переводные заимствования по Интернету (EnRu), Переводные заимствования по Интернету (KkRu), Переводные заимствования по Интернету (KyRu), Переводные заимствования (KkEn), Переводные заимствования (KyEn), Переводные заимствования издательства Wiley (RuEn), eLIBRARY.RU, СПС ГАРАНТ, Медицина, Диссертации НББ, Перефразирования по eLIBRARY.RU, Перефразирования по Интернету, Перефразирования по коллекции издательства Wiley, Патенты СССР, РФ, СНГ, СМИ России и СНГ, Шаблоны фразы, Кольцо вузов, Издательство Wiley, Переводные заимствования



ЗАИМСТВОВАНИЯ

3,34%

САМОЦИТИРОВАНИЯ

0%

ЦИТИРОВАНИЯ

2,58%

ОРИГИНАЛЬНОСТЬ

94,08%

Заимствования — доля всех найденных текстовых пересечений, за исключением тех, которые система отнесла к цитированиям, по отношению к общему объему документа.
Самоцитирования — доля фрагментов текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника, автором или соавтором которого является автор проверяемого документа, по отношению к общему объему документа.
Цитирования — доля текстовых пересечений, которые не являются авторскими, но система посчитала их использование корректным, по отношению к общему объему документа. Сюда относятся оформленные по ГОСТу цитаты; общеупотребительные выражения; фрагменты текста, найденные в источниках из коллекций нормативно-правовой документации.
Текстовое пересечение — фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.
Источник — документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.
Оригинальность — доля фрагментов текста проверяемого документа, не обнаруженных ни в одном источнике, по которым шла проверка, по отношению к общему объему документа.
Заимствования, самоцитирования, цитирования и оригинальность являются отдельными показателями и в сумме дают 100%, что соответствует всему тексту проверяемого документа. Обращаем Ваше внимание, что система находит текстовые пересечения проверяемого документа с проиндексированными в системе текстовыми источниками. При этом система является вспомогательным инструментом, определение корректности и правомерности заимствований или цитирований, а также авторства текстовых фрагментов проверяемого документа остается в компетенции проверяющего.

№	Доля в отчете	Источник	Актуален на	Модуль поиска	Комментарии
[01]	2,16%	не указано	13 Янв 2022	Библиография	
[02]	1,44%	White Paper. A Next-Generation Smart Contract and Decentralized Application Platform. Table of Contents. ethereum / wiki. Basics - PDF http://docplayer.net	04 Янв 2018	Переводные заимствования (RuEn)	
[03]	0%	Ethereum White Paper A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM - PDF http://docplayer.net	04 Янв 2018	Переводные заимствования (RuEn)	
[04]	0,04%	Скачать - 1,2 МБ http://nauchkor.ru	22 Авг 2018	Интернет Плюс	
[05]	0%	https://dspace.spbu.ru/bitstream/11701/7299/1/Voropaeva_VKR.doc https://dspace.spbu.ru	28 Сен 2019	Интернет Плюс	
[06]	0%	Вербализация концепта hygge в датской языковой картине мира https://nauchkor.ru	24 Окт 2019	Интернет Плюс	
[07]	0,06%	http://www.bgitu.ru/upload/iblock/766/76622a111522113cbbc6cc8a82ae3ac4.pdf http://bgitu.ru	06 Июл 2020	Интернет Плюс	
[08]	0%	http://www.bgitu.ru/upload/iblock/766/76622a111522113cbbc6cc8a82ae3ac4.pdf http://bgitu.ru	21 Мар 2020	Интернет Плюс	
[09]	0%	http://naukaip.ru/wp-content/uploads/2018/04/%D0%9C%D0%9A-320-%D0%A1%D0%B1%D0%BE%D1%80%D0%BD%D0%B8%D0%BA-%D0%A7%D0%B0%D1%81%D1%82%D1%8C-2.pdf (1/2) http://naukaip.ru	20 Сен 2018	Интернет Плюс	
[10]	0%	Скачать - 3,1 МБ https://nauchkor.ru https://biti.mephi.ru/wp-	24 Окт 2019	Интернет Плюс	

