

티켓 예매 서비스



한화시스템 Beyond  
SW캠프

BACKEND  
PROJECT

**TEAM : 5RANGE**

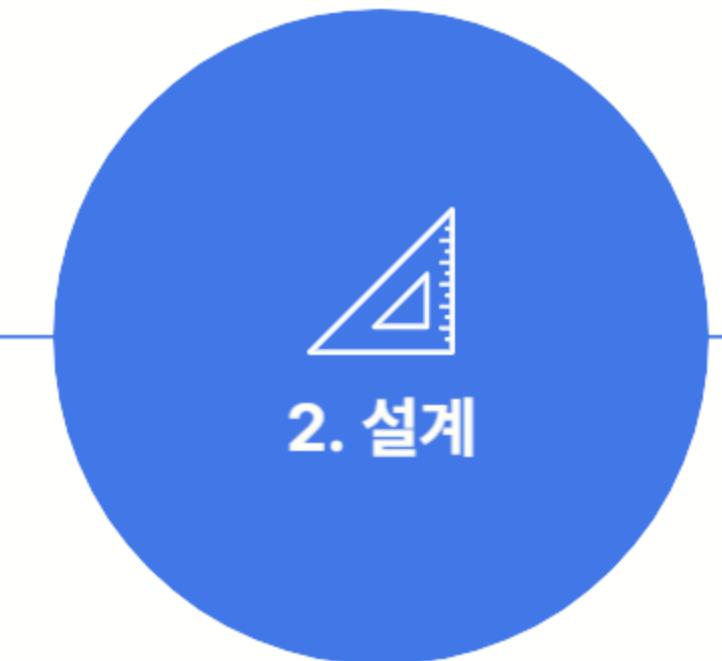
장현준 계용운 김지원 전지연 조혜인

# 01 목차



1. 소개

- [프로젝트 배경 및 소개](#)
- [프로젝트 기대효과](#)
- [TOOLS](#)



2. 설계

- 요구사항 명세서
- 테이블 명세서
- API 명세서
- ERD



3. 결과

- 주요 기능 시연
- 개선사항

01

소개

- POINT 01 프로젝트 배경 및 소개
- POINT 02 프로젝트 기대 효과

# 01 프로젝트 배경 및 소개



코로나 거리두기 해제 이후 티켓 예매 서비스 사용자는 22년 8월 기준 티켓 예매 서비스 사용률은 34.0% 수준으로 코로나 이전 대비 사용률은 4.5% 증가하였습니다

최근 문화 활동의 증가와 함께 온라인 예약이 주를 이루고 있는 가운데, 다양한 예약 사이트가 분산되어 있고 특화된 예약 서비스가 부족하여 고객들의 니즈를 충족시키는 통합 관리 방안이 필요하다는 인식에서 출발한 TICKET LINK 서비스입니다.

TICKET LINK는 뮤지컬, 연극, 콘서트 등 입장권 기반의 다양한 항목에 대한 예약 서비스를 통합하여 제공할 예정이며 공연 관람을 위한 전 과정을 효율적으로 지원하는 장점을 갖고 있습니다.

# 01 프로젝트 기대효과



## 편리한 접근성

사용자들은 집에서나 어디에서든 인터넷을 통해 티켓을 쉽게 구매할 수 있으며 이는 고객들에게 훨씬 편리하고 접근성이 높은 서비스를 제공합니다.



## 판매 데이터 분석

티켓팅 서비스를 통해 얻은 데이터를 바탕으로 마케팅 전략을 개선하고, 향후 이벤트 기획에 반영할 수 있습니다.



## 정보 투명성

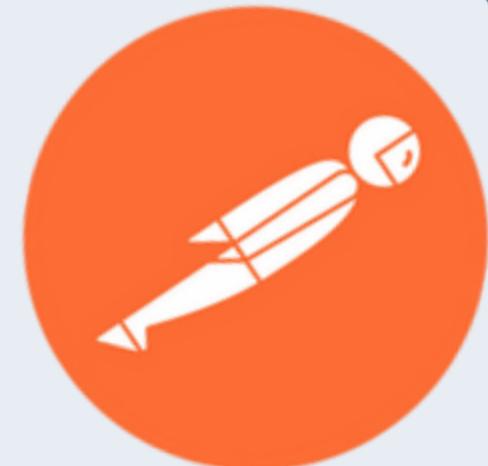
티켓 구매와 관련된 정보들이 명확하게 제공되어, 사용자들은 필요한 정보 및 후기와 평점을 통하여 구매 결정을 내리는 데 있어서 더 나은 정보 기반의 선택을 가능하게 합니다



## 고객 만족도 향상

신속하고 편리한 예약 시스템을 통해 고객 만족도를 높이고, 사이트의 재방문율을 향상시킬 수 있습니다.

# 01 TOOLS



02

## 설계

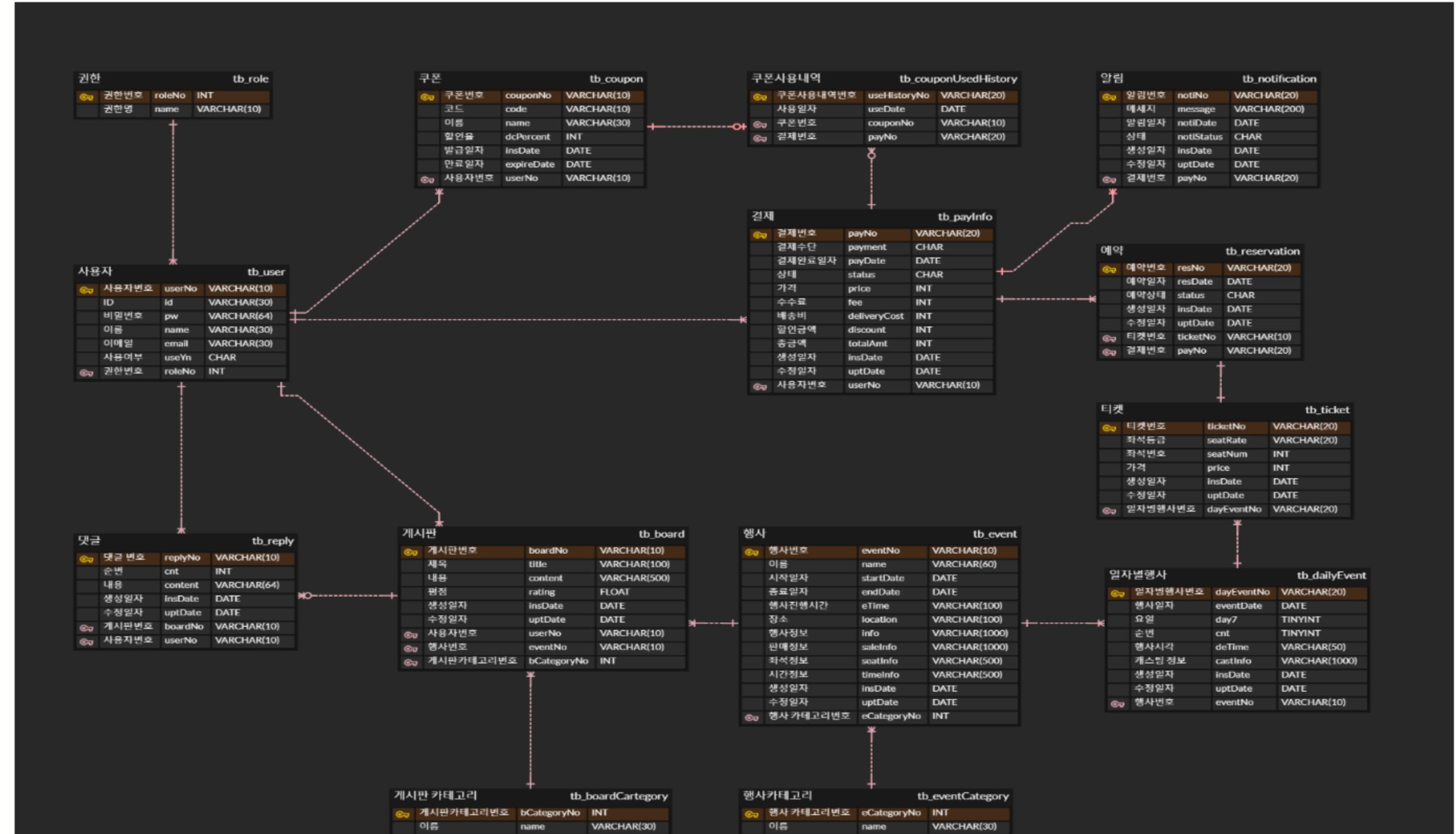
- POINT 01 요구사항 명세서
- POINT 02 ERD
- POINT 03 테이블 명세서
- POINT 04 API 명세서

# 02 요구사항 명세서

프로젝트 명	티켓 예매 서비스
시작일	2024년 06월 24일
종료일	2024년 07월 30일
작성자	개발팀 김지현, 정하준, 허지연, 조혜민
작성일	2024년 06월 24일

요구사항-ID	요구사항명		요구사항 내용	대비율팅
	대분류	소분류		
RQ-0101	사용자	개인 가입	<ul style="list-style-type: none"><li>사용자의 아이디를 입력 받는다.<ul style="list-style-type: none"><li>- 6 ~ 20자 영문 숫자</li><li>- 아이디는 중복되지 않아야한다.</li></ul></li><li>비밀번호를 입력 받는다.<ul style="list-style-type: none"><li>- 8 ~ 12자 영문 숫자 특수문자</li><li>- 사용자에게 비밀번호를 한 번 더 입력 받게해 비밀번호 확인을 받는다.</li><li>- 비밀번호는 암호화하여 DB에 저장한다.</li></ul></li><li>이름을 입력 받는다.</li><li>이메일을 입력 받는다.<ul style="list-style-type: none"><li>- 본인 확인을 위해 이메일 인증을 받는다.</li><li>- 인증번호를 전송해 온밀한 인증번호를 입력 받았는지 확인한다.</li></ul></li></ul>	사용자
RQ-0102		SNS 가입	<ul style="list-style-type: none"><li>이름 정보를 제공 받는다.</li><li>이메일 정보를 제공 받는다.</li></ul>	
RQ-0103		개인 로그인	<ul style="list-style-type: none"><li>아이디와 비밀번호를 입력 받아 일치할 경우 로그인 처리를 한다.</li></ul>	
RQ-0104		로그인 로그인	<ul style="list-style-type: none"><li>사용자는 가까운 로그인을 할 수 있다.</li><li>단순 로그인을 제공할 경우 로그인 처리를 한다.</li></ul>	
RQ-0105		아이디 찾기	<ul style="list-style-type: none"><li>이메일을 입력 받는다.</li><li>이메일에 인증을 위해 이메일에 인증번호를 전송한 뒤 사용자에게 입력 받는다.</li><li>인증번호가 동일할 경우 회원은 새로운 비밀번호를 입력한다.</li><li>유료회원 회원이 동일한 경우 회원은 해당 회원의 아이디를 전달한다.</li></ul>	
RQ-0106		비밀번호 찾기	<ul style="list-style-type: none"><li>아이디와 이름, 이메일을 입력 받는다.</li><li>아이디, 이름을 기반으로 이메일 인증 과정으로 넘어간다.</li><li>이메일 인증을 위해 이메일에 인증번호를 전송한 뒤 사용자에게 입력 받는다.</li><li>인증번호가 동일할 경우 회원은 새롭고 비밀번호를 입력한다.</li><li>유료회원 회원이 동일한 경우 회원은 해당 회원의 아이디를 전달한다.</li></ul>	
RQ-0107		사용자 정보 조회	<ul style="list-style-type: none"><li>사용자가 본인의 정보를 조회할 수 있다.<ul style="list-style-type: none"><li>- 이름, 이메일</li><li>- 생년</li><li>- 주소</li><li>- 즐겨찾기한 티켓</li><li>- 예매일, 예약번호, 공연명, 관람일, 매수, 티켓 가능일, 상대좌장, 티켓</li><li>- 트리너</li></ul></li></ul>	
RQ-0108		사용자 정보 수정	<ul style="list-style-type: none"><li>사용자가 본인의 정보를 수정할 수 있다.<ul style="list-style-type: none"><li>- 이름</li><li>- 배우자 주소</li></ul></li></ul>	
RQ-0109		사용자 신호 정보 카카오	<ul style="list-style-type: none"><li>사용자가 본인의 신호 정보를 카카오(공연/이벤트 카테고리, 지향, 알림설정 등)에 가입한 사용자는 동회될 수 있다.</li></ul>	
RQ-0110		달회	<ul style="list-style-type: none"><li>사용자가 달회를 경우 사용자와 친한친한 정보는 상태 정보를 수정해서 일반 사용자에게 안내지를 처리.<ul style="list-style-type: none"><li>- 구체 수정</li></ul></li></ul>	
RQ-0201	상품	상품 정보 조회	<p>상품</p> <ol style="list-style-type: none"><li>상품 이름</li><li>장르</li><li>장르시간(공연, 콘서트의 경우)</li><li>기간</li><li>관람여정</li><li>가격</li><li>장소 분류(가격)으로 구분됨</li></ol>	상품
RQ-0202		상품 예매	<ul style="list-style-type: none"><li>사용자는 본 상품에 대해 예매권 사용권리를 가진다.</li><li>예매 일자를 선택한다.</li><li>상품 분류를 선택한다.<ul style="list-style-type: none"><li>- 티켓, 콘서트와 같은 공연의 경우 회차</li><li>- 티켓의 경우 기간</li></ul></li></ul>	
RQ-0203		상품 정보 조회	<ul style="list-style-type: none"><li>상품에 대한 정보는 개시글로서 표시되어 마크나운 형식으로 작성된다.</li><li>영역 정보는 관리자에 의해 작성된다.</li></ul>	
RQ-0204		상품 판매 정보 조회	<ul style="list-style-type: none"><li>상품에 대한 판매 정보는 개시글로서 표시되어 마크나운 형식으로 작성된다.</li><li>상품에 대한 판매 정보는 관리자에 의해 작성된다.</li></ul>	
RQ-0301	티켓	예매 가능한 좌석 선택 가능	<ul style="list-style-type: none"><li>사용자는 예매 가능한 좌석을 선택할 수 있다.(좌석 배치도 제공)</li></ul>	티켓
RQ-0302		티켓 수량 및 종류 선택	<ul style="list-style-type: none"><li>티켓 수량 및 종류를 선택할 수 있다.(성인, 어린이, 할인 등)</li></ul>	
RQ-0303		길래 수단 선택	<ul style="list-style-type: none"><li>길래 수단을 선택할 수 있다.(신용카드, 계정이체, 간편결제, 저장된 길래정보 등)</li></ul>	
RQ-0304		예매 키스 가능	<ul style="list-style-type: none"><li>사용자가 예파한 티켓을 취소할 수 있다.</li></ul>	
RQ-0305		키스나 길래 대안으로 예매	<ul style="list-style-type: none"><li>사용자는 키스 티켓 예매가 모전이 되면 티켓을 예매할 수 있다.</li></ul>	
RQ-0401	쿠폰	쿠폰 조회	<ul style="list-style-type: none"><li>사용자는 본인이 가지고 있는 쿠폰을 조회할 수 있다.</li></ul>	쿠폰
RQ-0501		예매 키언 알통	<ul style="list-style-type: none"><li>사용자에게 예매 키언을 알리울 수 있다.</li></ul>	
RQ-0502		예매 키언 알통	<ul style="list-style-type: none"><li>사용자에게 예매 키언을 알리울 수 있다.</li></ul>	
RQ-0503		공연/이벤트 시작 전 알림	<ul style="list-style-type: none"><li>사용자에게 예매한 공연/이벤트 시작 전 알림을 보내줄 수 있다.</li></ul>	알림

# 02 ERD



# 02 테이블 명세서

## <권한 테이블, 사용자 테이블>

권한 테이블(tb\_role)

속성명	열 이름	데이터 유형	NULL 여부	제약조건	기본값	PK	FK	부모 테이블
권한 번호	roleNo	INT	NOT NULL		AUTO_INCREMENT	O		
권한명	name	VARCHAR(10)	NOT NULL	UNIQUE				

사용자 테이블(tb\_user)

속성명	열 이름	데이터 유형	NULL 여부	제약조건	기본값	PK	FK	부모 테이블
사용자 번호	userNo	VARCHAR(10)	NOT NULL			O		
ID	id	VARCHAR(30)	NOT NULL	UNIQUE				
비밀번호	pw	VARCHAR(64)	NOT NULL					
이름	name	VARCHAR(30)	NOT NULL					
이메일	email	VARCHAR(30)	NOT NULL					
사용여부	useYn	CHAR						
권한 번호	roleNo	INT				O		tb_role

