Федеральное государственное автономное образовательное учреждение высшего образования

Национальный исследовательский университет «Высшая школа экономики»

Факультет компьютерных наук

Основная образовательная программа
Прикладная математика и информатика

ГРУППОВАЯ КУРСОВАЯ РАБОТА

Программный проект
на тему
«Реализация NFT маркетплейса на базе Discord API»

Выполнили студенты:
Лущ Иван Сергеевич, группа 195, 3 курс,
Басалаев Максим Александрович, группа 195, 3 курс
Токкожин Арсен Ардакович, группа 194, 3 курс
Кусиденов Адильхан Маратович, группа 195, 3 курс

Руководитель КР:
Внешний руководитель, Рыжиков Никита Ильич

МОСКВА 2022

СОДЕРЖАНИЕ

1 Аннотация

1.1 Аннотация

В настоящее время все чаще популяризируется концепция блокчейна. В связи с этим растет количество разных приложений взаимозависимых с данной концепцией. Один из самых популярных объектов является NFT(non-fungible token, невзаимозаменяемый токен). На этой идее существует большое количество протоколов на разных блокчейнах, которые позволяют обмениваться NFT на торговых площадках. Целью данного командного проекта является реализация discord-бота с функционалом NFT маркетплейса в новом и быстроразвивающемся блокчейне NEAR Protocol и сервисом генерации NFT, используя генеративно-состязательную сеть. Для этого необходимо было реализовать smart-конракт NFT(согласно стандарту NEP-171), smart-контракт маркетплейса, подстроить API для взаимодействия с блокчеином NEAR-protocol под возможности discord и реализовать discord-бота, ...(тут что-то про модель).

1.2 Abstract

Currently, the concept of blockchain is increasingly popularized. In this regard, the number of different applications interdependent with this concept is growing. One of the most popular objects is NFT (non-fungible token). On this idea, there are a large number of protocols on different blockchains that allow you to exchange NFTs on trading floors. The goal of this team project is to implement a discord bot with NFT marketplace functionality on the new and rapidly growing NEAR Protocol blockchain and an NFT generation service using a generative adversarial network. To do this, it was necessary to implement the NFT smart contract (according to the NEP-171 standard), the marketplace smart contract, adjust the API for interacting with the NEAR-protocol blockchain under the capabilities of discord and implement the discord-bot, ... (here is something about the model).

1.3 Список ключевых слов

Блокчейн, near, smart-контракты, non-fungible token, генеративно-состязательная сеть, discord-бот, маркетплейс.

2 Введение

2.1 Актуальность и значимость

2.2 Постановка задачи

В качестве блокчейна используется NEAR protocol[1]. NEAR Protocol работает по схеме Proof-of-Stake(Pos). Отличительные черты относительно других блокчейнов - улучшенная масштабируемость, производительность, а также простота реализации приложений.

Определение. *Блокчейн - децентрализованная база данных, которая содержит информацию о всех операциях произведенных в ней. Информация об операциях хранится в виде цепочки блоков. Удалить или изменить цепочку блоков невозможно, все это защищено криптографическими методами. Самым первым блокчейном является Bitcoin [2].*

Определение. *DApps — это приложения, которые включают логику работы с функциями блокчейна [3].*

Самой значимой частью реализации DApp являются Smart-контракты. Копии Smart-контрактов разворачивается с помощью специальной транзакции на всех узлах-участниках и исполняются в сети блокчейна.

Определение. *Smart-контракт — это неизменяемый исполняемый код, представляющий логику DApp, работающий в блокчейне [3]. Часто сокращают до слова контракт. В некоторых протоколах называют по-другому, например в Solana - это программы[4].*

Определение. *Транзакция — это наименьшая единица работы, которая может быть назначена сети блокчейна. Работа в данном случае означает вычисление(выполнение функции) или хранение(чтение/запись данных)[5].*

Определение. *Узлы-участники/валидаторы - множество машин, которое обрабатывает транзакции в блокчейне.*

Для написания smart-контрактов Near protocol предоставляет sdk на языках Rust и AssemblyScript(near-sdk-rs[6] и near-sdk-as[7] соответственно). В

данном проекте smart-контракты NFT и маркетплейса реализовываются на языке Rust.

