Photo SYN

Photo SYN

01. 기획

PHOTO + SYN

(사진) + (함께, 동화)

PHOTOSYNTHESIS

광합성 (빛을 통한 성장)

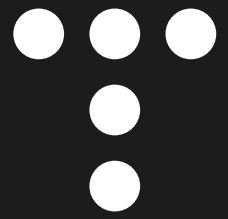
기획

소통

돈

정보

재미



기획

여행자 A

좋은 여행 콘텐츠가 있는데 수익화 할 순 없을까?

기획

여행자 B

여행 코스를 찾고 있는데 잘 정리된 곳은 없을까?

기획

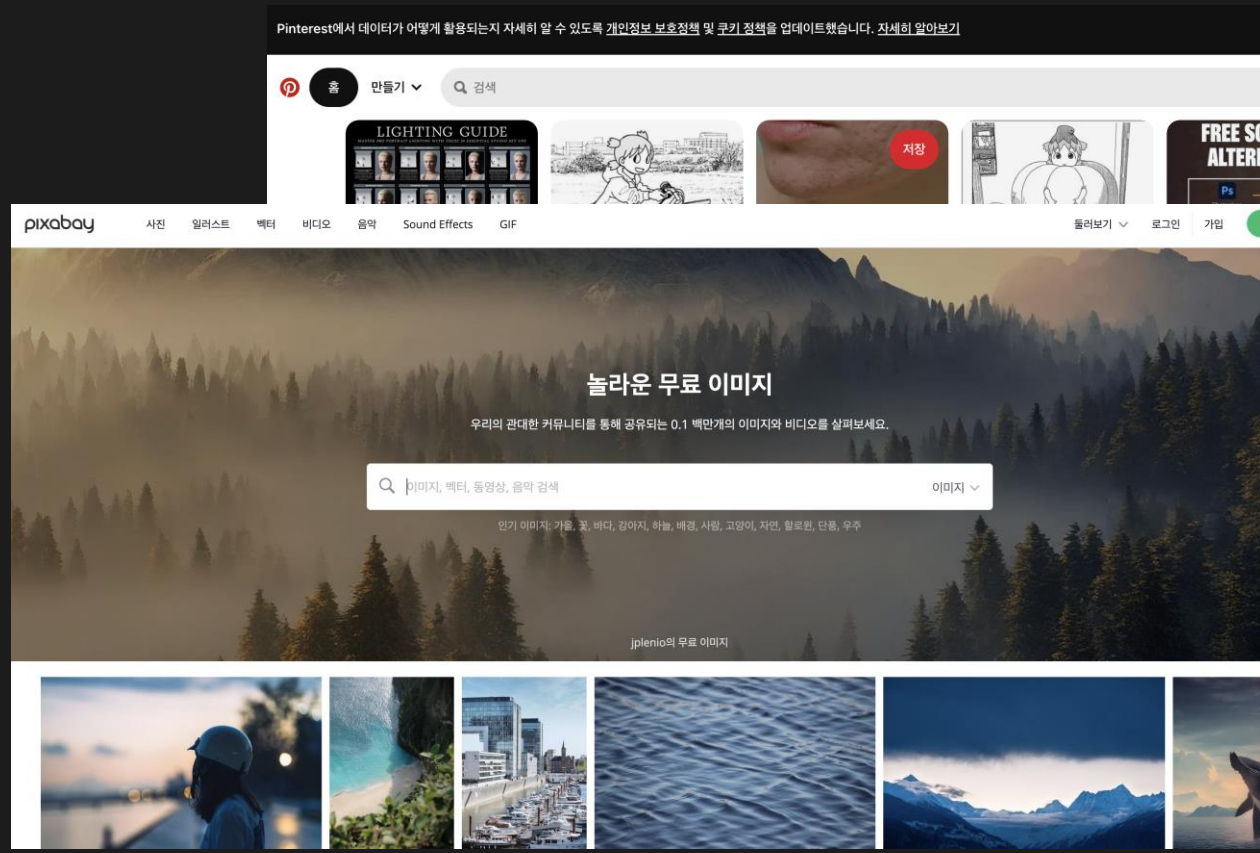
구매자 A

여행관련 사진을 싸고 간편하게 살 수 있는곳은 없을까?

