

## 목차

<b>1</b>	<b>요구 조건 및 제약 사항 분석 .....</b>	<b>2</b>
1.1	과제 목표 및 요구조건 .....	2
1.2	제약사항 분석 및 수정사항 .....	3
<b>2</b>	<b>설계 상세화 및 변경 내역 .....</b>	<b>4</b>
2.1	블록체인 프라이빗 네트워크 .....	4
2.2	스마트 컨트랙트 .....	5
2.3	웹페이지 .....	7
<b>3</b>	<b>과제 추진 계획 .....</b>	<b>8</b>
3.1	현재 과제 계획 .....	8
3.2	향후 과제 계획 .....	8
<b>4</b>	<b>과제 진행 내용 .....</b>	<b>9</b>
4.1	구성원별 진척도 .....	9
4.2	보고 시점까지의 과제 수행 내용 및 중간 결과 .....	10
<b>5</b>	<b>향후 계획 .....</b>	<b>19</b>

# 1 요구 조건 및 제약 사항 분석

## 1.1 과제 목표 및 요구조건

### 1.1.1 과제 목표

본 졸업과제는 블록체인 기반 수산물 유통 플랫폼 개발을 목표로 한다.

플랫폼에서 발생한 거래 내역은 원장에 기록하여 관리한다.

생산자와 도매상 간의 거래의 경우 국내 수산물 유통 시장의 특성을 감안해 경매 시스템을 도입할 예정이며 입찰과정과 낙찰 이후엔 거래 성사까지 진행된 입찰 내역을 모두 블록체인 원장에 기록하는 방식이다.

상품의 신선도 유지가 중요한 수산물인 만큼 IoT를 활용해 수집한 배송 환경(온/습도)과 발송/수송지를 주기적으로 원장에 기록, 제품의 배송과정 확인과 차후 추적에 용이하도록 관리한다.

생산자 계정으로 상품을 등록할 때 생산자 정보, 상품의 크기와 신선도, 산지 등을 원장에 기록하여 물품을 직접 확인하지 못하는 전산거래의 한계를 극복하고자 하며

이는 차후 오배송이나 허위매물 등 배송에 문제가 발생하였을 때 생산자에게 책임을 물을 수 있는 근거로 활용한다.

최종 소비자가 구매한 수산물의 유통과정을 손쉽게 확인 가능하도록 QR코드를 발급할 예정이며 유통경로를 기록한 링크를 QR코드에 담아 이를 카메라로 인식 시 해당 링크로 이동해 확인하는 방식이다.

### 1.1.2 요구조건

블록체인 네트워크	<ul style="list-style-type: none"><li>· 실제 퍼블릭 네트워크와 유사한 멀티노드 환경</li><li>· 안정성 높은 통신 및 동기화</li><li>· 탈중앙화 네트워크 구조</li></ul>
스마트 컨트랙트	<ul style="list-style-type: none"><li>· 기능별 스마트 컨트랙트 구현</li><li>· 과도한 gas 소모를 유발하는 데이터 접근 자제</li></ul>
웹 서비스 플랫폼	<ul style="list-style-type: none"><li>· 사용자 접근 편의성 제공을 위한 웹 서비스</li><li>· 사용자 별 블록체인 권한 기반 멤버십 서비스 제공</li><li>· Bootstrap 라이브러리 블록체인 활용한 front-end 및 Node.js 와 Express.js 기반의 back-end 구축</li><li>· 사용자 권한 별 리소스 연계</li></ul>

## 1.2 제약사항 분석 및 수정사항

블록 체인 노드	제약사항	<ul style="list-style-type: none"> <li>· 이더리움 로컬 클라이언트의 노드 간 통신이 불안정해 블록 동기화가 진행되지 않음</li> <li>· 로컬 네트워크에 참여한 노드들의 블록이 일정치 않아 결과적으로 트랜잭션(정보)의 손실이 발생</li> </ul>
	수정사항	<ul style="list-style-type: none"> <li>· 모든 노드들을 하나의 클라이언트(PC)에 배치</li> <li>· 불안정한 Peer search 기능을 제하고 static-nodes에 peer 노드들의 주소를 기록, 부팅과 동시에 기록된 노드들을 자동으로 연결</li> <li>· 특정 노드에 집중되었던 네트워크 구조를 분산시켜 탈중앙화 네트워크로 개편</li> </ul>
스마트 컨트랙트	제약사항	<ul style="list-style-type: none"> <li>· 컨트랙트 당 코드 용량이 제한적</li> <li>· 외부에서 컨트랙트 내 데이터 접근이 제한적</li> <li>· 데이터 접근 시 재화(gas)가 요구됨</li> <li>· 기존에 사용하던 웹 IDE(Remix)의 컴파일 및 배포 성능이 낮아 개발 진행에 차질을 줌</li> </ul>
	수정사항	<ul style="list-style-type: none"> <li>· 기존 데이터를 수정하는 함수들은 접근하는 데이터와 동일한 컨트랙트에 배치</li> <li>· 용량이 작은 자료형(uint256)을 통해 데이터 전달 및 접근</li> <li>· 컴파일 및 배포 안정성이 높은 로컬 프레임워크로 변경</li> </ul>

