

# 2022전기 중간보고서

스마트 컨트랙트 기반 농산물 포전거래 플랫폼



**부산대학교**  
PUSAN NATIONAL UNIVERSITY

제출일	2022.07.29	전공	정보컴퓨터공학부
		담당교수	김호원 교수님
학번	201724435		김재현
	201824549	이름	이세진
	201924500		신예주

# <목 차>

1. 요구조건 및 제약 사항 분석에 대한 수정사항
  - 1.1 요구조건
  - 1.2 제약 사항 분석에 대한 수정사항
    - 1.2.1 기존 제약사항에 대한 수정사항
    - 1.2.2 추가 제약사항 및 대책
2. 설계 상세화 및 변경 내역
3. 갱신된 과제 추진 계획
4. 구성원 별 진척도
5. 보고 시점까지의 과제 수행 내용 및 중간 결과
  - 5.1 DApp 부동산 거래 플랫폼 구현
  - 5.2 AWS 서버 구축
  - 5.3 SQL을 활용한 데이터베이스 구축
  - 5.4 앱 UI 구성
  - 5.5 하이퍼레저 패브릭

## 1. 요구조건 및 제약사항 분석에 대한 수정사항

### 1.1 과제 목표 및 요구조건

#### 1) 과제 목표

스마트 컨트랙트를 기반으로 농산물을 포전거래 함으로써 불법계약 방지, 산지 유통인 주도의 가격 결정 방지할 수 있는 플랫폼 개발

- 블록체인 네트워크를 구축하고 체인코드를 기반으로 농작물/ 토큰 거래와 같은 기능을 제공함
- 사용자들은 DApp을 통해 블록체인의 리소스에 접근하여 포전거래 상품을 조회 또는 업로드하거나, 토큰을 기반으로 농작물을 구매하는 등 시스템을 이용함

#### 2) 요구조건

- 블록체인 네트워크
  - peer는 크게 4개로 구분됨 (농부, 유통인, 관리자, 기관)
  - 1개의 메인 채널 및 3개의 채널 ( 인증, 등록 , 거래 )로 구성됨
- 농산물/토큰 거래
  - 각 채널 별 Chaincode 기반으로 기능 구현 (Golang)
  - 각 Chaincode 별 endorsement policy 수립
- 앱 서비스 플랫폼
  - 사용자 접근 편의성 제공을 위한 앱 서비스
  - 사용자 별 ID/PW/블록체인 권한 기반 멤버십 서비스 제공
  - 사용자 권한 별 블록체인 리소스 연계
  - app : Flutter 프레임워크를 활용한 android/ios 개발
  - web : React.js 기반의 frond-end 및 Node.js 기반의 back-end 구축

### 1.2 제약사항 분석 및 수정사항

블록체인 노드	제약사항	하이퍼레저 패브릭은 permissioned 블록체인이다. 따라서 허가된 참여자만 접근을 허용하고 접근 권한을 제한한다.
	해결방안	이용자들이 나쁜 의도를 갖지 않고 선별된 1개의 검증기관이 있다는 가정 하에 구현.
토큰	제약사항	토큰의 경우 리플과 같이 초기에 발행함. 개수가 유한하기 때문에 초기에 고려했던 판매 금액에 비해 가치가 달라질 수 있음.
	해결방안	본 과제에서는 인플레이션 정책 등은 모두 배제하되, 향후 스마트컨트랙트를 통해 업그레이드를 진행할 수 있도록 구현.

## 2. 설계 상세화 및 변경 내역

### 1) 참여 peer

농부	<ul style="list-style-type: none"> <li>· 개인 농부가 대상임</li> <li>· 농산물을 조회하고 업로드 및 판매를 할 수 있음</li> <li>· 토큰을 통해 블록체인 상의 다른 유통자와 거래할 수 있음</li> </ul>
유통인	<ul style="list-style-type: none"> <li>· 등록된 산지유통인이 대상임</li> <li>· 농산물을 조회하고 구매를 할 수 있음</li> <li>· 토큰을 통해 블록체인 상의 다른 농부와 거래할 수 있음</li> </ul>
관리자 (운영자)	<ul style="list-style-type: none"> <li>· 블록을 생성하고 메인 네트워크를 관할하는 조직임</li> <li>· 블록체인 네트워크 및 전체 앱을 총괄/운영하는 조직임</li> <li>· 해당 조직에서는 블록을 생성할 때 트랜잭션 정렬을 위해 합의 과정이 필요하며, RAFT 알고리즘을 적용함</li> </ul>
기관	<ul style="list-style-type: none"> <li>· 관리자에 의해 선별된 검증된 조직임</li> <li>· 농작물과 인증된 사용자를 관리하고 산지유통인 검증을 수행하는 등 리소스를 제공</li> </ul>

### 2) 블록체인 네트워크 채널

main channel	토큰을 발행하고, 거래 데이터를 등록하는 채널로 농부, 관리자가 참여
농부-기관-관리자 Channel	농산물 등록 후 인증 절차를 위한 채널로 농부, 기관, 관리자가 참여
농부-유통인-관리자 Channel	농산물 거래를 위한 채널로 농부, 유통인, 관리자가 참여

### 3) Back-end

메인 서버를 API, 데이터베이스, Chaincode Modules, Libraries 등으로 구성하기로 상세화하였음.

API	· Front-end로부터 전달 받은 사용자의 입력에 따라 서비스 수행
데이터베이스	<ul style="list-style-type: none"> <li>· 회원가입 및 로그인 등과 관련된 기능을 로컬 데이터베이스를 통해 지원함</li> <li>· 농부들이 등록한 농작물에 대한 정보를 저장해둠</li> </ul>
Chaincode Modules	· 하이퍼레저 패브릭 리소스에 대한 요청에 따라 Fabric-Client를 통해 체인코드를 연계해서 제공함
Libraries	· Eslint, Typescript 등의 추가 라이브러리 등을 활용하여 코드에 대한 안정성, 유지보수의 편리성 등을 제공함

#### 4) WEB Front-end

React.js를 기반으로 구현하며 이용자 별로 다른 기능을 제공함.