기획

사진 판매사이트

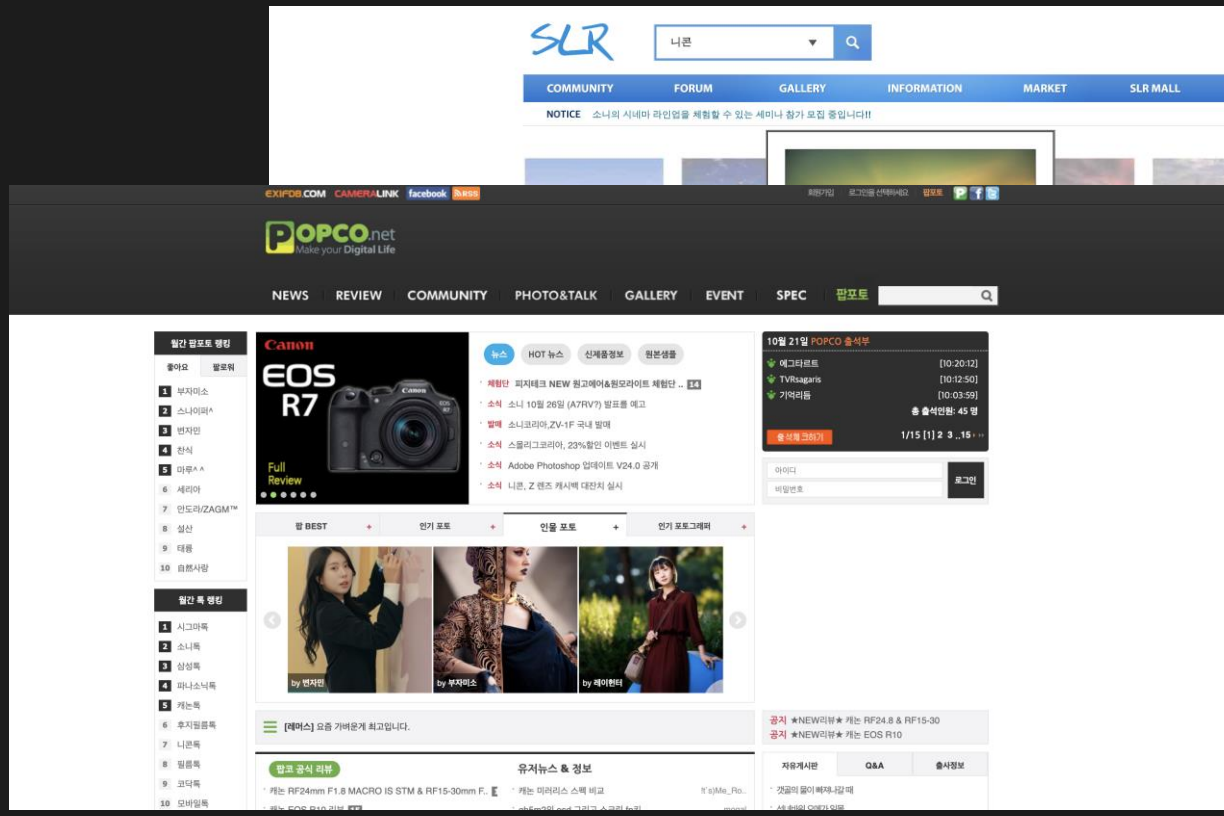
- 소통 어려움
- 제한된 목적



기획

사진 커뮤니티

- 수익화 불가
- 제한된 테마



02. 구현 목표 및 개발환경

구현 목표

구현 목표

갤러리

- 사진 구매 · 판매
- 메타데이터 등록 · 조회

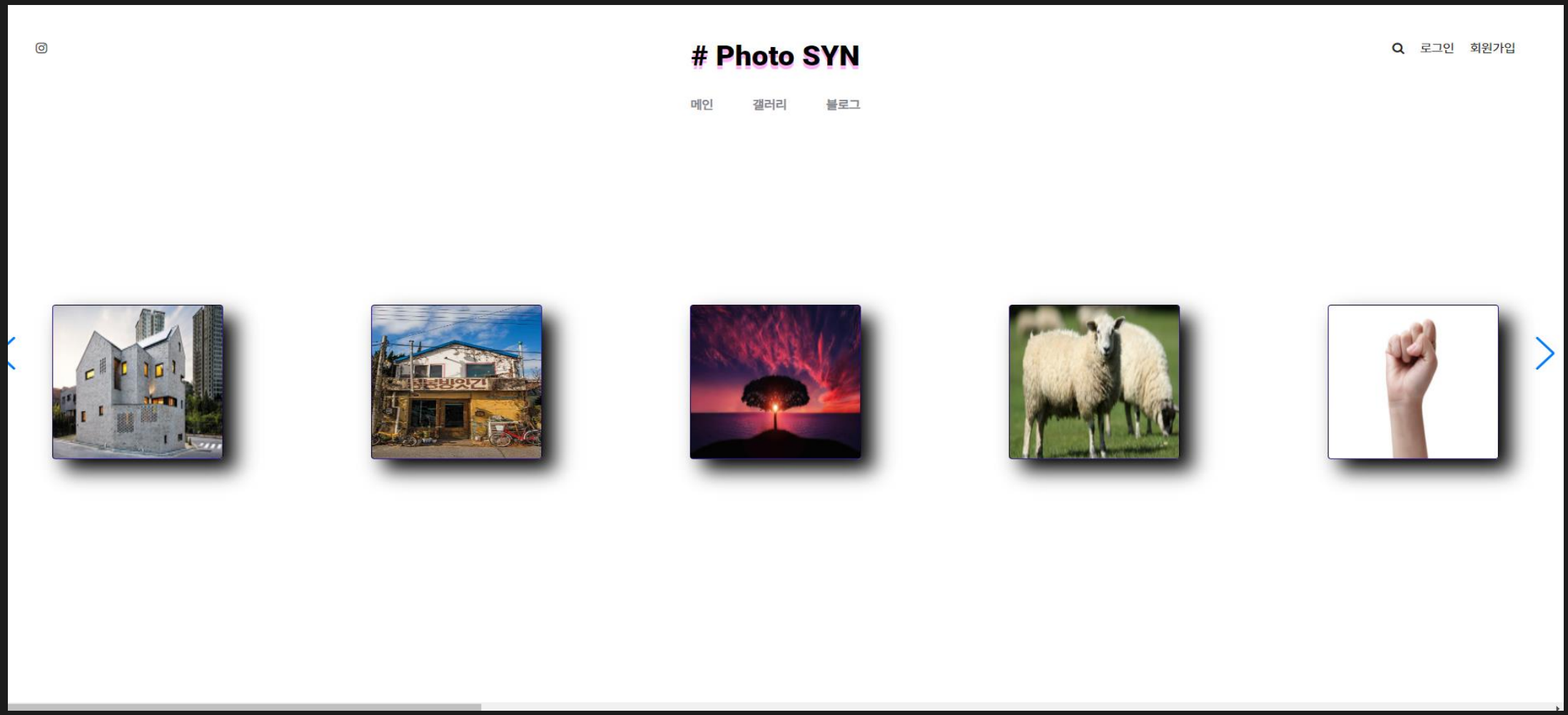
블로그

- 블로그 작성
- 태그 · 댓글

유지보수

- 회원 정산 관리
- 게시글 관리