[11]	0%	content/uploads/2020/10/%D0%A2%D0%9E%D0%9C-2.pdf https://biti.mephi.ru	13 Апр 2022	Интернет Плюс	
[12]	0%	Еськов, Станислав Сергеевич Специальное математическое и программное обеспечение взаимного информационного согласования в системах распределенного реестра : диссертация ... кандидата технических наук : 05.13.11 Воронеж 2020 http://dlib.rsl.ru	12 Янв 2021	Сводная коллекция РГБ	
[13]	0,08%	2016 год http://narfu.ru	13 Дек 2016	Интернет Плюс	
[14]	0,52%	23541-2_Мурзин_ПИ	06 Июн 2019	Кольцо вузов	
[15]	0,42%	не указано	13 Янв 2022	Шаблонные фразы	
[16]	0%	Введение в глобальную политическую экономию https://e.lanbook.com	22 Янв 2020	Сводная коллекция ЭБС	
[17]	0%	http://www.bgitu.ru/upload/iblock/bad/%D0%A1%D0%B1%D0%BE%D1%80%D0%BD%D0%B8%D0%BA%20%D1%81%D1%82%D0%B0%D1%82%D0%B5%D0%B9%20%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%BE%D0%B9%20%D1%80%D0%B5%D0%B3%D0%B8%D0%BE%D0%BD_2019.pdf http://bgitu.ru	19 Мая 2022	Интернет Плюс	
[18]	0%	Как блокчейн изменит сферу недвижимости https://bitnovosti.com	01 Июн 2022	Интернет Плюс	
[19]	0,32%	Уязвимости смарт-контрактов блокчейн-платформы Ethereum. http://elibrary.ru	10 Фев 2020	ПЕРЕФРАЗИРОВАНИЯ ПО eLIBRARY.RU	
[20]	0%	Исследование очередей доступа потоков к разделяемому ресурсу при использовании условных переменных http://library.eltech.ru	19 Сен 2019	Интернет Плюс	
[21]	0,25%	https://www.tsu.ru/upload/medialibrary/759/rekomenduemyy_shablon_programmy_gia-26.01.2021.docx https://tsu.ru	24 Янв 2022	Интернет Плюс	
[22]	0,07%	Универсальная технология: об альтернативах применения блокчейна Mind.ua https://mind.ua	26 Мая 2022	Интернет Плюс	
[23]	0%	Универсальная технология: об альтернативах применения блокчейна Mind.ua https://mind.ua	27 Апр 2020	Интернет Плюс	
[24]	0,28%	ПЕРСПЕКТИВЫ ПРИМЕНЕНИЯ ТЕХНОЛОГИЙ БЛОКЧЕЙН В ЛОГИСТИКЕ. http://elibrary.ru	20 Авг 2020	eLIBRARY.RU	
[25]	0,22%	Diplom_Liubimov_final	08 Июн 2018	Кольцо вузов	
[26]	0%	23541-3_Круминьш_ДВ	06 Июн 2019	Кольцо вузов	
[27]	0,05%	ВКР АЙБАТОВ Т.И. гр26К, Информационная система на основе технологии блокчейн (1).docx	15 Июн 2020	Кольцо вузов	
[28]	0%	ВКР АЙБАТОВ Т.И. гр26К, Информационная система на основе технологии блокчейн	20 Июн 2020	Кольцо вузов	
[29]	0%	https://lib.tsu.ru/win/produkcija/metodichka/pril1.pdf https://lib.tsu.ru	25 Мая 2022	Интернет Плюс	Источник исключен. Причина: Маленький процент пересечения.
[30]	0%	Вагнер, Дмитрий Викторович Высокочастотные электромагнитные характеристики композиционных радиоматериалов на основе гексагональных ферритов : диссертация ... кандидата технических наук : 01.04.03 Томск 2019 http://dlib.rsl.ru	27 Дек 2019	Сводная коллекция РГБ	Источник исключен. Причина: Маленький процент пересечения.
[31]	0%	Атака Сибиллы - Sybil attack - qaz.wiki https://ru.qaz.wiki	07 Июн 2022	Интернет Плюс	Источник исключен. Причина: Маленький процент пересечения.
[32]	0%	ВКР (Плоткин А.С. Группа 151-331) промежуточная версия	10 Июн 2020	Кольцо вузов	Источник исключен. Причина: Маленький процент пересечения.
[33]	0%	Смарт Контракт (Эфир) - что это такое? Как Работает? https://bytwork.com	05 Июн 2022	Интернет Плюс	Источник исключен. Причина: Маленький процент пересечения.
[34]	0%	Разработка оракула для связи блокчейн сети с внешним сервисом	14 Июн 2019	Кольцо вузов	Источник исключен. Причина: Маленький процент пересечения.
[35]	0%	О технологиях NFT и Блокчейн — NightRunner на DTF https://dtf.ru	01 Июн 2022	Интернет Плюс	Источник исключен. Причина: Маленький процент пересечения.
[36]	0%	vishnyakov_a_a_analiz-trebovaniy-k-informatizacionnoy-sisteme-raspredelenno-reestra.docx	09 Июн 2018	Кольцо вузов	Источник исключен. Причина: Маленький процент пересечения.
[37]	0%	vishnyakov_a_a_algoritmy-konsensusa-v-raspredelennyh-sistemah.docx	20 Мая 2019	Кольцо вузов	Источник исключен. Причина: Маленький процент пересечения.
[38]	0%	АТАКА НА ФОСГЕН	03 Янв 2019	СМИ России и СНГ	Источник исключен. Причина: Маленький процент пересечения.
[39]	0%	Full article: Macro and Exogenous Factors in Computational Advertising: Key Issues and New Research Directions https://tandfonline.com	07 Июн 2022	Интернет Плюс	Источник исключен. Причина: Маленький процент пересечения.
[40]	0%	https://www.rea.ru/ru/org/faculties/mngfak/Documents/%D0%A1%D0%B1%D0%BE%D1%80%D0%BD%D0%B8%D0%BA.pdf https://rea.ru	13 Апр 2022	Интернет Плюс	Источник исключен. Причина: Маленький процент пересечения.
[41]	0%	What are proof of stake and proof of work in cryptocurrency/blockchain? - Quora https://translated.turbopages.org	07 Июн 2022	Интернет Плюс	Источник исключен. Причина: Маленький процент пересечения.