## 2 설계 상세화 및 변경 내역

### 2.1 블록체인 프라이빗 네트워크

#### 2.1.1 구조

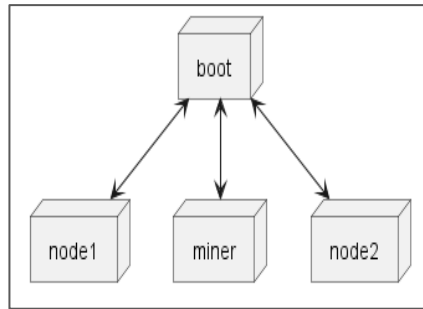


그림 1 변경 전

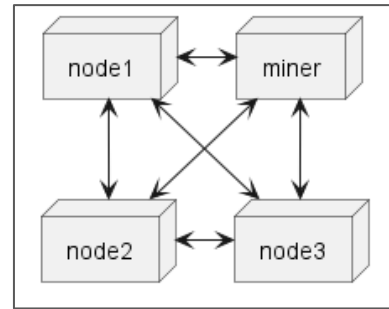


그림 2 변경 후

변경 전	Boot node를 중심으로 다른 노드들과 연결 및 동기화
변경 후	중심 노드를 두지 않고 모든 노드 간 연결 및 동기화

#### 2.1.2 연결 방식

변경 전	<ul style="list-style-type: none"> <li>• Docker로 분리한 가상공간에서 멀티노드를 구성 후 연결</li> <li>• 네트워크에서 자동으로 동일한 genesis_block을 가진 peer 탐색</li> </ul>
변경 후	<ul style="list-style-type: none"> <li>• 동일한 클라이언트에서 멀티노드를 구성 후 연결</li> <li>• --nodiscover 옵션으로 peer 탐색을 막고 static-nodes로 항상 연결</li> </ul>

## 2.2 스마트 컨트랙트

### 2.2.1 컴파일 및 배포

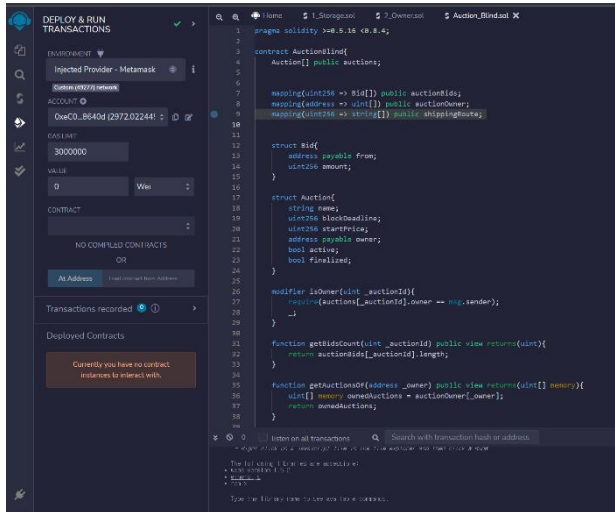


그림 3 변경 전 (Remix)

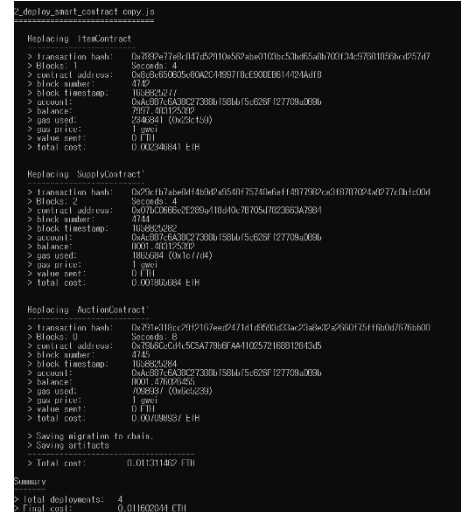


그림 4 변경 후 (Truffle)

