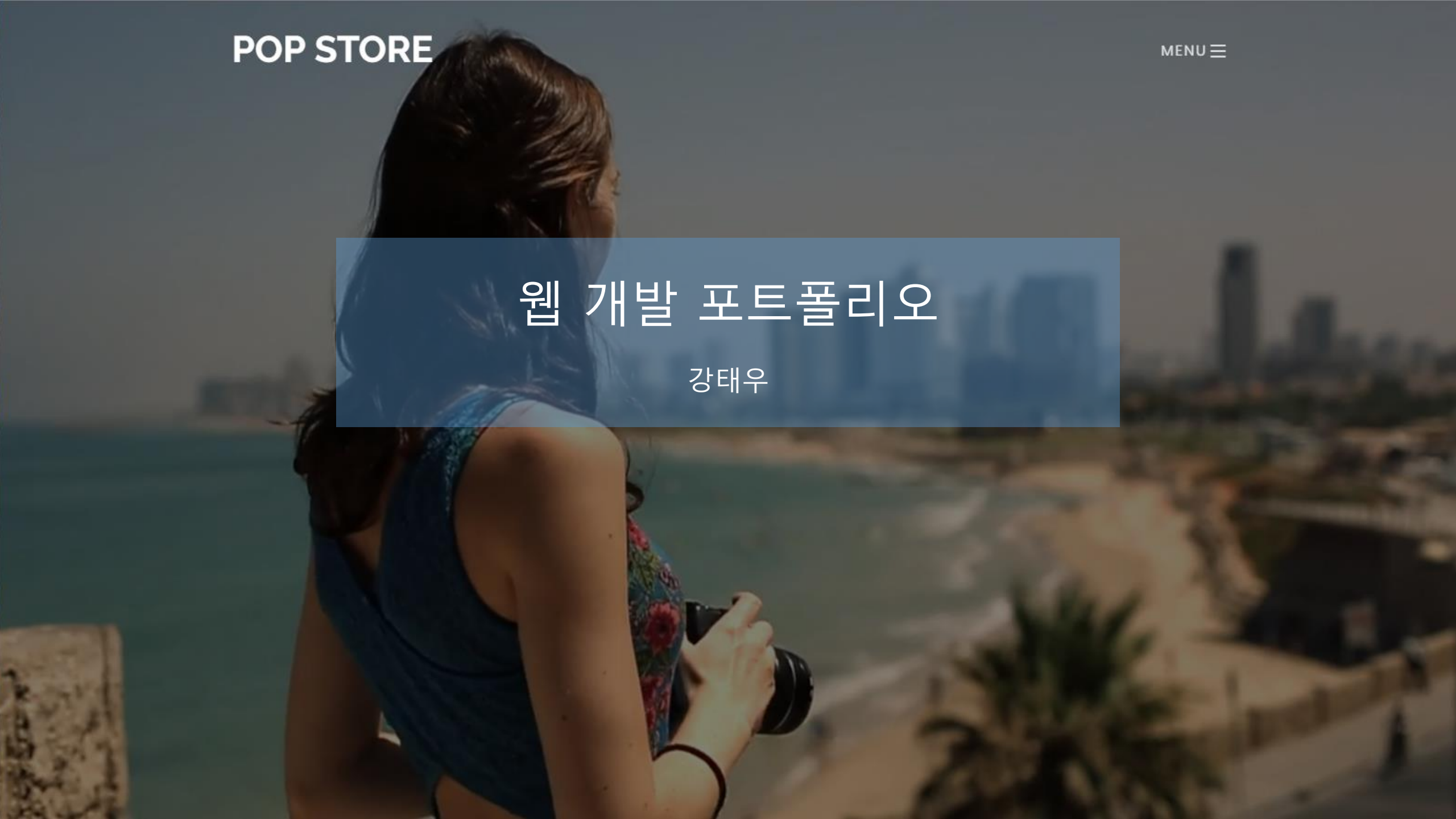


웹 개발 포트폴리오

강태우

목차

팀 소개

연구흠

멘토

박두현

프론트 엔드
백 엔드
블록체인 연동

강태우

백 엔드
프론트 엔드
체인코드

이의영

프론트 엔드
UI UX
블록체인 연동

정한결

비즈니스 모델링
UI UX
블록체인 망 구성

본인의 역할

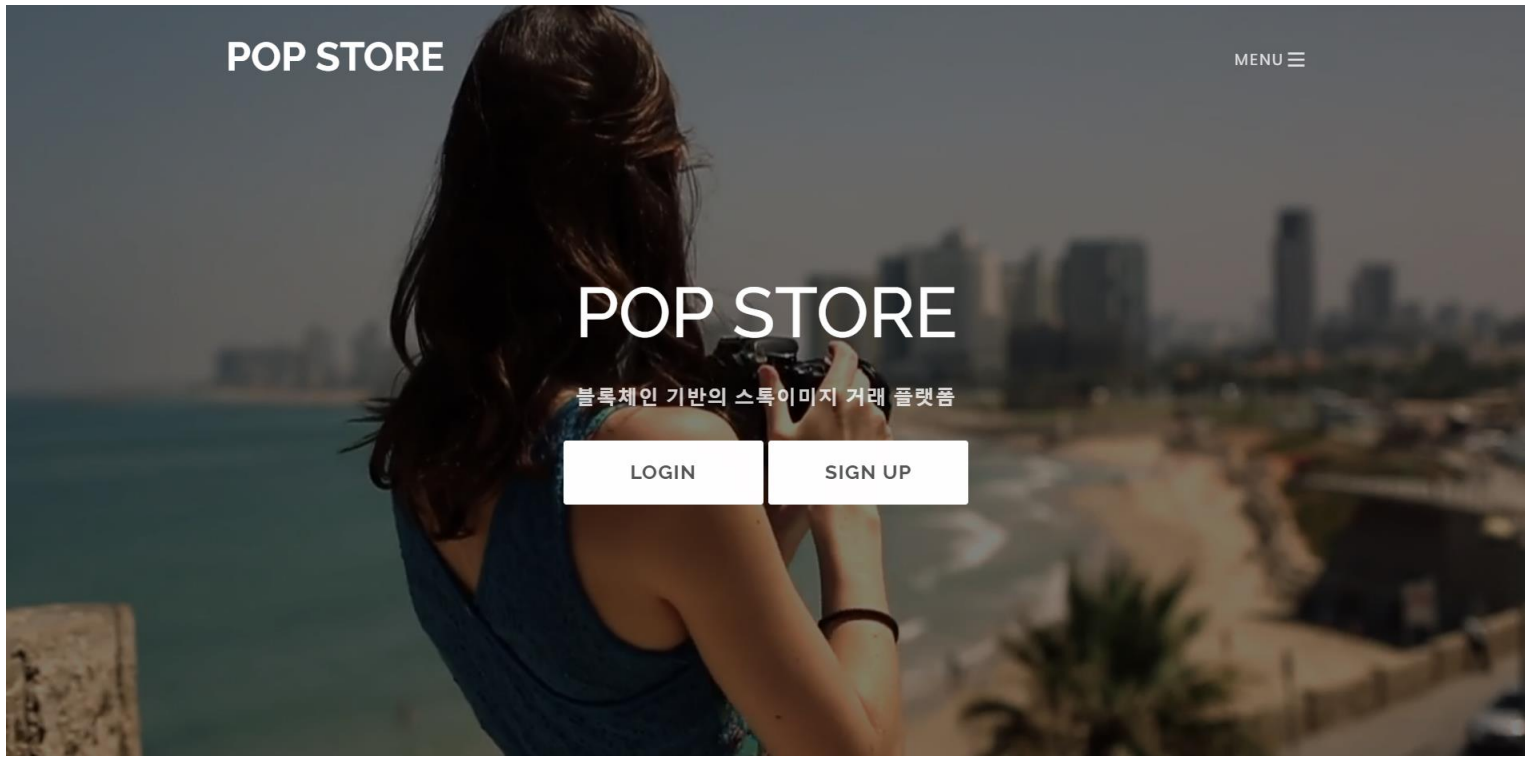
강태우

- DB 설계(DB설계서)
- GUI 설계(화면설계서)
- 회원가입
- 아이디 중복검사