## < 행사 관련 테이블 >

행사 카테고리 테이블(tb\_eventCategory)

속성명	열 이름	데이터 유형	NULL 여부	제약조건	기본값	PK	FK	부모 테이블
행사 카테고리 번호	eCategoryNo	INT	NOT NULL		AUTO_INCREMENT	O		
이름	name	VARCHAR(30)	NOT NULL					

행사 테이블(tb\_event)

속성명	열 이름	데이터 유형	NULL 여부	제약조건	기본값	PK	FK	부모 테이블
행사 번호	eventNo	VARCHAR(10)	NOT NULL			O		
이름	name	VARCHAR(60)	NOT NULL					
시작일자	startDate	DATE	NOT NULL					
종료일자	endDate	DATE	NOT NULL					
행사진행시간	eTime	VARCHAR(100)						
장소	location	VARCHAR(100)	NOT NULL					
행사정보	info	VARCHAR(1000)						
판매정보	saleInfo	VARCHAR(1000)						
좌석정보	seatInfo	VARCHAR(500)						
시간정보	timeInfo	VARCHAR(500)						
사용여부	useYn	CHAR		Y				
행사 카테고리 번호	eCategoryNo	INT	NOT NULL			O	tb_eventCategory	
생성일자	insDate	DATE						
수정일자	uptDate	DATE						

일자별 행사 테이블(tb\_dailyEvent)

속성명	열 이름	데이터 유형	NULL 여부	제약조건	기본값	PK	FK	부모 테이블
일자별 행사 번호	dailyEventNo	VARCHAR(20)	NOT NULL			O		
행사일자	eventDate	DATE						
요일	day7	TINYINT						
순번	cnt	TINYINT						
행사시각	deTime	VARCHAR(50)						
캐스팅정보	castInfo	VARCHAR(1000)						
행사 번호	eventNo	VARCHAR(10)	NOT NULL			O	tb_event	
생성일자	insDate	DATE			CURDATE()			
수정일자	uptDate	DATE			CURDATE()			

## <티켓 테이블, 결제 테이블>

티켓 테이블(tb\_ticket)

속성명	열 이름	데이터 유형	NULL 여부	제약조건	기본값	PK	FK	부모 테이블
티켓 번호	ticketNo	VARCHAR(20)	NOT NULL			O		
좌석 등급	seatRate	VARCHAR(20)						
좌석 번호	seatNum	INT						
가격	price	INT	NOT NULL					
일자별 행사 번호	dailyEventNo	VARCHAR(20)	NOT NULL				O	tb_dailyEvent
생성일자	insDate	DATE			CURDATE()			
수정일자	uptDate	DATE			CURDATE()			

결제 테이블(tb\_payinfo)

속성명	열 이름	데이터 유형	NULL 여부	제약조건	기본값	PK	FK	부모 테이블
결제 번호	payNo	VARCHAR(20)	NOT NULL			O		
결제수단	payment	CHAR	NOT NULL					
결제완료일자	payDate	DATE						
상태	Status	CHAR	NOT NULL		W			
가격	price	INT			0			
수수료	fee	INT			0			
배송비	deliveryCost	INT			0			
할인금액	discount	INT			0			
총금액	totalAmt	INT			0			
사용자 번호	userNo	VARCHAR(10)	NOT NULL				O	tb_user
생성일자	insDate	DATE			CURDATE()			
수정일자	uptDate	DATE			CURDATE()			

## <예약 테이블, 알림 테이블>

**예약 테이블(tb\_reservation)**

속성명	열 이름	데이터 유형	NULL 여부	제약조건	기본값	PK	FK	부모 테이블
예약 번호	resNo	VARCHAR(20)	NOT NULL			O		
예약일자	resDate	DATE	NOT NULL		CURDATE()			
상태	Status	STATUS	NOT NULL		W			
티켓 번호	ticketNo	VARCHAR(20)	NOT NULL				O	tb_ticket
결제 번호	payNo	VARCHAR(20)	NOT NULL				O	tb_payinfo
생성일자	insDate	DATE			CURDATE()			
수정일자	uptDate	DATE			CURDATE()			

**알림 테이블(tb\_notification)**

속성명	열 이름	데이터 유형	NULL 여부	제약조건	기본값	PK	FK	부모 테이블
알림 번호	notiNo	VARCHAR(20)	NOT NULL			O		
메세지	message	VARCHAR(200)	NOT NULL					
알림일자	notiDate	DATE	NOT NULL		CURDATE()			
알림상태	notiStatus	CHAR	NOT NULL		Y			
결제 번호	payNo	VARCHAR(20)	NOT NULL				O	tb_payinfo
생성일자	insDate	DATE			CURDATE()			
수정일자	uptDate	DATE			CURDATE()			

## <쿠폰 테이블, 쿠폰사용내역 테이블>

쿠폰 테이블(tb\_coupon)

속성명	열 이름	데이터 유형	NULL 여부	제약조건	기본값	PK	FK	부모 테이블
쿠폰 번호	couponNo	VARCHAR(10)	NOT NULL			O		
코드	code	VARCHAR(10)						
이름	name	VARCHAR(30)						
할인율	dcPercent	INT						
생성일자	insDate	DATE			CURDATE()			
만료일자	expireDate	DATE			CURDATE()			
사용자 번호	userNo	VARCHAR(10)	NOT NULL				O	tb_user

쿠폰사용내역 테이블(tb\_couponUsedHistory)

속성명	열 이름	데이터 유형	NULL 여부	제약조건	기본값	PK	FK	부모 테이블
쿠폰사용내역 번호	useHistoryNo	VARCHAR(20)	NOT NULL			O		
사용일자	useDate	DATE			CURDATE()			
쿠폰 번호	couponNo	VARCHAR(10)	NOT NULL				O	tb_coupon
결제 번호	payNo	VARCHAR(20)	NOT NULL				O	tb_payinfo

## < 게시판 관련 테이블 >

게시판 카테고리 테이블(tb\_boardCategory)

속성명	열 이름	데이터 유형	NULL 여부	제약조건	기본값	PK	FK	부모 테이블
게시판 카테고리 번호	bCategoryNo	INT	NOT NULL		AUTO_INCREMENT	O		
이름	name	VARCHAR(30)	NOT NULL					

게시판 테이블(tb\_board)

속성명	열 이름	데이터 유형	NULL 여부	제약조건	기본값	PK	FK	부모 테이블
게시판 번호	boardNo	VARCHAR(10)	NOT NULL			O		
제목	title	VARCHAR(100)						
내용	content	VARCHAR(500)						
평점	rating	FLOAT						
사용자 번호	userNo	VARCHAR(10)	NOT NULL			O	tb_user	
행사 번호	eventNo	VARCHAR(10)	NOT NULL			O	tb_event	
게시판 카테고리 번호	bCategoryNo	INT	NOT NULL			O	tb_boardCategory	
생성일자	insDate	DATE			CURDATE()			
수정일자	uptDate	DATE			CURDATE()			

댓글 테이블(tb\_reply)

속성명	열 이름	데이터 유형	NULL 여부	제약조건	기본값	PK	FK	부모 테이블
댓글 번호	replyNo	VARCHAR(10)	NOT NULL			O		
순번	cnt	INT						
내용	content	VARCHAR(64)						
게시판 번호	boardNo	VARCHAR(10)	NOT NULL			O	tb_board	
사용자 번호	userNo	VARCHAR(10)	NOT NULL			O	tb_user	
생성일자	insDate	DATE			CURDATE()			
수정일자	uptDate	DATE			CURDATE()			

# 02 API 명세서

## <유저>

유저								
Index	description	Method	endpoint	request	response	http code	authority	
1	회원가입	POST	/api/v1/user/register	id: String pw: String name: String email: String	-	[성공] 201  [실패] 1. request 형식이 잘못되었을 경우 - 400 2. 중복된 아이디일 경우 - 409 3. 이메일 인증을 받지 않았을 경우 - 401 4. 서버 내 문제 - 500	ALL	
2	아이디, 비밀번호 로그인	POST	/api/v1/user/login	id: String pw: String	accessToken: String refreshToken: String	[성공] 200  [실패] 1. request 형식이 잘못되었을 경우 - 400 2. 아이디 틀렸을 경우 - 404 3. 비밀번호 틀렸을 경우 - 400 4. 서버 내 문제 - 500	ALL	
3	아이디 중복확인	POST	/api/v1/user/check-duplicate	id: String	-	[성공] 200  [실패] 1. request 형식이 잘못되었을 경우 - 400 2. 중복된 아이디일 경우 - 409	ALL	
4	단일 유저 조회	GET	/api/v1/user/[id]	-	id: String name: String email: String useYn: char role: String	[성공] 200  [실패] 1. 해당 유저를 찾지 못했을 경우 - 404 2. 요청 권한이 없는 경우 - 401 3. 사용자 인증에 실패한 경우 - 403	ADMIN	
매일 인증								
Index	description	Method	endpoint	request	response	http code	authority	
1	인증 코드 발송	POST	/api/v1/mail/verification-code	email: String	성공 시 이메일로 인증코드 전송	[성공] 200  [실패] 1. request 형식이 잘못되었을 경우 - 400 2. 서버 내 문제 - 500	ALL	
2	인증 코드 확인	POST	/api/v1/mail/verify	code: String	-	[성공] 200  [실패] 1. request 형식이 잘못되었을 경우 - 400 2. 인증 코드가 잘못되었을 경우 - 400 4. 서버 내 문제 - 500	ALL	

# 02 API 명세서

## < 이벤트 >

이벤트							
Index	description	Method	endpoint	request	response	http code	authority
1	행사 목록 조회	GET	/api/v1/events	name: String ecategoryNo: int	eventNo: String name: String startDate: String endDate: String info: String saleInfo: String seatInfo: String timeInfo: String dayEvents: String etime: String ecategoryNo: int	[성공] 200	ALL
2	행사 상세 조회	GET	/api/v1/events/{eventNo}	-	eventNo: String name: String startDate: String endDate: String info: String saleInfo: String seatInfo: String timeInfo: String dayEvents: String etime: String ecategoryNo: int	[성공] 200	ALL
3	일자별 행사 목록 조회	GET	/api/v1/events/{eventNo}/{eventDate}	-	dayEventNo: String eventDate: Date dayT: int cnt: int deTime: String castInfo: String eventNo: String name: String startDate: String endDate: String location: String info: String saleInfo: String seatInfo: String timeInfo: String dayEvents: String ecategoryNo: int etime: String	[성공] 200	ALL
4	행사 잔여 티켓(좌석) 조회	GET	/api/v1/events/{eventNo}/{eventDate}/{dayEver}	-	dayEventNo: String seatRate: String cnt: String price: int	[성공] 200	ALL
5	행사 정보 생성	POST	/api/v1/events/register	eventNo: String name: String startDate: String endDate: String info: String saleInfo: String seatInfo: String timeInfo: String dayEvents: String etime: String ecategoryNo: int	-	[성공] 201  [실패] 1. request body 형식이 올바르지 않을 경우 -400 2. 관리자가 아닌 경우 -401	ADMIN
6	행사 정보 수정	PUT	/api/v1/events/{eventNo}	name: String location: String info: String saleInfo: String seatInfo: String etime: String ecategoryNo: int	-	[성공] 201  [실패] 1. request body 형식이 올바르지 않을 경우 -400 2. 관리자가 아닌 경우 -401 3. 수정하고자 하는 행사가 존재하지 않는 경우 -404	ADMIN

# 02 API 명세서

## <결제 및 예약>

결제, 예약							
Index	description	Method	endpoint	request	response	http code	authority
1	나의 예약 정보 상세 조회	GET	/api/v1/rex/my/info/{payNo}	-	payNo: String payment: String payDate: Date status: String price: Int fee: Int deliveryCost: Int discount: Int totalAmt: Int userNo: String insDate: Date resNo: String resDate: Date status: String ticketNo: String seatRate: String seatNum: Int dayEventNo: String eventDate: Date dayT: Int crt: Int dtlTime: String caclInfo: String eventNo: String name: String startDate: Date endDate: Date location: String info: String saleInfo: String seatInfo: String timeInfo: String dayEvents: String stime: String reservation: String	[성공] 200  [실패] 1. 로그인 상태가 아닌 경우 -401	USER
2	선택한 행사의 티켓(좌석) 인증 조회	GET	/api/v1/rex/{dayEventNo}	-	ticketNo: String seatRate: String seatNum: Int price: Int dayEventNo: String eventDate: Date dayT: Int crt: Int dtlTime: String caclInfo: String eventNo: String name: String startDate: Date endDate: Date location: String info: String saleInfo: String seatInfo: String timeInfo: String dayEvents: String stime: String reservation: String	[성공] 200  [실패] 1. 로그인 상태가 아닌 경우 -401	USER

# 02 API 명세서

2	선택한 행사의 티켓(좌석) 현황 조회	GET	/api/v1/reviews/dayEventNo/{dayEventNo}	-	ticketNo: String seatState: String seatNum: int price: int dayEventNo: String eventDate: Date dayT: int cnt: int deTime: String caTime: String eventNo: String startDate: Date endDate: Date location: String rnk: String saleInfo: String seatInfo: String timeInfo: String dayEvents: String etime: String reservation: String	[정상] 200  [실패] 1. 로그인 상태가 아닌 경우 -401	USER
3	내의 예약 정보 조회	GET	/api/v1/reviews/myInfo	-	payNo: String payment: String payDate: Date status: String price: int dayT: int deliveryCost: int discount: int totalAmt: int userNo: String insDate: Date resNo: String resDate: Date status: String ticketNo: String payNo: String ticketNo: String seatState: String seatNum: int price: int dayEventNo: String eventDate: Date dayT: int cnt: int deTime: String caTime: String eventNo: String name: String startDate: Date endDate: Date location: String rnk: String saleInfo: String seatInfo: String timeInfo: String dayEvents: String etime: String reservation: String	[정상] 200  [실패] 1. 로그인 상태가 아닌 경우 -401	USER
4	행사 결제 및 예약 정보 생성	POST	/api/v1/reviews/{dayEventNo}	-	ticketNo: String seatState: String seatNum: int price: int dayEventNo: String eventDate: Date dayT: int cnt: int deTime: String caTime: String eventNo: String name: String startDate: String endDate: String location: String rnk: String saleInfo: String seatInfo: String timeInfo: String dayEvents: String etime: String categoryNo: int etime: String reservation: String couponAppliedInfo: String price: int fee: int deliveryCost: int discount: int totalAmt: int payment: String couponNo: String	[정상] 201  [실패] 1. 결제된 요청 -400 2. 결제된 상태가 아닌 경우 -401 3. 이미 예약된 좌석을 예약할 경우 -403	USER
5	예약 취소	PUT	/api/v1/reviews/myInfo/{payNo}	-	-	[정상] 201  [실패] 1. request body 형식이 올바르지 않을 경우 -400 2. 결제된 상태가 아닌 경우 -401 3. 수령하고자 하는 결제가 존재하지 않는 경우 -404	USER

# 02 API 명세서

## < 게시글 >

게시글								
Index	description	Method	endpoint	request	response	http code	authority	
1	게시판 전체 조회	GET	/api/v1/boards	-	boardNo: String title: String content: String rating: int insDate: Date uptDate: Date boardCategoryNo: int boardCategoryName: String userName: String replyNo: String cnt: int content: String email: String	[성공] 200	ALL	
2	게시판 단일 조회	GET	/api/v1/boards/{boardNo}?boardCategoryNo	-	boardNo: String title: String content: String rating: int insDate: Date uptDate: Date boardCategoryNo: int boardCategoryName: String userName: String replyNo: String cnt: int content: String email: String	[성공] 200	ALL	
3	게시글 작성	POST	/api/v1/boards	title: String content: String eventNo: String bCategoryNo: int	-	[성공] 201  [실패] 1. 잘못된 요청 -400	USER, ADMIN	
4	게시글 수정	PUT	/api/v1/boards/{boardNo}	title: String content: String	-	[성공] 200  [실패] 1. request body 형식이 올바르지 않을 경우 -400 2. 본인의 작성한 게시물이 아닌 경우 -401 3. 수정하고자 하는 결제가 존재하지 않는 경우 -404	USER, ADMIN	
5	게시글 삭제	DELETE	/api/v1/boards/{boardNo}	-	-	[성공] 200  [실패] 1. 본인이 작성한 게시물이 아닌 경우 -401 2. 수정하고자 하는 게시판이 존재하지 않는 경우 -404	USER, ADMIN	

