중간보고서

부산대학교 PUSAN NATIONAL UNIVERSITY

공익활동의 기념비적 NFT 발행 서비스

팀 명 P-chain

지도교수 권동현

제 출 일 2022. 08. 01

팀 번 호 36

분 과 D

팀 원

201424536 정태영

201722316 박예서

201924664 황진하

목 차

1. 요구조건 및 제약 사항 분석에 대한 수정사항

1.1 기존 요구조건 및 수정사항

a) 공익활동 증명의 주체 수정

기존에는 공익활동을 했는지 판단하기 위해 공익활동 증명 데이터를 업로드한 후에 운영자가 이 증명서를 확인해 활동에 맞는 이미지 심볼을 제공하도록 했다. 그러나 운영자에게는 증명서의 사실 여부 판단이 어렵고 그에 따른 업무 처리량이 늘어나 이미지 심볼을 제공하는 시간이 늦어질 수 있다고 판단했다. 따라서 공익활동을 주관하는 기관 전용의 발급 계정을 만들어 기관에서 공익활동 여부를 체크하면 이를 반영하여 업적 및 랭킹 시스템에 추가하도록 수정했다.

b) 발급할 이미지 심볼 기반 정보 수정

기존에는 사용자 계정 정보를 입력하면 이 계정 정보를 기반으로 NFT로 발행하기 위한 이미지에 필요한 고유의 이미지 심볼을 제공하도록 했다. 그러나 계정 정보를 사용하게 되면 이미지에서 계정 정보를 추측할 수 있어 정보 유출 문제가 생길 가능성이 있다고 생각해 이미지 심볼은 메타마스크 지갑 이미지에 들어가는 Jazzicon을 이미지 심볼로 제공하기로 했다.

c) 공익활동 분류

기존에 공익활동 여부를 판단할 때 단순히 자원봉사활동만이 아니라 사회에 긍정적으로 작용하는 활동도 공익활동으로 포함시켰지만, 세부적인 판단 기준까지는 완성되지 않은 상태였다. 그 후에 판단 기준을 만들어 공익활동을 봉사활동 시간, 기부 금액, 1회성 활동 3가지로 나누어 분류하였다.

기존 요구조건	수정사항
공익활동 증명 데이터를 운영자가 판단해 처리	공익활동 기관 전용 계정을 만들어 기관에서 처리
계정 정보를 기반으로 고유 이미지 심볼 생성	메타마스크 지갑 주소를 기반으로 고유 이미지 심볼 생성
공익활동 판단 기준의 모호성	공익활동의 분류를 통한 판단 기준의 구체화

[표 1] 기존 요구조건 및 수정사항

1.2 기존 제약사항 및 수정사항

a) 실제 마켓의 연결 비용으로 인한 개발자 마켓 사용

본 서비스의 개발 방향은 실제 사용되는 폴리곤 기반 마켓의 네트워크에 직접 연결하는 방식이었다. 그러나 API 신청을 하기 위해서는 신청자가 이더리움 코인을 소지하고 있어야 했다. 그로 인해 이더리움의 구매 비용이 발생했고, 그 비용이 우리가 충당할 수 있는 수준 이상임에 따라 실제 마켓이 아닌 개발자 마켓을 사용하기로 결정했다.

개발자 마켓은 실제 사용되는 마켓과 유사하며, NFT의 구매 및 판매를 자유롭게 테스트할 수 있다. 또한 실제 마켓의 네트워크로 변경할 때에도 API 키만 바꿔주면 된다는 이점이 있다.

b) 봉사실적의 데이터화

본 서비스의 초안에서 봉사실적은 원래 그 내용이나 진위여부를 판단할 수 있을 만한 내용을 같이 NFT에 등록을 하면서 그 내용을 통합하는 등의 방식을 사용할 계획이었다.