구현 목표



구현 목표

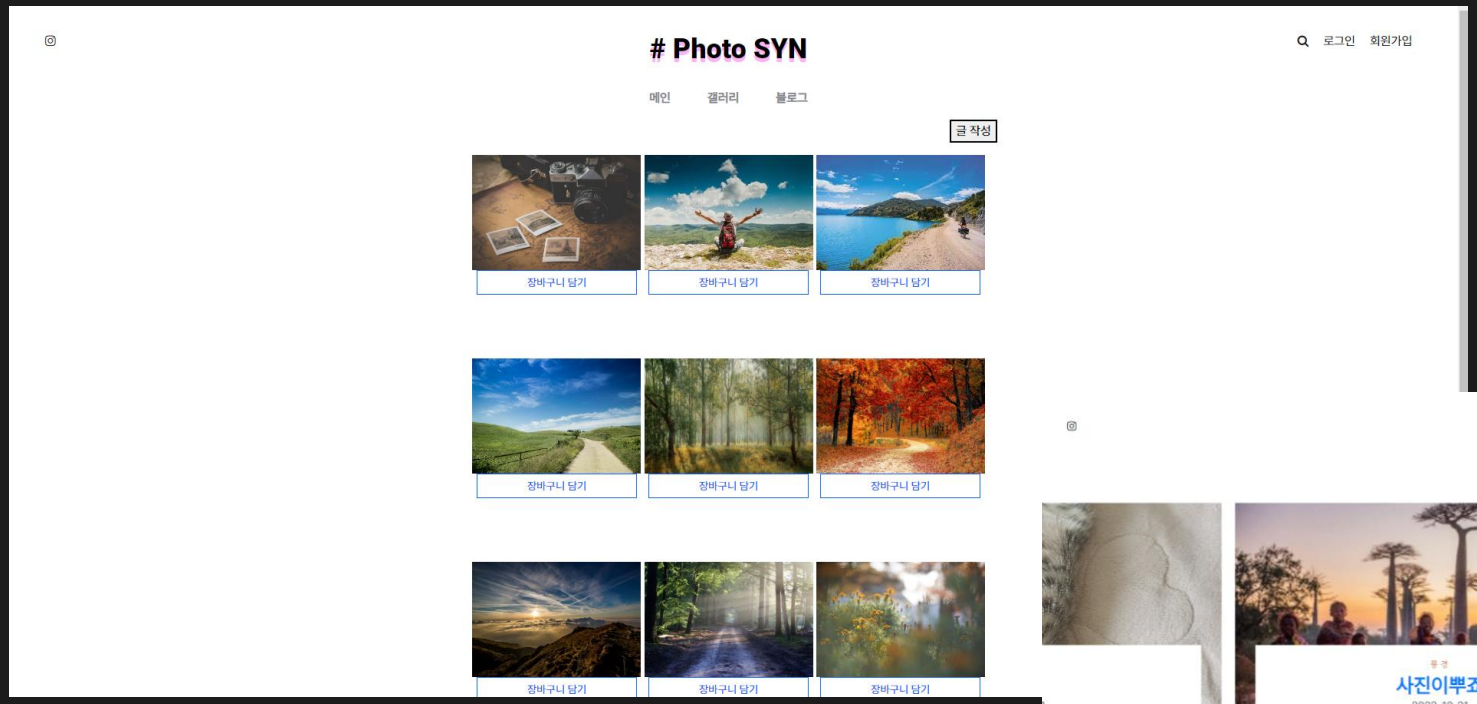


Photo SYN
로그인 회원가입
메인 갤러리 블로그
글 작성
장바구니 담기

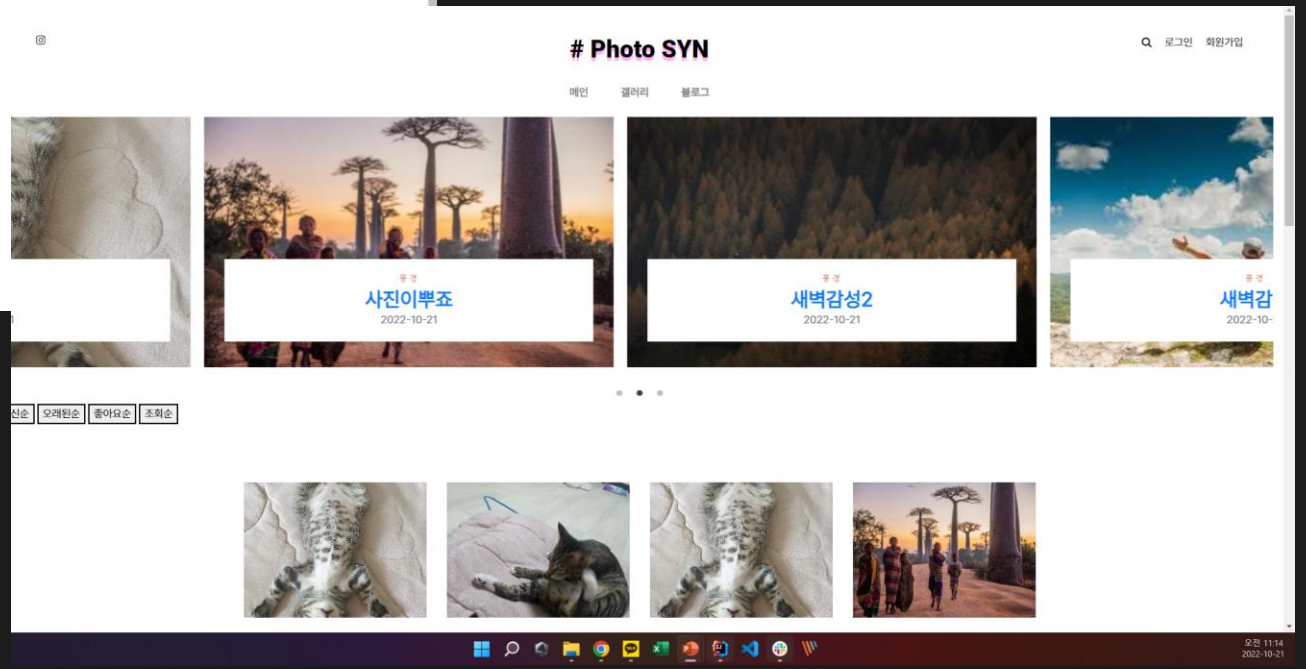


Photo SYN
로그인 회원가입
메인 갤러리 블로그
사진이뿌죠
2022-10-21
새벽감성2
2022-10-21
새벽감
오래된순
좋아요순
조회순

구현 목표

수익화

나의 노력이 수익이 될 순 없을까?