게시글 카테고리								
Index	description	Method	endpoint	request	response	http code	authority	
1	카테고리 전부 조회	GET	/api/v1/board-categories	-	data: [ bCategoryNo: int, name: String ]	[성공] 200	ALL	

# 02 API 명세서

## < 댓글, 쿠폰 >

댓글								
Index	description	Method	endpoint	request	response	http code	authority	
1	댓글 작성	POST	/api/v1/boards/{boardNo}/reply	content: String	cnt: int content: String insDate: String uptDate: String boardNo: String userNo: String	[성공] 201  [실패] 1. request body 형식이 올바르지 않은 경우 -400 2. 댓글 작성 권한이 없을 경우 -401 3. 댓글 작성 대상 게시글이 존재하지 않을 경우 -404 4. 서버 내부 에러 -500	USER, ADMIN	
2	댓글 삭제	DELETE	/api/v1/boards/{boardNo}/reply/{replyNo}	-	-	[성공] 201  [실패] 1. request body 형식이 올바르지 않은 경우 -400 2. 댓글 삭제 권한이 없을 경우 -401 3. 댓글 작성 대상 게시글이 존재하지 않을 경우 -404 4. 서버 내부 에러 -500	USER, ADMIN	
3	댓글 수정	PUT	/api/v1/boards/{boardNo}/reply/{replyNo}	content: String	replyNo: String content: String cnt: int insDate: String uptDate: String boardNo: String	[성공] 201  [실패] 1. request body 형식이 올바르지 않은 경우 -400 2. 댓글 수정 권한이 없을 경우 -401 3. 댓글 작성 대상 게시글이 존재하지 않을 경우 -404 4. 서버 내부 에러 -500	USER	
쿠폰								
Index	description	Method	endpoint	request	response	http code	authority	
1	쿠폰 번호로 쿠폰 조회	GET	/api/v1/coupons/callCouponNo/{couponNo}	-	couponNo: String code: String name: String dcPercent: int insDate: Date expireDate: Date userNo: String	[성공] 200  [실패] 1. 쿠폰을 찾을 수 없는 경우 -404	ADMIN	
2	사용자 번호로 쿠폰 목록 조회	GET	/api/v1/coupons/callUserNo/{userNo}	-	couponNo: String code: String name: String dcPercent: int insDate: Date expireDate: Date userNo: String	[성공] 200  [실패] 1. 쿠폰을 찾을 수 없는 경우 -404	ALL	
3	쿠폰 생성	POST	/api/v1/coupons/create	name: String dcPercent: int userNo: String expireDate: Date	-	[성공] 201  [실패] 1. 잘못된 요청 - 400	ADMIN	
4	쿠폰 업데이트	PUT	/api/v1/coupons/update/{couponNo}	name: String dcPercent: int expireDate: Date	-	[성공] 200  [실패] 1. 요청 형식이 올바르지 않은 경우 -400 2. 쿠폰을 찾을 수 없는 경우 -404	ADMIN	
5	쿠폰 삭제	DELETE	/api/v1/coupons/delete/{couponNo}	-	-	[성공] 200  [실패] 1. 쿠폰을 찾을 수 없는 경우 -404	USER	

# 02 API 명세서

## < 알림 >

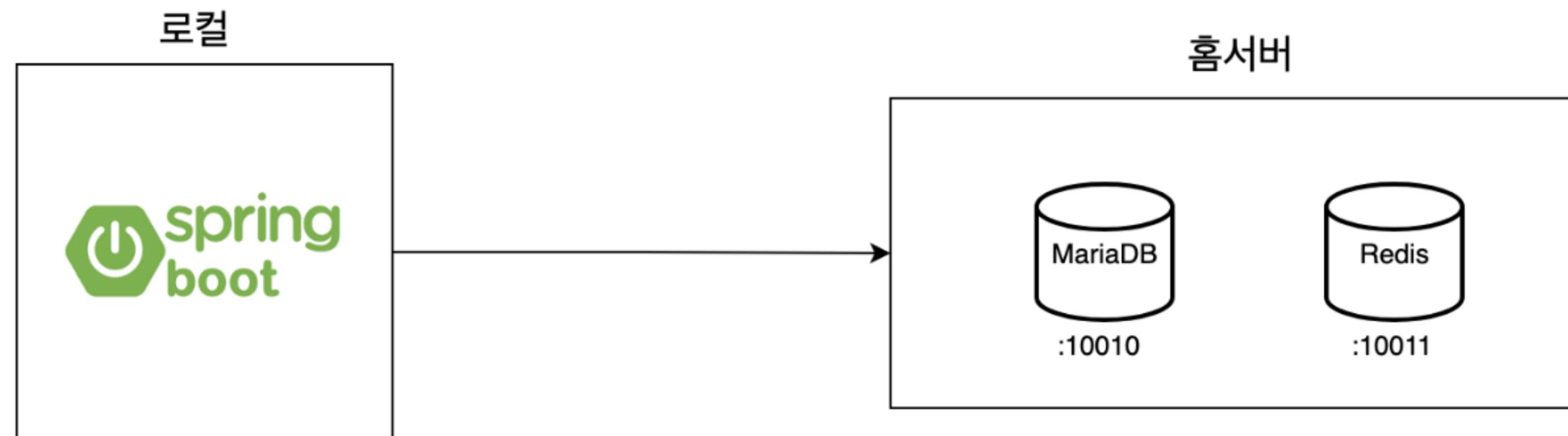
알림								
index	description	Method	url endpoint	request	response	http code	autherity	
1	알림 정보 상세 조회	GET	/api/v1/notices/[noticeNo]	-	noticeNo: String message: String notiDate: Date notiStatus: String payNo: String payment: String notiDate: Date status: String price: int fee: int deliveryCost: int discount: int totalAmount: int ticketNo: String invDate: Date notiNo: String notiDate: Date status: String ticketNo: String payNo: String seatRate: String seatNum: int price: int dayEventNo: String eventDate: Date seatT: int cnt: int seatTime: String cardInfo: String eventNo: String name: String startDate: Date endDate: Date info: String seatInfo: String seatInfo: String time: String reservation: String	[성공] 200  [실패] 도록인 상태가 아닌 경우 -403	USER, ADMIN	
2	알림 정보 조회	GET	/api/v1/notices/	-	noticeNo: String message: String notiDate: Date notiStatus: String payNo: String payment: String notiDate: Date status: String price: int fee: int deliveryCost: int discount: int totalAmount: int ticketNo: String invDate: Date notiNo: String notiDate: Date status: String ticketNo: String payNo: String seatRate: String seatNum: int price: int dayEventNo: String eventDate: Date seatT: int cnt: int seatTime: String cardInfo: String eventNo: String name: String startDate: Date endDate: Date info: String seatInfo: String seatInfo: String time: String reservation: String	[성공] 200  [실패] 도록인 상태가 아닌 경우 -403	USER, ADMIN	
3	알림 생성	POST	/api/v1/notices/	noticeNo: String message: String payNo: String	-	[성공] 200	ADMIN	
4	알림 삭제	PUT	/api/v1/notices/[noticeNo]	-	-	[성공] 200  [실패] 도록인 알림 정보가 아닌 경우 -403	USER, ADMIN	

03

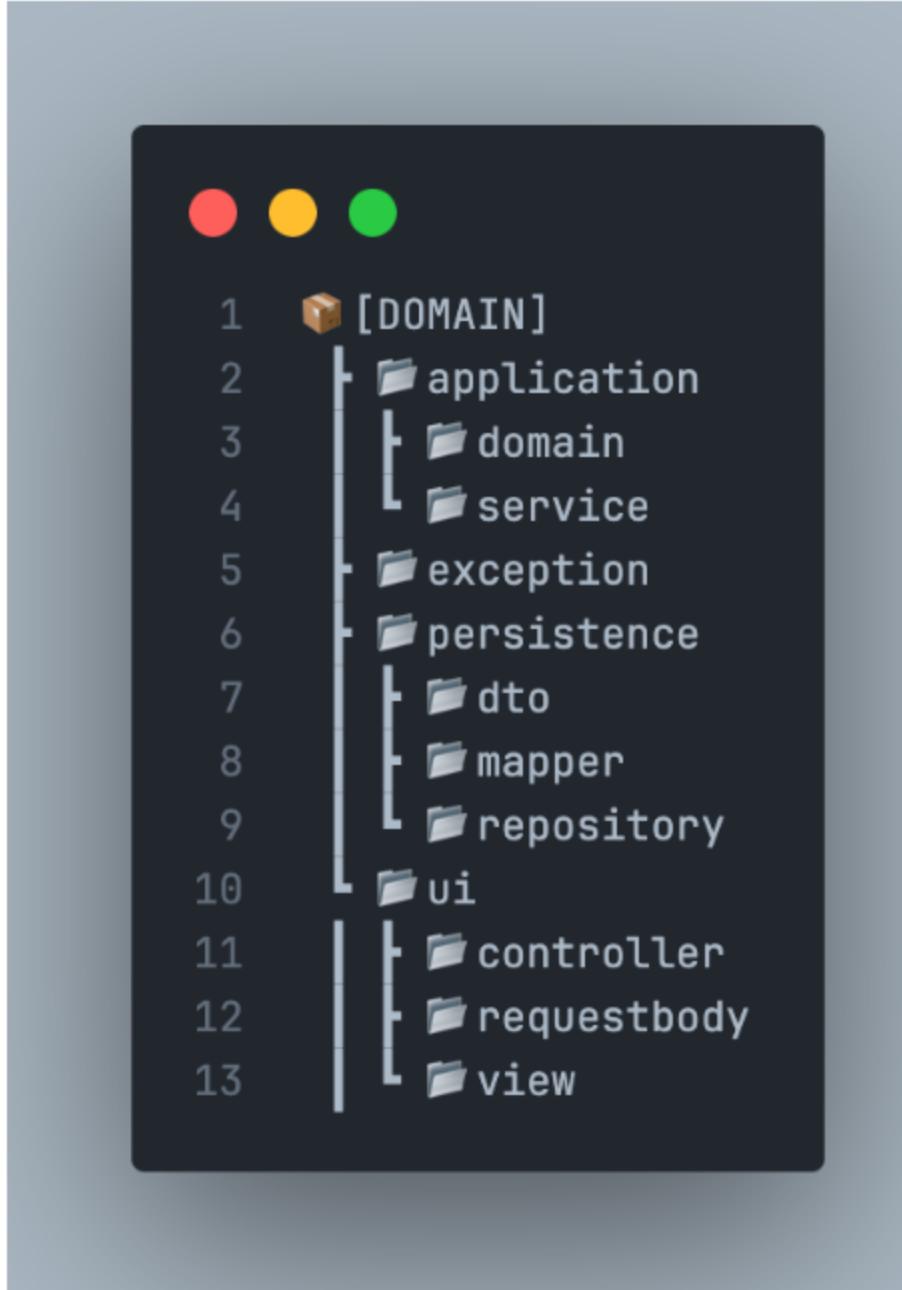
## 결과

- POINT 01 주요 기능 시연
- POINT 02 개선 사항

# 03 프로젝트 아키텍처



# 03 프로젝트 구조



- **application**
  - 도메인의 주요 VO와 비즈니스 로직
- **exception**
  - 도메인의 서비스에서 발생 가능한 예외
    - Enum으로 예외 메세지 정의
- **persistence**
  - 데이터베이스 관련 로직
- **ui**
  - 비즈니스 로직과 상관없는 controller, view, request

# 03 프로젝트 Global 설정

```
1 common
2   ↳ advice
3     ↳ TicketLinkControllerAdvice.java
4   ↳ doc
5     ↳ SwaggerConfig.java
6   ↳ exception
7     ↳ CommonMessageType.java
8     ↳ MessageType.java
9     ↳ TicketLinkException.java
10  ↳ security
11    ↳ config
12      ↳ TicketLinkSecurityConfig.java
13    ↳ exception
14      ↳ JwtAccessDeniedHandler.java
15      ↳ JwtAuthenticationEntryPoint.java
16    ↳ filter
17      ↳ JwtAuthenticationFilter.java
18    ↳ provider
19      ↳ JwtAuthenticationProvider.java
20  ↳ view
21    ↳ ApiErrorResponse.java
22    ↳ ApiResponseView.java
```

- **advice**
  - [프로젝트내 모든 컨트롤러에 적용할 RestControllerAdvice](#)
- **doc**
  - 스웨거 설정
- **exception**
  - 전역 에러 및 커스텀 에러
- **security**
  - 스프링 시큐리티 설정
- **view**
  - 일관된 response를 위한 View, ErrorView

# 03 공통 예외 처리

## 서비스 공통 컨트롤러 어드바이스

```
***  
@Slf4j  ± yongun2  
@RestControllerAdvice  
public class TicketLinkControllerAdvice extends ResponseEntityExceptionHandler {  
  
    @ExceptionHandler(ClientAbortException.class)  ± yongun2  
    public ResponseEntity<?> clientAbortException() {  
        log.error("clientAbortException");  
        return new ResponseEntity<>(new ApiErrorView(Collections.singletonList(CommonMessageType.INTERNAL_SERVER_ERROR)),  
            HttpStatus.INTERNAL_SERVER_ERROR);  
    }  
  
    @ExceptionHandler(TicketLinkException.class)  ± yongun2  
    public ResponseEntity<?> onMessageException(TicketLinkException ex) {  
        return new ResponseEntity<>(new ApiErrorView(ex), ex.getStatus());  
    }  
  
    @ExceptionHandler({SQLException.class, DataAccessException.class})  ± yongun2  
    public ResponseEntity<Object> handleSQLException(SQLException ex) {  
        log.warn("SQL Exception >>> " + ex.getMessage());  
        return new ResponseEntity<>(  
            new ApiErrorView(CommonMessageType.SQL_EXCEPTION_ERROR, ex.getMessage()),  
            HttpStatus.INTERNAL_SERVER_ERROR  
        );  
    }  
}
```

## 도메인 별 예외 메세지

```
***  
package com.beyond.ticketLink.event.exception;  
  
import ...  
  
@Getter 4 usages  ± mabem95  
@RequiredArgsConstructor  
public enum EventMessageType implements MessageType {  
  
    EVENT_OPERATION_UNAUTHORIZED( message: "You don't have permission to manipulate event.", HttpStatus.UNAUTHORIZED),  
    EVENT_NOT_FOUND( message: "No Events were found for your request.", HttpStatus.NOT_FOUND)  
};  
  
***  
package com.beyond.ticketLink.board.exception;  
  
import ...  
  
@Getter 7 usages  ± yongun2  
@RequiredArgsConstructor  
public enum BoardMessageType implements MessageType {  
  
    // board  
    BOARD_OPERATION_UNAUTHORIZED( message: "You don't have permission to manipulate board.", HttpStatus.BAD_REQUEST),  
    BOARD_NOT_FOUND( message: "No Boards were found for your request.", HttpStatus.NOT_FOUND)  
;  
  
    private final String message;  
    private final HttpStatus status;  
};
```

# 03 공통 예외 처리

## 예외 전용 View(ApiErrorView.class)

### 일관된 예외 메세지

```
***  
@Getter 58 usages  ± yongun2  
@ToString  
public class ApiErrorView {  
    private final List<Error> errors;  
  
    public ApiErrorView(List<MessageType> messageTypes) { 1 usage  ± yongun2  
        this.errors = messageTypes.stream().map(Error::errorWithMessageType).collect(Collectors.toList());  
    }  
  
    public ApiErrorView(TicketLinkException exception) { 1 usage  ± yongun2  
        this.errors = Collections.singletonList(Error.errorWithException(exception));  
    }  
  
    public ApiErrorView(CommonMessageType messageType, String message) { 9 usages  ± yongun2  
        this.errors = Collections.singletonList(Error.errorWithTypeAndMessage(messageType, message));  
    }  
}
```

```
{  
    "errors": [  
        {  
            "errorType": "BAD_REQUEST",  
            "errorMessage": "Required request parameter 'bCategoryNo' for method parameter type Integer is  
                not present"  
        }  
    ]  
}
```