하지만 진위여부를 판단할 수 있을 만한 내용이라는 것이 모호하다는 문제점이 있다. 랭킹 시스템의 순위 매김이나 업적 시스템에서의 달성 여부 및 진척도 판별 등을 판단하기 위해서, 단순히 봉사 실적 등을 인증해줄 수 있을 만한 내용을 같이 저장하는 것으로는 데이터를 통합하기도 힘들 것이고, 데이터의 구체화가 부족하여 어떤 내용을 어떻게 NFT에 담을 지 결정하는 것 또한 쉽지 않을 것이다. (또 NFT는 내용을 보호한다는 개념은 아니기 때문에 안전하지 않을 수도 있다.)

그러므로, 봉사실적을 NFT로 발행하는 것과 별개로 봉사 내용을 항목화하여 저장하는 데이터베이스를 구체화하여 내용을 저장하려고 한다. 시간 등의 항목은 봉사 인증 기관에서 데이터를 가져오기로 하였다. 기존 방식에서의 문제점들을 해결할 수 있을 뿐만 아니라, 반드시 지갑을 연동할 필요가 없으므로 더 빠른 데이터 처리를 기대할 수 있다.

2. 설계 상세화 및 변경 내역

2.1 설계 상세화

a) 개발 환경

- 개발 언어

HTML, CSS, JavaScript

- 필요 오픈 API

Metamask API, Opensea API, Polygon API

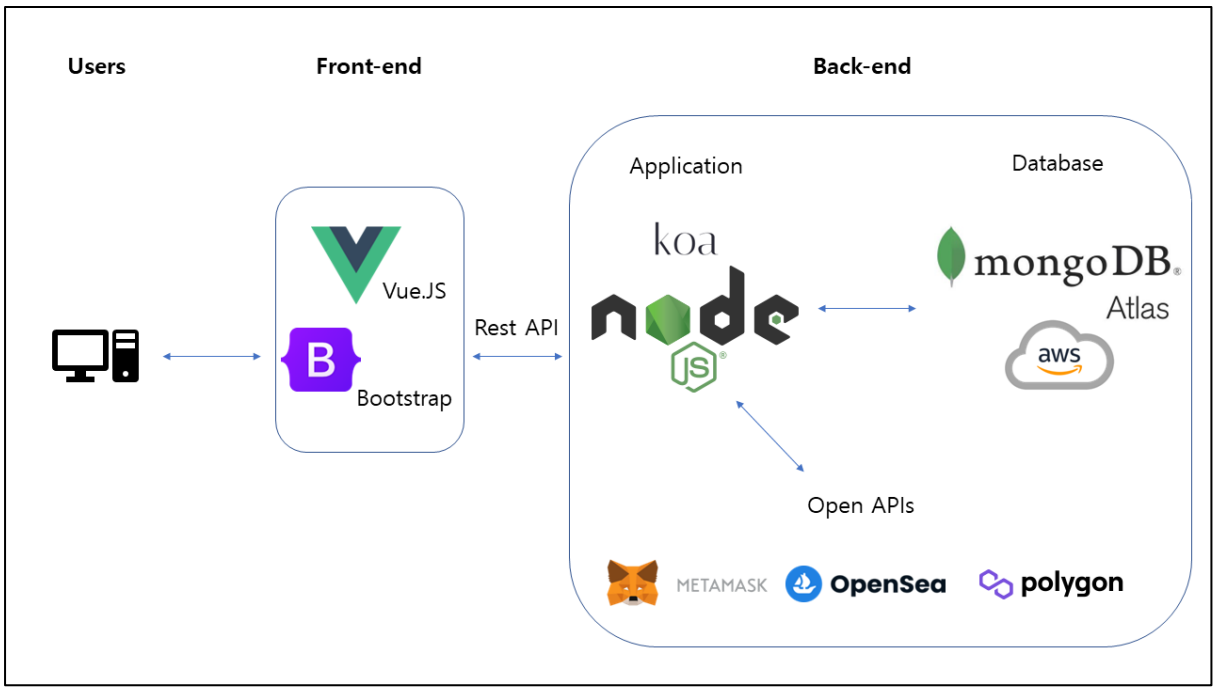
- 데이터베이스

MongoDB(cloud)

