

WebSocket 기반 실시간 스포츠 매칭 플랫폼

3팀

MatchON



목차

01 서비스 개요

- 기획배경 및 기대효과
- 로고 소개
- 팀원 소개

02 서비스 설계

- WBS
- 요구사항 정의서
- 메뉴트리
- 유스케이스
- 엔티티 정의서
- 테이블 정의서
- ERD
- 화면 설계

03 서비스 상세

- 기술 스택
- 시스템 설계도
- 주요 기능 리뷰
- 단위 테스트
- 통합 테스트
- 서비스 개발 기록
- 서비스 시연

04 회고

- 보완 및 추후 계획
- 소감
- 출처 및 참고문헌
- Q&A

기획배경 및 기대효과

= 스포츠 팀 매칭을 위한 실시간 플랫폼

“팀 스포츠를 즐기고 싶은 사람들끼리, 실시간으로 팀을 만들고 경기를 매칭하며, 커뮤니티까지 함께 즐길 수 있는 플랫폼이 있다면 어떨까? ”라는 고민에서 출발

-  실시간 매치업 및 경기 참여
-  팀 중심 커뮤니티 구성 및 소통
-  대회 일정 등록 및 열람
-  경기장 정보 및 위치 확인의 간편화
-  고객센터와 챗봇 기능으로 사용자 문의 해결

로고 소개



스포츠의 역동성과 사람 간 연결을 상징하며, **실시간 매칭**의 의미를 담은
'Match On'의 정체성을 직관적으로 표현

Team MatchON



전준혁

커뮤니티 기능
댓글 기능
사용자 신고 및 정지
기능
서비스 UI



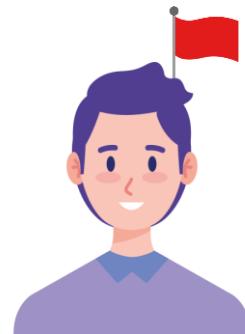
정준열

Team 기능
리뷰 기능
Team 채팅
발표



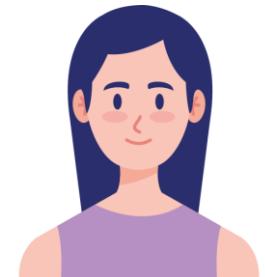
최성은

FAQ 기능
구장 조회



최효민(팀장)

MatchUP 기능
개인, 단체 채팅
매너온도 평가
알림 서비스
DB & git 관리
배포



홍주희

회원 관리
메일 알림 서비스
1:1 문의
대회 이벤트
AI 챗봇
시스템 설계
노션 관리
PPT

WBS

구글드라이브 링크

5월

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

유사프로그램 분석 2025년 4월 29일 → 2025년 5월 13일 업무분석 및 요건정의 ● 완료

업무 분배 + 기능 정의 2025년 4월 30일 → 2025년 5월 16일 업무분석 및 요건정의 ● 요건정의 ● 완료

메뉴트리 + 유스케이스 설계 2025년 5월 4일 → 2025년 5월 19일 업무분석 및 요건정의 ● 요구분석 ● 완료

요구사항 정의서 작성 2025년 5월 7일 → 2025년 5월 21일 업무분석 및 요건정의 ● 요건정의 ● 완료

피그마 설계 2025년 5월 8일 → 2025년 5월 20일 기획 / 설계 ● 화면설계 ● 완료

기획안 작성 2025년 5월 8일 → 2025년 5월 19일 기획 / 설계 ● 기획안 ● 완료

중간보고 2025년 5월 10일 → 2025년 5월 21일 기획 / 설계 ● 멘토링 ● 완료

기능 관련 도메인 리서치 2025년 5월 12일 → 2025년 5월 25일 기획 / 설계 ● 기획 ● 완료

ERD 설계 + 엔터티 정의서 + 테이블 정의서 작성 2025년 5월 10일 → 2025년 5월 27일 기획 / 설계 ● DB설계 ● 완료

RestAPI 설계 2025년 5월 14일 → 2025년 5월 27일 기획 / 설계 ● 프로그램설계 ● 완료

공공데이터 포털 데이터 수집 및 분석 2025년 5월 14일 → 2025년 5월 30일 기획 / 설계 ● 프로그램설계 ● 완료

메인 개발 2025년 5월 14일 → 2025년 5월 25일 개발 ● 프로그램개발 ● 완료

6월

UI 통합 2025년 6월 9일 → 2025년 6월 19일 개발 ● 디자인 ● 완료

배포 2025년 6월 10일 → 2025년 6월 20일 배포 ● 배포 ● 완료

1차 단위테스트 2025년 6월 11일 → 2025년 6월 23일 테스트 ● 단위테스트 ● 완료

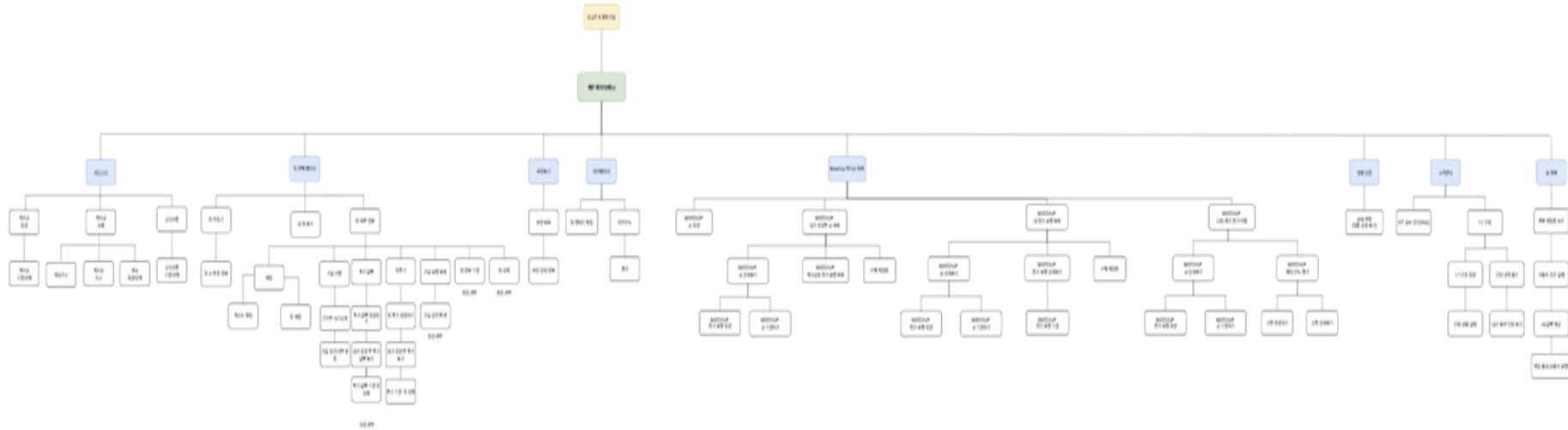
2차 통합테스트 2025년 6월 13일 → 2025년 6월 25일 테스트 ● 통합테스트 ● 완료

