2023 중간보고서

농산물 근원 탐구와 시스템관리 플랫폼

부산대학교
PUSAN NATIONAL UNIVERSITY

팀명	조이름좀추천해조
지도교수	권동현 교수님
팀 번호	35
분과	D
제출일	2023.8.4
팀원	202055654 장우정 202055535 리보양 202055562 쑹위판

<목차 >

1. 요구 조건 및 제약 사항 분석 3
1.1. 과제 목표 3
1.2. 요구조건 4
1.3. 제약 사항 분석 및 수정사항 4

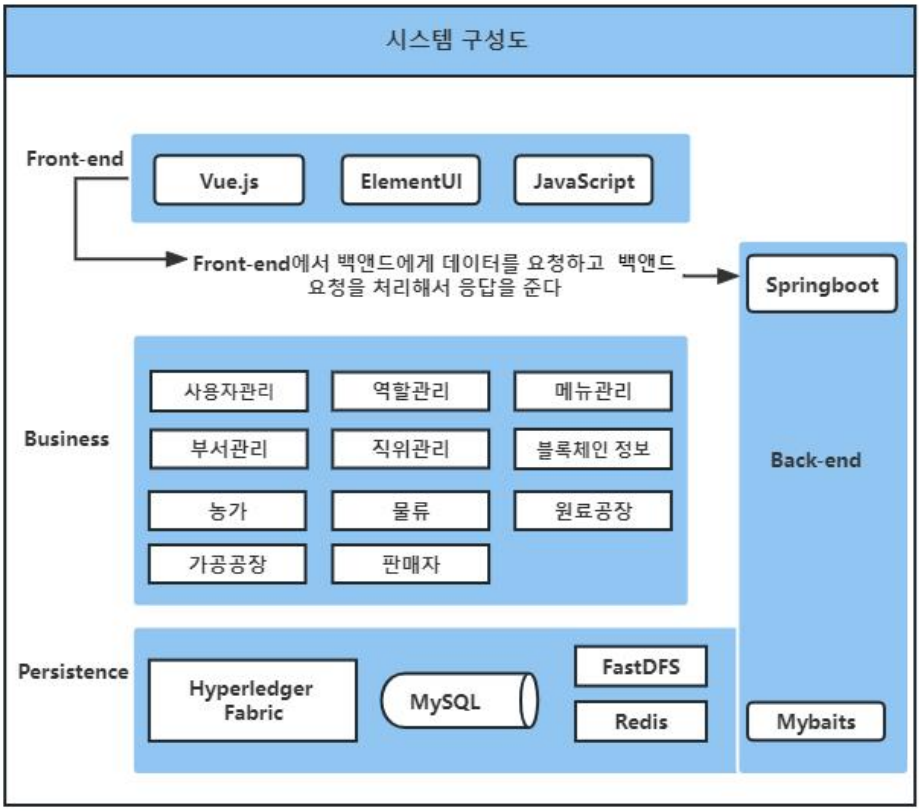
2. 설계 상세화 및 변경내용 5
2.1. 네트워크 구성 5
2.2. 전체 시스템 권한 설계도 6
2.2. 변경내용 7

3. 과제 추진 계획 7
3.1. 개발일정 7
3.2. 구성원 별 진척도 8

4. 과제 진행 내용 8
4.1. 서버 환경 설정 8
4.2. 블록체인 네트워크 9
4.3. 실현 페이지 21

1. 요구 조건 및 제약 사항 분석

과제 전체 시스템 주성도



1.1.과제 목표

- 기업관리자에게 웹사이트 형태의 서비스 제공

  이 시스템은 농가/기업 가공 공장에 농산물 정보를 기록하고 관리할 수 있는 도구를 제공하여 제품을 더 잘 이해하고 후속 제품 판매 및 마케팅 전략에 도움이 된다.

- 소비자에게 농산물 정보 조회 제공

  소비자 로그인 없이 소비자조회창에서 판매자가 제공하는 조회번호를 통해 농산물 및 블록체인 정보를 조회할 수 있다

- 시스템관리자는 기업관리자 및 소비자를 위해 안정한 시스템 제공

  농산물 근원 탐구 시스템은 블록체인 기술을 활용하여 제품의 완전성과 신뢰성을 보장한다. 데이터 변경과 위조를 방지하여 제품의 안전성 을 높이다. 이 시스템은 제품의 투명성과 근원 탐구 가 능성을 높이면서 기업관리자 및 소비자에게 더 많은 선택과 안정을 제공한다.

1.2. 요구조건

1.2.1. 블록체인 네트워크

조직(Org)	농가부/ 식품 원료공장/ 가공공장/ 판매부/ 물류
채널	각 peer가 입력한 정보 기록하기

1.2.2. Front-end 및 Back-end

Front-end	● 프론트는 Vue를 통한 라우팅(Spa,Single Page Applicatoin)으로 구성한다. ● 웹디자인은 Elenment UI의 컴포넌트를 사용하여 디자인하였다. ● 백엔드와통신은 Axios를 사용하여 javascript 요청을 비동기적으로 하도록 만들었다.
Back-end	● Spring Boot framework를 이용해 MySQL, Redis에 접근 및 데이터 추출 및 클라이언트와 통신. ● Spring Security를 이용해 사용자에게 부여된 권한에 따른 접근 제어 메커니즘 구현 ● Redis를 이용해 로그인 정보(JWT) 저장. ● Hyperledger Fabric를 사용하여 블록체인 네트워크 구성. ● FastDFS를 사용하여 블록체인 네트워크에 업로드된 이미지 저장.

1.3.제약 사항 분석 및 수정사항

블록체인 노드	제약사항	실행 start.sh 스크립트 오류 보고
	해결방안	docker-compose에서 peer와 order의 미러를 1.4.12버전으로 지정해야 합니다.
	제약사항	Node.js 18 버전이 너무 높으면 Hyperledger Fabric 1.2 버전의 SDK와 호환되지 않습니다.
	해결방안	Node.js버전을 12로 조정

2. 설계 상세화 및 변경내용

2.1. 네트워크 구성

Organizations				
farmer	material	product	retailer	diver
농가부	식품 원료공장	가공공장	판매부	물류

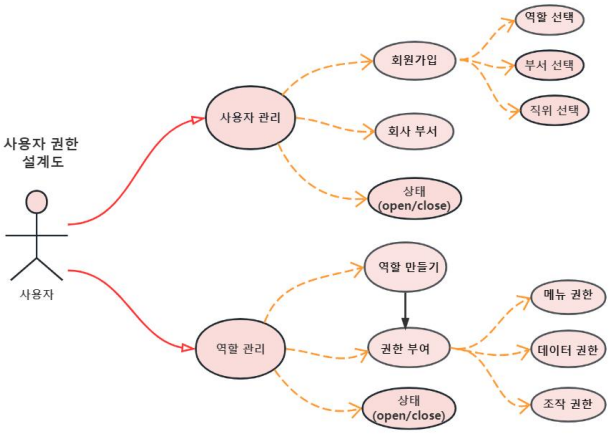
2.1.1 참여 peer

농부	● 농산물 기본정보 입력 ● 농산물 재배정보 입력
원료공장	● 농산물 품질검사정보 입력
가공공장	● 가공공장 기본정보 입력 ● 농산물 가공내용 조회 ● 가공공장 직원에게 작업 분배
가공공장 직원	● 작업 완성도 입력
마트	● 마트정보 입력 ● 상품정보 입력
운송	● 기사 정보 입력 ● 운송 정보 입력

2.1.2. 하이퍼레저 패브릭

체인코드	농산물의 생산, 가공, 운송, 판매 등의 과정을 탐구하고 관련 데이터를 블록체인에 기록한다.