운영자	<ul style="list-style-type: none"> <li>· 전체 네트워크 정보와 현황을 모니터링할 수 있는 관리용 대쉬보드</li> <li>· 블록체인 관리 툴 제공</li> </ul>
기관	<ul style="list-style-type: none"> <li>· 검증기관용 대쉬보드</li> <li>· 농작물 검증 관리 페이지</li> </ul>
사용자(농부&유통인)	<ul style="list-style-type: none"> <li>· 농작물을 조회하고 정보를 얻는 페이지</li> <li>· 등록된 농작물을 검색하고 구매 가능한 페이지</li> <li>· 토큰 거래 페이지</li> <li>· 사용자 개인 정보를 관리하는 페이지</li> </ul>

#### 5) 하이퍼레저 패브릭

체인코드	<ul style="list-style-type: none"> <li>· 계약 상세 내용을 반영하기 위해 실제로 사용되는 포전매매 표준계약서의 내용에서 변수를 추출하여 구현</li> <li>· 체인코드에는 포전 매매에 필요한 함수들을 구현</li> </ul>
------	--

### 3. 갱신된 과제 추진 계획

6월		7월					8월					9월				
4주	5주	1주	2주	3주	4주	5주	1주	2주	3주	4주	5주	1주	2주	3주	4주	5주
블록 체인 스터디 [부동산 Dapp 개발]																
		서버&DB 구축														
		앱 UI 디자인														
		하이퍼레저 패브릭 체인코드 개발														
				중간 보고서 작성												
				앱 퍼블리싱 및 개발												
							API 웹 서버 개발									
										rest API 앱 연동						
										테스트 및 디버깅						
												오류 수정				
														최종 발표 준비 및 보고서 제작		

#### 4. 구성원 별 진척도

이름	역할
이세진	<ul style="list-style-type: none"><li>- 앱 퍼블리싱 완료 및 상세 기능 구현 진행 중</li><li>- 앱 UI 정의 및 적용 완료</li></ul>
김재현	<ul style="list-style-type: none"><li>- 블록체인 네트워크 구축</li><li>- 스마트 컨트랙트(체인코드) 개발 진행 중</li></ul>
신예주	<ul style="list-style-type: none"><li>- AWS 서버 구축 및 원격 접속을 위한 SSH 서버 구축 완료</li><li>- 유저 데이터베이스 구축 및 AWS와 연결 완료</li></ul>

#### 5. 보고 시점까지의 과제 수행 내용 및 중간 결과

##### 5.1 DApp 부동산 거래 플랫폼 구현

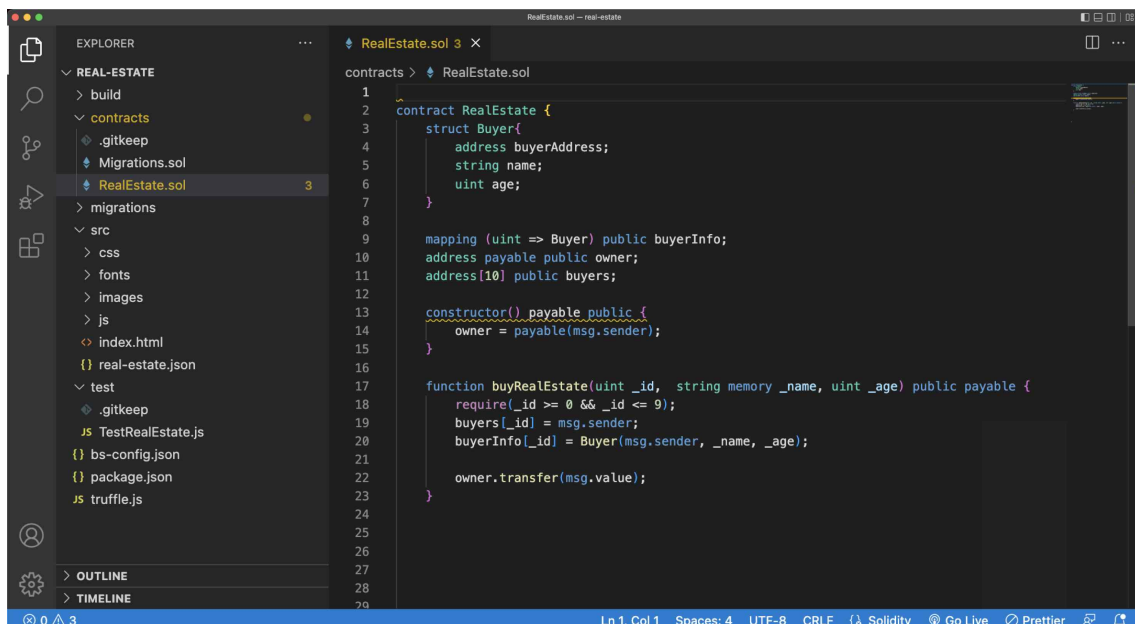
우선적으로, 블록체인 기반의 DApp의 원리와 제작에 대한 전반적인 학습을 위해 인프런에서 ‘블록체인 이더리움 부동산 DApp 만들기-기본편’을 수강함. Solidity를 통해 스마트 컨트랙트 및 분산 어플리케이션을 공부하며 이더리움 블록체인이 어떻게 작동하는지 배우고 프론트엔드와 연결시켜 분산 어플리케이션을 완성시킬 수 있었음. 스마트 컨트랙트 기술을 기반으로 웹 사이트를 구축하고, Web3 등을 통한 UI 업데이트와 배포까지 진행하며 쌓은 경험은 이후 우리의 과제를 진행하는 데 있어서도 많은 도움이 되었음.

```
<> index.html x
src > <> index.html > html > body > div.container > div.row
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 <meta name="viewport" content="width=device-width, initial-scale=1">
7 <!-- The above 3 meta tags *must* come first in the head; any other head content
8 <title>이더리움 부동산</title>
9
10 <!-- Bootstrap -->
11 <link href="css/bootstrap.min.css" rel="stylesheet">
12 <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries
13 <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
14 <!--[if lt IE 9]>
15 <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>
16 <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
17 <![endif]-->
18 </head>
19 <body>
20 <div class="container">
21 <div class="row">
22 <div class="col-xs-12 col-sm-8 col-sm-push-2">
23 <h1 class="text-center">이더리움 부동산</h1>
24 <hr/>
25 <br/>
26 </div>
27 </div>
28
```