프로젝트 보고서 및 PPT 제작 2025년 6월 11일 → 2025년 6월 24일 완료 ● 산출물정리 ● 완료

프로젝트 발표 2025년 6월 16일 → 2025년 6월 25일 완료 ● 시연발표 ● 완료

메뉴트리

구글드라이브 링크



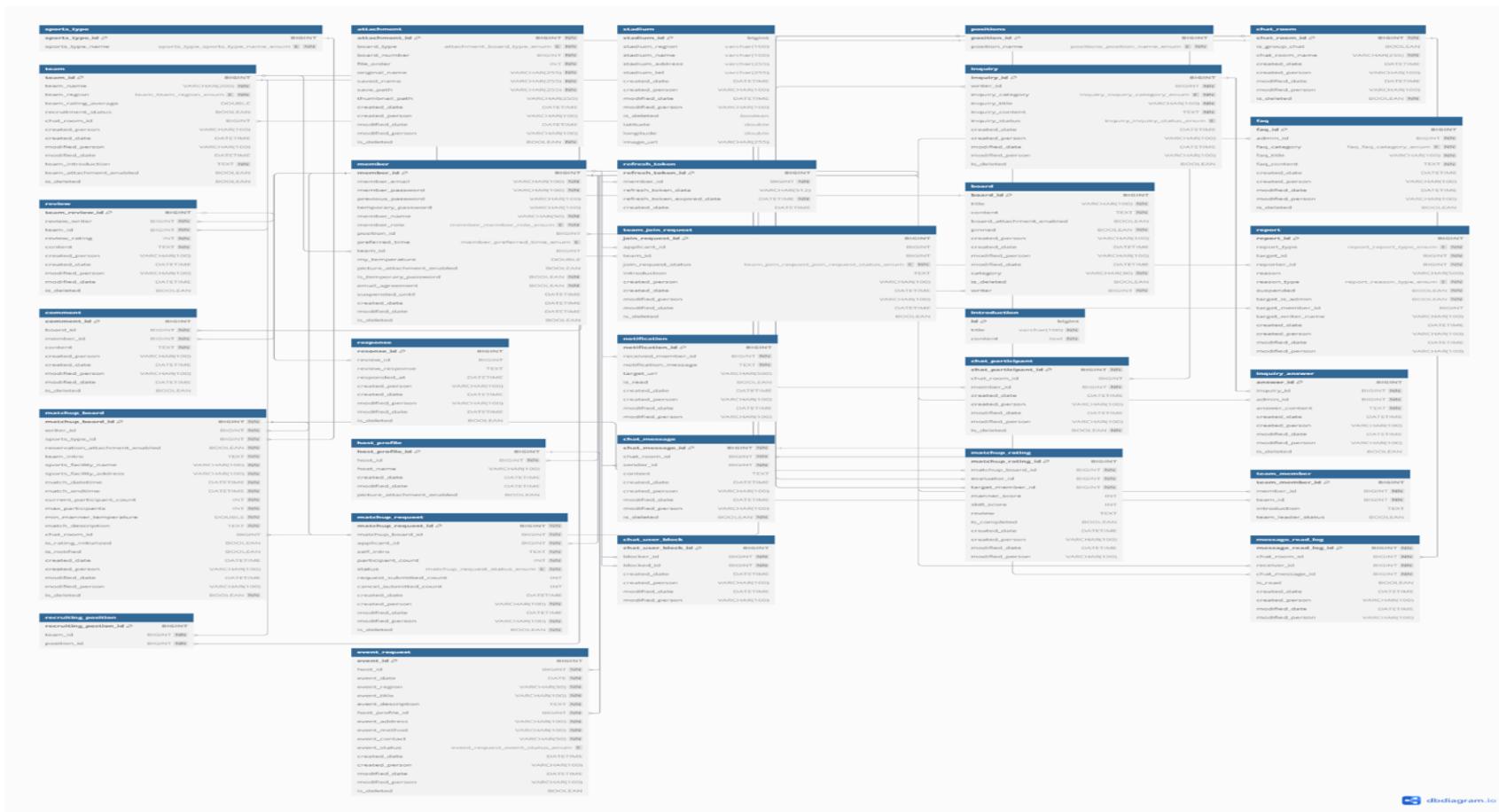
엔티티 정의서

구글드라이브 링크

ENTITY 정의서					
Entity명	Entity설명	관련속성	유사어	비고	
구장 목록	각 구장의 목록을 관리합니다.	구장 ID(PK), 구장 지역, 구장 이름, 구장 이미지	경기장 목록	구장별 ID 기준으로 목록 제공, 삭제 시 관련 상세/위치 정보도 함께 삭제 필요	
구장 상세	구장의 추가적인 상세 정보를 관리합니다.	구장 상세 ID(PK), 구장 ID(FK), 구장 지역, 구장 이름, 구장 위치, 구장 이미지, 구장 주소	경기장 상세	구장 ID로 상세정보 연결, 위치 및 이미지 포함한 종합 정보 제공	
구장 위치	구장의 위치를 지도로 표시합니다.	구장 위치 ID(PK), 구장 지도, 구장 주소, 구장 상세 ID(FK)	구장 지도	지도 연동을 위한 위치 정보 관리, 주소와 위경도 기반 표시	
FAQ	사용자들이 자주 묻는 질문 등록/조회	FAQ 번호(PK), 관리자 ID(FK), 카테고리, 질문 제목, 질문 답변, 생성자, 수정자, 생성일, 수정일, 삭제여부	자주 묻는 질문	카테고리는 Enum{team, guest, competitor, tutorial, manner, community, account, report}으로 구분, 관리자면 작성, 수정, 삭제 가능	
1:1 문의	사용자와 1:1 문의 등록/조회	문의 ID(PK), 작성자 ID(FK), 카테고리, 문의 제목, 문의 내용, 답변 상태, 생성자, 수정자, 생성일, 수정일, 삭제여부	개인 문의	문의 답변 엔티티와 1:1 관계, 카테고리는Enum{team, guest, competitor, tutorial, manner, community, account, report}으로 구분(gteam, guest, competitor, tutorial, manner, community, account, report), 사용자는 답변이 전달되어야 처리되도록 처리, 삭제 불가	
1:1 문의 답변	관리자가 남긴 답변 정보	답변 ID(PK), 연결된 문의와(PK, UNIQUE), 관리자 ID, 답변 내용, 생성자, 수정자, 생성일, 수정일, 삭제여부	개인 문의 답변	연결된 문의를 참조하여, UNIQUE 세약조건으로 1:1 관계, 답변 상태를 '답변완료'로 업데이트하는 로직 필요	
대회 등록 요청	주최자가 등록하는 대회 정보	대회 ID(PK), 주최자 ID(FK), 대회 날짜, 지역, 대회명, 참가자리수(PK), 대회 분수, 대회 설명, 신청 방법, 문의 연락처, 등록 상태, 생성자, 수정자, 생성일, 수정일, 삭제여부	경기 정보	대회 등록 엔티티와 1:1 관계를 맺어야 하는 상위 테이블 관리, 차례는Enum{capital, juju, aeolia, gangwon, gyeongsang, chungcheong}으로 기본	
회원	회원 정보	회원 ID(PK), 이메일, 비밀번호, 이전 비밀번호, 이동, 권한, 선호 지역코드, 선호 시간대, 사진_첨부파일_여부, 아이온도, 정치기자, 이메일 수신 동의, 임시 비밀번호 여부, 생성일, 수정일, 삭제여부	사용자	회원은 ENUM{user, admin, host}로 구분, 일부 필드는 주후 일자 기능(s속성, 이메일 등), 선호 시간대 ENUM{weekday_morning, weekday_afternoon, weekday_evening, weekend_morning, weekend_afternoon, weekend_evening}로 구분, 선호 지역코드는 기존(서예 아름다움)에서 딜레이 이후 할 수 있음	
주최자 프로필	주최기관 전문 정보	주최기관 ID(PK), 주최자 ID(FK), 주최기관 명, 사진_첨부파일_여부, 생성일, 수정일	관계기관 프로필		
제발급 토큰	refresh token 관리	토큰 ID(PK), 회원 ID(FK), 리프레시 토큰, 만료시간, 생성일	갱신 토큰	JWT 새발급 토큰, 리프레시 토큰은 일회화 저장	
종목	스포츠 종목 정보	종목 ID(PK), 종목명	종류	관리자만 추가 가능	
포지션	종목별 포지션 정보	포지션 ID(PK), 포지션명	역할	관리자만 추가 가능	
매지업 계시글	사용자가 매지업 계시글 등록/수정/삭제	매지업_등록시작일_ID(PK), 작성자_ID(FK), 종목_ID(FK), 내용_ID(FK), 예약_날짜, 원소개_이름, 원소개_설명, 원소개_주소, 원소개_주소_주소, 원소개_주소_주소_주소, 원시_사진파일, 원시_증명파일, 현시_상가주소, 최대_상가주소, 최소_상가주소, 웹_설명, 페인트샵_여부, 블라인드_여부, 생성일, 수정일, 삭제여부	팀 매칭 글	게시글 삭제 시 삭제 여부 'y'로 변경	
매지업 요청	사용자가 매지업 계시글 작성하여 경기 참가 요청, 조회/수정/삭제	매지업_요청_ID(PK), 매지업_캐시화_ID(FK), 신청자_ID(FK), 주소_주소_상가주소, 상태, 요청횟수, 승인취소요청횟수, 생성일, 생성자, 수정일, 수정자, 수정여부	경기 참가 신청	참가 요청 삭제 시 삭제 여부 'y'로 변경	