Каждый smart-контракт в Near(написанный на Rust/Assembly Script) переводится в WebAssembly(Wasm), который непосредственно исполняет виртуальная машина на участвующем узле(валидаторе) блокчейна. У smart-контракта, есть два вида функций: которые меняют состояние блокчейна - «change operations» и так называемые «view operations», которые не меняют состояние машины, из названия данных операций можно понять, что первый вид операций, что-то сохраняет в блокчейн, а другая получает некоторую информацию с блокчейна, то есть readonly операция. Каждая операция имеет некоторую стоимость, которая измеряется в «Gas» [3, 8]. Также есть «payable» операции, которые запрашивают некоторую сумму токена, но это больше не как вид функций, а дополнение к ним.

Замечание. *Gas: сборы на исполнение транзакции не рассчитываются в токенах NEAR, вместо это она рассчитываются через Gas. Преимущество в том, что данные единицы - детерминированы, то есть одна и та же транзакция будет всегда будет стоить одинаковое количество Gas. Стоимость Gas пересчитывается в зависимости от загруженности сети в блокчейне [8].*

Для того, чтобы уметь работать с NFT, нужно написать соответствующий Smart-контракт, он базируется на описанном стандарте в спецификации Near Protocol [8, 9].

2.3 Этапы проекта

В рамках групповой курсовой работы была поставлена цель реализации discord-бота с функционалом NFT маркетплейса в NEAR protocol и сервисом генерации NFT, используя генеративно-состязательную сеть. Для реализации данной цели были выделены следующие этапы:

— Изучить теоретический базис связанный с NEAR Protocol(Лущ, Басалаев, Токкожин, Кусиденов)

— Реализовать smart-контракты(Басалаев):

  –

— Разработать discord-бота(Лущ):

  – Изучить Javascript/Typescript;

– Изучить основы работы с браузером через Javascript(сессионное/локальное хранилище браузера, класс window);

– Изучить near-api-js и его кода для дальнейшего его переписывания под функциональность discord;

– Реализовать KeyStore[10] работающий через Redis[11];

– Написать реализацию авторизации в Near Wallet[12] через discord-бота, который использует вышеописанный KeyStore;

– Написать реализацию создания url на подпись транзакции/транзакций(одна транзакция[1], один Action[14]; одна транзакция, несколько Action; несколько транзакций, несколько Action);

– Изучение децентрализованных распределенных хранилищ;

– Создание «Профиля пользователя»(Вызов осуществляется через slash-команду[15] или контекстное меню[16]);

– Реализация просмотра списка NFT, которыми владеет пользователь, которые продает пользователь, которые продаются на всем маркетплейсе(Вызовы осуществляются через контекстные меню, slash-команды, кнопки в профиле пользователя. Список выглядит по-разному в зависимости от количества NFT, если NFT много, то будет подгружаться только часть в целях оптимизации ресурсов);

– Поддержка покупки, продажи, отмены продажи NFT(Вызовы в виде кнопок при просмотре NFT списка);

– Поддержка изменения цены NFT // пока что не сделано, но сделан метод в smart-контракте;

– Поддержать сервис GAN в discord-bot // пока что не сделано;

– Сделать docker образ для удобного деплоя discord-бота;

– Деплой discord-бота на облачный сервис(Кусиденов);

— Реализовать GAN(Токкожин):

— Реализовать сервис GAN(Кусиденов):

[1)] В данном контексте класс Transaction[13]

2.4 Структура работы

Обзор существующих работ и решений

На сегодняшний день NFT маркетплейсов огромное количество, на блокчейне Ethereum, как примеры: opensea[2], rarible[3], на Solana: solanart[4]. На Near protocol, на данный момент самыми популярными являются: Paras[5], Mintbase[6].

Из двух представленных маркетплейсов, наиболее популярным является Paras. Он представляет базовый функционал маркетплейса: аутентификация в кошелек; покупка, продажа, просмотр NFT. У Paras также есть свой токен - Paras. В Paras представлены следующие медиа файлы: картины, фотографии, пиксель-арты.

Хоть и Mintbase менее популярен, чем Paras, но представляет гораздо больше видов медиа объектов: 3D модели, gif-изображения, аудио объекты. Но сама подача(оформление сайта) выглядит гораздо хуже, чем в Paras.

Объединяет данные маркетплейсы то, что у них единый стандарт реализации NFT, предоставленный Near protocol [9]. Уже как 3 года есть стандарт ERC-721[7], который был утвержден сообществом Ethereum. Стандарт в Near немного расширяет возможности, из-за уникальных возможностей реализованных в Near protocol [9]. Наша команда планирует придерживаться данного стандарта и реализовывать его, возможно потребуется некоторое его расширение, но об этом пока неизвестно. Важным вопросом является хранение в блокчейне самого NFT.

В данном стандарте количество, балансы ограничены 128-битным беззнаковым целочисленным типом. Вся сериализация, десериализация происходит с помощью JSON. Сам контракт отслеживает изменения в хранилище.