변경 전	Remix로 compile 및 배포
변경 후	Truffle로 compile 및 배포

### 2.2.2 구성 요소

변경 전	<ul style="list-style-type: none"> <li>Auction, Item, Supply Contract로 구성</li> </ul>	
변경 후	<ul style="list-style-type: none"> <li>상품 별 배송 데이터를 관리하는 rwSupplyData Contract 추가</li> </ul>	
	Item	<ul style="list-style-type: none"> <li>상품 객체 생성 및 저장</li> </ul>
	Auction	<ul style="list-style-type: none"> <li>경매 객체 생성 및 저장</li> <li>재화 및 상품의 이동을 기록 및 반영</li> </ul>
	Supply	<ul style="list-style-type: none"> <li>상품의 운송 정보를 저장</li> <li>유통 이력 추적 기능 제공</li> </ul>
	rwSupplyData	<ul style="list-style-type: none"> <li>거래에 맞춰 상품별 운송 정보를 수정 및 관리</li> </ul>

## 2.2.3 구조

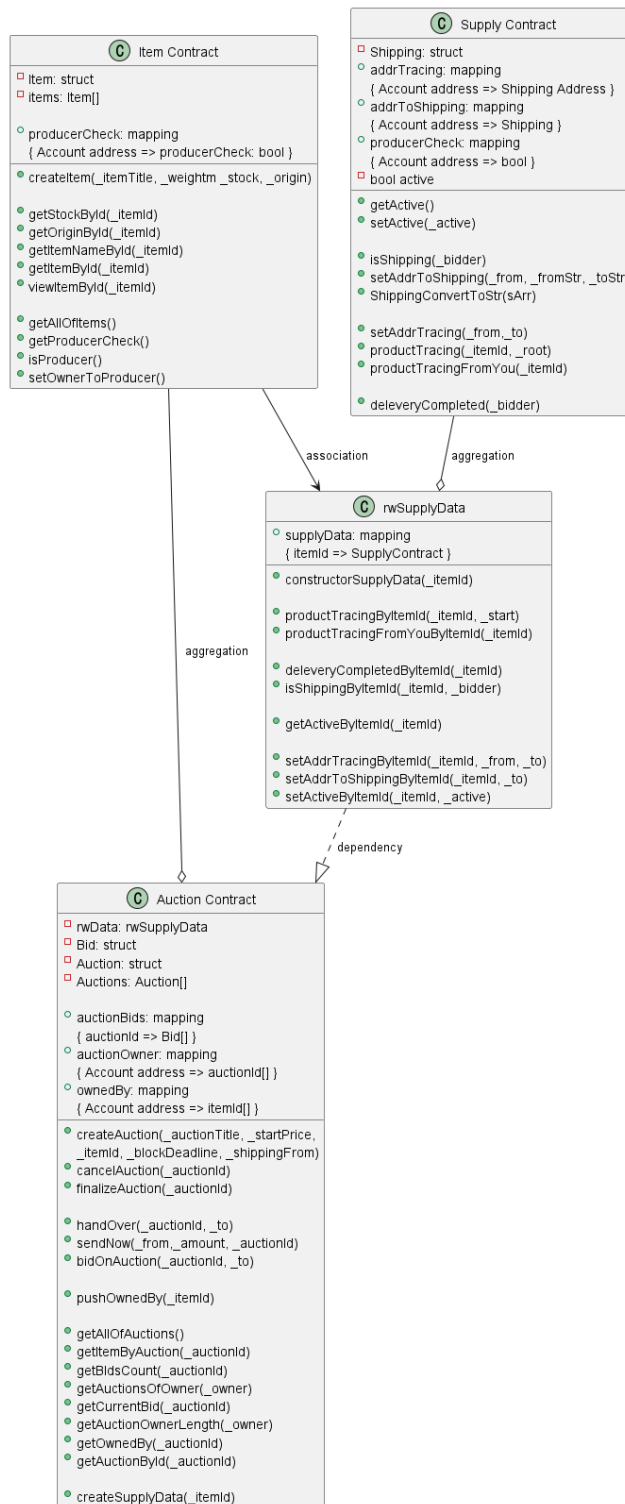


그림 5 스마트 컨트랙트 다이어그램

## 2.3 웹페이지

### 2.3.1 개발 환경

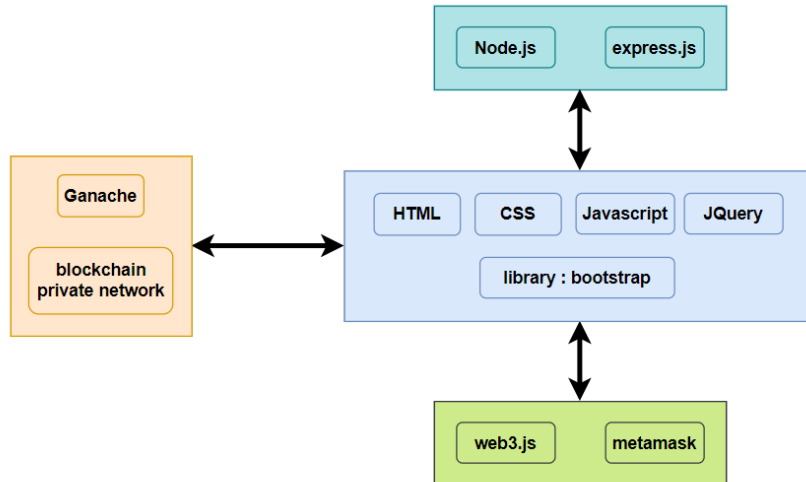


그림 6 개발 환경

### 2.3.2 구성요소

페이지	설명
생산자 등록 페이지	생산자 등록을 완료해야만 등록하기 버튼 생성 생산자 등록을 하지못하면 소비자로 분류되어 소비자는 경매입찰과 My page 에서 낙찰된 내역, 배송추적 가능
경매 등록 페이지	생산자는 경매에 올릴 상품이 여러 개 일 수 있음. 따라서, 경매대에 올리기 전 상품을 사용자가 원하는 만큼 등록한 후, my page 의 <경매대에 물품 등록> 버튼을 누르면 등록된 상품들이 테이블로 출력되어 보여지게 됨
경매 물품 페이지	웹페이지 상단의 <properties>버튼을 누르게 되면, 생산자가 경매대에 등록해놓은 상품이 전부 보여지게 됨

### 3 과제 추진 계획

#### 3.1 현재 과제 계획

5월			6월					7월				
3주	4주	5주	1주	2주	3주	4주	5주	1주	2주	3주	4주	5주
블록체인 스터디												
	블록체인 네트워크 구축											
						smart contract 설계						
						UI 디자인 및 기능 개발 구현						
											중간보고서	

그림 7 과제 진행 상황

#### 3.2 향후 과제 계획

8월					9월			
1주	2주	3주	4주	5주	1주	2주	3주	4주
smart contract와 웹페이지 연동 테스트								
IoT서비스 개발 및 qr 코드 개발								
			테스트 및 디버깅					
					최종 보고서 작성			
						포스터 작성		