- 개발 도구, 개발 소프트웨어, 프레임워크

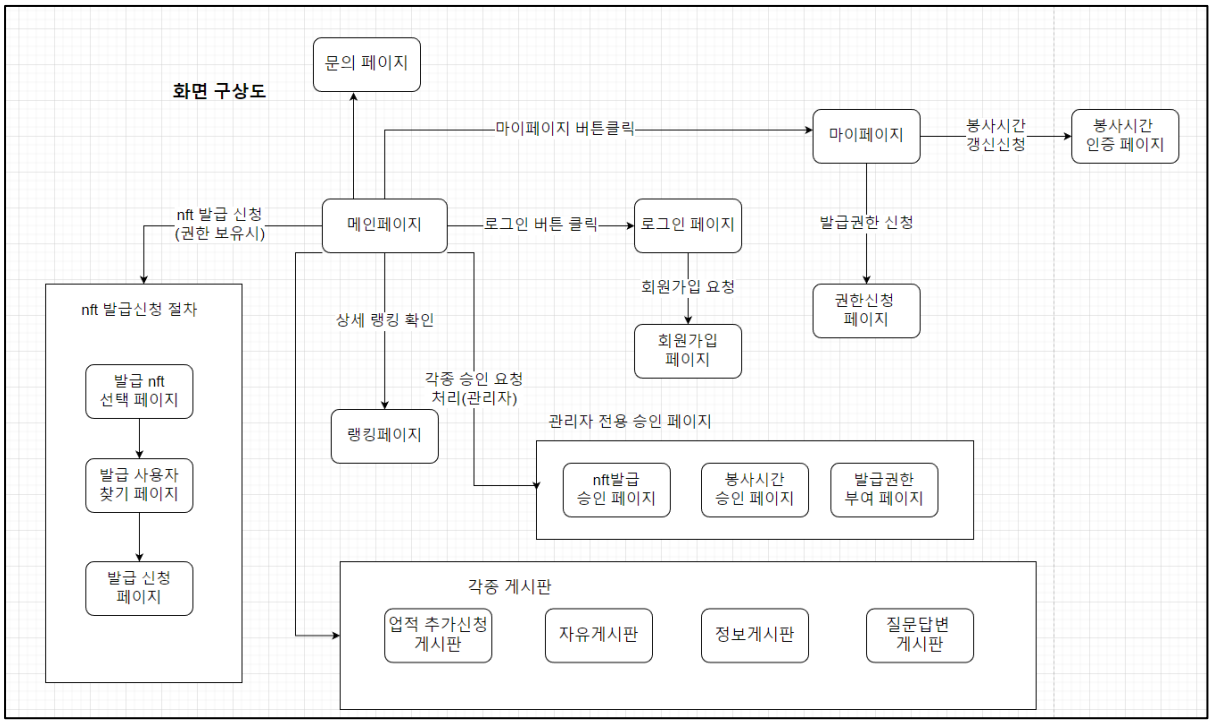
VSCode, Node.js, Koa, Vue.js, Bootstrap

b) 시스템 구성도



Front-end의 프레임워크로 Vue.js와 Bootstrap을 이용한다. Back-end에서는 nodeJS 런타임 환경에서 koa를 이용해 서버를 구축한다. DB로는 몽고DB를 이용한다.

c) 화면 구상도



(1) 메인 페이지

메인 페이지에는 간략한 상위 랭킹 정보 및 본인의 달성 임박 업적(로그인 시)가 표시된다. 또한 로그인 버튼(로그인 중에는 로그아웃), 마이페이지 이동 버튼, 랭킹 페이지 이동 버튼이 모든 사용자에게 표시된다. 한 개라도 발급권한을 보유한 사용자에게는, NFT 발급신청 버튼이 표시된다. 또한 관리자 계정으로 로그인 시 승인 페이지로 이동하는 버튼이 표시된다. 또한 커뮤니티 이동 버튼을 누르면 각종 게시판들이 있는 페이지로 이동이 가능하고, 고객센터 버튼을 누르면 문의 페이지로 이동이 가능하다

(2) 로그인

로그인 페이지는 아이디 비밀번호를 입력해 로그인 할 수 있고, 구글, 페이스북 계정으로도 연동해 로그인이 가능하다. 또한 회원가입 버튼이 표시된다.

