

이더리움 기반 분산형 수산물 유통 플랫폼



류지환

엄혜림

박서현

지도교수 권 준 호

목 차

1. 서론.....	1
1.1. 연구 배경	1
1.2. 기존 문제점	1
1.3. 연구 목표.....	3
2. 연구 배경	3
2.1. 블록체인 네트워크.....	3
2.1.1. Private 네트워크.....	3
2.1.2. 이더리움	4
2.2. EVM	4
2.3. 스마트 컨트랙트	5
2.4. IPFS	6
2.5. DApp 구성도.....	7
3. 연구 내용	8
3.1. 블록체인 프라이빗 네트워크 구축.....	8
3.2. 스마트 컨트랙트 구현	10
3.3. MetaMask 연결.....	12
3.4. IoT 서비스	13
3.5. QR 코드.....	15
4. 연구 결과 분석 및 평가	16
5. 결론 및 향후 연구 방향	20
6. 참고 문헌	20

1. 서론

1.1. 연구 배경

우리가 흔히 먹고 있는 생선들이 어떤 과정을 거쳐 유통 되었는지 알 지 못한다. 또한, 수산물 유통 과정 중 경매가 이루어지는데 기존 경매 내역들은 수기로 작성되어 위변조가 가능하다. 따라서, 플랫폼에서 발생한 거래 내역과 유통과정을 블록체인에 기록하여 경매 내역의 위변조를 불가능하게하고 거래 내역의 역추적이 가능하게 함으로써 정보의 투명성을 확보하고자 한다.

1.2. 기존 문제점

소비자들은 구매하려는 수산물의 유통 과정과 원산지를 투명하게 확인하기 어려워 수산물 구매 시 원산지 표시에 의존할 수밖에 없다.

하지만 해양수산부 조사에 따르면 지난 4년간 원산지 미 표시 및 표시방법 위반 사례가 4936건, 거짓표시는 1700건으로 집계되어 그 신뢰성을 의심받고 있다.

"수산물 원산지 허위 표시 4년간 19% 증가...158억원 규모"

원산지 허위 표시 규모

(단위 : 건)

위반현황				
순위	미표시 및 표시방법 위반		거짓표시	
	생산지 (비중 %)	건수	생산지 (비중 %)	건수
1	국내산 (56.2%)	2,772	중국 (39.8%)	401
2	중국 (19.9%)	983	일본 (15.8%)	159
3	러시아 (6.4%)	314	원양산 (7.2%)	73
4	일본 (5.6%)	274	러시아 (6.8%)	68
5	베트남 (1.8%)	88	국내산 (3.4%)	34

자료: 해양수산부, 어기구원실 재구성

원산지 표시 위반 현황

이 같은 원산지 표기를 위반한 허위매물 단속은 쉽지 않은 실정이다. 거래 증빙 자료를 보관할 의무가 없기 때문이다. 한국해양수산개발원 현안보고서에 따르면

수산물 은 농산물과 축산물에 비해 유통과정이 불투명해 거래 증빙 자료 발급 의무 부과가 어려운 것으로 파악됐다.

유통 과정 뿐만이 아니다.

현재 수산물은 주로 생산자 -> 산지위판장 -> 소비지 도매시장 -> 도매상 -> 소매상을 거쳐 소비자로 유통된다.

위 과정 중 경매가 이루어지는데 산지 위판장에서 1차 상장 경매, 소비지 도매시장에 와서 2차 상장 경매를 거쳐야 한다.

기존 경매 내역들은 수기로 작성되기에 위변조가 용이하고 불투명한 거래가 이뤄질 위험이 크다.

이를 해결하기 위해선 경매의 투명성을 높여 경매 기록의 위변조를 방지해야 한다.

법원, 낙찰 수산물 '경매기록 위조' 불법거래 중도매인 벌금형

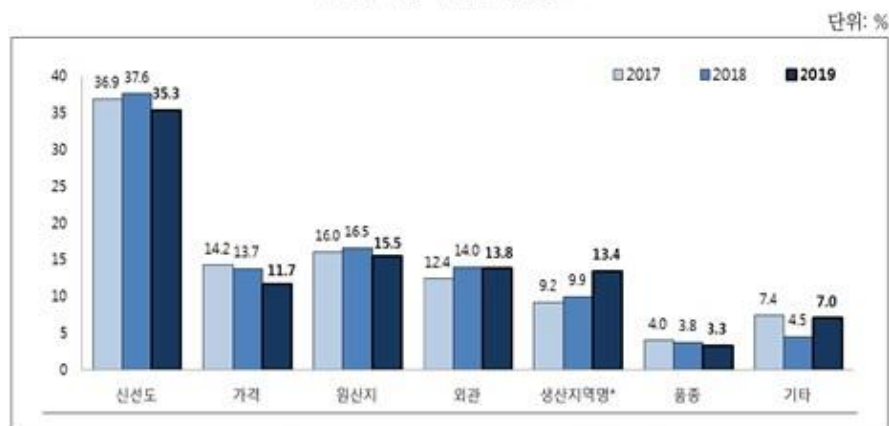
김태홍 기자 | 승인 2019.02.01 13:02 | 댓글 0

경매 위변조 사례

수산물 시장에서 유통과정과 경매내역의 투명성을 확보해 원산지와 가격 정보의 신뢰성을 높이는 것만큼 중요한 요소가 신선도 유지이다.

해양수산부의 조사에 따르면, 수산물을 구입할 때 소비자들이 중요하게 생각하는 요소는 1위 신선도(32.3%), 2위 원산지(16.2%), 3위 가격(15.2%)으로 나타났다.

수산물 구입 시 우선 확인 정보



주 1) 우선순위 응답의 결과에 가중치를 부여(1순위×3+2순위×2+3순위×1)하여 백분율로 계산한 수치임.

2) *은 2018년 대비 2019년의 결과가 95% 신뢰 수준에서 유의한 차이가 있는 것으로 나타남.

수산물 구입 시 우선 고려 요소

소비자가 수산물을 구입할 때 신선도를 가장 중요하게 생각하는 만큼 소비자 입장에서 수산물이 유통과정을 거칠 때 신선도를 확인하기 위한 환경 정보를 제공해야 할 필요성이 있다.