그림 8 향후 계획



## 4 과제 진행 내용

### 4.1 구성원별 진척도

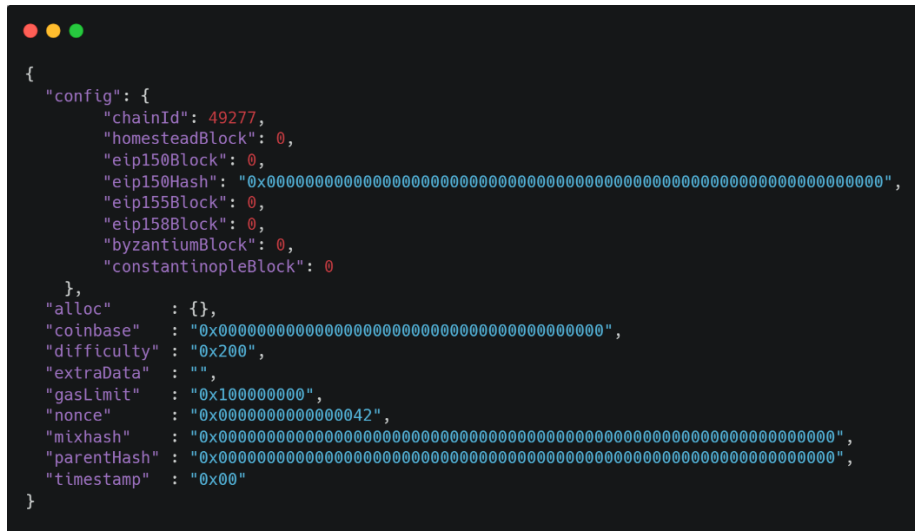
- ☒ : 개발 완료    ☐ : 미완성

류지환	<ul style="list-style-type: none"><li>· <input checked="" type="checkbox"/> 이더리움 로컬 네트워크 구축</li><li>· <input checked="" type="checkbox"/> 스마트 컨트랙트 개발 완료</li><li>· <input type="checkbox"/> Web3.js 와 IPFS 연동</li><li>· <input type="checkbox"/> 사용자와 스마트 컨트랙트 간 이더 송금 연동</li></ul>
엄혜림	<ul style="list-style-type: none"><li>· <input checked="" type="checkbox"/> 소유자, 생산자 구분 서비스 개발</li><li>· <input checked="" type="checkbox"/> 블록체인 기반 수산물 경매 사이트 구현</li><li>· <input checked="" type="checkbox"/> Web3.js 와 MetaMask 연동</li><li>· <input checked="" type="checkbox"/> 스마트 컨트랙트와 웹페이지 연동</li><li>· <input type="checkbox"/> 배송지 경로를 담고 있는 QR code 개발</li></ul>
박서현	<ul style="list-style-type: none"><li>· <input checked="" type="checkbox"/> Bootstrap 라이브러리 템플릿 선정</li><li>· <input checked="" type="checkbox"/> 블록체인 기반 수산물 경매 사이트 디자인 및 구현</li><li>· <input checked="" type="checkbox"/> My page, 총 경매 물품 내역 출력의 서비스 개발</li><li>· <input checked="" type="checkbox"/> Web3.js 와 MetaMask 연동</li><li>· <input type="checkbox"/> IoT 서비스 개발</li></ul>

## 4.2 보고 시점까지의 과제 수행 내용 및 중간 결과

### 4.2.1 이더리움 로컬 네트워크

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



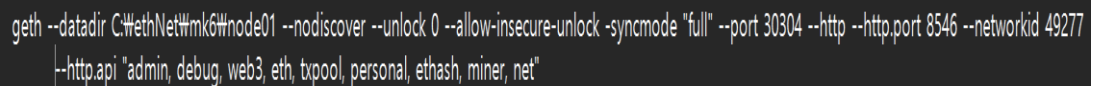
```
{
  "config": {
    "chainId": 49277,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip150Hash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0