소통

관심사에 맞는 양질의 정보를 얻을 순 없을까?

간편함

번거로운 절차없이 사진을 바로 사용할 순 없을까?
간편하게 구매할 순 없을까?

구현 목표

개발 환경

개발 환경

O/S


Window 10


Mac OS

개발 언어


JAVA


JavaScript


HTML5


CSS

개발툴


IntelliJ IDEA Ultimate


Visual Studio Code

API

- 카카오페이
- MetaData 2.18
- CKEditor 4
- Lombok 1.18
- JavaMailSender
- Ojdbc8
- hikariCP 3.4.5
- commons-io 2.6
- commons-fileupload 1.4

DB


Oracle Database 19c

프레임워크


Spring FrameWork 5.2.9


Spring Security


BootStrap

MyBatis 3

Server


TOMCAT 9.0

03. 역할분담 및 일정

역할 분담

장예성

- 회원가입 · 로그인
- 마이페이지

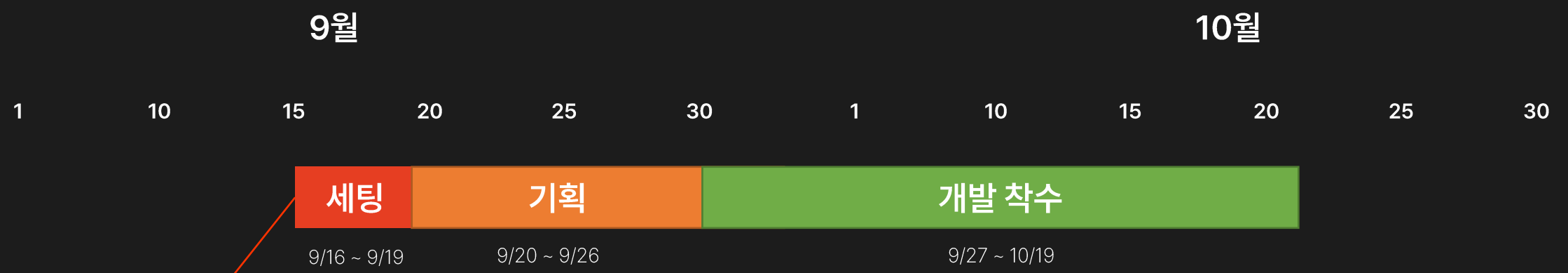
김찬욱

- 갤러리
- 메타데이터

김계정

- 블로그 + 댓글
- 관리자 페이지

일정



- 협업툴 연동
  (Jira, Git Hub, Slack)

- 구현 목표 및 주제선정
- 유스케이스 다이어그램
- 기능 및 요구사항 정의
- 개발 기능 우선순위 선정
- 업무 분담
- DB 설계 및 연동

04. 동작 구현

05. 프로젝트 산출물

Photo SYN

요구사항 정의서

Photo SYN

요구사항 정의서

요구사항정의서		
로그인 & 회원가입	로그인	유효성 검사 및 스프링 시큐리티 로그인
	회원가입	필수 기입 사항 유효성 검사 및 일반, 관리자 구분지어 가입 가능
마이페이지	프로필 정보	가입시 입력한 회원정보 수정
		프로필 수정시 비밀번호 변경 및 회원탈퇴 가능
		갤러리 구매시 필요한 포인트 충전
		총 판매금액 확인
	글 관리	사용자가 업로드 한 갤러리 상품 숨김처리 및 해제 기능
	장바구니	선택삭제 및 구매 기능 (구매시 구매내역으로 화면 이동)
갤러리	메인	모든 갤러리를 화면에 보여주는 페이지
	싱글	하나의 게시물의 사진과 메타데이터 값들을 보여줌
	업로드	사진을 올리고 메타데이터를 변경할 수 있는 페이지
블로그	메인	블로그 글들 보여주기
	싱글	글 한 개의 정보 및 글 수정, 삭제
	작성	글 작성
	유저메인	해당 유저의 블로그 글들만 보여주기
관리자 페이지	메인	간단하게 정보들 보여주기
	유저	모든 유저 정보들 보여주기
	유저싱글	유저 정보 수정
	갤러리	갤러리 글 삭제 및 취소
	블로그	블로그 글 삭제 및 취소
	메인태그	메인태그 생성 및 수정, 사진 업로드

요구사항 정의서

Photo SYN

Spring FrameWork 디렉토리 구조				
	member	gallery	blog	admin
controller	○	○	○	○
domain	○	○	○	○
service	○	○	○	○
mapper	○	○	○	○
security	○			
aop			○	○

JSP							
member		gallery		blog		admin	
파일명	설명	파일명	설명	파일명	설명	파일명	설명
login.jsp	로그인	galleryMain3.jsp	모든 갤러리	blogmain.jsp	모든 블로그 글	adminalluser.jsp	모든 유저 확인
main.jsp	메인 화면	galleryMetadata.jsp	갤러리 메타데이터	blogsingle.jsp	블로그 글 1개	adminblog.jsp	블로그글 관리
searchPw.jsp	비밀번호 찾기	gallerySingle2.jsp	갤러리 상세 정보	blogusermain.jsp	한 유저의 블로그글	admingallery.jsp	갤러리글 관리
signup.jsp	회원가입	uploadForm.jsp	갤러리 등록	checkmainimg.jsp	작성후 메인사진 설정	adminmain.jsp	관리자 메인
profile.jsp	내 정보			update.jsp	글 수정	adminmaintag.jsp	메인태그 관리
profileBuy.jsp	구매내역			updatemainimage.jsp	수정후 메인사진 설정	adminusers.jsp	유저 정보 관리
profileSell.jsp	판매내역			write.jsp	글 작성		
profileDelete.jsp	회원탈퇴						
profileGallery.jsp	갤러리 관리						
profileModify.jsp	프로필 수정						
profilePointAdd.jsp	포인트 충전						
profileCart.jsp	장바구니						
updatePw.jsp	비밀번호 변경						

