

SNACK _24

간식 정기 구독 웹 서비스 구현



snack

한국소프트웨어기술진흥협회

대용량 웹 서비스를 위한 MSA-Full Stack

SW 개발자 양성과정 213기 3조

김성휘 고석우 강성빈 남후승 임세혁

contents

1. 기획의도 및 개발환경

2. 팀원 및 개발일정

3. 프로젝트 소개

4. 시연 & QnA

01

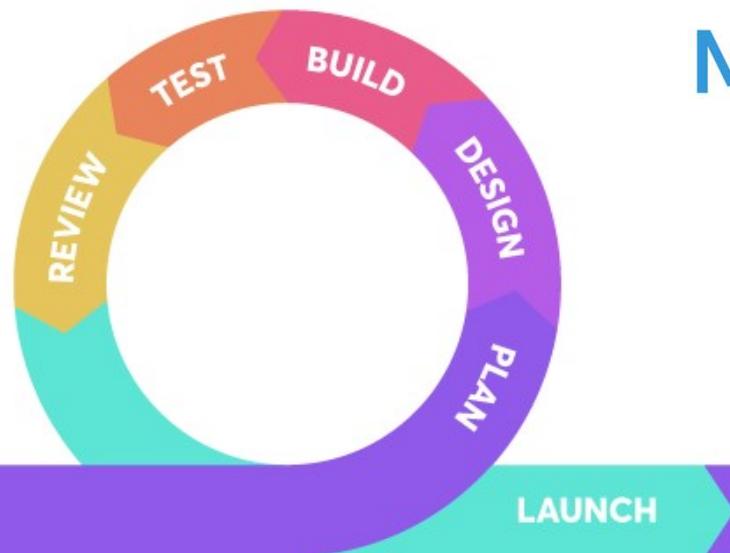
기획의도





기획의도

Agile

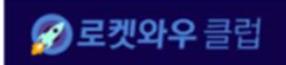


Microservices

구독 + 콘텐츠



구독 + 유통사



구독 + 소프트웨어



구독 + 버티컬 커머스



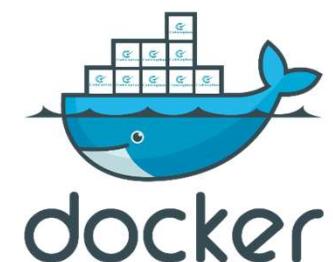
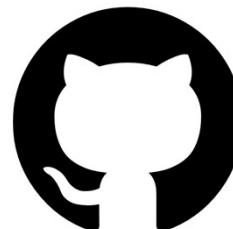
구독 + 렌탈



구독 + 오프라인 매장



개발 환경 및 사용 기술



Thymeleaf



JPA
Java Persistence API





02

팀원 및 개발일정

팀원소개



강성빈

- imPort 결제모듈 API
- 카카오페이 정기결제
- 패키지 구성



고석우

- 팀장
- 고객관리
- 로그인, 회원가입



김성휘

- PM
- 자유게시판, 문의게시판
- 후기, 별점 및 평점
- Git 관리



남후승

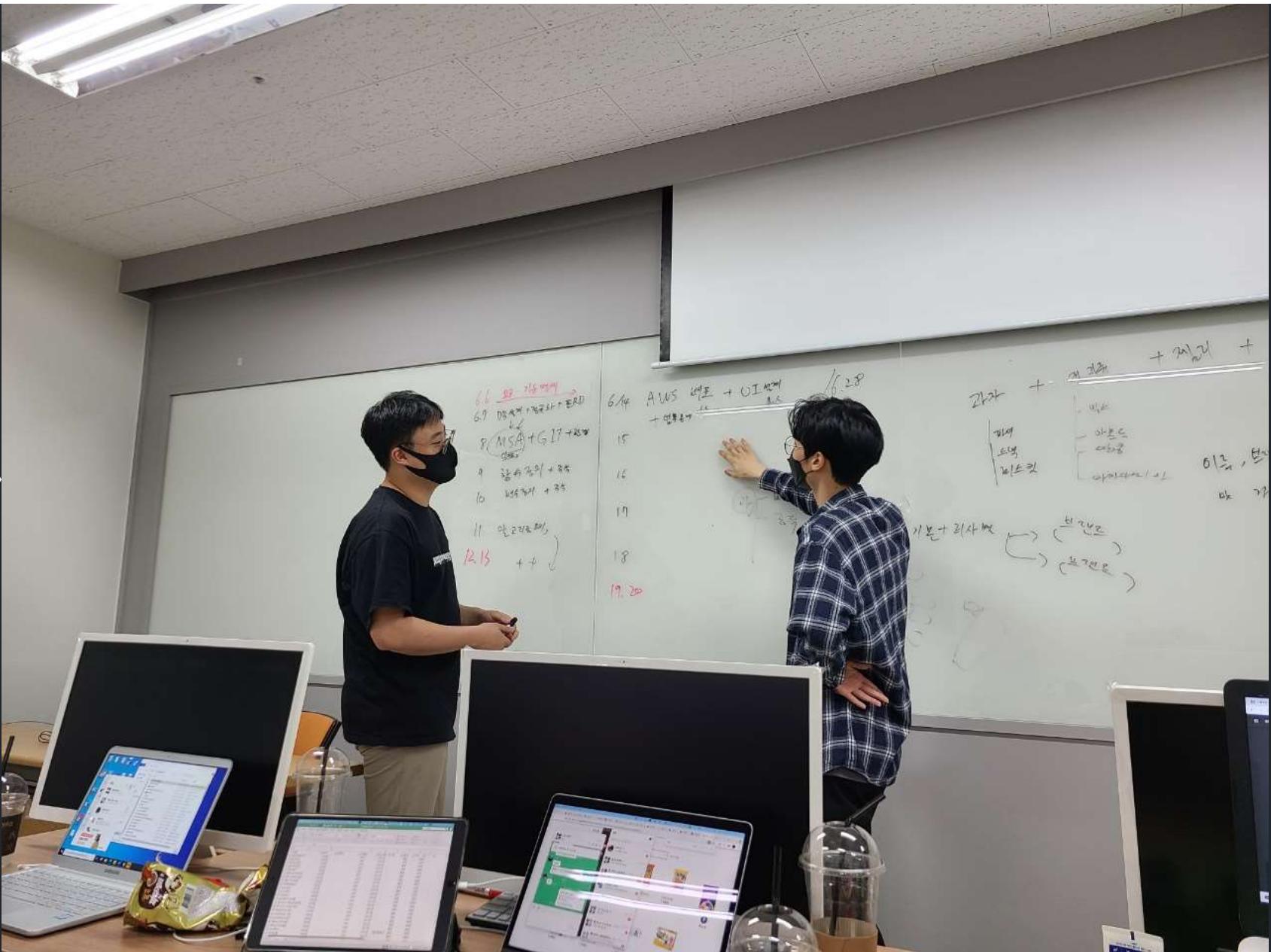
- 패키지 커스터마이징
- 배송관리
- 패키지 관리자 기능



임세혁

- 데이터 관리

개발일정



개발일정



03 Sanck24

프로젝트 소개

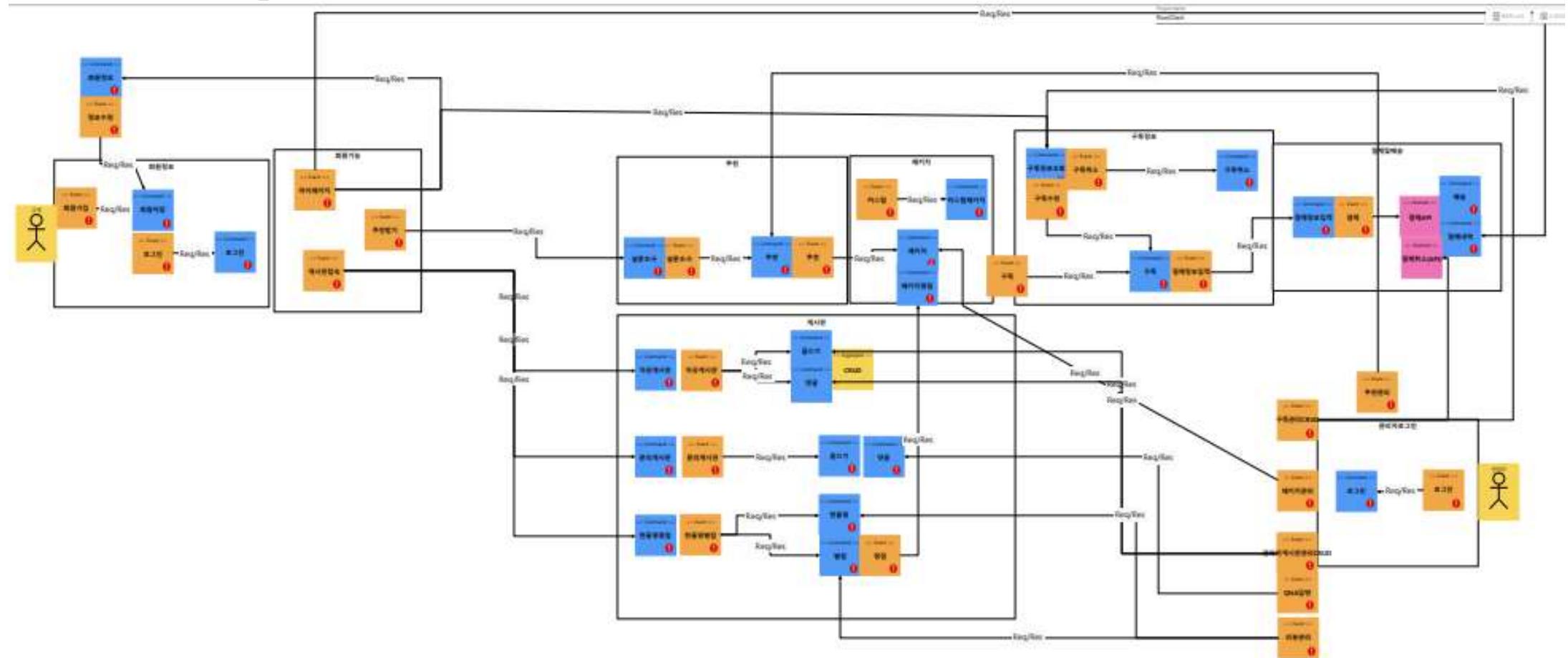


프로젝트 소개

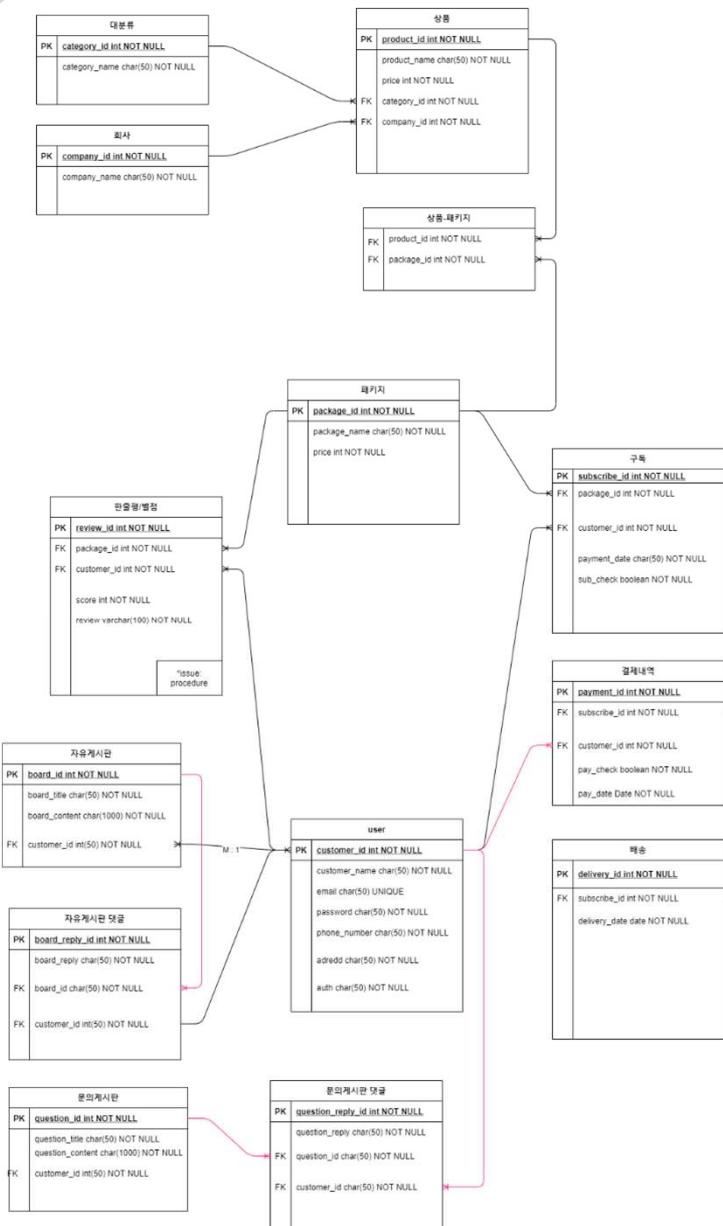
- 사무실, 가정, 학교 등을 대상으로 하는 간식 정기 구독 서비스를 구현한다.
- 사용자는 구성되어 있는 간식 패키지 상품을 선택하거나, 혹은 자신이 원하는 간식 종류들을 골라 나만의 패키지를 구성하여 구독을 신청할 수 있다.
- 구독을 신청하면 매 월 자동으로 결제가 진행되며, 사용자가 구독 취소를 요청하면 결제가 종료된다.
- 간식 제품 및 패키지들에 대해 자유롭게 의견을 나눌 수 있는 과자 갤러리(게시판), 간식 패키지 상품에 대해 한 줄 평 및 별점을 남길 수 있는 한 줄 리뷰 게시판이 존재 한다.



프로젝트 소개



프로젝트 소개



프로젝트 소개

고석우

- ✓ 고객관리 기능1 로그인 회원가입
- ✓ 마이페이지



로그인 가능

The screenshot shows a website layout with a header and a main content area. The header includes a logo with a blue star-like icon and the word 'snack'. Navigation links include 'Home', '과자모음' (Snack Collection), '구독서비스' (Subscription Service), '상세보기' (View Details) with a red 'HOT' badge, 'Contact', and search icons. The main content has two sections: one for new users ('저희 사이트에 처음이신가요?') with a '회원 가입' button, and one for existing users ('어서오세요! 지금 로그인해주세요') with fields for '이메일' and '비밀번호', a '로그인 유지' checkbox, and a 'LOG IN' button. A link for forgotten passwords ('비밀번호를 잊으셨나요?') and a red circular arrow icon are also present.

localhost:8888/index

어서오세요!
지금 로그인해주세요

저희 사이트에 처음이신가요?

간식 구독 서비스는 회원제로 운영되고 있습니다. 가입을
통해 혜택을 누려보세요

회원 가입

이메일

비밀번호

로그인 유지

LOG IN

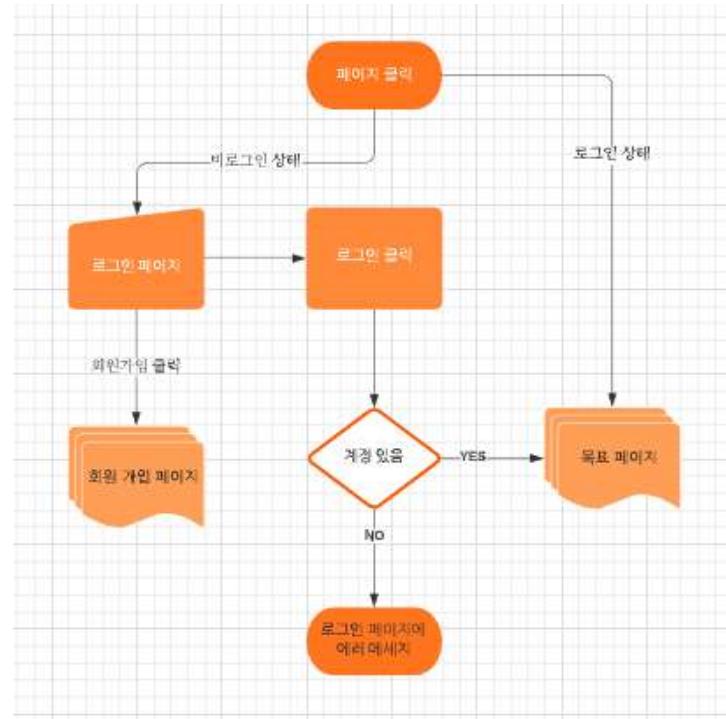
비밀번호를 잊으셨나요?

↑

畏惧 이메일과 비밀번호를 넣어서 로그인 가능



로그인 플로우차트



로그인 및 회원가입 DB 구조

COLUMN_N...	DATA_TYPE
CUSTOMER_ID	NUMBER
CUSTOMER_NAME	VARCHAR2 (255 BYTE)
EMAIL	VARCHAR2 (255 BYTE)
PASSWORD	VARCHAR2 (255 BYTE)
PHONE_NUMBER	VARCHAR2 (255 BYTE)
ADDRESS	VARCHAR2 (255 BYTE)
AUTH	VARCHAR2 (255 BYTE)
POSTNUMBER	VARCHAR2 (250 BYTE)
ADDRESSDETAIL	VARCHAR2 (250 BYTE)

- TP_USER 테이블 하나로 이루어져 있음

로그인 기능 코드

```
}

@Bean
public HttpFirewall allowUrlEncodedSlashHttpFirewall() {
    DefaultHttpFirewall firewall = new DefaultHttpFirewall();
    firewall.setAllowUrlEncodedSlash(true);
    return firewall;
}

protected void configure(HttpSecurity http) throws Exception {
    log.info("security config.....");
    // antMatchers url 패턴에 대한 접근허용
    // permitAll: 모든 사용자가 접근 가능하다는 의미
    // hasRole : 특정 권한을 가진 사용만 접근 가능하다는 의미
    http.authorizeRequests() // HttpServletRequie에 따라 접근(access)을 제한
        .antMatchers("/Home/**", "/auth/**", "/login/**", "/index").permitAll() // 누구나 접근 허용
        .antMatchers("/admin/**").hasRole("ADMIN") // /admin으로 시작하는 경로는 ADMIN권한을 가진 사용자만 접근 가능(자동으로 ROLE_가
        .anyRequest().authenticated() // 나머지 요청들은 권한의 종류에 상관 없이 권한이 있어야 접근 가능
        .and().formLogin() // form 기반으로 인증을 하도록 한다. 로그인 정보는 기본적으로 HttpSession을 이용
        .loginPage("/login") // 로그인 페이지 맵핑 .... post의 이름이 같다면 loginProcessingUrl 생략
        .defaultSuccessUrl("/index") // 로그인 성공 후 리다이렉트 주소
        .failureUrl("/login?error") // 로그인 실패시 에러 메세지
        // 스프링 시큐리티가 해당주소로 오는 요청을 가로채서 대신한다.

    permitAll()
    and()
    logout() // 로그아웃에 관한 설정을 의미
    logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
    logoutSuccessUrl("/login") // 로그아웃 성공시 리다이렉트 주소
    invalidateHttpSession(true) // 세션 지우기
    and().csrf().disable(); // csrf(크로스사이트 위조요청에 대한 설정) 트噜 비활성화 (test시에는 disable 권장)
    http.exceptionHandling().accessDeniedPage("/accessFail"); // 403 예외처리 햄들링 권한이 없는 대상이 접속을 시도했을 때
    http.rememberMe().key("RememberMe").userDetailsService(UserService).tokenRepository(tokenRepository()).tokenValiditySeconds(60*60)
}

@Bean
public PersistentTokenRepository tokenRepository() {
    JdbcTokenRepositoryImpl jdbcTokenRepository = new JdbcTokenRepositoryImpl();
    jdbcTokenRepository.setDataSource(datasource);
    return jdbcTokenRepository;
}
```

```
1 package com.kosta.KOSTA_3_final.security;
2
3 import java.util.ArrayList;
4
5
6
7
8
9
10 @Getter
11 @Setter
12 public class UserSecurity extends User{
13     private static final long serialVersionUID = 1L;
14     private static final String ROLE_PREFIX="ROLE_";
15
16     private Member member;
17     public UserSecurity(String username, String password, Collection<? extends GrantedAuthority> authorities) {
18         super(username, password, authorities);
19     }
20     public UserSecurity(Member member) {
21         super(member.getEmail(),member.getPassword(),makeRole(member));
22         this.member = member;
23     }
24     //Role을 여러개 가질수 있도록 되어있음
25     private static List<GrantedAuthority> makeRole(com.kosta.KOSTA_3_final.model.user.Member member) {
26         List<GrantedAuthority> roleList = new ArrayList<>();
27         roleList.add(new SimpleGrantedAuthority(ROLE_PREFIX + member.getAuth()));
28
29         return roleList;
30     }
31
32
33
34
35     }
36
37
38
39
40 }
```



스프링 시큐리티를 활용하여 로그인/로그아웃 처리



회원가입 가능

회원 가입

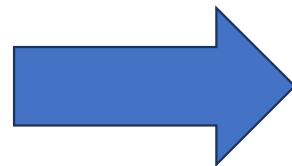
이메일

이름

비밀번호

전화번호

주소



localhost:8888 내용:
이메일을 확인하시기 바랍니다.

이메일

이름

비밀번호

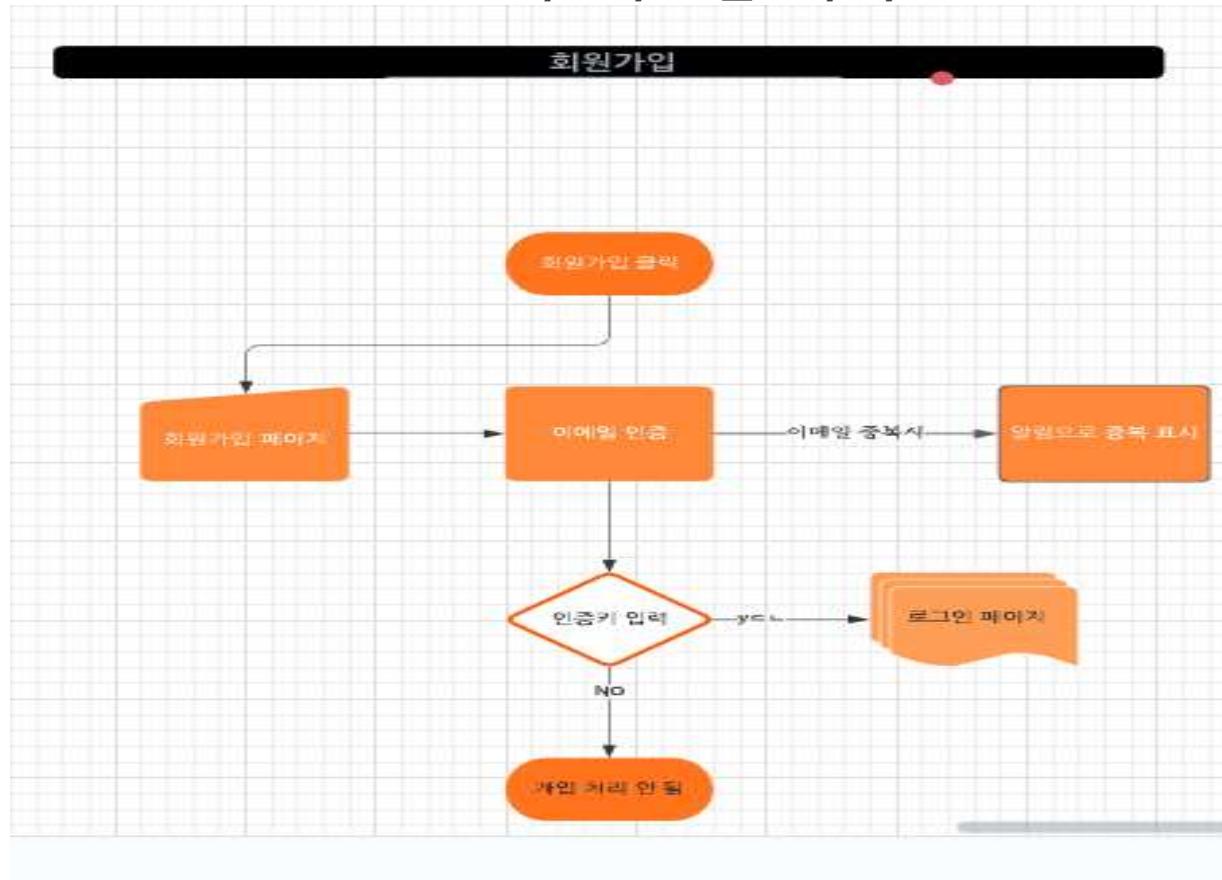
전화번호

주소

이메일을 아이디로 하여 회원 가입 가능.

