ДЗ “Взаимодействие со СмК”

Форма: код в github репозитории

Область: Ethereum

Цель работы: получить практические навыки взаимодействия с СмК из внешних информационных систем

Результат: доработан код СмК, написан JS-скрипт взаимодействующий с СмК

Описание

Теория

Общая концепция

Размещаемые в блокчейне смарт-контракты находятся в изолированной среде, они не могут самостоятельно взаимодействовать с внешними информационными системами. Для реализаций полезных бизнес задач, со смарт-контрактами нужно как-то взаимодействовать. Чаще всего, разработчики платформ создают в своих платформах отдельный сервер, который принимает от внешнего пользователя запросы и преобразует в понимаемый блокчейном вид. В платформах основанных на Ethereum (и во многих других) этот сервер реализуется с помощью JSON-RPC, который имеет множество методов, которые может вызывать пользователь, список конечных точек API приведен в ссылках. На вход API приходят подготовленные данные, задача пользователя правильно подготовить данные, чтобы блокчейн их мог понять. Для решения этой задачи есть набор библиотек web3.

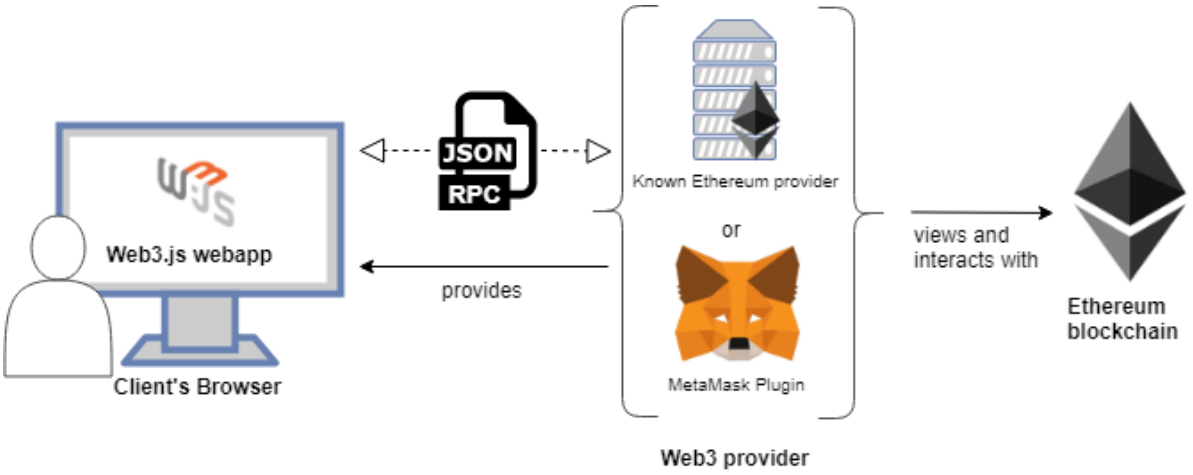
Web3

Web3 - это промежуточное программное обеспечение построенное поверх JSON-RPC, которое упрощает взаимодействие с системной информацией хранимой в блокчейне и смарт-контрактами, а также набор полезных инструментов разработчика (например, для работы с ключами).

Web3 - это общее название для множества библиотек, которые могут взаимодействовать с блокчейном. Из популярных: web3js (JavaScript), web3j (Java), ethers (JavaScript) и множество других реализаций на других языках

программирования. Web3 позволяет писать приложения как разработчик, чтобы пользователи могли взаимодействовать с блокчейном Ethereum.

Самая популярная и часто используемая библиотека - web3js, которые позволяют создавать web-приложения, для которых не нужен web-сервер, они напрямую взаимодействуют с блокчейном. Поставщиком данных и подключения в этом случае выступает одно из расширений для обозревателя (например, MetaMask), которые передают все запросы в блокчейн через свой шлюз.



Основные возможности web3 библиотек:

- Взаимодействие с функциями контракта
- Размещать контракты
- Отправлять данные в контракты и инициировать транзакции
- Запрашивать из блокчейна данные
- Отслеживание событий в контрактах из данных блока

Способы получения данных

Получить данные из блокчейна можно несколькими способами: вызов методы смарт-контракта, который возвращает данные и прослушивать события.