DB 테이블 정의서

DB 테이블 정의서

회원

테이블명	NO.	컬럼명	타입	길이	설명	KEY	DEFAULT	비고
USER_LIST	1	U_ID	VARCHAR2	50	이메일	PK		
	2	U_PW	VARCHAR2	100	비밀번호			
	3	U_PHONE	VARCHAR2	30	전화번호			
	4	U_NICK	VARCHAR2	100	닉네임	UNIQUE		중복 불가
	5	U_REG	DATE		가입일			
	6	U_STATUS	NUMBER		상태			
	7	U_PIC	VARCHAR2	50	프로필사진			유저 프로필 사진도 사진 테이블과 별도로 저장
	8	U_POINT	NUMBER		포인트		0	
	9	MS_MEMBERSHIP	NUMBER		유저 맴버쉽 등급		0	
	10	U_SNS	VARCHAR2	50	로그인SNS종류			naver, kakao etc..
	11	U_SNSTOKEN	VARCHAR2	100	로그인토큰값저장			

테이블명	NO.	컬럼명	타입	길이	설명	KEY	DEFAULT	비고
MEMBERSHIP	1	MS_MEMBERSHIP	NUMBER	50	맴버 등급	PK		
	2	MS_ATT	NUMBER	100	출석수	NOT NULL		
	3	MS_SELL	NUMBER	30	판매수	NOT NULL		
	4	MS_REGIST	NUMBER	100	갤러리 사진 등록 수	NOT NULL		

DB 테이블 정의서

갤러리 및 메타데이터

테이블명	NO.	컬럼명	타입	길이	설명	KEY	DEFAULT	비고
GALLERY	1	G_NO	NUMBER		갤러리 고유번호	PK		
	2	MT_NAME	VARCHAR2	50	메인태그	NOT NULL		
	3	U_ID	VARCHAR2	50	작성자	FK, NOT NULL		
	4	G_HNAME	VARCHAR2	50	파일명(고화질)			사진 파일명
	5	G_LNAME	VARCHAR2	50	파일명(저화질)			
	6	G_HPRICE	NUMBER		고화질가격			
	7	G_LPRICE	NUMBER		저화질가격			
	8	G_CONTENT	VARCHAR2	2000	내용(설명)			
	9	G_REG	DATE		작성일		SYSDATE	
	10	G_EDIT	DATE		수정일		SYSDATE	
	11	G_STATUS	NUMBER		상태			0: 판매중, 1: 판매완료, 2: 판매중지
	12	G_SALES	NUMBER		판매량		0	
	13	G_LIKE	NUMBER		좋아요 수		0	
	14	G_READCOUNT	NUMBER		조회수		0	
	15	G_TAG1	VARCHAR2	50	추가태그			#강아지#멍멍이#헬로 -> 강아지,멍멍이,헬로
	16	G_TYPE	VARCHAR2	100	사진 확장자명(.jpg .png 등)			jpg / png / bmp / gif

테이블명	NO.	컬럼명	타입	길이	설명	KEY	DEFAULT	비고
METADATA	1	G_NO	NUMBER		갤러리 고유번호	PK, NOT NULL		
	2	M_LONGITUDE	VARCHAR2	50	경도			
	3	M_LATITUDE	VARCHAR2	50	위도			
	4	M_IMGLENS	VARCHAR2	50	렌즈			
	5	M_IMGCAMERA	VARCHAR2	50	카메라			
	6	M_IMGHQLY	VARCHAR2	50	고화질픽셀사이즈			
	7	M_IMGLQLY	VARCHAR2	50	저화질픽셀사이즈			
	8	M_APERTURE	NUMBER		조리개			소수점 0.1
	9	M_ISO	NUMBER		감도			소수점 0.1
	10	M_FLENGTH	NUMBER		초점거리			소수점 0.1
	11	M_SHUTTERSPEED	NUMBER		셔터스피드			소수점 0.001

DB 테이블 정의서

블로그

테이블명	NO.	컬럼명	타입	길이	설명	KEY	DEFAULT	비고
Blog	1	B_NO	NUMBER		블로그 고유번호	PK, NOT NULL		
	2	U_ID	VARCHAR2	50	작성자			
	3	MT_NAME	VARCHAR2	50	메인태그명			
	4	B_SUBJECT	VARCHAR2	100	제목			
	5	B_CONTENT	VARCHAR2	4000	내용			
	6	B_LIKE	NUMBER		좋아요 수			
	7	B_READCOUNT	NUMBER		조회수			
	8	B_REG	DATE		작성일			
	9	B_EDIT	DATE		수정일			
	10	B_STATUS	NUMBER		상태			
	11	B_TAG1	VARCHAR2	1000	서브 태그들			
	12	B_REPORTCOUNT	NUMBER		신고 수			

테이블명	NO.	컬럼명	타입	길이	설명	KEY	DEFAULT	비고
Blog_IMG	1	BI_NO	VARCHAR2		블로그 사진 고유번호	PK, NOT NULL		
	2	B_NO	NUMBER		블로그 고유번호			
	3	BI_NAME	VARCHAR2	900	사진명			
	4	BI_MAIN	NUMBER		메인사진 여부			
	5	BI_UUID	VARCHAR2	100	사진고유번호			
	6	BI_ORIGINNAME	VARCHAR2	1000	사진기존이름			