  },
  "alloc" : {},
  "coinbase"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

그림 9 genesis.json

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

#### 2. geth로 노드 실행

옵션: port, networkid, peer 탐색 중지, 계정 unlock, unlock 권한 부여, api



```
geth --datadir C:\ethNet\mk6\node01 --nodiscover --unlock 0 --allow-insecure-unlock -syncmode "full" --port 30304 --http --http.port 8546 --networkid 49277
--http.api "admin, debug, web3, eth, txpool, personal, ethash, miner, net"
```

그림 10 노드 실행 커맨드

### 3. geth javascript console을 통해 노드 조작

http 포트: node1-8546/node2-8547/miner-8538/node3-8549

```
C:\Windows\System32\cmd.exe - geth attach http://localhost:8546
Microsoft Windows [Version 10.0.22000.795]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>geth attach http://localhost:8546
Welcome to the Geth JavaScript console!

instance: Geth/v1.10.18-stable-de23cf91/windows-amd64/go1.18.1
coinbase: 0xac8b7c6a3bc273bbb58bbf5c626ff27709adb9b
last block: 8289 (Wed Jul 27 2022 05:36:50 GMT+0900 (KST))
datadir: C:\Metamask5\node01
modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
> _
```

그림 11 geth js console

### 4. 채굴을 통해 트랜잭션을 반영하여 로컬 네트워크 가동

채굴노드에서 생성한 블록을 연결된 노드가 import

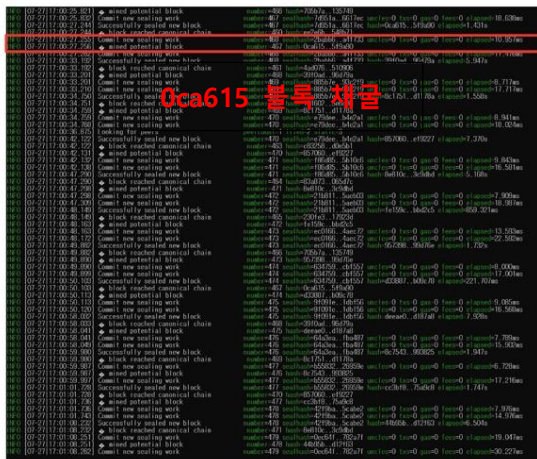


그림 12 채굴노드

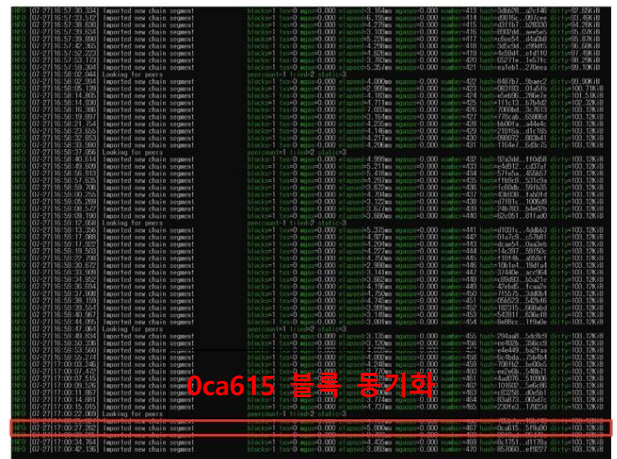


그림 13 연동

## 4.2.2 스마트 컨트랙트