2.2. 전체 시스템 권한 설계도



시스템 디자인 생각의 근원:

RBAC (Role-Based Access Control) 모델은 시스템에서 자원에 대한 접근 권한을 관리하고 제어하는 접근 제어 모델이다. RBAC 모델은 접근 권한을 역할과 관련시키며, 사용자는 해당하는 역할에 할당되어 시스템의 권한 제어를 간소화하고 관리한다.

블록체인의 다섯 개 채널에 대응하기 위해 시스템에서는 관리자, 농가, 기사, 원료 공장책임자, 가공 공장책임자, 판매자 등 다양한 사용자 정의된다.각 역할에는 서로 다른 권한과 액세스할 수 있는 페이지 모듈이 있다.다음은 권한을 설정하는 순서이다

권한 설정 순서:

1. 메뉴 관리로 이동: 새로운 메뉴와 권한을 추가한다.

권한 유형 메뉴는 시스템 리소스로도 이해되며 사용자 메뉴 트리에 표시되지 않다.

2. 메뉴의 "권한 문자열" 은 권한을 제어하는 키 필드입니다.

예를 들어이 권한 식별자는 @PreAuthorize("@ss.hasPermi('system:menu:add')") 해당하며 일치하는 경우에만 URL에 액세스할 수 있다.

```
// PreAuthorize 어노테이션에서 지정된대로 현재 인증된 사용자가 'system:menu:add' 권한을 가지고 있는 경우에만 접근할 수 있습니다.
@PreAuthorize("@ss.hasPermi('system:menu:add')")
@Log(title = "메뉴 관리", businessType = BusinessType.INSERT) // 이는 '메뉴 관리' 제목으로 로그에 작업을 기록합니다.
@PostMapping // 이 메서드는 HTTP POST 요청을 처리합니다.
public AjaxResult add(@Validated @RequestBody SysMenu menu) // 메서드는 SysMenu 객체를 입력으로 받아들이고, 요청 본문에 포함되어 있는 내용을 유효성 검사합니다.
{
    // 메뉴 이름이 시스템에서 이미 존재하는지 확인합니다.
    if (UserConstants.NOT_UNIQUE.equals(menuService.checkMenuNameUnique(menu)))
    {
        return AjaxResult.error( msg: "새 메뉴'" + menu.getMenuName() + "'을(를) 추가하지 못했습니다. 이미 존재하는 메뉴 이름입니다.");
        // 메뉴 이름이 이미 존재하는 경우 오류 메시지를 반환합니다.
    }
    else if (UserConstants.YES_FRAME.equals(menu.getIsFrame())
            && !StringUtils.startsWithAny(menu.getPath(), Constants.HTTP, Constants.HTTPS))
    {
        return AjaxResult.error( msg: "새 메뉴'" + menu.getMenuName() + "'을(를) 추가하지 못했습니다. 주소는 반드시 http(s)://로 시작해야 합니다.");
        // 메뉴가 프레임으로 표시되며 주소가 http(s)://로 시작하지 않는 경우 오류 메시지를 반환합니다.
    }
    menu.setCreateBy(SecurityUtils.getUsername());
    return toAjax(menuService.insertMenu(menu));
    // 새 메뉴 정보를 저장하고 성공 결과를 반환합니다.
}
```

3. 메뉴 구성이 완료된 후 역할 관리: 역할에 새로 추가되고 부여된 기능 권한, 해당 메뉴 및 권한을 선택하고 역할과 메뉴의 관계를 설정한다.

4. 역할 구성이 완료된 후 부서 관리 및 사용자 관리에 들어가 부서를 만들고 사용자를 만들고 해당 역할을 선택하고 사용자와 역할 관계를 설정한다

5. 이렇게 하면 전체 권한 시스템이 구축되고 사용자가 시스템에 로그인한 후 사용자와 관련된 역할을 통해 메뉴/권한을 얻을 수 있다.

2.3. 변경내용

소비자에게 농산물 정보 조회 제공	● 소비자 QR코드 조회 -> 소비자 로그인 없이 소비자조회창에서 판매자가 제공하는 조회번호를 통해 농산물 및 블록체인 정보를 조회할 수 있다

3. 과제 추진 계획

3.1. 개발일정

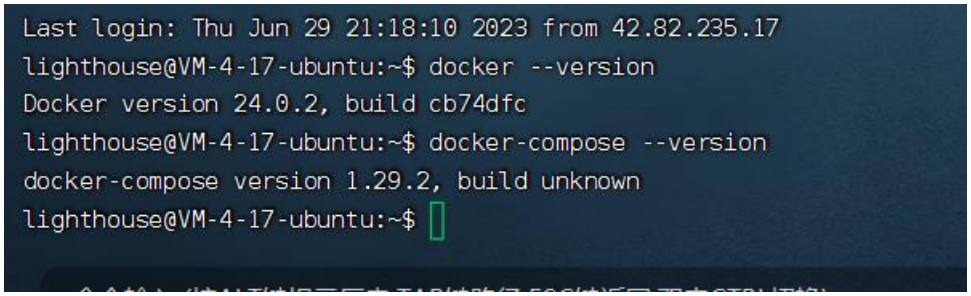
6월		7월					8월					9월				
4주	5주	1주	2주	3주	4주	5주	1주	2주	3주	4주	5주	1주	2주	3주	4주	5주
기본 지식 공부																
블루체인 네트워크 구성																
		서버 황경 구축														
			Spring-boot기반 코딩 개발													
			웹사이트 UI제작													
					중간보고서 적성											
						웹사이트 기능 제작										
								API연동								
											테스트 및 디버깅					
													오류 수정			
														최종 발표 준비 및 보고서 제작		

3.2. 구성원 별 진척도

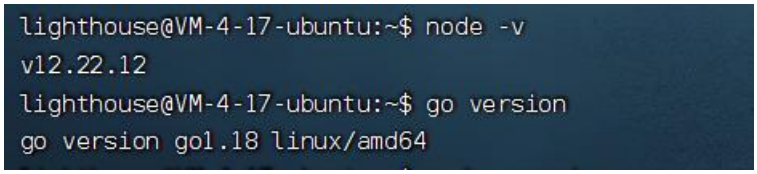
학번	이름	역할 분담
202055535	리보양 Front-end	● Vue를 통한 라우팅(Spa,Single Page Applicatoin)으로 구성원료 ● Elenment UI의 컴포넌트를 사용하여 디자인하였다. ● 블록체인 정보 모듈 실현중
202055562	쑹위판 Back-end	● Database 및 서버환경 구축 원료 ● 블록체인 네트워크 구성 원료 ● 백엔드 시스템에서 블록체인 기술 기능 실현중
202055654	장우정 Back-end	● 로그인, 추가 삭제 수정 기능 버튼등 user 관련 API 기능 구현 완료 ● Spring Boot framework를 이용해 MySQL, Redis에 접근 및 데이터 추출 및 클라이언트와 통신 ● Spring Boot Security 적용 관한 설계 실현중

4. 과제 진행 내용

4.1. 서버 환경 설정



클라우드 서버에 Docker 서비스와 Docker Compose를 설치했다.