DB 테이블 정의서

관리자

테이블명	NO.	컬럼명	타입	길이	설명	KEY	DEFAULT	비고
MainTag	1	MT_NAME	VARCHAR2	50	메인태그명			
	2	MT_IMG	VARCHAR2	50	이미지			
테이블명	NO.	컬럼명	타입	길이	설명	KEY	DEFAULT	비고
TAG_LIST	1	T_ID	NUMBER		태그 ID	PK, NOT NULL		
	2	T_NAME	VARCHAR2	100	태그명			
	3	T_COUNT	NUMBER		태그 사용 횟수			
	4	T_DAYCOUNT	NUMBER		일일 태그 사용 횟수			
테이블명	NO.	컬럼명	타입	길이	설명	KEY	DEFAULT	비고
MemberShip	1	MS_MEMBERSHIP	NUMBER		맴버 등급	PK, NOT NULL		
	2	MS_ATT	NUMBER		출석수			
	3	MS_SELL	NUMBER		판매수			
	4	MS_REGIST	NUMBER		갤러리 사진 등록 수			

06. 주요코드

MEMBER

임시비밀번호 메일 전송

```
@RequestMapping(value = "searchPwPro", method = RequestMethod.POST)
    public ModelAndView sendEmailAction(@RequestParam Map<String, Object> paramMap, ModelMap model,
    ModelAndView mv, HttpServletResponse response, MemberDTO memberDTO) throws Exception {

        String id = (String) paramMap.get("id");
        String name = (String) paramMap.get("name");
        String password = "";

        memberDTO.setId(id);

        if (memberDTO.getId().equals(id)) {
            for (int i = 0; i < 12; i++) {
                password += (char) ((Math.random() * 26) + 97);
            }
            memberDTO.setPw(encoder.encode(password));

        }

        try {
            MimeMessage msg = mailSender.createMimeMessage();
            MimeMessageHelper messageHelper = new MimeMessageHelper(msg, true, "UTF-8");

            messageHelper.setSubject(name + "님 임시 비밀번호 발급 안내드립니다.");
            messageHelper.setText("임시 비밀번호는 " + password + " 입니다.");
            messageHelper.setTo(id);
            msg.setRecipients(MimeMessage.RecipientType.TO, InternetAddress.parse(id));
            mailSender.send(msg);

        } catch (MessagingException e) {
            System.out.println("MessagingException");
            e.printStackTrace();
        }
        memberService.updatePw(memberDTO);

        mv.setViewName("/member/login");
        return mv;
    }
```

MEMBER

시큐리티 정보 갱신

```java
public boolean renewalAuth() {
    // 기존 정보 꺼내기
    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    // principal만 꺼내서 담기
    MemberUser userAccount = (MemberUser) authentication.getPrincipal();

    // 현재 Authentication로 사용자 인증 후, 새 Authentication 정보를 SecurityContextHolder에 세팅
    SecurityContextHolder.getContext().setAuthentication(createNewAuthentication(authentication, userAccount.getUsername()));

    return true;
}

/* 기존 권한과 사용자 id를 받아서,  new principal로 인증과 토큰 갱신해주는 메서드 */
public Authentication createNewAuthentication(Authentication currentAuth, String username) {
    // DB 가서 새로운 정보로 가져와 pricipal 새로 만들기
    UserDetails newPrincipal = memberUserDetailsService.loadUserByUsername(username);


    // 새로운 principal로 시큐리티 인증 권한(토큰) 생성
    UsernamePasswordAuthenticationToken newAuth = new
    UsernamePasswordAuthenticationToken(newPrincipal, currentAuth.getCredentials(), newPrincipal.getAuthorities());

    newAuth.setDetails(currentAuth.getDetails()); // 현재 정보 추가

    return newAuth;
}
```

MEMBER

비밀번호 변경

```
@PostMapping("updatePwPro")
public String updatePwPro(Authentication auth,
                          @RequestParam("pw1") String pw1,
                          @RequestParam("pw2") String pw2,
                          @RequestParam("pw3") String pw3, MemberDTO memberDTO, RedirectAttributes rttr) throws Exception {

    MemberUser user = (MemberUser) auth.getPrincipal();
    String password = user.getMember().getPw();

    // 현재 비밀번호와 새 비밀번호 둘 다 일치할 때, 비밀번호 변경
    if (encoder.matches(pw1, password) && pw2.equals(pw3)) {
        String changePw = encoder.encode(pw2);

        memberDTO.setPw(changePw);
        memberDTO.setId(user.getMember().getId());
        memberService.updatePw(memberDTO);

        ((MemberUser) auth.getPrincipal()).getMember().setPw(changePw);

        rttr.addFlashAttribute("msg1", "비밀번호 변경이 완료되었습니다.");
    }
    // 현재 비밀번호는 일치하지만 새 비밀번호가 일치하지 않음.
    if (encoder.matches(pw1, password) && !pw2.equals(pw3)) {
        rttr.addFlashAttribute("msg2", "새 비밀번호가 일치하지 않습니다.");

        return "redirect:/member/mypage/updatePw";
    }
    // 새 비밀번호는 일치하지만 현재 비밀번호가 일치하지 않음.
    if (!encoder.matches(pw1, password) && pw2.equals(pw3)) {
        rttr.addFlashAttribute("msg3", "현재 비밀번호가 일치하지 않습니다.");

        return "redirect:/member/mypage/updatePw";
    }
    // 시큐리티 정보 갱신
    renewalAuth();

    return "redirect:/member/mypage/profile";
}
```