매지업 평가	매지업 경기 종료 후 상대방 평가 등록	매지업_평가_ID(PK), 게시글_평가_ID(FK), 평가자_ID(FK), 평가_내용_평가자_ID(FK), 평가_내용_평가자_내용, 평가점수, 주기, 주기_작성_여부, 생성일, 생성자, 수정일, 수정자, 수정여부	상대방 평가	매니저수와 실력 점수는 1, 2, 3, 4, 5 중 하나	
첨부파일	첨부파일 관리	첨부파일_설명_ID(PK), 개시글_번호, 파일_순서, 원본이름, 서명이름, 서명경로, 생성일, 생성자, 수정일, 수정자, 삭제여부	업로드 파일	여러 개시글에서 공동 가능, 첨부파일 삭제 시 삭제 여부 'y'로 변경	
채팅방	채팅방 관리	채팅방_ID(PK), 그룹채팅여부, 채팅방이름, 생성일, 생성자, 수정일, 수정자, 삭제여부	대화방	1대1 및 그룹 채팅 지원, 채팅방 삭제 시 삭제 여부 'y'로 변경	
채팅방 메시지	채팅방 내 메시지 관리	채팅방_메시지_ID(PK), 채팅방_ID(FK), 보내는회원(ID(FK), 예약자, 생성일, 생성자, 수정일, 수정자, 삭제여부	대화메시지	채팅방 삭제 시 차동으로 나감 여부 'y'로 변경, 채팅방 나감 여부 'y'로 변경	
채팅방 참가자	채팅방 참가자 관리	채팅방_참가자_ID(PK), 채팅방_ID(FK), 학생ID_ID(FK), 학생명, 생성일, 생성자, 수정일, 수정자, 나감여부	대화참가자	채팅방 삭제 시 차동으로 나감 여부 'y'로 변경, 채팅방 나감 여부 'y'로 변경	
채팅방 차단	채팅 방 차단 관리	채팅방_차단ID_ID(PK), 차단ID_ID(FK), 피해당사자_ID(FK), 차단_내용, 차단일, 생성일, 수정일, 수정자	블랙리스트 등록	채단 ID단해제하면 영구 삭제	
메시지 읽기 기록	메시지 읽기 기록 관리	메시지_읽기_기록_ID(PK), 채팅방_ID(FK), 수신자_ID(FK), 채팅_메시지_ID(FK), 읽음여부, 생성일, 생성자, 수정일, 수정자	대화 읽기 기록	채팅방 담으면 읽음 여부 'y'로 변경	
팀	모든 팀 정보를 관리합니다	팀 아이디(PK), 팀 이름, 팀지역, 팀 평균, 모집여부, 생성일, 생성자, 수정일, 수정자, 첨부파일 여부, 삭제여부	전체 팀	팀 후기 별점을 참고하여 별점 평균 생성, (팀지역, 팀 이름, 팀 소개문) NOTNULL	
후기	팀에 대한 후기 관리합니다	후기 번호(PK), 팀에 대한 평가(PK), 팀 아이디(PK), 후기 별점, 후기 내용, 생성일, 수정일, 수정자, 삭제 여부	팀 후기	작성한 후기 별점과 참고하여 별점 평균 생성, 팀에 대한 평가 NOTNULL	
후기 답변	팀 후기에 대한 답변 관리	후기_답변_번호(PK), 팀 번호(PK), 후기 번호(PK), 후기 내용, 생성일, 수정일, 수정자, 삭제여부	팀 후기 답변	후기 번호를 참조하여 기준에 맞게 수정여부를 표시함	
팀원	팀원	팀원 ID(PK), 팀 아이디(PK), 팀원아이디(PK), 팀원명, 생성일, 생성자, 수정일, 수정자, 나감여부	팀원_소개	회원 정보를 참조해서 팀원 정보에 저장, 팀원 권한은 BOOLEAN	
팀 보조선	팀 필요 보조선	필요_보조선_ID(PK), 팀 ID(FK), 보조선_ID(FK)	팀별 필요 보조선 헤이블	팀 ID, 보조선ID 를 복합 UNIQUE로 정의해서 같은팀이 같은 보조선 신경하지 못하도록	
팀 가입 신청	팀 가입 신청 승인/반려	팀_가입_신청_ID(PK), 팀_가입_신청_내용_ID(FK), 팀 ID(FK), 팀_가입_신청_상태, 생성일, 생성자, 수정일, 수정자, 삭제 여부	가입 신청	팀 가입 신청 상태를 ENUM{approved, pending, denied}	
커뮤니티	사용자들이 게시글을 작성하거나 다른 사용자들이 작성한 게시글을 조회	id: 게시글 ID (PK) title: 게시글 제목 content: 게시글 내용 (TEXT, NonNull) category: 게시글 카테고리 (String, NonNull) isDeleted: 삭제 여부 (Boolean, NonNull) pinned: 고정 여부 (Boolean, NonNull)	게시판	공지사항 게시판, 자유게시판, 정보 게시판으로 카테고리로 분류	
댓글	게시글에 대한 사용자 댓글	id: 댓글 ID (PK) body: 댓글 내용 (Text, NonNull) member: 작성자 (PK, NonNull) content: 댓글 내용 (Text, NonNull) isDeleted: 삭제 여부 (Boolean, NonNull)	코멘트		
신고	커뮤니티 내에서 규정을 위반한 사용자를 신고 및 관리자 모형, 신고자와 신고 대상, 신고 사유, 분류, 상세 등을 서정	id: 신고 ID (PK) reporter: 신고한 유저 (Non, NonNull) target: 신고 대상 (Non, NonNull) reasonType: 신고 사유 (Non, NonNull) targetsAdmin: 대상이 관리자 여부 (false) targetMemberId: 신고 대상 회원 ID (nullable) targetUserName: 신고 대상 회원 이름 (nullable)	리포트	같은 대상의 중복 신고 가능	