1.3. 연구 목표

이더리움 기반 수산물 유통 플랫폼 개발을 목표로 한다. 플랫폼에서 발생한 거래 내역과 유통 과정을 원장에 기록하여 관리함으로써 정보의 투명성을 확보한다. 경매의 경우, **입찰에서 낙찰까지의 모든 과정을 블록체인 원장에 기록한다.**

경매 과정을 원장에 기록함으로써 경매 기록의 위변조와 같은 불법적인 거래의 추적에 용이하다

소비자가 유통과정 중 신선도 유지 여부를 확인할 수 있도록 **IoT를 활용해 수집한 배송 환경(온/습도) 또한 원장에 기록한다.**

또한, 거래가 이루어진 수산물의 거래 및 유통 과정을 손쉽게 확인 가능한 **QR코드를 발급** 또한 지원한다. 이를 통해 소비자가 수산물을 구매하기 전, QR코드로 앞선 거래 및 유통과정 모두 확인 가능하다.

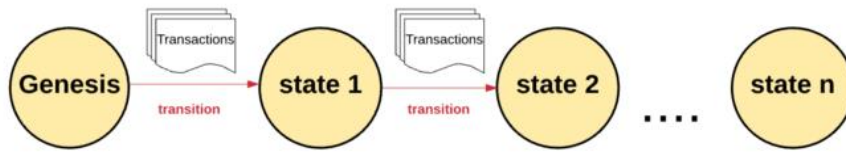
2. 연구 배경

2.1. 블록체인 네트워크

2.1.1. Private 네트워크

허가된 조직이나 개인들만 참여할 수 있는 폐쇄형 블록체인 네트워크다. 주로 기업에서 활용되며 중앙 기관이 대다수의 권한을 가지고 있다. 그만큼 확장이 간편하며 거래속도가 빠르고 인증을 거친 증명자만 참가하기에 합의 알고리즘 또한 간단하다.

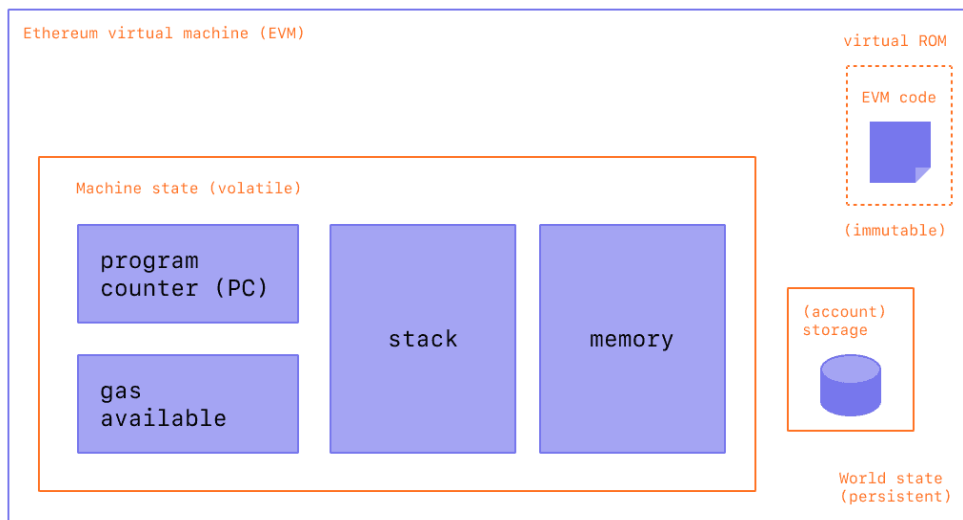
2.1.2. 이더리움



이더리움 상태 머신 구조

이더리움은 public 블록체인의 일종으로 트랜잭션에 기반한 상태 머신이다. 이더리움 상태 머신은 아무런 트랜잭션이 기록되지 않은 제네시스 상태에서 출발한다. 이더리움의 상태 전이는 스택 기반 가상머신 EVM에 의해 처리되며 하이레벨 프로그래밍 언어로 작성된 스마트 컨트랙트를 컴파일한 바이트코드를 실행한다.

2.2. EVM



EVM 구조

EVM은 이더리움 블록체인 네트워크의 노드들이 공유하는 가상머신이다.

EVM은 모든 이더리움 노드에서 로컬 인스턴스로 실행되지만 EVM의 모든 인스턴스는 동일한 초기 상태에서 작동해 동일한 최종 상태를 생성하기에 시스템 전체가 단일 컴퓨터로 작동한다.

모든 형태의 알고리즘을 처리할 수 있는 튜링 기계로서 먼저 들어온 데이터(Byte

code)를 우선적으로 처리하는 스택 구조를 가진다.

저장공간은 휘발성 저장공간인 Memory와 비휘발성 저장공간인 ROM과 Storage로 구성되어 있으며 Memory는 함수 호출 및 로컬변수를 저장하고 ROM은 영구히 저장되는 바이트 코드(스마트 컨트랙트), Storage는 스마트 컨트랙트의 상태변수를 저장한다.

2.3. 스마트 컨트랙트

스마트 컨트랙트는 바이트 코드로 컴파일 되어 EVM을 통해 모든 노드에서 실행되는 불변적인 프로그램이다.

일반적으로 솔리디티(Solidity), 바이퍼(Viper)와 같은 고급언어로 작성된다.

바이트코드로 컴파일된 후 컨트랙트 생성 트랜잭션을 사용하여 이더리움 블록체인 네트워크에 배포되며 이후 변경이 불가능하다.

수정을 위해선 기존 인스턴스를 삭제 후 수정한 인스턴스를 재배포해야 한다.

배포된 스마트 컨트랙트는 EOA(External Owned Account)의 트랜잭션으로만 호출된 경우에만 실행된다.

Contract를 기본 객체로 삼는 구조이며 내부 method로만 상태변수에 접근하도록 하며 휘발성 데이터는 EVM의 메모리, 비휘발성 데이터는 EVM의 Storage에 저장한다. 본 연구에서 구현할 스마트 컨트랙트는 아래와 같다.

- Auction 컨트랙트

거래별로 경매 내역을 상태 변수 배열에 저장한다.