이메일 미인증시 가입 불가
중복 이메일은 필터링됨

회원가입 플로우차트





```
$(function() {
    var isCertification = false;
    var email;
    $(".sendEmail").click(function() { // 메일 입력 유효성 검사
        var email = $(".email").val(); // 사용자의 이메일 입력값.

        if (email == "") {
            alert("이메일 주소가 입력되지 않았습니다.");
        }
        else {
            $.ajax({
                type: 'get',
                url: '/auth/emailCheck',
                data: {
                    email: email
                },
                datatype: 'json',
                success: function(data){
                    if(data==true){
                        alert("이미 존재하는 이메일입니다.");
                    }else{
                        $.ajax({
                            type: 'post',
                            url : '/auth/checkMail',
                            dataType : 'json',
                            async : "false",
                            data : {
                                email : email
                            },
                            success : function(data) {
                                console.log(data.key);
                                key = data.key;
                                alert("이메일을 확인하시기 바랍니다.");
                            }
                        });
                    }
                }
            });
            isCertification=true; //주소 인증 여부를 알기위한 값
            $(".compare").css("display","block");
            $(".compare-text").css("display","block");
        });
    });
});

$(".btn").click(function submitCheck(){
    if(isCertification==false){
        alert("이메일이 유효하지 않습니다.");
        return false;
    }else{
        true;
    }
});

$(".compare").on("propertychange change keyup paste input", function() {
    if ($(".compare").val() == key) { //비밀 키 값을 비교를 위해 텍스트입력과 일치를 비교
        $(".compare-text").text("인증 성공!");$(".color", "black");
        isCertification = true; //인증 상태 check
    } else {
        $(".compare-text").text("틀림!");$(".color", "red");
        isCertification = false; //인증 실패
    }
});
});
```

이메일을 중복 체크한 후 이메일에 인증 번호를 보냄.
인증 번호 미입력시 다음 페이지로 넘어가지 않음

회원가입 가능 코드

```
function Postcode() {
    new daum.Postcode({
        onComplete: function(data) {
            // 팝업에서 검색결과 항목을 클릭했을때 실행할 코드를 작성하는 부분.

            // 도로명 주소의 노출 규칙에 따라 주소를 표시한다.
            // 내려오는 변수가 값이 없는 경우엔 공백(' ')값을 가지므로, 이를 참고하여 분기 한다.
            var roadAddr = data.address; // 도로명 주소 변수
            var extraRoadAddr = ''; // 참고 항목 변수

            // 법정동명이 있을 경우 추가한다. (법정리는 제외)
            // 법정동의 경우 마지막 문자가 "동/로/가"로 끝난다.
            if(data.bname !== '' && /([동|로|가]$/).test(data.bname)){
                extraRoadAddr += data.bname;
            }
            // 건물명이 있고, 공동주택일 경우 추가한다.
            if(data.buildingName != '' && data.apartment === 'Y'){
                extraRoadAddr += (extraRoadAddr !== '' ? ', ' + data.buildingName : data.buildingName);
            }
            // 표시할 참고항목이 있을 경우, 괄호까지 추가한 최종 문자열을 만든다.
            if(extraRoadAddr !== ''){
                extraRoadAddr = '(' + extraRoadAddr + ')';
            }

            // 우편번호와 주소 정보를 해당 필드에 넣는다.
            document.getElementById('postnumber').value = ' ' +data.zonecode;
            document.getElementById("address").value = ' ' +roadAddr;
            document.getElementById("address").value += ' ' +data.jibunAddress;

            // 참고항목 문자열이 있을 경우 해당 필드에 넣는다.
            if(roadAddr !== ''){
                document.getElementById("address").value += ' ' +extraRoadAddr;
            } else {
                }
            }
        }
    }).open();
}
```

주소찾기 api를 사용해 주소를 구함.

회원가입 가능 코드

```
spring.mail.host=smtp.gmail.com  
spring.mail.port=587  
spring.mail.username=swk8632@gmail.com  
spring.mail.password=[REDACTED]  
spring.mail.properties.mail.smtp.auth=true  
spring.mail.properties.mail.smtp.starttls.enable=true
```



구글 SMTP를 적용하여 이메일 인증을 처리하기 위해 설정

회원가입 가능 코드

```
@Autowired  
private JavaMailSender sender;  
  
@PostMapping("/auth/CheckMail")  
public Map<String, Object> SendMail(String email, HttpSession session) {  
    // 투aptcha 랜덤 문자 만들기  
    Map<String, Object> map = new HashMap<>();  
    Random random = new Random();  
    String key = "";  
    // 메일 보낼 것 구성  
    MimeMessage message=sender.createMimeMessage();  
    MimeMessageHelper helper=new MimeMessageHelper(message);  
    try {  
        helper.setTo(email);  
        // 스크립트에서 보낸 메일을 받을 사용자 이메일 주소  
        // 임의 키를 위한 코드  
        for (int i = 0; i < 3; i++) {  
            int index = random.nextInt(25) + 65; // A-Z까지 한글 알파벳 생성  
            key += (char) index;  
        }  
  
        int numIndex = random.nextInt(8999) + 1000; // 4자리 정수를 생성  
        key += numIndex;  
  
        helper.setSubject("인증번호 입력을 위한 메일 전송");  
        message.setText("인증번호:"+key);  
        helper.setFrom("swk863@gmail.com");  
  
        sender.send(message);  
        map.put("key", key);  
    } catch (MessagingException e) {  
        // TODO Auto-generated catch block  
        System.out.println("문제 발생");  
        e.getMessage();  
    }  
    return map;  
}
```



무작위 문자와 숫자로 인증 번호를 만들고
JavaMailSender로 이메일 전송

비밀번호 찾기 가능

이메일

가입시 등록한 이메일을 입력하세요.

이름

가입시 등록한 이름을 입력하세요.

OK



이메일과 이름을 넣어 임시 비밀번호를 발급 가능

프로젝트 소개

고석우

- ✓ 고객관리 기능1 로그인 회원가입
- ✓ 마이페이지



마이페이지



Home 과자모음 구독서비스 상세보기 HOT Contact

Q X Y

이름	고석우
Email	swk9514@naver.com
전화번호	01086320320
우편번호	63249
주소	제주특별자치도 제주시 아봉로 88 제주특별자치도 제주시 아라이동 901 (아라이동, 다담빌리지), 103동 402호
수정 구독내역	



자신의 정보를 확인 가능.

마이페이지 코드

```
@GetMapping("/mypage/profile")
public String profile(Model model) {
    Object principal=SecurityContextHolder.getContext().getAuthentication().getPrincipal(); //로그인시 정보 받아오기
    UserSecurity ss=(UserSecurity)principal;
    Member mem=service.getMemberInfo(ss.getUsername()); //정보 찾기
    model.addAttribute("memberinfo", mem); //저장 후 마이페이지 내에 출력

    return "mypage/profile";
}
```



시큐리티에서 정보를 받아와 출력

자신 정보 수정

회원 정보 수정

이름

고석우

비밀번호

만드시 바꿔주세요

전화번호

01086320320

주소

63249

제주특별자치도 제주시 아봉로 88 제주특별자치도 제주시 아라이동 9

[우편번호 찾기](#)

103동 402호

[정보 수정](#)



자신의 정보를 수정 가능

자신 정보 수정

```
@PostMapping("/mypage/edit")
public String profileAfterEdit(Model model, Member member) {
    //회원 정보 수정
    Member mem=service.getMemberInfo(member.getEmail()); //이메일로 정보 받아옴
    mem.setAddress(member.getAddress());
    mem.setAddressdetail(member.getAddressdetail());
    mem.setCustomerName(member.getCustomerName());
    mem.setPassword(member.getPassword());
    mem.setPhone_number(member.getPhone_number());
    mem.setPostnumber(member.getPostnumber()); //정보 수정
    service.joinUser(mem); //저장

    Object principal=SecurityContextHolder.getContext().getAuthentication().getPrincipal(); //이후 로그인 상태시 정보 받아오기
    UserSecurity ss=(UserSecurity)principal;
    Member mem2=service.getMemberInfo(ss.getUsername());
    model.addAttribute("memberinfo", mem2); //이 후 데이터 저장해서 프로필 페이지에 띄우게한다
```



자신의 정보를 수정 후 시큐리티에서 수정된 값을 받아와
마이페이지에서 출력

프로젝트 소개

강성빈

- ✓ imPort 결제모듈 API, 카카오페이지 정기결제



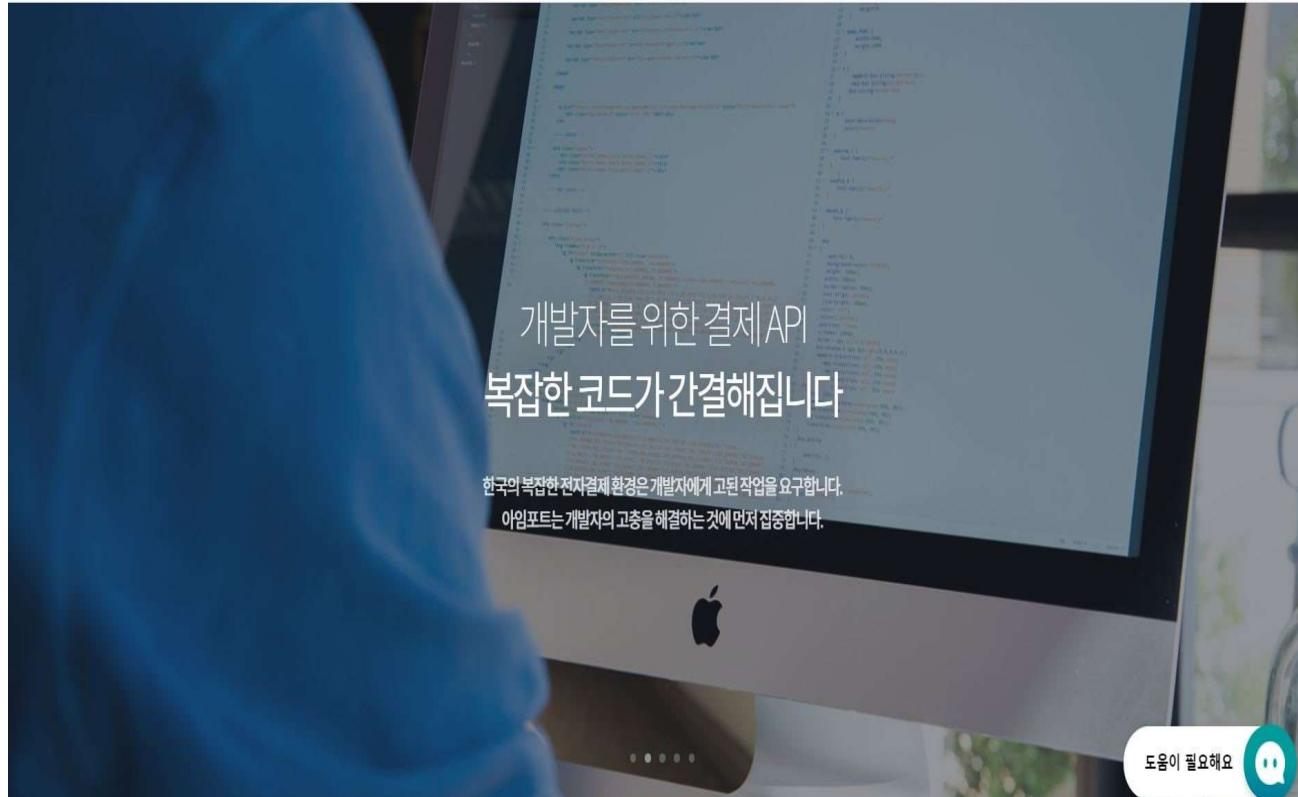


I'mport;

알아보기 개발가이드 이용요금 도움말

PG가입하기 고객 활용사례 보기

연동매뉴얼 개발관련링크 대시보드



아임포트에서 제공하는
보안 토큰 및 정기결제 관련
REST API 사용

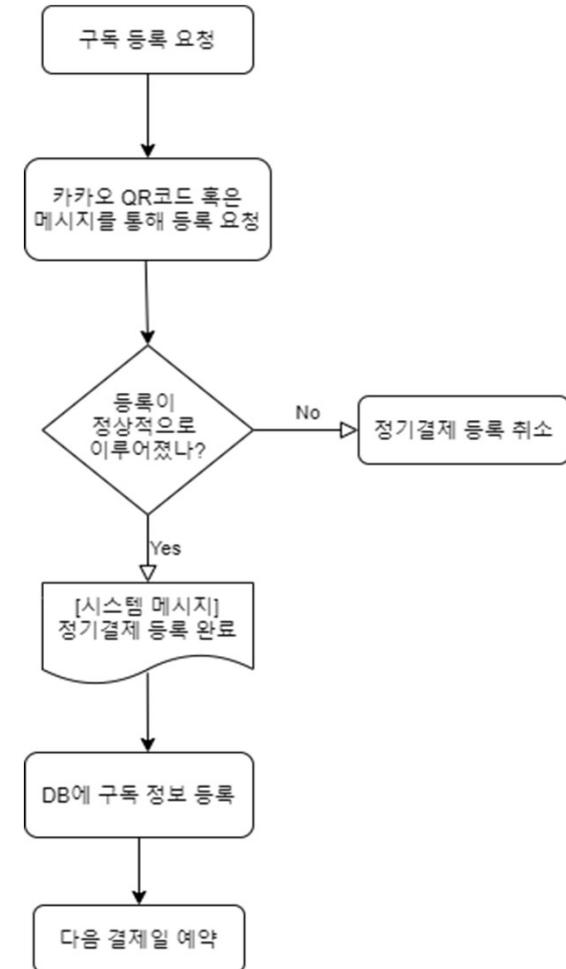


Ajax를 활용하여 웹 페이지의
Form을 통해 수집한 정보를
자바로 전송 후 RestTemplate을
사용하여 API 호출

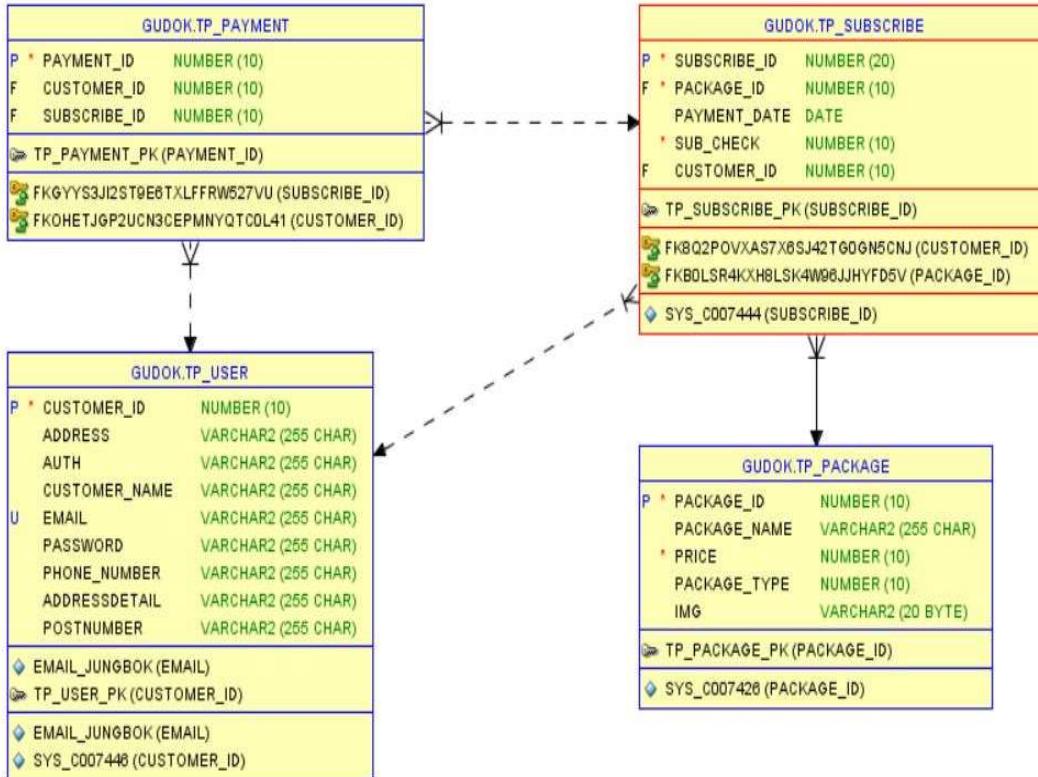
정기결제 프로세스

1. QR코드 or 카카오톡 메시지를 통한 정기결제 등록
2. 정기결제가 등록되면 데이터베이스에 구독 정보 삽입
3. 등록된 사용자 정보를 이용하여 서버에 결제요청
4. 요청한 결제가 정상적으로 이루어지면 다음 결제 일정 예약
예약일에 자동으로 결제 요청 전송

<Flowchart>



정기결제 프로세스



1. 사용자가 카카오페이지를 통해 정기결제 정보를 등록하면 구독(Subscribe) 테이블에 사용자의 정보가 등록된다.
2. 등록된 구독정보를 이용하여 서버에 결제요청을 보내면 결제(payment)테이블에 결제정보가 등록된다.
3. 각각의 테이블은 그림과 같은 참조 관계를 가지고 있다.



서버에 정기결제 등록하기

➤ 빌링 키 발급

```
<script>
    function kakaopay(){
        var IMP = window.IMP;
        var merchantid = new Date().getTime();
        IMP.init('imp41751598');//iamport에서 발급받은 고유 회원번호
        IMP.request_pay({
            //정기결제 등록
            pay_method : 'card',
            merchant_uid : merchantid,
            name : 'snack24',
            amount : 0,
            customer_uid : $('#customer_id').val(),
            buyer_email : $('#customer_email').val(),
            buyer_name : $('#customer_name').val(),
            buyer_tel : $('#customer_phone').val()
        }, function(rsp) {
            if ( rsp.success ) {
                //서버로 API호출 요청을 보내기 위해 필요한 데이터들을 ajax를 통해 controller로 전송
                $.ajax({
                    url:'paymentrequest',
                    type : 'POST',
                    data:{
                        "customer_uid" : $('#customer_id').val(),
                        "price" : $('#package_price').val(),
                        "merchant_uid" : merchantid,
                        "packageId" : $('#package_id').val(),
                    },
                    success:function(result) {
                        window.location.href = "/index";
                    }
                });
            }
        });
    }
</script>
```



서버에 정기결제 등록하기

1. 사용자가 '정기결제' 버튼을 클릭하면 QR코드 혹은 카카오 메시지를 통한 결제정보 등록창이 나타난다.
2. 사용자가 등록을 완료하면 고유 정기결제 정보를 저장하는 빌링 키가 발급되어 서버에 저장된다. 해당 빌링 키는 오직 아임포트 서버에서만 관리하며, 따라서 서비스 관리자는 각 정기결제에 대해 고유한 `merchant_id`를 저장하여야 한다.

빌링키란?

구독형 정기결제, 종량제 과금 결제 등 원하는 시점에 재결제를 진행할 수 있는 결제용 암호화 키이다.

가맹점이 고객의 카드정보를 소유할 수 없기 때문에 카드사로부터 해당 카드에 대응하는 빌링 키를 발급 받는다.

이때 발급받은 빌링 키를 저장하고, 원하는 시점에 해당 빌링 키로 결제를 청구할 수 있다.

빌링 키 발급을 위해서는 카드사에 카드번호, 카드 유효기간, 생년월일(또는 사업자등록번호), 비밀번호 앞 2자리를 전달한다.

최초결제 요청

- Controller를 통해 매개변수 받기

```
@Controller
public class PaymentController {
    @Autowired
    ReqPaymentScheduler scheduler;
    @Autowired
    RequestSubPayment reqSubpay;
    @PostMapping("/payment1")
    public @ResponseBody void getImportToken(@RequestParam Map<String, Object> map)
        throws JsonMappingException, JsonProcessingException {

        long customer_uid = Long.parseLong((String) map.get("customer_uid"));
        int price = Integer.parseInt((String) map.get("price"));
        long merchant_uid = Long.parseLong((String) map.get("merchant_uid"));
        long packageId = Long.parseLong((String) map.get("packageId"));
        reqSubpay.requestSubPay(customer_uid, merchant_uid, price);
        scheduler.startScheduler(customer_uid, price, packageId);
    }
}
```

- 웹 페이지의 kakaopay() 메소드에서 ajax 통신을 통해 전송한 데이터들을
@RequestParam으로 넘겨받은 후 필요한 메서드로 다시 전달한다.



최초결제 요청

➤ Controller를 통해 매개변수 받기

- 웹 페이지의 kakaopay()에서 컨트롤러로 전송한 ajax 요청의 데이터를 Map<String, Object> 형태로 받아온다.
 - map.get()을 통해 원하는 Object 형태의 value에 접근한 뒤 적합한 형태로 Parsing 한다.
- ✓ @RequestParam은 요청 파라미터를 메서드에서 1:1로 받기 위해 사용한다. 이를 통해 사용자가 원하는 매개변수에 데이터를 매핑할 수 있다. 만일 해당 어노테이션을 생략하더라도 Spring에서는 사용자가 입력한 파라미터의 키 값과 매개변수의 이름을 비교하여 적절한 값을 넣어준다. 그러나 원하는 이름으로 값을 받기 위해 @RequestParam을 적절히 활용한다.



Access_Token 발급받기

```
@Service
public class ImportPay {
    public String getToken() {
        RestTemplate restTemplate = new RestTemplate();
        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON); // 서버로 요청할 Header
        Map<String, Object> map = new HashMap<>();
        map.put("imp_key", "5945712414472844");
        map.put("imp_secret", "92c551f52382ba2dfa3e77e87c83dd75250f4906f917d271ba391197849fb8a4a63d6be5d72783fd");
        Gson var = new Gson();
        String json = var.toJson(map);
        HttpEntity<String> entity = new HttpEntity<>(json, headers); // 서버로 요청할 Body
        return restTemplate.postForObject("https://api.iamport.kr/users/getToken", entity, String.class);
    }
}
```

- 아임포트에서 제공하는 REST API를 사용하기 위해 권한 필요
회원에게 발급되는 key와 secret 코드를 아임포트 서버로 보내면 access_token을 포함한 json 형태의 데이터를 받을 수 있다.