ERD

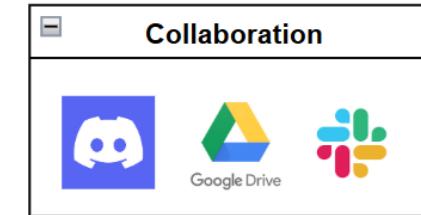
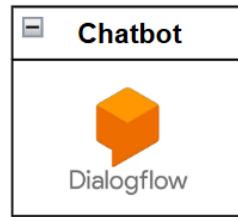
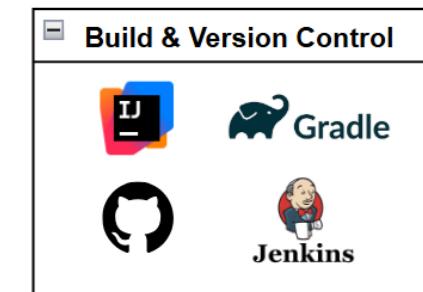
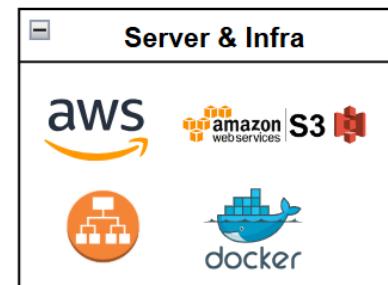
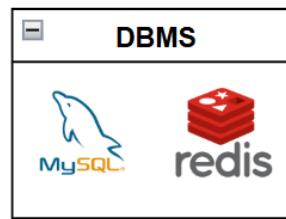
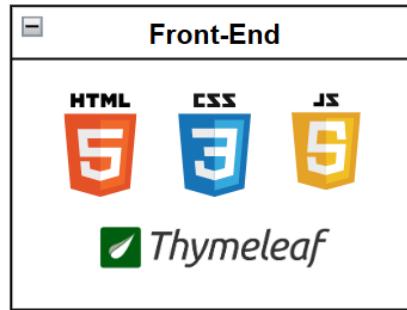
ERD 링크



화면 설계

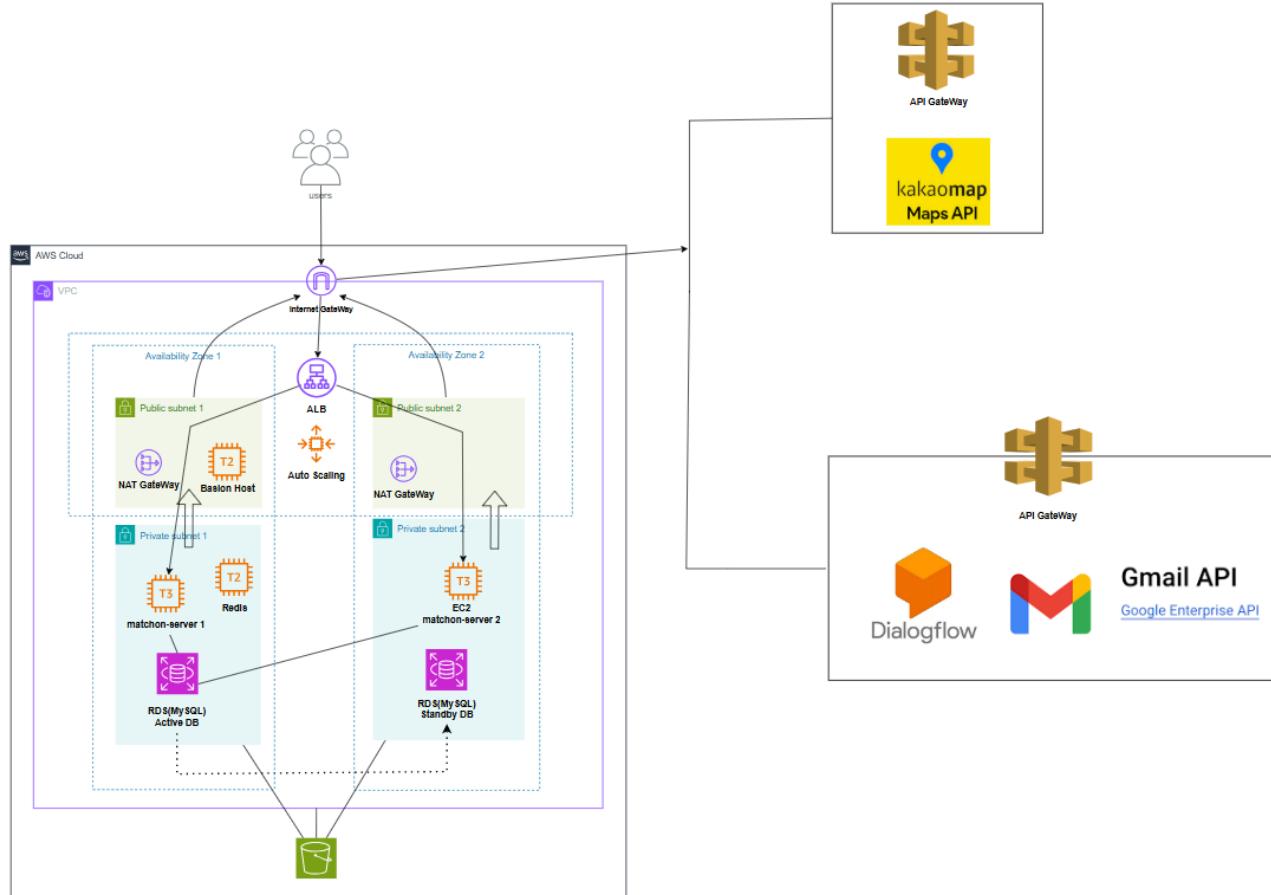


기술 스택



시스템 설계도

AWS 배포 링크



주요 기능 - 회원정보

사용자

SIGN UP

사용자 이메일 @naver.com

중복 확인

사용 가능한 이메일입니다.

확인

※ 8자 이상, 대문자/소문자/숫자/특수문자 포함, 같은 문자 3번 이상 불가 사용 가능한 비밀번호입니다.

이름

이메일 수신에 동의합니다.

회원가입

사용자

SIGN UP

사용자 이메일 @naver.com

중복 확인

사용 가능한 이메일입니다.

확인

※ 8자 이상, 대문자/소문자/숫자/특수문자 포함, 같은 문자 3번 이상 불가 비밀번호에 숫자를 포함해야 합니다.

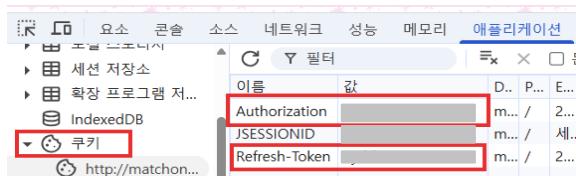
이름

이메일 수신에 동의합니다.