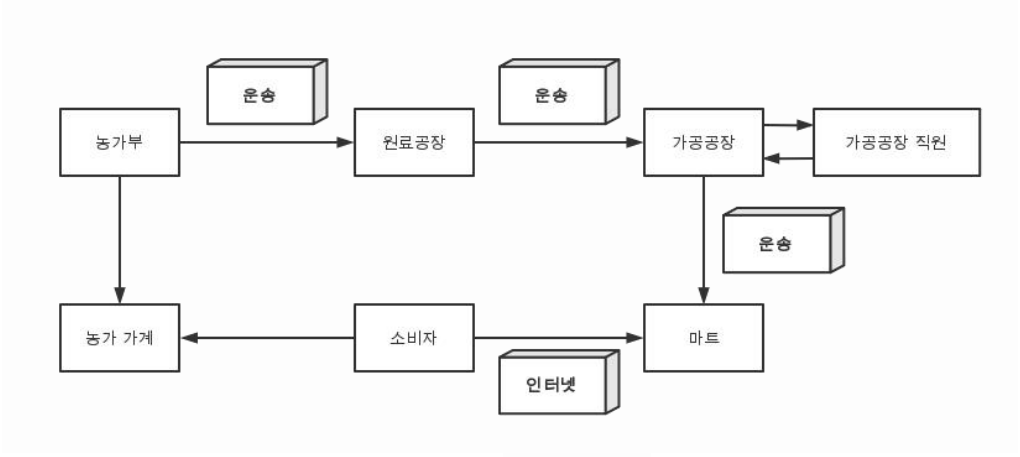
Golang와 node.js.를 설치했다.

```
docker-compose -f docker-compose.yml up -d
Creating network "network_basic" with the default driver
Creating volume "network_orderer.trace.com" with default driver
Creating volume "network_peer0.org1.trace.com" with default driver
Creating volume "network_peer0.org2.trace.com" with default driver
Creating volume "network_peer0.org3.trace.com" with default driver
Creating volume "network_peer0.org4.trace.com" with default driver
Creating volume "network_peer0.org5.trace.com" with default driver
Creating orderer.trace.com ... done
Creating ca.trace.com      ... done
Creating couchdb           ... done
Creating peer0.org4.trace.com ... done
Creating peer0.org5.trace.com ... done
Creating peer0.org2.trace.com ... done
Creating peer0.org1.trace.com ... done
Creating peer0.org3.trace.com ... done
Creating cli               ... done
```

5개 피어(peer) 만들기

peer0.org1.trace.com는 농가 peer
peer0.org2.trace.com는 원료공장 peer
peer0.org3.trace.com는 식품생산 공장 peer
peer0.org4.trace.com 는 마트 peer
peer0.org5.trace.com는 운송peer

4.2. 블록체인 네트워크



[그림] 조직구조도

먼저 농가는 농산물 정보를 시스템에 입력하고 농산물이 익으면 물류에 통보하여 식품 원료공장으로 배송한다. 원료공장은 농산물을 검사하고 품질검사 합격하고 가공 공장으로 물류 운송을 통지한다. 식품 가공 공장에서 농산물을 가공하면 농산물이 식용 식품이 된다. 가공이 완료된 후 상품을 마트으로 배송하여 판매하도록 운송한다. 마지막으로 소비자는 조회를 통해 농산물의 성장, 가공 및 운송 등 정보를 얻을 수 있다.

```yaml
Organizations:

    # SampleOrg defines an MSP using the sampleconfig.  It should never be used
    # in production but may be used as a template for other definitions
    - &OrdererOrg
        # DefaultOrg defines the organization which is used in the sampleconfig
        # of the fabric.git development environment
        Name: OrdererOrg

        # ID to load the MSP definition as
        ID: OrdererMSP
        # MSPDir is the filesystem path which contains the MSP configuration
        MSPDir: crypto-config/ordererOrganizations/trace.com/msp
    - &Org1
        # DefaultOrg defines the organization which is used in the sampleconfig
        # of the fabric.git development environment
        Name: Org1MSP

        # ID to load the MSP definition as
        ID: Org1MSP

        MSPDir: crypto-config/peerOrganizations/org1.trace.com/msp

        AnchorPeers:
            # AnchorPeers defines the location of peers which can be used
            # for cross org gossip communication.  Note, this value is only
            # encoded in the genesis block in the Application section context
            - Host: peer0.org1.trace.com
```

[그림]configtx.yaml의 일부

configtx.yaml파일을 수정하여 네트워크의 상세 내용을 설정해 줄 수 있다. 해당 파일을 사용해서 블록체인 네트워크의 gennesis.block이 생성된다.

```yaml
Profiles:
    FiveOrgsOrdererGenesis:
        Capabilities:
            <<: *ChannelCapabilities
        Orderer:
            <<: *OrdererDefaults
            Organizations:
                - *OrdererOrg
            Capabilities:
                <<: *OrdererCapabilities
        Consortiums:
            SampleConsortium:
                Organizations:
                    - *Org1
                    - *Org2
                    - *Org3
                    - *Org4
                    - *Org5
    FiveOrgsChannel:
        Consortium: SampleConsortium
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *Org1
                - *Org2
                - *Org3
                - *Org4
                - *Org5
            Capabilities:
```

[그림]configtx.yaml Profiles

[그림]은 configtx.yaml의 Profiles이다. 두 개의 프로파일이 정의되어 있다: FiveOrgsOrdererGenesis와 FiveOrgsChannel.

FiveOrgsOrdererGenesis 프로파일은 초기 블록에 대한 구성을 정의한다. 이 프로파일에는 Orderer 노드의 구성 정보가 포함되며, 사용되는 기능, 조직 정보 및 기능에 대한 설정이 포함된다. 또한 SampleConsortium라는 연합이 정의되어 있으며, 여러 조직이 포함되어 있다.

FiveOrgsChannel 프로파일은 FiveOrgsChannel이라는 채널의 구성을 정의한다. 이 프로파일은 해당 채널이 속한 연합을 SampleConsortium로 지정하고, 응용 프로그램 부분의 구성 정보를 정의한다. 응용 프로그램 부분에서는 참여하는 조직 및 해당 기능에 대한 설정이 지정된다.

```
type Crops struct {
    CropsId             string `json:"crops_id"`            //농산물ID
    CropsName           string `json:"crops_name"`          //농산물이름
    Address             string `json:"address"`             //상세주소
    RegisterTime        string `json:"register_time"`       //재배일
    Year                string `json:"year"`                //년도
    FarmerName          string `json:"farmer_name"`         //농가이름
    FarmerID            string `json:"farmer_id"`           //농가ID
    FarmerTel           string `json:"farmer_tel"`          //전화번호
    FertilizerName      string `json:"fertilizer_name"`     //비료이름
    PlatMode            string `json:"plant_mode"`          //재배방식
    BaggingStatus       string `json:"bagging_status"`      //북대여부
    GrowSeedlingsCycle  string `json:"grow_seedlings_cycle"`//육묘 주기
    IrrigationCycle     string `json:"irrigation_cycle"`    //관개 주기
    ApplyFertilizerCycle    string `json:"apply_fertilizer_cycle"` //거름 주기
    WeedCycle           string `json:"weed_cycle"`          //제초 주기
    Remarks             string `json:"remarks"`             //비고
}
```