그림 1. 부동산 DApp HTML 파일

```
{ } real-estate.json ×
src > { } real-estate.json > ...
1  [
2
3      {
4          "id": 0,
5          "type": "아파트",
6          "picture": "images/apartment.jpg",
7          "area": "85A/59",
8          "price": 1.23
9      },
10     {
11         "id": 1,
12         "type": "연립주택",
13         "picture": "images/townhouse.jpg",
14         "area": "115/76",
15         "price": 1.56
16     },
17     {
18         "id": 2,
19         "type": "하우스",
20         "picture": "images/house.jpg",
21         "area": "204/72",
22         "price": 2.74
23     },
24     {
25         "id": 3,
26         "type": "하우스",
27         "picture": "images/house.jpg",
28         "area": "693/82",
29         "price": 1.87
30     }
31 ]
```

그림 2. 부동산 DApp 데이터(json)



```
contracts > RealEstate.sol
1
2  contract RealEstate {
3      struct Buyer {
4          address buyerAddress;
5          string name;
6          uint age;
7      }
8
9      mapping (uint => Buyer) public buyerInfo;
10     address payable public owner;
11     address[10] public buyers;
12
13     constructor() payable public {
14         owner = payable(msg.sender);
15     }
16
17     function buyRealEstate(uint _id, string memory _name, uint _age) public payable {
18         require(_id >= 0 && _id <= 9);
19         buyers[_id] = msg.sender;
20         buyerInfo[_id] = Buyer(msg.sender, _name, _age);
21
22         owner.transfer(msg.value);
23     }
24
25
26
27
28
29
```

그림 3. 부동산 DApp 코드

## 이더리움 부동산

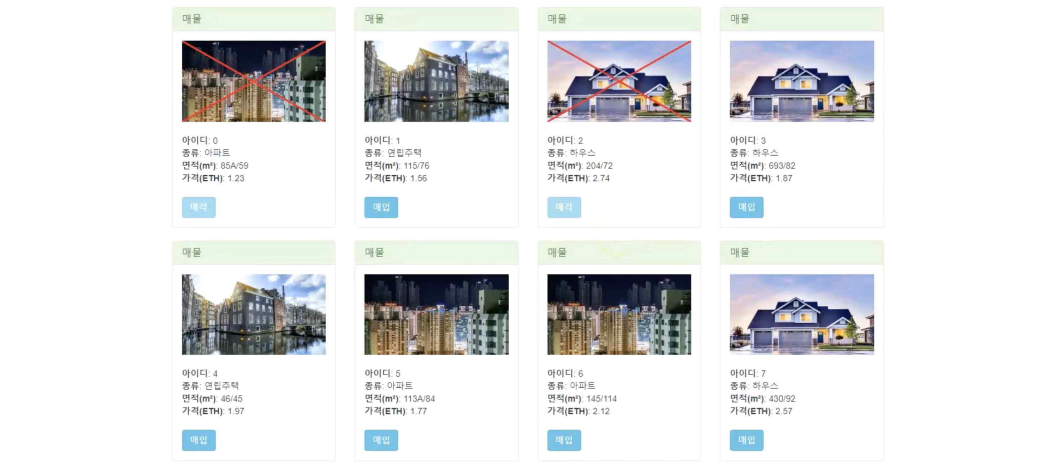
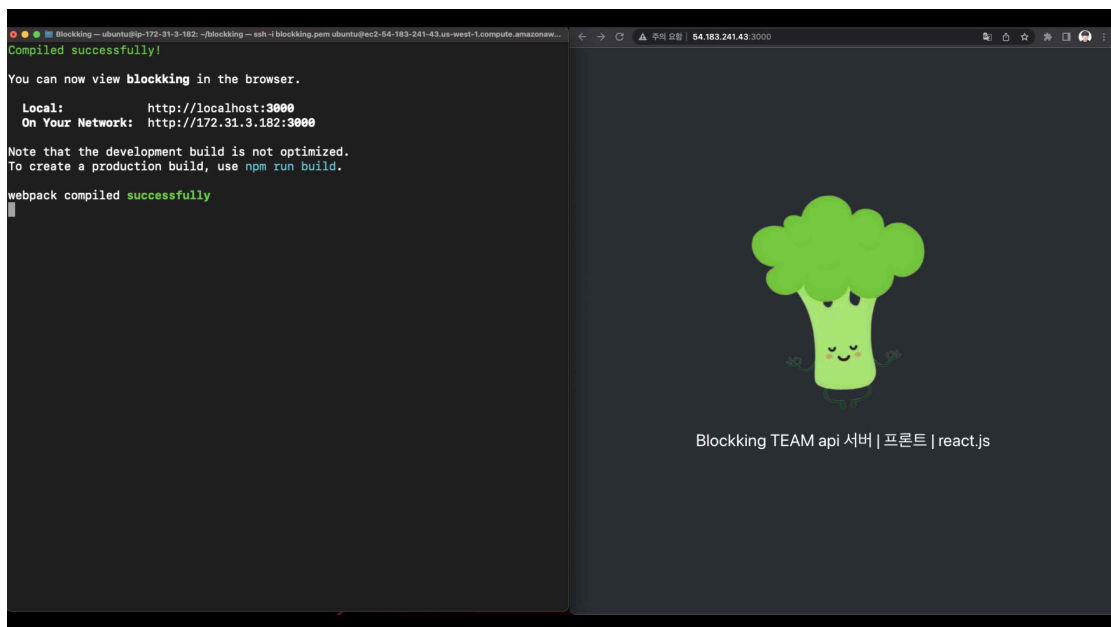