# 03 API Security 설정

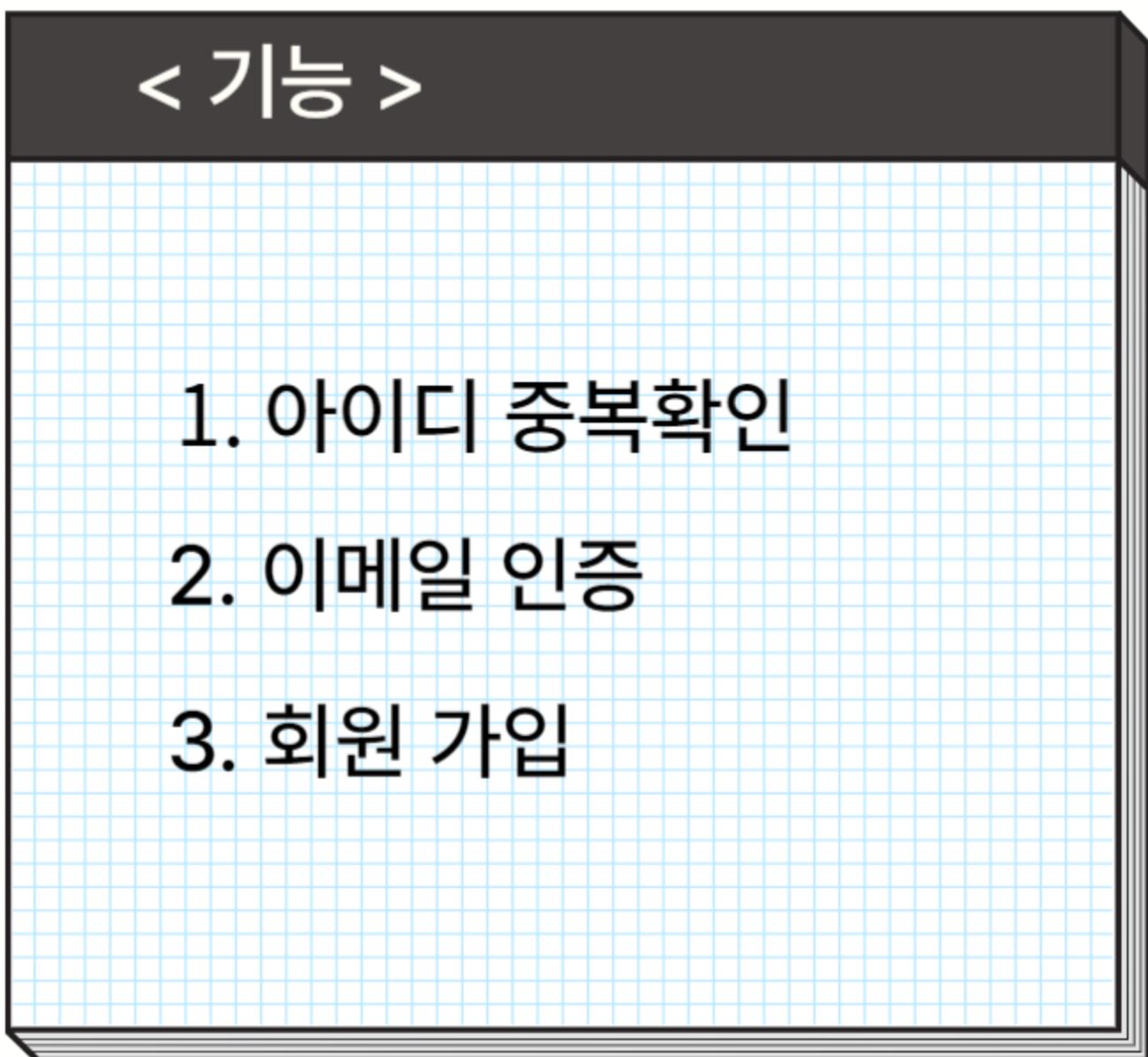
```
...  
http.authorizeHttpRequests((registry -> {  
    registry.requestMatchers(  
        "/api/v1/user/register",  
        "/api/v1/user/check-duplicate",  
        "/api/v1/user/login",  
        "/api/v1/mail/**",  
        "/api/v1/board-categories"  
    ).permitAll();  
  
    // swagger config  
    registry.requestMatchers(HttpMethod.GET, "/swagger-ui/**", "/v3/api-docs/**").permitAll();  
})
```

인증이 필요 없는 요청

```
...  
// event role config  
registry.requestMatchers(HttpMethod.GET, "/api/v1/events/**").permitAll();  
registry.requestMatchers("/api/v1/event/**").hasRole(ADMIN);  
  
// reservation role config  
registry.requestMatchers("/api/v1/res/**").hasAnyRole(...roles: USER, ADMIN);  
registry.requestMatchers(HttpMethod.POST, "/api/v1/res/*").hasRole(USER);  
  
// coupon config  
registry.requestMatchers(HttpMethod.POST, "/api/v1/coupons").hasRole(ADMIN);  
registry.requestMatchers(HttpMethod.PUT, "/api/v1/coupons/**").hasRole(ADMIN);  
registry.requestMatchers(HttpMethod.DELETE, "/api/v1/coupons/**").hasRole(ADMIN);  
})
```

인증과 권한이 필요한 요청

## 03 유저 기능(회원가입)



## < 1. 아이디 중복확인 >

```
POST /user/check-duplicate
{
  "id": "string"
}
```

```
@Override 3 usages ± yongun2
public void checkIdDuplicated(String id) {

    if (userRepository.findUserById(id).isPresent()) {
        throw new TicketLinkException(UserMessageType.DUPLICATE_USER_ID);
    }
}
```

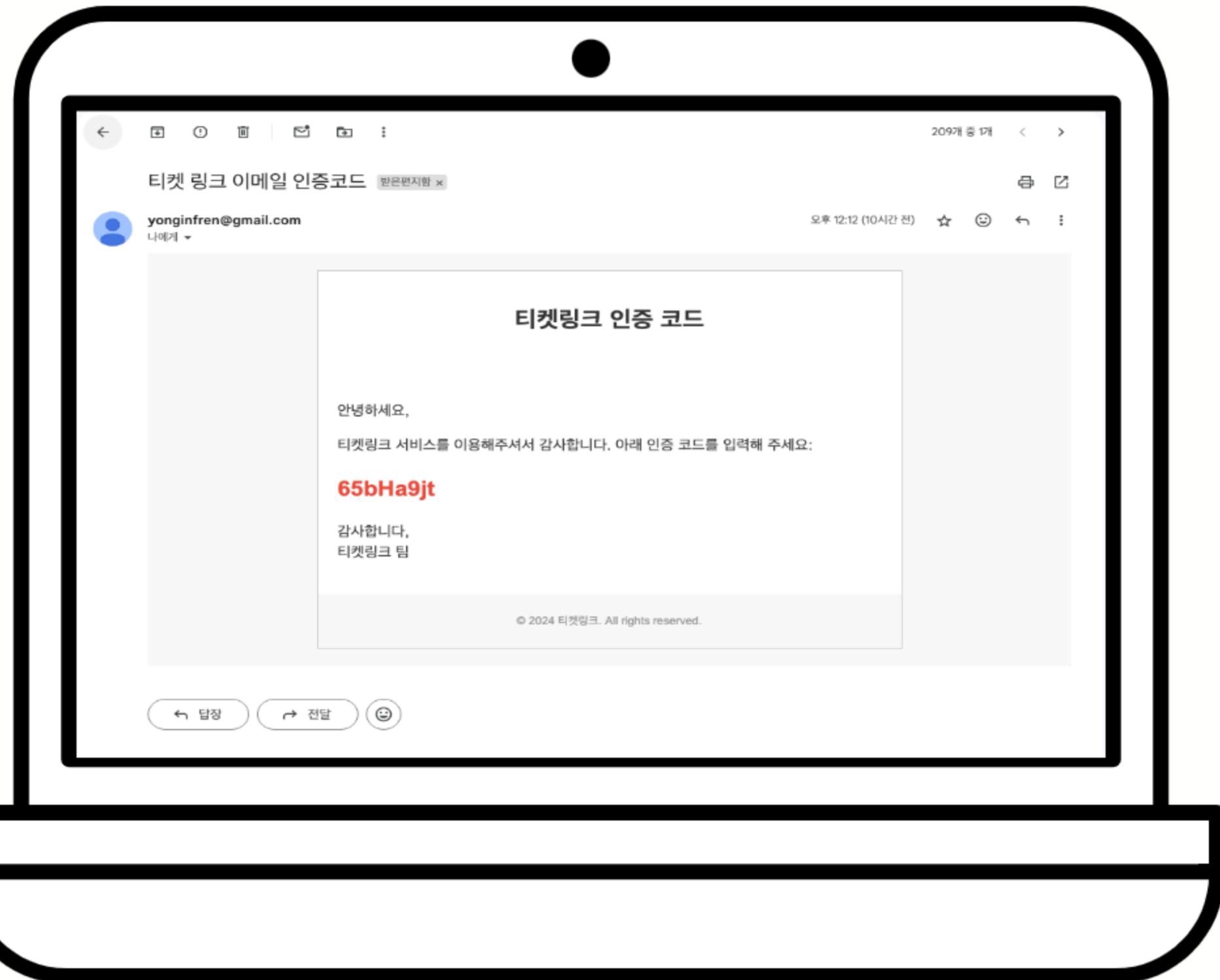
## < 2.1 이메일 인증 - 인증번호 메일 요청 >

POST /mail/verification-code

```
{  
  "email": "string"  
}
```

```
@Override 2 usages ▲ yongun2  
@Transactional  
public void sendMail(EmailVerificationRequest request) {  
  
    final String SENDER_EMAIL_ADDRESS = "yonginfren@gmail.com";  
  
    log.info("sender = {}", SENDER_EMAIL_ADDRESS);  
    // html 형식으로 내용을 첨부하기 위한 객체  
    MimeMessage message = mailSender.createMimeMessage();  
  
    try {  
        // 송신자 메일 설정  
        message.setFrom(new InternetAddress(SENDER_EMAIL_ADDRESS));  
        // 인증 코드 생성  
        String verificationCode = generateVerificationCode();  
  
        // 현재 이메일과 인증코드 레디스에 저장  
        verificationCodeRepository.save(  
            VerificationCode.builder()  
                .email(request.email())  
                .code(verificationCode)  
                .ttl(VERIFICATION_EXPIRED_TIME)  
                .build()  
        );  
  
        // 메일 내용 html로 생성  
        String htmlContent = generateEmailContent(verificationCode);  
        // 수신자 이메일 주소 설정  
        message.setRecipients(MimeMessage.RecipientType.TO, request.email());  
        // 메일 제목 설정  
        message.setSubject(EMAIL SUBJECT);  
        // 메일 내용 설정  
        message.setContent(htmlContent, type: "text/html; charset=utf-8");  
  
        // 메일 전송  
        mailSender.send(message);  
    } catch (MessagingException e) {  
        throw new TicketLinkException(CommonMessageType.INTERNAL_SERVER_ERROR);  
    }  
}
```

## < 2.1 이메일 인증 - 인증메일 >



## < 2.2 이메일 인증 - 인증번호 확인 >

POST /mail/verify

```
{  
  "verificationCode": "string"  
}
```

```
● ● ●  
@Override 3 usages ▲ yongun2  
@Transactional  
public void verifyEmail(EmailVerificationCodeRequest request) {  
    String verificationCode = request.verificationCode();  
  
    VerificationCode verified = verificationCodeRepository.findByCode(verificationCode)  
        .orElseThrow(() -> new TicketLinkException(MailMessageType.VERIFICATION_CODE_INVALID));  
  
    // 이메일 인증 성공 했을 경우  
    // 인증된 이메일 저장소에 저장  
    verifiedEmailRepository.save(VerifiedEmail.builder()  
        .email(verified.getEmail())  
        .build());  
  
    // 인증 코드 삭제  
    verificationCodeRepository.delete(verified);  
}
```

## < 3. 회원가입 >

POST /user/register

```
{  
    "id": "string",  
    "pw": "string",  
    "name": "string",  
    "email": "string"  
}
```

```
@Transactional  
public void register(UserCreateRequest request) {  
  
    final char ENABLE_USER = 'Y';  
    final String ROLE_USER = "일반사용자";  
  
    // 아이디가 중복 되었을 경우 409_Error throw  
    if (userRepository.findUserById(request.id()).isPresent()) {  
        throw new TicketLinkException(UserMessageType.DUPLICATE_USER_ID);  
    }  
  
    // db에 user role 설정이 잘못 되어 있을 경우 500_Error throw  
    UserRole userRole = userRoleRepository.findByRoleName(ROLE_USER)  
        .orElseThrow(() -> new TicketLinkException(UserMessageType.USER_ROLE_NOT_FOUND));  
  
    // 이메일이 인증되지 않았을 경우 401_Error throw  
    VerifiedEmail verifiedEmail = verifiedEmailRepository.findById(request.email())  
        .orElseThrow(() -> new TicketLinkException(MailMessageType.EMAIL_UNAUTHORIZED));  
  
    // 패스워드 암호화  
    String encryptedPassword = encryptPassword(request);  
  
    // 유저 저장  
    userRepository.save(  
        new UserCreateDto(  
            request.id(),  
            encryptedPassword,  
            request.name(),  
            request.email(),  
            ENABLE_USER,  
            userRole.getId()  
        )  
    );  
  
    // 유저 저장 이후 이메일 인증 완료 정보 삭제  
    verifiedEmailRepository.delete(verifiedEmail);  
}
```

# 03 유저 기능(로그인)

POST /user/login

```
{  
  "id": "string",  
  "pw": "string"  
}
```

```
@Override 6 usages ± yongun2  
public FindJwtResult login(UserLoginRequest request) {  
  
    // 로그인 요청한 유저가 존재하지 않을 경우 404_Error throw  
    TicketLinkUserDetails loginUser = userRepository.findById(request.id())  
        .orElseThrow(() → new TicketLinkException(UserMessageType.USER_NOT_FOUND));  
  
    // 비밀번호 검증  
    if (invalidPassword(request, loginUser)) {  
        throw new TicketLinkException(UserMessageType.INVALID_PASSWORD);  
    }  
  
    // JwtToken 생성  
    String accessToken = jwtUtil.createAccessToken(loginUser);  
    String refreshToken = jwtUtil.createRefreshToken(loginUser);  
  
    return FindJwtResult.findByAll(accessToken, refreshToken);  
}
```

## 03 유저 기능(로그아웃)

POST /user/logout  
-H "Authorization Bearer accessToken"

```
@Override
@Transactional
public void logout(LogoutCommand command) {
    String accessToken = command.getAccessToken();

    // 현재 로그아웃 하는 accessToken 만료 토큰으로 등록
    expiredAccessTokenRepository.save(
        ExpiredAccessToken.builder()
            .accessToken(accessToken)
            // 8분으로 지정
            .ttl(480L)
            .build()
    );
}
```

## 03 사용자 인증(JWT)

- 로그인 JWT 토큰 발급
  - accessToken
  - refreshToken(HttpOnly Cookie)
  
- 인증이 필요한 요청시
  - 헤더에 accessToken 포함 필요
  - JwtAuthenticationFilter, JwtAuthenticationProvider를 통한 사용자 인증처리 진행

# 03 사용자 인증(JWT)

- SecurityConfig
  - sessionManagement 정책 stateless 설정
  - Jwt 커스텀 filter 및 provider 등록

```
http.httpBasic(AbstractHttpConfigurer::disable)
    .formLogin(AbstractHttpConfigurer::disable)
    .csrf(AbstractHttpConfigurer::disable)
    .sessionManagement(session → {
        session.sessionCreationPolicy(SessionCreationPolicy.STATELESS);
    })
    .anonymous(AbstractHttpConfigurer::disable)
    .addFilterBefore(
        jwtAuthenticationFilter(jwtAuthenticationProvider),
        UsernamePasswordAuthenticationFilter.class
    );
```

# 03 사용자 인증(JWT)

```
●●●
@Slf4j 3 usages ▾ yongun2
@RequiredArgsConstructor
public class JwtAuthenticationFilter extends OncePerRequestFilter {

    private final JwtAuthenticationProvider jwtAuthenticationProvider;

    @Override no usages ▾ yongun2
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)
        throws ServletException, IOException {
        String authorizationHeader = request.getHeader("Authorization");

        if (authorizationHeader != null && authorizationHeader.startsWith("Bearer ")) {
            log.info("JwtAuthenticationFilter");
            String accessToken = authorizationHeader.substring(7);
            Authentication authenticate = jwtAuthenticationProvider.authenticate(
                new UsernamePasswordAuthenticationToken(accessToken, ""))
            );
            SecurityContextHolder.getContext().setAuthentication(authenticate);
        }
        filterChain.doFilter(request, response);
    }
}
```