[그림]농가 data structure

```
type CropsGrowInfo struct {
    CropsGrowId         string `json:"crops_grow_id"`       //농산물 재배ID
    CropsBakId          string `json:"crops_bak_id"`        //농산물ID
    RecordTime          string `json:"record_time"`         //기록 시간
    CropsGrowPhotoUrl   string `json:"crops_grow_photo_url"`//농작물 사진
    Temperature         string `json:"temperature"`         //온도
    GrowStatus          string `json:"grow_status"`         //성장 상태
    WaterContent        string `json:"water_content"`       //수분
    IlluminationStatus  string `json:"illumination_status"` //조명 상태
    Remarks             string `json:"remarks"`             //비고
}
```

[그림]농산물 재배정보data structure

```
type Machining struct {
        MachiningId             string `json:"machining_id"`    //가공ID
        Leader                  string `json:"leader"`          //원료공장 사장
        CropsId                 string `json:"crops_id"`        //농산물ID
        LeaderTel               string `json:"leader_tel"`      //원료공장사장 전화
        FactoryName             string `json:"factory_name"`    //원료공장이름
        TestingResult           string `json:"testing_result"`  //품질검사 결과
        InFactoryTime           string `json:"in_factory_time"` //입고시간
        OutFactoryTime          string `json:"out_factory_time"`//출고시간
        TestingPhotoUrl         string `json:"testing_photo_url"`//품질검사 사진
        Remarks                 string `json:"remarks"`         //비고
}
```

[그림]원료공장 data structure

```
type ProductInfo struct {
        ProductId              string `json:"product_id"`       //제품ID
        CropsId                string `json:"crops_id"`         //농산물ID
        ProductName            string `json:"product_name"`     //제품이름
        MixedIngredients       string `json:"mixed_ingredients"`//혼합 재료
        Leader                 string `json:"leader"`           //가공공장 사장
        LeaderTel              string `json:"leader_tel"`       //사정 전화
        Workshop               string `json:"workshop"`         //작업장
        Factory                string `json:"factory"`          //공장이름
        WorkHours              string `json:"work_hours"`       //근무 시간
        Time                   string `json:"time"`             //시간
        KeepMethod             string `json:"keep_mathod"`      //방법 유지
        NetContent             string `json:"Net_Content"`      //용량
        CookingRecommend       string `json:"cooking_recommend"`//요리법
        Remarks                string `json:"remarks"`          //비고
}
```

[그림]가공공장 data structure

```
type Operation struct {
        OperationId            string `json:"operation_id"`            //적업ID
        CropsId                string `json:"crops_id"`                //농산물ID
        OperationPeopleName    string `json:"operation_people_name"`   //직원이름
        OperationPeopleTel     string `json:"operation_people_tel"`    //직원전화
        Time                   string `json:"time"`                    //시간
        WorkContent            string `json:"work_content"`            //근무내용
        Remarks                string `json:"remarks"`                 //비고
}
```

[그림]가공공장 직원 data structure

```
type Retailer struct {
        ProductId              string `json:"product_id"`       //상품ID
        CropsId                string `json:"crops_id"`         //농산물ID
        RetailerName           string `json:"retailer_name"`    //마트사장 이름
        RetailerTel            string `json:"retailer_tel"`     //마트전화
        RetailerId             string `json:"retailer_id"`      //마트ID
        Retailer               string `json:"retailer"`         //마트이름
        Remarks                string `json:"remarks"`          //비고
}
```

[그림]마트 data structure

```
type Transport struct {
    TransportId          string `json:"transport_id"`           //운송ID
    DriverId             string `json:"driver_id"`              //기사ID
    DriverName           string `json:"driver_name"`            //기사이름
    DriverTel            string `json:"driver_tel"`             //기사전화
    DriverDept           string `json:"driver_dept"`            //부서
    CropsId              string `json:"crops_id"`               //농산물ID
    TransportToChainTime string `json:"transport_to_chain_time"` //체인으로 전송된 시간
    TransportToAddress   string `json:"transport_to_address"`    //이동 주소
    Remarks              string `json:"remarks"`                //비고
}
```

[그림]운송 data structure

먼저, 각 캐릭터가 입력 필요한 정보을 만들었다. 농가는 농산물 이름, 재배방식, 농가 정보 등. 원료공장은 품질검사 결과, 공장기분 정보 등. 가공공장은 가공이 필요한 제품이름, 식용방법, 가공 필요한 기간, 공장 직원이 해야 할 일 등. 가공공정 직원은 근무 시간, 근무내용 등. 마트는 판매하는 상품 정보, 마트의 정보 등. 운송은 기사정보, 이동동선 등 정보를 내부에 가지고 있다.

```
const express = require("express");
const router = express.Router();

//########################   BlockExplorer   #############

const Blockchain = require("../sysapi/blockexplorer");
const FarmerApi = require("../sysapi/farmer");
const DriverApi  = require("../sysapi/driver");
const MaterialApi = require("../sysapi/material");
const ProductApi = require("../sysapi/product");
const RetailerApi = require("../sysapi/retailer");

//##########################    use  route      ########################

router.use("/blockexplorerapi",Blockchain);
router.use("/farmerapi",FarmerApi);
router.use("/driverapi",DriverApi);
router.use("/materialapi",MaterialApi);
router.use("/productapi",ProductApi);
router.use("/retailerapi",RetailerApi);

//##########################  exports     ##############################
module.exports = router;
```

[그림] Express 애플리케이션으로 보이다

[그림]은 각 API 모듈은 Blockchain, Farmer, Driver, Material, Product 및 Retailer와 같은 각 엔터티와 관련된 특정 API 호출을 처리할 책임이다.

```
router.get("/channelBlockInfo",async function (req,res) {
try {
  var state_store = await Fabric_Client.newDefaultKeyValueStore({path: store_path});
  fabric_client.setStateStore(state_store);

  var crypto_suite = Fabric_Client.newCryptoSuite();
  var crypto_store = Fabric_Client.newCryptoKeyStore({path: store_path});
  crypto_suite.setCryptoKeyStore(crypto_store);
  fabric_client.setCryptoSuite(crypto_suite);

  // get the enrolled user from persistence, this user will sign all requests
  var user_from_store = await fabric_client.getUserContext('user1', true);
  if (user_from_store && user_from_store.isEnrolled()) {
    console.log('Successfully loaded user1 from persistence');
    member_user = user_from_store;
  } else {
    throw new Error('Failed to get user1.... run registerUser.js');
  }
  var query_responses = await channel.queryInfo(peer);
  res.send(query_responses)
}catch (e) {
  console.error('Failed to query successfully :: ' + e);
}
```

라우터 함수/channelBlockInfo: 채널 정보 가져오기

- 이 함수는 /channelBlockInfo의 GET 요청을 처리한다.
- 먼저 Hyperledger Fabric 네트워크와 상호 작용하는 Fabric_Client 인스턴스를 만든다.
- 그런 다음 tracechannel이라는 새로운 채널을 만들고 채널에 주문(orderer)과 피어(peer)를 추가한다.
- 그런 다음 함수는 파일 시스템에서 user1이라는 사용자의 컨텍스트 정보를 로드합니다.사용자가 이미 등록되어 있고 컨텍스트가 성공적으로 로드된 경우 이 사용자 컨텍스트는 후속 작업에서 사용할 수 있도록 member_user 변수에 할당된다.
- 마지막으로, channel을 호출합니다.queryInfo(peer) 방법은 채널의 정보를 조회하고 조회 결과를 JSON 응답으로 클라이언트에 보낸다.

```
router.get("/queryBlockInfo",async function(req,res){
console.log("number   "+req.query.number)
try {
  var state_store = await Fabric_Client.newDefaultKeyValueStore({path: store_path});
  fabric_client.setStateStore(state_store);

  var crypto_suite = Fabric_Client.newCryptoSuite();
  var crypto_store = Fabric_Client.newCryptoKeyStore({path: store_path});
  crypto_suite.setCryptoKeyStore(crypto_store);
  fabric_client.setCryptoSuite(crypto_suite);

  var user_from_store = await fabric_client.getUserContext('user1', true);
  if (user_from_store && user_from_store.isEnrolled()) {
    console.log('Successfully loaded user1 from persistence');
    member_user = user_from_store;
  } else {
    throw new Error('Failed to get user1.... run registerUser.js');
  }

  var query_responses = await channel.queryBlock(Number(req.query.number),peer,null);
  res.send(query_responses)
}catch (e) {
  console.error('Failed to query successfully :: ' + e);
}
});
```

라우터 함수/queryBlockInfo: 채널에서 블록 번호를 사용하여 블록을 쿼리한다.