```
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;
    }
}
```

그림 14 Auction Contract

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

```
truffle.exports = {
  //
  // Networks define how you connect to your ethereum client and let you set the
  // defaults (e.g. which client, port) on a per network basis. You can also
  // specify the network id, e.g.
  //
  // run: truffle test --network test
  //
  // truffle test --network test
  //
  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: '4222', // Any network (default: none)
    },
  },
};
```

그림 15 Truffle\_config.js

```
truffle.deployments = {
  //
  // Networks define how you connect to your ethereum client and let you set the
  // defaults (e.g. which client, port) on a per network basis. You can also
  // specify the network id, e.g.
  //
  // run: truffle test --network test
  //
  // truffle test --network test
  //
  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: '4222', // Any network (default: none)
    },
  },
};
```

그림 16 Truffle deploy.js

## 2. Truffle 설정 및 migrate 코드 작성

```
C:\Web3\truffle>truffle migrate --reset

Compiling your contracts...
> Compiling ./contracts/Auction.sol
> Compiling ./contracts/Item.sol
> Compiling ./contracts/Supply.sol
> Compiling ./contracts/Migrations.sol
> Compiling ./contracts/BuildContracts.sol
> Artifacts written to C:\Web3\truffle\build\contracts
> Compiled successfully using:
   - solc: 0.8.15+commit.e142714.Emscripten.clang

Starting migrations...
> Network name:    'development'
> Network id:     43272
> Block gas limit: 42012848 (0x2811040)

1. initial_migration.js

Replacing 'Migrations'
> transaction hash: 0x620f0647503e9033e155847b4705e00c303b77208bce161717b6e5078
> blocks: 0
> contract address: 0x525000a9561810000c20732a30305a11a05f
> block number: 4758
> block timestamp: 165825254
> account: 0x6b07c0a9c27308b158baf5c20f127709a09b
> balance: 7891.48125382
> gas used: 230542 (0x36116)
> gas price: 1 gwei
> value sent: 0 ETH
> total cost: 0.000230542 ETH

> Saving migration to chain.
> Saving artifacts
=====
> Total cost: 0.000230542 ETH

2. deploy_smart_contract copy.js

Replacing 'ItemContract'
> transaction hash: 0x7850e77ebc847652910e562abe0103bc53bd5a8b70134c97881856acd51d7
> blocks: 1
> contract address: 0x6b07c0a9c27308b158baf5c20f127709a09b
> block number: 4759
> block timestamp: 165825277
> account: 0x6b07c0a9c27308b158baf5c20f127709a09b
> balance: 7891.48125382
> gas used: 2348841 (0x23c150)
> gas price: 1 gwei
> value sent: 0 ETH
> total cost: 0.002348841 ETH
```

그림 17 Truffle migrate

## 3. Truffle console을 통한 컴파일 및 배포

## 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);
}
```