그림 4. 완성된 부동산 DApp 웹페이지

## 5.2 AWS 서버 구축

### ① React 실행 및 동작 화면





② blockking.pem 키를 공개적으로 볼 수 없도록 설정 | ssh 접속

```
Blockking — ubuntu@ip-172-31-3-182: ~ — ssh -i blockking.pem ubuntu@ec2-54-183-241-43.us-west-1.compute.amazonaws.com — 8...
sejin@sejinui-MacBookPro Blockking % chmod 400 blockking.pem

sejin@sejinui-MacBookPro Blockking % ssh -i "blockking.pem" ubuntu@ec2-54-183-241-43.us-west-1.compute.amazonaws.com
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1081-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Wed Jul 27 16:36:36 UTC 2022

System load:  0.22                Processes:            122
Usage of /:   64.9% of 7.57GB     Users logged in:     1
Memory usage: 46%                IP address for eth0:  172.31.3.182
Swap usage:   0%                 IP address for docker0: 172.17.0.1

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

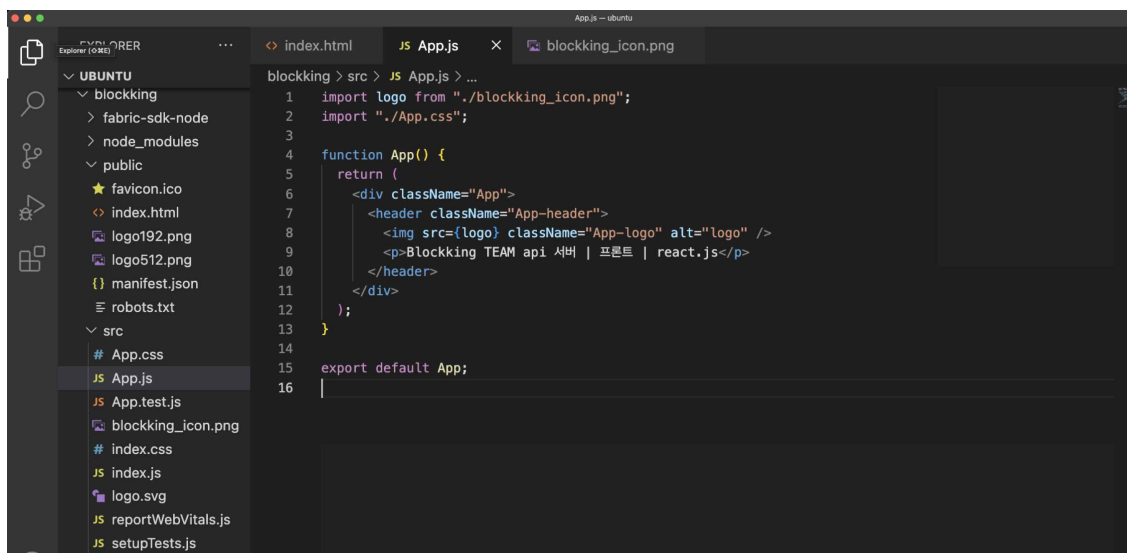
https://ubuntu.com/aws/pro

0 updates can be applied immediately.

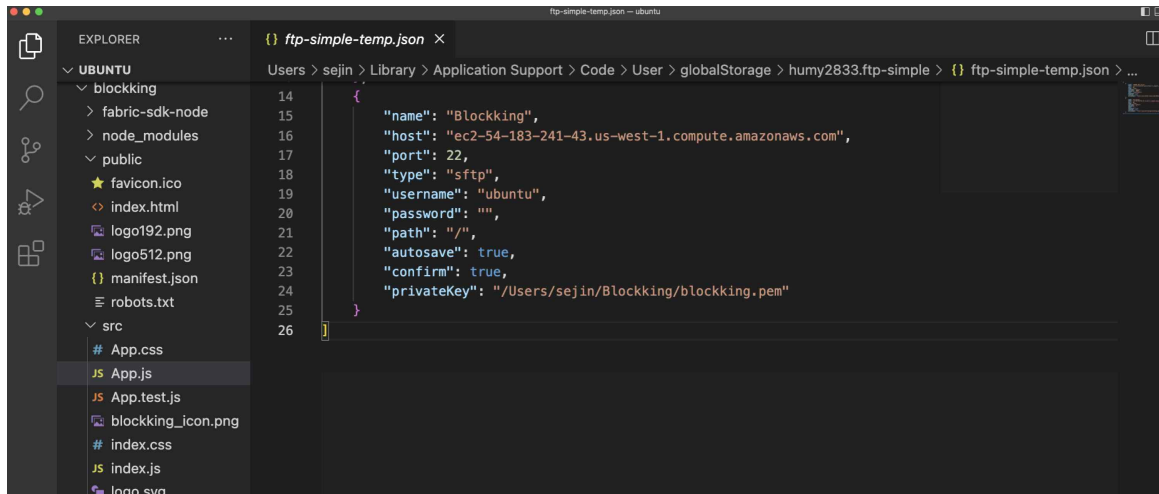
New release '20.04.4 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Wed Jul 27 16:11:08 2022 from 164.125.222.153
ubuntu@ip-172-31-3-182:~$
```

③ sftp를 통한 서버 접속



#### ④ 코드 편집기(VScode)에서의 sftp 접속 설정



### 5.3 SQL을 활용한 데이터베이스 구축

```
Blockking - ubuntu@ip-172-31-3-182: ~ -- ssh -i blockking.pem ubuntu@ec2-54-183-241-43.us-west-1.compute.amazonaws.com -- 8...
ubuntu@ip-172-31-3-182:~$ sudo mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.38-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| contract |
| information |
| merchandise |
| mysql |
| performance_schema |
| sys |
+-----+
7 rows in set (0.01 sec)
```

회원 가입 시 입력받은 회원정보, 농부가 등록한 농산물에 대한 정보, 포전거래시의 계약에 대한 정보를 저장할 수 있는 데이터베이스로 총 3개를 구성함.



## 5.4 앱 UI 구성

‘스마트 컨트랙트 기반 농산물 포전거래 플랫폼’이라는 목적에 어울리게 앱 UI를 초록색 계열로 구성. 하단에 네이게이션바를 구성하고 순서대로 홈, 매매, 알림, my 탭을 배치함.



그림 5. 블록킹 앱

다음은 앱 구성화면의 목업으로, 현재까지는 로그인 및 회원가입 화면, 아이디 찾기, 비밀번호 찾기, 회원가입, 가입 후 첫 화면까지 구현함. 어도비xd를 활용해 ui를 개발중이며, UI 작업 완료 후, flutter를 활용한 퍼블리싱 작업을 예정 중.