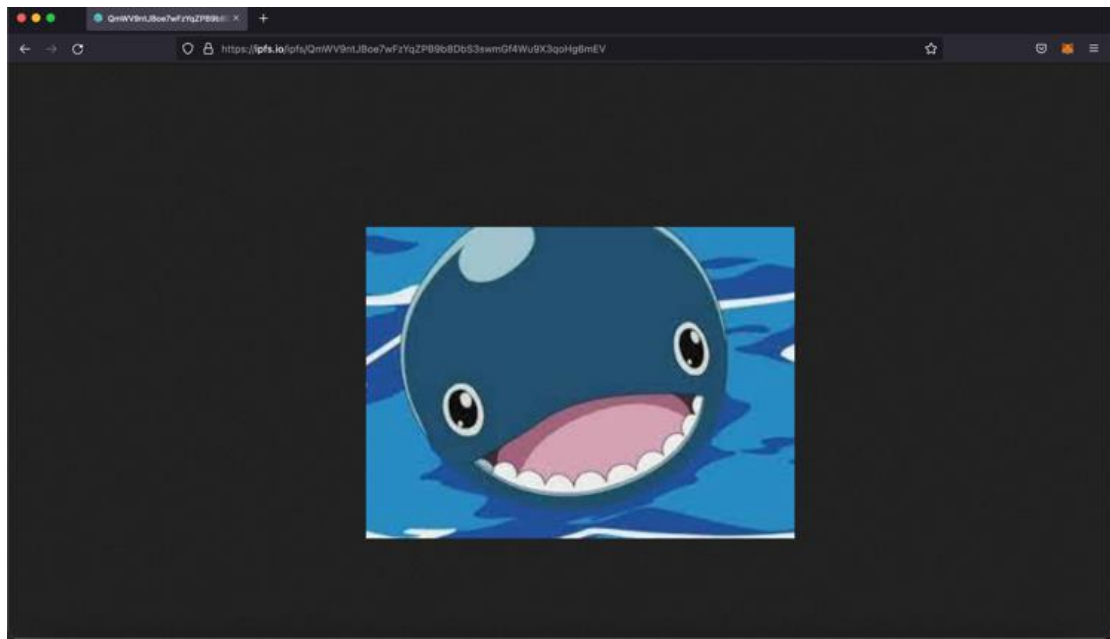
(거래->내역 mapping 방식)

DApp을 통해 생성된 내역 조회 트랜잭션을 수신하면 사용자가 지정한 내역 정보를 제공하는 트랜잭션(view)을 생성한다.

- Item 컨트랙트

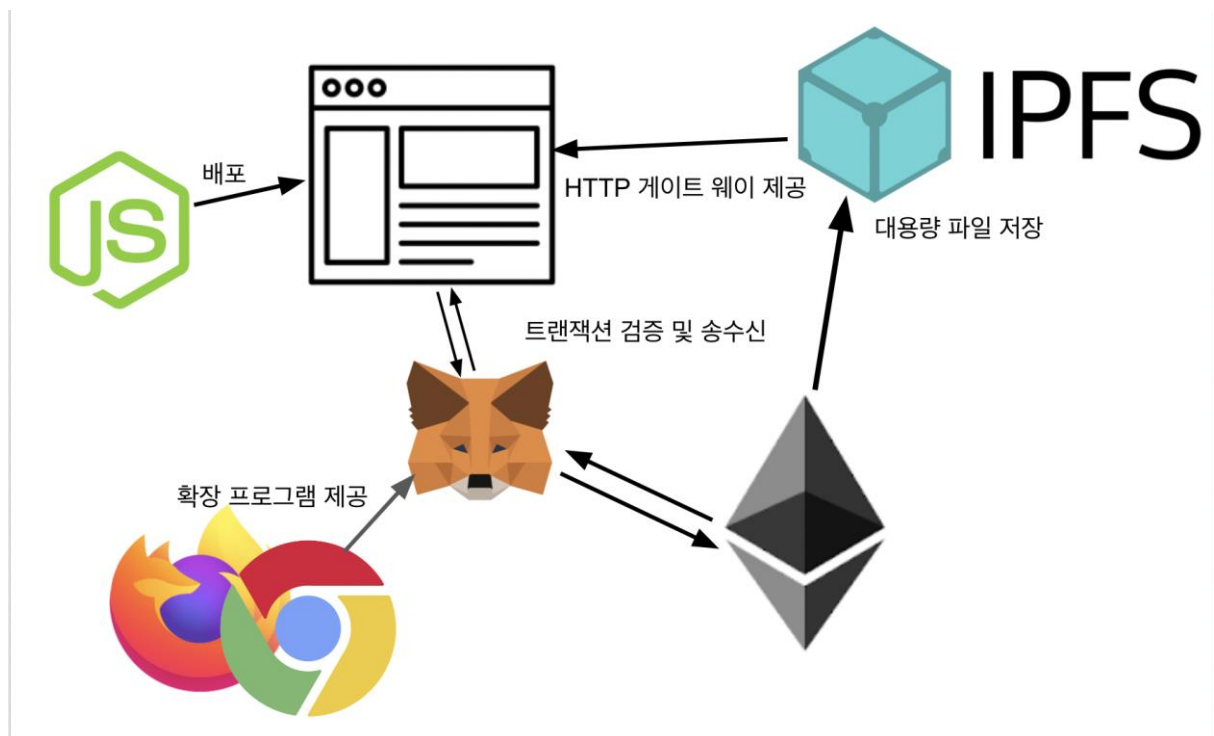
상품별로 상품 정보를 상태 변수 배열에 저장한다.