- 이 함수는 /queryBlockInfo의 GET 요청을 처리한다.
- 먼저 URL 조회 매개변수에서 number라는 값을 가져와 콘솔로 출력한다.
- 다음 위와 유사하게 Fabric_Client 인스턴스를 만들고 사용자 컨텍스트를 로드하며 암호화 키트와 키 저장소를 Fabric 클라이언트에 설정한다.
- 그런 다음, 채널(channel)을 호출합니다.queryBlock(Number(req.query.number), peer, null) 방법으로 블록 번호에 따라 특정 블록의 상세 정보를 조회하고 조회 결과를 JSON 응답으로 클라이언트에 보낸다.

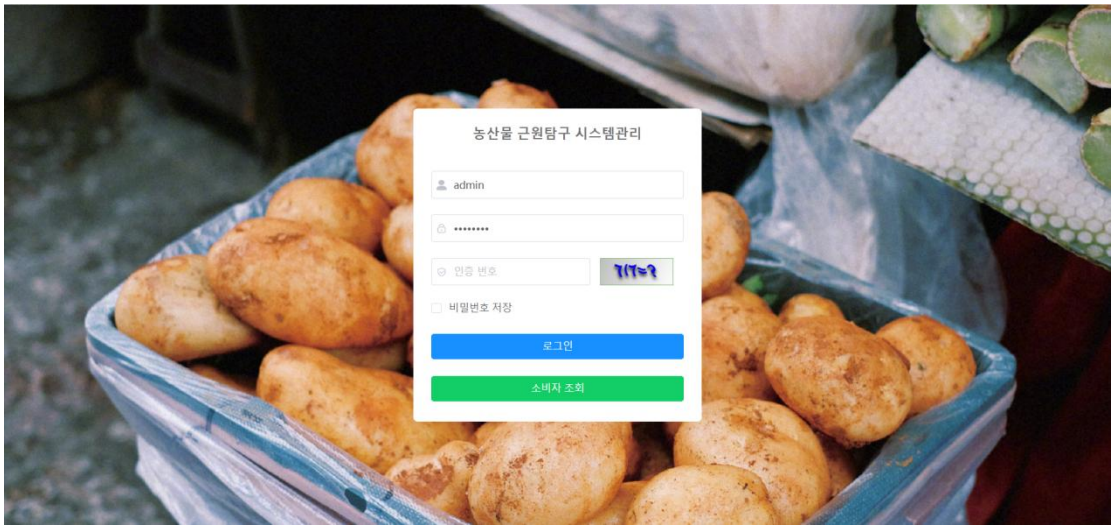
```
router.get("/queryBlockBuHash",async function(req,res){
console.log("number   "+req.query.hash)
try {
  var state_store = await Fabric_Client.newDefaultKeyValueStore({path: store_path});
  fabric_client.setStateStore(state_store);

  var crypto_suite = Fabric_Client.newCryptoSuite();
  var crypto_store = Fabric_Client.newCryptoKeyStore({path: store_path});
  crypto_suite.setCryptoKeyStore(crypto_store);
  fabric_client.setCryptoSuite(crypto_suite);

  // get the enrolled user from persistence, this user will sign all requests
  var user_from_store = await fabric_client.getUserContext('user1', true);
  if (user_from_store && user_from_store.isEnrolled()) {
    console.log('Successfully loaded user1 from persistence');
    member_user = user_from_store;
  } else {
    //error
    throw new Error('Failed to get user1.... run registerUser.js');
  }
  var query_responses = await channel.queryBlockByHash(new Buffer(req.query.hash,"hex"),peer);
  res.send(query_responses)
}catch (e) {
  console.error('Failed to query successfully :: ' + e);
}
});
```

라우터 함수/queryBlockBuHash: 해시 값 가져오기

- 이 함수는 /queryBlockBuHash의 GET 요청을 처리한다.
- 위의 함수와 유사하게 URL 쿼리 매개변수에서 hash라는 값을 가져와 콘솔로 인쇄한다.
- 그런 다음 Fabric_Client 인스턴스를 만들고 사용자 컨텍스트를 로드하며 암호화 키트와 키 저장소를 Fabric 클라이언트에 설정한다.

■ 그런 다음, 채널(channel)을 호출합니다.queryBlockByHash(new Buffer(req.query.hash, "hex", peer) 메서드는 블록 해시 값에 따라 특정 블록의 세부 정보를 조회하고 조회 결과를 JSON 응답으로 클라이언트에 보낸다.