(3) 회원가입

회원가입 페이지는 일반 사용자, 발급계정 중 선택하여 가입이 가능하다. 일반사용자의 경우, 개인정보와 로그인 정보를 입력해 가입하게 되고, 발급계정의 경우에는 기관 정보와 로그인 정보를 입력해 가입하게 된다.

(4) 마이페이지

마이페이지에는 크게 진행 정보와 개인정보가 표시된다. 진행 정보에는, 달성 업적, 각종 진행도, 보유 NFT가 표시되고, 개인정보에는 아이디, 비밀번호, 휴대폰 번호, 지갑 주소등이 표시된다. 또한 개인정보 수정이 가능하여 비밀번호 등의 정보 수정이 가능하고, 암호화폐 지갑 연동도 여기서 진행된다. 또한 마이페이지에는 권한 신청 버튼이 있어, 본인이 활동을 주관할 경우에, 발급 권한을 신청할 수 있다. 그리고 봉사시간 갱신 버튼을 누르면, 봉사시간 인증 페이지로 넘어가서 vms 인증번호를 입력하는 방식으로 본인의 봉사시간을 갱신할 수 있다.

(5) 권한 신청 페이지

권한 신청 페이지를 들어가면, 현재 발급 중인 NFT 목록이 있고, 그 중에서 본인이 주관하는 활동을 선택해 발급 권한을 신청할 수 있다. 신청시에는 본인이 주관한다는 증빙서류가 필요하고, 이를 관리자가 판단해 발급 권한을 부여해준다.

(6) 봉사시간 갱신 페이지

봉사시간 갱신 페이지로 들어가면, vms 인증번호를 입력할 수 있고, vms 사이트에서 인증서를 발급, 발급번호를 입력하면 시간 확인 후, 본인의 봉사시간이 갱신되게 된다.

(7) 랭킹 페이지

랭킹 페이지에는 NFT 보유랭킹, 업적 달성 갯수 랭킹, 각종 진행도 랭킹이 있다. 상위 랭킹 사용자들의 닉네임과 달성 수치가 표시되고, 그 밑에 본인의 등수와 달성 수치가 표시된다.

(8) 발급신청 절차 페이지

발급 신청은 발급종류 선택, 대상 찾기, 발급 요청 3단계로 나누어서 신청이 이루어진다. 각각 페이지에서 발급할 NFT를 선택하고(본인이 권한 보유중인 NFT 중에서), 발급해줄 사용자를 검색해 찾고, 증빙 서류와 함께 발급을 신청하게 된다.

(9) 승인 페이지

승인 페이지는 관리자 계정으로만 접근이 가능하고, NFT 발급 승인, 봉사시간 갱신 승인, 발급 권한 부여를 진행할 수 있다.

(10) 게시판 페이지

게시판에는 자유게시판, 정보게시판, 질문답변 게시판, 업적 추가신청 게시판이 있다. 자유게시판은 유저 친목 등에 이용하고, 정보 게시판은 여러 종류의 공익적 활동에 대한 팁을 정리하는 게시판이다. 질문 답변 게시판은 유저 간의 질문과 답변이 이루어지는 게시판이다. 또한 업적 추가신청 게시판에서, 활동의 종류와, 증빙서류 종류 등을 포함해 업적 추가 신청을 할 수 있고, 관리자가 공익성과, 가치, 인증의 신뢰도 등을 판단해 업적에 추가할 수 있다.

(11) 문의 페이지

문의 페이지에는 관리자가 등록하는 공지사항과 자주하는 질문 목록이 있고, 또한 개인적으로 1:1 문의를 할 수 있다. 해당 문의는 관리자가 확인 후 답변하는 방법을 사용한다.