- 닉네임 중복검사
- 로그인
- 아이디 찾기
- 비밀번호 찾기
- 회원정보 수정
- 프로필사진 등록
- 이미지 등록 시 워터마크 삽입
- 이미지 거래 게시판 구성
- 이미지 거래 기능
- 글 쓰기
- 댓글 작성,수정,삭제
- 이미지 등록, 거래 시 블록체인에 등록

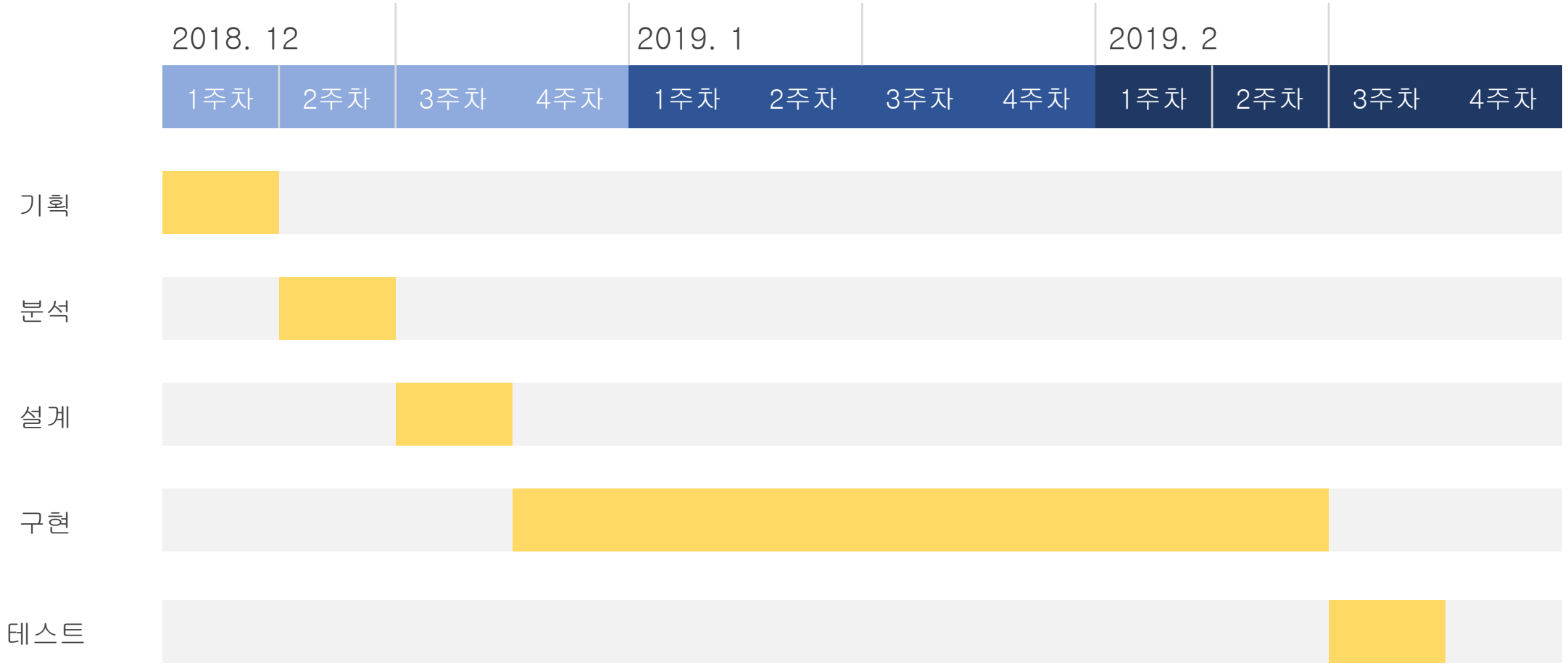
서비스 개요

[블록체인 기반 스톡이미지 거래 플랫폼]

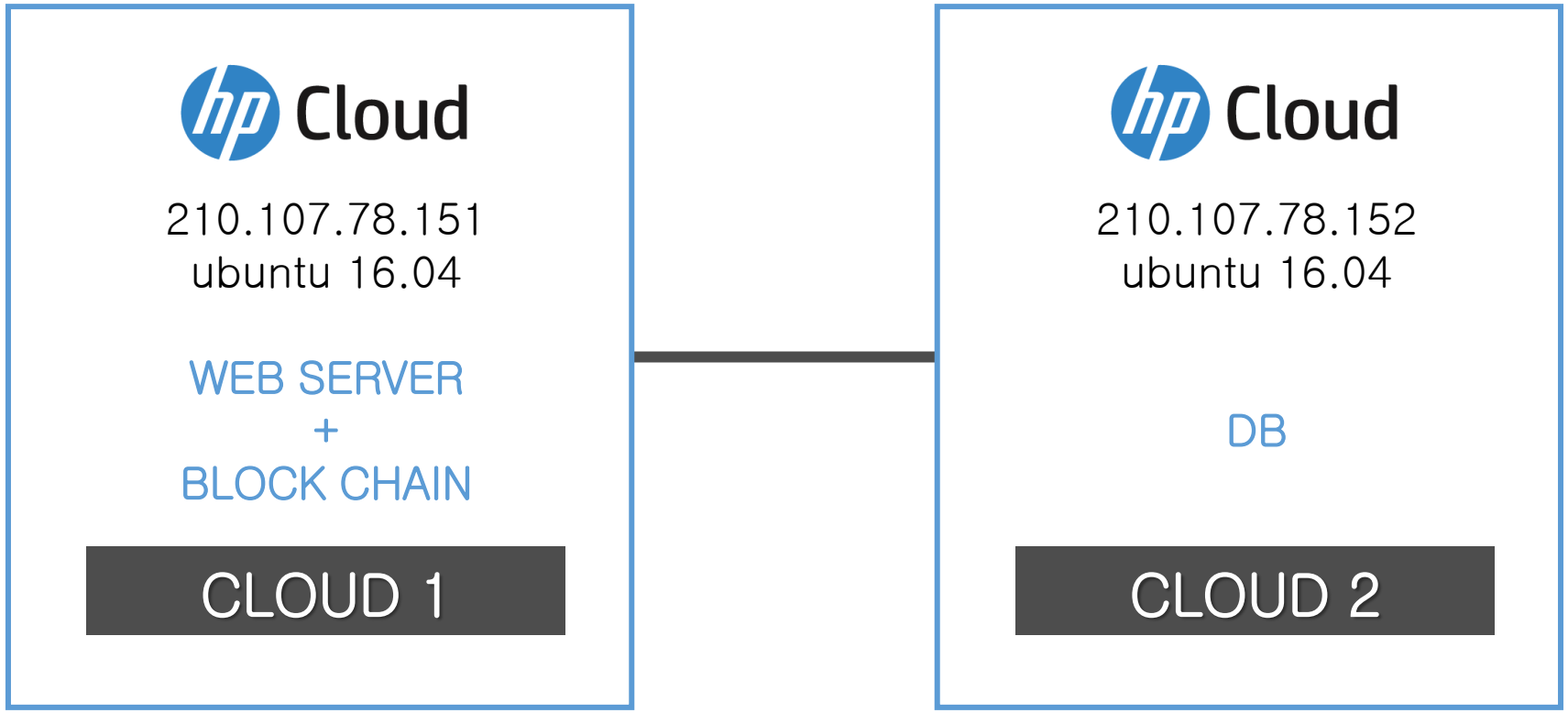
(Popular-Images + Store)