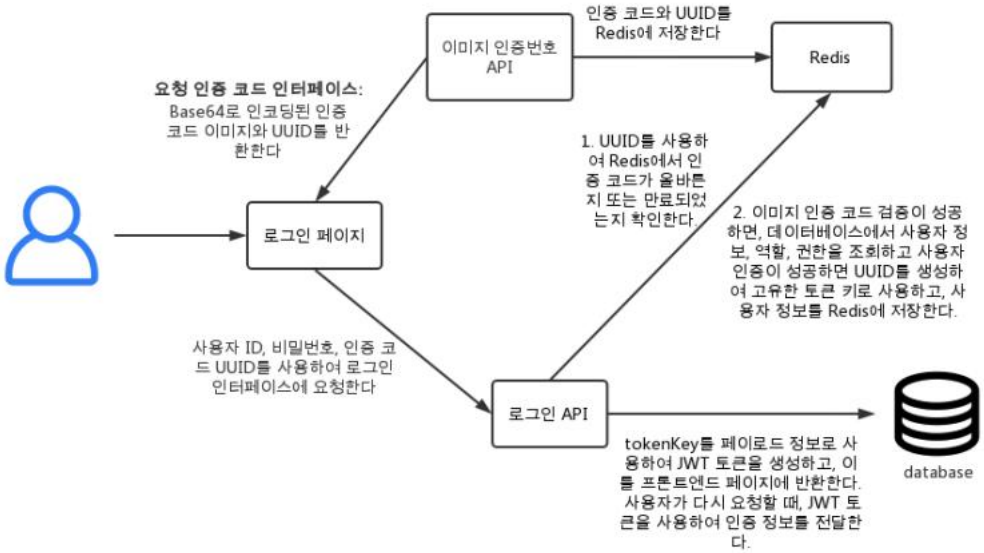
4.3. 실현 페이지



Page1.로그인 페이지

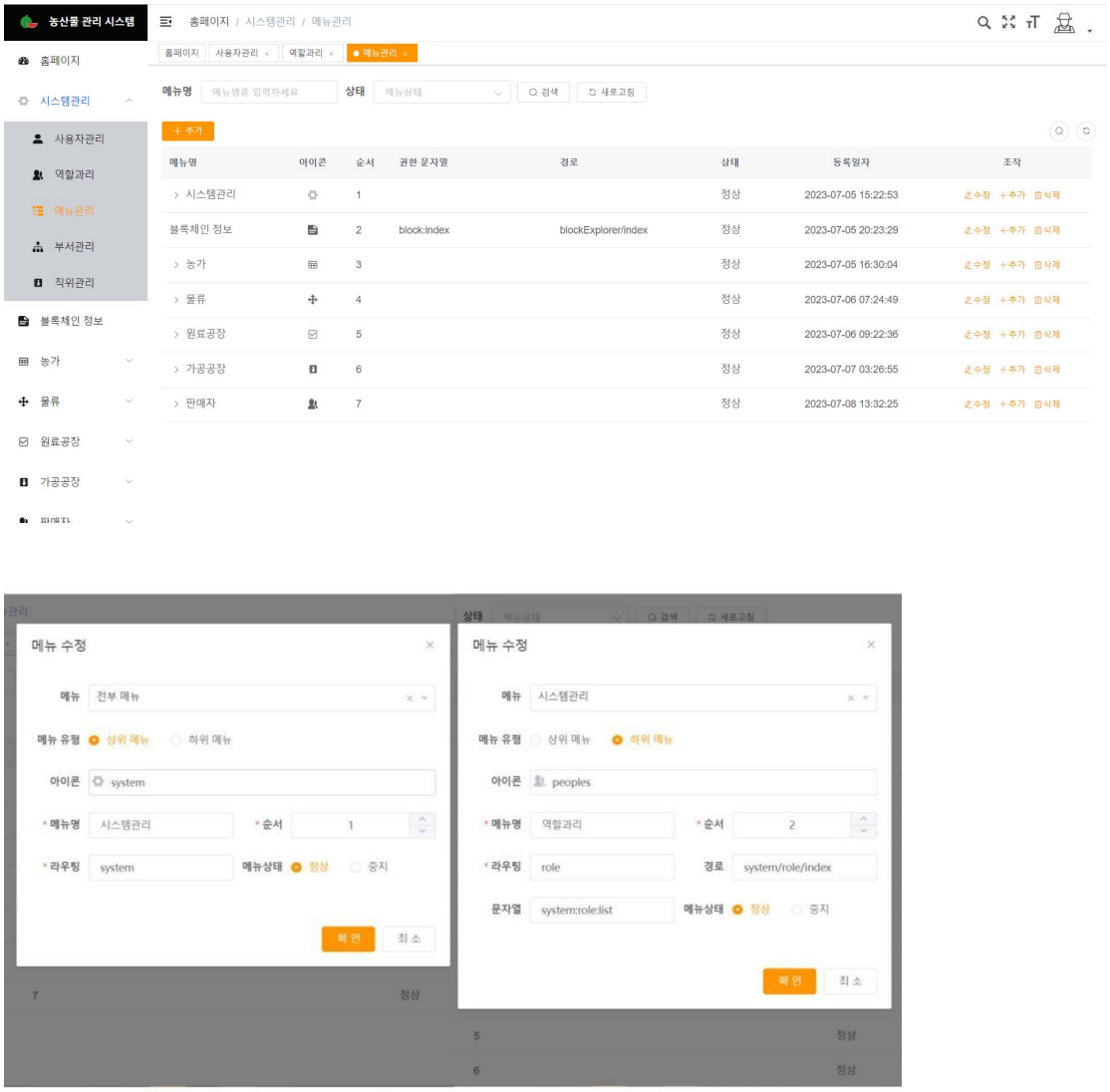
로그인 페이지의 코드는 로그인 폼과 관련된 이벤트 처리 함수를 포함하고 있다. 로그인 폼에는 사용자 이름, 비밀번호, 인증 번호 및 "비밀번호 저장" 옵션이 포함되어 있다. 사용자가 올바른 사용자 이름, 비밀번호 및 인증 번호를 입력한 후 "로그인" 버튼을 클릭하여 로그인 작업을 수행할 수 있다.

다음은 로그인 과정 그림이다.



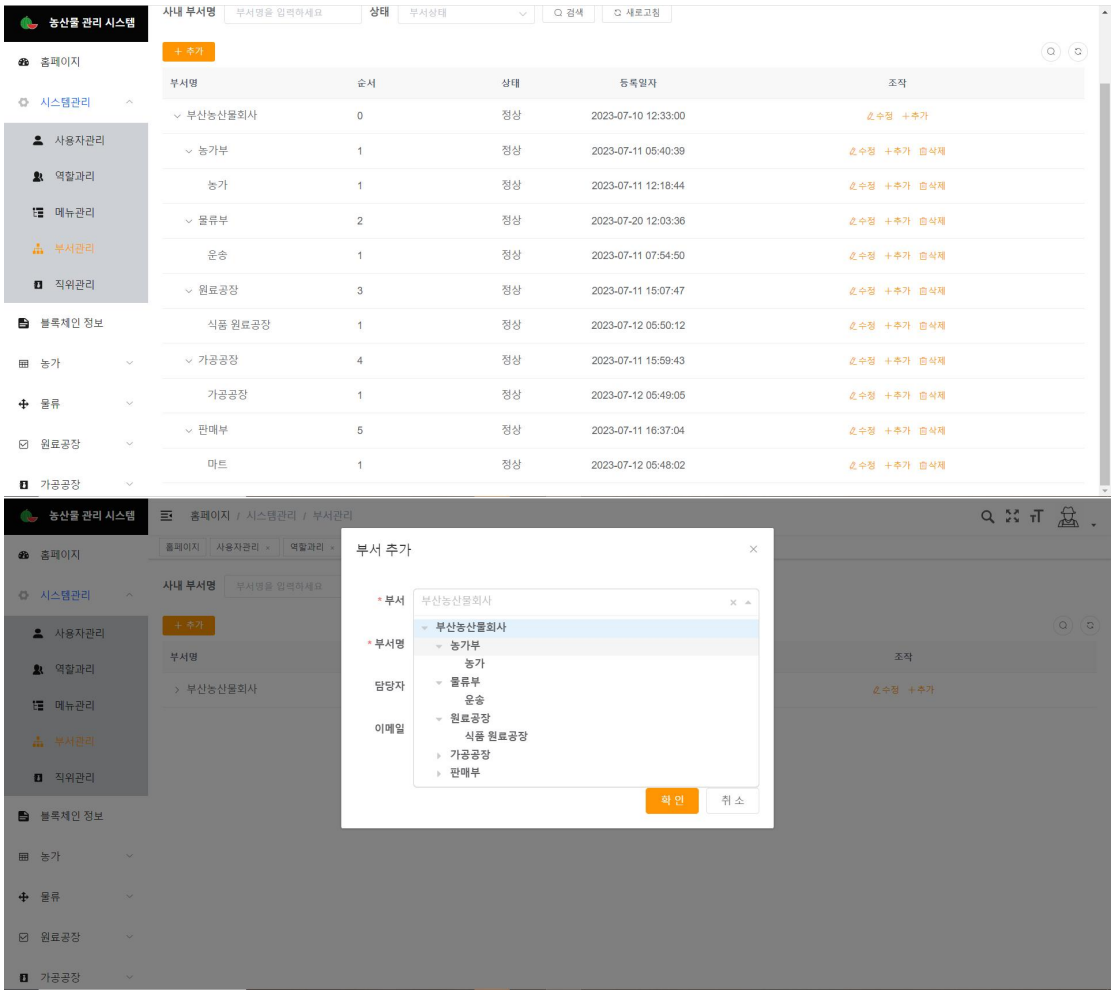

```java
// 로그인 처리를 위한 메서드이다.
// 사용자가 HTTP POST 요청으로 로그인 정보(LoginBody)를 전송하면,
// 해당 정보를 기반으로 사용자를 인증하고,
// 성공 시에는 토큰을 발급하여 반환한다.
@PostMapping("/login")
public AjaxResult login(@RequestBody LoginBody loginBody)
{
    AjaxResult ajax = AjaxResult.success();
    // 토큰 생성
    String token = loginService.login(loginBody.getUsername(), loginB
            loginBody.getUuid());
    ajax.put(Constants.TOKEN, token);
    return ajax;
}

/**
 * 사용자 정보를 가져오는 메서드이다.
 *
 * @return 사용자 정보
 */
@GetMapping("getInfo")
public AjaxResult getInfo()
{
    LoginUser loginUser = tokenService.getLoginUser(ServletUtils.getR
    SysUser user = loginUser.getUser();
    // 역할 목록
    Set<String> roles = permissionService.getRolePermission(user);
    // 권한 목록
    Set<String> permissions = permissionService.getMenuPermission(use
    AjaxResult ajax = AjaxResult.success();
    ajax.put("user", user);
    ajax.put("roles", roles);
    ajax.put("permissions", permissions);
    return ajax;
}

/**
 * 라우터 정보를 가져오는 메서드이다.
 *
 * @return 라우터 정보
 */
@GetMapping("getRouters")
public AjaxResult getRouters()
{
    LoginUser loginUser = tokenService.getLoginUser(ServletUtils.getR
    // 사용자 정보
    SysUser user = loginUser.getUser();
    List<SysMenu> menus = menuService.selectMenuTreeByUserId(user.get
    return AjaxResult.success(menuService.buildMenus(menus));
}
```

```javascript
// 로그인 메서드
无用法
export function login(username, password, code, uuid) {
    const data : {code: any, password: any, username: any, uuid: a
        username,
        password,
        code,
        uuid
    }
    return request({
        url: '/login',
        method: 'post',
        data: data
    })
}

// 사용자 세부 정보 가져오기
无用法
export function getInfo() {
    return request({
        url: '/getInfo',
        method: 'get'
    })
}

// 로그아웃 메서드
无用法
export function logout() {
    return request({
        url: '/logout',
        method: 'post'
    })
}

// 인증 코드 가져오기
无用法
export function getCodeImg() {
    return request({
        url: '/captchaImage',
        method: 'get'
    })
}
```