NFT хранит следующую информацию о себе: сам идентификатор токена, ID владельца и метаданные этого токена. Метаданные делятся на типа: первый класс, который хранит версию, название токена, ссылка на информацию, которая представляется в виде JSON, ссылка на sha256 хеш. Другой класс хра-

[2] https://opensea.io/
[3] https://rarible.com/
[4] https://solanart.io/
[5] https://paras.id/
[6] https://www.mintbase.io/
[7] https://ru.bitcoinwiki.org/wiki/ERC-721

нит: название, описание, время создания, время истечение представленное в UNIX epoch.

В Smart-контракте NFT должны быть реализованы следующие функции:

— nft_transfer - «change operation> функция передачи NFT от одного аккаунта другому.;

— nft_transfer_call - такая же «change operation» функция, что и предыдущая, за исключением того, что вызывается nft_on_transfer при удачной передачи NFT и nft_resolve_transfer при неудачной попытке;

— nft_token - «view operation» функция, которая по ID токена возвращает всю информацию о нем.

Предлагаемые подходы и методы

Для проведения транзакций(исполнения Smart-контрактов) NEAR protocol предоставляет near-cli[8] и near-api-js[9] [8]. Для наших целий требуется near-api-js, а следовательно вытекает потребность использовать язык javascript(nodejs, typescript) для написания «frontend» части бота.

Для реализации самого бота, как следует из названия проекта, используется Discord API. Но конечно, будет использоваться обертка над HTTP запросами. Существует множество всевозможных модулей для разнообразных языков. Для наших целей нам понадобится discord.js(javascript/nodejs)[10]. Также было принято решение, взять обвязку над этой библиотекой - discord.ts(typescript)[11]

Аккаунт

Аккаунты в NEAR protocol устроены так, что они имеют человекочитаемый ID в отличие от большинства других блокчейнов, где обычно используется некоторый hash. Длина логина от 2 до 64 символов и содержит в конце суффикс обозначающий сеть блокчейна: main, test, beta. Аккаунт может создавать подаккаунты, которые по своему функционалу ничем не отличаются от обычного аккаунта. Но они нам понадобятся так как, на один аккаунт

[8] https://github.com/near/near-cli

[9] https://github.com/near/near-api-js

[10] https://discord.js.org/

[11] https://discord-ts.js.org/

можно развернуть только один smart-контракт, а связанно это с тем, что при предложении подписания smart-аккаунта мы будем указывать название аккаунта, что сразу обозначает, который smart-контракт мы хотим инициировать для подписания. И самое главное, что нам нужно так это dev аккаунты. С помощью этих тестовых аккаунтов мы и будем тестировать наши smart-контракты. Они создаются с помощью ранее упомянутого near-cli [8]. Для того, чтобы мы могли предлагать на подпись smart-контракты, нам нужно, чтобы этот пользователь авторизовался в NEAR wallet или иными словами подписать некоторую транзакцию на образование Access Key.

Access Key

Каждый аккаунт имеет создавать множество public/private ключей, которые в Near называются Access Key. Существует два типа Access Key: FullAccess и FunctionCall. Из названия первого можно предположить, что он дает полный доступ и он нам не годится, так как пользователи попросту не будут доверять нашему приложению. FunctionalCall ключ уникален и дает разрешения только на подписание функций контрактов, которые не являются «payable». Он имеет несколько атрибутов: количество Near, которые разрешается тратить на Gas, название контракта, чьи функции разрешается вызывать и названия этих функций, которые будут вызываться. Рисунок 2.1

Хранение в блокчейне

Важным вопросом, который не затрагивался при описании стандарта является хранение медиа объекта.В описанном стандарте хранится url на медиа объект, но никак не сам этот медиа объект, связанно это с тем, что хранение объекта невероятно дорогое. Авторы стандарта предлагают несколько подходов, один из них: пользователь изначально просто дает url и мы храним его, но в данном решении есть огромный минус - это то, что мы накладываем обязательство на пользователя об поддержании данного url валидным. Следующий подход заключается в том, что мы - сервис будем хранить данные в централизованном хранилище и тогда нашим опознавательным знаком будет какой-нибудь ключ, который будет соответствовать адресу NFT. И в этом подходе есть тоже минус в виде того, что непонятно какого размера должно быть это хранилище, какие ограничения вводить на размер медиа файла. И

последнее предложение, так это хранить в специальном децентрализованном контенто-адресованном хранилище. Одним из примеров таких хранилищ является filecoin[12]. Минусом данного решения, наверное, является то, что мы начинаем использовать еще один блокчейн, который требует времени для понимания его спецификации. Наша команда приняла решение двигаться от самого простого подхода, то есть использование отданного пользователем url, к более сложному - использование filecoin.