Access_Token 발급받기

```
@Service
public class RequestSubPayment {
    @Setter(onMethod_ = @Autowired)
    private ImportPay pay;

    public String requestSubPay(long customer_uid, long merchant_uid, int price) {
        String token = pay.getToken();
        Gson str = new Gson();
        token = token.substring(token.indexOf("response") + 10);
        token = token.substring(0, token.length() - 1);
        GetTokenVO vo = str.fromJson(token, GetTokenVO.class);
        String access_token = vo.getAccess_token();
    }
}
```

Response Class (Status 200)

Model Model Schema

```
{
    "code": 0,
    "message": "string",
    "response": {
        "access_token": "string",
        "expired_at": 0,
        "now": 0
    }
}
```

- getToken()의 결과는 오른쪽 위와 같은 형태이다.
substring()과 Gson을 사용하여 필요한 access_token 데이터만 추출한다.



➤ Json 데이터 직렬화/역직렬화

- GSON은 json 구조를 띠는 직렬화 데이터를 JAVA의 객체로 역직렬화, 혹은 직렬화 해주는 자바 라이브러리이다. 즉 JSON Object -> Java Object 또는 그 반대의 행위를 돋는다.
- `fromJson`을 통해 json형태의 데이터를 GetTokenVO 객체에 매핑한다.
- GSON에서 제공하는 여러 라이브러리를 통해 `JsonObject` 혹은 `JSONArray`형태의 데이터들을 활용하기 보다 편리하게 변환할 수 있다.

Access_Token 발급받기

- 문자열에서 원하는 정보만 따로 얻어내기

```
@Service
public class RequestSubPayment {
    @Setter(onMethod_ = @Autowired)
    private ImportPay pay;

    public String requestSubPay(long customer_uid, long merchant_uid, int price) {
        String token = pay.getToken();
        Gson str = new Gson();
        token = token.substring(token.indexOf("response") + 10);
        token = token.substring(0, token.length() - 1);
        GetTokenVO vo = str.fromJson(token, GetTokenVO.class);
        String access_token = vo.getAccess_token();
    }
}
```

```
import lombok.Data;
@Data
public class GetTokenVO {
    private String access_token;
    private long now;
    private long expired_at;
}
```

- 우선 Response model에서 "response"에 대응하는 value인 access_token, now, expired_at 을 token에 저장한다.
- 문자열 형태의 token을 Gson에서 제공하는 fromJson 함수를 통해 GetToken 객체로 저장한다.
- GetTokenVO의 getter를 통해 access_token에 해당하는 값을 문자열의 형태로 얻어 API 권한 요청에 사용한다.



토큰과 데이터를 이용하여 결제 요청 전송

```
RestTemplate restTemplate = new RestTemplate();
HttpHeaders headers = new HttpHeaders();
headers.setContentType(MediaType.APPLICATION_JSON);
headers.setBearerAuth(access_token);

Map<String, Object> map = new HashMap<>();
map.put("customer_uid", customer_uid);
map.put("merchant_uid", merchant_uid);
map.put("amount", price);
map.put("name", "test05");

Gson var = new Gson();
String json = var.toJson(map);
HttpEntity<String> entity = new HttpEntity<>(json, headers);

return restTemplate.postForObject("https://api.iamport.kr/subscribe/payments/again", entity, String.class);
}
```



토큰과 데이터를 이용하여 결제 요청 전송

1. HTTP 요청 후 JSON, XML, String 과 같은 형태로 응답을 받을 수 있는 `RestTemplate`를 사용하여 Rest API 서비스를 요청한다. 이를 통해 단순히 메소드를 호출하는 것 만으로 복잡한 작업을 쉽게 처리할 수 있었다.
2. `HttpHeader`를 사용하여 전송할 요청의 `header`를 설정한다. 요청의 `body` 부분은 `map`을 활용하여 `key`와 `value`의 형태로 저장한 후 `Gson` 라이브러리를 통해 json 데이터로 `parsing`한다. `header`를 설정할 때는 반드시 서버에서 요구하는 데이터들을 포함해야 한다.
3. `HttpEntity`를 활용하여 `header`와 `body`를 합친 후, 아임포트의 Rest API가 `post` 형식을 요구하기 때문에 `postForObject`를 통해 데이터를 전송한다. `getForObject`는 `header`를 추가할 수 없기 때문에 `exchange()` 메서드를 사용해야 한다.

토큰과 데이터를 이용하여 결제 요청 전송

- exchange 메서드를 사용하여 get요청 전송하기

```
@JsonIgnoreProperties
@Service
public class GetPayementStatus {
    public String paymentStatus(long merchantUid) throws JsonMappingException, JsonProcessingException {
        String merchant_uid = Long.toString(merchantUid);
        String access_token = vo.getAccess_token();
        String url = "https://api.iamport.kr/subscribe/payments/schedule/" + merchant_uid;
        RestTemplate restTemplate = new RestTemplate();
        HttpHeaders headers = new HttpHeaders();
        headers.setBearerAuth(access_token);
        HttpEntity<JsonObject> entity = new HttpEntity<>(headers);
        ResponseEntity<String> respEntity = restTemplate.exchange(url, HttpMethod.GET, entity, String.class);
```

토큰과 데이터를 이용하여 결제 요청 전송

➤ `exchange` 메서드를 사용하여 `get`요청 전송하기

1. 위는 `get` 방식으로 결제 상태 조회 API를 호출하는 코드이다. 호출 url의 끝에 조회하고자 하는 결제의 고유번호인 `merchant_uid`를 더해 조회하는 방식이다.
2. `HttpHeaders`를 통해 `header`의 `bearerAuth`에 `access_token`의 값을 저장한다.
`Http`요청 또는 응답에 해당하는 `HttpHeader`와 `HttpBody`를 포함하는 클래스인 `HttpEntity`를 통해 `JsonObject`형태의 요청에 `header` 값을 저장한다.
3. `exchange`메서드를 사용하여 요청을 보낼 주소, 방식, 보내는 데이터(여기서는 url에 `merchant_uid`를 보내기 때문에 `header`만 전송), 그리고 받을 응답의 형태를 지정한다.

정기결제 요청

➤ 결제 예약 요청 예시

```
// 결제 예약
axios({
  url: `https://api.iamport.kr/subscribe/payments/schedule`,
  method: "post",
  headers: { "Authorization": access_token }, // 인증 토큰 Authorization header에 추가
  data: {
    customer_uid: "gildong_0001_1234", // 카드(빌링키)와 1:1로 대응하는 값
    schedules: [
      {
        merchant_uid: "order_monthly_0001", // 주문 번호
        schedule_at: 1519862400, // 결제 시도 시각 in Unix Time Stamp. ex. 다음 달 1일
        amount: 8900,
        name: "월간 이용권 정기결제",
        buyer_name: "홍길동",
        buyer_tel: "01012345678",
        buyer_email: "gildong@gmail.com"
      }
    ]
  });
});
```

1. **header**에는 API 사용 권한 획득을 위한 **Token**이 포함된다.
2. **schedules**에 대응하는 값은 배열의 형태로 저장된다.
3. 결제 시도 시각은 **Unix Time Stamp** 형태로 저장된다.

정기결제 요청

- Unix Time 형식으로 결제 스케줄 설정

```
@Service
public class SchedulePayment {
    @Setter(onMethod_ = @Autowired)
    private ImportPay pay;
    public String schedulePay(long customer_uid, int price) {
        String token = pay.getToken();
        long timestamp = 0;
        Calendar cal = Calendar.getInstance();
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy/MM/dd HH:mm", Locale.KOREA);
        cal.add(Calendar.MONTH, +1);
        String date = sdf.format(cal.getTime());
        try {
            Date stp = sdf.parse(date);
            timestamp = stp.getTime()/1000;
            System.out.println(timestamp);
        } catch (ParseException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

- 결제 등록을 한 날부터 한 달 뒤 자동으로 결제 요청을 보내기 위해 한달 뒤의 날짜를 UnixTimeStamp 형식으로 변환하여 저장한다.



정기결제 요청

- 필요한 값들을 올바른 형태로 전송

```
RestTemplate restTemplate = new RestTemplate();
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    headers.setBearerAuth(access_token);

    JSONObject jsonObject = new JSONObject();
    jsonObject.addProperty("merchant_uid", timestamp);
    jsonObject.addProperty("schedule_at", timestamp);
    jsonObject.addProperty("amount", price);

    JSONArray jsonArr = new JSONArray();

    jsonArr.add(jsonObject); JsonObject reqJson = new JSONObject();

    reqJson.addProperty("customer_uid", customer_uid);
    reqJson.add("schedules",jsonArr);
    String json = str.toJson(reqJson);
    System.out.println(json);
    HttpEntity<String> entity = new HttpEntity<>(json, headers);
    return restTemplate.postForObject("https://api.iamport.kr/subscribe/payments/schedule", entity, String.class);
}
```

정기결제 요청

➤ 필요한 값들을 올바른 형태로 전송

1. 결제 예약을 올바르게 요청하기 위해 아임포트 서버에서 요구하는 요청의 형태를 이해해야 한다.
2. `schedules`에 대응하는 값을 배열의 형태로 저장하기 위해 우선 `JsonObject` 객체를 선언한 후 저장할 값을 넣어준다.
3. `schedules` 키에 대응하는 값을 `JSONArray` 객체에 넣어 필요에 맞는 형태로 바꾼다.
4. 데이터를 `reqJson` 변수에 저장한 후 전송한다.

✓ JSONArray

- `JSONArray` 안에 여러 `JsonObject`를 담을 수 있다.
- 각 오브젝트나 문자열의 구분은 ';'로 한다.
- 순서를 가진 SET이다.



정기결제 요청

- 구독 정보 DB에 INSERT

```
<input type="hidden" id="package_id" th:value="${pack.packageId}">
<input type="hidden" id="customer_id" th:value="#${authentication.principal.member.customer_id}">
```

```
$.ajax({
    url: '/insertSubscribe',
    data: {
        package_id : $('#package_id').val(),
        customer_id : $('#customer_id').val()
    },
    success:function(result) {
    }
});
```

- Thymeleaf를 통해 구독 테이블에 입력할 값을 얻는다.
- Ajax요청에 필요한 값들을 포함하여 insert를 요청한다.



정기결제 요청

➤ 구독 정보 DB에 INSERT

```
@Autowired
SubscribeService subService;
@GetMapping("/insertSubscribe")
@ResponseBody
public void insertSubscribe(long package_id, int customer_id) {
    subService.insertSubscribe(package_id, customer_id);
    log.info("구독정보 입력성공");
}

@Service
public class SubscribeService {
    @Autowired
    MemberRepository memberRepo;
    @Autowired
    PackageRepository packRepo;
    @Autowired
    SubscribeRepository subRepo;
    public void insertSubscribe(long package_id, long customer_id ){
        PackageVO pack= packRepo.findById(package_id).get();
        Member member = memberRepo.findById(customer_id).get();
        Subscribe sub = Subscribe.builder()
            .subscribeId(new Date().getTime())
            .pack(pack)
            .customer(member)
            .build();
        subRepo.save(sub);
    }
}
```

- Controller에서는 SubscribeService의 insertSubscribe()에 매개변수를 전달한다.
- Service에선 전달받은 값을 이용하여 구독 객체를 생성한 후, SubscribeRepository에 save한다.
- JPA Repository를 통해 간편하게 DB 테이블에 정보를 입력할 수 있다.

스케줄러를 사용하여 매달 결제 요청

- 컨트롤러에서 스케줄러로 매개변수 전달

```
@Controller
public class PaymentController {
    @Autowired
    ReqPaymentScheduler scheduler;
    @Autowired
    RequestSubPayment reqSubpay;
    @PostMapping("/payment1")
    public @ResponseBody void getImportToken(@RequestParam Map<String, Object> map)
        throws JsonMappingException, JsonProcessingException {

        long customer_uid = Long.parseLong((String) map.get("customer_uid"));
        int price = Integer.parseInt((String) map.get("price"));
        long merchant_uid = Long.parseLong((String) map.get("merchant_uid"));
        long packageId = Long.parseLong((String) map.get("packageId"));
        reqSubpay.requestSubPay(customer_uid, merchant_uid, price);
        scheduler.startScheduler(customer_uid, price, packageId);
    }
}
```

- startScheduler() 메소드로 결제 요청에 필요한 정보 전달

스케줄러를 사용하여 매달 결제 요청

```
@Component
public class ReqPaymentScheduler {
    private ThreadPoolTaskScheduler scheduler;
    @Autowired
    SchedulePayment setSchedulePay;
    @Autowired
    DeliveryService deliService;
    @Autowired
    GetDate getDate;
    public void stopScheduler() {
        //구독 취소 시 scheduler shutdown을 통해 결제 요청 멈춤
        scheduler.shutdown();
    }

    public void startScheduler(long customer_uid, int price, long packageId) {
        scheduler = new ThreadPoolTaskScheduler();
        scheduler.initialize();
        // 스케줄러가 시작되는 부분
        scheduler.schedule(getRunnable(customer_uid, price, packageId), getTrigger());
    }
}
```

1. 스케줄러를 시작한다.
2. 업무를 수행할 `getRunnable()` 메소드에 필요한 매개변수를 전달하며 호출한다.
3. 결제 일정 예약을 위해 필요한 사용자 아이디, 결제 가격, 주문한 상품 아이디는 컨트롤러에서 매개변수로 받아온다.

스케줄러를 사용하여 매달 결제 요청

```
private Runnable getRunnable(long customer_uid, int price, long packageId){  
    //스케줄러가 실행할 로직  
    Date date = getDate.getDate();  
    Calendar cal = Calendar.getInstance();  
    cal.setTime(date);  
    cal.add(Calendar.MONTH, 1);  
    cal.add(Calendar.DATE, 1);  
    Date date = convertFromJAVADateToSQLDate(cal.getTime());  
    return () -> {  
        setSchedulePay.schedulePay(customer_uid, price);  
        deliService.deliveryInsert(packageId, customer_uid, date);  
    };  
}
```

- 스케줄러는 아임포트 서버로 예약 결제 요청을 보내고, 동시에 데이터베이스의 배송 테이블에 배송정보를 입력한다.

- ✓ 결제가 이루어지는 날로부터 한달 뒤의 날짜를 구하기 위해 **Calendar** 인스턴스를 사용하였다.
- ✓ 데이터베이스에 삽입할 **date**의 값은 **java.sql**에 포함된 **date**이어야 하는데 **Calendar**를 통해 얻은 **date** 값은 **java.util**에 속한 값이다. 따라서 이를 변환하는 작업이 추가되었다.

스케줄러를 사용하여 매달 결제 요청

```
public static java.sql.Date convertFromJAVADateToSQLDate(java.util.Date javaDate) {  
    java.sql.Date sqlDate = null;  
    if (javaDate != null) {  
        sqlDate = new Date(javaDate.getTime());  
    }  
    return sqlDate;  
}
```

→ java.util.Date를 java.sql.Date로 변환하는 함수

```
private Trigger getTrigger() {  
    // 작업 주기 설정  
    return new PeriodicTrigger(1, TimeUnit.MINUTES);  
}
```

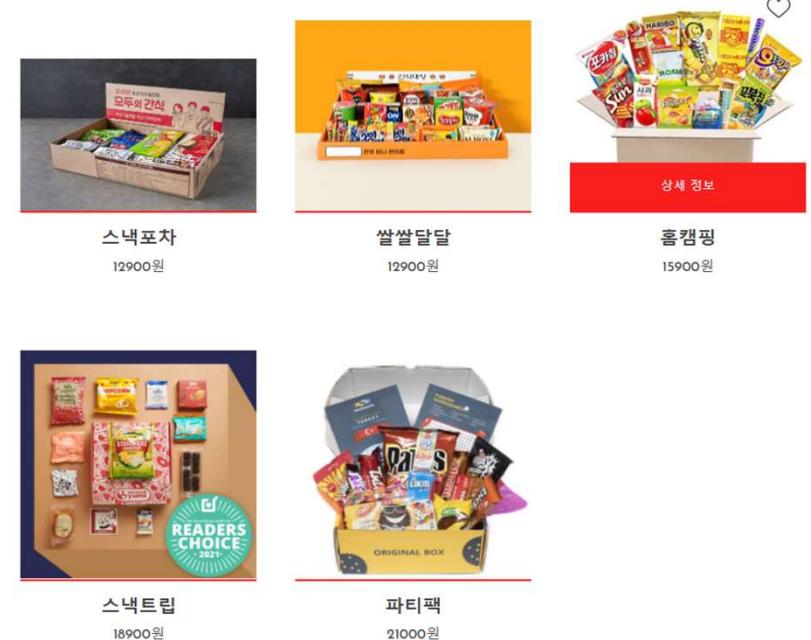
→ 스케줄러의 작업 주기 설정

✓ 스케줄러는 관리자가 지정한 시간마다 `schdulepay()` 메서드를 수행한다. 사용자가 정기결제를 등록함과 동시에 스케줄러가 시작된다.

✓ 데이터베이스에 삽입할 `date`의 값은 `java.sql`에 포함된 `date`이어야 하는데 `Calendar`을 통해 얻은 `date` 값은 `java.util`에 속한 값이다. 따라서 이를 변환하는 작업이 추가되었다.

실행 예시

```
<div class="row" th:each="pack, packState2:${plist}" >
    <div class="col-xl-4 col-lg-4 col-md-6 col-sm-6" >
        <div class="single-popular-items mb-50 text-center" >
            <div class="popular-img" >
                
                <div class="img-cap" >
                    <a th:href="@{/product_details(pno=${pack.packageId})}"><span>상세 정보</span></a>
                </div>
                <div class="favorit-items" >
                    <span class="flaticon-heart" style="color: #f08080;"></span>
                </div>
            </div>
            <div class="popular-caption" style="text-align: center;" >
                <h3><a th:href="@{/product_details(pno=${pack.packageId})}" th:text="${pack.packageName}"></a></h3>
                <span th:text="| ${pack.price}원 |"></span>
            </div>
        </div>
    </div>
</div>
```



- Thymeleaf를 사용하여 데이터베이스에 저장된 정보 웹 페이지에 출력

간식 패키지 리스트



JavaScript code

➤ 주문자 정보, 배송 정보

```
<script>
$(function(){
    $("#downArrow").click(function(){
        if($(".UserInfo").css("display")=="none"){
            $(".UserInfo").show();
            $("#orderInfo").hide();

        }else{
            $(".UserInfo").hide();
            $("#orderInfo").show();
        }
    });
    function setDisplay(){
        if($('input:radio[id=selectDelivery1]').is(':checked')){
            $('#same_address').show();
            $('#new_address').hide();
        }else{
            $('#new_address').show();
            $('#same_address').hide();
        }
    }
</script>
```

➤ 우편번호 찾기