# 03 사용자 인증(JWT)

```
●●●
@Override  ✎ yongun2
public Authentication authenticate(Authentication authentication) throws AuthenticationException {

    String accessToken = (String) authentication.getPrincipal();

    if (jwtUtil.isExpired(accessToken) || bannedToken(accessToken)) {
        throw new TicketLinkException(JwtMessageType.TOKEN_EXPIRED);
    }

    String userNo = jwtUtil.getUserNo(accessToken);
    String role = jwtUtil.getRole(accessToken);

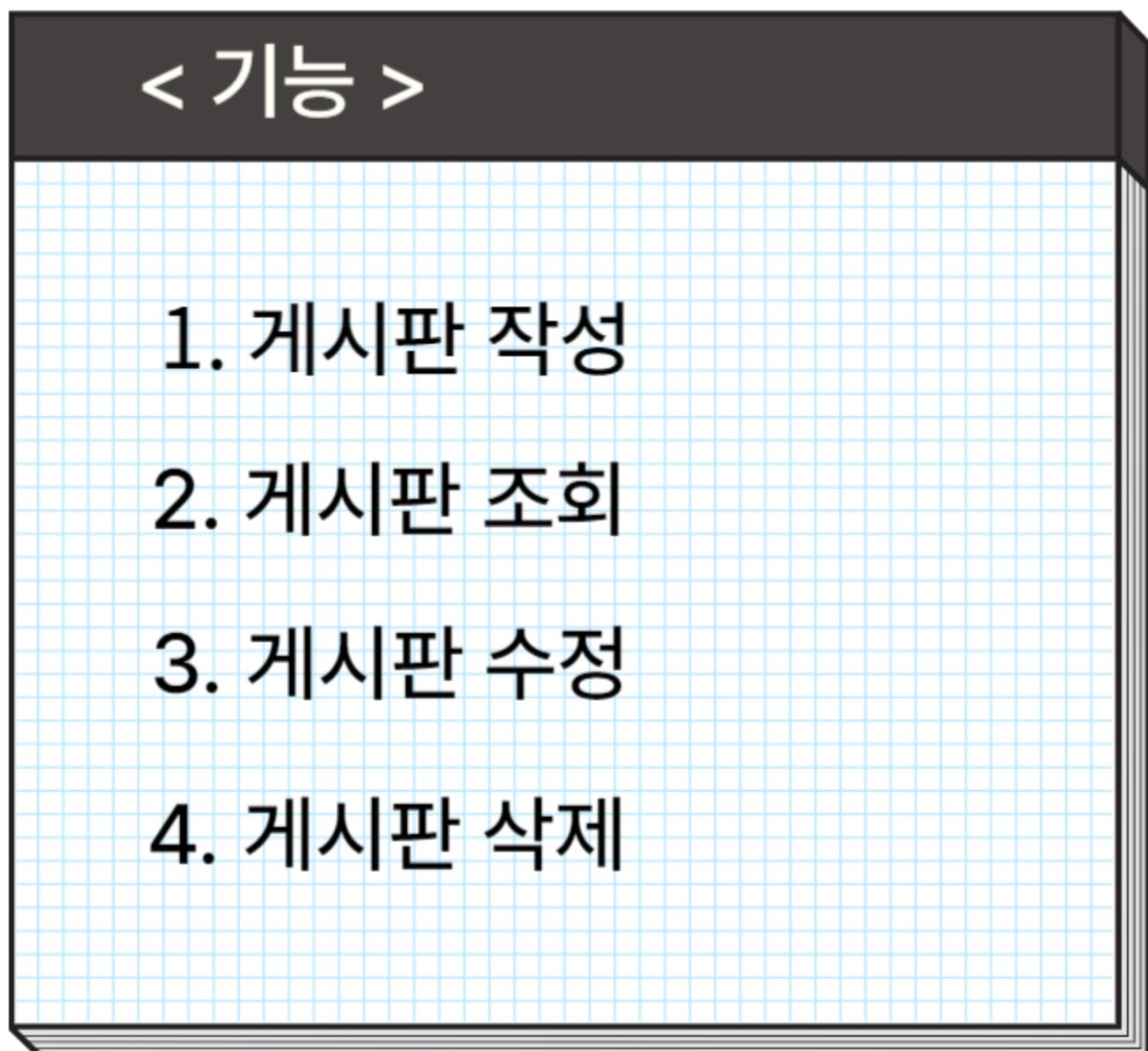
    return new UsernamePasswordAuthenticationToken(
        userNo,
        credentials: null,
        List.of(new SimpleGrantedAuthority(role: "ROLE_" + role))
    );
}
```

# 03 사용자 인증(JWT)

- **@AuthenticationPrincipal 어노테이션 사용**
  - 인증된 사용자 정보 불러와 사용

```
•••  
@PutMapping("boards/{boardNo}")  
public ResponseEntity<ApiResponseView<BoardView>> modifyBoard(  
    @PathVariable String boardNo,  
    @AuthenticationPrincipal String userNo,  
    @RequestBody @Validated BoardUpdateRequest request  
) {  
  
    BoardUpdateCommand updateCommand = BoardUpdateCommand.builder()  
        .boardNo(boardNo)  
        .userNo(userNo)  
        .title(request.title())  
        .rating(request.rating())  
        .content(request.content())  
        .build();  
  
    FindBoardResult result = boardService.modifyBoard(updateCommand);  
  
    return ResponseEntity.status(HttpStatus.OK)  
        .body(new ApiResponseView<>(new BoardView(result)));  
}
```

## 03 게시판 기능



## < 1. 게시글 작성 >

```
POST /boards  
-H "Authorization Bearer accessToken"  
{  
    "title": "string",  
    "content": "string",  
    "rating": 0,  
    "eventNo": "string",  
    "bCategoryNo": 0  
}
```

```
@Override 4 usages ▾ yongun2 +1  
@Transactional  
public void createBoard(BoardCreateRequest request, String userNo) {  
  
    Event event = validateEvent(request);  
  
    String boardNo = autoNoRepository.getData(tableName: "tb_board");  
  
    boardRepository.save(  
        new BoardCreateDto(  
            boardNo,  
            request.title(),  
            request.content(),  
            request.rating(),  
            NOW,  
            NOW,  
            userNo,  
            event.getEventNo(),  
            request.bCategoryNo()  
    );  
}
```

## < 2. 게시글 조회 - 전체 조회 >

GET /boards?bCategoryNo&page

```
{  
    "boardNo": "string",  
    "title": "string",  
    "content": "string",  
    "rating": 0,  
    "insDate": "2024-07-30",  
    "uptDate": "2024-07-30",  
    "boardCategoryNo": 0,  
    "boardCategoryName": "string",  
    "username": "string",  
    "replies": [  
        {  
            "replyNo": "string",  
            "cnt": 0,  
            "content": "string",  
            "insDate": "2024-07-30T14:21:10.104Z",  
            "uptDate": "2024-07-30T14:21:10.104Z",  
            "username": "string",  
            "email": "string"  
        }  
    ]  
}
```

```
● ● ●  
@Override 1 usage  ± yongun2  
public List<FindBoardResult> getAllBoard(BoardFindQuery query) {  
  
    int limit = query.getRow();  
    int offset = (query.getPage() - 1) * limit;  
  
    RowBounds rowBounds = new RowBounds(offset, limit);  
  
    List<Board> boards = boardRepository.selectBoardAll(query, rowBounds);  
  
    return boards.stream()  
        .map(FindBoardResult::findByBoard)  
        .toList();  
}
```

## < 2. 게시글 조회 - 전체 조회 쿼리 >

```
<select id="selectBoardAll" parameterType="BoardFindQuery" resultMap="boardWithAllResultMap">
    SELECT tb_b.boardNo,
           tb_b.title,
           tb_b.content,
           tb_b.rating,
           tb_b.insDate,
           tb_b.updDate,
           tb_u.userNo      AS 'u_userNo',
           tb_u.id          AS 'u_id',
           tb_u.name        AS 'u_name',
           tb_u.email       AS 'u_email',
           tb_u.useYn       AS 'u_useYn',
           tb_u.roleNo      AS 'u_ur_roleNo',
           tb_r.name        AS 'u_ur_name',
           tb_e.eventNo     AS 'e_eventNo',
           tb_e.name        AS 'e_name',
           tb_e.startDate   AS 'e_startDate',
           tb_e.endDate     AS 'e_endDate',
           tb_e.eTime       AS 'e_eTime',
           tb_e.location    AS 'e_location',
           tb_e.info        AS 'e_info',
           tb_e.saleInfo    AS 'e_saleInfo',
           tb_e.seatInfo   AS 'e_seatInfo',
           tb_e.eCategoryNo AS 'e_eCategoryNo',
           tb_e.insDate     AS 'e_insDate',
           tb_e.updDate     AS 'e_updDate',
           tb_bc.bCategoryNo AS 'bc_bCategoryNo',
           tb_bc.name       AS 'bc_name',
           tb_br.replyNo    AS 'br_replyNo',
           tb_br.content    AS 'br_content',
           tb_br.cnt        AS 'br_cnt',
           tb_br.updDate    AS 'br_updDate',
           reply_u.id       AS 'br_u_id',
           reply_u.name     AS 'br_u_name',
           reply_u.email    AS 'br_u_email',
           reply_u.useYn    AS 'br_u_useYn'
    FROM tb_board tb_b
        INNER JOIN tb_user tb_u ON tb_b.userNo = tb_u.userNo
        INNER JOIN tb_role tb_r ON tb_u.roleNo = tb_r.roleNo
        LEFT JOIN tb_event tb_e ON tb_b.eventNo = tb_e.eventNo
        LEFT JOIN tb_boardCategory tb_bc ON tb_b.bCategoryNo = tb_bc.bCategoryNo
        LEFT JOIN tb_reply tb_br ON tb_b.boardNo = tb_br.boardNo
        LEFT JOIN tb_user reply_u ON tb_br.userNo = reply_u.userNo
    <where>
        <if test="query != null">
            <if test="query.bCategoryNo != null">
                AND tb_bc.bCategoryNo = #{query.bCategoryNo}
            </if>
        </if>
    </where>
</select>
```

```
<where>
    <if test="query != null">
        <if test="query.bCategoryNo != null">
            AND tb_bc.bCategoryNo = #{query.bCategoryNo}
        </if>
    </if>
</where>
```

### < 3. 게시글 수정 >

```
POST /boards/{boardNo}
-H "Authorization Bearer accessToken"
{
  "title": "string",
  "content": "string",
  "rating": 0
}
```

```
@Override 4 usages ± yongun2
public FindBoardResult modifyBoard(BoardUpdateCommand command) {

    Board board = retrieveBoard(command.getBoardNo());

    if (hasNoOperationAuthority(command.getUserNo(), board)) {
        throw new TicketLinkException(BoardMessageType.BOARD_OPERATION_UNAUTHORIZED);
    }

    BoardUpdateDto boardUpdateDto = new BoardUpdateDto(
        board.getBoardNo(),
        command.getContent(),
        command.getTitle(),
        command.getRating(),
        NOW
    );

    boardRepository.updateBoard(boardUpdateDto);

    return FindBoardResult.findByBoardUpdateDto(boardUpdateDto);
}
```

## < 4. 게시글 삭제 >

DELETE /boards/{boardNo}  
-H "Authorization Bearer accessToken"

```
@Override 1 usage ȏ yongun2
public void deleteBoard(BoardDeleteCommand command) {
    Board board = retrieveBoard(command.getBoardNo());

    if (hasNoOperationAuthority(command.getUserNo(), board)) {
        throw new TicketLinkException(BoardMessageType.BOARD_OPERATION_UNAUTHORIZED);
    }

    boardRepository.deleteBoard(command.getBoardNo());
}
```

## < 5. 내부 함수 >

```
private Board retrieveBoard(String boardNo) { 3 usages ▲ yongun2
    return boardRepository.selectBoardByBoardNo(boardNo)
        .orElseThrow(() -> new TicketLinkException(BoardMessageType.BOARD_NOT_FOUND));
}
```

게시글 가져오기

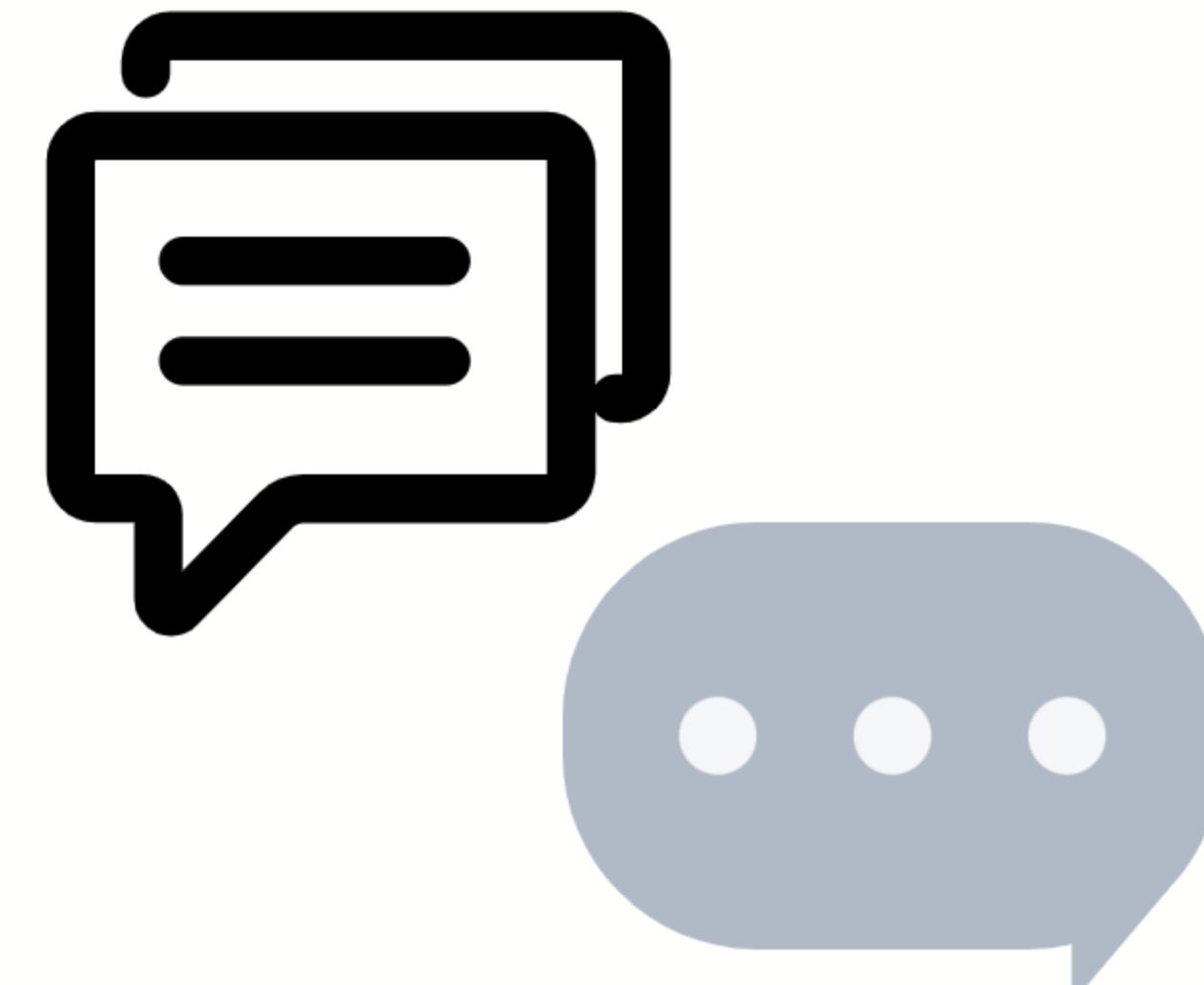
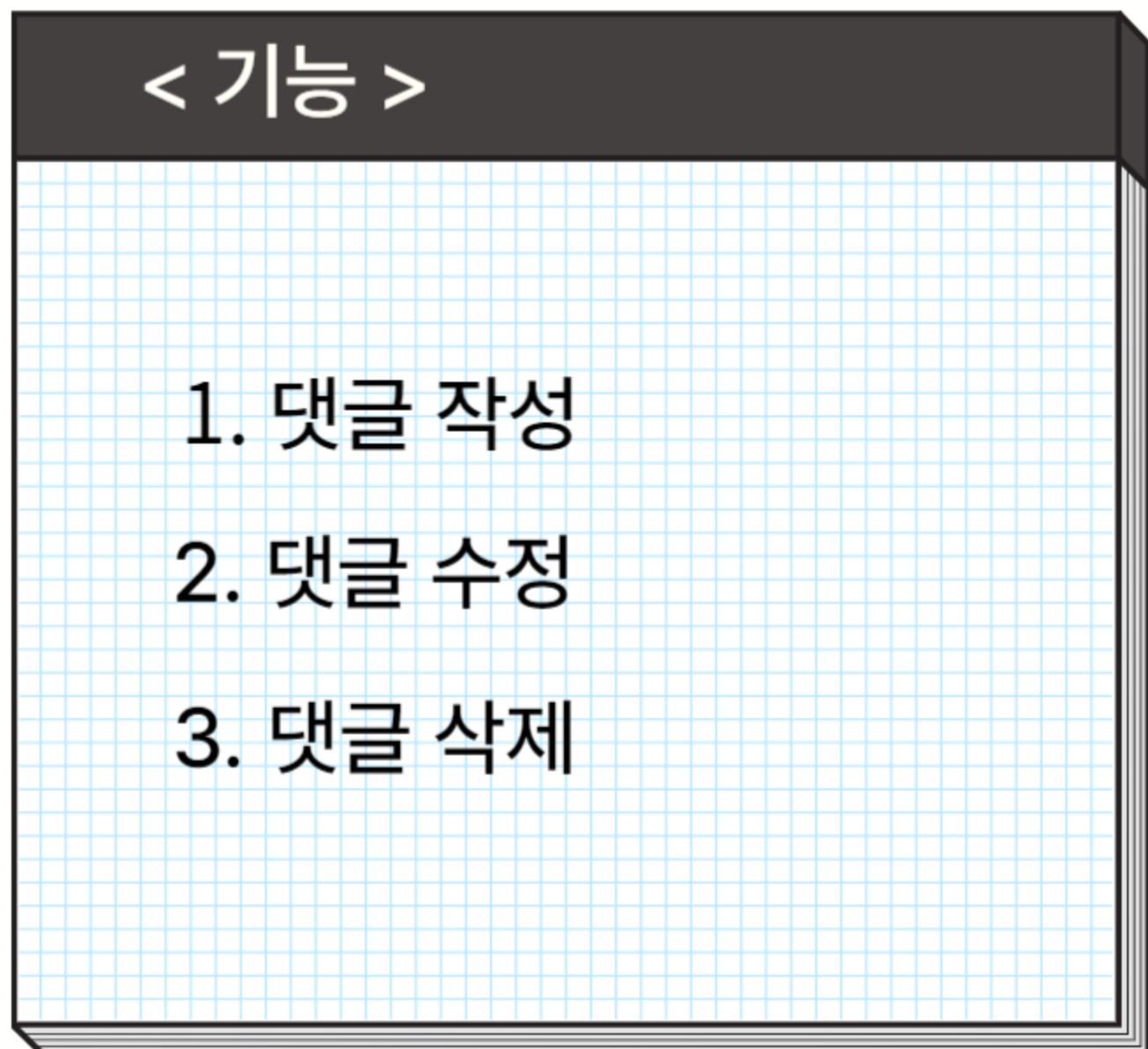
```
private Event validateEvent(BoardCreateRequest request) { 1 usage ▲ yongun2
    return eventRepository.getData(request.eventNo(), dto: null)
        .orElseThrow(() -> new TicketLinkException(EventMessageType.EVENT_NOT_FOUND));
}
```

