블록체인을 활용한
쇼핑몰 상품 순위 조작 방지 서비스

1871005 강예준, 1871227 임세진

https://youtu.be/KsHxfKBl3HQ

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

Contents

CryptoCraft LAB

1. 아이디어

1. 아이디어

- 쇼핑몰 랭킹 시스템을 통해 순위 조작 => 블록체인을 통해 투명한 시스템 환경 구성

- 예) 상품 판매량 변수 조작, 랭킹 시스템 계산 함수 조작 등

온라인쇼핑몰 인기 상품, 알고 보니 순위 조작

“좋아요”는 항상 맨위로, 재고를 인기상품 조작한 쇼핑몰

별5개 평점만 보이더니...'SNS쇼핑몰' 임블리, 리뷰 조작 적발

공정위, SNS 기반 온라인 쇼핑몰 7곳 적발
임블리, 평이 좋은 후기만 상단 노출
베스트 상품 메뉴에 재고 많은 제품 올려

1. 아이디어

- 사재기를 통한 랭킹 조작 => 상품 랭킹에 영향을 줄 수 있는 구매 개수 제한

- 예) 고객에게 판매량이 저조한 상품을 구매하도록 사주하여 판매량을 올리는 방식

'못 믿을 베스트셀러' 사재기로 순위 조작한 출판업자들

출판사가 도서 사재기로 베스트셀러 순위 조작

1. 아이디어

- 블록체인

- 상품 판매량을 블록에 저장하여 조작을 방지

- rank함수와 쇼핑몰 프로그램을 이더리움 상에서 솔리디티로 구현하여 시스템 함수 조작 방지

- 그 외

- 사재기를 방지하기위해 고객은 일정량 이상 구매하면 순위 count에 반영되지 않게 함

- 구매 함수에 Payable 속성을 지정하여 이더를 통한 쇼핑몰 거래 구현

2. 설계

2. 설계

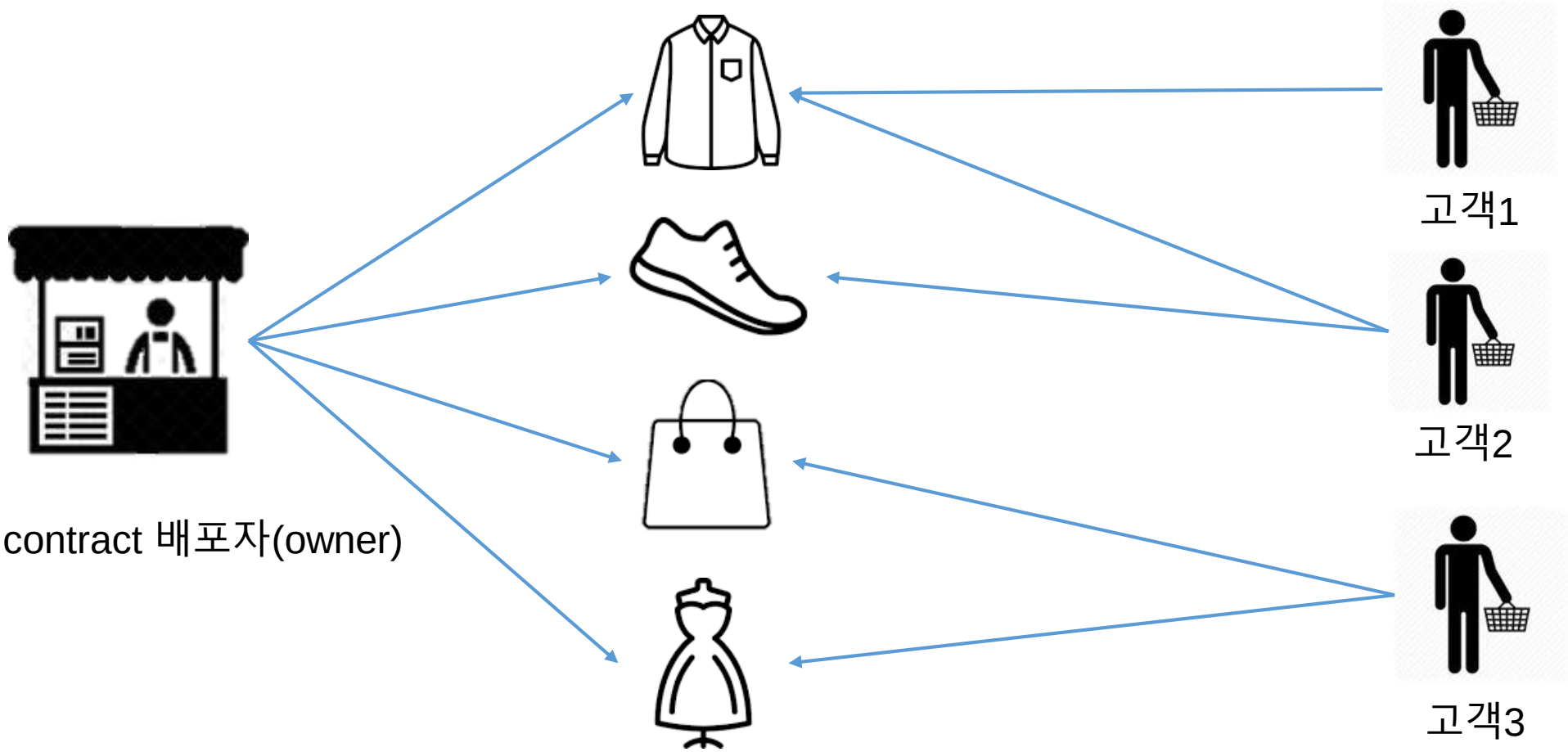
- Owner : 쇼핑몰 운영자
- 각각의 계정 : 소비자
- 핵심 아이디어 : 상품이름과 소비자 address를 결합한 byte32형태의 데이터를 개인별 구매량으로 매핑해주는 personalsales 이용하여 코드 간결화