```
<!-- 주소찾기 -->
<script>
function Postcode1() {
    new daum.Postcode({
        onComplete: function(data) {
            var roadAddr = data.address;
            var extraRoadAddr = '';
            if(data.bname != '' && /[동|로|가]$/g.test(data.bname)){
                extraRoadAddr += data.bname;
            }
            if(data.buildingName != '' && data.apartment === 'Y'){
                extraRoadAddr += (extraRoadAddr != '' ? ', ' + data.buildingName : data.buildingName);
            }
            if(extraRoadAddr != ''){
                extraRoadAddr = '(' + extraRoadAddr + ')';
            }
            document.getElementsByClassName('postnumber1')[0].value = '' +data.zonecode;
            document.getElementsByClassName('postnumber1')[1].value = '' +data.zonecode;
            document.getElementsByClassName("address1")[0].value = '' +roadAddr;
            document.getElementsByClassName("address1")[1].value = '' +roadAddr;
            document.getElementsByClassName("address1")[0].value += '' +data.jibunAddress;
            document.getElementsByClassName("address1")[1].value += '' +data.jibunAddress;
            if(roadAddr != ''){
                document.getElementsByClassName("address1").value += '' +extraRoadAddr;
            } else {
                }
            }
        }).open();
}
</script>
```



실행 예시

주문/결제

주문자 정보

• 주문자

• 이메일

• 휴대전화

• 주소
경기 김포시 김포한강8로 365 경기 김포시 구래동 6888-4 (구래동, 구)



배송지 정보

• 배송지 선택 주문자 정보와 동일 새 배송지 입력

• 주문자

• 이메일

• 휴대전화

• 주소
경기 김포시 김포한강8로 365 경기 김포시 구래동 6888-4 (구래동, 구)



주문/결제

주문자 정보

강성빈 (12) ▼

배송지 정보

• 배송지 선택 주문자 정보와 동일 새 배송지 입력

• 주문자

• 이메일

• 휴대전화

• 주소
경기 김포시 김포한강8로 365 경기 김포시 구래동 6888-4 (구래동, 구)

주문자의 정보 확인.

javascript를 활용하여 주문자 정보를
펼쳤다 접을 수 있다.

실행 예시

- 주문자 정보와 동일하게 배송지 정보를 저장하거나,
- 혹은 새 배송지를 입력할 수 있다.

주문/결제

주문자 정보

• 주문자

• 이메일

• 휴대전화

• 주소



주문/결제

주문자 정보

• 주문자

• 이메일

• 휴대전화

• 주소

배송지 정보

• 배송지 선택 주문자 정보와 동일 새 배송지 입력

• 주문자

• 이메일

• 휴대전화

• 주소

배송지 정보

• 배송지 선택 주문자 정보와 동일 새 배송지 입력

• 주문자

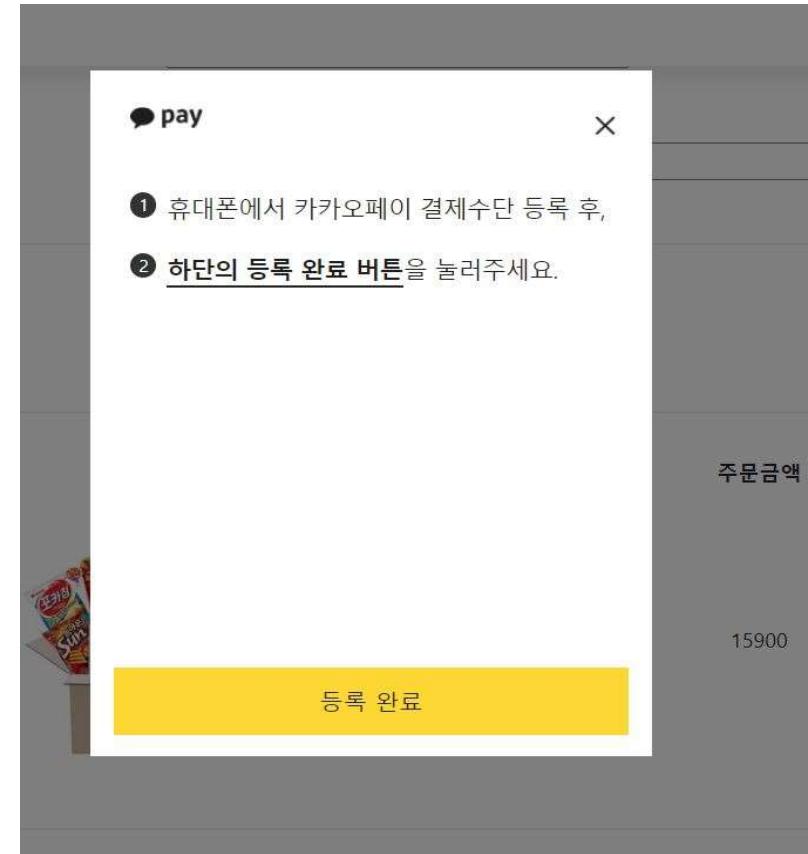
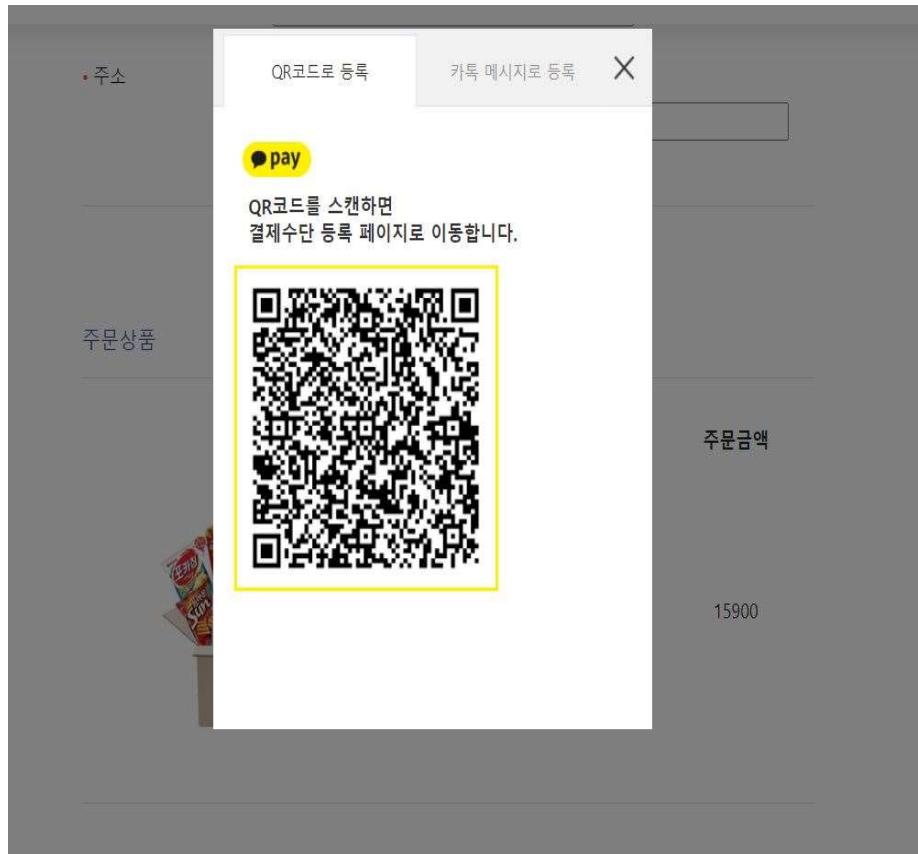
• 이메일

• 휴대전화

• 주소

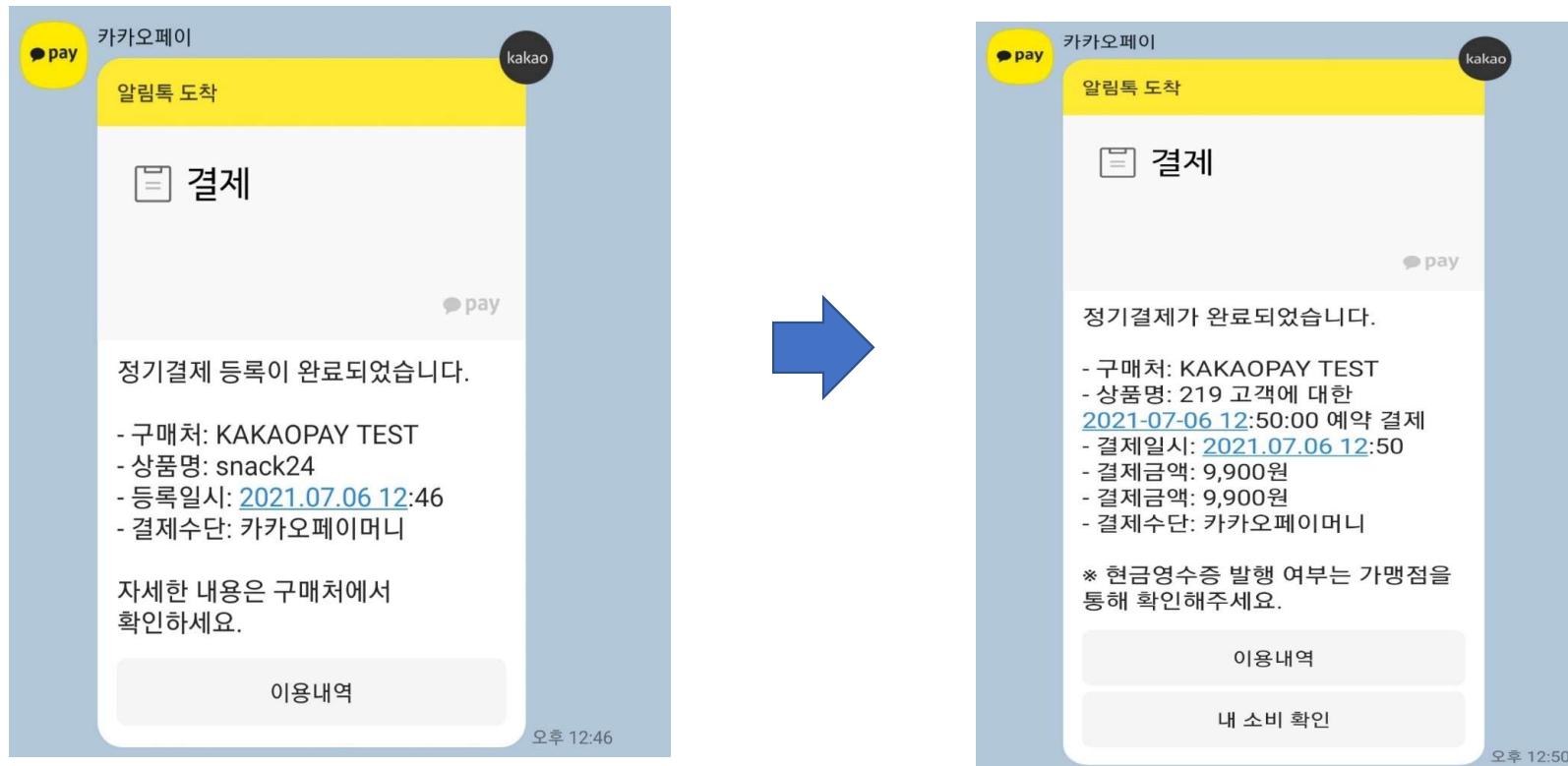
실행 예시

- 사용자가 등록 버튼을 클릭하면 QR코드 혹은 카톡 메시지
를 통해 간편한 정기결제 등록이 가능하다.



실행 예시

- 정기결제가 성공적으로 등록되면 사용자에게 카카오톡 메시지가 전송된다.
- 예약한 시간이 되면 자동으로 결제가 이루어진다.



프로젝트 소개

남후승

- ✓ 패키지 커스터마이징
- ✓ 패키지 관리자 기능
- ✓ 배송관리





커스텀 및 패키지/제품관리 MODEL Category

```
@Getter  
@Setter  
@Entity  
@ToString(exclude = "products")  
@Builder  
@AllArgsConstructor  
@NoArgsConstructor  
@Table(name="tp_largecl")  
public class Category {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    int categoryId;  
    String categoryName;  
  
    @JsonIgnore  
    @OneToMany(mappedBy = "category", cascade = CascadeType.ALL, fetch = FetchType.LAZY)  
    List<Product> products;  
}
```

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID
1 CATEGORY_ID	NUMBER(10,0)	No	(null)	1
2 CATEGORY_NAME	VARCHAR2(255 CHAR)	Yes	(null)	2

LIBRARY.TP_PRODUCT

P	*	PRODUCT_ID	NUMBER (10)
F	*	CATEGORY_ID	NUMBER (10)
F	*	COMPANY_ID	NUMBER (10)
	*	PRICE	NUMBER (10)
		PRODUCT_NAME	VARCHAR2 (255 CHAR)
TP_PRODUCT_PK (PRODUCT_ID)			
FK9PVO0OQ6G1JKMVMFEL2RQFLQM (CATEGORY_ID)			
FKQQIO7HA5B43B06LFXXEM8QTL2 (COMPANY_ID)			

LIBRARY.TP_LARGECL

P	*	CATEGORY_ID	NUMBER (10)
		CATEGORY_NAME	VARCHAR2 (255 CHAR)
TP_LARGECL_PK (CATEGORY_ID)			



커스텀 및 패키지/제품관리 MODEL PackageVO

```

@Getter
@Setter
@Entity
@ToString(exclude = {"reviews","subscribes","ppList"})
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Table(name="tp_package")
public class PackageVO {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "package_id")
    long packageId;

    @Column(name = "package_name")
    String packageName;

    @Column(name = "package_type")
    int packageType;

    int price;

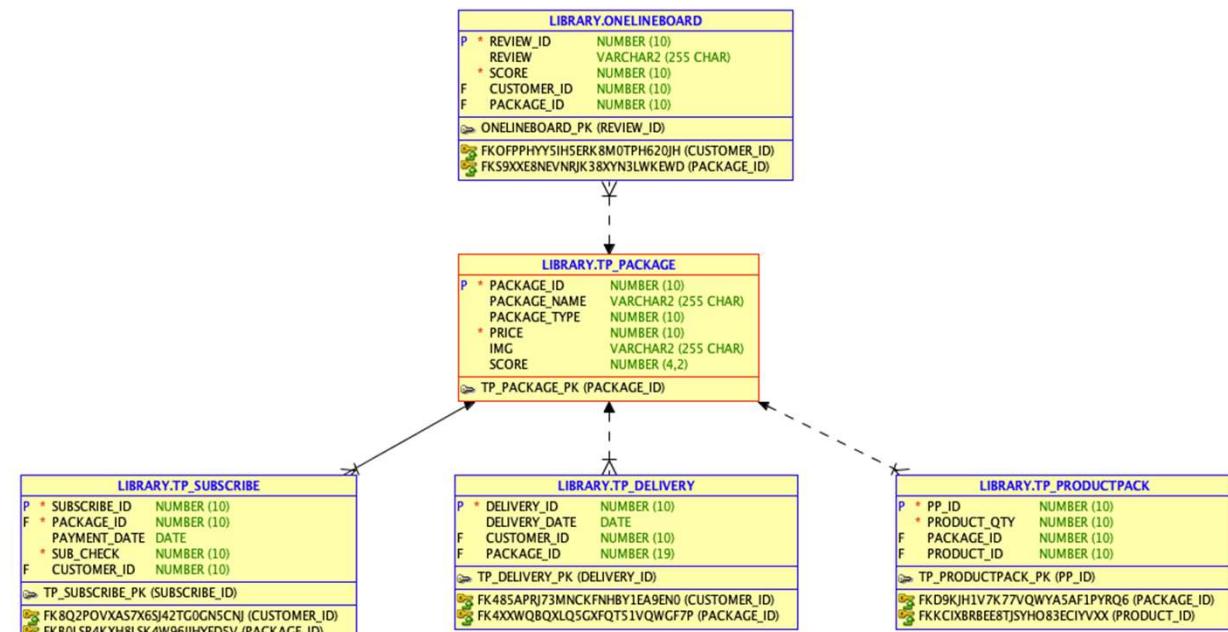
    @Column(columnDefinition = "number(4,2) default 0")
    float score;

    String img;
    @JsonIgnore
    @OneToMany(mappedBy = "pack", cascade = CascadeType.ALL)
    List<Review> reviews;

    @JsonIgnore
    @OneToMany(mappedBy = "pack", cascade = CascadeType.ALL)
    List<Subscribe> subscribes;

    @OneToMany(mappedBy = "pack")
    List<Product_Package> ppList;
}
  
```

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID
1 PACKAGE_ID	NUMBER(10,0)	No	(null)	1
2 PACKAGE_NAME	VARCHAR2(255 CHAR)	Yes	(null)	2
3 PACKAGE_TYPE	NUMBER(10,0)	Yes	(null)	3
4 PRICE	NUMBER(10,0)	No	(null)	4
5 IMG	VARCHAR2(255 CHAR)	Yes	(null)	5
6 SCORE	NUMBER(4,2)	Yes	0	6

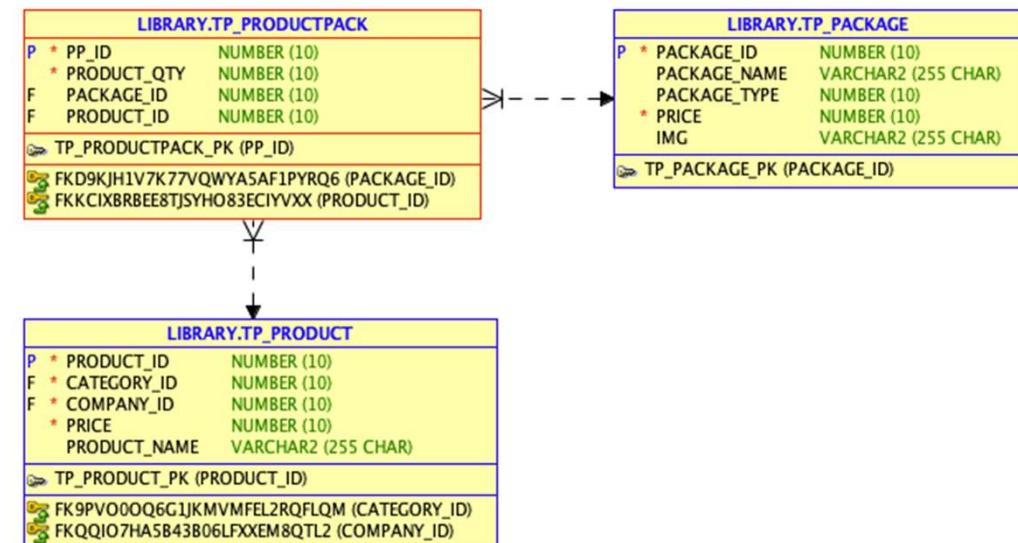




커스텀 및 패키지/제품관리 MODEL Product_Package

```
@Getter  
@Setter  
@ToString  
@Builder  
@AllArgsConstructor  
@NoArgsConstructor  
@Entity  
@Table(name = "tp_productpack")  
@SequenceGenerator(  
    name = "PP_ID_EX_SEQ",  
    sequenceName = "PP_ID_SEQ", // 매핑할 데이터베이스 시퀀스 이름  
    allocationSize = 1)  
public class Product_Package {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.SEQUENCE,  
    generator = "PP_ID_EX_SEQ")  
    int ppId;  
  
    @ManyToOne(fetch = FetchType.LAZY)  
    @JoinColumn(name = "product_id")  
    Product product;  
  
    @ManyToOne(fetch = FetchType.LAZY)  
    @JoinColumn(name = "package_id")  
    PackageVO pack;  
  
    int product_qty;  
}
```

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID
1 PP_ID	NUMBER(10,0)	No	(null)	1
2 PRODUCT_QTY	NUMBER(10,0)	No	(null)	2
3 PACKAGE_ID	NUMBER(10,0)	Yes	(null)	3
4 PRODUCT_ID	NUMBER(10,0)	Yes	(null)	4





커스텀 및 패키지/제품관리 MODEL Product

```

@Getter
@Setter
@Entity
@ToString(exclude = {"category", "company", "ppList"})
@Builder
@AllArgsConstructorConstructor
@NoArgsConstructorConstructor
@Table(name="tp_product")
@SequenceGenerator(
    name = "PP_ID_EX_SEQ",
    sequenceName = "PP_ID_SEQ", // 매핑할 데이터베이스 시퀀스 이름
    allocationSize = 1)
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE,
    generator = "PP_ID_EX_SEQ")
    @Column(name = "product_id")
    int productId;
    @Column(name = "product_name")
    String productName;
    int price;
    @Column(name = "category_id")
    int categoryId;
    @Column(name = "company_id")
    int companyId;

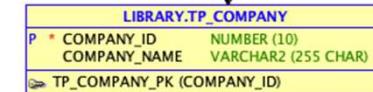
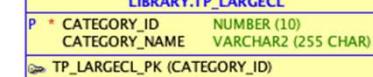
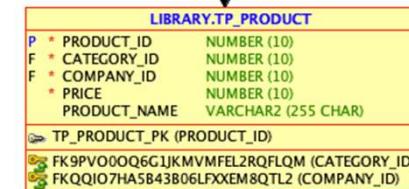
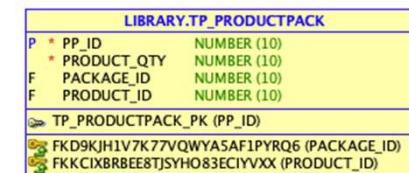
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "category_id", insertable = false, updatable = false)
    Category category;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "company_id", insertable = false, updatable = false)
    Company company;

    @JsonIgnore
    @OneToMany(mappedBy = "product", cascade = CascadeType.ALL,
        fetch = FetchType.LAZY)
    List<Product_Package> ppList;
}

```

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID
1 PRODUCT_ID	NUMBER(10,0)	No	(null)	1
2 CATEGORY_ID	NUMBER(10,0)	No	(null)	2
3 COMPANY_ID	NUMBER(10,0)	No	(null)	3
4 PRICE	NUMBER(10,0)	No	(null)	4
5 PRODUCT_NAME	VARCHAR2(255 CHAR)	Yes	(null)	5



커스텀 및 패키지/제품관리 MODEL Company

```

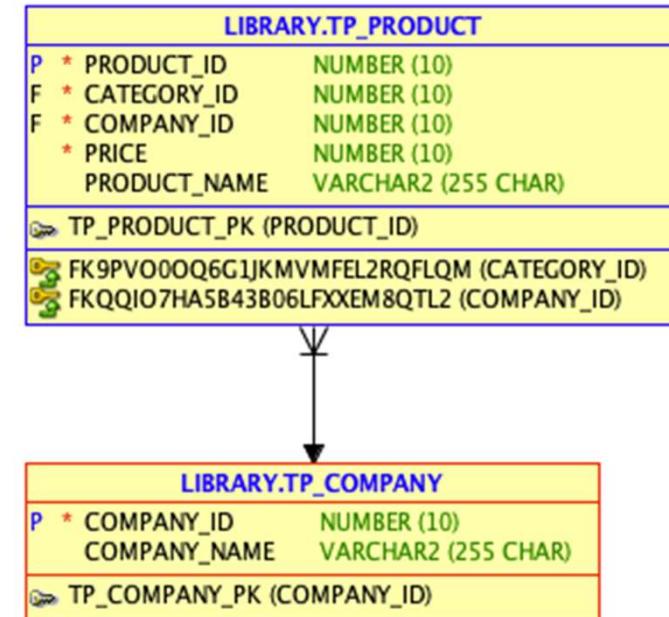
@Getter
@Setter
@Entity
@ToString(exclude = "products")
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Table(name="tp_company")
public class Company {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    int company_id;
    String company_name;

    @JsonIgnore
    @OneToMany(mappedBy = "company", cascade = CascadeType.ALL, fetch = FetchType.LAZY)
    List<Product> products;
}

```

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID
1 COMPANY_ID	NUMBER(10,0)	No	(null)	1
2 COMPANY_NAME	VARCHAR2(255 CHAR)	Yes	(null)	2





패키지 커스터마이징 화면 기능

```
@GetMapping("/product/customPackaging3")
public String selectProduct1(Model model) {
    model.addAttribute("categorylist", productService.selectCategoryAll());
    model.addAttribute("productlist", productService.selectProductAll());
}
return "/product/customPackaging4";
```

The screenshot shows a web application interface for product customization. It includes a navigation bar, a search bar, a category grid, a product list, a selection area, and a summary section.

- ① Category Grid: A grid showing categories like 과과류 (Fruit), 과자 (Snacks), 껌 (Gum), 마른안주 (Dried Snacks), and 냉 (Cold). Sub-categories like 아이스크림 (Ice Cream), 음료 (Beverage), 젤리 (Jelly), 초콜릿 (Chocolate), and 캔디 (Candy) are listed under their respective main categories.
- ② Search Bar: A search input field with placeholder "상품명을 입력해주세요." (Enter product name) and a "검색" (Search) button.
- ③ Product List: A grid of products with images, names, and prices. Examples include 콘초 초코가나슈 (1200원), 콘치 치즈크림 (1200원), 콜드오렌지 (1400원), 쿨드포도 (1400원), 퍼시침 사워아니언 맛 (1500원), 퍼시침 트리플솔트 맛 (1500원), and 쿠크디스.
- ④ Selection Area: A panel where selected items are dragged and dropped. It contains the message "상품을 드래그해서 담아주세요!" (Drag and drop the product here!).
- ⑤ Summary: A summary box showing "선택상품삭제" (Delete selected item), "장바구니비우기" (Empty cart), "상품개수: 0개" (Number of products: 0), and "합계금액: 0원" (Total amount: 0 won).
- ⑥ Order Button: A button labeled "선택한 상품 주문" (Order selected products).

- ① 대분류 카테고리로 클릭 시 카테고리에 해당하는 제품이 표출
- ② 제품명 또는 제품명에 포함되는 키워드로 검색 시 해당하는 제품이 표출
- ③ 조건에 맞는 제품이 표출되는 공간
- ④ 표출된 제품을 드래그 앤 드롭 할 시 리스트로 표출
- ⑤ 리스트에서 선택한 상품을 리스트에서 삭제하는 버튼과 선택한 모든 상품을 리스트에서 삭제하는 버튼.
상품 개수, 합계금액, 패키지에 따른 할인 금액이 표출
- ⑥ 선택한 상품을 주문하는 버튼



대분류 카테고리 클릭 / 키워드 검색

견과류	과자	껌
아이스크림	음료	젤리
상품검색 <input type="text" value="상품명을 입력해주세요."/>		

허니버터 땅콩
1900원

허니버터 아몬드
4500원

군옥수수맛 땅콩
1900원

군옥수수맛 아몬드
4500원

뉴 에브리데이 낫초
300원

떡볶이맛 아몬드
4500원

미니봉+아몬드

그린비트

작사작과

견과류	과자	껌
아이스크림	음료	젤리
상품검색 <input type="text" value="깡"/>		

감자깡
1000원

고구마깡
1200원

매운 생새우로 만든 새우깡
600원

생새우로 만든 새우깡
900원

쌀생새우로 만든 새우깡
900원

양파깡
1000원

- 견과류 카테고리 클릭 시 대분류가 견과류인 제품들이 표출
- "깡" 키워드 입력 시 "깡"을 포함한 모든 제품이 표출



대분류 카테고리 클릭 / 키워드 검색

- 대분류 카테고리

```
<div class="category_row">
  <div class="col-xs-12" style="padding: 0px">
    <div class="product_category">
      <div class="category" th:each="categorylist:${categorylist}">
        <button class="category_name"
          th:text="${categorylist.categoryName}" name="category"
          th:value="${categorylist.categoryId}"></button>
      </div>
    </div>
  </div>
</div>
```

- 키워드 검색

```
<div id='search_form'>
  <div class='search_box'>
    <span style="margin-right:46px">상품검색</span>
    <input type='text' id="search_keyword" placeholder="상품명을 입력해주세요.">
    <button id="search_product">검색</button>
  </div>
</div>
```

- 카테고리 클릭 및 검색 JavaScript