(상품 -> 정보 mapping 방식)



대용량 파일

2.5. DApp 구성도



DApp 구성도

3. 연구 내용

3.1. 블록체인 프라이빗 네트워크 구축

1. genesis.json 작성 및 genesis block 생성 (geth init)

```
{
  "config": {
    "chainId": 49277,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip150Hash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0
  },
  "alloc": {},
  "coinbase": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "difficulty": "0x200",
  "extraData": "",
  "gasLimit": "0x100000000",
  "nonce": "0x0000000000000042",
  "mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "timestamp": "0x00"
}
```

genesis.json

- 원활한 개발을 위해 채굴 난이도는 낮게 설정 (difficulty: 0x2)

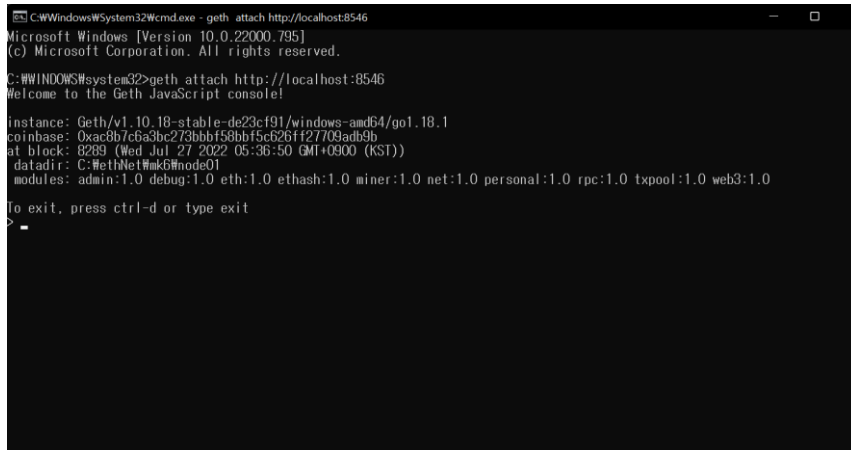
2. geth로 노드 실행

```
Microsoft Windows [Version 10.0.22000.795]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ryoo>geth --datadir C:\Users\ryoo0\MethNet --nodiscover --allow-insecure-unlock --syncmode "full" --port 30304 -
-http --http.port 8546 --networkid 49277 --http.api "admin, debug, web3, eth, txpool, personal, ethash, miner, net"
INFO [09-30|17:03:53.930] Maximum peer count ETH=50 LES=0 total=50
INFO [09-30|17:03:53.950] Set global gas cap cap=50,000,000
INFO [09-30|17:03:53.951] Allocated trie memory caches clean=154.00MiB dirty=256.00MiB
INFO [09-30|17:03:53.954] Allocated cache and file handles database=C:\Users\ryoo0\MethNet\geth\chaindata cache=5
12.00MiB handles=8192
INFO [09-30|17:03:54.000] Opened ancient database database=C:\Users\ryoo0\MethNet\geth\chaindata\ancient
readonly=false
INFO [09-30|17:03:54.004] Initialised chain configuration config="[ChainID: 1 Homestead: 1150000 DAO: 1920000 D
AOSupport: true EIP150: 2463000 EIP155: 2675000 EIP158: 2675000 Byzantium: 4370000 Constantinople: 7280000 Petersburg: 7
280000 Istanbul: 9069000 Muir Glacier: 9200000 Berlin: 12244000 London: 12965000 Arrow Glacier: 13773000 MergeFork:
<nil>, Terminal TD: <nil>, Engine: ethash]"
INFO [09-30|17:03:54.017] Disk storage enabled for ethash caches dir=C:\Users\ryoo0\MethNet\geth\ethash count=3
INFO [09-30|17:03:54.019] Disk storage enabled for ethash DAGs dir=C:\Users\ryoo0\AppData\Local\Methash count=2
INFO [09-30|17:03:54.022] Initialising Ethereum protocol network=49277 dbversion=8
INFO [09-30|17:03:54.025] Loaded most recent local header number=0 hash=d4e567..cb8fa3 td=17,179,869,184 age=53
y6mo5d
INFO [09-30|17:03:54.027] Loaded most recent local full block number=0 hash=d4e567..cb8fa3 td=17,179,869,184 age=53
y6mo5d
INFO [09-30|17:03:54.030] Loaded most recent local fast block number=0 hash=d4e567..cb8fa3 td=17,179,869,184 age=53
y6mo5d
WARN [09-30|17:03:54.033] Loaded snapshot journal diskroot=d7f897..0f0544 diffs=missing
INFO [09-30|17:03:54.035] Resuming state snapshot generation root=d7f897..0f0544 at=3d45db..20ec91 accounts=2171 s
lots=0 storage=100.00KiB dangling=0 elapsed=0s
INFO [09-30|17:03:54.036] Loaded local transaction journal transactions=0 dropped=0
INFO [09-30|17:03:54.042] Regenerated local transaction journal transactions=0 accounts=0
```