personalsales : (상품이름+소비자 address) → 구매량

고유한 값

2. 설계

하나의 제품에 한사람 당 50개까지 랭킹 count에 반영 가능



판매자 : contract 배포자(owner)

3. 서비스 시연

3. 서비스 시연

KangStore.sol

```
pragma solidity 0.4.24;

import "./Safemath.sol";

contract KangStore {
    struct Product {
        string name;
        uint price;
        uint sales;
    }

    using SafeMath for uint256;

    Product[] public productList;
    address owner;

    mapping (bytes32=>uint) personalsales;    //상품명+고객의 id를 합친 값을 매핑 (개인의 구매량)

    mapping (string=>uint) productId;

    //event는 web3에서 필요함
    event NewProduct(uint id, string name);
    event Set(address owner);

    constructor() public{
        owner = msg.sender;
        emit Set(owner);
    }

```

3. 서비스 시연

```
30    modifier onlyOwner{
31        require(msg.sender==owner);
32        _;
33    }
34
35    function addProduct(string _name, uint _price) onlyOwner {
36        uint id = productList.push(Product(_name, _price, 0))-1;
37        productId[_name] = id;
38        emit NewProduct(id, _name);
39    }
40
41    function withdrawFunds() public onlyOwner {
42        require(owner.send(address(this).balance));
43    }
44
45    function purchase(string _name, uint _amount) public payable {
46        require(msg.value==productList[productId[_name]].price * _amount);
47
48        bytes32 personalsales_Id=keccak256(abi.encodePacked(productList[productId[_name]].name, msg.sender));
49
50        if(personalsales[personalsales_Id]+_amount<= 50){ // (기존에 샀던 개수 + 사려는 개수) <= 50 인 경우
51          productList[productId[_name]].sales=productList[productId[_name]].sales.add(_amount);    // 총판매량 amount만큼 ++
52          personalsales[personalsales_Id]=personalsales[personalsales_Id].add(_amount);   //개인 구매량 amount 값만큼 ++
53      }
54      else if (personalsales[personalsales_Id] > 50){    //개인 구매량이 50보다 큰 경우
55          personalsales[personalsales_Id]=personalsales[personalsales_Id].add(_amount);   //개인 구매량 amount 값만큼 ++
56      }
57      else {   // (기존에 샀던 개수 + 사려는 개수) > 50 인 경우
58
```

3. 서비스 시연

```
57        else {   // (기존에 샀던 개수 + 사려는 개수) > 50 인 경우
58            productList[productId[_name]].sales=productList[productId[_name]].sales.add(50-personalsales[personalsales_Id]); // 총판매량에 기여할 수 있는 값은 50이기 때문
59            personalsales[personalsales_Id]=personalsales[personalsales_Id].add(_amount);   //개인 구매량 amount 값만큼 ++
60        }
61    }
62
63    function rank() public {
64        uint size = productList.length;
65        require(size > 0); //상품이 아예 없을 때 실행 불가
66
67        for(uint i=0;i<size;i++)
68        {
69            for(uint j=i+1;j<size;j++)
70            {
71                if(productList[i].sales < productList[j].sales)
72                {
73                    Product memory temp= productList[j];
74                    productId[productList[i].name] = j;
75                    productId[productList[j].name] = i;
76                    productList[j] = productList[i];
77                    productList[i] = temp;
78
79                }
80            }
81        }
82    }
83 }
```

3. 서비스 시연

Safemath.sol

```
pragma solidity 0.4.24;

library SafeMath {
  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
    if (a == 0) {
      return 0;
    }
    uint256 c = a * b;
    assert(c / a == b);
    return c;
  }

  function div(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a / b;
    return c;
  }

  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    assert(b <= a);
    return a - b;
  }

  function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    assert(c >= a);
    return c;
  }
}

```

감사합니다.