```
// 카테고리 클릭/키워드 검색 기능
$(".category_name, #search_product").on("click",function() {
  //
  var typeStr = $('.category_name').attr("name");
  var keywordStr;

  // 카테고리 클릭인지 키워드 검색인지 조건 구분 if문
  if($(this).hasClass('category_name')) {
    keywordStr = $(this).val();
  } else {
    typeStr = 'product';
    keywordStr = $('#search_keyword').val();
  }
  $.ajax({
    type:"GET",
    url:"/product/productList",
    data:{
      type:typeStr,
      keyword:keywordStr
    },
    success : function(list){
      $("#area1").html(list);
    },
    complete: function() {
      // ajax 이후에도 함수 활성화
      drag();
      adminShowAndHide();
    }
  });
});
```

- 검색 버튼 누를 시 카테고리 클릭 또는 검색 조건 구분 후 ajax 호출



드래그 앤 드롭 기능 및 리스트 업

건과류	과자	껌	마른안주	빵
아이스크림	음료	젤리	초콜릿	캔디
상품검색 <input type="text" value="상품명을 입력해주세요."/> <input type="button" value="검색"/>				
콘초 초코가나슈 1200원	콘치 치즈크림 1200원	콜드오렌지 1400원		
콜드포도 1400원	파삭칩 사워어니언 맛 1500원	파삭칩 트러플솔트 맛 1500원		
쿠션쿠션	쿠션쿠션	쿠크다스		

선택	상품명	수량	합계	삭제
<input checked="" type="checkbox"/>	콘초 초코가나슈	3 3,600원		<input type="button" value="삭제"/>
<input checked="" type="checkbox"/>	콘치 치즈크림	3 3,600원		<input type="button" value="삭제"/>
<input checked="" type="checkbox"/>	키즈톨	4 12,000원		<input type="button" value="삭제"/>

① ② ③ ④

상품갯수: 10개

합계금액: 19,200원

④ 할인금액: 18,240원

① 제품 선택하는 체크박스

② 제품 수량 선택 및 수량에 따른 가격 표출

③ 제품 삭제 버튼

④ 합계금액에서 할인율 계산한 금액



드래그 앤 드롭

- 드래그 기능

```
// 물품을 드래그 할 수 있게 만들어준다.
function drag() {
    $(".product").draggable({
        zIndex: 10000,
        helper: "clone"
    });
}
```

- 드롭 기능

```
// 드래그 후 드롭할 때 가져갈 데이터와 html코드를 추가해준다.
function drop() {
    $(".basketdiv").droppable({
        drop: function(e, ui) {
            var product_id_duplicate_check = $(ui.draggable).find(".product_id").html();
            if($(".product_id_duplicate_check").length == 0){
                $('.basketdiv_list').html($('.basketdiv_list').html()

                    + "<div>";
                    rowFind();
                    basket.reCalc(); // 패키지 리스트에 담았을 때 가격을 다시 계산
                    basket.updateUI(); // 패키지 리스트에 담을 때마다 화면 업데이트
                    addIndex(); // 패키지 리스트에 담을 때마다 index + 1
                } else {
                    alert("이미 담긴 상품입니다.");
                }
            });
        }
    });
}
```

<HTML 코드>

- 추가할 제품 리스트 HTML코드

```
<div id='row_data' class='row data'>
    <div class='subdiv'>
        <div class='check'>
            <input type='checkbox' name='buy' value='260' checked='checked' onclick='javascript:basket.checkItem();'>&ampnbsp
        </div>
        <div id='product_img' class='img'>
            <img src='$(ui.draggable).find('.product_img').attr('src')' width='60'>
        </div>
        <div class='pname'>
            <span class='product_id_duplicate_check'>$(ui.draggable).find('.product_name').html()</span>
        </div>
        <div class='subdiv'>
            <div id='row_basketprice' class='basketprice'>
                <input type='hidden' name='p_price' id='p_price1' class='p_price' value='$(ui.draggable).find('.product_price').text()'>
                <span onclick='javascript:basket.changePNum('+indexCounter+);'>
                    <span style='float:left; border:1px solid #ccc; padding:2px 5px; margin-right:5px;'>
                        <i class='fas fa-arrow-alt-circle-up up'></i>
                    </span>
                    <span style='float:left; border:1px solid #ccc; padding:2px 5px; margin-right:5px;'>
                        <i class='fas fa-arrow-alt-circle-down down'></i>
                    </span>
                </span>
                <div id='row_sum' class='sum'>$(ui.draggable).find('.product_price').text().formatNumber() 원</div>
            </div>
            <div class='subdiv'>
                <div class='basketcmd'>
                    <a href='void(0)' id='abutton' class='abutton' onclick='javascript:basket.delItem();'>삭제</a>
                </div>
                <div class='check_div' style='display:none'>

```

- HTML 제품을 드래그 앤 드롭 할 때마다 리스트를 구성하는 JavaScript 코드를 추가해주는 방식으로 리스트 업



제품 수량 선택 및 수량에 따른 가격 표출

```
changePNum: function (pos) {
    var item = document.querySelector('input[name=p_num'+pos+']');
    var p_num = parseInt(item.getAttribute('value'));
    var newval = event.target.classList.contains('up') ? p_num+1 : event.target.classList.contains('down') ? p_num-1 : event.target.value;
    if (parseInt(newval) < 1 || parseInt(newval) > 99) { return false; }
    item.setAttribute('value', newval);
    item.value = newval;

    var price = item.parentElement.parentElement.previousElementSibling.firstChild.getAttribute('value');
    item.parentElement.nextElementSibling.textContent = (newval * price).formatNumber()+'원';
    //AJAX 업데이트 전송

    //전송 처리 결과가 성공하면
    this.reCalc();
    this.updateUI();
}
```

```
//재계산
reCalc: function(){
    this.totalCount = 0;
    this.totalPrice = 0;
    document.querySelectorAll(".p_num").forEach(function (item) {
        if(item.parentElement.parentElement.parentElement.previousElementSibling.firstChild.firstChild.checked == true){
            var count = parseInt(item.getAttribute('value'));
            this.totalCount += count;
            var price = item.parentElement.parentElement.previousElementSibling.firstChild.getAttribute('value');
            this.totalPrice += count * price;
        }
    }, this);
}
```

```
//화면 업데이트
updateUI: function () {
    document.querySelector('#sum_p_num').textContent = '상품갯수: ' + this.totalCount.formatNumber() + '개';
    document.querySelector('#sum_p_price').textContent = '합계금액: ' + this.totalPrice.formatNumber() + '원';
    document.querySelector('#package_price').setAttribute('value',salePrice(this.totalPrice));

    // 합계금액이 0원일 때 할인금액란을 없앤다.
    if(this.totalPrice == 0) {
        document.querySelector('#sale_p_price').textContent = '';
    } else {
        document.querySelector('#sale_p_price').textContent = '할인금액: ' + salePrice(this.totalPrice).formatNumber() + '원';
    }

    // 0원 이상이면 합계 금액에 줄긋기
    textDeco();
}
```

- 검색 수량 변경 버튼 누를 때마다 개수만큼 상품 가격을 계산한 후 재계산 함수와 화면 업데이트 함수 호출
- 체크 돼 있는 상품들의 값과 수량을 곱한 값을 totalPrice에 누적시켜 합계 금액을 계산
- 체크 돼 있는 상품들의 전체 수량을 totalCount에 누적시켜 전체 상품 개수를 계산
- 계산된 값으로 화면을 업데이트 해주는 함수



상품 리스트 삭제 기능 및 나머지 기능(1)

선택	상품명	수량	합계	삭제
<input checked="" type="checkbox"/>	과식칩 트러플 슬트 맛	4	6,000원	<input type="button" value="삭제"/>
<input type="checkbox"/>				

(1) (2) (3) (4)

① 상품 체크 시 화면 업데이트 기능

```
checkItem: function () {
    this.reCalc();
    this.updateUI();
}
```

② 삭제 버튼

```
delItem: function () {
    event.target.parentElement.parentElement.parentElement.remove();
    this.reCalc();
    this.updateUI();
    rowFind();
}
```

③ 체크한 리스트 상품 삭제

```
//체크한 장바구니 상품 비우기
delCheckedItem: function(){
    document.querySelectorAll("input[name=buy]:checked").forEach(function (item) {
        item.parentElement.parentElement.parentElement.remove();
    });
    //AJAX 서버 업데이트 전송
    //전송 처리 결과가 성공이면
    this.reCalc();
    this.updateUI();
    rowFind();
}
```

④ 리스트 전체 상품 삭제

```
//장바구니 전체 비우기
delAllItem: function(){
    document.querySelectorAll('.row.data').forEach(function (item) {
        item.remove();
    });
    //AJAX 서버 업데이트 전송
    //전송 처리 결과가 성공이면
    this.totalCount = 0;
    this.totalPrice = 0;
    this.reCalc();
    this.updateUI();
    rowFind();
    resetIndex();
}
```



상품 리스트 삭제 기능 및 나머지 기능(2)

선택	상품명	수량	합계	삭제
<input checked="" type="checkbox"/>	과삭칩 트러플 슬트 맛	4	6,000원	<button>삭제</button>
<input checked="" type="checkbox"/>	과삭칩 사워어 니언 맛	3	4,500원	<button>삭제</button>
<input checked="" type="checkbox"/>	쿠크다스 케이 크	5	12,500원	<button>삭제</button>
<input checked="" type="checkbox"/>	키즈툴	1	3,000원	<button>삭제</button>
<input checked="" type="checkbox"/>	트레비금귤	1	1,800원	<button>삭제</button>

선택상품삭제

장바구니비우기

상품갯수: 14개

① 합계금액: 27,800원

② 할인금액: 26,410원

① 합계금액에 줄 긋기

```
function textDeco() {
    // 합계금액이 0보다 커질 때 줄긋기
    if($("#row_data").length > 0) {
        $("#sum_p_price").css('text-decoration','line-through');
    } else {
        $("#sum_p_price").css('text-decoration','none');
    }
}
```

- 합계금액이 0보다 커질 때 세일 표시 가로선 긋기

② 합계금액 할인 계산

```
// 할인가 계산
function salePrice(totalPrice) {

    var result = 0;
    if($('#auth').val() == "ADMIN" && $('#auth_url').val() == "admin") {
        result = totalPrice * 0.85;
    } else {
        if(totalPrice >= 30000 && totalPrice < 50000) {
            result = totalPrice * 0.90;
        } else if(totalPrice >= 50000) {
            result = totalPrice * 0.85;
        } else {
            result = totalPrice * 0.95;
        }
    }
    return result;
}
```

- 가격대마다 할인이 다르게 적용



선택 패키지 결제 후 DB저장(1)

상품검색 상품명을 입력해주세요.

포테토칩 샤워크림어니언 1000원	포테토칩 에그토스트맛 1000원	포테토칩 엣지 통감자구이맛 900원
포테토칩 오리지널 1000원	프레첼 솔티카라멜맛 1220원	프로틴바 1200원
헬프리 헬프리	헬프리 헬프리	헬프리 헬프리

선택 상품명 수량 합계 삭제

- 콘초 초코가니슈 1 1,200원
- 쿨드포도 1 1,400원
- 과식친 사워 어니언 맛 1 1,500원
- 쿠크다스 케이크 1 2,500원
- 클래식카스테라 4 4,000원
- 키커 시리얼 현미 바 1 1,000원
- 투유 1 2,000원

상품갯수: 10개
합계금액: 13,600원
할인금액: 12,920원

Click!

선택한 상품 주문

주문/결제

주문자 정보

남후승 (hsnam1211@naver.com) ▾

배송지 정보

- 배송지 선택 주문자 정보와 동일 새 배송지 입력
- 주문자 남후승
- 이메일 gudok1211@naver.com
- 휴대전화 010212345678
- 주소 08532 서울 금천구 가산동

주문상품

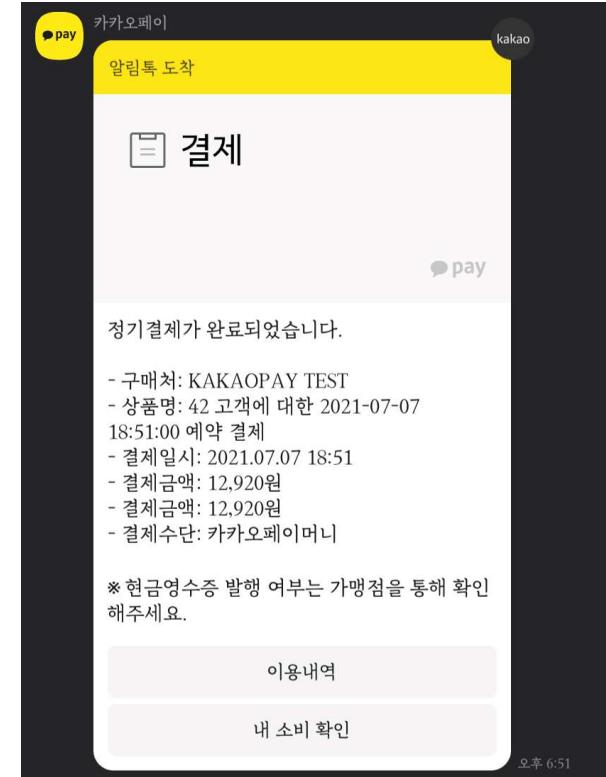
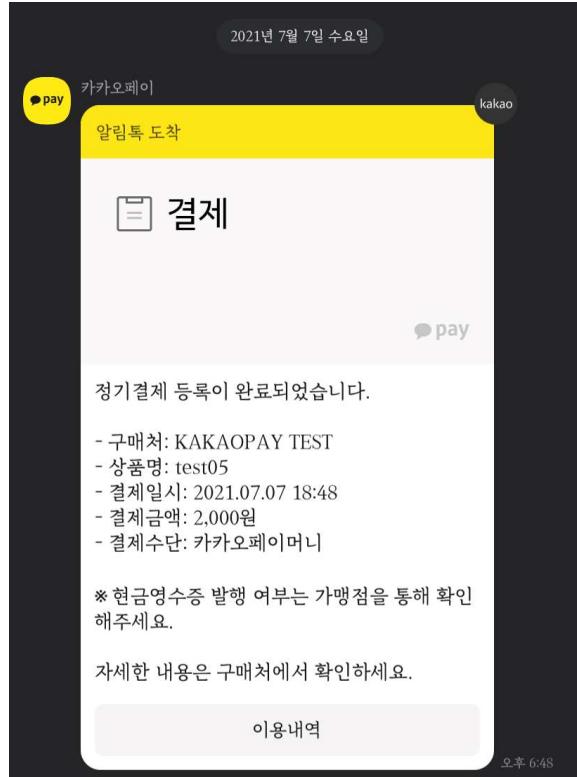
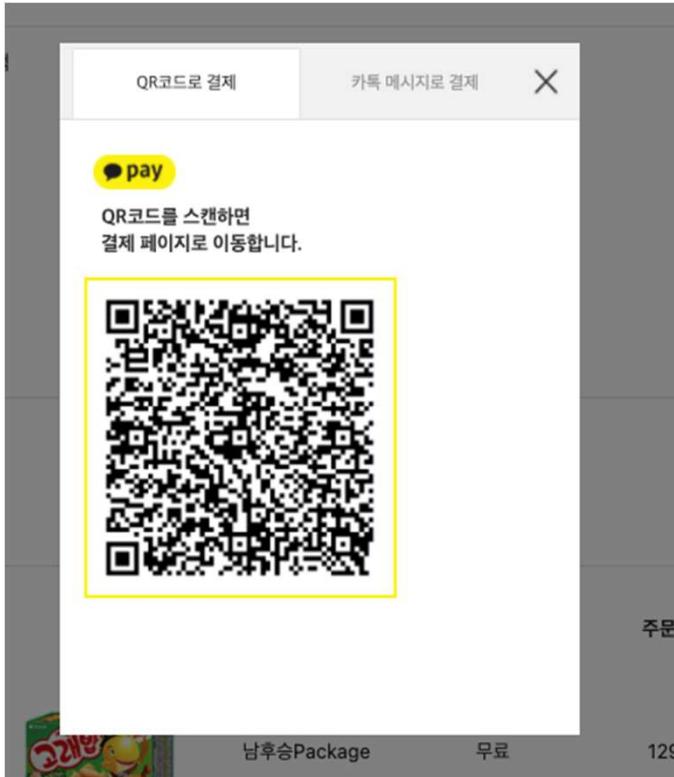
상품정보	배송비	주문금액
남후승 Package	무료	12920

정기결제

- 제품을 선택한 후 상품 주문 버튼 클릭
- 주문/결제 창에서 주문자 정보 확인 후 정기구독 클릭



선택 패키지 결제 후 DB저장(2)



- 정기 결제 성공 후 사용자에게 카카오톡 메시지가 전송 됨
- 설정한 스케줄러에 의해 예약 시간이 되면 자동 결제가 이루어짐



선택 패키지 결제 후 DB저장(3)

- 선택한 상품주문 버튼을 클릭 시 작동하는 js코드로 /payment로 submit() 요청을 보냄

```
// 고객 패키지 커스터마이징
if($("#package_price").val() > 0) {
    alert("submit");
    $('#orderform').submit();
} else {
    alert("상품을 선택해주세요.");
}
```

- /payment로 요청을 받은 PaymentController의 payment함수에서 다시 PackageService의 insertPackage함수를 호출

```
@PostMapping("/payment")
public String payment (Model model, @ModelAttribute
ProductListVO productList) {

    PackageVO newPackageVO = packService.insertPackage(productList);
    return "redirect:/payment?pno="+newPackageVO.getPackageId();
}
```

- 커스텀 패키지에선 패키지 이미지 값이 없기 때문에 이미지 클래스에 접근하지 못하게 하며, 매개변수로 받은 제품 값을 패키지 관련 테이블에 저장

```
public PackageVO insertPackage(ProductListVO productList) {

    String pack_img = "";
    // 관리자 모드의 패키지 추가가 아닌 커스텀 패키지 주문창에선 패키지 이미지가 없기 때문에
    // 이미지 클래스에 접근하지 못하게 함
    if(productList.getPackageType() == 0) {
        // 사진저장 클래스
        insertImg.insertImg(productList);
        pack_img = productList.getFileList()[0].getOriginalFilename();
    }

    // 따로 클래스로 만들자
    PackageVO packageVO = PackageVO.builder().
        packageName(productList.getPackageName()). // 이름도 login한 사람의 이름+"package"로 사용
        price(productList.getPackagePrice()).
        packageType(productList.getPackageType()).
        img(pack_img).
        build();

    PackageVO newPackage = packageRepo.save(packageVO);

    IntStream.range(0, productList.getProductId().length).forEach(idx->{
        Product p = productRepo.findById(productList.getProductId()[idx]).get();
        PackageVO packageVO1 = packageRepo.findById(packageVO.getPackageId()).get();

        Product_Package productPackage = Product_Package.builder().
            product_qty(productList.getProductQty()[idx]).
            product(p).
            pack(packageVO1).
            build();

        productPackageRepo.save(productPackage);
    });
    return newPackage;
}
```



선택 패키지 결제 후 DB저장(4)

- TP_PACKAGE 테이블

PACKAGE_ID	PACKAGE_NAME	PACKAGE_TYPE	PRICE	IMG
223	test1	0	1200	스케치.png
224	테스트 패키지	0	8400	test1.jpg
225	남후승Package	1	1140	(null)
226	남후승Package	1	12920	(null)
214	test1	0	2400	(null)
215	test2	0	11200	(null)
220	1211	0	123	(null)
221	test4	0	18000	스케치.png
222	test123	0	2900	스케치.png
218	(null)	0	0	(null)
219	test3	0	5000	(null)

- TP_PRODUCTPACK 테이블

PP_ID	PRODUCT_QTY	PACKAGE_ID	PRODUCT_ID
3031	1	226	2405
3033	1	226	2409
3034	1	226	2414
3035	4	226	2421
3036	1	226	2423
3037	1	226	2430

- TP_SUBSCRIBE 테이블

SUBSCRIBE_ID	PACKAGE_ID	PAYMENT_DATE	SUB_CHECK	CUSTOMER_ID
1	1	24 21/06/24	1	25
2	2	24 21/06/24	1	1
3	3	226 21/07/07	1	42

- 결제한 패키지가 TP_PACKAGE 테이블과 중간 테이블인 TP_PRODUCTPACK 테이블에 패키지 속 제품 데이터 저장
- 결제 시 TP_SUBSCRIBE 테이블에 구독 정보 저장

프로젝트 소개

남후승

- ✓ 패키지 커스터마이징
- ✓ 패키지 관리자 기능
- ✓ 배송관리





제품/패키지 관리 (관리자 기능)

건과류	과자	껌	마른안주	빵
아이스크림	음료	젤리	초콜릿	캔디

상품검색 [검색]

	선택	상품명	수량	합계	삭제
		콘초 초코기나스		1200원	
		콘치 치즈크림		1200원	
		콜드오렌지		1400원	

[선택상품삭제] [장바구니바우기]

상품개수: 0개
합계금액: 0원

선택한 상품 주문

패키지 추가

건과류	과자	껌	마른안주	빵
아이스크림	음료	젤리	초콜릿	캔디

상품검색 [검색]

	선택	상품명	수량	합계	삭제
		콘초 초코기나스		1200원	
		콘치 치즈크림		1200원	
		콜드오렌지		1400원	

[선택상품삭제] [장바구니바우기]

상품개수: 0개
합계금액: 0원

패키지 가격 :
패키지 이름 :
패키지 이미지 : [파일선택] 선택된 파일 없음

패키지 생성

+ 목록추가

제품명	카테고리	회사	가격	이미지 파일첨부
ex) 세우깡	건과류	lkas	ex) 1000	<input type="button" value="파일선택"/> 선택된 파일 없음

제품 넣기

- 같은 customPackage4.html을 호출해서 권한에 따라서 화면 표출이 다르게 설정



제품/패키지 관리자 기능

패키지 추가 ①

제품 추가 ②

패키지 추가

건과류 과자 껌 마른안주 빵

아이스크림 음료 젤리 초콜릿 캔디

상품검색



콘초 초코가나슈
1200원



콘치 치즈크림
1200원



폴드오렌지
1400원



콜드포도
1400원



과식칩 사워에니언
맛
1500원



과식칩 트리플솔트
맛
1500원



쿠션쿠션



쿠션쿠션



쿠키다스

상품을 드래그해서
담아주세요!