이벤트 검증

```
private boolean hasNoOperationAuthority(String userNo, Board board) {
    return !board.getUser().getUserNo().equals(userNo);
}
```

수행 권한 확인

# 03 댓글 가능



## < 1. 댓글 작성 >

```
POST /boards/{boardNo}/reply  
-H "Authorization Bearer accessToken"  
{  
    "content": "string"  
}
```

```
@Override 2 usages ± yongun2  
@Transactional  
public FindReplyResult createReply(ReplyCreateCommand command) {  
  
    // 댓글 작성 대상 게시판 검증  
    String validBoardNo = validateBoard(command.getBoardNo());  
  
    String replyNo = generateReplyNo();  
  
    int nextReplyCnt = replyRepository.selectNextReplyCnt(command.getBoardNo());  
  
    ReplyCreateDto createDto = new ReplyCreateDto(  
        replyNo,  
        command.getContent(),  
        nextReplyCnt,  
        NOW,  
        NOW,  
        command.getUserNo(),  
        validBoardNo  
    );  
  
    replyRepository.insertReply(createDto);  
  
    return FindReplyResult.findByCreateDto(createDto);  
}
```

## < 2. 댓글 수정 >

```
PUT /boards/{boardNo}/reply/{replyNo}
-H "Authorization Bearer accessToken"
{
  "content": "string"
}
```

```
@Override 6 usages ± yongun2
@Transactional
public FindReplyResult modifyReply(ReplyUpdateCommand command) {

    validateBoard(command.getBoardNo());

    Reply reply = retrieveReply(command.getReplyNo());

    if (hasNoOperationAuthority(command.getUserNo(), reply)) {
        throw new TicketLinkException(ReplyMessageType.REPLY_OPERATION_UNAUTHORIZED);
    }

    ReplyUpdateDto updateDto = new ReplyUpdateDto(
        reply.getReplyNo(),
        command.getContent(),
        NOW
    );

    replyRepository.updateReply(updateDto);

    return FindReplyResult.findByUpdateDto(updateDto);
}
```

### < 3. 댓글 삭제 >

DELETE /boards/{boardNo}/reply/{replyNo}  
-H "Authorization Bearer accessToken"

```
@Override 5 usages ± yongun2
@Transactional
public void deleteReply(ReplyDeleteCommand command) {

    validateBoard(command.getBoardNo());

    Reply validReply = retrieveReply(command.getReplyNo());

    if (hasNoOperationAuthority(command.getUserNo(), validReply)) {
        throw new TicketLinkException(ReplyMessageType.REPLY_OPERATION_UNAUTHORIZED);
    }

    replyRepository.deleteReply(validReply.getReplyNo());
}
```

## < 4. 내부 함수 >

```
private String validateBoard(String boardNo) { 3 usages ± yongun2
    return boardRepository.selectBoardByBoardNo(boardNo)
        .orElseThrow(() -> new TicketLinkException(ReplyMessageType.REPLY_TARGET_BOARD_NOT_FOUND));
    }
}
```

게시글 유효성 검증

```
private Reply retrieveReply(String replyNo) { 2 usages ± yongun2
    return replyRepository.selectReplyByReplyNo(replyNo)
        .orElseThrow(() -> new TicketLinkException(ReplyMessageType.REPLY_NOT_FOUND));
}
}
```

댓글 가져오기

```
private boolean hasNoOperationAuthority(String userNo, Reply reply) {
    return !userNo.equals(reply.getUser().getUserNo());
}
}
```

수행 권한 확인

```
private String generateReplyNo() { 1 usage ± yongun2
    final String TABLE_NAME = "tb_reply";

    AutoNo replyNo = autoNoMapper.getData(TABLE_NAME)
        .orElseThrow(() -> new TicketLinkException(ReplyMessageType.GENERATE_REPLY_NO_FAILED));
    return replyNo.getAutoNo();
}
}
```

댓글 PK 생성

# 03 행사 기능

## < 기능 >

1. 행사 조회
  - a. 행사 전체 조회
  - b. 행사 상세 조회
  - c. 일자별 행사 조회
  - d. 행사 잔여 티켓
  - e. 행사 예매 현황 조회
2. 행사 정보 생성
3. 행사 정보 수정



## < 1-a. 행사 전체 조회 >

```
GET /events
-B "name", "eCategoryNo"
{
  "eventNo": string,
  "name": string,
  "startDate": date,
  "endDate": date,
  "location": string,
  "info": string,
  "saleInfo": string,
  "seatInfo": string,
  "timeInfo": string,
  "useYn": char,
  "dayEvents": Event,
  "ecategoryNo": int,
  "etime": string
}
```

```
@Override
public List<Event> getList(EventSearchCond cond) {
    return eventRepository.getList(cond);
}
```

```
▼ @AllArgsConstructor 17개 사용 위치
@getter
public class EventSearchCond {
    private String name;      // 행사 이름
    private Integer eCategoryNo;
}
```

## < 1-a. 행사 전체 조회 쿼리 >

```
<select id="getList" resultMap="eventMap">
    SELECT e.eventNo,
        e.name,
        e.startDate,
        e.endDate,
        e.eTime,
        e.location,
        e.info,
        e.saleInfo ,
        e.seatInfo ,
        e.eCategoryNo
    FROM tb_event e
    WHERE e.useYn = 'Y'
        <if test="eCategoryNo != null and eCategoryNo > 0">
            AND e.eCategoryNo = #{eCategoryNo}
        </if>
        <if test="name != null and name != ''">
            AND e.name LIKE concat('%', #{name}, '%')
        </if>
</select>
```

## < 1-b. 행사 상세 조회 >

```
GET /events/{eventNo}
{
  "eventNo": string,
  "name": string,
  "startDate": date,
  "endDate": date,
  "location": string,
  "info": string,
  "saleInfo": string,
  "seatInfo":string,
  "timeInfo": string,
  "useYn": char,
  "dayEvents": DayEvent
}
```

```
@Override
public Optional<Event> getData(String eventNo, DayEventSearchCond dto) {
    if (dto != null) {
        if (dto.getDayInfo() != null) {
            dto.setDays(Arrays.asList(dto.getDayInfo().split(regex: ",")));
        }
        if (dto.getTimeInfo() != null) {
            dto.setTimes(Arrays.asList(dto.getTimeInfo().split(regex: ",")));
        }
        if (dto.getCastInfo() != null) {
            dto.setCasts(Arrays.asList(dto.getCastInfo().split(regex: ",")));
        }
    }
    return eventRepository.get(eventNo);
}

public class DayEventSearchCond {
    private LocalDate sDate;
    private LocalDate eDate;
    private String dayInfo;
    private List<String> days;
    private String timeInfo;
    private List<String> times;
    private String castInfo;
    private List<String> casts;
}
```

## < 1-b. 행사 상세 조회 쿼리 >

```

SELECT e.eventNo,
       e.name,
       e.startDate,
       e.endDate,
       e.eTime,
       e.location,
       e.info,
       e.saleInfo,
       e.seatInfo,
       e.timeInfo,
       e.useYn,
       e.eCategoryNo,
       de.dailyEventNo AS de_dailyEventNo,
       de.eventDate AS de_eventDate,
       de.day7 AS de_day7,
       de.cnt AS de_cnt,
       de.deTime AS de_deTime,
       de.castInfo AS de_castInfo,
       de.eventNo AS de_eventNo
FROM tb_event e
LEFT OUTER JOIN tb_dailyEvent de ON de.eventNo = e.eventNo
WHERE e.eventNo = #{eventNo}
<if test="dto != null">
  <if test="dto.sDate != null and dto.eDate != null">
    AND (de.eventDate >= #{dto.sDate} AND de.eventDate <= #{dto.eDate})
  </if>
  <if test="dto.dayInfo != null and dto.dayInfo != ''">
    AND de.day7 IN
      <foreach collection="dto.days" item="day" open="(" close=")" separator=",">
        #{day}
      </foreach>
  </if>
  <if test="dto.timeInfo != null and dto.timeInfo != ''">
    AND de.deTime IN
      <foreach collection="dto.times" item="time" open="(" close=")" separator=",">
        #{time}
      </foreach>
  </if>
  <if test="dto.castInfo != null and dto.castInfo != ''">
    <foreach collection="dto.casts" item="cast" open=" AND " separator=" OR ">
      de.castInfo LIKE CONCAT('%', #{cast}, '%')
    </foreach>
  </if>
</if>

```

```

      SELECT e.eventNo,
             e.name,
             e.startDate,
             e.endDate,
             e.eTime,
             e.location,
             e.info,
             e.saleInfo,
             e.seatInfo,
             e.timeInfo,
             e.useYn,
             e.eCategoryNo,
             de.dailyEventNo AS de_dailyEventNo,
             de.eventDate AS de_eventDate,
             de.day7 AS de_day7,
             de.cnt AS de_cnt,
             de.deTime AS de_deTime,
             de.castInfo AS de_castInfo,
             de.eventNo AS de_eventNo
      FROM tb_event e
      LEFT OUTER JOIN tb_dailyEvent de ON de.e
      WHERE e.eventNo = #{eventNo}
      <if test="dto != null">
        <if test="dto.sDate != null and dto.eDate != null">
          AND (de.eventDate >= #{dto.sDate} AND de.eventDate <= #{dto.eDate})
        </if>
        <if test="dto.dayInfo != null and dto.dayInfo != ''">
          AND de.day7 IN
            <foreach collection="dto.days" item="day" open="(" close=")" separator=",">
              #{day}
            </foreach>
        </if>
        <if test="dto.timeInfo != null and dto.timeInfo != ''">
          AND de.deTime IN
            <foreach collection="dto.times" item="time" open="(" close=")" separator=",">
              #{time}
            </foreach>
        </if>
        <if test="dto.castInfo != null and dto.castInfo != ''">
          <foreach collection="dto.casts" item="cast" open=" AND " separator=" OR ">
            de.castInfo LIKE CONCAT('%', #{cast}, '%')
          </foreach>
        </if>
      </if>
      select>

```

## < 1-c. 행사 예매 현황 조회 >

```
GET /events/{eventNo}/{eventDate}
{
  "dayEventNo":string,
  "eventDate": date,
  "day7": int,
  "cnt": int,
  "deTime": string,
  "castInfo": string,
  "eventNo": string,
  "event": Event
}
```

```
@Override  ↗ DESKTOP-2RCK4PJ\jangd
public List<DailyEvent> getList(String eventNo, String eventDate) {
    return dayEventRepository.getList(eventNo, eventDate);
}

SELECT de.dailyEventNo,
       de.eventDate,
       de.day7,
       de.cnt,
       de.deTime,
       de.castInfo,
       de.eventNo,
       e.eventNo as e_eventNo|,
       e.name as e_name,
       e.eTime as e_eTime,
       e.location as e_location,
       e.eCategoryNo as e_eCategoryNo
FROM tb_dailyEvent de
JOIN tb_event e ON e.eventNo = de.eventNo
WHERE e.eventNo = #{eventNo}
      AND de.eventDate = #{eventDate};
```

## < 1-d. 일자별행사 잔여 티켓 조회>

```
GET  
/events/{eventNo}/{eventDate}/{dayEventNo}  
{  
    "dayEventNo":string,  
    "seatRate": string,  
    "cnt": int,  
    "price": int}
```

```
public class TicketCount {  
    String dayEventNo;  
    String seatRate;  
    String cnt;  
    Integer price;  
}
```

```
@Override 1개 사용 위치 ♫ DESKTOP-2RCK4PJ₩jangd  
public List<TicketCount> getCounts(String dayEventNo) {  
    return ticketRepository.getCounts(dayEventNo);  
}
```

## < 1-d. 일자별행사 잔여 티켓 조회 쿼리>

```
SELECT MAX(t.dailyEventNo) AS dailyEventNo,
       t.seatRate,
       COUNT(*) AS cnt,
       MAX(t.price) AS price
  FROM
    (SELECT t.*
     FROM tb_ticket t
    LEFT OUTER JOIN tb_reservation r ON r.ticketNo = t.ticketNo AND r.status != 'C'
    WHERE t.dailyEventNo = #{dayEventNo}
      AND r.ticketNo IS NULL
    ) t
 GROUP BY t.seatRate
 ORDER BY MAX(t.price) DESC
```

## < 1-e. 예매 현황 조회>

```
GET /event/stats  
-H "Authorization Bearer accessToken"  
{  
    "eventNo": string,  
    "name": string,  
    "startDate": date,  
    "endDate": date,  
    "resCnt": int,  
    "allCnt": int,  
    "eCategoryNo": int  
}
```

```
public class EventStatsDto {  
    private String eventNo;  
    private String name;  
    private LocalDate startDate;  
    private LocalDate endDate;  
    private Integer resCnt;  
    private Integer allCnt;  
    private Integer eCategoryNo;  
}
```

```
@Override 1개 사용 위치 DESKTOP-2RCK4PJ\jangd  
public List<EventStatsDto> getStats(EventSearchCond cond) {  
    return eventRepository.getStats(cond);  
}
```