[그림] 노드 실행 커맨드

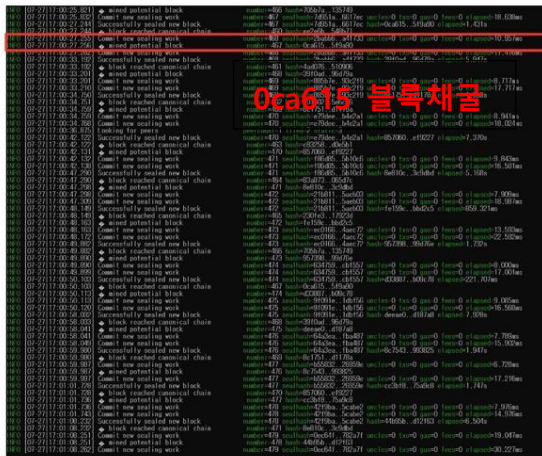
3. geth javascript RPC console로 노드를 조작



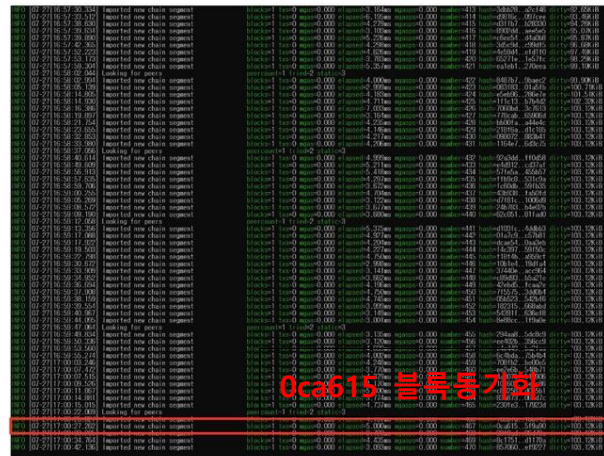
```
geth js console
```

4. 채굴을 통해 트랜잭션을 반영하여 로컬 네트워크 가동

채굴 노드에서 생성된 블록을 연결된 노드가 import하는 구조



채굴노드



연동

3.2. 스마트 컨트랙트 구현

1. Item, Auction, Supply, rwSupplyData Contract 개발 완료

```
pragma solidity >=0.5.16;
import "./Supply.sol";
contract AuctionContract {
    rwSupplyData rwData = new rwSupplyData();

    Auction[] public auctions;

    mapping(uint256 => Bid[]) public auctionBids;
    mapping(address => uint[]) public auctionOwner;

    mapping(address => uint256[]) public ownedBy;

    struct Bid{
        address payable from;
        uint256 amount;
        string shippingTo;
    }

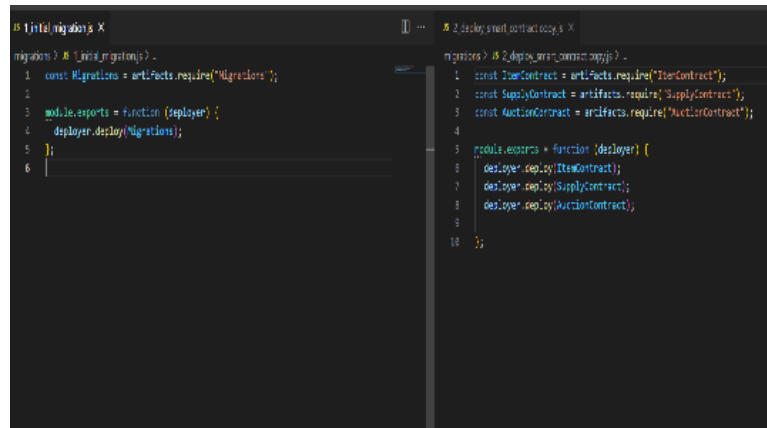
    struct Auction{
        string name;
        uint256 blockDeadline;
        uint256 startPrice;
        address payable owner;
        bool active;
        bool finalized;
        uint256 itemId;
        string shippingFrom;
        string shippingTo;
    }
}
```

Auction Contract

2. Truffle 설정 및 migrate 코드 작성

```
module.exports = {
  /**
   * Networks define how you connect to your ethereum client and let you set the
   * defaults web3 uses to send transactions. If you don't specify one truffle
   * will spin up a development blockchain for you on port 8545 when you
   * run 'develop' or 'test'. You can ask a truffle command to use a specific
   * network from the command line, e.g
   *
   * $ truffle test --network (network-name)
   */
  networks: {
    // Useful for testing. The 'development' name is special - truffle uses it by default
    // if it's defined here and no other network is specified at the command line.
    // You should run a client (like ganache, geth, or parity) in a separate terminal
    // tab if you use this network and you must also set the 'host', 'port' and 'network_id'
    // options below to some value.
    //
    development: {
      host: "127.0.0.1", // Localhost (default: none)
      port: 8546, // Standard Ethereum port (default: none)
      network_id: "49277", // Any network (default: none)
      gasLimit: 60000000
    },
  },
}
```

Truffle_config.js



Truffle deploy.js

3. Truffle console을 통한 컴파일 및 배포



Truffle migrate

4. 배포된 스마트 컨트랙트와 Web3 연동

```
var Auction_sol;  
var Item_sol;  
var Supply_sol;  
  
async function startApp() {  
  var AuctionAddress = "0xC3384d4e98F8751774f2475DDA7092a8Ca1A87aa";  
  var ItemAddress = "0xF8bb1Ed2Db631BA65469D5fa1e13992FCf3BB456";  
  var SupplyAddress = "0x019A210eB5f35C2753485ff3FEa94cff833C4cB8";  
  
  Auction_sol = await new web3.eth.Contract(AuctionABI, AuctionAddress);  
  Supply_sol = await new web3.eth.Contract(SupplyABI, SupplyAddress);  
  Item_sol = await new web3.eth.Contract(ItemABI, ItemAddress);  
}
```

web3 연동

3.3. MetaMask 연결

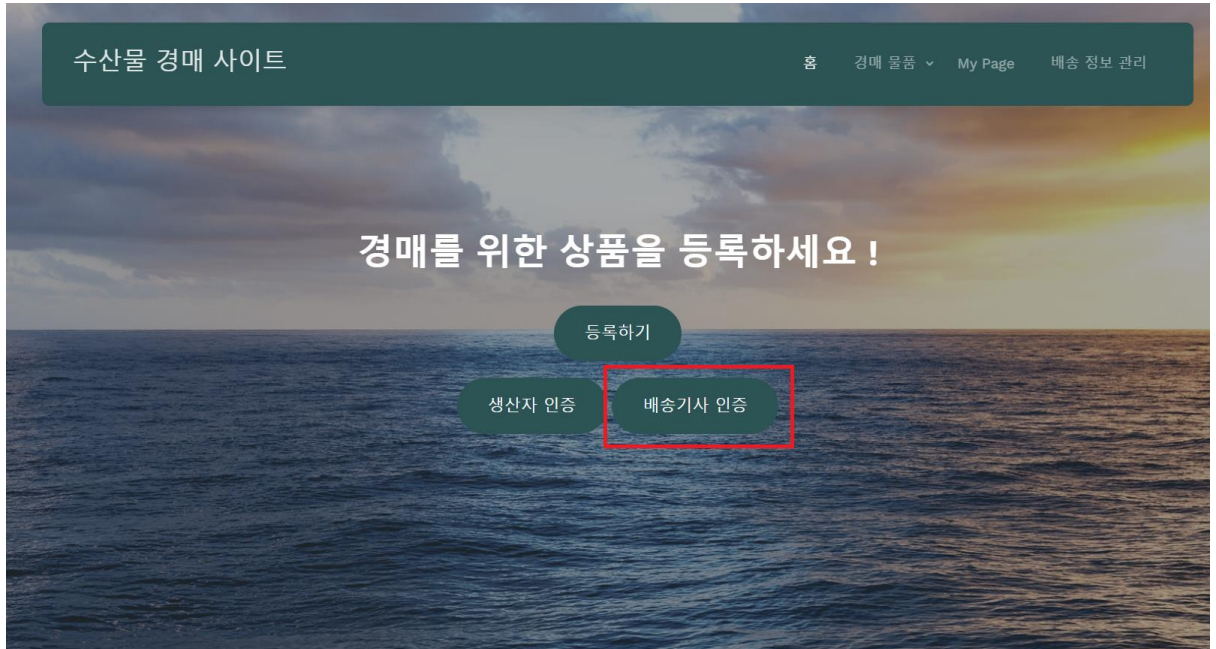
1. Web3와 MetaMask

```
window.addEventListener('load', function() {  
  if (typeof web3 !== 'undefined') {  
    web3 = new Web3(window.ethereum);  
  }  
  startApp();  
})
```