```
@GetMapping("/product/adminMakeProductAndPackage")
public String adminMakeProductAndPackage(Model model) {
    model.addAttribute("admin_check", "admin");
    model.addAttribute("categorylist", productService.selectCategoryAll());
    model.addAttribute("productlist", productService.selectProductAll());
    model.addAttribute("companylist", productService.selectCompanyAll());

    return "/product/customPackaging4";
}
```

① 패키지 추가하는 부분으로 이동

② 제품 추가하는 부분으로 이동

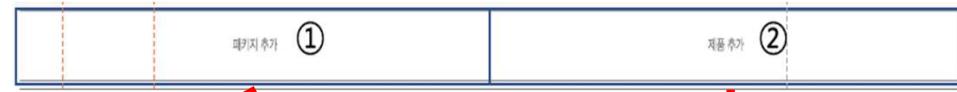
③ 제품 삭제 버튼



제품/패키지 관리자 기능 스크롤 이동

```
<div id="move_btn">
  <button class="move_btn" onclick="fnMove('add_package_h1')>패키지 추가</button>
  <button class="move_btn" onclick="fnMove('add_product_h1')>제품 추가</button>
  <!-- <button class="move_btn" onclick="fnMove('')>패키지 업데이트</button> -->
</div>
```

```
// 어드민 패키지/제품 페이지에서 클릭 시 각 기능의 위치로 이동시키는 기능
function fnMove(seq){
  var offset = $( "." + seq ).offset();
  $('html, body').animate({scrollTop : offset.top-215}, 1000);
}
```



패키지 추가

견과류	과자	껌	마른안주	빵
아이스크림	음료	젤리	초콜릿	캔디

상품검색 상품명을 입력해주세요.

선택	상품명	수량	합계	삭제
	콘초 초코가나슈	1200원		
	콘치 치즈크림	1200원		
	콜드오렌지	1400원		

콘초 초코가나슈
1200원

콘치 치즈크림
1200원

콜드오렌지
1400원

상품을 드래그해서 담아주세요!

제품 생성

제품명	카테고리	회사	가격	이미지 파일첨부
ex) 새우깡	견과류	ikas	ex) 1000	<input type="button" value="파일 선택"/> 선택된 파일 없음
	견과류	ikas	<input type="button" value="파일 선택"/> 선택된 파일 없음	<input type="button" value="삭제"/>
	견과류	ikas	<input type="button" value="파일 선택"/> 선택된 파일 없음	<input type="button" value="삭제"/>
	견과류	ikas	<input type="button" value="파일 선택"/> 선택된 파일 없음	<input type="button" value="삭제"/>

- 패키지 추가/제품 추가 버튼 클릭 시 설정한 위치까지 스크롤 이동



제품/패키지 추가

상품을 드래그해서 담아주세요!

선택상품삭제 정바구니비우기

상품개수: 0개
합계금액: 0원

①

패키지 가격 : 패키지 가격을 입력해주세요.

패키지 이름 : 패키지 이름을 입력해주세요.

패키지 이미지 : 파일 선택 선택된 파일 없음

패키지 생성

과식칩 트러플솔트 맛 1500원

쿠크다스 비엔나커피

제품 생성 ③

+ 목록추가

제품명	카테고리	회사	가격	이미지 파일첨부
ex) 새우깡	건과류	lkas	ex) 1000	파일 선택 선택된 파일 없음
	건과류	lkas		파일 선택 선택된 파일 없음
	건과류	lkas		파일 선택 선택된 파일 없음
	건과류	lkas		파일 선택 선택된 파일 없음

②

③ 제품 넣기

① 리스트에 제품을 담은 뒤 패키지 가격, 이름, 이미지 입력하는 Input 부분

② 제품 추가하는 부분

③ 여러 제품 추가 시 목록 추가 할 수 있는 버튼



제품 추가(1)

제품 생성

제품명	카테고리	회사	가격	이미지 파일첨부
test1	과자	농심	2000	<input type="button" value="파일 선택"/> test1.jpg
test2	껌류	그린낫츠	1500	<input type="button" value="파일 선택"/> test2.jpg <input type="button" value="삭제"/>

제품 넣기

↓

패키지 추가

상품검색	검색
아이스크림	과자
음료	껌
초콜릿	마른안주
캔디	빵

TEST TEST

선택	상품명	수량	합계	삭제
<input checked="" type="checkbox"/>	test2	<input type="button" value="1"/> <input type="button" value="증가"/> <input type="button" value="감소"/>	1,500원	<input type="button" value="삭제"/>
<input checked="" type="checkbox"/>	test1	<input type="button" value="1"/> <input type="button" value="증가"/> <input type="button" value="감소"/>	2,000원	<input type="button" value="삭제"/>

test2 1500원
test1 2000원

- 제품 생성 시 test 제품 추가된 화면

- 제품명/카테고리/회사/가격/이미지 파일 첨부 후 제품 넣기 버튼 클릭

```
// 제품 insert
$('#product_add').click(function(e) {
  e.preventDefault();
  if($('.add_productName').val()=='') {
    if($('.add_productPrice').val()=='') {
      if(window.confirm("제품을 생성하시겠습니까?")) {
        alert("submit");
        $('#add_product_form').submit();
      } else {
        alert("취소하였습니다.");
      }
    } else {
      alert("제품 가격을 입력해주세요.");
    }
  } else {
    alert("제품명을 입력해주세요.");
  }
});
```

- submit()에서 /product/adminProductInsert Controller 호출

```
<form id="add_product_form" method="post" action="/product/adminProductInsert"
enctype="multipart/form-data">
```

- Controller에서 Service 호출

```
@PostMapping("/product/adminProductInsert")
public String adminProductInsert(Model model, @ModelAttribute ProductListVO productList) throws Exception {
  productService.adminInsertProduct(productList);
  return "redirect:/product/adminMakeProductAndPackage";
}
```



제품 추가(2)

제품 생성

제품명	카테고리	회사	가격	이미지 파일첨부
test1	과자	농심	2000	파일 선택 test1.jpg
test2	껌	그린的表情	1500	파일 선택 test2.jpg

+ 목록추가

제품 넣기



패키지 추가

상품검색	상품명	수량	합계	삭제
test	test2	1	1,500원	삭제
	test1	1	2,000원	삭제

TEST TEST

test2 1500원
test1 2000원

- 제품 생성 시 test 제품 추가된 화면

- 호출된 서비스의 adminInsertProduct 함수에서 처음 입력한 제품 정보들을 입력한 후 들어온 정보 수만큼 DB에 저장

```
public void adminInsertProduct(ProductListVO productList) {
    // ...
    insertImg.insertImg(productList);
    IntStream.range(0, productList.getProductName().length).forEach(idx->{
        Product product = Product.builder().
            productName(productList.getProductName()[idx]).
            price(productList.getProductPrice()[idx]).
            categoryId(productList.getCategoryId()[idx]).
            companyId(productList.getCompanyId()[idx]).
            build();
        productRepo.save(product);
        System.out.println("insert 성공" + idx);
    });
}
```

- InsertImg 클래스의 insertImg 메서드에서 들어온 제품 이미지 수만큼 입력된 경로에 저장

```
@Component
public class InsertImg {
    public void insertImg(ProductListVO productList) {
        // 패키지 추가 시 productName이 null일 때 길이를 1로 맞춰준다.
        if(productList.getProductName() == null) {
            String[] length = {"0"};
            productList.setProductName(length);
        }

        IntStream.range(0, productList.getProductName().length).forEach(idx->{
            String rootPath = FileSystemView.getFileSystemView().getHomeDirectory().toString();
            String basePath = rootPath;
            String filePath = basePath + "/git/final_project_subscribe/final_project_subscribe/src/main"
                + "/resources/static/images/product_images/"
                + productList.getFiles()[idx].getOriginalFilename();

            File dest = new File(filePath);
            try {
                productList.getFiles()[idx].transferTo(dest);
            } catch (IllegalStateException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        });
    }
}
```



패키지 추가(1)

The screenshot shows a shopping cart interface with two items:

- 쿠쉬쿠쉬 크로 와상 (1 item, 1,500원)
- test2 (4 items, 6,000원)

Below the cart, there are buttons for "선택상품삭제" and "장바구니비우기".

상품갯수: 16개
합계금액: 26,400원
할인금액: 22,440원

Package creation fields:

- 패키지 가격: 22440
- 패키지 이름: 테스트 패키지
- 패키지 이미지: 파일 선택 test1.jpg

A large button at the bottom right says "패키지 생성".

- 패키지 가격/이름/이미지 입력 후 패키지 생성

```
// 클릭 시 패키지 정보를 form으로 전송 후 db에 저장하는 버튼
function packageAdd() {
    if($('#auth').val() == "ADMIN" && $('#auth_url').val() == "admin") {
        // admin 패키지 추가
        if($("#package_price").val() > 0) {
            if($("#make_package_price").val() != '') {
                if($("#make_package_name").val() != '') {
                    if(window.confirm("패키지를 생성하시겠습니까?")) {
                        alert("submit");
                        $('#orderform').submit();
                    } else {
                        alert("취소하였습니다.");
                    }
                } else {
                    alert("패키지명을 입력해주세요.");
                }
            } else {
                alert("패키지 가격을 입력해주세요.");
            }
        } else {
            alert("상품을 선택해주세요.");
        }
    } else {
        // 고객 패키지 커스터마이징
        if($("#package_price").val() > 0) {
            alert("submit");
            $('#orderform').submit();
        } else {
            alert("상품을 선택해주세요.");
        }
    }
}
```

- 패키지 가격/이름을 반드시 입력해야 패키지 생성 가능



패키지 추가(2)

- Controller에서 Service 호출 (권한이 admin이면 action의 경로가

/product/customInsert로 변경

```
<form name="orderform" id="orderform" method="post" class="orderform"
      action="/payment" enctype="multipart/form-data">
```

- submit()에서 /product/customInsert Controller 호출

```
@PostMapping("/product/customInsert")
public String insertPackage(Model model, @ModelAttribute ProductListVO productList) {
    productService.insertPackage(productList);

    return "redirect:/product/adminMakeProductAndPackage";
}
```

- 호출된 서비스의 insertPackage 함수에서 처음 입력한 제품 정보들을 입력한 후 패키지 정보를 DB에 저장

```
public PackageVO insertPackage(ProductListVO productList) {
    String pack_img = "";
    // 관리자 모드의 패키지 추가가 아닌 커스텀 패키지 주문창에선 패키지 이미지가 없기 때문에
    // 이미지 클래스에 접근하지 못하게 함
    if(productList.getPackageType() == 0) {
        // 사진저장 클래스
        insertImg.insertImg(productList);
        pack_img = productList.getFileNames()[0].getOriginalFilename();
    }

    // 바로 클래스로 만들자
    PackageVO packageVO = PackageVO.builder().
        packageName(productList.getPackageName()). // 이름도 login한 사람의 이름+"package"로 사용
        price(productList.getPackagePrice()).
        packageType(productList.getPackageType()).
        img(pack_img).
        build();

    PackageVO newPackage = packageRepo.save(packageVO);

    IntStream.range(0, productList.getProductId().length).forEach(idx->{
        Product p = productRepo.findById(productList.getProductId()[idx]).get();
        PackageVO packageV01 = packageRepo.findById(packageVO.getPackageId()).get();

        Product_Package productPackage = Product_Package.builder().
            product_qty(productList.getProductQty()[idx]).
            product(p).
            pack(packageV01).
            build();

        productPackageRepo.save(productPackage);
    });
    return newPackage;
}
```

- TP_PACKAGE 테이블

PACKAGE_ID	PACKAGE_NAME	PACKAGE_TYPE	PRICE	IMG
224	테스트 패키지	0	22440	test1.jpg

- TP_PRODUCTPACK 테이블

PP_ID	PRODUCT_QTY	PACKAGE_ID	PRODUCT_ID
3024	3	224	2406
3025	4	224	2405
3026	1	224	2409
3027	3	224	2413
3028	1	224	2412
3029	4	224	3022

- 리스트에 추가한 제품들이 패키지 테이블에 들어가 있다.



제품 삭제 기능(1)

- 추가 테스트 시에 추가한 test2 제품을 삭제

A screenshot of a web application interface. At the top, there is a search bar with the text "test". Below the search bar, there are two product cards. The first card on the left has the text "TEST" and "test2 1500원". The second card on the right has the text "TEST" and "test1 2000원". A blue arrow points from the bottom of this interface to the next one.

- 삭제 확인 버튼 클릭

A screenshot of a "Package Add" dialog box. At the top, there is a confirmation dialog with the text "localhost:8888 내용: 제품을 삭제하시겠습니까?" and two buttons: "취소" and "확인". A red arrow points from the "확인" button to the next step. Below the dialog, the main dialog shows a grid of categories (간식류, 과자, 껌, 마른안주, 빵) and a sub-grid for "이스크림" (음료, 젤리, 초콜릿, 캔디). There is also a search bar with "test" and a list of products: test2 (1500원) and test1 (2000원). A blue arrow points from the bottom of this dialog to the final result screen.

- 검색 시 test2가 삭제된 것을 확인

A screenshot of a product search interface. The search bar at the top contains the text "test". Below the search bar, there is a single product card for "test1" with the price "2000원". A blue arrow points from the bottom of this interface to the final step.

- 연관 테이블 데이터 변경

PP_ID	PRODUCT_QTY	PACKAGE_ID	PRODUCT_ID
3024	3	224	2406
3025	4	224	2405
3026	1	224	2409
3027	3	224	2413
3028	1	224	2412

PACKAGE_ID	PACKAGE_NAME	PACKAGE_TYPE	PRICE	IMG
224	테스트 패키지	0	8400	test1.jpg

- TP_PRODUCPACK 테이블에서 test2(id=3022)가 삭제
- TP_PACKAGE에서 price가 감소



제품 삭제 기능

- 삭제 버튼 클릭

```
<button onclick="adminProductDelete(this)" class="product_delete_btn">X</button>
```

- 삭제 확인 클릭 시 /product/adminProductDelete로 ajax요청 보내기

```
// 관리자 페이지 제품 삭제
function adminProductDelete(productId) {
    if(window.confirm("제품을 삭제하시겠습니까?")) {
        var product_id = $(productId).parent().children(".product_id").text();
        $.ajax({
            url: "/product/adminProductDelete",
            data: {
                productId: product_id
            },
            success: function() {
                location.reload();
            }
        });
    }
}
```

- Controller에서 service 호출

```
@GetMapping("/product/adminProductDelete")
public String adminDeleteProduct(Model model, @ModelAttribute ProductListVO productList) throws Exception {
    productService.adminDeleteProduct(productList);
    return "redirect:/product/adminMakeProductAndPackage";
}
```

- 호출된 서비스의 adminDeleteProduct함수에서 제품과 관련 패키지 테이블에서 데이터 삭제 진행

```
public void adminDeleteProduct(ProductListVO productList) {
    // 삭제할 제품의 product_id
    Product product = productRepo.findById(productList.getProductid()[0]).get();

    // 프로덕트_패키지 테이블에서 package_id를 조회한 row를 조회한 후 저장
    List<Product_Package> product_pack_list = productPackageRepo.findByProduct(product);

    // 프로덕트_패키지 테이블에서 package_id를 조회한 후 저장. 여러개면 set으로 저장
    HashSet<Long> package_id_list = new HashSet<>();

    for(Product_Package product_pack:product_pack_list) {
        // set에 package_id저장
        package_id_list.add(product_pack.getPack().getPackageId());

        // 프로덕트_패키지 테이블에서 삭제 선택한 제품 로우 삭제
        productPackageRepo.deleteById(product_pack.getPpId());
    }

    // 프로덕트 테이블에서 선택한 프로덕트를 삭제
    productRepo.delete(product);

    // 프로덕트_패키지 테이블에서 저장한 package_id를 조회한 후 패키지 타입과 price를 조인
    Iterator<Long> it = package_id_list.iterator();

    while(it.hasNext()){

        // package_id를 한 곳에 저장
        long package_id = it.next();

        for(Integer product_price:productPackageRepo.findByPackageId(package_id)) {
            price_result += product_price;
        }
        packageRepo.findById(package_id).ifPresent(b->{
            // 패키지 타입별로 할인 가격을 적용시켜 계산한다.

            // 계산된 가격을 패키지에 저장
            b.setPrice(price_result);

            // 새로운 패키지 가격이 저장된 패키지를 테이블에 업데이트한다.
            packageRepo.save(b);
        });

        // 변수를 클래스 단에서 선언했기 때문에 다음 update를 위해 reset해준다.
        price_result = 0;
    }
}
```

프로젝트 소개

남후승

- ✓ 패키지 커스터마이징
- ✓ 패키지 관리자 기능
- ✓ 배송관리





배송 리스트 MODEL Delivery

```

@Getter
@Setter
@ToString(exclude = {"pack", "customer"})
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Builder
@Table(name="tp_delivery")

//배송
public class Delivery {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "delivery_id")
    long deliveryId;

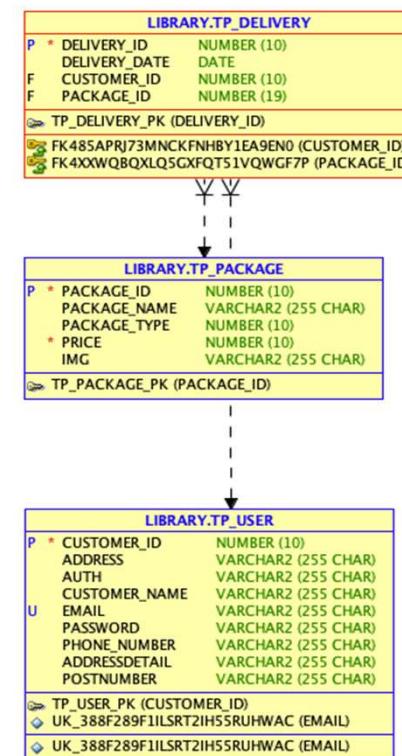
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "package_id")
    PackageVO pack;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "customer_id")
    Member customer;

    @Column(name = "delivery_date")
    Date deliveryDate;
}

```

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID
1 DELIVERY_ID	NUMBER(10,0)	No	(null)	1
2 DELIVERY_DATE	DATE	Yes	(null)	2
3 CUSTOMER_ID	NUMBER(10,0)	Yes	(null)	3
4 PACKAGE_ID	NUMBER(19,0)	Yes	(null)	4





배송 리스트 조회 (관리자 기능)

- 달력의 일자 클릭 시 그 일자에 있을 배송 건이 리스트로 표출

2021년 7월

④

일	월	화	수	목	금	토
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

① ③ ②

<	Today	>
---	-------	---

- ① 이전 달로 달력 변경
- ② 다음 달로 달력 변경
- ③ 오늘 날짜로 이동
- ④ 날짜 클릭 시 해당 날짜의 배송 건 표출



- 2021-06-27 배송 건

패키지명	주소	주문자	배송날짜
남후승 package	08532 서울 금천구 가산동 149-63 서울 금천 구 가산동 149-63 (가산동)	남후승	2021-06-27
남후승 package	08532 서울 금천구 가산동 149-63 서울 금천 구 가산동 149-63 (가산동)	남후승	2021-06-27
남후승 package	08532 서울 금천구 가산동 149-63 서울 금천 구 가산동 149-63 (가산동)	남후승	2021-06-27
남후승 package	08532 서울 금천구 가산동 149-63 서울 금천 구 가산동 149-63 (가산동)	남후승	2021-06-27
남후승 package	08532 서울 금천구 가산동 149-63 서울 금천 구 가산동 149-63 (가산동)	남후승	2021-06-27
1 2 3 4 5 6 7 8 9 10 NEXT			

- 2021-05-20 배송 건

패키지명	주소	주문자	배송날짜
남후승 package	08532 서울 금천구 가산동 149-63 서울 금천 구 가산동 149-63 (가산동)	남후승	2021-05-20



배송 리스트 조회 (관리자 기능)

- 배송 리스트 페이지 표출

```
@GetMapping("/delivery/deliveryList")
public void selectList(Model model, PageVO pagevo) {
    pagevo = PageVO.builder().page(1).size(5).date(date.getDate()).build();
    Page<Delivery> result = deliService.selectDeliveryList1(pagevo);
    model.addAttribute("deliverylist", result);
    model.addAttribute("pagevo", pagevo);
    model.addAttribute("result", new PageMake(result));
}
```

- 달력 표출

```
@GetMapping("/delivery/deliveryCalender")
public String getCalender() {
    return "/delivery/deliveryCalender";
}
```

- Date를 sql형식에 맞게 바꾸는 함수

```
function makeDate(day) { // date를 sql형식에 맞게 바꾼뒤 리턴 ex) yyyy-MM-dd
    var thisDate = $('.year-month').text();
    var yearAndMonth = thisDate.replace(/[^0-9]/g, '');
    var splitYear = yearAndMonth.substring(0,4);
    var splitMonth = yearAndMonth.substring(4,);
    var setDate = splitYear + '-' + splitMonth + '-' + day;
    return setDate;
}
```

- 달력의 일자 클릭 시 그 일자에 있을 배송 건이 리스트로 표출

```
function sendDate(date) {
    var selectDate = $("#select_date");
    // 페이지 넘버 클릭 시 사용될 ajax에 넘길 날짜 생성
    selectDate.val(makeDate($(date).text()));

    // 달력의 일자 클릭 시 그날에 맞는 배송리스트 조회
    $.ajax({
        type:"GET",
        url:"/delivery/deliveryListSearch",
        data:{
            date:makeDate($(date).text())
        },
        success:function(result) {
            $("#hoos1").html(result);
        }
    });
}
```

- 페이지 된 배송리스트 리턴

```
public Page<Delivery> selectDeliveryList1(PageVO pvo) {
    Predicate p = deliRepo.makePredicate(pvo.getDate());
    Pageable pageable = pvo.makePaging(0, "deliveryId");
    Page<Delivery> result = deliRepo.findAll(p, pageable);
    return result;
}
```



배송 리스트 조회 (관리자 기능)

- 패키지 결제 또는 다음 결제 예약 시 PaymentController의 /delivery/deliveryInsert 함수를 호출

```
@GetMapping("/delivery/deliveryInsert")
public @ResponseBody void deliveryInsert(@RequestParam Map<String, Object> map)
    throws JsonMappingException, JsonProcessingException{
    long customerId = Long.parseLong((String) map.get("customerId"));
    long packageId = Long.parseLong((String) map.get("packageId"));
    Date deliveryDate = Date.valueOf((String) map.get("deliveryDate"));

    deliService.deliveryInsert(packageId, customerId, deliveryDate);
}
```

- PaymentController에서 DeliveryService의 deliveryInsert 함수를 호출

```
public void deliveryInsert(long packageId, long customerId, Date deliveryDate) {
    PackageVO package_id = packRepo.findById(packageId).get();
    Member customer_id = memberRepo.findById(customerId).get();

    Delivery delivery = Delivery.builder().
        pack(package_id).
        customer(customer_id).
        deliveryDate(deliveryDate).
        build();

    deliRepo.save(delivery);
}
```

- 배송 건마다 데이터 베이스에 저장되며, **customer_id**로 유저의 주소지 및 이름까지 조인
- 결제 및 예약 시 다음 배송 건이 테이블에 저장
- 결제 직후에는 하루 뒤로 **delivery_date**값이 저장
- 예약 건에 한해서 한 달 뒤의 스케줄로 **delivery_date**값이 저장



	DELIVERY_ID	DELIVERY_DATE	CUSTOMER_ID	PACKAGE_ID
1	41	21/06/24	24	24
2	117	21/06/27	24	24
3	118	21/06/27	24	24
4	119	21/06/27	24	24
5	120	21/06/27	24	24
6	121	21/06/27	24	24
7	122	21/06/27	24	24
8	123	21/06/27	24	24
9	124	21/06/27	24	24

프로젝트 소개

임세혁

✓ 구독목록 조회





TestController.java(구독 조회/기능)

```
@GetMapping("/subscribe")
public String viewed(Model model) {
    //구독 조회
    Object principal=SecurityContextHolder.getContext().getAuthentication().getPrincipal(); //로그인시 정보 받아오기
    UserSecurity ss=(UserSecurity)principal;
    Member mem=service.getMemberInfo(ss.getUsername()); //정보 찾기