## < 1-e. 예매 현황 조회 쿼리 >

```

SELECT e.eventNo,
       e.name,
       e.startDate,
       e.endDate,
       IFNULL(t1.cnt, 0) AS resCnt,
       IFNULL(t2.cnt, 0) AS allCnt,
       e.eCategoryNo
  FROM tb_event e
    LEFT OUTER JOIN
  (SELECT e.eventNo,
          COUNT(*) AS cnt
     FROM tb_event e
      JOIN tb_dailyEvent de ON e.eventNo = de.eventNo
      JOIN tb_ticket t ON t.dailyEventNo = de.dailyEventNo
      JOIN tb_reservation r ON r.ticketNo = t.ticketNo
     WHERE r.status != 'C'
   GROUP BY e.eventNo) t1 ON t1.eventNo = e.eventNo
    LEFT OUTER JOIN
  (SELECT e.eventNo,
          CASE
            WHEN de.eventDate IS NULL THEN 0
            ELSE COUNT(*)
          END AS cnt
     FROM tb_event e
      JOIN tb_dailyEvent de ON de.eventNo = e.eventNo
      LEFT OUTER JOIN tb_ticket t ON t.dailyEventNo = de.dailyEventNo
   GROUP BY e.eventNo) t2 ON t2.eventNo = e.eventNo
<where>
<if test="name != null and name != ''">
  AND e.name LIKE concat('%', #{name}, '%')
</if>
<if test="eCategoryNo != null and eCategoryNo > 0">
  AND e.eCategoryNo = #{eCategoryNo}
</if>
</where>
  
```

```

      SELECT e.eventNo,
             e.name,
             e.startDate,
             e.endDate,
             IFNULL(t1.cnt, 0) AS resCnt,
             IFNULL(t2.cnt, 0) AS allCnt,
             e.eCategoryNo
        FROM tb_event e
          LEFT OUTER JOIN
  (SELECT e.eventNo,
          COUNT(*) AS cnt
     FROM tb_event e
      JOIN tb_dailyEvent de ON e.eventNo = de.eventNo
      JOIN tb_ticket t ON t.dailyEventNo = de.dailyEventNo
      JOIN tb_reservation r ON r.ticketNo = t.ticketNo
     WHERE r.status != 'C'
   GROUP BY e.eventNo) t1 ON t1.eventNo = e.eventNo
    LEFT OUTER JOIN
  (SELECT e.eventNo,
          CASE
            WHEN de.eventDate IS NULL THEN 0
            ELSE COUNT(*)
          END AS cnt
     FROM tb_event e
      JOIN tb_dailyEvent de ON de.eventNo = e.eventNo
      LEFT OUTER JOIN tb_ticket t ON t.dailyEventNo = de.dailyEventNo
   GROUP BY e.eventNo) t2 ON t2.eventNo = e.eventNo
  
```

## < 2. 행사 정보 생성>

POST /event/register  
-H "Authorization Bearer accessToken"  
{  
    "name": string,  
    "startDate": date,  
    "endDate": date,  
    "etime": string,  
    "location": string,  
    "info": string,  
    "saleInfo": string,  
    "seatInfo": string(총좌석수@좌석등급:좌석  
    수:가격@...),  
    "timeInfo" : string(요일-순번-시각@...),  
    "ecategoryNo": int  
}

```
public Event insData(Event event) {  
    String eventNo = autoNoRepository.getData(tableName: "tb_event");  
    DailyEvent dayEvent;  
    String dayEventNo;  
  
    // event 데이터 생성  
    event.setEventNo(eventNo);  
  
    eventRepository.insData(event);
```

## < 2. 행사 정보 생성 - 일자별행사, 티켓 정보 동시 생성>

seatInfo, timeInfo 값이 존재하면 동시 생성

DailyEvent 객체 생성을 위한 변수 선언 및  
TimeInfo 클래스 생성

```
static class TimeInfo {  
    int day;  
    int cnt;  
    String time;  
  
    @Override  
    public boolean equals(Object obj) {  
        TimeInfo timeInfo = (TimeInfo) obj;  
  
        return timeInfo.day == day;  
    }  
}
```

```
// 회차 정보와 좌석정보가 존재하는 행사는 일자별 행사, 티켓 정보 동시 생성  
if (event.getSeatInfo() != null && event.getTimeInfo() != null) {  
  
    // dayEvent  
    LocalDate startDate = event.getStartDate();  
    LocalDate endDate = event.getEndDate();  
    int sDay = startDate.getDayOfWeek().getValue(); // 시작 날짜 요일(1(월) ~ 7(일))  
    long dayDiff = ChronoUnit.DAYS.between(startDate, endDate); // 시작일자와 종료일자 차이  
    StringTokenizer timeInfoSt = new StringTokenizer(event.getTimeInfo(), delim: "@");  
    ArrayList<TimeInfo> timeInfoAl = new ArrayList<>();  
  
    while (timeInfoSt.hasMoreTokens()) {  
        TimeInfo timeInfo = new TimeInfo();  
  
        StringTokenizer timeSt = new StringTokenizer(timeInfoSt.nextToken(), delim: "-");  
        timeInfo.setDay(Integer.parseInt(timeSt.nextToken()));  
        timeInfo.setCnt(Integer.parseInt(timeSt.nextToken()));  
        timeInfo.setTime(timeSt.nextToken());  
  
        timeInfoAl.add(timeInfo);  
    }  
}
```

## < 2. 행사 정보 생성 - 티켓 생성을 위한 변수 선언>

총좌석수@좌석등급:좌석수:가격 형태로  
입력된 데이터를 변환

seatCount : 총 좌석수

seatRate : 좌석등급

seatRateCnt : 좌석수

seatRatePrice : 가격

curSeatCnt : 현재 등급 생성한 좌석 수

sIdx, eIdx : 동일한 요일에 여러 행사가  
존재하는 순번만큼 데이터 생성을 위한  
변수

```
// ticket
Ticket ticket;
String ticketNo;
StringTokenizer seatInfoSt = new StringTokenizer(event.getSeatInfo(), delim: "@");
int seatCount = Integer.parseInt(seatInfoSt.nextToken());
ArrayList<Map<String, Object>> al = new ArrayList<>();
Map<String, Object> map;

while (seatInfoSt.hasMoreTokens()) {
    map = new HashMap<>();
    StringTokenizer seatRateSt = new StringTokenizer(seatInfoSt.nextToken(), delim: ":");

    map.put("seatRate", seatRateSt.nextToken());
    map.put("seatRateCnt", Integer.parseInt(seatRateSt.nextToken()));
    map.put("seatRatePrice", Integer.parseInt(seatRateSt.nextToken()));

    al.add(map);
}

int curSeatCnt = 0;
String seatRate = null;
int seatRateCnt = 0;
int seatRatePrice = 0;
int sIdx, eIdx;
```

## < 2. 행사 정보 생성 - 일자별 행사 정보 생성>

행사 일정에 해당하는 요일이 아니면 pass

해당 요일이면 순번만큼 일자별 행사 정보 생성

생성 정보마다 티켓 정보 생성

```
for (int i = 0; i <= dayDiff; i++) {
    // 행사 일정이 아니면 pass
    if (!timeInfoAl.contains(new TimeInfo(sDay, cnt: 0, time: null))) {
        startDate = startDate.plusDays(daysToAdd: 1);
        sDay = sDay < 7 ? sDay + 1 : 1;
        continue;
    } else {
        sIdx = timeInfoAl.indexOf(new TimeInfo(sDay, cnt: 0, time: null));
        eIdx = timeInfoAl.lastIndexOf(new TimeInfo(sDay, cnt: 0, time: null));

        for (int j = sIdx; j <= eIdx; j++) {
            dayEvent = new DailyEvent();
            dayEventNo = autoNoRepository.getData(tableName: "tb_dailyEvent");

            dayEvent.setDayEventNo(dayEventNo);
            dayEvent.setEventDate(startDate);
            dayEvent.setDay7(sDay);
            dayEvent.setCnt(timeInfoAl.get(j).getCnt());
            dayEvent.setDeTime(timeInfoAl.get(j).getTime());
            dayEvent.setEventNo(eventNo);

            // dayEvent ins
            dayEventRepository.insData(dayEvent);
            int seatIdx = 0;
```

## < 2. 행사 정보 생성 - 티켓(좌석) 정보 생성>

현재 등급 생성한 좌석수가 0이면  
새로운 좌석 정보 초기화

티켓 정보 세팅

티켓 정보 생성

현재 등급 생성한 좌석수가  
전달받은 해당 좌석 등급의 좌석수와 동일하면  
현재 등급 좌석수를 0으로 초기화

```
for (int k = 0; k < seatCount; k++) {
    ticket = new Ticket();
    ticketNo = autoNoRepository.getData(tableName: "tb_ticket");

    if (curSeatCnt == 0) {
        seatRate = al.get(seatIdx).get("seatRate").toString();
        seatRateCnt = Integer.parseInt(al.get(seatIdx).get("seatRateCnt").toString());
        seatRatePrice = Integer.parseInt(al.get(seatIdx).get("seatRatePrice").toString());
        seatIdx++;
        //log.info("등급 : {}, 좌석수: {}, 가격 : {}", seatRate, seatRateCnt, seatRatePrice);
    }

    ticket.setTicketNo(ticketNo);
    ticket.setSeatRate(seatRate);
    ticket.setSeatNum(k);
    ticket.setPrice(seatRatePrice);
    ticket.setDayEventNo(dayEventNo);

    // ticket ins
    ticketRepository.insData(ticket);
    curSeatCnt++;

    if (curSeatCnt == seatRateCnt) {
        curSeatCnt = 0;
    }
}
```

## < 2. 행사 정보 생성 - 상시상품의 경우 행사 정보 생성>

자식 테이블인 일자별 행사 정보 하나 생성

```
        }
    }

    startDate = startDate.plusDays( daysToAdd: 1);
    sDay = sDay < 7 ? sDay + 1 : 1;
}

// 상시상품(날짜와 좌석이 상관 없는) 행사는 티켓정보 생성X, 결제단계에서 티켓정보 생성
} else {
    dayEvent = new DailyEvent();
    dayEventNo = autoNoRepository.getData( tableName: "tb_dailyEvent");
    dayEvent.setDayEventNo(dayEventNo);
    dayEvent.setEventNo(eventNo);
    // dayEvent ins
    dayEventRepository.insData(dayEvent);
}

return event;
```

### < 3. 행사 정보 수정>

```
PUT /event/{eventNo}
-H "Authorization Bearer accessToken"
{
  "name": string,
  "eTime" : string,
  "location" : string,
  "info" : string,
  "saleInfo" : string,
  "useYn" : char,
  "eCategoryNo" : int
}
```

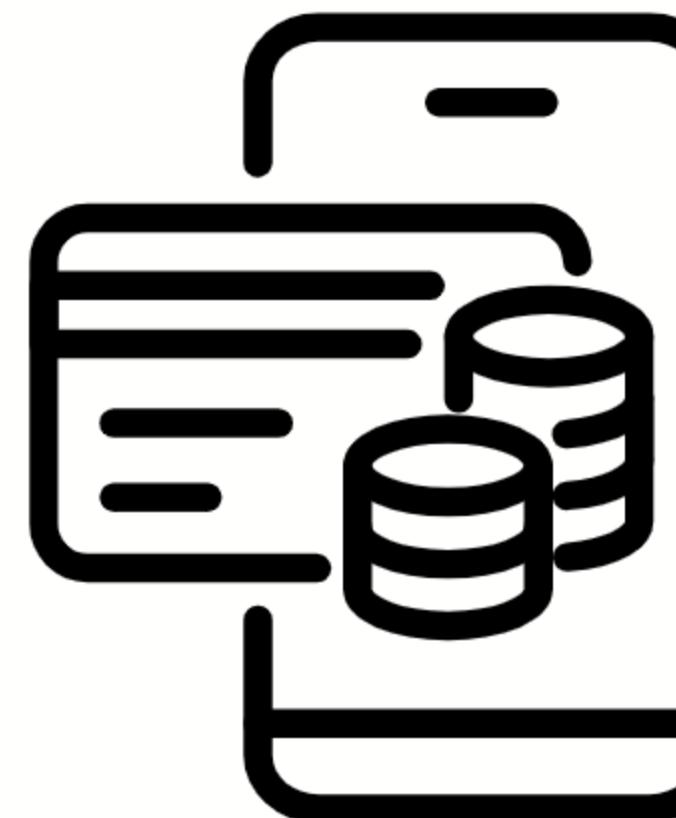
```
public class EventUpdateDto {
  private String name;      // 행사 이름
  private String eTime;     // 시작 시간
  private String location;  // 행사 장소
  private String info;      // 행사 설명
  private String saleInfo;  // 판매 정보
  private Character useYn;
  private Integer eCategoryNo;
}
```

```
@Override
@Transactional
public void uptData(String eventNo, EventUpdateDto updateParam) {
    eventRepository.uptData(eventNo, updateParam);
}
```

# 03 결제 및 예약 가능

## < 기능 >

1. 행사 티켓 현황 조회
2. 결제(예약) 정보 생성
3. 결제(예약) 조회
  - a. 전체 조회
  - b. 상세 조회
4. 결제 정보 취소



## < 1. 행사 티켓 현황 조회 >

GET /res/{dayEventNo}  
-H "Authorization Bearer accessToken"  
{  
 "ticketNo": stirng,  
 "seatRate": string,  
 "seatNum": int,  
 "price": int,  
 "dayEventNo":string,  
 "dailyEvent": DailyEvent,  
 "reservation": Reservation  
}

```
@Override  ↗ DESKTOP-2RCK4PJ\jangd
public List<Ticket> getList(String dayEventNo) {
    return ticketRepository.getList(dayEventNo);
}
```

## < 1. 행사 티켓 현황 조회 SQL >

```
SELECT t.ticketNo,
       t.seatRate,
       t.seatNum,
       t.price,
       t.dailyEventNo,
       de.dailyEventNo AS de_dailyEventNo,
       de.eventDate AS de_eventDate,
       de.day7 AS de_day7,
       de.cnt AS de_cnt,
       de.deTime AS de_deTime,
       de.castInfo AS de_castInfo,
       de.eventNo AS de_eventNo,
       res.resNo AS res_resNo,
       res.resDate AS res_resDate,
       res.status AS res_status
  FROM tb_ticket t
 JOIN tb_dailyEvent de ON de.dailyEventNo = t.dailyEventNo
 LEFT OUTER JOIN tb_reservation res ON res.ticketNo = t.ticketNo AND res.status != 'C'
 WHERE t.dailyEventNo = #{dayEventNo}
 ORDER BY t.ticketNo
```

## < 2. 결제 정보 생성>

```
POST /event/register  
-H "Authorization Bearer accessToken"  
{  
    "ticketList" : [Ticket],  
    "couponUsedInfo" : string,  
    "price" : int,  
    "fee" : int,  
    "deliveryCost" : int,  
    "discount" : int,  
    "totalAmt" : int,  
    "payment" : char  
}
```

```
public class PayDto {  
    private List<Ticket> ticketList;  
    private String couponUsedInfo;  
    private Long price;  
    private Long fee;  
    private Long deliveryCost;  
    private Long discount;  
    private Long totalAmt;  
    private Character payment;  
    private String couponNo;  
}
```

## < 2. 결제 정보 생성 - 변수 선언 및 초기화 >

결제, 예약, 알림 관련 변수 선언 및 초기화

```
public PayInfo insData(String dayEventNo, PayDto dto, String userNo) {  
    // 결제 관련 변수  
    PayInfo payInfo = new PayInfo();  
    String payNo = autoNoRepository.getData(tableName: "tb_payinfo");  
    // 예약 관련 변수  
    Reservation reservation;  
    String resNo = null;  
    String status = "W";  
    // 알림 관련 변수  
    NotificationDto notiDto = new NotificationDto();  
    String notiMsg = "";  
  
    // 결제 정보 세팅  
    payInfo.setPayNo(payNo);  
    payInfo.setPayment(dto.getPayment());  
  
    // 결제 수단이 무통장입금(T)이 아닐경우 결제완료(S)로 세팅  
    if (dto.getPayment() != 'T') {  
        payInfo.setPayDate(LocalDate.now());  
        status = "S";  
    }  
  
    payInfo.setStatus(status.charAt(0));
```