GALLERY

메타데이터

```
public MetadataDTO checkMetadata(String imgpath) {
  MetadataDTO metadataDTO = new MetadataDTO();

  File file = new File(imgpath);

  try {
      Metadata metadata = ImageMetadataReader.readMetadata(file);

      ExifSubIFDDirectory Directory = metadata.getFirstDirectoryOfType(ExifSubIFDDirectory.class);
      ExifIFD0Directory Directory2 = metadata.getFirstDirectoryOfType(ExifIFD0Directory.class);
      ExifSubIFDDescriptor descriptor = new ExifSubIFDDescriptor(Directory);

      if (Directory != null) {

        Double Aperture = Directory.getDoubleObject(Directory.TAG_FNUMBER);
        Date date = Directory.getDate(ExifSubIFDDirectory.TAG_DATETIME_ORIGINAL);

        Double ISO = Directory.getDoubleObject(Directory.TAG_ISO_EQUIVALENT);
        Double FocalLength = Directory.getDoubleObject(Directory.TAG_FOCAL_LENGTH);
        Double ShutterSpeed = Directory.getDoubleObject(Directory.TAG_SHUTTER_SPEED);
        Double ShutterSpeed2 = Directory.getDoubleObject(Directory.TAG_EXPOSURE_TIME);
        Double Width = Directory.getDoubleObject(Directory.TAG_EXIF_IMAGE_WIDTH);
        Double Height = Directory.getDoubleObject(Directory.TAG_EXIF_IMAGE_HEIGHT);
        String LensModel = Directory.getString(Directory.TAG_LENS_MODEL);


        String CameraModel = Directory2.getString(Directory2.TAG_MODEL);

        // 값 DTO에 저장
        metadataDTO.setM_APERTURE(Aperture);
        metadataDTO.setM_ISO(ISO);
        metadataDTO.setM_FLENGTH(FocalLength);
        metadataDTO.setM_SHUTTERSPEED(ShutterSpeed2);

        if(Width > 3000){
            metadataDTO.setM_IMGHQLY(Width);
        } else
            metadataDTO.setM_IMGLQLY(Width);

            metadataDTO.setM_IMGCAMERA(CameraModel);
            metadataDTO.setM_IMGLENS(LensModel);

            log.info(metadataDTO + "metadataDTO");

            //todo DB에 저장 (이거아니고, 나중에 사용자가 저장버튼 누르면 그때 디비에 저장 )
            //int MetadataUploadResult = galleryMapper.insertMetadata(metadataDTO);

        }
```

GALLERY

메타데이터 뿌리기

```
$(document).ready(function () {
        $("#imgFile").on('change', function () {
            console.log("change");
            console.log(this.files[0]);
            var form = $("#imgFile")[0].files[0];
            var formData = new FormData();
            formData.append("files", form);
            console.log(form);
            console.log(formData);
            // ajax로 파일 업로드하고 메타데이터 받아와 화면에 뿌리기
            $.ajax({
                url: "/gallery/getMetadata.json",
                type: "POST",
                enctype: "multipart/form-data",
                data: formData,
                processData: false,
                contentType: false,
                cache: false,
                beforeSend: function (xhr) {
                    xhr.setRequestHeader(header, token);
                },
                success: function (result) {
                    console.log("success");
                    console.log(result);
                    //$("#orgName").val(result.orgName);
                    // 폼태그에 가져온 메타데이터 정보 작성해주기
                    $("#M_IMGCAMERA").val(result.m_IMGCAMERA);
                    $("#M_IMGLENS").val(result.m_IMGLENS);
                    $("#M_APERTURE").val(result.m_APERTURE);
                    $("#M_ISO").val(result.m_ISO);
                    $("#M_FLENGTH").val(result.m_FLENGTH);
                    $("#M_SHUTTERSPEED").val(result.m_SHUTTERSPEED);
                    $("#M_HNAME").val(result.m_HNAME);
                    $("#M_CONTENT").val(result.m_CONTENT);

                },
                error: function (e) {
                    console.log("error.........");
                    console.log(e);
                }
            });
        });
    });
```

BLOG

#태그 관련

```java
@Override
public int insertBlog(BlogDTO blogDTO) {

  //태그 등록
  String[] tag = blogDTO.getB_TAG1().substring(1).trim().split("#");

  List<String> tags = Stream.of(tag).collect(Collectors.toList());

  blogMapper.upsertTag(tags);

  return blogMapper.insertBlog(blogDTO);
}
```

#태그 관련 SQL

```
<update id="upsertTag" parameterType="java.util.List">
  BEGIN
  <foreach collection="list" item="item" separator="">
      MERGE INTO TAGLIST T
      USING DUAL
      ON (T.T_NAME = #{item})
      WHEN MATCHED THEN
      UPDATE SET T.T_COUNT = T.T_COUNT + 1, T.T_DAYCOUNT = T.T_DAYCOUNT + 1
      WHEN NOT MATCHED THEN
      INSERT VALUES (TAGLIST_SEQ.NEXTVAL, #{item}, 1, 1);
  </foreach>
  END;
</update>
```