그림 18 web3 연동

#### 4.2.3 웹페이지

##### 1. Web3와 metamask

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

그림 19 metamask, 스마트 컨트랙트 연동

##### 2. 첫 화면에서 생산자를 인증해야만 상품을 등록할 수 있는 버튼이 활성화

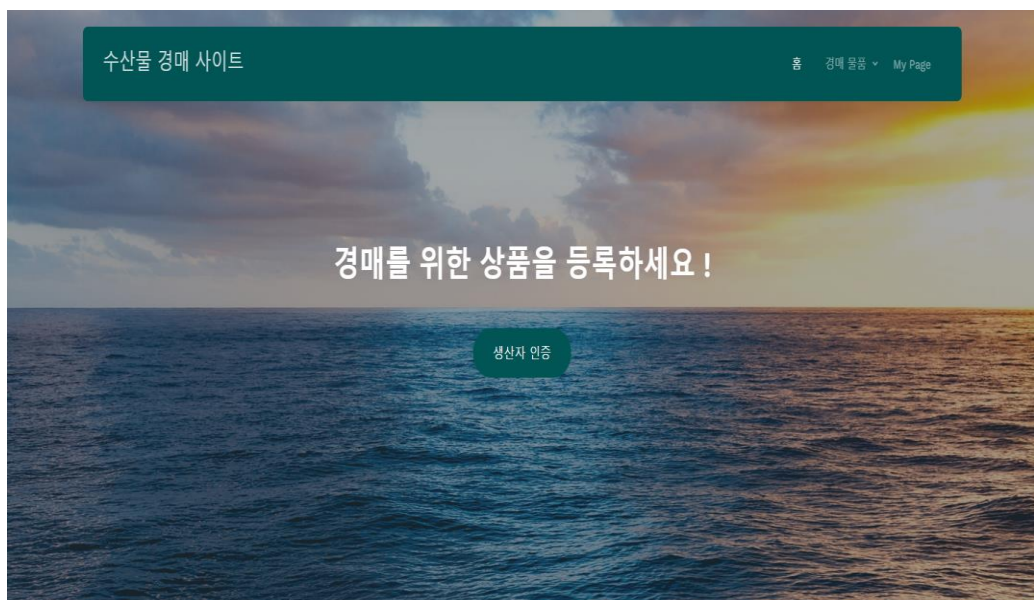


그림 20 첫 화면

##### 3. 생산자 등록 함수 호출로 트랜잭션 발생, 해당 계정을 생산자로 등록할 수 있음



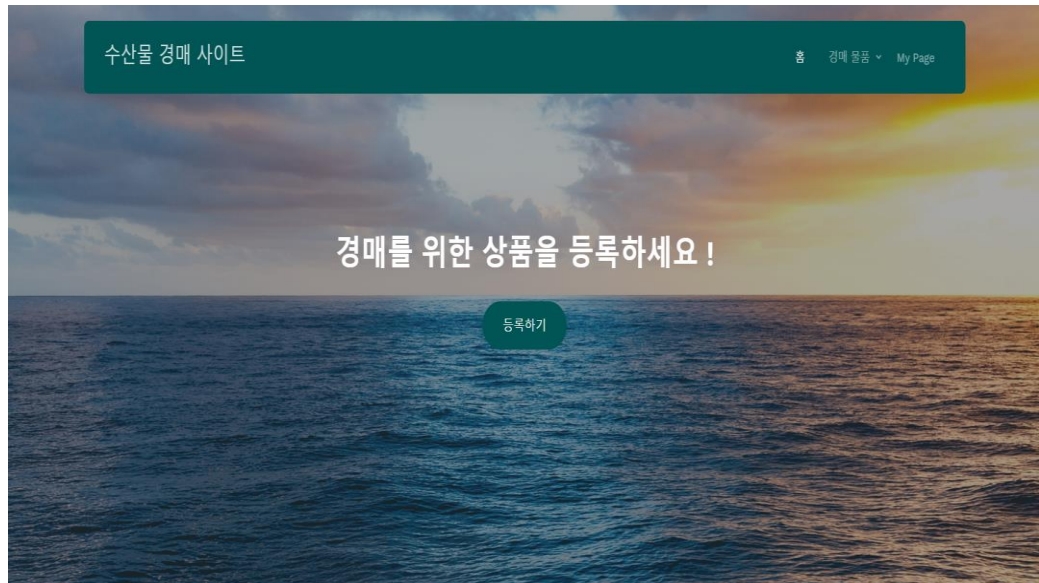


그림 21 생산자 등록 후



그림 22 생산자 등록 트랜잭션

4. 상품 등록 함수 호출로 트랜잭션 발생, 상품이 이더리움 로컬 네트워크에 저장

그림 23 상품 생성 화면

```

properties_register.html:305
{transactionHash: '0x690af719edd74f87d795e5d01aeaa5097219f3a96599d10c24b05640920
7928', transactionIndex: 0, blockHash: '0x72ab0987da677edd1f3142fa40b8172c8de0434
ec96879bac6a26b47d81c3cdd', blockNumber: 113, from: '0x8ac1bcc1aa016d64478a12150e
cdc5d4a609d348', ...}
  blockHash: "0x72ab0987da677edd1f3142fa40b8172c8de0434ec96879bac6a26b47d81c3cdd"
  blockNumber: 113
  contractAddress: null
  cumulativeGasUsed: 201666
  events: {CreateItemEvent: {...}}
  from: "0x8ac1bcc1aa016d64478a12150ecdc5d4a609d348"
  gasUsed: 201666
  logsBloom: "0x0000000000000000000000000000000000000000000000000000000000000000"
  status: true
  to: "0xb462e04fdb3882ccabccce83fb17eeaa945875807"
  transactionHash: "0x690af719edd74f87d795e5d01aeaa5097219f3a96599d10c24b05640920"
  transactionIndex: 0
  [[Prototype]]: Object
아이템 아이디 ▶ ['0']
properties_register.html:306

```

그림 24 상품 등록 트랜잭션

- 본 계정이 소유한 상품 리스트를 호출, 불러온 상품들을 테이블 형식으로 출력



**경매대에 물품 등록**

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

No.	생선품명	무게(kg)	갯수	원산지	등록
1	갈치	200	80	제주	등록

**낙찰된 내역 확인**

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

**배송추적**

구매하신 물품의 배송추적이 가능합니다.