## < 2. 결제 정보 생성 - 쿠폰 적용 및 결제 정보 세팅>

```
// 쿠폰을 사용했을 경우 할인가 적용하기
Long price = dto.getPrice();
Coupon validCoupon = null;
if (dto.getCouponNo() != null) {
    String couponNo = dto.getCouponNo();

    // 쿠폰을 조회 및 쿠폰 유효성 검증
    validCoupon = validateCoupon(couponNo, userNo);

    // 할인된 가격 계산
    price = calculateDiscountPrice(validCoupon, price);
}

payInfo.setPrice(price);
payInfo.setFee(dto.getFee());
payInfo.setDeliveryCost(dto.getDeliveryCost());
payInfo.setDiscount(dto.getDiscount());
payInfo.setTotalAmt(dto.getTotalAmt());
payInfo.setUserNo(userNo);

payRepository.insData(payInfo);
//log.info("payInfo = {}", payInfo);
if (couponUsed(validCoupon)) {
    createCouponUsedHistory(validCoupon, payNo);
}
```

## < 2. 행사 정보 생성 - 예약 정보 생성>

상시 상영 행사의 경우 티켓 정보가 없기 때문에  
예약 정보 생성 전에 티켓 정보 생성

티켓 중복 예약 여부 체크

예약 정보 세팅

예약 정보 생성

```
List<Ticket> tickets = dto.getTicketList();
String ticketNo;

// 예약 정보 생성 로직
for (Ticket ticket : tickets) {
    ticketNo = ticket.getTicketNo();
    // 티켓정보가 없는 상시상품일 경우 티켓정보도 동시에 생성
    if (ticketNo == null) {
        ticketNo = autoNoRepository.getData(tableName: "tb_ticket");
        ticket.setTicketNo(ticketNo);
        ticket.setDayEventNo(dayEventNo);

        ticketRepository.insData(ticket);
    }
    // 중복 예약 체크
    int chk = resRepository.getChkRes(ticketNo);

    if (chk > 0) {
        throw new TicketLinkException(ResMessageType.RES_OPERATION_DUPLICATE);
    }

    // 중복이 아닐 경우 예약 정보 세팅
    reservation = new Reservation();
    resNo = autoNoRepository.getData(tableName: "tb_reservation");
    reservation.setResNo(resNo);

    if (status != null) {
        reservation.setStatus(status.charAt(0));
    }

    reservation.setTicketNo(ticketNo);
    reservation.setPayNo(payNo);

    resRepository.insData(reservation);
}
```

## < 2. 행사 정보 생성 - 알림 정보 생성>

무통장입금일시 메세지 문구 변경

메세지 문구 세팅

알림 생성

```
// 알림 정보 생성 로직
notiDto.setNotiNo(autoNoRepository.getData(tableName: "tb_notification"));
payInfo = payRepository.getData(payInfo.getPayNo()).get();

// 메세지 문구 설정
if (dto.getPayment() == 'T') {
    notiMsg += "입금 요청\n\n";
} else {
    notiMsg += "예매 완료\n\n";
}

notiMsg += "상품명 : " + payInfo.getReservations().getFirst()
    .getTicket().getDailyEvent().getEvent().getName() + "\n";
notiMsg += "예매번호 : " + resNo + " [총" + tickets.size() + "장]\n";

if (dto.getPayment() == 'T') {
    notiMsg += "입금기한 : " + LocalDate.now().plusDays(daysToAdd: 1)
        + " (" + LocalDate.now().getDayOfWeek().getDisplayName(TextStyle.SHORT, Locale.KOREAN)
        + ") 23시 59분";
}

notiDto.setMessage(notiMsg);
notiDto.setPayNo(payNo);
notificationRepository.insertNoti(notiDto);

return payInfo;
}
```

### < 3-a. 결제 전체 조회>

```
GET /res/myInfo  
-H "Authorization Bearer accessToken"  
{  
    "payNo":string,  
    "resDate": date,  
    "resNo":string,  
    "name": string,  
    "eventDate": string,  
    "deTime": string,  
    "cnt": int,  
    "status": char  
}
```

```
public class PayListDto {  
    private String payNo;  
    private LocalDate resDate;  
    private String resNo;  
    private String name;  
    private String eventDate;  
    private String deTime;  
    private Integer cnt;  
    private Character status;  
}
```

```
@Override  ↗DESKTOP-2RCK4PJWjangd  
public List<PayListDto> getList(String userNo) {  
    return payRepository.getList(userNo);  
}
```

## < 3-a. 결제 전체 조회 쿼리>

```
SELECT p.payNo,
       MAX(r.resDate) AS resDate,
       MAX(r.resNo) AS resNo,
       e.name,
       IFNULL(de.eventDate, CONCAT(e.startDate, ' ~ ', e.endDate)) AS eventDate,
       de.deTime,
       COUNT(*) AS cnt,
       p.status
  FROM tb_payinfo p
 JOIN tb_reservation r ON r.payNo = p.payNo
 JOIN tb_ticket t ON t.ticketNo = r.ticketNo
 JOIN tb_dailyEvent de ON de.dailyEventNo = t.dailyEventNo
 JOIN tb_event e ON e.eventNo = de.eventNo
 WHERE p.userNo = #{userNo}
   AND r.status != 'C'
 GROUP BY p.payNo
 ORDER BY MAX(r.resDate) DESC
```

### < 3-b. 결제 상세 조회>

POST /res/myInfo/{payNo}

-H "Authorization Bearer accessToken"  
{

    "payNo":string,  
    "payment": char,  
    "payDate": date,  
    "status": char,  
    "price": int,  
    "fee": int,  
    "deliveryCost": int,  
    "discount": int,  
    "totalAmt": int,  
    "userNo": string,  
    "insDate": date,  
    "reservations": [Reservation]  
}

```
@Override ✎ mabem95
public Optional<PayInfo> getData(String payNo) {
    return payRepository.getData(payNo);
}
```

## < 3-b. 결제 상세 조회 쿼리 >

```
SELECT e.name,
       e.startDate,
       e.endDate,
       e.location,
       de.eventDate,
       de.day7,
       de.deltaTime,
       p.payNo,
       p.payment,
       p.payDate,
       p.status,
       p.fee,
       p.deliveryCost,
       p.totalAmt,
       p.insDate,
       r.resNo AS rs_resNo,
       r.resDate AS rs_resDate,
       r.status AS rs_status,
       r.payNo AS rs_payNo,
       r.ticketNo AS rs_ticketNo,
       t.ticketNo AS t_ticketNo,
       t.seatRate AS t_seatRate,
       t.price AS t_price
  FROM tb_payinfo p
    JOIN tb_reservation r ON r.payNo = p.payNo
    JOIN tb_ticket t ON t.ticketNo = r.ticketNo
    JOIN tb_dailyEvent de ON de.dailyEventNo = t.dailyEventNo
    JOIN tb_event e ON e.eventNo = de.eventNo
 WHERE p.payNo = #{payNo}
```

### < 3. 결제 취소 >

PUT /res/myInfo/{payNo}  
-H "Authorization Bearer accessToken"  
{  
    "payNo":string,  
    "payment": char,  
    "payDate": date,  
    "status": char,  
    "price": int,  
    "fee": int,  
    "deliveryCost": int,  
    "discount": int,  
    "totalAmt": int,  
    "userNo": string,  
    "insDate": date,  
    "reservations": [Reservation]  
}

```
<update id="uptData" parameterType="string">  
    UPDATE tb_payinfo  
    SET status = 'C',  
        uptDate = CURDATE()  
    WHERE payNo = #{payNo}  
</update>
```

```
<update id="uptData" parameterType="string">  
    UPDATE tb_reservation  
    SET status = 'C',  
        uptDate = CURDATE()  
    WHERE payNo = #{payNo}  
</update>
```

### < 3. 결제 취소 로직 >

```
public PayInfo updateData(String payNo) {
    PayInfo payInfo = payRepository.getData(payNo).get();

    List<Reservation> reservations = payInfo.getReservations();
    NotificationDto notiDto = new NotificationDto();

    payRepository.uptData(payNo);
    resRepository.uptData(payNo);

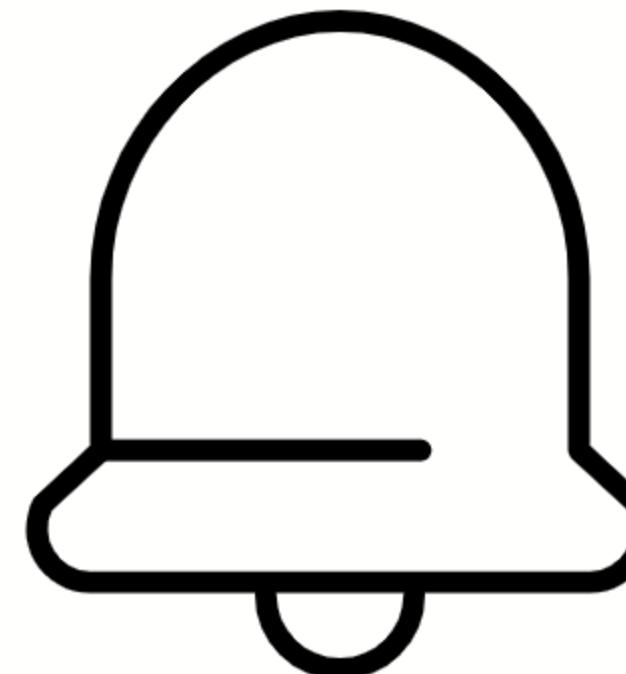
    // 알림 정보 생성 로직
    notiDto.setNotiNo(autoNoRepository.getData(tableName: "tb_notification"));
    notiDto.setMessage("예매 취소\n\n상품명 : " + reservations.get(0).getTicket()
        .getDailyEvent().getEvent().getName()
        + "\n예약번호 : " + reservations.get(reservations.size() - 1).getResNo());
    notiDto.setPayNo(payNo);
    notificationRepository.insertNoti(notiDto);

    return payRepository.getData(payNo).get();
}
```

# 03 알림 기능

## < 기능 >

1. 알림 조회
  - a. 전체 조회
  - b. 상세 조회
2. 알림 삭제



## < 1-a. 알림 전체 조회>

GET /notices

-H "Authorization Bearer accessToken"  
{

  "notiNo": string,  
  "message": string,  
  "notiDate": date,  
  "notiStatus": char,  
  "payNo": string,  
  "payInfo": Payinfo  
}

```
@Override 1개 사용 위치 byHyen
public List<Notification> getAll(String userNo) {
    return notificationRepository.getAll(userNo);
}
```

## < 1-a. 알림 전체 조회 쿼리 >

```
SELECT p.userNo as p_userNo,
       n.message,
       r.resNo,
       n.notiDate ,
       p.payment as p_payment,
       n.notiDate,
       n.notiStatus
FROM tb_notification n
      LEFT OUTER JOIN tb_payinfo p ON p.payNo = n.payNo
      LEFT OUTER JOIN tb_reservation r ON r.payNo = n.payNo
WHERE p.userNo = #{userNo}
```

## < 1-b. 알림 상세 조회>

GET /notices/{notiNo}

-H "Authorization Bearer accessToken"  
{

  "notiNo": string,  
  "message": string,  
  "notiDate": date,  
  "notiStatus": char,  
  "payNo": string,  
  "payInfo": Payinfo  
}

*@Override* 2개 사용 위치  byHyen  
*public Notification getNoti(String notiNo) {*  
  *| return notificationRepository.getNoti(notiNo);*  
  *}*

## < 1-b. 알림 상세 조회 쿼리 >

```
SELECT  n.notiNo,
        n.message,
        r.resNo,
        n.notiDate,
        n.notiDate,
        n.notiStatus,
        n.payNo,
        p.payNo AS p_payNo,
        p.payment AS p_payment,
        p.price AS p_price
FROM tb_notification n
    LEFT OUTER JOIN tb_payinfo p ON p.payNo = n.payNo
    LEFT OUTER JOIN tb_reservation r ON r.payNo = n.payNo
WHERE n.notiNo = #{notiNo}
```

### < 3. 알림 삭제(수정) >

```
PUT /notices/{notiNo}
-H "Authorization Bearer accessToken"
{
  "notiNo": string,
  "message": string,
  "notiDate": date,
  "notiStatus": char,
  "payNo": string,
  "payInfo": Payinfo
}
```

```
@Override 1개 사용 위치 byHyen 1
public void updateNoti(String notiNo) {
    notificationRepository.updateNoti(notiNo);
}
```

```
UPDATE tb_notification
SET notiStatus = 'N'
WHERE notiNo = #{notiNo}
```

## 03 쿠폰 가능

< 가능 >

1. 관리자 쿠폰발급
2. 쿠폰 조회
3. 쿠폰 수정
4. 쿠폰 삭제



## < 1. 쿠폰 발급 >

POST /coupons  
-H "Authorization Bearer accessToken"  
{  
  "name": "string",  
  "dcPercent": 0,  
  "userNo": "string",  
  "expireDate": "2024-07-31"  
}

```
@Override 2 usages ▾ yongun2
@Transactional
public void createCoupon(CouponCreateRequest request) {

    String couponNo = generateCouponNo();

    String userNo = validateUserNo(request.getUserNo());

    String generatedCode = generateCouponCode();

    CouponCreateDto couponCreateDto = new CouponCreateDto(
        couponNo,
        generatedCode,
        request.getName(),
        request.getDcPercent(),
        LocalDate.now(),
        request.getExpireDate(),
        userNo
    );
    couponRepository.save(couponCreateDto);
}
```

## < 2. 쿠폰 조회 >

GET /coupons?userNo  
-H "Authorization Bearer accessToken"  
{  
 "couponNo": "string",  
 "code": "string",  
 "name": "string",  
 "dcPercent": 0,  
 "insDate": "2024-07-31",  
 "expireDate": "2024-07-31",  
 "userNo": "string"  
}

```
@Override 2 usages ▾ yongun2
@Transactional
public void updateCoupon(String couponNo, CouponUpdateRequest request) {

    // 유효한 쿠폰에 대한 요청인지 확인
    Coupon coupon = retrieveCoupon(couponNo);

    CouponUpdateDto couponUpdateDto = new CouponUpdateDto(
        coupon.getCouponNo(),
        request.getName(),
        request.getDcPercent(),
        request.getExpireDate()
    );
    couponRepository.updateCoupon(couponUpdateDto);
}
```

```
<select id="selectAllCoupons" parameterType="CouponFindQuery" resultMap="couponResultMap">
    <include refid="selectCouponSQL"/>
    <where>
        <if test="userNo != null">
            AND userNo = #{userNo}
        </if>
    </where>
</select>
```

### < 3. 쿠폰 수정 >

```
PUT /coupons/{couponNo}
-H "Authorization Bearer accessToken"
{
  "name": "string",
  "dcPercent": 0,
  "expireDate": "2024-07-31"
}
```

```
@Override 2 usages ▾ yongun2
@Transactional
public void updateCoupon(String couponNo, CouponUpdateRequest request) {
    // 유효한 쿠폰에 대한 요청인지 확인
    Coupon coupon = retrieveCoupon(couponNo);

    CouponUpdateDto couponUpdateDto = new CouponUpdateDto(
        coupon.getCouponNo(),
        request.getName(),
        request.getDcPercent(),
        request.getExpireDate()
    );
    couponRepository.updateCoupon(couponUpdateDto);
}
```

```
<update id="updateCoupon" parameterType="CouponUpdateDto">
    UPDATE tb_coupon
    SET name      = #{name},
        dcPercent = #{dcPercent},
        expireDate = #{expireDate}
    WHERE couponNo = #{couponNo}
</update>
```

## < 4. 쿠폰 삭제 >

DELETE /coupons/{couponNo}  
-H "Authorization Bearer accessToken"

```
@Override 2 usages ▾ yongun2
@Transactional
public void deleteCoupon(String couponNo) {

    // 유효한 쿠폰에 대한 요청인지 확인
    Coupon coupon = retrieveCoupon(couponNo);

    couponRepository.deleteCoupon(coupon.getCouponNo());
}
```

```
<delete id="deleteCoupon" parameterType="string">
    DELETE FROM tb_coupon
    WHERE couponNo = #{couponNo}
</delete>
```

## 03 개선사항



1. 티켓 예약시 과도한 트래픽에 대한 대처 방한
  - a. 대기열 기능
2. 웹 푸시 알림 기능 도입
3. 스프링 스케줄러를 통한 상영 시작 전 알림 서비스
4. BLOB을 이용한 행사 정보 저장 및 출력

# 감사합니다

이상으로 발표를 마치겠습니다.