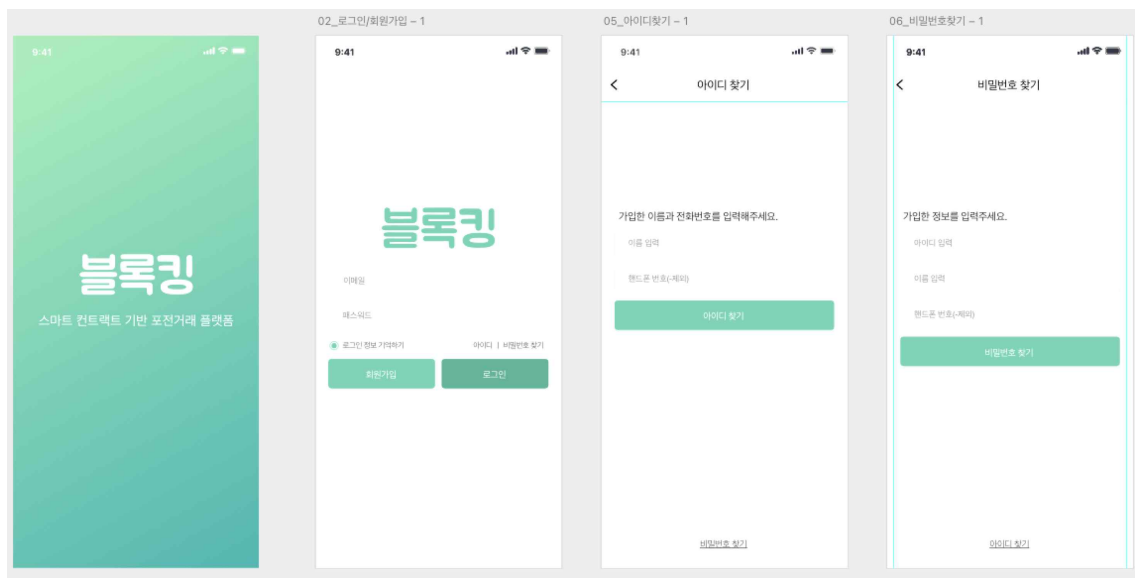


그림 6. 블록킹 앱 UI 1

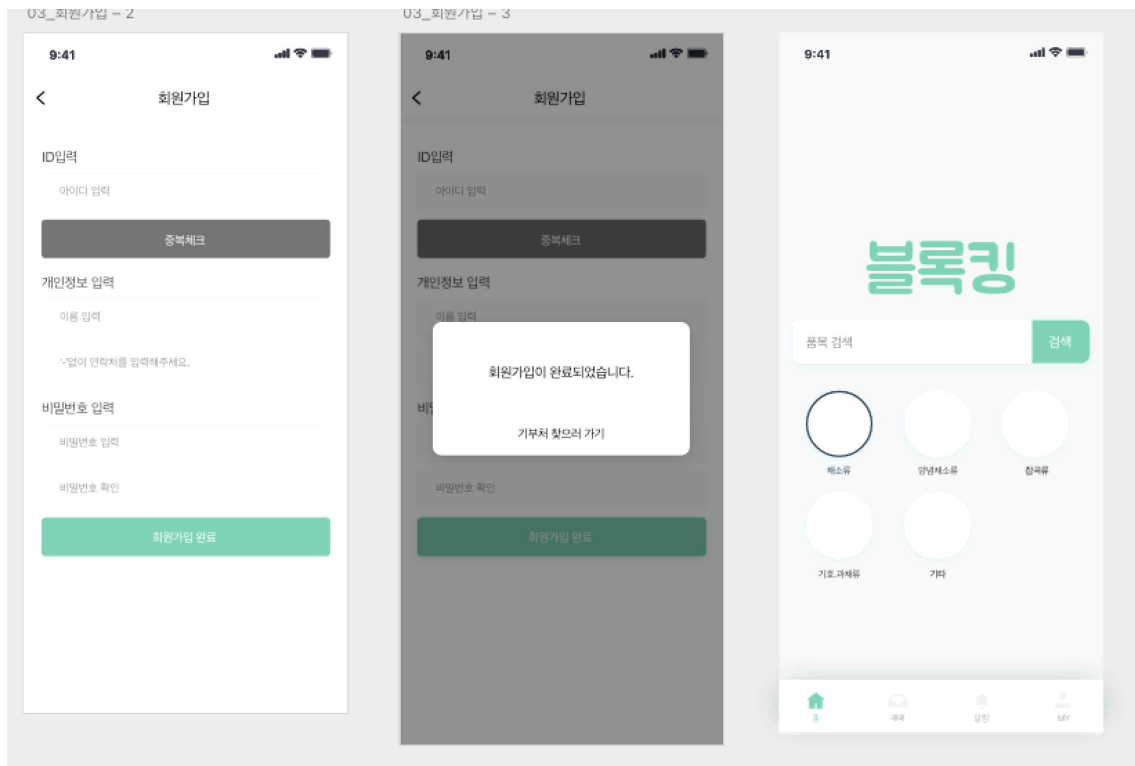


그림 7. 블록킹 앱 UI 2

## 5.5 하이퍼레저 패브릭

```

1 package chaincode
2
3 import (
4     "encoding/json"
5     "fmt"
6
7     "github.com/hyperledger/fabric-contract-api-go/contractapi"
8 )
9
10 // SmartContract provides functions for managing an Asset
11 type SmartContract struct {
12     contractapi.Contract
13 }
14
15 // Asset describes basic details of what makes up a simple asset
16 //Insert struct field in alphabetic order -> to achieve determinism across languages
17 // golang keeps the order when marshal to json but doesn't order automatically
18 type Asset struct {
19     ID          string `json:"id"`
20     Owner       string `json:"owner"`
21     c_date      int    `json:"c_date"`
22     c_merchandise string `json:"c_merchandise"`
23     price       int    `json:"price"`
24     detailAddress string `json:"detailAddress"`
25     c_finaldate int    `json:"c_finaldate"`
26     deposit     int    `json:"deposit"`
27 }
28
29 // InitLedger adds a base set of assets to the ledger
30 func (s *SmartContract) InitLedger(ctx contractapi.TransactionContextInterface) error {
31     assets := []Asset{
32         {ID: "asset1", Owner: "Jaehyun", c_date: 20220728, c_merchandise: "apple", price: 100, detailAddress: "busan jingu", c_finaldate: 20220729, deposit: 30},
33     }
34
35     for _, asset := range assets {
36         assetJSON, err := json.Marshal(asset)
37         if err != nil {
38             return err
39         }
40
41         err = ctx.GetStub().PutState(asset.ID, assetJSON)
42         if err != nil {
43             return fmt.Errorf("failed to put to world state. %v", err)
44         }
45     }
46
47     return nil
48 }
49
50 // CreateAsset issues a new asset to the world state with given details.
51 func (s *SmartContract) CreateAsset(ctx contractapi.TransactionContextInterface, id string, owner string, date int, merchandise string, price int, detailAddress string, finaldate int, deposit int) error {
52     exists, err := s.AssetExists(ctx, id)
53     if err != nil {
54         return err
55     }
56     if exists {
57         return fmt.Errorf("the asset %s already exists", id)
58     }
59
60     asset := Asset{
61         ID:          id,
62         Owner:       owner,
63         c_date:      date,
64         c_merchandise: merchandise,
65         price:       price,
66         detailAddress: detailAddress,
67     }

```

그림 8. 체인코드 구현

스마트 컨트랙트를 기반으로 농산물을 포전거래 함으로써 불법 계약 방지, 산지 유통인 주도의 가격 결정 방지할 수 있는 플랫폼 개발이 목적이기 때문에 법제처에서 제공하는 포전거래 표준계약서에서 체인코드에 들어가는 요소들을 추출해서 작성함.

체인코드에는 아래와 같은 함수들을 구현하였으며, 필요에 따라 사용할 예정이며 필요에 따라 추후 더 구현할 예정.

- InitLedger//원장초기화 함수(초기 자산 설정)
- CreateAsset//자산 생성 함수
- ReadAsset//자산 읽기 함수
- UpdateAsset//자산 수정 함수
- DeleteAsset//자산 삭제 함수
- AssetExists//자산 확인 함수
- TransferAsset//자산 거래 함수
- GetAllAssets//모든 자산 출력 함수