그림 25 경매 등록 가능한 상품 목록과 등록 버튼

6. 경매 생성 함수 호출로 트랜잭션 발생, 경매가 이더리움 로컬 네트워크에 저장

생선 품명 (ex. 고등어, 갈치 등등 .. )

시작가격

-- --:--:--

주소

등록

취소

그림 26 상품을 등록할 경매 생성

	my_page.html:268
<pre> {transactionHash: '0xdb159e4b51171612f7c5e82653d0976ad78a5415d85e53a773990e41a 12b6515', transactionIndex: 0, blockHash: '0x7743957b99af7d5663bd63b398e83a7b4 bd02576ffe01e3ddf5eb1792f0047ec', blockNumber: 114, from: '0x8ac1bcc1aa016d644 78a12150ecdc5d4a609d348', ...}   blockHash: "0x7743957b99af7d5663bd63b398e83a7b4bd02576ffe01e3ddf5eb1792f0047e   blockNumber: 114   contractAddress: null   cumulativeGasUsed: 178685   events: {AuctionCreated: {...}}   from: "0x8ac1bcc1aa016d64478a12150ecdc5d4a609d348"   gasUsed: 178685   logsBloom: "0x00   status: true   to: "0x8deadb5256b2a1c7fc32973742fd780fa706c60e"   transactionHash: "0xdb159e4b51171612f7c5e82653d0976ad78a5415d85e53a773990e41a   transactionIndex: 0   [[Prototype]]: Object </pre>	
1 마이디	my_page.html:274
>	

그림 27 경매 생성 트랜잭션

7. 사전에 생성한 상품들을 포함한 경매 등록 가능

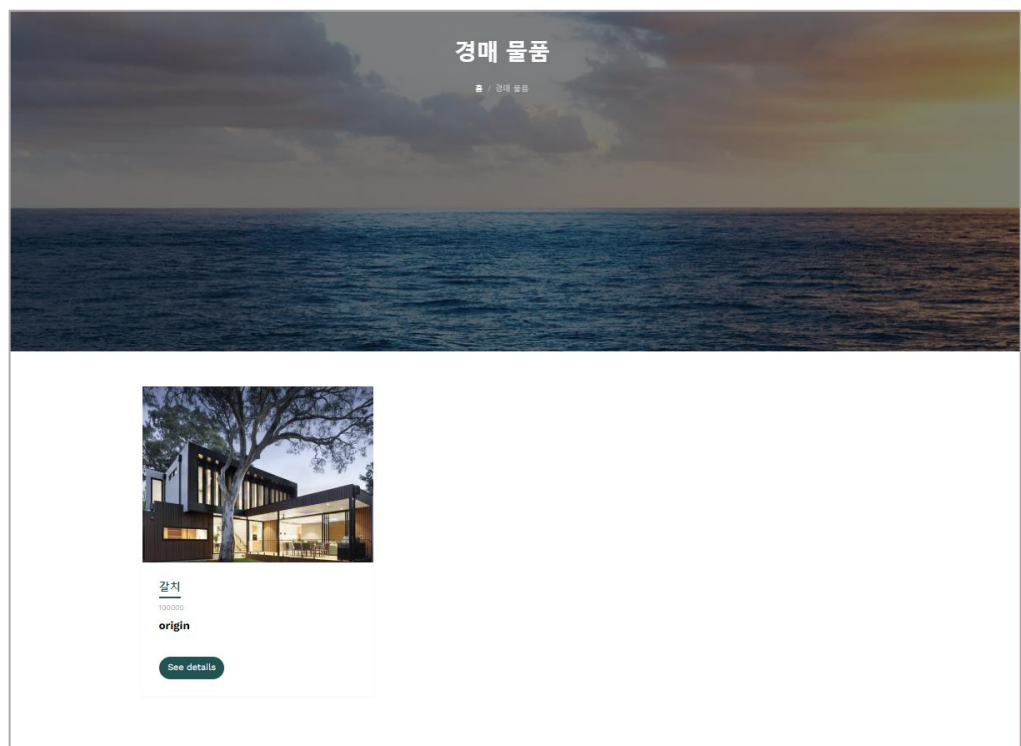


그림 28 진행 중인 경매 목록

## 5 향후 계획

- IPFS와 연동해 이미지 저장 및 노출
- 웹 IoT 센서와 스마트 컨트랙트 연동
- 사용자와 스마트 컨트랙트 간 이더 송금 동기화
- 웹과 Supply 컨트랙트 연동
- 배송지 경로 담고 있는 QR code 개발