MetaMask, 스마트 컨트랙트 연동

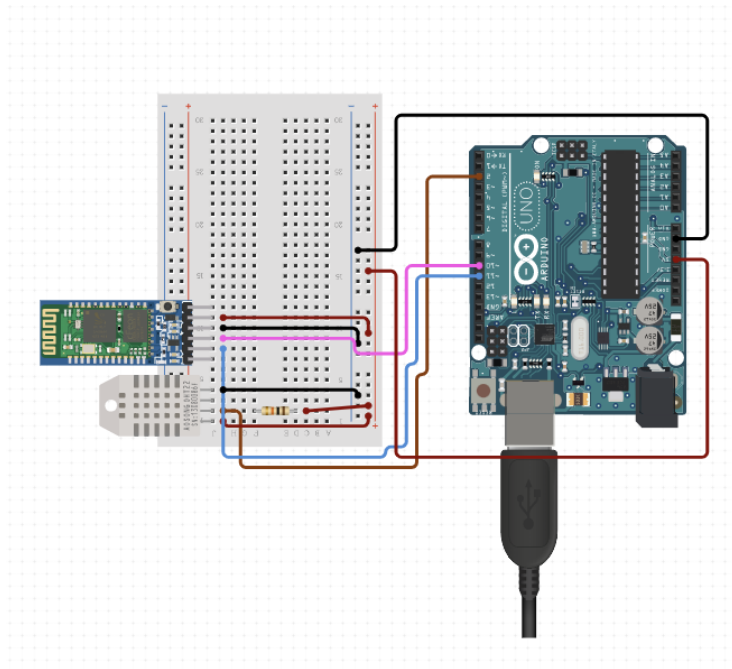
3.4. IoT 서비스

배송중인 수산물의 환경에 대한 온습도를 배송 기사가 측정하여 블록체인 원장에 기록하고 소비자가 온습도 정보를 언제든지 열람 가능하도록 한다.



홈페이지에서 배송기사 인증 버튼

수산물 경매 사이트는 배송 기사가 온습도를 측정할 수 있도록 배송 정보 관리 페이지를 운영하는데, 이 페이지는 블록체인에 배송기사로 등록된 사용자만이 이용 가능하다.



아두이노 배선도

위의 그림은 아두이노 나노와 블루투스 모듈, 온습도 센서의 배선도 이다. 온습도 센서를 이용해 온습도를 측정하고 측정된 값을 블루투스 모듈을 이용해 웹 페이지로 전송한다. 웹페이지는 온습도 값을 받은 즉시 블록체인에 기록한다.

블루투스 연결하기				배송할 목록들 보기
No.	생선품명	무게(kg)	개수	온습도 측정 시작 버튼
1	갈치	10	5	시작
2	고등어	5	10	시작
3	참돔	4	1	시작
4	가자미	12	5	시작
5	오징어	6	2	시작

배송 정보 관리 페이지

배송기사가 온습도를 측정하기 전 블루투스 연결하기 버튼을 통해 웹페이지와 아두이노를 연결한다. 그 다음 “배송할 목록들 보기” 버튼을 클릭하여 배송기사에게 할당된 배송 목록들을 출력하고 각 목록들마다 온습도 측정하기 시작 버튼을 눌러 온습도 값을 블록체인에 기록하기 시작한다.

3.5. QR 코드

No.	생선품명	경로		QR코드	온습도 정보
		from	to		
1	갈치	서울	부산	QR코드	보기
2	갈치	자갈치시장	부산양정	QR코드	보기
		자갈치시장장장	서울서초구		

QR코드 생성버튼

배송 추적 테이블에서 QR코드 버튼을 클릭하면 QR코드 페이지가 생성된다.

경로 확인 QR코드



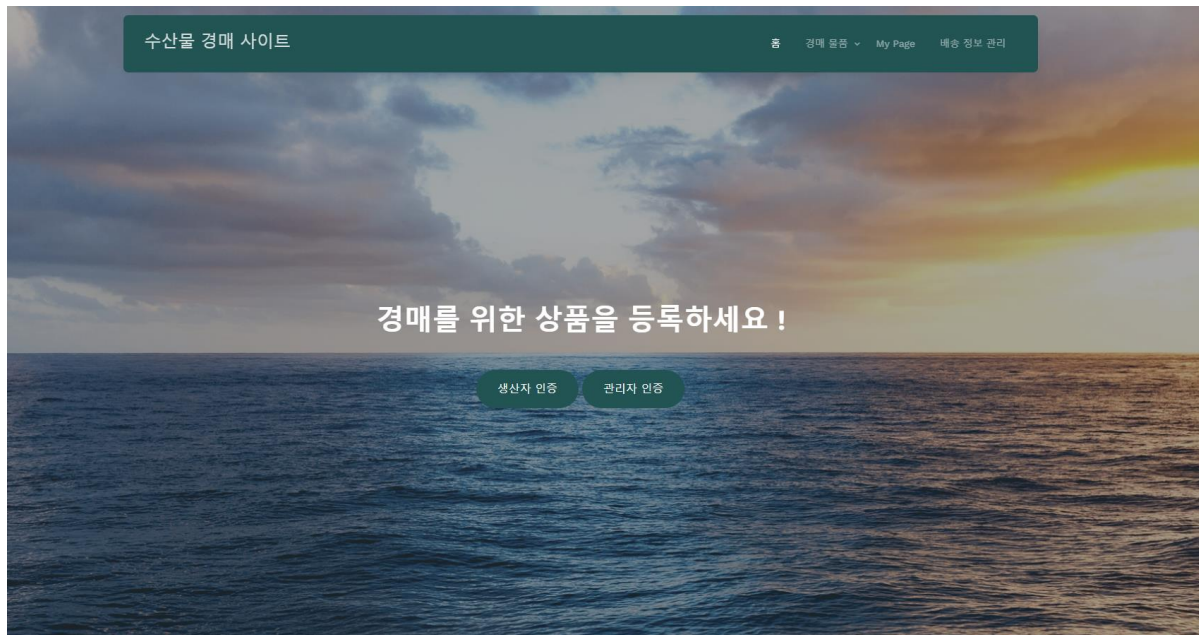
인쇄하기

낙찰한물품의 경로정보를 담고 있는 QR코드

생성된 QR코드를 통해 낙찰한 물품의 배송 정보를 확인할 수 있다.