Page2.메뉴 관리 페이지

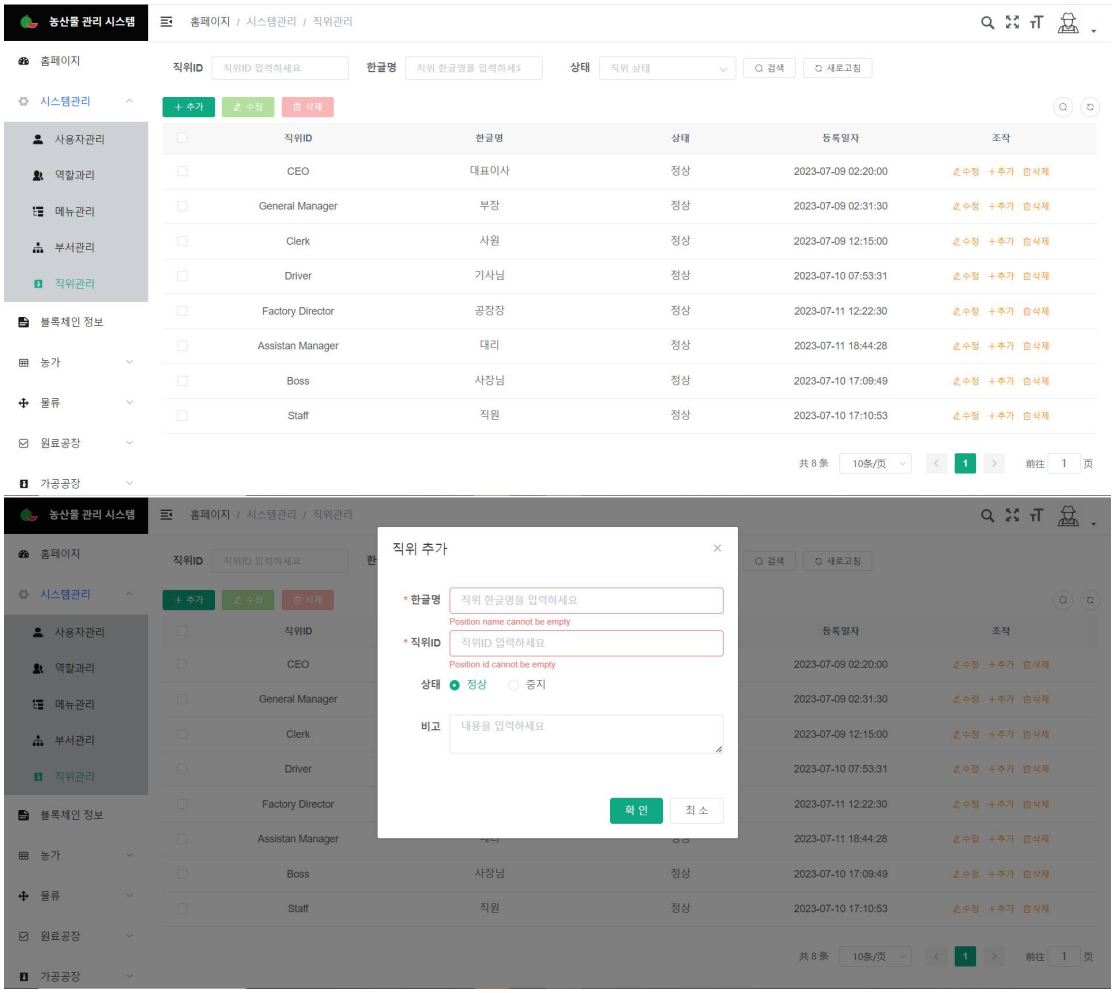
라우팅 : 웹 내비게이션 추가로사 용 한다.페이지를 이동할 때 서버에 요청해 갱신하는 것이 아니라, 미리 해당 페이지를 구성해놓고 클라이언트의 라우팅을 이용하여 화면을 갱신한다.이러한 방식을 SPA (Single Page Application) 라고도 한다.라우팅을 이용하면 매끄럽게 화면 전환을 할 수 있다. 메뉴의 "권한 문자열(Permission)" 는 권한을 제어하는 키 필드이다. 이름 지정 규격은 다음과 같습니다. 모듈: 기능: 동작. 예: sys: user:add



Page3.부서 관리 페이지

농산물 회사에 농산물을 관리할 수 있는 서비스를 제공하기 위해 회사 부서를 제작을 한다.