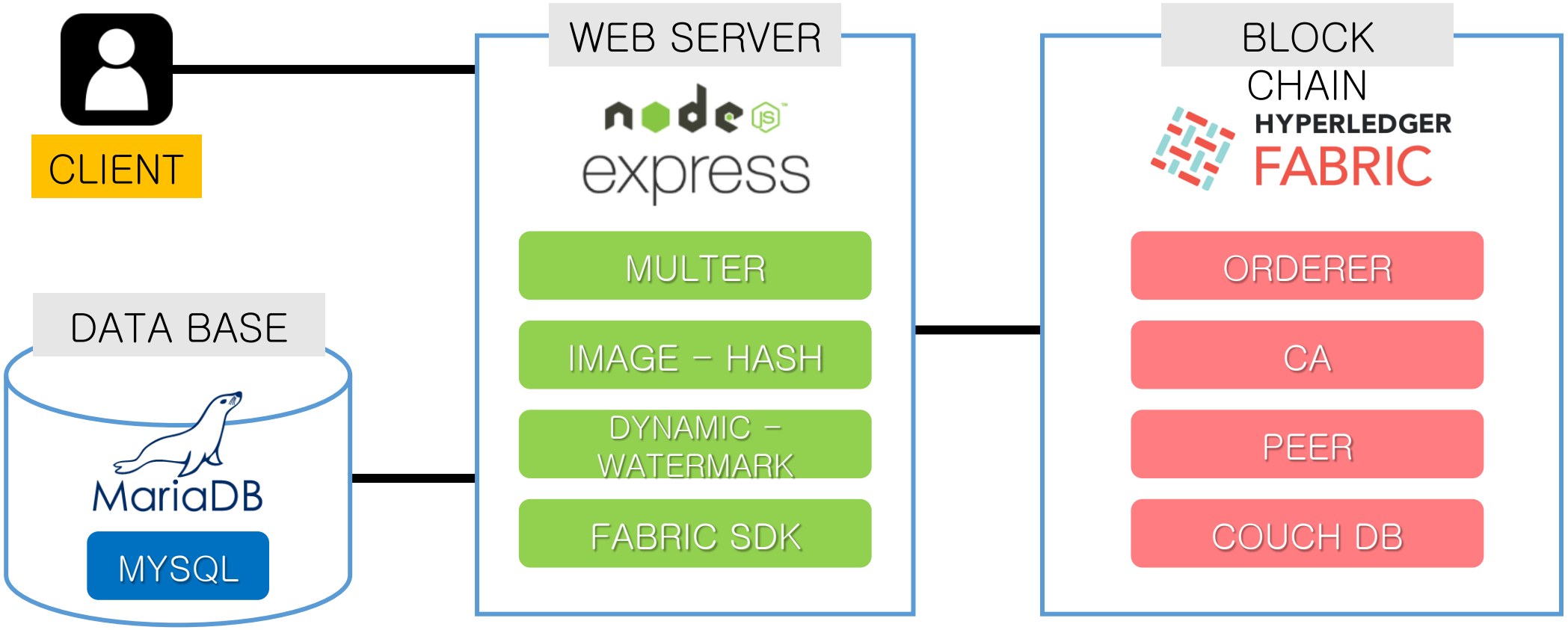
프로젝트 일정표


2018. 12
2019. 1
2019. 2
1주차
2주차
3주차
4주차
1주차
2주차
3주차
4주차
1주차
2주차
3주차
4주차
기획
분석
설계
구현
테스트

INFRA


Cloud
210.107.78.151
ubuntu 16.04
WEB SERVER
+
BLOCK CHAIN
CLOUD 1
Cloud
210.107.78.152
ubuntu 16.04
DB
CLOUD 2