회원가입

회원가입 성공

회원가입 실패



마이페이지

사용 등록

회원 탈퇴

탈퇴하기

이름

활동 시간대

위치

라이온도

소속팀

이메일 수신 동의

수정하기

// accessToken → HttpOnly 쿠키로 발급 (JS 접근 불가)
 ResponseCookie accessTokenCookie = ResponseCookie.from("Authorization", tokenResponse.getAccessToken())
 .httpOnly(false)
 .path("/")
 .maxAge(Duration.ofHours(1))
 .build();

// refreshToken HttpOnly 쿠키로 발급
 ResponseCookie refreshTokenCookie = ResponseCookie.from("Refresh-TOKEN", tokenResponse.getRefreshToken())
 .httpOnly(true)
 .path("/")
 .maxAge(Duration.ofDays(14))
 .build();

- 로그인시 **accesstoken, refresh token** **httponly**로 발급
- 로그아웃시 자동 삭제(DB에서도 삭제)
- 삭제되지 않은 계정 기준으로 이메일 중복 체크, 비밀번호 조건 함수
- 회원 탈퇴시 토큰삭제, 개인정보 초기화, S3 삭제(**DB는 존재-softdelete**), 재로그인시 **탈퇴계정 확인**

주요 기능 - 마이페이지

마이페이지

이름 **이름**

활동 시간대 **별일 오전**

포지션 **글키퍼**

화이온도 **36.5 °C**

소속팀 **팀이 없습니다**

이메일 수신 동의

수정하기

1

사용자

마이페이지

이름 **이름**

주최 기관 이름 **기관명 입력** **등록** **대회 등록**

이메일 수신 동의

2

주최자

```
public void uploadProfileImage(Member member, MultipartFile file) { 1 usage  ▲ Juhee Hong

    // 확장자 검사
    String ext = FilenameUtils.getExtension(file.getOriginalFilename()).toLowerCase();
    if (!List.of("jpg", "jpeg", "png").contains(ext)) {
        throw new IllegalArgumentException("jpg, jpeg, png 파일만 업로드할 수 있습니다.");
    }

    // 1. 기존 첨부 파일이 있으면 삭제
    Optional<Attachment> existingAttachmentOpt =
        attachmentRepository.findLatestAttachment(BoardType.MEMBER, member.getId());

    existingAttachmentOpt.ifPresent( Attachment att -> {
        awsS3Utils.deleteFile(att.getSavePath(), att.getSavedName());
        attachmentRepository.delete(att); // DB에서도 삭제
    });
}
```

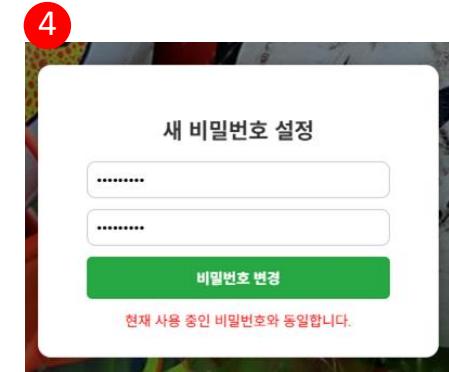
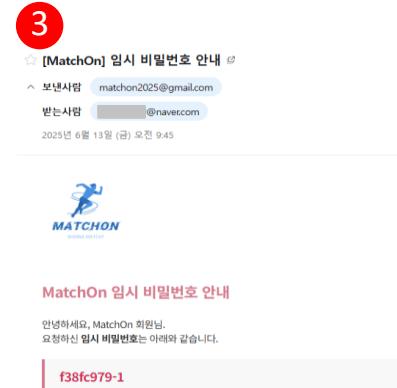
액체 (2) **C** **S3 URI 복사** **URL 복사** **다운로드** **열기** **삭제** **작업** **폴더 만들기** **업로드**

액체는 Amazon S3에 저장되어 있는 기본 엔터티입니다. Amazon S3 인벤토리 [?]를 사용하여 버킷에 있는 모든 객체의 목록을 얻을 수 있습니다. 다른 사용자가 액체에 액세스할 수 있게 하려면 명시적으로 권한을 부여해야 합니다. [자세히 알아보기](#) [?]

접두사로 액체 찾기	1 >	1 <		
이름	유형	마지막 수정	크기	스토리지 클래스
attachments/	폴더	-	-	-
community/	폴더	-	-	-

- **s3 attachments/profile/폴더에 s3 presignedURL** 이용해 프로필 이미지 조회
- 활동시간대, 포지션(enum으로 저장된 데이터 중 사용자가 직접 변경) 변경
- 존재하지 않는 주최기관명 등록, hostprofile 테이블에서 중복검사

주요 기능 - 임시비밀번호 발송



1 2

```

@PostMapping(value = "/reset-password") ▲ Juhe Hong
public ResponseEntity<String> sendTemporaryPassword(@RequestBody Map<String, String> request) {
    String email = request.get("email");
    Optional<Member> optional = memberRepository.findByIdAndDeletedFalse(email);

    if (optional.isEmpty()) {
        return ResponseEntity.badRequest().body("존재하지 않는 이메일입니다.");
    }

    String tempPassword = UUID.randomUUID().toString().substring(0, 10);
    String encoded = passwordEncoder.encode(tempPassword);

    Member member = optional.get();
    member.setTemporaryPassword(encoded); // 기존 비밀번호는 유지
    member.setIsTemporaryPassword(true); // 풀리고 ON
    memberRepository.save(member);

    mailService.sendTemporaryPassword(email, tempPassword);
    return ResponseEntity.ok(body: "임시 비밀번호가 이메일로 전송되었습니다.");
}

```

3

```

@Async("asyncExecutor") ▲ Juhe Hong +1
public void sendTemporaryPassword(String toEmail, String tempPassword) {
    if (!isEmailValid(toEmail)) {
        log.warn("임시 비밀번호 메일 전송 실패: 유효하지 않은 이메일 주소 {}", toEmail);
        return;
    }

    MimeMessage message = mailSender.createMimeMessage();
    MimeMessageHelper helper = new MimeMessageHelper(message, multipart: true, encoding: "UTF-8");

    helper.setTo(toEmail);
    helper.setSubject("[MatchOn] 임시 비밀번호 안내");

```

4

```

// 로그인 후 새 비밀번호로 변경
@PostMapping(value = "/change-password") ▲ Juhe Hong
public ResponseEntity<String> changePassword(
    @AuthenticationPrincipal CustomUser user,
    @RequestBody Map<String, String> request) {
    String newPassword = request.get("newPassword");
    String confirmPassword = request.get("confirmPassword");

    Member member = user.getMember();
    authService.changePassword(newPassword, confirmPassword, member);

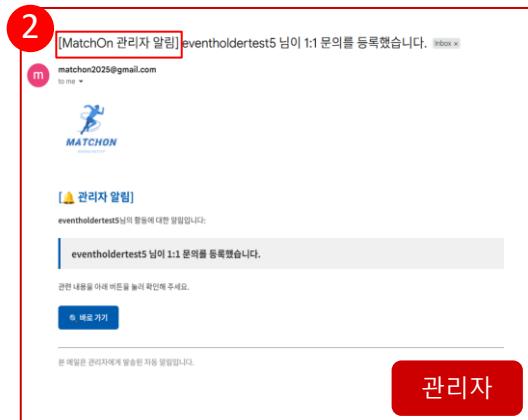
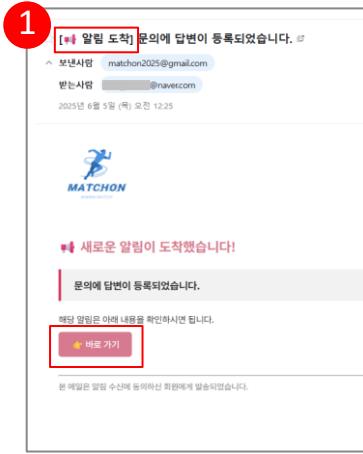
    return ResponseEntity.ok(body: "비밀번호가 성공적으로 변경되었습니다.");
} catch (IllegalArgumentException e) {
    return ResponseEntity.badRequest().body(e.getMessage());
}

private boolean isValidPassword(String password) { no usages ▲ Juhe Hong
    return password.matches(regex: "(?=.*[A-Z])(?=.*[a-z])(?=.*[\\d])(?=.*[!#$%^&*()]).{8,$)");
}