```

138
139 // AssetExists returns true when asset with given ID exists in world state
140 func (s *SmartContract) AssetExists(ctx contractapi.TransactionContextInterface, id string) (bool, error) {
141     assetJSON, err := ctx.GetStub().GetState(id)
142     if err != nil {
143         return false, fmt.Errorf("failed to read from world state: %v", err)
144     }
145
146     return assetJSON != nil, nil
147 }
148
149 // TransferAsset updates the owner field of asset with given id in world state, and returns the old owner.
150 func (s *SmartContract) TransferAsset(ctx contractapi.TransactionContextInterface, id string, newOwner string) (string, error) {
151     asset, err := s.ReadAsset(ctx, id)
152     if err != nil {
153         return "", err
154     }
155
156     oldOwner := asset.Owner
157     asset.Owner = newOwner
158
159     assetJSON, err := json.Marshal(asset)
160     if err != nil {
161         return "", err
162     }
163
164     err = ctx.GetStub().PutState(id, assetJSON)
165     if err != nil {
166         return "", err
167     }
168
169     return oldOwner, nil
170 }
171
172 // GetAllAssets returns all assets found in world state
173 func (s *SmartContract) GetAllAssets(ctx contractapi.TransactionContextInterface) ([]*Asset, error) {
174     // range query with empty string for startKey and endKey does an
175     // open-ended query of all assets in the chaincode namespace.
176     resultsIterator, err := ctx.GetStub().GetStateByRange("", "")
177     if err != nil {
178         return nil, err
179     }
180     defer resultsIterator.Close()
181
182     var assets []*Asset
183     for resultsIterator.HasNext() {
184         queryResponse, err := resultsIterator.Next()
185         if err != nil {
186             return nil, err
187         }
188
189         var asset Asset
190         err = json.Unmarshal(queryResponse.Value, &asset)
191         if err != nil {
192             return nil, err
193         }
194         assets = append(assets, &asset)
195     }
196
197     return assets, nil
198 }
199

```

그림 9. 네트워크 구현



The diagram illustrates a distributed system architecture with three nodes, each represented by a light blue rounded rectangle. The nodes are arranged horizontally and separated by vertical white lines. The first node on the left is labeled 'orderer' and has the domain 'orderer.example.com' written below it. The second node in the middle is labeled 'peer0' and has the domain 'org1.example.com' written below it. The third node on the right is labeled 'peer1' and has the domain 'org2.example.com' written below it.

[illegible]

그림 11. deploying 작업 화면 2



## ②작성해두었던 체인코드를 채널에 deploying.

Fabric에서 스마트 계약은 체인코드라고 하는 패키지로 네트워크에 배포함. 체인코드는 조직의 피어에 설치된 다음 채널에 배포되어 트랜잭션을 승인하고 블록체인 원장과 상호 작용하는 데 사용할 수 있음. 체인코드를 채널에 배포하기 전에 채널 구성원은 체인코드 거버넌스를 설정하는 체인코드 정의에 동의해야 함. 필요한 수의 조직이 동의하면 체인코드 정의를 채널에 커밋할 수 있고 체인코드를 사용할 수 있음.