Architecture



적용 기술

Node JS

1) 프론트엔드부터 백엔드까지 통합 개발 가능

2) 하이퍼레저 패브릭 SDK 사용가능

HyperLedger Fabric

1) 이더리움 사용 시, 트랜잭션이 발생하면 수수료(pee) 발생

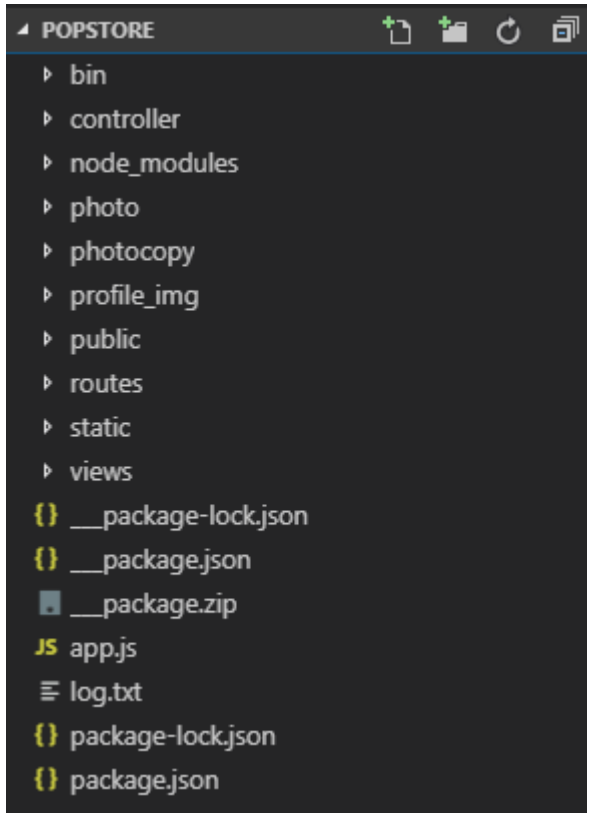
2) 서비스 확장 고려 시, 데이터를 분류에 하기 용이

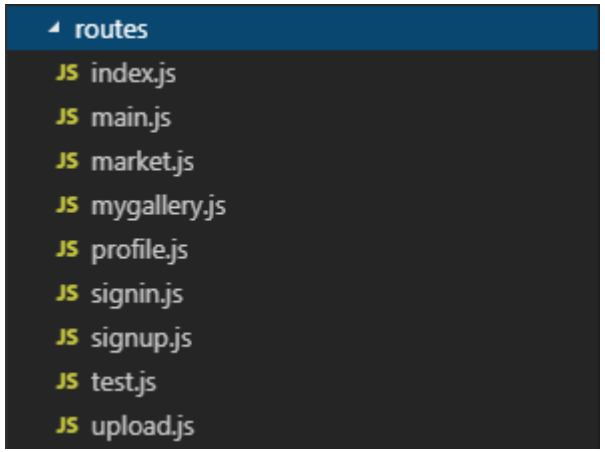
MySQL

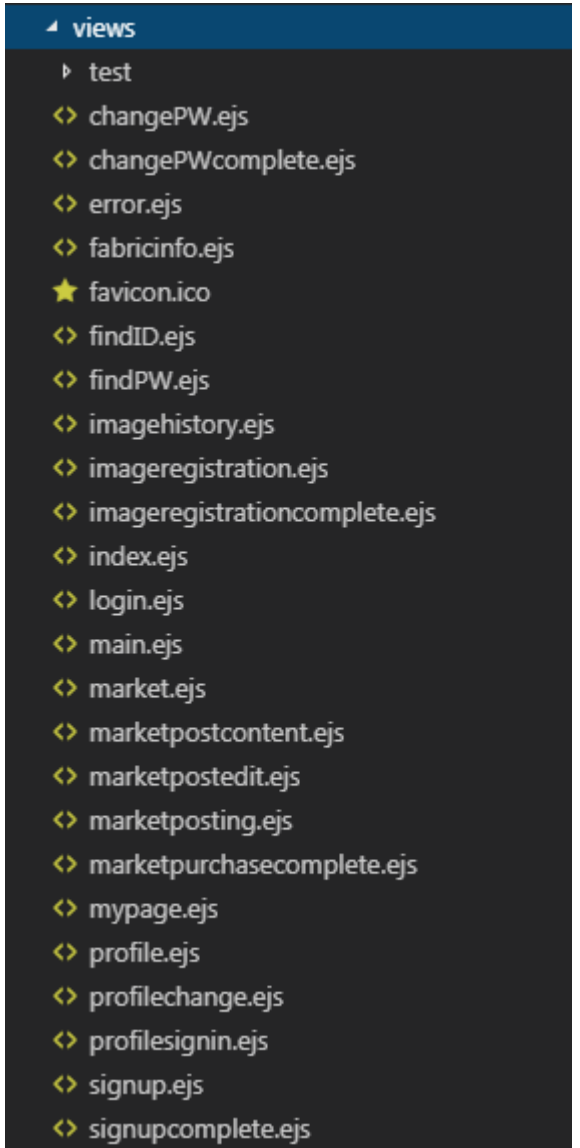
오픈소스 기반이며 다양한 운영체제에서 사용 가능하고,

널리 알려진 표준 SQL형식을 지원

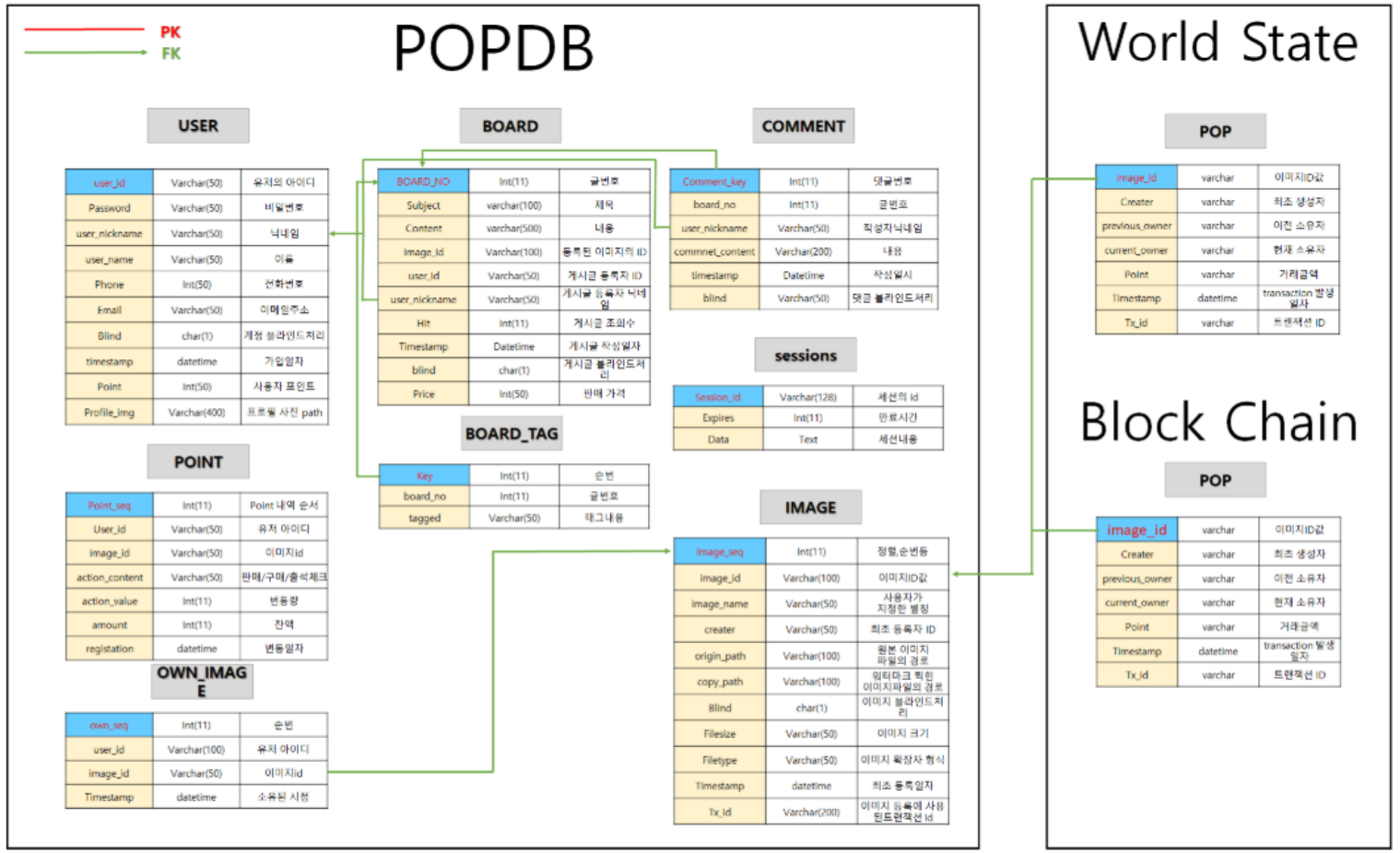
패키지 구조


POPSTORE
bin
controller
node_modules
photo
photocopy
profile_img
public
routes
static
views
__package-lock.json
__package.json
__package.zip
app.js
log.txt
package-lock.json
package.json

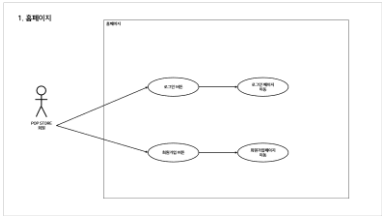

routes
index.js
main.js
market.js
mygallery.js
profile.js
signin.js
signup.js
test.js
upload.js

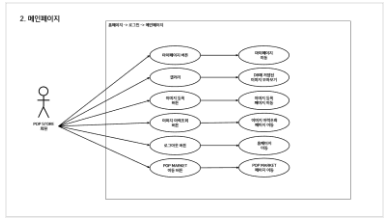

views
test
changePW.ejs
changePWcomplete.ejs
error.ejs
fabricinfo.ejs
favicon.ico
findID.ejs
findPW.ejs
imagehistory.ejs
imageregistration.ejs
imageregistrationcomplete.ejs
index.ejs
login.ejs
main.ejs
market.ejs
marketpostcontent.ejs
marketpostedit.ejs
marketposting.ejs
marketpurchasecomplete.ejs
mypage.ejs
profile.ejs
profilechange.ejs
profilesignin.ejs
signup.ejs
signupcomplete.ejs