```

- 비밀번호 분실 시 메일 수신 동의한 이용자만 가입한 **이메일로 임시비밀번호 메일 발송**
- 발급된 임시 비밀번호로 로그인시 임시 비밀번호로 로그인 이력이 확인되며 **새 비밀번호 설정 팝업**이 반드시 뜸
- 새 비밀번호 설정시 서버에 저장된 **이전 비밀번호**, **임시비밀번호**가 저장되므로 사용하지 않은 비밀번호로 설정하도록 구현

주요 기능 - 메일 알림 서비스



```
// 이메일 통의
@PutMapping("/email-agreement")
public ResponseEntity<String> updateEmailAgreement(@AuthenticationPrincipal CustomUser user,
                                                       @RequestBody Map<String, Boolean> payload) {
    boolean agreement = payload.getOrDefault("emailAgreement", false);
    Member member = memberService.findByEmail(user.getUsername());
    mypageService.updateEmailAgreement(member, agreement);
    return ResponseEntity.ok("변경 완료");
}

// 이메일 유통성 검사
private boolean isEmailValid(String email) {
    try {
        String domain = email.substring(email.indexOf('@') + 1);
        javax.naming.directory.InitialDirContext ctx = new javax.naming.directory.InitialDirContext();
        javax.naming.directory.Attributes attrs = ctx.getAttributes("dns://" + domain, new String[]{"MX"});
        return attrs != null && attrs.get("MX") != null;
    } catch (Exception e) {
        log.warn("이메일 유통성 검사 실패 또는 MX 레코드 없음: {}", email);
        return false;
    }
}

// 관리자용 알림
@Transactional
public void notifyAdmins(String content, String url, Member sender) {
    List<Member> admins = memberRepository.findByMemberRoleAndIsDeletedFalse(MemberRole.ADMIN);
    String senderName = sender.getMemberName();

    for (Member admin : admins) {
        sendNotificationWithoutMail(admin, content, url);
    }

    // 예외 처리
    if (Boolean.TRUE.equals(admin.getEmailAgreement())) {
        mailService.sendAdminNotificationEmail(
            admin.getMemberEmail(),
            content,
            mailService.buildAdminNotificationBody(senderName, content, url)
        );
    }
}

// 예외 처리
notificationService.notifyAdmins(content, user.getMember().getMemberName() + "님이 1:1 문의를 등록했습니다.", url + "/admin/inquiry", user.getMember());
return "redirect:/inquiry";
}
```

- (Gmail API키 이용) 사전 이메일 수신 동의한 사용자에게 **비동기 방식(AsyncConfig)**으로 이메일 전송
- 유효한 이메일 검사, **redirecturl** 생성해 바로가기 버튼 클릭 시 해당 알림 페이지로 이동
- 관리자용 알림 사용할 기능 컨트롤러에 따로 **호출해서 사용**

주요 기능 - 대회일정

1. 대회 일정

2. 대회 등록

3. 대회 등록 내역

4. 관리자

```
#PostMapping(value = "/event/{id}/status")
public String updateEventStatus(@PathVariable Long id, @RequestParam("status") Status status) {
    eventRepository.updateEventStatus(id, status);
    EventRequest event = eventRepository.findById(id);
    .orElseThrow(() -> new IllegalStateException("해당 대회를 찾을 수 없습니다."));
}

// 만약 on 만드는 시 등록 가능
if (status == Status.APPROVED || status == Status.DENIED) {
    String message = (status == Status.APPROVED) ? "대회가 승인되었습니다." : "대회가 반려되었습니다.";
    notificationService.sendNotification(
        event.getMember(),
        message,
        message.replace("{event}/" + event.getId());
}
}

// 승인 상태가 아닌 경우 삭제 불가
if (event.getEventStatus() != Status.APPROVED) {
    throw new IllegalStateException("승인된 대회만 삭제할 수 있습니다.");
}
```

- 관리자가 승인 or 반려시 상태 실시간 변경
- **승인된 대회만** 삭제 가능

```
label.addEventListener("mouseover", listener: (event: MouseEvent) => {
    tooltip.innerHTML =
        `> ${e.eventTitle}\n` +
        `■ ${localDateStr}\n` +
        `■ ${e.hostName} || '대회'\n` +
        `▶ ${e.eventAddress} || '장소'\n` +
        `■ ${e.eventMethod} || '방법'\n` +
        `● ${e.eventContact} || '연락처'\n`;
    tooltip.style.display = 'block';
});
```

- 승인된 대회는 달력에 등록
- **마우스 오버**시 툴팁 사용해 요약된 내용 확인
- 라벨 클릭시 대회 상세페이지 이동



```
Member member = customer.getMember();
Optional<HostProfile> optionalHostProfile = hostProfileRepository.findById(member);
if(optionalHostProfile.isPresent()) {
    redirectAttributes.addFlashAttribute("attributeName", "errorMessage", "▲ 주최자에게 미등록 고객입니다. 미이벤트에서 먼저 등록해주세요.");
    return "redirect:/page";
}

<script src="//ti.daumcdn.net/mapjsapi/bundle/postcode/prod/postcode.v2.js"></script>
```

- 주최기간 미등록시 대회등록 불가, 마이페이지로 redirect,
- 지도 api키 이용해 대회 장소 등록 구현 -> 등록시 등록 내역에서 내 등록내역 확인
- 대회 **반려**시 삭제 가능

주요 기능 - 1:1 문의

1 내 문의 내역

2 문의 작성

3 대회일정 어떻게 보나요?

4 전체 문의 목록

5 대회일정 어떻게 보나요?

사용자 & 주최자

```
@GetMapping("/inquiry") ▲ Juhee Hong
public String listInquiries(@RequestParam("q") String keyword,
                           @RequestParam("category") String category,
                           @RequestParam("page") int page,
                           @RequestParam("size") int size) {
    Page<InquiryResDto> inquiries = inquiryService.listInquiries(keyword, getMember(), pageable, category, page);
    model.addAttribute("category", "inquiries");
    model.addAttribute("categoryList", categoryService.listCategory());
    model.addAttribute("category", category);
    model.addAttribute("categoryList", categoryService.listCategory());
    if (!"Y".equals(keyword)) {
        return "list/inquiry/inquiry-list";
    }
    return "list/inquiry/inquiry-list";
}

@GetMapping("/inquiry/{id}") ▲ Juhee Hong
public String answer(@PathVariable Long id) {
    model.addAttribute("inquiry", new InquiryResDto());
    return "edit/inquiry/inquiry-answer";
}
```

```
@Transactional 1 usage ▲ Juhee Hong
public void deleteCompletedInquiry(Long id) {
    Inquiry inquiry = inquiryRepository.findById(id).orElseThrow();
    if (inquiry.gettingStatus() == Status.COMPLETED) {
        inquiry.markDeleted();
    } else {
        throw new IllegalStateException("답변 완료된 문의만 삭제할 수 있습니다.");
    }
}
```