    List<Subscribe> lel=subService.findByCustomer(mem);
    System.out.println(lel);
    model.addAttribute("subscribe",lel);

    return "subscribe";
}
```

구독화면에 뜨는 구독 된 패키지 리스트를 출력하는 기능



subscribe.html

```
92      </div>
93  </div>
94  <!-- Header End -->
95 </header>
96<main>
97  <!-- Hero Area Start-->
98<div class="slider-area ">
99<div class="single-slider slider-height2 d-flex align-items-center">
100<div class="container">
101<div class="row">
102<div class="col-xl-12">
103<div class="hero-cap text-center">
104  <h2>선택 회원님 정보</h2>
105<table>
106<tr th:each="sub:${subscribe}">
107  <td th:text="${sub.subscribeId}"></td>
108  <td th:text="${sub.pack.packageName}"></td>
109  <td th:text="${sub.paymentDate}"></td>
110
111
112</tr>
113</table>
114</div>
115</div>
116</div>
117</div>
118</div>
119</div>
120<!-- Hero Area End-->
121<!--=====Single Product Area =====-->
122<div class="product_image_area">
123<div class="container">
124<div class="row justify-content-center">
125<div class="col-lg-12">
126<div class="product_img_slide owl-carousel">
127<div class="single_product_img">
128  
130</div>
131<div class="single_product_img">
132  
133</div>
134<div class="single_product_img">
```

사용자가 구독한 리스트 목록을 보여주는 화면
(회원정보수정창에서 들어감)



subscribe.html

```
143          th:text="${pack.packageName}"></a> -->
144      </h3>
145      <p></p>
146      <h2>구독 패키지 정보</h2>
147      <table class="table mb-0">
148          <thead>
149              <tr>
150                  <th class="border-gray-200" scope="col">구독번호</th>
151                  <th class="border-gray-200" scope="col">패키지명</th>
152                  <th class="border-gray-200" scope="col">결제일</th>
153                  <th class="border-gray-200" scope="col">고객님 아이디</th>
154                  <th></th>
155                  <th></th>
156
157          </tr>
158      </thead>
159      <tbody>
160          <tr th:each="sub:${subscribe}">
161              <td th:text="${sub.subscribeId}"></td>
162              <td th:text="${sub.pack.packageName}"></td>
163              <td th:text="${sub.paymentDate}"></td>
164              <td th:text="${sub.customer.email}" />
165              <td><button class="btn btn-sm btn-primary" type="button" th:onClick="call([$sub.pack.packageId])">리뷰달기</button>
166              <td><button class="btn btn-sm btn-primary" type="button" onclick="deleteRow(this);">구독취소</button></td>
167          </tr>
168      </tbody>
169  </table>
170
171
172      <!-- <div class="add_to_cart">
173          <a class="btn_3" th:href="@{/payment(pno=${pack.packageId})}"></a>
174      </div> -->
175
176      </div>
177      </div>
178  </div>
179 </div>
180 </div>
181 <!--=====End Single Product Area =====-->
182 <!-- subscribe part here -->
183 <section class="subscribe_part section_padding">
184     <div class="container">
185         <div class="row justify-content-center">
```

사용자가 구독한 리스트 목록을 보여주는 화면이다.



구독 패키지 정보

구독번호	패키지명	결제일	고객님 아이디
------	------	-----	---------

1	홈캠핑	2021-07-21	lsh2757@naver.com
---	-----	------------	-------------------

[리뷰달기](#)[구독취소](#)

subscribe.html 실행화면

프로젝트 소개

김성휘

- ✓ 유저토론 게시판(과자갤러리)
- ✓ 문의게시판(QnA)
- ✓ 리뷰 및 별점(평점)



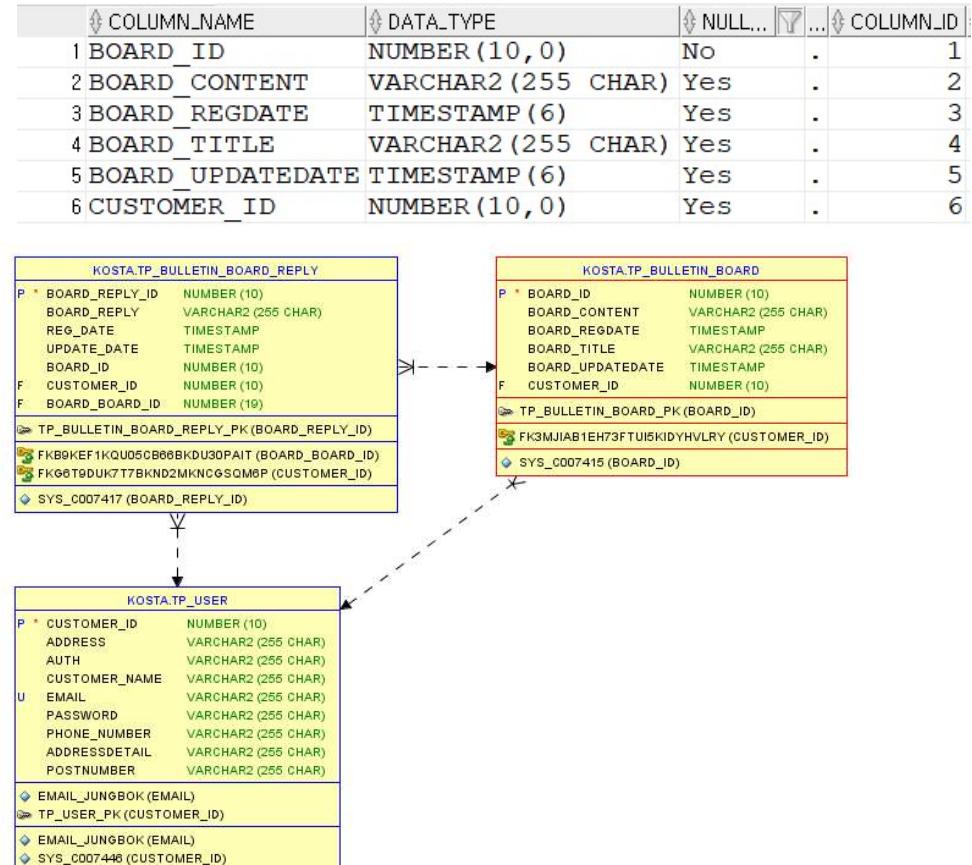


게시판 MODEL board

```

@Getter
@Setter
@ToString(exclude = "replies")
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Builder
@EqualsAndHashCode(of = "bid")
@Table(name = "tp_bulletin_board")

public class Board {
    @Id // 필수PK
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name="board_id")
    Long bid;
    @Column(name="board_title")
    String btitle;
    @ManyToOne
    @JoinColumn(name = "customer_id")
    Member customer; //댓글작성자
    @Column(name="board_content")
    String bcontent;
    @CreationTimestamp
    @Column(name="board_regdate")
    Timestamp bregdate;
    @UpdateTimestamp
    @Column(name="board_updatedate")
    Timestamp bupdatedate;
    @JsonIgnore // tostring과 유사, JSON만들때 무한loop 방지
    @OneToMany(mappedBy = "board", cascade = CascadeType.ALL, fetch = FetchType.LAZY)
    List<BoardReply> replies;
}
  
```





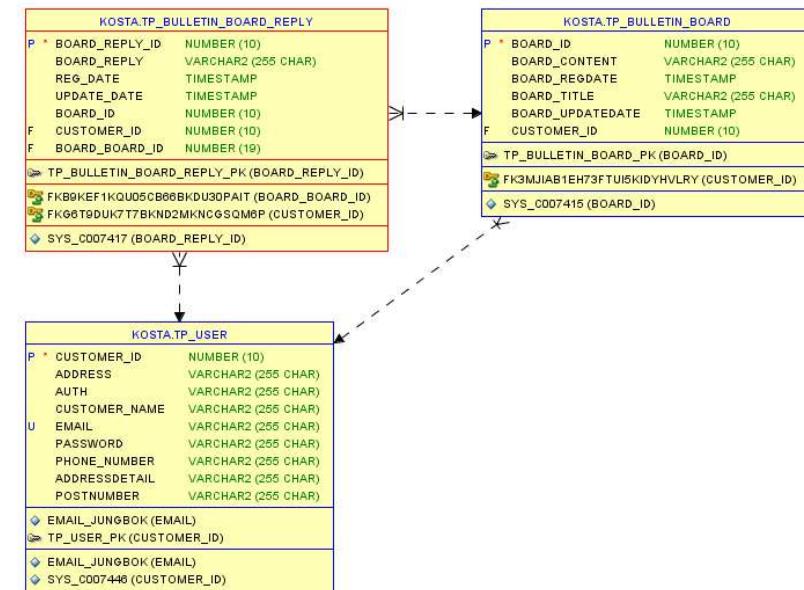
게시판 MODEL board_reply

```

    @Getter
    @Setter
    @ToString(exclude = "customer")
    @AllArgsConstructor
    @NoArgsConstructor
    @Entity
    @Builder
    @EqualsAndHashCode(of = "rid")
    @Table(name = "tp_bulletin_board_reply")
    public class BoardReply {
        @Id
        @GeneratedValue(strategy = GenerationType.AUTO)
        @Column(name="board_reply_id")
        Long rid;
        @Column(name="board_reply")
        String reply; //댓글내용
        @ManyToOne
        @JoinColumn(name = "customer_id")
        Member customer;
        @UpdateTimestamp
        @Column(name="update_date")
        Timestamp rupdatedate;
        @JsonIgnore
        @ManyToOne(fetch = FetchType.EAGER)
        Board board;
    }
}

```

COLUMN_NAME	DATA_TYPE	NULLABLE	COLUMN_ID
1 BOARD_REPLY_ID	NUMBER (10, 0)	No	1
2 BOARD_REPLY	VARCHAR2 (255 CHAR)	Yes	2
3 REG_DATE	TIMESTAMP (6)	Yes	3
4 UPDATE_DATE	TIMESTAMP (6)	Yes	4
5 BOARD_ID	NUMBER (10, 0)	Yes	5
6 CUSTOMER_ID	NUMBER (10, 0)	Yes	6
7 BOARD_BOARD_ID	NUMBER (19, 0)	Yes	7





게시판 MODEL pageVO

- 게시판 구현을 위한 페이징 기능
- 기본 페이지 기능
- 검색을 위한 키워드 페이지 기능
- 댓글, 게시판, 리뷰 기능에서 사용 (모듈화)

```
@Builder
@AllArgsConstructor
@Builder
public class PageVO {
    private static final int DEFAULT_SIZE = 10;
    private static final int DEFAULT_MAX_SIZE = 50;
    int page;
    int size;
    String type; //title or content or writer
    String keyword;
    public PageVO() {
        this.page = 1;
        this.size = DEFAULT_SIZE;
    }
    public void setPage(int page) {
        this.page = page<0?1:page;
    }
    public void setSize(int size) {
        this.size = size < DEFAULT_SIZE||size>DEFAULT_MAX_SIZE?DEFAULT_SIZE:size;
    }
    public Pageable makePaging(int direction, String...props) {
        Sort.Direction dir = direction==0?Direction.DESC:Direction.ASC;
        return PageRequest.of(this.page-1, this.size, dir, props );
    }
    public void setType(String type) {
        this.type = type;
    }
    public void setKeyword(String keyword) {
        this.keyword = keyword;
    }
}
```



게시판 MODEL pageMake

- 게시판 구현을 위한 페이징 기능
- 각 페이지 당 10개의 게시물로 구성
- 페이지는 10개의 리스트로 구성
- 10개 초과시 다음 리스트에 표출
- 최근 페이지를 기억해 상세보기 페이지에서 이전 페이지로 이동
- 댓글, 게시판, 리뷰 기능에서 사용 (모듈화)



```
@Getter  
@ToString(exclude = "pageList")  
public class PageMake<T> {  
    private Page<T> result;  
    private Pageable prevPage;  
    private Pageable nextPage;  
    private Pageable currentPage;  
    private int currentPageNum;  
    private int totalPageNum;  
    private List<Pageable> pageList;  
    public PageMake(Page<T> result) {  
        this.result = result;  
        this.currentPage = result.getPageable();  
        this.currentPageNum = currentPage.getPageNumber() + 1;  
        this.totalPageNum = result.getTotalPages();  
        this.pageList = new ArrayList<Pageable>();  
        calcPage();  
    }  
    public void calcPage() {  
        int cnt = 10;  
        int tempEndNum = (int) (Math.ceil(currentPageNum / (cnt * 1.0)) * cnt);  
        int startNum = tempEndNum - (cnt - 1);  
        Pageable startPage = this.currentPage;  
        for (int i = startNum; i < this.currentPageNum; i++) {  
            startPage = startPage.previousOrFirst();  
        }  
        this.prevPage = startPage.getPageNumber() <= 0 ? null :  
        startPage.previousOrFirst();  
        log.info("tempEndNum:" + tempEndNum);  
        log.info("totalPageNum:" + totalPageNum);  
        if (this.totalPageNum < tempEndNum) {  
            tempEndNum = this.totalPageNum;  
            this.nextPage = null;  
        }  
        for (int i = startNum; i <= tempEndNum; i++) {  
            pageList.add(startPage);  
            startPage = startPage.next();  
        }  
        this.nextPage = startPage.getPageNumber() + 1 < totalPageNum ? startPage :  
        null;  
    }  
}
```



게시판 - 유저토론게시판 (과자갤러리)

과자 갤러리

출출할 땐....??

번호	제목	작성자	작성일
1143	[월드콘X김연경 갓연경 에디션 출시 기념 EVENT] o	swk9...	3분 전
1142	전 세계 60억 포켓몬스터들을 위해 나 서윗한 대리가 준비한 특.급.이.벤트! o	swk9...	3분 전
1141	덥다 더워! 빼속 까지 시원하게 해줄 빙수가 당깁니다.... 음,, 빙수를 만들어 먹어야겠어! (✿'◡`✿) 2	tjdg...	4분 전
1140	※5만원 초과 경품에 대한 제세공과금 22%는 당첨자 부담입니다. o	tjdq...	5분 전
1137	나 서윗한대리, 칙촉 시크릿에 진심이었던 만큼 많은 분들이 드셔셨으면 좋겠다고 매일 꿈꿨었는데 이번에! 칙촉 시크릿 700만봉 판매 소식을 듣게 됐습니다! ^_^(=▽=)^o 2	tjdq...	7분 전
1135	[월드콘X김연경 갓연경 에디션 출시 기념 EVENT] 1	swk9...	9분 전

글쓰기

1. 게시판 로드 시 패키지(제품) 이미지 랜덤 노출
2. 페이지 검색기능(제목, 사용자)
3. 댓글 갯수 표시
4. 작성자 이메일 일부 표시(javascript)
5. 작성시간 표시
6. 페이지 표시

```
@GetMapping("/board/boardlist")
public void selectAll(Model model, PageVO pagevo) {
    if(pagevo==null) pagevo = PageVO.builder().page(1).size(10).keyword("").type("").build();
    Page<Board> result = service.selectAll(pagevo);
    model.addAttribute("boardResult", result);
    model.addAttribute("pagevo", pagevo);
    model.addAttribute("result", new PageMake<>(result));
}
```



게시판 - 유저토론게시판 (과자갤러리)

```
@GetMapping("/board/boardlist") //페이지 불러옴
public void selectAll(Model model, PageVO pagevo) {
    if(pagevo==null) pagevo = PageVO.builder().page(1).size(10).keyword("").type("").build();
    Page<Board> result = service.selectAll(pagevo);
    model.addAttribute("boardResult", result);
    model.addAttribute("pagevo", pagevo);
    model.addAttribute("result", new PageMake<T>(result));
}

@Autowired
BoardPersistance persistance;
// 검색어에 따른 페이지 결과 불러옴
public Page<Board> selectAll(PageVO pvo) {
    Predicate p = persistance.makePredicate(pvo.getType(),pvo.getKeyword());
    Pageable pageable = pvo.makePaging(0, "bid");
    Page<Board> result = persistance.findAll(p, pageable);
    return result;
}

public interface BoardPersistance extends CrudRepository<Board, Long>, QuerydslPredicateExecutor<Board>{
    //페이지 검색기능
    public default Predicate makePredicate(String type, String keyword) {
        BooleanBuilder builder = new BooleanBuilder();
        QBoard board = QBoard.board;
        if(type==null) return builder;
        builder.and(board.bid.gt(0L)); // and board_id>0
        String whereStr = "%" + keyword + "%";
        switch(type) {
            case "btitle":
                builder.and(board.btitle.like(whereStr));
                break;
            case "bcontent":
                builder.and(board.bcontent.like(whereStr));
                break;
            case "email":
                builder.and(board.customer.email.like(whereStr));
                break;
            default:
                break;
        }
        return builder;
    }
}
```

```
@Log
@Getter
@ToString(exclude = "pageList") //페이지 수를 계산하고 페이지 정보를 기억함
public class PageMake<T> {
    private Page<T> result;
    private Pageable prevPage; // 이전으로 이동하는데 필요한 정보를 가짐
    private Pageable nextPage;
    private Pageable currentPage;
    private int currentPageNum; // 화면에 보이는 1부터 시작하는 페이지번호
    private int totalPageNum;
    private List<Pageable> pageList;
    public PageMake(Page<T> result) {
        this.result = result;
        this.currentPage = result.getPageable();
        this.currentPageNum = currentPage.getPageNumber() + 1;
        this.totalPageNum = result.getTotalPages();
        this.pageList = new ArrayList<Pageable>();
        calcPage();
    }
    public void calcPage() {
        int cnt = 10;
        int tempEndNum = (int) (Math.ceil(currentPageNum / (cnt * 1.0)) * cnt);
        int startNum = tempEndNum - (cnt - 1);
        Pageable startPage = this.currentPage;
        for (int i = startNum; i < this.currentPageNum; i++) {
            startPage = startPage.previousOrFirst();
        }
        this.prevPage = startPage.getPageNumber() <= 0 ? null : startPage.previousOrFirst();
        log.info("tempEndNum: " + tempEndNum);
        log.info("totalPageNum: " + totalPageNum);
        if (this.totalPageNum < tempEndNum) {
            tempEndNum = this.totalPageNum;
            this.nextPage = null;
        }
        for (int i = startNum; i <= tempEndNum; i++) {
            pageList.add(startPage);
            startPage = startPage.next();
        }
        this.nextPage = startPage.getPageNumber() + 1 < totalPageNum ? startPage : null;
    }
}
```



게시판 – 유저토론게시판 (과자갤러리)

- 페이지 로드 시마다 배열내 랜덤 이미지 출력

```
var imgArray= new Array();
imgArray[0]="/images/packages/pack1.jpg"; //사진
imgArray[1]="/images/packages/pack2.jpg"; //사진
imgArray[2]="/images/packages/pack3.jpg"; //사진
imgArray[3]="/images/packages/pack4.jpg"; //사진
imgArray[4]="/images/packages/pack5.jpg"; //사진
imgArray[5]="/images/packages/pack6.jpg"; //사진
function showImage()
{
    var imgNum=Math.round(Math.random()*6);
    var objImg=document.getElementById("introImg");
    objImg.src=imgArray[imgNum];
}
```

<p>출출할 땐.....??</p>

- Thymleaf를 통한 게시물별 댓글 개수 출력

```
<tr th:each="board:${boardResult.content}">
<span class="cnt_cmt" style="color: #7e00ff">[${${board.replies.size()}]}]</span></td>
```

- 자바스크립트 함수를 이용해 특정 문자 길이 이후 ... 출력

```
<!-- 문자열의 한계치를 설정하고, length가 한계치 이상이라면 잘라낸후 ...으로 표시해준다. -->
<div class="email" th:text="${#strings.abbreviate(board.customer.email,7)}"></div>
```





게시판 – 유저토론게시판 (과자갤러리)

```
$function() {
    var today = new Date();
    $("tr td:last-child div")
        .each(function(index, item) {
            var aa = new Date($(item).html());
            var time = today - aa;
            console.log(item);
            time = Math.ceil(time / 1000);
            if (time < 60) {
                $(item).html(time+'초 전');
            } else {
                time = Math.ceil(time / 60);
                if (time < 60) {
                    $(item).html(time+'분 전');
                } else {
                    time = Math.ceil(time / 60);
                    if (time < 24) {
                        $(item).html(time+'시간 전');
                    } else {
                        $(item).html((aa.getMonth() + 1 >= 10 ? aa.getMonth() + 1 : '0'
                            + (aa.getMonth() + 1))
                            + "-" + aa.getDate());
                    }
                }
            }
        });
    });
});
```

- 현재시간 대비 게시시간 출력
- 24시간 이후 날짜 출력

```
function formatDate(t) {
    var date = new Date(t)
    return (date.getMonth() + 1 >= 10 ? date.getMonth() + 1 : '0'
        + (date.getMonth() + 1))
        + "-" + date.getDate()
};
```





게시판 – 유저토론게시판 (과자갤러리_게시물 등록)

글쓰기

작성자	swk9514@naver.com
제목	<input type="text" value="【월드콘X김연경 갓연경 에디션 출시 기념 EVENT】"/>
내용	<p>나 서워하, 월를 김연경 선수의 월드콘 온팩을 세상에 널리 알리고자 월클급 경품을 걸고 두 가지 이벤트를 준비했다!</p> <p>모두들 집중의 박수를 ●●●</p> <p>★ EVENT 1. 월드콘 구매인증 이벤트 월드콘 구매를 인증하면 월클급 경품이 우르르</p> <p>👉 참여방법 1. 월드콘 구매 후 영수증과 함께 인증샷 찰칵! 2. 활용한 구매인증 사진과 함께 네이버 이벤트 신청폼을 작성하면 참여 완료 <input checked="" type="checkbox"/> 신청폼: http://naver.me/5SWTGCwA</p> <p>👉 응모 기간</p>
<input style="background-color: red; color: white; border: none; padding: 5px 10px; margin-right: 10px;" type="button" value="등록하기"/> <input style="background-color: red; color: white; border: none; padding: 5px 10px;" type="button" value="RESET"/>	

```

@GetMapping("/board/register")
public void boardRegister() {
}

@PostMapping("/board/register")
public String boardRegisterPost(Board board, RedirectAttributes rttr, Principal principal) {
    Member member = userService.getMemberInfo(principal.getName());
    board.setCustomer(member);
    Board ins_board = service.insertBoard(board);
    rttr.addFlashAttribute("resultMessage", ins_board==null?"입력실패":"입력성공");
    return "redirect:/board/boardlist";
}