ADMIN

스케줄러

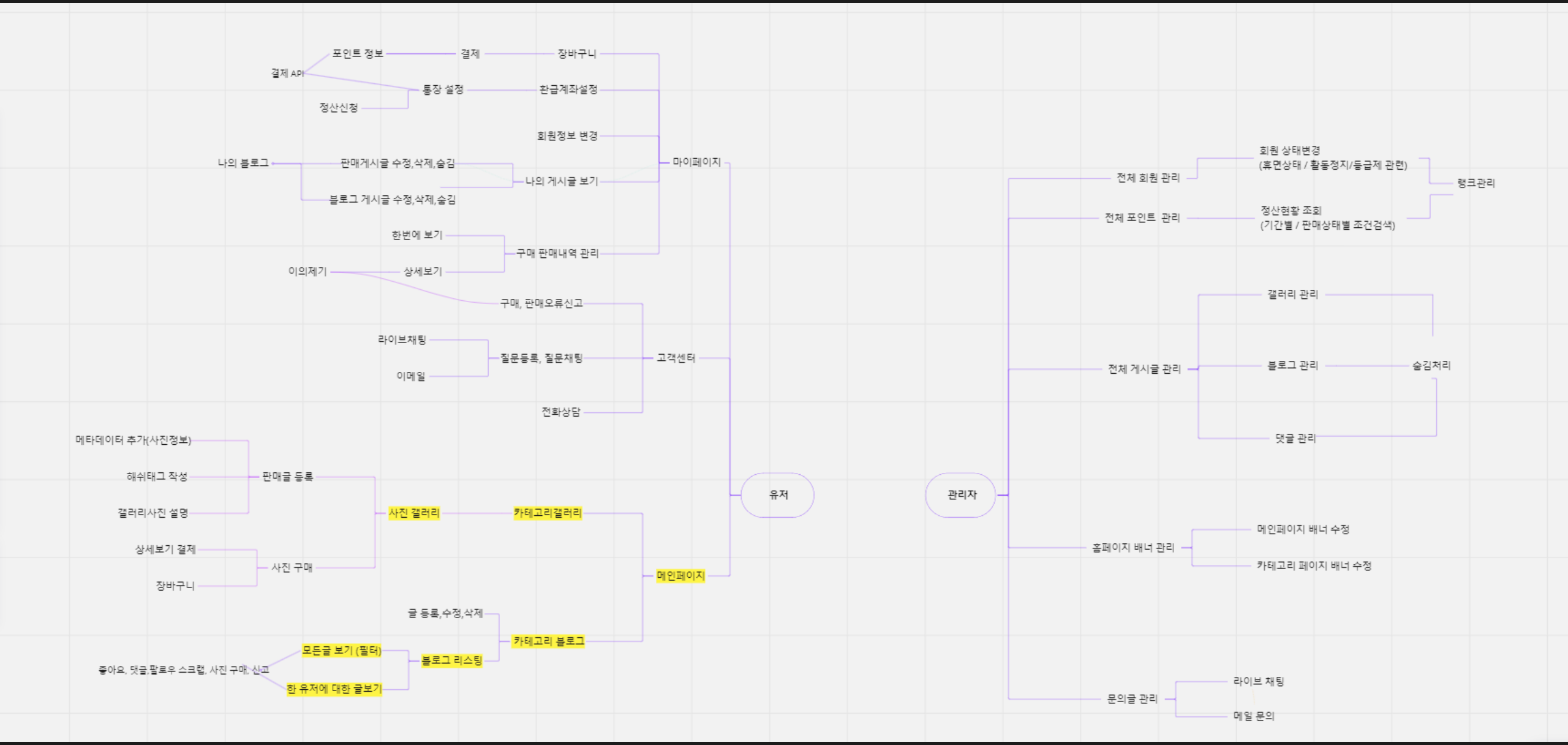
```
//3시 정각 맴버쉽 시작 + 데일리 카운트 태그 초기화 + 블로그 임시 사진 저장소 삭제
@Scheduled(cron = "0 0 3 * * *")
public void testPrint() throws Exception {
    Admin_MemberShip admin_memberShip = null;
    Admin_MemberShipInfo admin_memberShipInfo = null;

    //모든 유저들 원하는 정보 list로 담겨있음
    List<Admin_MemberShip> list = adminMapper.getUserMemberList();

    //유저 멤버쉽 비교를 위해 데이터들 가져옴.
    List<Admin_MemberShipInfo> membership = adminMapper.getMemberShipInfo();

    for (int i = 0; i < list.size(); i++) {
        admin_memberShip = list.get(i);
        for (int j = 0; j < membership.size(); j++) {
            admin_memberShipInfo = membership.get(j);

            if (admin_memberShip.getSinceDate() > admin_memberShipInfo.getMs_att() &&
                admin_memberShip.getSellerCount() > admin_memberShipInfo.getMs_sell() &&
                admin_memberShip.getGalleryCount() > admin_memberShipInfo.getMs_regist()) {
                admin_memberShip.setU_status(admin_memberShipInfo.getMs_membership());
            }

        }

    }

    int result = adminMapper.updateUserMemberShip(list);
    int delresult = adminMapper.deleteDailyCountTag();

    Date date = new Date();

    int deleteBlogTempImgresult = adminMapper.deleteBlogTempImg();

}
```

ADMIN

스케줄러 SQL

```
<select id="getUserMemberList" resultType="com.admin.domain.Admin_MemberShip">
  select A.u_id, A.u_status, sinceDate, galleryCount, sellerCount
  from
   <![CDATA[
       (select u_id, SYSDATE - U_REG sinceDate, U_STATUS from USER_LIST where u_reg < SYSDATE) A,
  ]]>
  (select u_id, count(u_id) galleryCount from gallery group by u_id) B,
       (select o_seller, count(o_seller) sellerCount from buy group by o_seller) C
  where a.u_id = b.u_id(+)
    and a.u_id = c.o_seller(+)
</select>
```

유즈케이스 다이어그램


유저
마이페이지
장바구니
결제
포인트 정보
결제 API
환급계좌설정
통장 설정
정산신청
회원정보 변경
나의 게시글 보기
나의 블로그
판매게시글 수정,삭제,숨김
블로그 게시글 수정,삭제,숨김
구매 판매내역 관리
한번에 보기
상세보기
이의제기
구매, 판매오류신고
고객센터
질문등록, 질문채팅
라이브채팅
이메일
전화상담
메인페이지
카테고리갤러리
사진 갤러리
판매글 등록
메타데이터 추가(사진정보)
해쉬태그 작성
갤러리사진 설명
사진 구매
상세보기 결제
장바구니
카테고리 블로그
블로그 리스팅
글 등록,수정,삭제
모든글 보기 (필터)
한 유저에 대한 글보기
좋아요, 댓글,팔로우 스크랩, 사진 구매, 신고
관리자
전체 회원 관리
회원 상태변경
(휴면상태 / 활동정지/등급제 관련)
랭크관리
전체 포인트 관리
정산현황 조회
(기간별 / 판매상태별 조건검색)
전체 게시글 관리
갤러리 관리
블로그 관리
숨김처리
댓글 관리
홈페이지 배너 관리
메인페이지 배너 수정
카테고리 페이지 배너 수정
문의글 관리
라이브 채팅
메일 문의

향후 계획

- 공유기능
- 팔로우
- 중고거래
- 스팟지도

07. 소감


목표
학원
계획
스프링
취업

Q&A