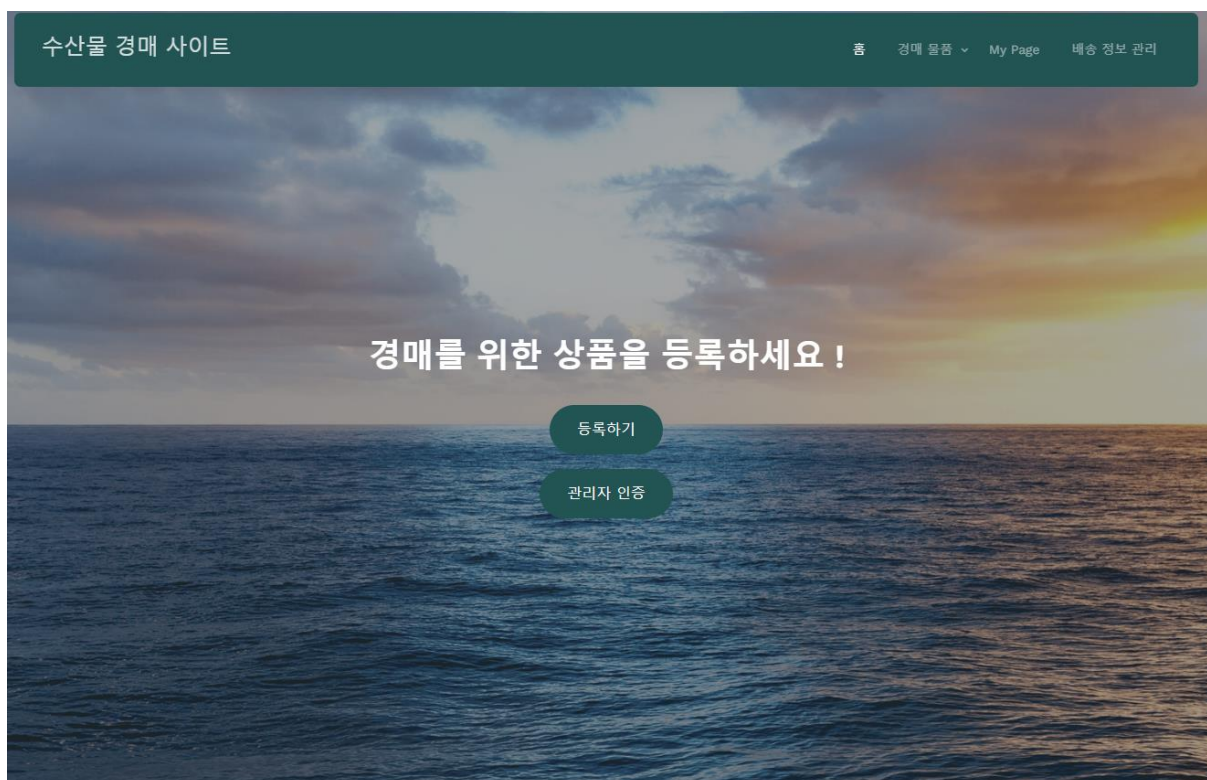
모바일로 QR코드를 스캔하면 해당 물품의 배송 정보를 담은 페이지로 이동한다.

4. 연구 결과 분석 및 평가

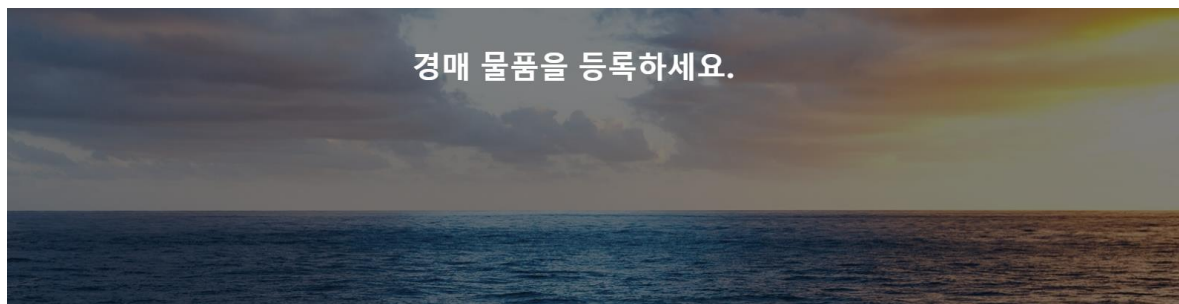


수산물 경매 사이트 첫 화면

두개의 인증 버튼을 통해 생산자와 관리자를 인증할 수 있다. 아래의 그림을 보면 생산자 등록이 완료되어 물품을 등록할 수 있는 등록하기 버튼이 보여진다.



생산자 등록이 완료된 상태



파일 선택

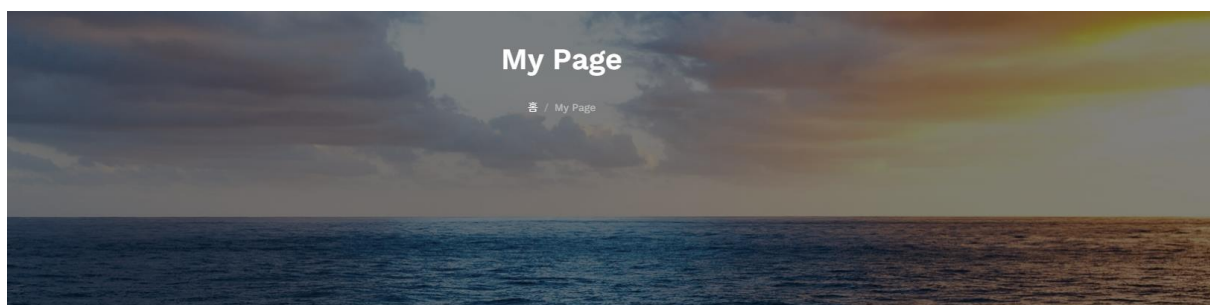
선택된 파일 없음

250px-Scomber_japonicus_(Matsuwababa).jpg

등록하기

경매 물품 등록하는 화면

위의 그림은 경매 물품을 등록하는 화면이다. 물품의 이름, 수량, 개수, 원산지와 물품의 이미지를 함께 첨부하여 등록할 수 있다.