deployCC하위 명령은 자산전송 체인코드를 설치한 다음 채널 플래그를 사용하여 지정된 채널(또는 지정된 채널이 없는 경우)에 체인코드를 배포함. -ccid플래그를 사용해 체인코드가 GO언어로 작성되었음을 명시함.

```
jeonhyun@jeonhyun-ub:~/fabric$ go env -w GOPATH=/home/jeonhyun/go
~/network.sh deployCC -ccid basic -ccp ../asset-transfer-basic/chaincode-go -ccli go -c tradingchannel

Installing chaincode on channel 'tradingchannel'

executing with the following
- CHANNEL_NAME: tradingchannel
- CC_NAME: basic
- CC_SRC_PATH: ../asset-transfer-basic/chaincode-go
- CC_SRC_LANGUAGE: go
- CC_VERSION: 1.0
- CC_SEQUENCE: 1
- CC_END_POLICY: NA
- CC_COLL_CONFIG: NA
- CC_INIT_FCN: NA
- DELAY: 1
- MAX_RETRY: 1
- VERBOSE: false

~/network.sh deployCC -ccid basic -ccp ../asset-transfer-basic/chaincode-go
/home/jeonhyun/go/src/fabric-samples/asset-transfer-basic/chaincode-go /home/jeonhyun/go/src/fabric-samples/test-network
/home/jeonhyun/go/src/fabric-samples/test-network
Finished vendoring Go dependencies
+ peer lifecycle chaincode package basic.tar.gz --path ../asset-transfer-basic/chaincode-go --lang golang --label basic_1.0
+ res=0
Chaincode is packaged
Installing chaincode on peer0.org1...
+ peer lifecycle chaincode install basic.tar.gz
+ res=0
[2021-09-08 14:31:11.766 EST 000] INFO [cli.lifecycle.chaincode] submitInstallProposal -> Installed remotely: response=status:200 payload:"\n2basic_1.0:285df1398eaeae9d82f97ae1ea895c63b7c20043b0b0648d24941860df5f670221basic_1.0" >
[2021-09-08 14:31:11.766 EST 000] INFO [cli.lifecycle.chaincode] submitInstallProposal -> Chaincode code package Identifier: basic_1.0:285df1398eaeae9d82f97ae1ea895c63b7c20043b0b0648d24941860df5f67
Chaincode is installed on peer0.org1
[Install] chaincode on peer0.org1...
Using organization 0
+ peer lifecycle chaincode install basic.tar.gz
+ res=0
[2021-09-08 14:31:11.766 EST 000] INFO [cli.lifecycle.chaincode] submitInstallProposal -> Installed remotely: response=status:200 payload:"\n2basic_1.0:285df1398eaeae9d82f97ae1ea895c63b7c20043b0b0648d24941860df5f670221basic_1.0" >
[2021-09-08 14:31:11.766 EST 000] INFO [cli.lifecycle.chaincode] submitInstallProposal -> Chaincode code package Identifier: basic_1.0:285df1398eaeae9d82f97ae1ea895c63b7c20043b0b0648d24941860df5f67
Chaincode is installed on peer0.org1
Using organization 1
+ peer lifecycle chaincode queryInstalled
+ res=0
Installed chaincodes on peers:
Package ID: basic_1.0:285df1398eaeae9d82f97ae1ea895c63b7c20043b0b0648d24941860df5f67, Label: basic_1.0
Query installed successful on peer0.org1 on channel
Using organization 2
+ peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile /home/jeonhyun/go/src/fabric-samples/test-network/organizations/ordererOrganizations/example.com/tlsca/example.com-cert.pem --channelID tradingchannel --name basic --version 1.0 --package-id basic_1.0:285df1398eaeae9d82f97ae1ea895c63b7c20043b0b0648d24941860df5f67 --sequence 1
+ res=0
[2021-09-08 14:31:11.877 EST 000] INFO [chaincodeCmd] ClientWait -> txid [8cdf30a5ccc5948973cacabca221e998dc66da499a36dd15f88eb4be193d] committed with status (VALID) at localhost:7051
Chaincode definition approved on peer0.org1 on channel 'tradingchannel'
Using organization 3
Checking the commit readiness of the chaincode definition on peer0.org1 on channel 'tradingchannel'...
Attempting to check the commit readiness of the chaincode definition on peer0.org1, delay after 1 seconds.
+ peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile /home/jeonhyun/go/src/fabric-samples/test-network/organizations/ordererOrganizations/example.com/tlsca/example.com-cert.pem --channelID tradingchannel --name basic --version 1.0 --package-id basic_1.0:285df1398eaeae9d82f97ae1ea895c63b7c20043b0b0648d24941860df5f67 --sequence 1
+ res=0
[2021-09-08 14:31:12.063 EST 000] INFO [chaincodeCmd] ClientWait -> txid [44b380495c81d0ee79464eeaf32411e1e325693ab57f0e36df45161b3761ce] committed with status (VALID) at localhost:9051
Chaincode definition approved on peer0.org1 on channel 'tradingchannel'
Using organization 4
Checking the commit readiness of the chaincode definition on peer0.org1 on channel 'tradingchannel'...
Attempting to check the commit readiness of the chaincode definition on peer0.org1, delay after 1 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID tradingchannel --name basic --version 1.0 --sequence 1 --output json
+ res=0
{
  "approvals": {
    "Org1MSP": true,
    "Org2MSP": false
  }
}
Checking the commit readiness of the chaincode definition successful on peer0.org1 on channel 'tradingchannel'
Using organization 0
Checking the commit readiness of the chaincode definition on peer0.org1 on channel 'tradingchannel'...
Attempting to check the commit readiness of the chaincode definition on peer0.org1, delay after 1 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID tradingchannel --name basic --version 1.0 --sequence 1 --output json
+ res=0
{
  "approvals": {
    "Org1MSP": true,
    "Org2MSP": false
  }
}
Checking the commit readiness of the chaincode definition successful on peer0.org1 on channel 'tradingchannel'
Using organization 1
+ peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile /home/jeonhyun/go/src/fabric-samples/test-network/organizations/ordererOrganizations/example.com/tlsca/example.com-cert.pem --channelID tradingchannel --name basic --version 1.0 --package-id basic_1.0:285df1398eaeae9d82f97ae1ea895c63b7c20043b0b0648d24941860df5f67 --sequence 1
+ res=0
[2021-09-08 14:31:12.063 EST 000] INFO [chaincodeCmd] ClientWait -> txid [44b380495c81d0ee79464eeaf32411e1e325693ab57f0e36df45161b3761ce] committed with status (VALID) at localhost:9051
Chaincode definition approved on peer0.org1 on channel 'tradingchannel'
Using organization 2
Checking the commit readiness of the chaincode definition on peer0.org1 on channel 'tradingchannel'...
Attempting to check the commit readiness of the chaincode definition on peer0.org1, delay after 1 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID tradingchannel --name basic --version 1.0 --sequence 1 --output json
+ res=0
{
  "approvals": {
    "Org1MSP": true,
    "Org2MSP": true
  }
}
Checking the commit readiness of the chaincode definition successful on peer0.org1 on channel 'tradingchannel'
Using organization 3
Checking the commit readiness of the chaincode definition on peer0.org1 on channel 'tradingchannel'...
Attempting to check the commit readiness of the chaincode definition on peer0.org1, delay after 1 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID tradingchannel --name basic --version 1.0 --sequence 1 --output json
+ res=0
{
  "approvals": {
    "Org1MSP": true,
    "Org2MSP": true
  }
}
Checking the commit readiness of the chaincode definition successful on peer0.org1 on channel 'tradingchannel'
Using organization 4
```

```
+ peer lifecycle chaincode checkcommitreadiness --channelID tradingchannel --name basic --version 1.0 --sequence 1 --output json
+ res=0
{
  "approvals": {
    "Org1MSP": true,
    "Org2MSP": false
  }
}
Checking the commit readiness of the chaincode definition successful on peer0.org1 on channel 'tradingchannel'
Using organization 0
Checking the commit readiness of the chaincode definition on peer0.org1 on channel 'tradingchannel'...
Attempting to check the commit readiness of the chaincode definition on peer0.org1, delay after 1 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID tradingchannel --name basic --version 1.0 --sequence 1 --output json
+ res=0
{
  "approvals": {
    "Org1MSP": true,
    "Org2MSP": false
  }
}
Checking the commit readiness of the chaincode definition successful on peer0.org1 on channel 'tradingchannel'
Using organization 1
+ peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile /home/jeonhyun/go/src/fabric-samples/test-network/organizations/ordererOrganizations/example.com/tlsca/example.com-cert.pem --channelID tradingchannel --name basic --version 1.0 --package-id basic_1.0:285df1398eaeae9d82f97ae1ea895c63b7c20043b0b0648d24941860df5f67 --sequence 1
+ res=0
[2021-09-08 14:31:12.063 EST 000] INFO [chaincodeCmd] ClientWait -> txid [44b380495c81d0ee79464eeaf32411e1e325693ab57f0e36df45161b3761ce] committed with status (VALID) at localhost:9051
Chaincode definition approved on peer0.org1 on channel 'tradingchannel'
Using organization 2
Checking the commit readiness of the chaincode definition on peer0.org1 on channel 'tradingchannel'...
Attempting to check the commit readiness of the chaincode definition on peer0.org1, delay after 1 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID tradingchannel --name basic --version 1.0 --sequence 1 --output json
+ res=0
{
  "approvals": {
    "Org1MSP": true,
    "Org2MSP": true
  }
}
Checking the commit readiness of the chaincode definition successful on peer0.org1 on channel 'tradingchannel'
Using organization 3
Checking the commit readiness of the chaincode definition on peer0.org1 on channel 'tradingchannel'...
Attempting to check the commit readiness of the chaincode definition on peer0.org1, delay after 1 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID tradingchannel --name basic --version 1.0 --sequence 1 --output json
+ res=0
{
  "approvals": {
    "Org1MSP": true,
    "Org2MSP": true
  }
}
Checking the commit readiness of the chaincode definition successful on peer0.org1 on channel 'tradingchannel'
Using organization 4
```