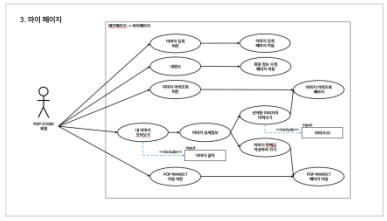
DB 구조

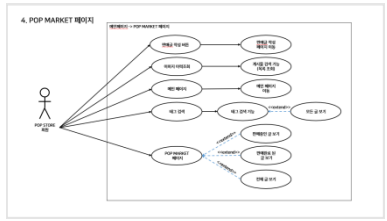

PK
FK
POPDB
USER
user_id Varchar(50) 유저의 아이디
Password Varchar(50) 비밀번호
user_nickname Varchar(50) 닉네임
user_name Varchar(50) 이름
Phone Int(50) 전화번호
Email Varchar(50) 이메일주소
Blind char(1) 계정 블라인드처리
timestamp datetime 가입일자
Point Int(50) 사용자 포인트
Profile_img Varchar(400) 프로필 사진 path
BOARD
BOARD_NO Int(11) 글번호
Subject varchar(100) 제목
Content varchar(500) 내용
Image_Id Varchar(100) 등록된 이미지의 ID
user_Id Varchar(50) 게시글 등록자 ID
user_nickname Varchar(50) 게시글 등록자 닉네임
Hit Int(11) 게시글 조회수
Timestamp Datetime 게시글 작성일자
blind char(1) 게시글 블라인드처리
Price Int(50) 판매 가격
COMMENT
Comment_key Int(11) 댓글번호
board_no Int(11) 글번호
user_nickname Varchar(50) 작성자닉네임
comment_content Varchar(200) 내용
timestamp Datetime 작성일시
blind Varchar(50) 댓글 블라인드처리
sessions
Session_id Varchar(128) 세션의 id
Expires Int(11) 만료시간
Data Text 세션내용
BOARD_TAG
Key Int(11) 순번
board_no Int(11) 글번호
tagged Varchar(50) 태그내용
POINT
Point_seq Int(11) Point 내역 순서
User_Id Varchar(50) 유저 아이디
image_id Varchar(50) 이미지id
action_content Varchar(50) 판매/구매/출석체크
action_value Int(11) 변동량
amount Int(11) 잔액
registation datetime 변동일자
OWN_IMAGE
own_seq Int(11) 순번
user_id Varchar(100) 유저 아이디
image_id Varchar(50) 이미지id
Timestamp datetime 소유된 시점
IMAGE
Image_seq Int(11) 정렬,순번등
image_id Varchar(100) 이미지ID값
image_name Varchar(50) 사용자가 지정한 별칭
creater Varchar(50) 최초 등록자 ID
origin_path Varchar(100) 원본 이미지 파일의 경로
copy_path Varchar(100) 워터마크 적힌 이미지파일의 경로
Blind char(1) 이미지 블라인드처리
Filesize Varchar(50) 이미지 크기
Filetype Varchar(50) 이미지 확장자 형식
Timestamp datetime 최초 등록일자
Tx_Id Varchar(200) 이미지 등록에 사용된 트랜잭션 id
World State
POP
image_id varchar 이미지ID값
Creater varchar 최초 생성자
previous_owner varchar 이전 소유자
current_owner varchar 현재 소유자
Point varchar 거래금액
Timestamp datetime transaction 발생일자
Tx_id varchar 트랜잭션 ID
Block Chain
POP
image_id varchar 이미지ID값
Creater varchar 최초 생성자
previous_owner varchar 이전 소유자
current_owner varchar 현재 소유자
Point varchar 거래금액
Timestamp datetime transaction 발생일자
Tx_id varchar 트랜잭션 ID