2.2 설계상 변경 내역

a) 활동 인증방법에 발급권한 도입

원래 개발방향은 활동을 인증할 수 있는 증빙서류를 업로드해, 확인 후 NFT를 발급하는 형태였으나, 증빙서류 종류의 한계와 조작가능성, 관리자 확인의 한계 등의 문제로 인해, 대안이 필요했다.

이에 대한 대안으로, 발급 권한이라는 개념을 도입해, 해당 활동을 주관하는 기관 혹은 개인이 인증 후, 권한을 얻어서, 발급받을 사용자와 발급할 NFT, 증빙서류를 통합해 신청하면, 관리자는 마지막 승인만 하는 형태로 구상했다.

a-1) 발급계정의 도입

발급 권한의 도입에 따라서, 해당 인증기관의 경우, 담당자 변경 등의 일이 발생할 수 있다는 것을 인지, 단순한 사용자 계정보다는 해당 기관의 별도 계정이 있으면, 이용이 편리하겠다는 생각을 하였다.

따라서, 이름, 휴대폰번호 등을 입력할 사용자 계정과 별도로, 기관명, 기관번호 등을 입력해서 가입할 기관/발급 전용 계정을 도입하게 되었다.

b) 봉사시간 인증절차의 상세화

가장 유명한 공익적 활동인 봉사시간에 대해서, 어떻게 인증 받을지, 결정하지 못하고, 어려움을 겪었었다. 이를 해결하기 위해 찾아보던 중, vms 사회복지자원봉사 인증관리 사이트 안에, 인증서 출력이 있는 것을 확인하였다.

인증서 발급 시, 인증번호가 발급되고, 해당 인증번호를 입력 시, 인증서 출력에 나와있는, 봉사횟수, 봉사 시간을 확인할 수 있다는 것을 발견해, 이를 활용해서, 봉사시간을 인증하기로 결정하였다.

c) 랭킹 및 동기부여를 위한 진행도 도입

기존 랭킹 시스템의 목적은, 활동 및 NFT 구매의욕을 증진시키고자 도입한 경쟁 시스템이었다.

하지만 단순한, NFT 보유랭킹, 개인 업적 달성 개수 랭킹으로는 충분한 동기부여가 되지 않는다고 판단해 봉사시간, 활동 횟수 등의 일정 수치달성을 목표로 하는 업적들에 대해서 진행도를 도입하기로 했다.

진행도는 개인별로 DB에 저장하여서, 별도의 랭킹 등록도 진행되고, 달성 임박 업적에 대해서는 메인 페이지에도 표시하기로 결정하였다.

d) 사용 프레임워크 변경

기존에는 웹 애플리케이션을 개발하기 위해 MEAN 스택을 사용하기로 했다. 그러나 Front-end는 변화가 굉장히 빠른 만큼, Angular 프레임워크는 요즘 잘 사용하지 않아 Vue.js를 Front-end 프레임워크로 선택했다. 또한 졸업과제에 디자인적인 부분은 Bootstrap 프레임워크를 사용하기로 했다.

그리고 웹 프레임워크의 경우 Express를 사용하기로 했으나 오픈소스 소유권 문제도 있고, Express보다 가벼운 Koa 프레임워크가 등장하면서 이를 사용하기로 했다.

데이터베이스의 경우 로컬 DB로 설계하였으나 여러 컴퓨터로 개발을 하는 점, 그리고 몽고DB 사이트에서 클라우드 DB를 제공했기 때문에 로컬DB에서 클라우드 DB로 변경하였다.