Машинное обучение в маркетплейсе

За данную часть я отвечаю в меньшей степени. Но в рамках данного проекта нашей командой принято решение реализовать две идеи: рекомендательную систему или GAN. Рекомендательная система также делится на две части - это предсказание цены за конкретную NFT и более сложная идея - это рекомендовать сам NFT-токен. Идея с GAN выглядит гораздо легче, сравнивая с рекомендательной системой, например, создание аниме персонажа, пиксель арта, которое в последствии пользователь сможет использовать в своих коллекциях или для продажи.

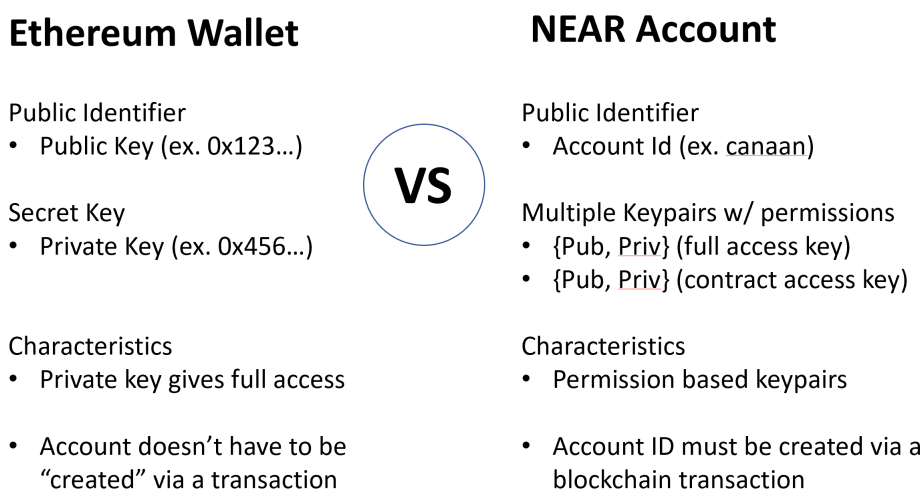

Рисунок 2.1. Сравнение спецификации аккаунтов и public/private ключей в блокчейнах Ethereum и Near. Источник: medium.com/@clinder

[12] https://filecoin.io/

3 Список источников

[1] *NEAR Protocol*. 2022. URL: https://near.org/.

[2] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2009. URL: http://www.bitcoin.org/bitcoin.pdf.

[3] Bina Ramamurthy. *Blockchain in Action*. S.l: Manning Publications, 2020. ISBN: 9781617296338.

[4] Solana Foundation. *Introduction*. 2022. URL: https://spl.solana.com/.

[5] NEAR Protocol. *Transaction*. 2022. URL: https://docs.near.org/docs/concepts/transaction.

[6] NEAR Protocol. *Near/near-sdk-rs: Rust library for writing near Smart Contracts*. 2022. URL: https://github.com/near/near-sdk-rs.

[7] NEAR Protocol. *Near/near-sdk-as: AssemblyScript library for writing near Smart Contracts*. 2022. URL: https://github.com/near/near-sdk-as.

[8] NEAR Protocol. *NEAR DOCS*. 2022. URL: https://docs.near.org/.

[9] NEAR Protocol. *NEAR Protocol Specification*. https://nomicon.io/ and https://nomicon.io/README.html. 2022.

[10] NEAR Protocol. *Class KeyStore*. 2022. URL: https://near.github.io/near-api-js/classes/key_stores_keystore.keystore.html.

[11] Redis Ltd. 2022. URL: https://redis.io/.

[12] Roketo Labs LTD. *Near wallet*. 2022. URL: https://wallet.near.org/.

[13] NEAR Protocol. *Class transaction*. URL: https://near.github.io/near-api-js/classes/transaction.transaction-1.html.

[14] NEAR Protocol. *Class action*. 2022. URL: https://near.github.io/near-api-js/classes/transaction.action.html.

[15] discord.js. *Discord.js guide*. URL: https://discordjs.guide/interactions/slash-commands.html#registering-slash-commands.

[16] discord.ts. *@ContextMenu*. URL: https://discord-ts.js.org/docs/decorators/gui/context-menu/.

4 Приложения

4.1 Ссылка на репозиторий

Ссылка на Gitlab репозиторий с проектом - Gitlab