@.Autowired
MemberRepository repo;
public Member getMemberInfo(String email) {
    return repo.findByEmail(email).get();
}

```

- 등록 -> save (board/register : post)
- RESET -> 내용 및 제목 지우기





게시판 - 유저토론게시판 (과자갤러리_게시물 상세보기)

자유게시판

전 세계 60억 포켓몬스터들을 위해 나 서윗한 대리가 준비한 특.급.이.벤.트!

작성자:swk9... 작성일:2021-07-06

귀엽고 앙큼한 꼬깔콘 모델 매드몬스터처럼 꼬깔콘을 활용한 사진을 찍어 꼬깔몬스터를 인증해주세요! 포켓몬스터라면 가보로 간직해야 할 매운 친필사 인지! 포문의 감성을 지켜줄 노르디스크 텐트와 웨건 카트와 매온의 노래를 고음질로 감상할 수 있는 마샬 블루투스 스피커까지 절대 참을 수 없는 선물이 와르르

자유게시판

전 세계 60억 포켓몬스터들을 위해 나 서윗한 대리가 준비한 특.급.이.벤.트!

작성자:swk9... 작성일:2021-07-06

귀엽고 앙큼한 꼬깔콘 모델 매드몬스터처럼 꼬깔콘을 활용한 사진을 찍어 꼬깔몬스터를 인증해주세요! 포켓몬스터라면 가보로 간직해야 할 매운 친필사 인지! 포문의 감성을 지켜줄 노르디스크 텐트와 웨건 카트와 매온의 노래를 고음질로 감상할 수 있는 마샬 블루투스 스피커까지 절대 참을 수 없는 선물이 와르르

내용	작성자	작성일
참여기간: 07월 01일 ~ 07월 31일	swk9***	07-6
1. 마트/편의점/롯데스퀘어 등에서 꼬깔콘을 구입한다. 2. 꼬깔콘을 다양하게 활용하여 나만의 매드몬스터를 표현한 사진을 활용한다. 3. SNS에 필수 해시태그와 함께 업로드하면 끝! (꼬깔콘 패키지를 풍성하게 인증하면 당첨권률 UP!)	swk9***	07-6
필수 해시태그: #꼬깔몬스터 #꼬깔콘 #레즈고잇	tjda***	07-6
댓글 내용	댓글추가	수정 삭제

복록보기

내용	작성자	작성일
참여기간: 07월 01일 ~ 07월 31일	swk9***	07-6
1. 마트/편의점/롯데스퀘어 등에서 꼬깔콘을 구입한다. 2. 꼬깔콘을 다양하게 활용하여 나만의 매드몬스터를 표현한 사진을 활용한다. 3. SNS에 필수 해시태그와 함께 업로드하면 끝! (꼬깔콘 패키지를 풍성하게 인증하면 당첨권률 UP!)	swk9***	07-6
필수 해시태그: #꼬깔몬스터 #꼬깔콘 #레즈고잇	tjda***	07-6
댓글 내용	댓글추가	수정 삭제

수정하기

삭제하기

수정 삭제

수정 삭제

- 로그인 정보와 게시물, 댓글 정보를 비교 후 수정 삭제 권한 부여
- 댓글 CRUD 기능
- 댓글 수정 중 수정, 삭제 버튼 비활성화

```
<div th:if="${board.customer.email}==${#authentication.name}" style="float: right;">
    <a href="javascript:boardMod()">수정하기</a>
    <a th:href="@{/board/delete(bid=${board.bid})}">삭제하기</a>
</div>
```





게시판 – 유저토론게시판 (과자갤러리_게시물 상세보기)

```
//특정 보드 번호에 해당하는 모든 댓글 조회
@GetMapping("/board/{bid}")
public ResponseEntity<List<BoardReply>> selectAll(@PathVariable Long bid) {
    Board board = Board.builder().bid(bid).build();
    return new ResponseEntity<>(service.selectAll(board), HttpStatus.OK);
}
//댓글상세보기
@GetMapping("{rid}")
public ResponseEntity<BoardReply> viewReply(@PathVariable Long rid) {
    return new ResponseEntity<>(service.selectById(rid), HttpStatus.OK);
}
//특정 보드에 댓글등록, 입력 후 다시 조회
@PostMapping("/{bid}")
public ResponseEntity<List<BoardReply>> addReply(@PathVariable("bid")Long bid, BoardReply reply, String email) {
    Board board = Board.builder().bid(bid).build();
    reply.setBoard(board);
    reply.setCustomer(userService.getMemberInfo(email));
    service.updateOrInsert(reply);
    return new ResponseEntity<>(service.selectAll(board), HttpStatus.CREATED);
}
//삭제
@DeleteMapping("/{bid}/{rid}")
public ResponseEntity<List<BoardReply>> deleteByrid(@PathVariable Long rid, @PathVariable Long bid) {
    service.delete(rid);
    Board board = Board.builder().bid(bid).build();
    return new ResponseEntity<>(service.selectAll(board), HttpStatus.OK);
}
//수정 {bid: "48", reply: "fewafe", customer: "25", rid: "90"}
@PutMapping("/{bid}/{rid}")
public ResponseEntity<List<BoardReply>> updateRep(@PathVariable Long bid,
String customer, @PathVariable Long rid, String reply) {
    BoardReply breply = new BoardReply();
    Board board = boardService.selectById(bid);
    breply.setBoard(board);
    breply.setCustomer(userService.getMemberInfoById(Integer.parseInt(customer)));
    breply.setRid(rid);
    breply.setReply(reply);
    service.updateOrInsert(brapid);
    return new ResponseEntity<>(service.selectAll(board), HttpStatus.OK);
}
```

```
var replyManager =(function(){
    var getAll = function(obj, callback){
        $.getJSON("/replies/board/"+obj, callback)
    };
    var add = function(obj, callback){
        $.ajax({
            url:"/replies/" + obj["bid"],
            data: obj,
            type: "post",
            success:callback
        });
    };
    var update = function(obj, callback){
        $.ajax({
            url:"/replies/" + obj["bid"] + "/" + obj["rid"],
            data: obj,
            type: "put",
            success:callback
        });
    };
    var remove = function(obj, callback){
        $.ajax({
            url:"/replies/" + obj["bid"] + "/" + obj["rid"],
            type:"delete",
            success:callback
        });
    };
    return{
        "getAll" : getAll,
        "add": add,
        "update" : update,
        "remove" : remove
    };
})();
```

프로젝트 소개

김성휘

- ✓ 유저토론 게시판(과자갤러리)
- ✓ 문의게시판(QnA)
- ✓ 리뷰 및 별점(평점)



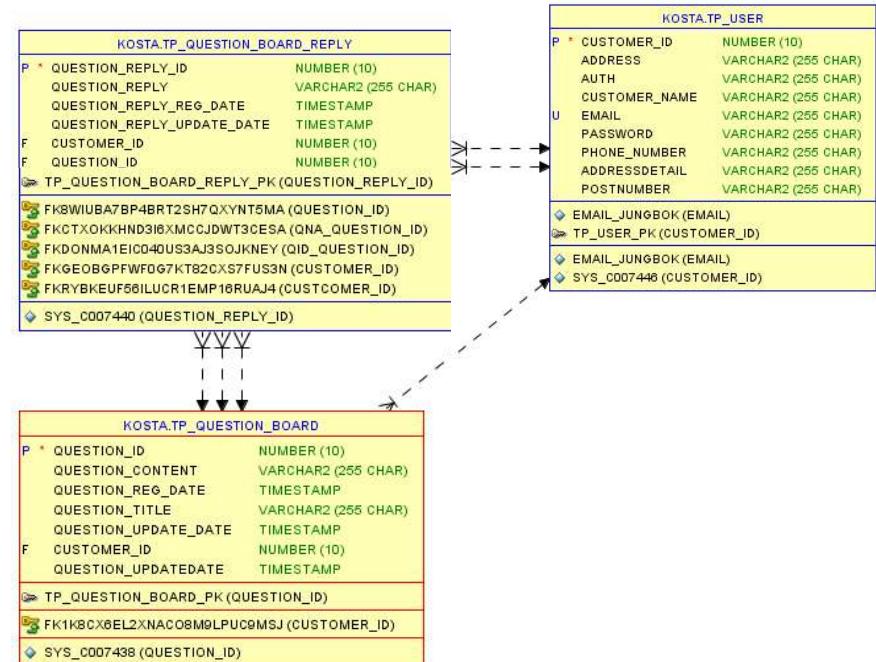


개시판 MODEL QnA

```

@Getter
@Setter
@ToString(exclude = "qreplies")
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Builder
@EqualsAndHashCode(of = "qid")
@Table(name = "tp_question_board")
public class QnA {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "question_id")
    Long qid;
    @Column(name = "question_title")
    String qtitle;
    @Column(name = "question_content")
    String qcontent;
    @ManyToOne
    @JoinColumn(name = "customer_id")
    Member customer; // 문의글 작성자
    @CreationTimestamp
    @Column(name = "question_reg_date")
    Timestamp qregDate;
    @UpdateTimestamp
    @Column(name = "question_updatedate")
    Timestamp qupdateDate;
    @JsonIgnore // tostring과 유사, JSON만들때 무한loop 방지
    @OneToMany(mappedBy = "qna", cascade = CascadeType.ALL, fetch = FetchType.LAZY)
    List<QnAReply> qreplies;
}

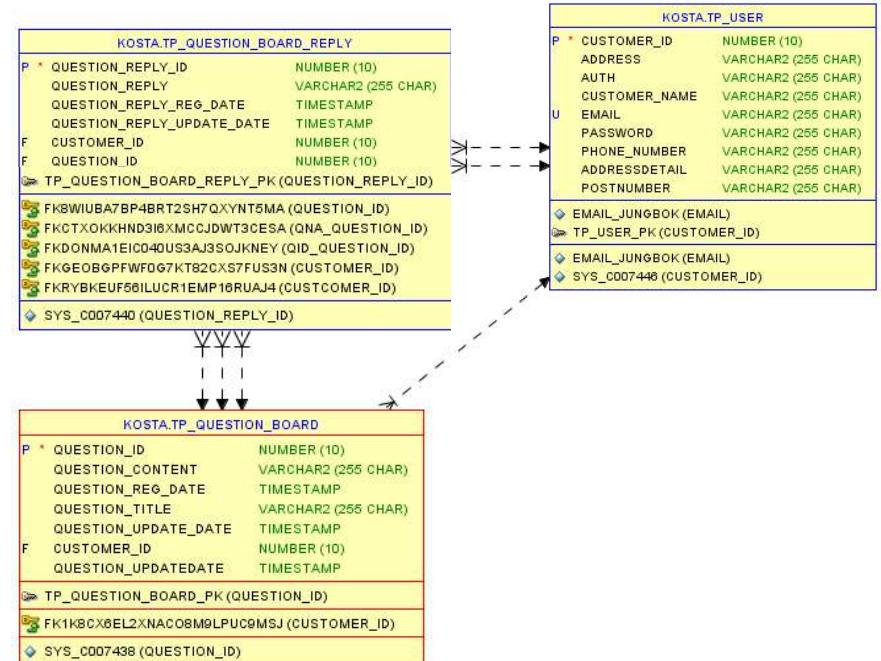
```





개시판 MODEL QnA_reply

```
@Getter  
@Setter  
@ToString(exclude = "admin")  
@AllArgsConstructor  
@NoArgsConstructor  
@Entity  
@Builder  
@EqualsAndHashCode(of = "qrid")  
@Table(name = "tp_question_board_reply")  
public class QnAReply{  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    @Column(name="question_reply_id")  
    Long qrid;  
    @Column(name="question_reply")  
    String qreply;  
    @ManyToOne(fetch = FetchType.EAGER)  
    QnA qna;  
    @ManyToOne  
    @JoinColumn(name = "customer_id")  
    Member admin;  
    @CreationTimestamp  
    @Column(name="question_reply_reg_date")  
    Timestamp qrepyregDate;  
    @UpdateTimestamp  
    @Column(name="question_reply_update_date")  
    Timestamp qrupdateDate;  
}
```





게시판 - 문의게시판

전체 ▾

번호	제목	작성자	작성일
286	문희?	고석우	06-30
262	스낵 24	고석우	06-28
224	프로젝트 잘되가요? 답변완료	고석우	06-24
216	테스트페이지 입니다 답변완료	고석우	06-23
215	테스트페이지 입니다	고석우	06-23

전체 ▾

문의하기

전체 ▾

번호	제목	작성자	작성일
284	문의사항	강성빈	06-30
283	프로젝트 잘되가요?	강성빈	06-30

문의하기

전체 ▾

번호	제목	작성자	작성일
1150	어디 과자가 맛있나요?	김성휘	10분 전
286	문희?	고석우	06-30
284	문의사항	강성빈	06-30
283	프로젝트 잘되가요?	강성빈	06-30
262	스낵 24	고석우	06-28
224	프로젝트 잘되가요? 답변완료	고석우	06-24
216	테스트페이지 입니다 답변완료	고석우	06-23
215	테스트페이지 입니다	고석우	06-23

```

@GetMapping("/board/qnalist")
public void selectAll(Model model, PageVO pagevo) {
    Object principal = SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    String email = ((User) principal).getUsername();
    String auth = uservice.getMemberInfo(email).getAuth().name();
    Integer id = (int) uservice.getMemberInfo(email).getCustomer_id();
    Page<QnA> result;
    if(auth.equals("ADMIN")) {
        result = service.selectAll(pagevo);
    } else {
        result = service.selectByUser(pagevo, id);
    }
    model.addAttribute("qnaResult", result);
    model.addAttribute("pagevo", pagevo);
    model.addAttribute("result", new PageMake(result));
}

```

- **로그인 계정과 qna 정보를 비교하여 본인이 작성한 글만 출력**
- **운영자 답변 시 답변완료 활성화**
- **운영자 계정에서는 전체 조회 가능**
- **운영자 계정에서만 답변작성 가능**

프로젝트 소개

김성휘

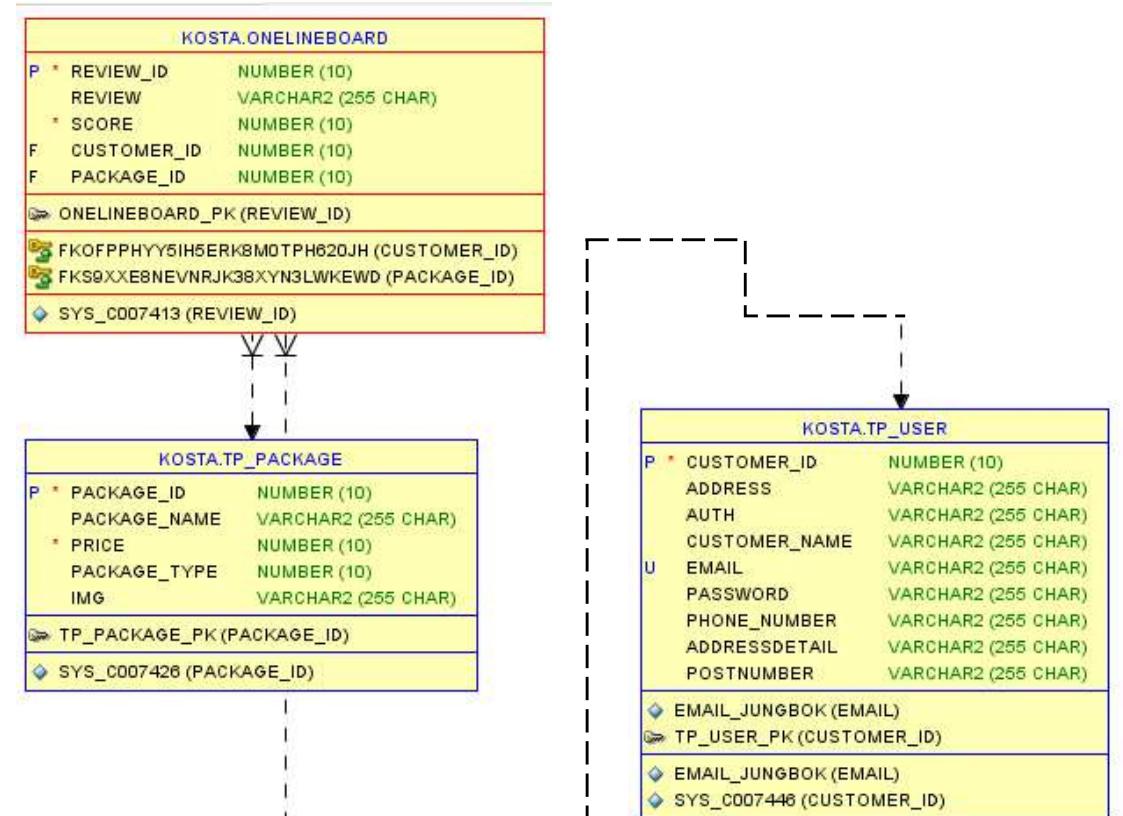
- ✓ 유저토론 게시판(과자갤러리)
- ✓ 문의게시판(QnA)
- ✓ 리뷰 및 별점(평점)





개시판 MODEL review

```
@Getter  
@Setter  
@ToString(exclude = "pack")  
@AllArgsConstructor  
@NoArgsConstructor  
@Entity  
@Builder  
@Table(name = "onelineboard")  
public class Review {  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    @Column(name = "review_id")  
    Long reviewId;  
    @ManyToOne  
    @JoinColumn(name = "package_id")  
    PackageVO pack;  
    @ManyToOne  
    @JoinColumn(name = "customer_id")  
    Member customer;  
    int score;  
    String review;  
}
```





게시판_리뷰

번호	패키지	한줄평	별점	작성자
301	LOTTE_PACKAGE	bad		swk9...
300	LOTTE_PACKAGE	good		swk9...

작성자 swk9514@naver.com

패키지 패키지를 선택하세요..... ^

내용 패키지를 선택하세요.....
LOTTE_PACKAGE

패키지선택은 구독 내역에서...

별점

등록하기 | **RESET**

- 별점기능(radioBox)
- 리뷰작성시 설정
- DB 저장시 트리거를 통한 평점 저장
- 구독이력이 있는 상품만 리뷰 가능
- 1계정 - 1상품 - 1리뷰



게시판_리뷰 (별점 기능)

별점



```
<tr th:each="review, row:${reviewResult.content}">
    <td>
        <div class="startRadio">
            <label class="startRadio_box" for="score" th:each="num:${#numbers.sequence(1,10)}">
                <input type="radio" th:name="'score' + ${row}" th:value="${num}" th:checked="${review.score>=num?true:false" disabled="disabled">
                <span class="startRadio_img"><span class="blind" th:text="${num}"></span></span>
            </label>
        </div>
    </td>

```

```
.startRadio {
display: inline-block;
overflow: hidden;
height: 40px;
}
.startRadio:after {
content: "";
display: block;
position: relative;
z-index: 10;
height: 40px;
background: url("data:image/png;base64,~~~~");
background-size: contain;
pointer-events: none;
}
```

- 별점 디자인 이미지 css
- 별점기능(radioButton)
- 리뷰작성시 설정
- DB 저장시 트리거를 통한 평점 저장
- 구독이력이 있는 상품만 리뷰 가능

```
    .startRadio_box {
position: relative;
z-index: 1;
float: left;
width: 20px;
height: 40px;
cursor: pointer;
}
.startRadio_box input {
opacity: 0 !important;
height: 0 !important;
width: 0 !important;
position: absolute !important;
}
.startRadio_box input:checked + .startRadio_img {
background-color: #0084ff;
}
.startRadio_img {
display: block;
position: absolute;
right: 0;
width: 500px;
height: 40px;
pointer-events: none;
}
```

- 별점 이벤트 이미지 css



게시판_리뷰 (별점 기능)

```
CREATE TABLE ascore
AS SELECT * FROM ONELINEBOARD;
```

- 리뷰_테이블 복사

```
CREATE OR REPLACE TRIGGER UPDATE_score
AFTER UPDATE ON ONELINEBOARD
FOR EACH ROW
BEGIN
    update ascore
    set score = :NEW.score
    where PACKAGE_ID = :NEW.PACKAGE_ID and REVIEW_ID = :NEW.REVIEW_ID and CUSTOMER_ID = :NEW.CUSTOMER_ID;
END;
```

- 리뷰_테이블 업데이트 시 트리거작동

```
CREATE OR REPLACE TRIGGER INSERT_score
AFTER INSERT ON ONELINEBOARD
FOR EACH ROW
BEGIN
    insert into ascore
    values (:NEW.REVIEW_ID, :NEW.SCORE, :NEW.PACKAGE_ID, :NEW.CUSTOMER_ID,
    :NEW.review);
END;
```

- 리뷰_테이블 삽입 시 트리거작동

```
CREATE OR REPLACE TRIGGER score
AFTER INSERT ON ONELINEBOARD
FOR EACH ROW
FOLLOWS INSERT_score
BEGIN update TP_PACKAGE
    set score = (select avg(score)
    from ascore
    group by PACKAGE_ID
    having PACKAGE_ID = :NEW.PACKAGE_ID)
    where score = :NEW.PACKAGE_ID;
END;
```

- ascore테이블 삽입 시 트리거작동
- Score의 평균 값을 패키지 테이블에 저장



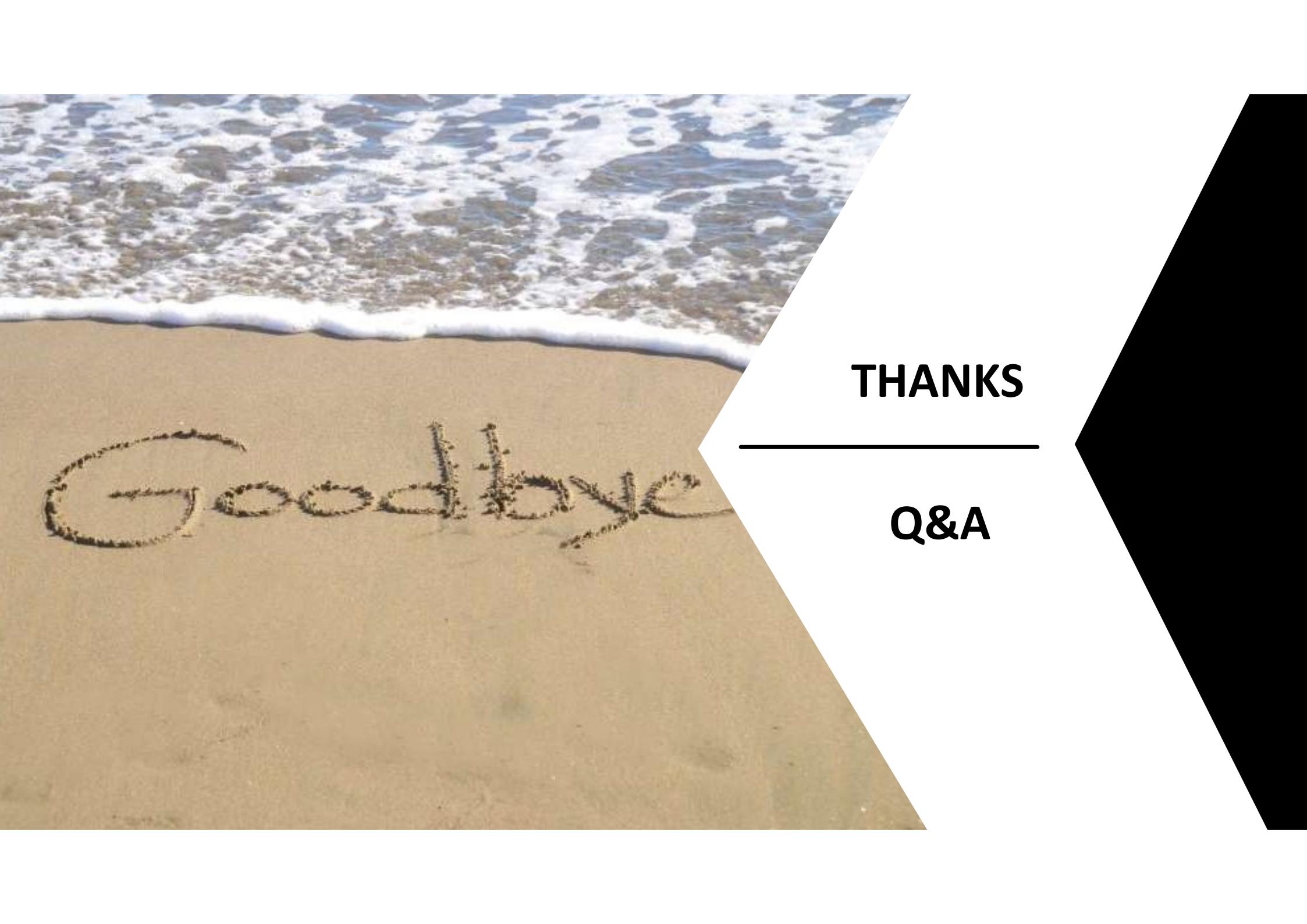
04

시연 & QnA

Snack_24

Snack_24





THANKS

Q&A