경매대에 물품 등록
 등록하신 물품을 경매대로 올리게 되어, 소비자가 물품을 구매할 수 있게 됩니다.

낙찰된 내역 확인
 블라인드 경매에 참여하셔서 물품을 응찰하셨다면, 낙찰된 물품을 확인할 수 있습니다.

배송추적
 구매하신 물품의 배송추적이 가능합니다.

No.	생선품명	무게(kg)	개수	원산지	등록
1	고등어	10	10	동해	등록

등록한 물품 My Page 에서 확인

성공적으로 경매 물품을 등록한 후 My page의 “왼쪽 경매대의 물품 등록” 버튼을 클릭하면 등록한 경매 물품을 확인 할 수 있다.

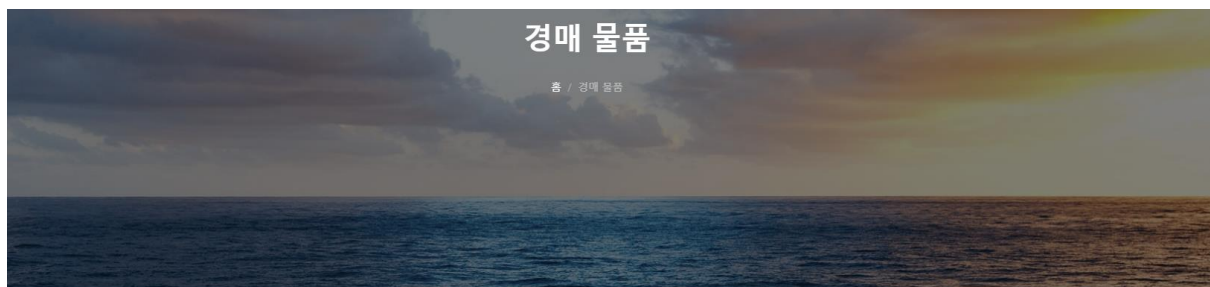
고등어	1	2022-09-30 오전 08:36:00
동해	2022-09-30 오전 08:39:03	

등록

취소

경매대에 올릴 물품을 등록


경매대에 물품을 등록하기 위해선 물품의 품명, 입찰 시작 가격, 주소, 경매 시작 시간, 경매 마감 시간 을 입력해야 한다.



경매대에 등록된 물품

경매대에 물품이 등록된 모습을 볼 수 있다. 이 페이지에서는 진행 중인 경매들을 리스트 형식으로 나열하여 보여준다.

입찰에 참여하세요.



고등어

원산지 : 동해

무게 : 10 개수 : 10

입찰하시겠습니까 ?

초기 가격 : 1 Wei

시작 시간 : 2022년 9월 30일 8시 36분 0초

마감 시간 : 2022년 9월 30일 8시 39분 3초

입찰 참여

경매 물품에 대한 상세정보

전체 경매 물품에서 세부적인 물품 정보를 확인 가능하다. 위의 페이지를 보면 물품의 대한 정보와 입찰 시작 시간과 마감 시간, 현재 입찰 가격을 알 수 있고, 물품을 등록한 생산자를 제외한 모든 사용자는 입찰에 참여할 수 있다.

또한 경매가 끝나도 상품과 거래 정보는 사라지지 않고 기록된다.

물품의 경우, 구매하면 해당 물품을 삭제하지 않고 소유권만 이전시켜 재경매가 가능하도록 구현하였기에 추적에 용이하고 여러 번의 재사용이 가능하다.

이는 이더리움 스토리지의 남용을 줄이고 생산자들의 활발한 경매가 이루어지는 효과를 기대할 수 있다.

최종적으로 낙찰 받은 사용자에게는 물품이 배송되는 경로와 배송 중일 때의 온도도 환경을 제공함으로써 물품의 신뢰성을 보장하며 이 또한 영구히 기록된다.

5. 결론 및 향후 연구 방향

본 졸업과제에서 기존 오프라인 수산물 경매 시스템의 낮은 투명성과 신뢰성을 극복하기 위해 이더리움 기반 수산물거래 플랫폼을 개발하였다.

이는 거래자입장에서 거래과정의 투명성을 보장받고, 소비자입장에서 유통과정의 투명성을 보장받아 수산물거래 및 유통에 대한 신뢰성을 준다. 이로 인해 거래시장의 활성화 효과를 불러올 수 있을 것이라 추측한다.

그러나 현 이더리움 스토리지는 느린 속도와 큰 경제적 비용이 요구되기에 대용량 정보를 처리하기에는 부적합하다.

따라서, 이더리움 스토리지를 대신에 **물품 정보를 JSON 형식의 메타 데이터로 기록해** IPFS에 업로드, 기반으로 토큰을 이용하면 많은 비용을 절감 할 수 있을 것이라고 판단되며 추후 추가 학습을 통해 진행할 예정이다.

6. 참고 문헌

- | |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>[1] 김효종, 한군희, 신승수 “이더리움 기반의 이더를 사용한 법원 경매 시스템에 관한 연구” , Journal of Convergence for Information Technology, Vol. 11, No. 2, pp. 31-40, 2021</p> <p>[2] bootstrap, “bootstrap” [Online]. Available: https://getbootstrap.com/</p> <p>[3] Web Bluetooth API. “MDN web docs” [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Web_Bluetooth_API</p> <p>[4] Ethereum, “Ethereum.org” [Online]. Available: https://ethereum.org/en/</p> <p>[5] MetaMask Documentation, Available: https://docs.metamask.io/guide/</p> <p>[6] web3.js Documentation, Available : https://web3js.readthedocs.io/en/v1.8.0/#</p> <p>[7] IPFS Documentation, Available : https://docs.ipfs.tech/</p> |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|