Вызов метода подразумевает создание специального вызова RPC, который возвращает ответ с запрашиваемыми данными, чаще всего это вызов метода смарт-контракта. Пример: [web3.eth.Contract — web3.js 1.0.0 documentation]

Работа с событиями требует отдельного пояснения.

Благодаря web3 работа с сетью Ethereum похожа на работу с обычным приложением, API которого представлено публичными методами смарт-контракта. Но мы не можем сохранять данные в блокчейне так, как бы мы делали при работе с обычной базой данных. Но нам требуется где-то хранить историю вызовов наших методов, например историю всех начислений и списаний дивидендов. Если мы будем при любой операции добавлять соответствующую запись в некий список, созданный для этого внутри смарт-контракта, то на каждой транзакции придётся перезаписывать этот список в сеть с добавлением новой записи, а это дорого. Для решения данной проблемы можно использовать events - события смарт-контракта, аналогичные EventEmitter в javascript. В коде смарт-контракта мы генерируем событие, в коде нашего клиента - слушаем его. В нашей задаче вместо того, чтобы слушать текущие события, мы будем анализировать уже произошедшие и из них собирать список изменений, произошедших в смарт-контрактах.

Если мы разрабатываем пользовательское приложение на javaScript, то в экземпляр нашего смарт-контракта (созданного в JS), будет набор функций по работе с Событиями, например getPastEvents.

```
contract.getPastEvents(
  'hasVoted',
  {
    fromBlock: 6321265,
    toBlock: 'latest'
  },
  (err, events) => {
console.log(events) }
)
```

- В поле ‘hasVoted’, мы можем указать ‘AllEvents’, чтобы слушать все события
- “fromBlock” и “toBlock” указывают с какого блока и до какого блока слушаем

События позволяют отображать изменения в пользовательском интерфейсе.

Механизм События позволяет получать информацию по изменениям в блокчейне, бз необходимости вызова методов.

Задание

Подготовка окружения (5 баллов)

1. Убедитесь, что у вас установлены node.js и npm
2. Создайте в папке проекта раздел client
3. Инициализируете в папке client nodejs приложение
4. Установите зависимости **web3.js** или **ethers.js**
5. Создайте конфигурационный файл с подключением к Infura или Alchemy

Доработка СмК (+2 бала)

1. Добавьте в структуру СмК **структуру** произвольного содержания (3-4 разнотипных поля)
2. Добавьте **отображение** ключа (произвольный тип) на значение (ваша структура шага 1)
3. Добавьте функцию добавление структуры в отображение
4. Добавьте функцию удаления структуры из отображения
5. Добавьте **события (event)** в созданные функции

Взаимодействие с СмК (+2 балла)

1. Разместите СмК в сети Goerli
2. Подготовьте JS-скрипт обращения к смарт-контракту по адресу через сеть Goerli
3. В скрипте сделать вызов функций смарт-контракта
4. Добавить в скрипт функцию запроса событий по адресу и фильтру

Реализация на другом ЯП (+ 1 балл)

1. Выполнить Подготовку окружения, Доработку СмК и Взаимодействие с СМк на другом ЯП, отличном от JS\TS

Контрольные вопросы (+1 балл)

1. Какие есть расширения для обозревателя, которые умеют взаимодействовать с Ethereum?
2. Что делает MetaMask, чтобы web-приложение могло взаимодействовать с блокчейном через JavaScript?
3. Каким образом обрабатываются пользовательские данные в web-приложении, чтобы они были приняты блокчейном?
4. Какие есть способы взаимодействия с блокчейном, кроме RPC-сервера и в чём их особенность?

5. Какие есть ещё способы получения данных из блокчейна, кроме отправка call методов и событий?

Условие

1. Вместо web3 можно использовать ethers
2. Разместить код в Гитхабе
3. В файле README.md ответить на вопросы

Полезные ссылки

1. web3.js - Ethereum JavaScript API — web3.js 1.0.0 documentation
2. json-rpc | Ethereum Wiki
3. web3.js - Ethereum JavaScript API — web3.js 1.0.0 documentation
4. https://soliditylang.org/
5.