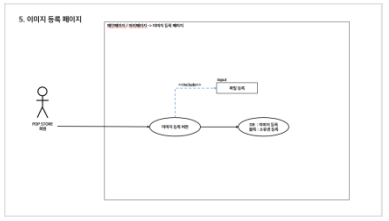
USE CASE

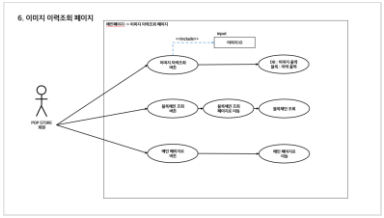

1. 홈페이지

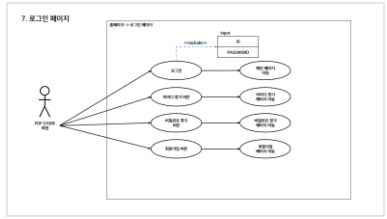

2. 메인페이지

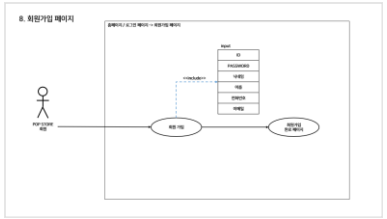

3. 마이 페이지

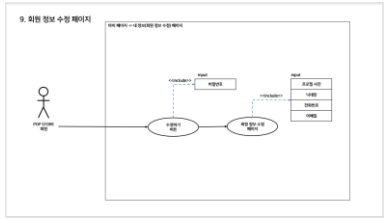

4. POP MARKET 페이지

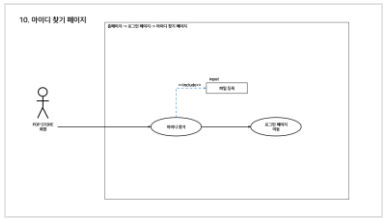

5. 이미지 등록 페이지

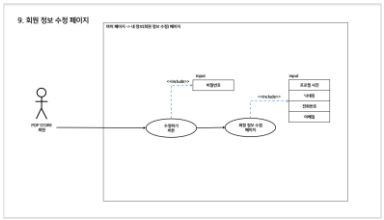

6. 이미지 이력조회 페이지

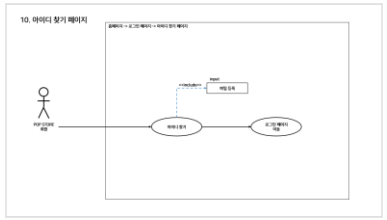

7. 로그인 페이지

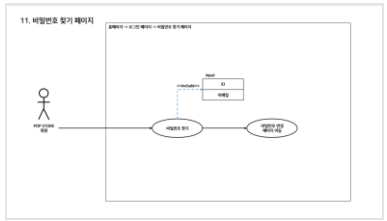

8. 회원가입 페이지

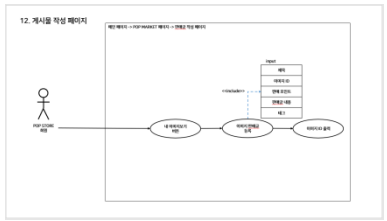

9. 회원 정보 수정 페이지

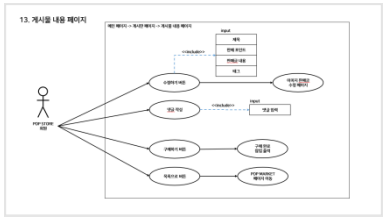

10. 아이디 찾기 페이지


9. 회원 정보 수정 페이지

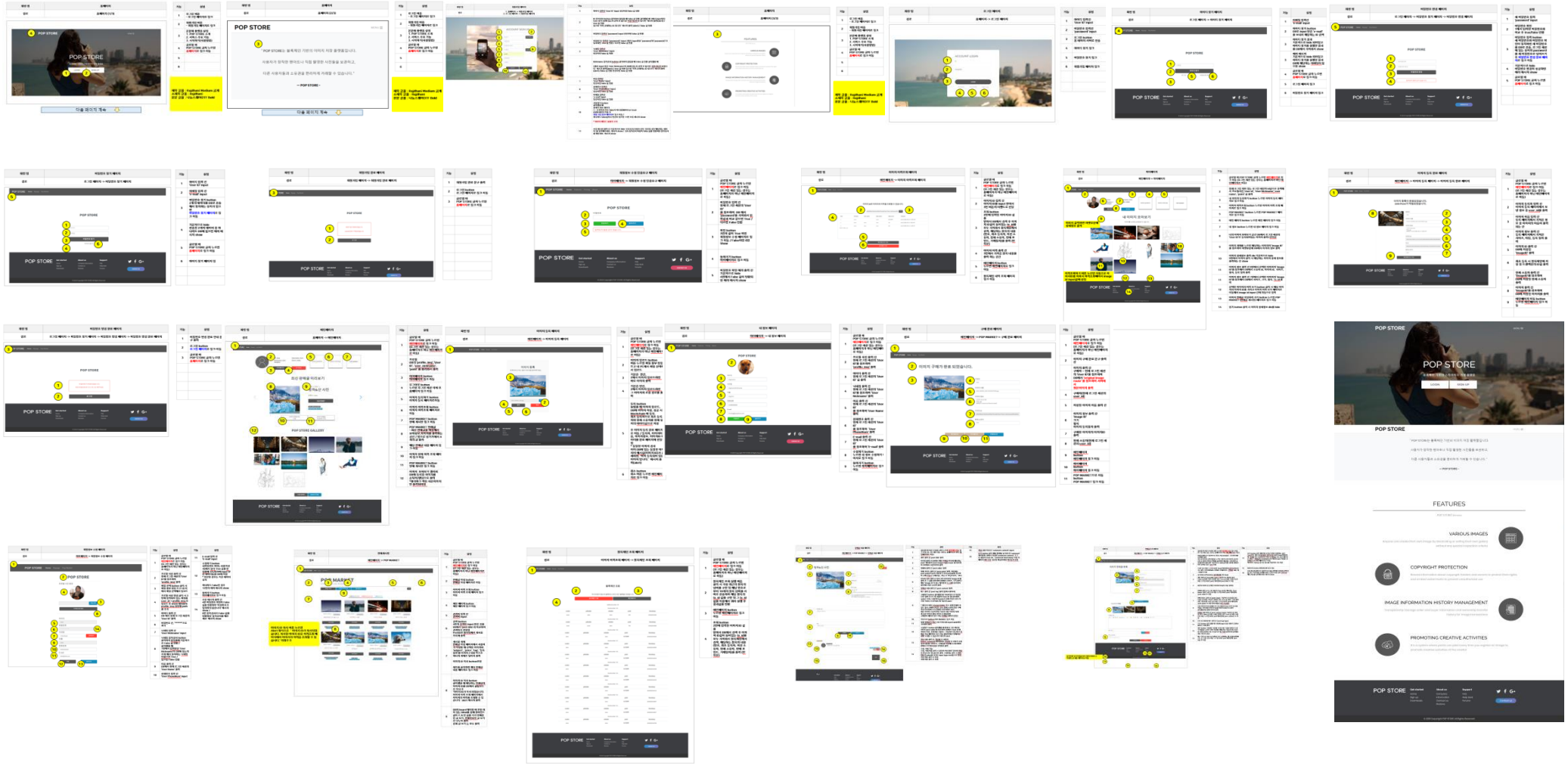

10. 아이디 찾기 페이지


11. 비밀번호 찾기 페이지


12. 게시물 작성 페이지

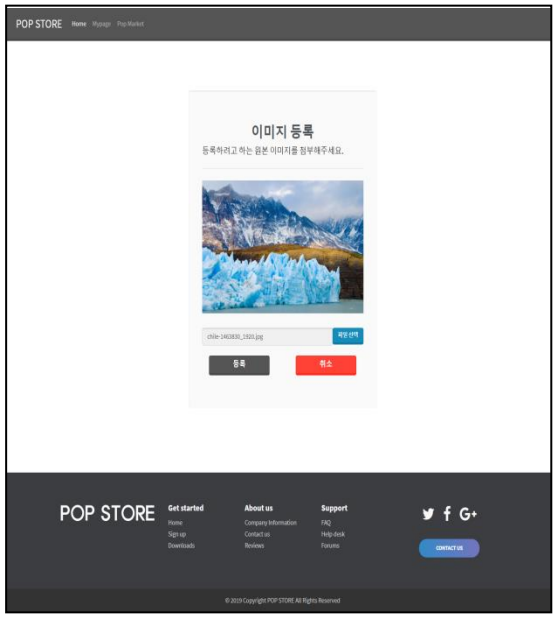

13. 게시물 내용 페이지

화면 구조


POP STORE
POP MARKET
FEATURES
VARIOUS IMAGES
COPYRIGHT PROTECTION
IMAGE INFORMATION HISTORY MANAGEMENT
PROMOTING CREATIVE ACTIVITIES

주요 기능

이미지 등록 ▶ 이미지 거래 ▶ 이미지 이력조회



업로드한 이미지를 photo/ 경로에 저장
이미지 파일의 Hash값 (image_id)을 생성하여 DB에 저장

multer
이미지 업로드 및 저장

image-Hash
이미지 파일의 Hash값 생성

생성된 이미지의 Hash값을 블록체인에 기록 (couch DB)

Chaincode
stub.PutState

업로드한 이미지에 WATERMARK를 입힌 사본 이미지 생성

dynamic-watermark
이미지 워터마크 생성

이미지 등록

```
$(function(){
  $("#enroll").click(function(){
    var form = $('form')[0];
    var formData = new FormData(form);
    var file_type = formData.get('imgFile').type;
    if (file_type != 'image/jpeg' && file_type != 'image/png'){
      alert("파일의 형식이 JPG또는 jpeg가 아닙니다.");
      return false;
    }
    else{
      $.ajax({
        url: '/upload/create',
        processData: false,
        contentType: false,
        data: formData,
        type: 'POST',
        success: function(result){
          if (result=="imageundefined") {
            alert("이미지를 선택해주세요");
          }else if (result =="typeError"){
            alert("파일의 형식이 JPG또는 jpeg가 아닙니다.");
          }else if (result =="ER_DUP_ENTRY"){
            alert("이미 등록되어 있는 이미지 입니다.");
          }else{
            $.ajax({
              url: '/imagehistory/add_tuna/',
              data: result,
              type: 'POST',
              success: function(result2){
                result_json = {
                  image_id : result.image_id,
                  txid : result2
                }
                $.ajax({
                  url: '/imagehistory/inserttxid/',
                  data: result_json,
                  type: 'POST',
                  success: function(result3){
                    if(result3=="success"){
                      location.href="/upload/imageregistrationcomplete";
                    }
                    else{
                      alert("실패");
                    }
                  }
                });
```

Imageregistration.ejs

1. 업로드하려는 이미지의 타입을 확인하고 image/jpeg 또는 image/png 타입이 아니라면 요청을 취소시킨다.