- 3 카테고리, 검색어로 문의 검색 가능
- 작성 시 카테고리 선택 / 제목 100자 제한
 - 답변이 완료된 경우에만 문의 삭제 가능

4 관리자

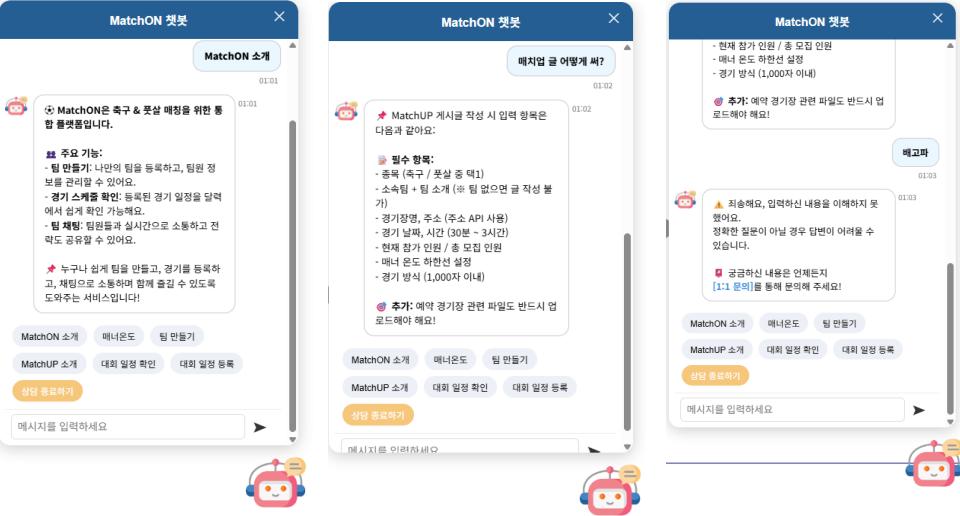
5 대회일정 어떻게 보나요?

```
@GetMapping("/{id}/inquiry/{id}/answer") ▲ Juhee Hong
public String getInquiryDetail(@PathVariable Long id, Model model) {
    Inquiry inquiry = inquiryRepository.findByIdAndIsDeletedFalse(id)
        .orElseThrow(() -> new NoSuchElementException("문의 없음"));
    InquiryResDto dto = new InquiryResDto(inquiry); // DTO 변환
    model.addAttribute("inquiryName", "inquiry", dto);
    return "admin/admin-inquiry-answer";
}

@PostMapping("/{id}/inquiry/{id}/answer") ▲ Juhee Hong
@Transactional
public String submitAnswer(@PathVariable Long id,
                           @RequestParam String answerContent,
                           @AuthenticationPrincipal CustomUser admin) {
    Inquiry inquiry = inquiryRepository.findByIdAndIsDeletedFalse(id)
        .orElseThrow(() -> new NoSuchElementException("문의가 존재하지 않습니다."));
    // 답변 존재 여부만으로 판단
    if (inquiryAnswerRepository.findActiveAnswerByInquiryId(inquiry.getId()).isPresent()) {
        throw new IllegalStateException("이미 답변이 등록은 했었습니다.");
    }
    InquiryAnswer answer = InquiryAnswer.builder()
        .inquiry(inquiry)
        .member(admin.getMember())
        .answerContent(answerContent)
        .build();
}
```

- 4 관리자는 등록된 문의 내역 조회 가능
- 5 답변은 답변 존재 여부 판단 후 답변 가능 (문의가 대기중 상태)

주요 기능 - AI챗봇(매치봇)



Dialogflow Essentials

- MatchON Bot
- Intents
- Entities
- Knowledge
- Fulfillment
- Integrations
- Training
- Validation
- History
- Analytics
- Predictive Agents
- Docs?

Intents

MatchUP 소개

Action and parameters

Enter action name: Default Fallback Intent

Responses

Text Response

1 MatchUP은 경기 일정과 팀 정보를 제공하는 서비스입니다. 팀 이름이나 경기 날짜, 장소 등으로 검색할 수 있습니다. 2 다른 서비스와 연동하여 팀 일정을 확인하거나 경기 결과를 알 수 있습니다.

```

@Configuration
public class DialogflowConfig {
    @Value("${app/keys/dialogflow-key.json}")
    private String dialogflowKeyPath;

    @Bean
    public SessionsClient sessionsClient() throws IOException {
        // 파일 경로의 FileInputStream 초기화
        FileInputStream serviceAccountStream = new FileInputStream(dialogflowKeyPath);

        // 구글 인증서체 설정
        GoogleCredentials credentials = GoogleCredentials.fromStream(serviceAccountStream);

        // 세션 설정의 인증서체 적용
        SessionsSettings sessionsSettings = SessionsSettings.newBuilder()
            .setCredentialsProvider(FixedCredentialsProvider.create(credentials))
            .build();

        return SessionsClient.create(sessionsSettings);
    }
}

@Controller
@RequiredArgsConstructor
public class MatchController {
    private final DialogflowService dialogflowService;
}

@PutMapping("api/aichat")
public ResponseEntity<Map<String, String>> chat(@RequestBody Map<String, String> payload, HttpSession session) {
    String sessionId = session.getId();
    String message = payload.get("message");
    String responseText = dialogflowService.detectIntent(sessionId, message);

    Map<String, Object> response = new HashMap<>();
    response.put("reply", responseText);
    return ResponseEntity.ok(response);
}

@GetMapping("api/aichat")
public String chatPage() { return "chatbot"; }

```

Google Dialogflow API 키 이용

- 서비스에 해당되는 내용 intent에 훈련
- 해당 질문에 맞는 Text Response 설정
- dialogflowkey.json project에서 저장해 연동
- "/api/chat" 주소로 dialogflow에 저장된 intent 내용 불러옴.
- 'chips'를 이용해 고정 질문 버튼 생성

```


<button onclick="sendChip('MatchON 소개')">MatchON 소개</button>
    <button onclick="sendChip('매너온도')">매너온도</button>
    <button onclick="sendChip('팀 만들기')">팀 만들기</button>
    <button onclick="sendChip('MatchUP 소개')">MatchUP 소개</button>
    <button onclick="sendChip('대회 일정 확인')">대회 일정 확인</button>
    <button onclick="sendChip('대회 일정 등록')">대회 일정 등록</button>
    <button onclick="sendChip('상담 종료하기')">상담 종료하기</button>


```

주요 기능 - 참가 인원 모집과 참가 요청

MATCHUP 글 작성

종목
FUTSAL

소속팀
GOGO

팀 소개(300자)
안녕하세요. 저희는 즐겁게 저녁에 모여서 간단한 경기하는 것을 목적으로 하는 팀이에요. 다른 오셔서 재미있는 경기 같아해요~

경기장 예약 내역
[파일 선택](#) 예약내역.pdf

경기장명(100자)
고덕풋살장

경기장 주소(100자)
서울 강동구 아리수로 185

경기 날짜
2025-06-20 오후 08:00

경기 진행 시간
1시간 30분

현재 참가 인원
1

총 모집 인원
12

입장 가능 매너 온도(내 온도: 36.9)
32

경기 방식 소개(1000자)
안녕하세요. 간단하게 풋살 경기하고자 합니다.
시간은 전반 후반 각각 30분씩
인원은 5대로 험하고자합니다.

등록 **취소**

MATCHUP 참가 요청 작성

종목
FUTSAL

작성자
Member10

경기장명
고덕풋살장

경기장 주소
서울 강동구 아리수로 185

경기장 위치

경기 날짜
6/20 20시 0분 - 21시 30분

현재 정원
(1 / 12)