## ③네트워크와 상호작용을 위한 환경변수 설정

```
jaehyung@jaehyun-M5-TCH4: /usr/src/fabric-samples/test-network$ export PATH=${PWD}/../bin:$PATH
export FABRIC_CFG_PATH=${PWD}/../config/
export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="org1sp"
export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
export CORE_PEER_ADDRESS=localhost:7001
```

peer 명령어를 사용해 CLI를 Org1로 작동할 수 있는 환경 변수를 설정.

CORE\_PEER\_TLS\_ROOTCERT\_FILE 및 환경변수는 폴더 CORE\_PEER\_MSPCONFIGPATH 의 Org1 암호화 자료를 가리킴.

## ④체인코드의 기능을 호출하여 원장에 있는 자산 생성 및 옮기기

```
jaehyung@jaehyun-M5-TCH4: /usr/src/fabric-samples/test-network$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C tradingchannel -n basic --peerAddresses localhost:7001 --tlsrootcertfiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses localhost:9051 --tlsrootcertfiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" -c '{"function": "InitAsset", "Args": ["asset1"]}'
[INFO] 020 15:40:10.123 UTC [chaincode] [chaincodeInvoke] -> Chaincode invoke successful. result: status:200
```

앞에서 자산 전송 체인코드를 설치했기 때문에 (Go) 체인코드의 기능을 호출하여 체인코드 Init함수를 구현하며 정해두었던 자산의 초기 목록을 원장에 넣음.

그 후 CLI에서 원장을 쿼리함. peer chaincode query -C tradingchannel -n basic -c '{"Args":["GetAllAssets"]}' 명령을 실행하여 채널 원장에 추가된 자산 목록을 가져오는데 성공함.

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C tradingchannel -n basic --peerAddresses localhost:7001 --tlsrootcertfiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses localhost:9051 --tlsrootcertfiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" -c '{"function": "TransferAsset", "Args": ["asset1", "sejin"]}'
```

명령을 실행해 asset1 자산 소유자를 jaehyun에서 sejin으로 변경하였음.

```
jaehyung@jaehyun-M5-TCH4: /usr/src/fabric-samples/test-network$ peer chaincode query -C tradingchannel -n basic -c '{"Args":["GetAllAssets"]}'
[{"id":"asset1","owner":"jaehyun"}]
jaehyung@jaehyun-M5-TCH4: /usr/src/fabric-samples/test-network$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C tradingchannel -n basic --peerAddresses localhost:7001 --tlsrootcertfiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" -c '{"function": "TransferAsset", "Args": ["asset1", "sejin"]}'
[INFO] 020 15:40:10.123 UTC [chaincode] [chaincodeInvoke] -> Chaincode invoke successful. result: status:200 payload:"jaehyun"
jaehyung@jaehyun-M5-TCH4: /usr/src/fabric-samples/test-network$ peer chaincode query -C tradingchannel -n basic -c '{"Args":["GetAllAssets"]}'
[{"id":"asset1","owner":"sejin"}]
```