2. 사용자가 선택한 이미지 파일을 AJAX를 이용해서 upload.js에 등록되어 있는 경로인 /create로 전송한다.

3. 전송 후 통신 성공 시 받은 결과를 이용해서 그에 맞는 오류를 표출하고, 오류가 없을 시 image_id 값을 사용하여 블록체인에 등록하는 함수를 실행시킨다.

이미지 등록

```
router.post('/create', upload.single("imgFile"), function(req, res, next) {
    let file = req.file;

    if (file ==undefined){
        res.send("imageundefined");
        // res.redirect('/upload');
    }else if (file.mimetype != 'image/jpeg' && file.mimetype !='image/png'){
        res.send("typeError");
    }else{
        let result = {
            originalname: file.originalname,
            filename: file.filename,
            path: file.path,
            size: file.size,
        };
        imageHash(file.path, 20, true, function(error,image_id){
            if(error){
                console.log("image hash make error!!!!!!!!!!!!:"+error);
                res.send("typeError");
            }else{
                var user_id=req.session.user_id;
                var image_id = image_id;
                var origin_path = file.path;
                var blind = "";
                var filesize = file.size;
                var filetype = path.extname(file.originalname);
        var image_name = filename2;
                var copy_name = "copy-"+image_name;
                var copy_path = "photocopy/"+copy_name;
                var logo_path = "static/watermark.png";

                var conn = mysql.createConnection(global.db_option);
                conn.connect(function (err){
                    if (err) throw err;
                    var sql = `insert into IMAGE (image_id, image_name, origin_path, copy_path, blind, filesize,filetype, creater, timestamp)
                    values ('${image_id}','${image_name}','${origin_path}','${copy_path}','${blind}','${filesize}','${filetype}','${user_id}', now());`;
                    conn.query(sql, function(err, result) {
                        if(err){
                            if (err["code"]=="ER_DUP_ENTRY"){
                                conn.end();
                                res.send("ER_DUP_ENTRY");
                                return false;
                            }
                            else{
                                throw error;
                            }
                        }
```

Upload.js > /create

1. Request에 담긴 file을 Multer 모듈을 이용하여 req.file에 담는다.

2. 이미지를 저장소에 photo디렉토리에 저장하고, 해당 파일을 이용해서 image-hash 모듈을 사용해서 이미지 ID값을 만든다.

3. 위의 작업이 성공 시 해당 DB에 접속하여 이미지를 DB에 기록한다.

이미지 등록

```
var sql2 = `insert into OWN_IMAGE (image_id, user_id, timestamp) values ('${image_id}','${user_id}',now());`;
conn.query(sql2, function (err, result2) {
    if (err) throw err;
    var sql3 = `update USER set point = point+1000 where user_id='${user_id}';`;
    conn.query(sql3, function(err,result3) {//hit + 1
        if (err) throw err;
        conn.end();
        var dimensions = sizeOf(origin_path);
        var logo_x = dimensions.width/12;
        var logo_y = dimensions.height/8;
        var logo_w = (dimensions.width/4);
        var logo_h = dimensions.height/4;
        try {
            var optionsTextWatermark = {
                type: "text",
                text: user_id,
                destination: copy_path,
                source: origin_path,
                position: {
                    logoX : logo_x,
                    logoY : logo_y,
                    logoHeight: logo_h,
                    logoWidth: logo_w
                },
                textOption: {
                    fontSize: logo_w,
                    color: '#010254'
                }
            };
            console.log("fontsizefontsizefontsizefontsizefontsize ::::"+optionsTextWatermark.textOption.fontSize)
            watermark.embed(optionsTextWatermark, function(status) {
                console.log("!!!!!!!!!!+"+status);
            });
        }
        catch (exception) {
            console.log("exception!!!"+exception);
        }
        setTimeout(function() {
            send_item={
                image_id:image_id,
                user_id:user_id
            }
            res.json(send_item);
        });
```

Upload.js > /create

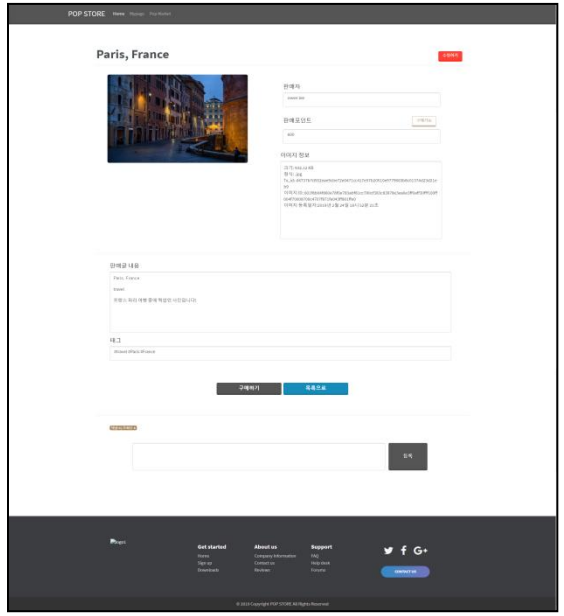
4. 이미지를 DB에 기록한 뒤 사용자와 이미지를 매칭하는 테이블에도 등록해준다.

5. DB기록이 완료되면 해당 이미지에 워터마크를 삽입한다.

6. 이미지ID와 유저ID를 Response 해준다.(이미지 등록 결과 페이지에서 사용)

주요 기능

이미지 등록 ▶ 이미지 거래 ▶ 이미지 이력조회



등록된 이미지로 판매글 작성 <-> 작성된 판매글을 통해 이미지 구매

거래를 통해 변경된 소유자로 블록체인에 이미지의 이력 기록
새로운 이력이 Couch DB에 들어가고, 이전에 있던 이력은 블록에 누적

Chaincode
stub.GetState + stub.PutState

이미지 거래

```
function purchase() {
    cart = {};
    board = $("#data").attr("b_id");
    point = $("#data").attr("b_point");
    seller_id = $("#data").attr("seller_id");
    image_id = $("#data").attr("i_id");
    blind = $("#data").attr("blind");
    cart = {board:board, point:point, seller_id:seller_id,image_id:image_id,blind:blind};
    purcahse_confirm = confirm("현재 포인트에서 "+point+"포인트가 차감됩니다.");
    if(!purcahse_confirm){
        return false;
    }else{
        $.ajax({
        type: "post",
        url: "/market/cart",
        data: cart,
        dataType: "json",
        async: false,
        success: function (data) {
            if (data == 4) {
                alert("이미 판매가 완료된 페이지 입니다.");
                return false;
            }
            else if (data == 0) {
                point = $("#data").attr("b_point");
                priholder = $("#data").attr("seller_id");
                image_id = $("#data").attr("i_id");
                creater = $("#data").attr("creater");

                history_item = {point : point, priholder : priholder,image_id : image_id,creater : creater};
                $.ajax({
                    type: "post",
                    url: "/imagehistory/trade_tuna/",
                    dataType: "json",
                    data : history_item,
                    success: function (data) {
                    location.href = '/market/marketpurchasecomplete';
                    }
                })
            } else if (data == 1) {
                alert("포인트가 부족합니다.");
                return false;
            } else if (data == 2) {
                alert("로그인 후 다시 시도해주세요.");
                location.href = "/login";
            } else if (data == 3) {
                alert("나의 이미지는 구매할 수 없습니다.");
                return false;
            } else {
                alert("다시 시도해주세요.");
                return false;
            }
```

Marketpostconjtent.ejs

1. 구매버튼 클릭 시 purchase()함수가 실행 된다.
2. Ajax를 통해 구매기능이 있는 /market/cart 주소로 글번호, 가격, 파는사람ID, 글 판매완료 여부를 전송한다.
3. 에러가 날 경우 해당 에러를 표시하고, 성공 시 블록체인에 거래를 기록하는 함수를 이용하여 거래를 기록한다.