3. 갱신된 과제 추진 계획

5월		6월					7월				8월				9월				
3	4	1	2	3	4	5	1	2	3	4	1	2	3	4	1	2	3	4	5
오픈 API 신청																			
		사이트 화면 구상																	
				라우터 구성															
						DB 스키마 정의													
							Front-end 구성												
							Back-end 구성												
									중간 보고서 작성										
									클라우드 서버 서비스 적용										
											인증 기관에서 데이터 추출 및 DB화								
												Open API 적용							
																테스트 및 디버깅			
																	최종 보고서 작성		
																		발표 준비	

4. 구성원별 진척도

이름	진척도
황진하	● 로그인 기능 back-end 부분 구현 ● 스키마 구성 내용 구현
정태영	● DB 스키마 정의 ● 클라우드 DB 활성화
박예서	● 웹사이트 화면 구상 ● 라우터 구성
공 통	● 필요 문서 작성 ● 보고서 작성

5. 보고 시점까지의 과제 수행 내용 및 중간 결과

5.1 회원가입

```
// 로컬 회원가입
exports.localRegister = async (ctx) => {
    // 데이터 검증
    const schema = Joi.object().keys({
        username: Joi.string().alphanum().min(4).max(15).required(),
        email: Joi.string().email().required(),
        password: Joi.string().required().min(6)
    });

    const result = schema.validate(ctx.request.body);

    if(result.error) {
        ctx.status = 400; // Bad request
        return;
    }

    // 아이디 / 이메일 중복 체크
    let existing = null;
    try {
        existing = await Account.findByEmailOrUsername(ctx.request.body);
    } catch (e) {
        ctx.throw(500, e);
    }

    if(existing) {
    // 중복되는 아이디/이메일이 있을 경우
        ctx.status = 409; // Conflict
        // 어떤 값이 중복되었는지 알려줍니다
        ctx.body = {
            key: existing.email === ctx.request.body.email ? 'email' : 'username'
        };
        return;
    }

    // 계정 생성
    let account = null;
    try {
        account = await Account.localRegister(ctx.request.body);
    } catch (e) {
        ctx.throw(500, e);
    }

    ctx.body = account.profile; // 프로필 정보로 응답합니다.
};
```

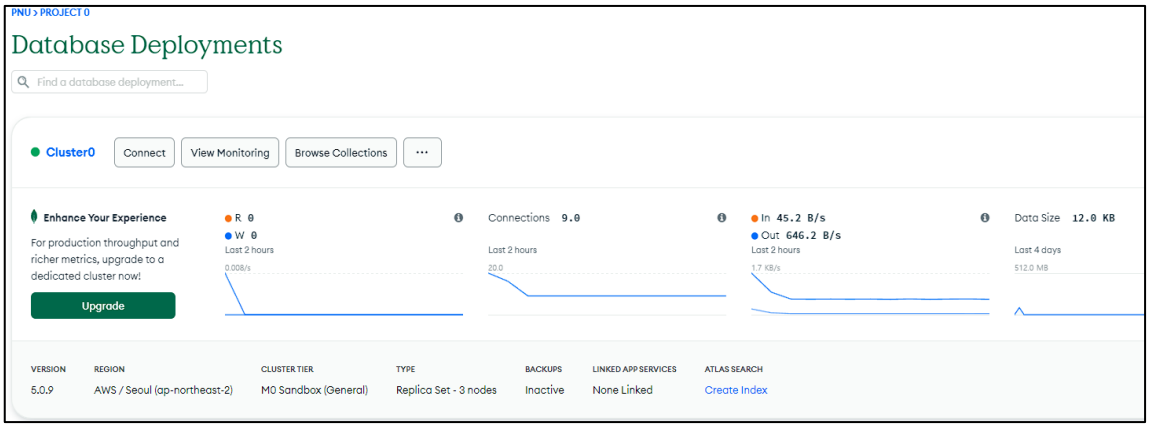
5.2 로그인/로그아웃

```
// 로컬 로그인
exports.localLogin = async (ctx) => {
    // 데이터 검증
    const schema = Joi.object().keys({
        email: Joi.string().email().required(),
        password: Joi.string().required()
    });

    const result = schema.validate(ctx.request.body);

    if(result.error) {
        ctx.status = 400; // Bad Request
        return;
    }

    const { email, password } = ctx.request.body;

    let account = null;
    try {
        // 이메일로 계정 찾기
        account = await Account.findByEmail(email);
    } catch (e) {
        ctx.throw(500, e);
    }

    if(!account || !account.validatePassword(password)) {
    // 유저가 존재하지 않거나 || 비밀번호가 일치하지 않으면
        ctx.status = 403; // Forbidden
        return;
    }

    ctx.body = account.profile;
};

// 이메일 / 아이디 존재유무 확인
exports.exists = async (ctx) => {
    const { key, value } = ctx.params;
    let account = null;

    try {
        // key 에 따라 findByEmail 혹은 findByUsername 을 실행합니다.
        account = await (key === 'email' ? Account.findByEmail(value) : Account.findByUsername(value));
    } catch (e) {
        ctx.throw(500, e);
    }

    ctx.body = {
        exists: account !== null
    };
};

// 로그아웃
exports.logout = async (ctx) => {
    ctx.body = 'logout';
};
```