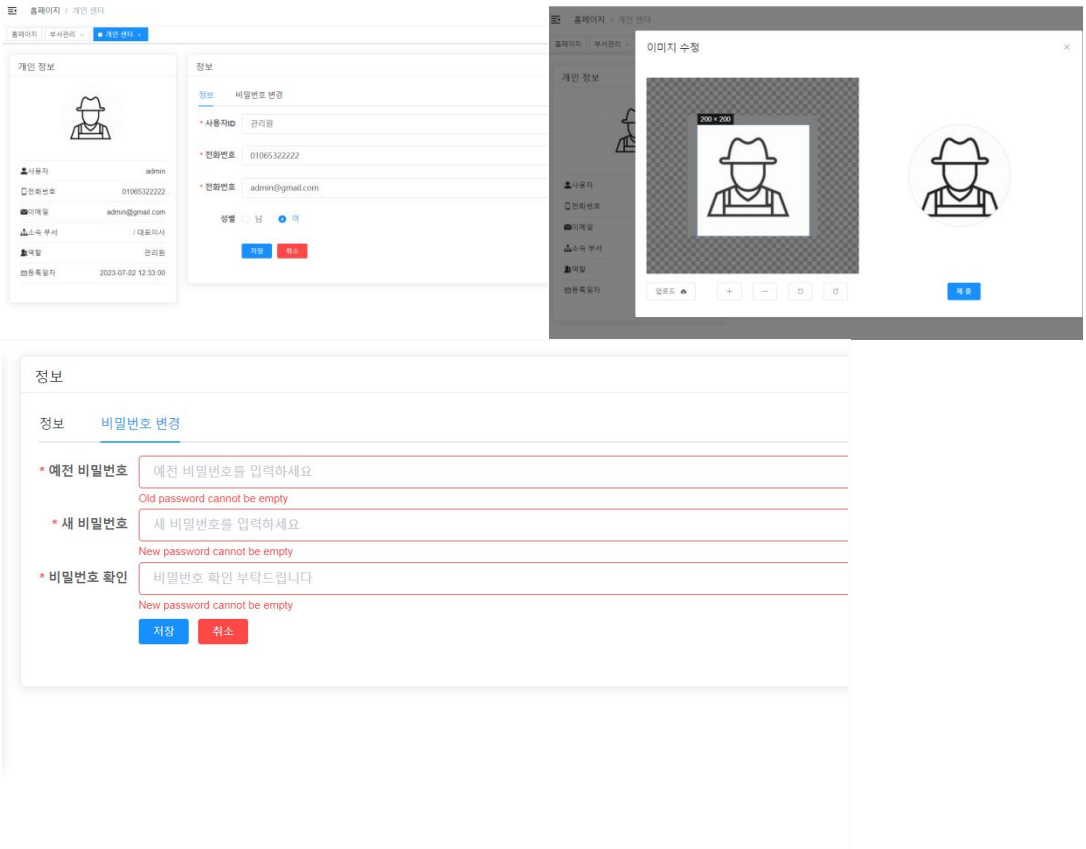
관리자는 페이지에 회사 및 부서를 추가/수정/삭제할 수 있다



Page4.직위 관리 페이지

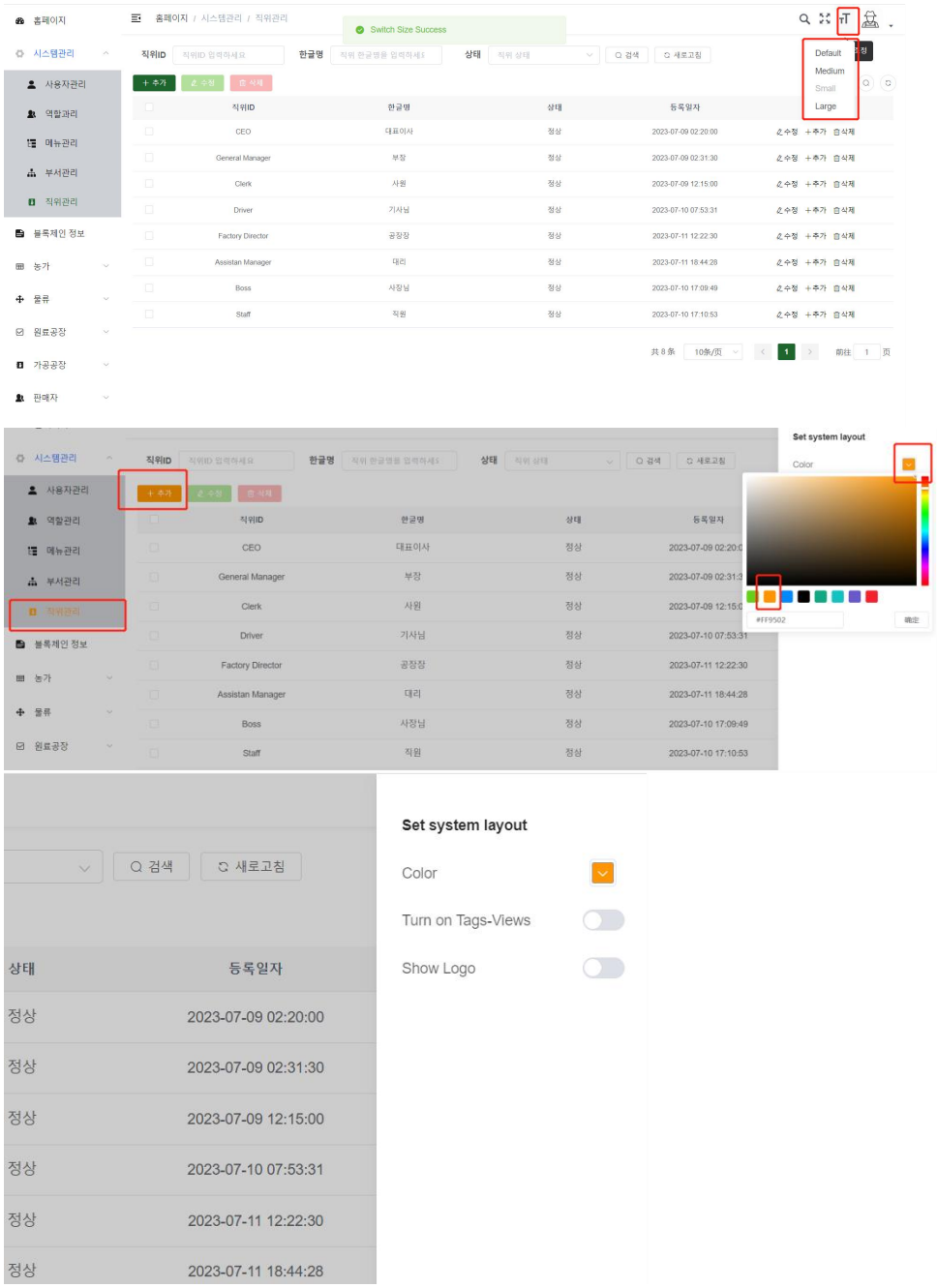
농산물 회사에 농산물을 관리할 수 있는 서비스를 제공하기 위해 회사 직위를 제작을 한다.

관리자는 페이지에 회사 및 직위를 추가/수정/삭제할 수 있다



Page5.개인 정보 수정 페이지

사용자는 자신의 개인 정보를 확인하고 프로필 사진을 업로드하며 비밀번호를 변경할 수 있다.



Page9.Layout페이지

Element ui를 사용하여 사용자에게 편리한 인터페이스를 제공한다.
검색:사용자는 검색을 통해 빠르게 페이지로 이동할 수 있다.
font size:글자 크기를 조절할 수 있다
color:일부 버튼의 색상을 변경할 수 있다.
Turn on Tags-Views:탭을 열거나 숨길 수 있다
logo:로고도 숨길 수 있다