이미지 거래

```
router.post('/cart', function (req, res){
    var data = -1;
    var board = req.body.board;
    var user_id = req.session.user_id;
    var board_point = req.body.point;
    var image_id = req.body.image_id;
    var seller_id = req.body.seller_id;
    var blind = req.body.blind;
    if(blind == "Y"){
        data = 4;
        res.json(data);
        return false;
    }
    else if(!user_id){
        data = 2;
        res.json(data);
        return false;
    }
    else if(user_id == seller_id){
        data = 3;
        res.json(data);
        return false;
    }else if(board_point > 1000000){
        data = -3;
        res.json(data);
        return false;
    }
    else{
        var conn = mysql.createConnection(global.db_option);
        conn.connect(function(err){
            if (err){
                res.json(data);
                return false;
            }
```

Market.js > /cart

1. 전송받은 정보를 body-parser 모듈을 사용해서 변수에 저장한 뒤 검사하여 해당되는 오류가 있는지 확인한다.
2. 오류가 없을 시 전역변수로 저장해둔 Database 옵션을 사용해서 커넥션을 생성한다.

이미지 거래

```
conn.beginTransaction(function(err){
    if (err){
        conn.end();
        res.json(data);
        return false;
    }
    var sql = `update USER set point = point - ${board_point} where user_id = '${user_id}';`;
    conn.query(sql, function(err, result){
        if (err){
            conn.rollback(function (){
                data=1;
                conn.end();
                res.json(data);
                return false;
            });
        }
        var sql1 = `update USER set point = point + ${board_point} where user_id = '${seller_id}';`;
        conn.query(sql1, function(err, result){
            if (err){
                conn.rollback(function (){
                    return false;
                });
            }
            var sql2 = `update OWN_IMAGE set user_id = '${user_id}', timestamp=now() where image_id = '${image_id}';`;
            conn.query(sql2, function(err, result){
                if (err){
                    conn.rollback(function (){
                        return false;
                    });
                }
                var sql3 = `update BOARD set blind = 'Y' where BOARD_NO = '${board}';`;
                conn.query(sql3, function(err, result){
                    if (err){
                        conn.rollback(function (){
                            return false;
                        });
                    }
                    else{
                        var sql4 = `select * from IMAGE where image_id = '${image_id}';`;
                        conn.query(sql4, function(err, result){
                            if (err){
                                conn.rollback(function (){
                                    return false;
                                });
                            }
                            else{
                                var origin_path = result[0].origin_path;
                                var copy_path = result[0].copy_path;
                                var dimensions = sizeOf(origin_path);
                                var logo_x = dimensions.width/12;
                                var logo_y = dimensions.height/10;
                                var logo_w = (dimensions.width/10);
                                var logo_h = (logo_w/5*2);
                                try {
                                    var optionsTextWatermark = {
                                        type: "text",
                                        text: user_id, // This is optional if you have provided text Watermark
                                        destination: copy_path,
                                        source: origin_path,
                                        position: {
                                            logoX : logo_x,
                                            logoY : logo_y,
                                            logoHeight: logo_h,
                                            logoWidth: logo_w
                                        },
                                        textOption: {
                                            fontSize: logo_w, //In px default : 28
                                            color: '#AAF122' // Text color in hex default: #000000
                                        }
                                    };
                                    watermark.embed(optionsTextWatermark, function(status) {
                                    });
                                }
                                catch (exception) {
                                    console.log("exception!!!"+exception);
                                }
                                conn.commit(function (){
                                    console.log("이미지 구매 성공");
                                    data = 0;
```

Market.js > /cart

3. 커넥션이 생성되면 beginTransaction 함수를 사용하여 거래에 들어가는 쿼리들을 트랜잭션으로 묶는다.

4. 포인트 +-, 소유권 변경, 판매글 판매완료 처리를 실행한다.

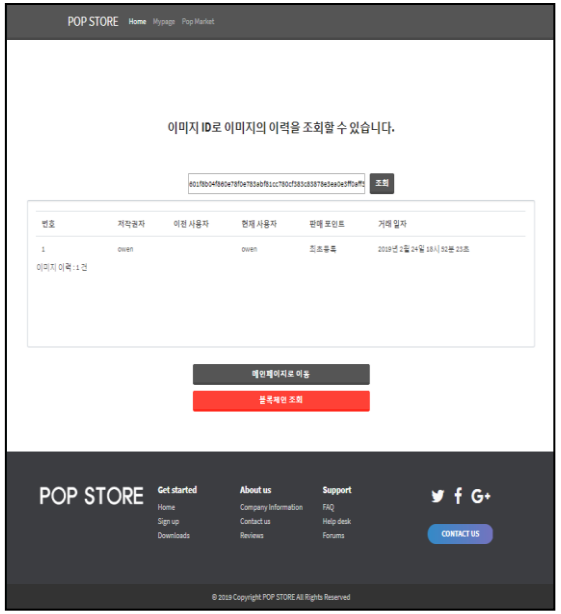
5. 이미지 워터마크를 새로운 사용자의 ID로 바꾼다.

6. 위의 과정에서 오류가 생기면 rollback()함수를 사용해서 모두 취소처리 하고, 오류가 없다면 commit()함수를 사용하여 저장한다.

7. 성공 결과를 Response한다.

주요 기능

이미지 등록 ▶ 이미지 거래 ▶ 이미지 이력조회



이미지의 Hash값(image_id)로 블록체인에 기록된 데이터 조회

Chaincode
stub.GetHistoryForKey

마치며

강태우

본 프로젝트를 진행 할 때 어려웠던 점은 체인코드 작성 시 문법과 사용할 수 있는 함수들에 대한 공부와 체인코드 배포 시 동일한 이름으로 배포하면 적용이 되지 않던 문제가 있었는데 문법과 함수 부분은 shim-GoDoc과 강사님의 조언을 통해 해결이 되었고, 배포 문제는 체인코드 배포 시 이전에 사용했던 이름과 버전을 사용하면 도커의 이미지에 자동으로 등록 된 체인코드가 배포된다는 것을 알게 되어 삭제 후 다시 하니 정상적으로 작동하게 되었습니다.

본 프로젝트를 수행하기 전에는 새로운 기술을 적용하여 프로젝트를 진행하는 것에 대한 두려움이 있었는데 구글링과 전문강사님들의 조언과 멘토링, 팀원들과의 협업을 통해 진행해나가면서 도전하는 것은 마냥 두려워할 것이 아니라 제가 발전하는 것에 한 걸음 내딛는 것이라고 느꼈습니다. 도전하는 능력과 새로운 것에 적응하는 능력이 앞으로 다른 일들을 해나갈 때에도 많은 도움이 될 것입니다.

프로젝트 저장소: https://github.com/TaeWooKang/POP_STORE
시연 영상: https://www.youtube.com/watch?v=UuiEw4heWGM&feature=youtu.be