5.3 로그인 기능을 위한 모델

```
function hash(password) {
    return crypto.createHmac('sha256', process.env.SECRET_KEY).update(password).digest('hex');
}

const Account = new Schema({
    profile: {
        username: String,
        thumbnail: { type: String, default: '/static/images/default_thumbnail.png'}
    },
    email: { type: String },
    // 소셜 계정으로 회원가입을 할 경우에는 각 서비스에서 제공되는 id와 accessToken을 저장
    social: {
        facebook: {
            id: String,
            accessToken: String
        },
        google: {
            id: String,
            accessToken: String
        }
    },
    isIssuer: Boolean,
    password: String, // 로컬 계정의 경우엔 비밀번호를 해싱해서 저장
    walletAddress: String, // 수정가능.
    achievementProgress: [String], // 수정가능
    issuanceCount: {type: Number, default: 0}, // 서비스에서 nft 발행을 받을때마다 1올라간다.
    createdAt: {type: Date, default: Date.now}
});

Account.statics.findByUsername = function(username) {
    // 객체에 내장되어있는 값을 사용할 때는 객체명.키
    return this.findOne({'profile.username': username}).exec();
};

Account.statics.findByEmail = function(email) {
    return this.findOne({email}).exec();
};

Account.statics.findByEmailOrUsername = function({username, email}) {
    return this.findOne({
        // $or 연산자를 통해 둘중에 하나를 만족하는 데이터를 찾습니다
        $or: [
            { 'profile.username': username },
            { email }
        ]
    }).exec();
};

Account.statics.localRegister = function({ username, email, password}) {
    //데이터를 생성할 때는 new this()를 사용
    const account = new this({
        profile: {
            username
            // thumbnail 값을 설정하지 않으면 기본값으로 설정
        },
        email,
        password: hash(password)
    });

    return account.save();
}

Account.methods.vaildatePassword = function(password) {
    // 함수로 전달받은 password의 해시값과, 데이터에 담겨있는 해시값과 비교를 합니다.
    const hashed = hash(password);
    return this.password === hashed;
};
```

5.4 DB 클라우드 활성화



5.5 라우터 구성

```
/* link part */
const auth_account = require('./auth_account');
const auth_apply = require('./auth_apply');
const log_in = require('./log_in');
const main = require('./main');
const manage_page = require('./manage_page');
const mypage = require('./mypage');
const nft_choice = require('./nft_choice');
const nft_issue = require('./nft_issue');
const ranking = require('./ranking');
const user_search = require('./user_search');
const vms_ins = require('./vms_ins');

app.use(bodyParser()); // have to be upward of router

router.get('/', (ctx, next) => {
    ctx.body = '첫 화면 (홈)';
})

router.use('/auth_account', auth_account.routes());
router.use('/auth_apply', auth_apply.routes());
router.use('/log_in', log_in.routes());
router.use('/main', main.routes());
router.use('/manage_page', manage_page.routes());
router.use('/mypage', mypage.routes());
router.use('/nft_choice', nft_choice.routes());
router.use('/nft_issue', nft_issue.routes());
router.use('/ranking', ranking.routes());
router.use('/user_search', user_search.routes());
router.use('/vms_ins', vms_ins.routes());

app.use(router.routes()).use(router.allowedMethods());
```