Слайд 1:

Добрый день, дорогие слушатели. Сегодня я вам представлю вторую часть доклада «Создание децентрализованной торговой платформы с применением эскроу смарт-контракта на базе платформы Ethereum»

Слайд 2:

Целью работы является разработка децентрализованного приложения с применением эскроу смарт-контракта на базе блокчейн платформы.
А наши задачами были:

- Разработать умный контракт
- Разработать back-end и front-end части сайта
- Связать умный контракт и front-end часть сайта

Слайд 3:

Так как наш смарт контракт должен работать в сети Ethereum, разработка проходила на языке Solidity. Существует несколько площадок для разарботки смарт контрактов, но самый удобный - Remix IDE. Он позволяет отлаживать, компилировать а так же размещать смарт контракт в сети ethereum или на виртуальной JavaScript машине. Лично я использую VS Code для написания и компилирования кода, а проверяю его работоспособность в remix IDE.

Слайд 4

Для хранения заказов на сайте, мы используем умный контракт. Данные о заказе хранятся в структуре. На слайде приведен пример создания структуры с комментрированием строчек. Далее создается маппинг для хранения заказов. Маппинг можно представить как хэш-таблицу, которая инициализируются таким образом, что все возможные ключи существуют и сопоставляются со значениями, байтовое представление которых - нули(значение по умолчанию).

Слайд 5

Связь умного контракта с сайтом осуществляется по средствам библиотеки web3.js или web3.py. Данные библиотеки позволяют взаимодействовать с Ethereum блокчейн и умными контрактами. web3.js позволяет взаимодействовать как на front-end'e, так и на back-end'e(например если у вас серверная часть сделана на node.js). web3.py в свою очередь поможет создать взаимодействие только на серверной части сайта.

Слайд 6

Варианты взаимодействия умного контракта с сайтом показаны на слайде

Слайд 7

Для связи смарт-контракта была использована библиотека web3.js — API для JavaScript, которая позволяет взаимодействовать с MetaMask. Это криптовалютный кошелек, который работает как расширение для браузеров. Через MetaMask покупатель производит основные транзакции на счет смарт-контракта. Также эта библиотека позволяет работать с блоками сети Ethereum, то есть напрямую обращаться к смарт-контракту, используя HTTP, IPC или WebSocket. В нашем случае для связи клиента с сервером использовался протокол HTTP.

Слайд 8

На данном слайде представлен пример взаимодействия с умным контрактом. Объявляется функция cancelOrder-user. Первым делом создается соединения с блокчен. Далее создается сущность контракта из функции, которую мы разберем чуть позже. После этого выполняется основное действие данной функции - происходит вызов функции самого смарт контракта через api web3.js

В функции initContract создается сущность контракта. Указав abi(не знаю надо ли говорить что такое abi или будет лишнее), индекс сети, в которой размещен наш умный контракт, а так же его адрес, вызывается функция определения сети, к которой сейчас подключен web3. После данныk действий вызывается функция создания сущности контракта.

abi(application binary interface) - Двоичный(бинартный) интерфейс приложений ABI определяет, как вызываются функциии и в каком двоичном формате информация должна быть передана от одного программного компонента к другому. Умный контракт Ethereum это байткод, размещенный на Ethereum блокчейн. В контракте может быть несколько функций. ABI необоходим, для того, чтобы можно было указать какую функцию в контракте вызвать. А так же ABI нужен для того, чтобы у нас была гарантия, что функция вернет данные в формате, который мы ожидаем.

Слайд 9

Выводы