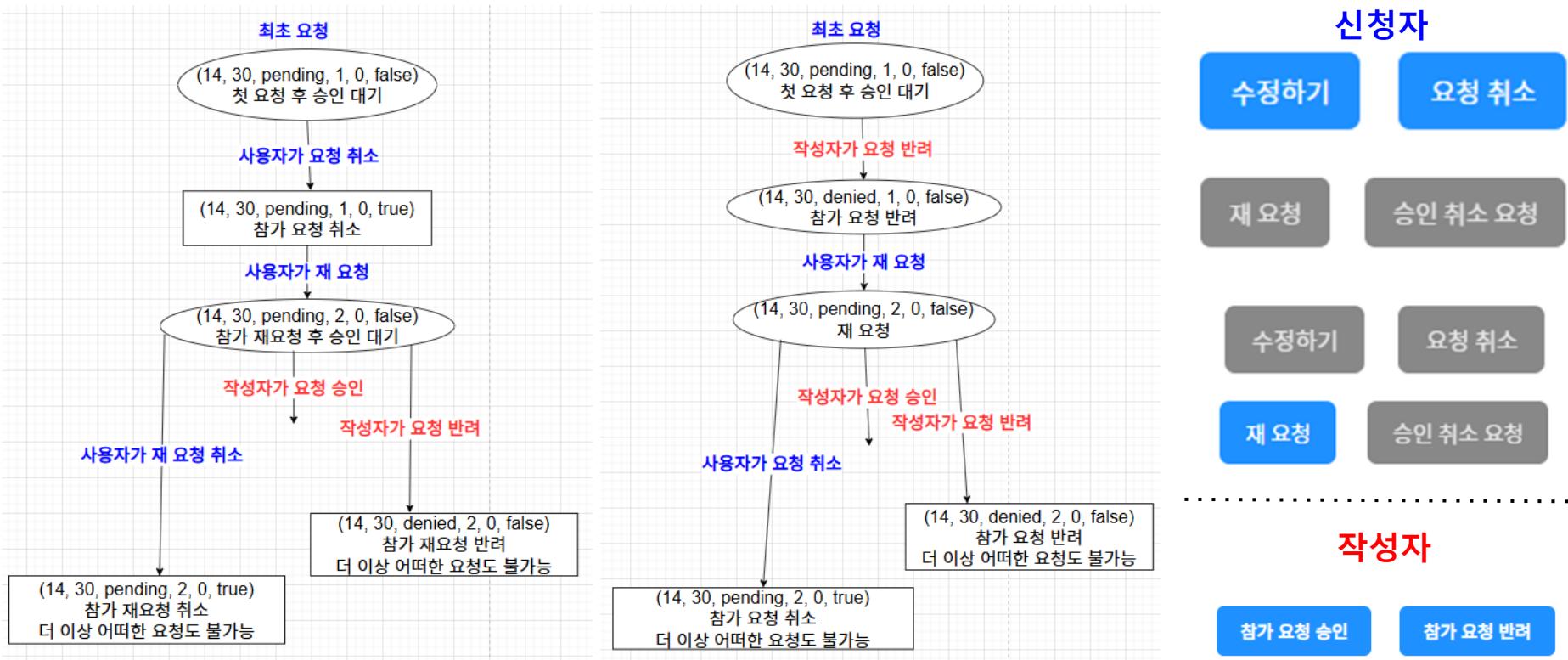
소개(300자)
안녕하세요. 저희는 축 3팀 참가하고자 합니다.
저희 모두 즐겁게 게임하자면 마이드루 참가해주세요...

참가 인원
3

요청하기 **취소하기**

- 게시글 작성은 하루 최대 2번
- 작성자에게 1대1 채팅으로 문의 가능
- 입장 가능 매너 온도를 설정
- 참가 요청 승인 후에는 그룹 채팅방 참여
- 참가 요청은 최대 2회
- 승인 취소 요청은 최대 1회
- 경기 시간(시작~종료)이 겹치면, 다른 게시글 작성이나 참가 요청은 불가능
- 경기 종료 후 참가자들간에 매너 온도 평가

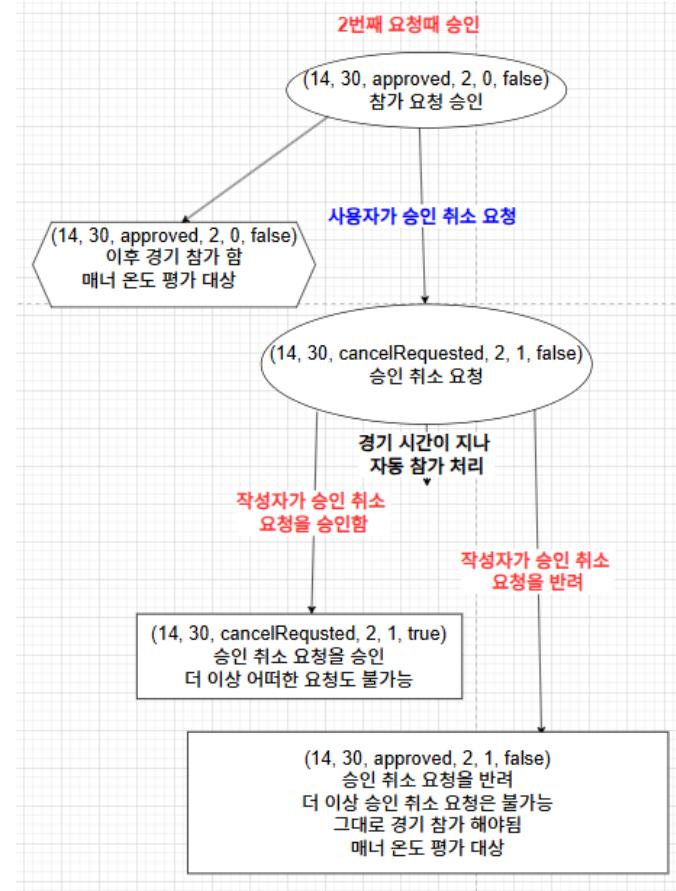
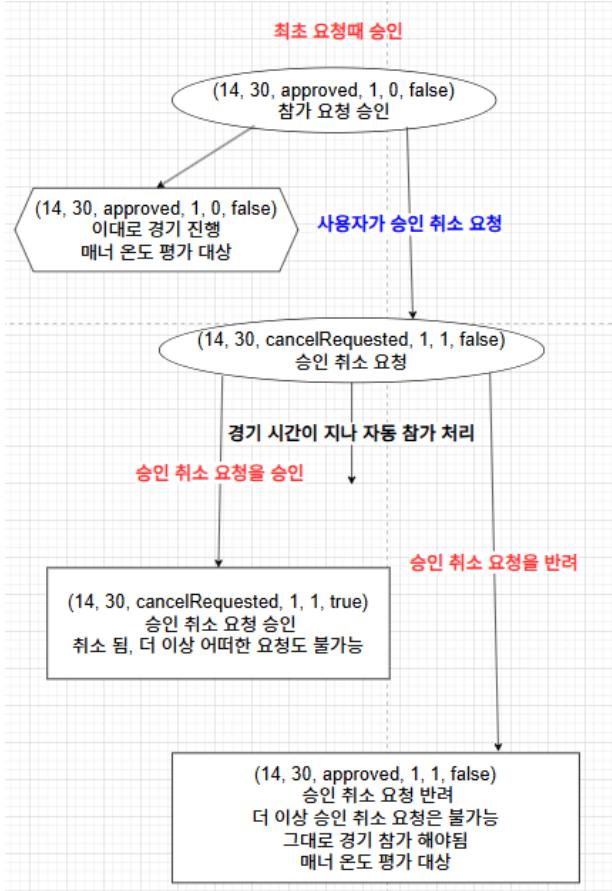
주요 기능 - 참가 요청 승인 전 단계



- (게시글 번호, 요청 번호, 상태, 요청 횟수, 취소 횟수, isDeleted)
- 상태, 요청 횟수, 취소 횟수, isDeleted 총 4가지 정보를 기반으로 승인 과정을 관리
- 총 요청 횟수는 2회, 승인 전 상태는 PENDING, DENIED



주요 기능 - 참가 요청 승인 후 단계



- (게시글 번호, 요청 번호, 상태, 요청 횟수, 취소 횟수, isDeleted)
- 취소 요청 횟수는 1회
- 승인 후 단계 상태는 APPROVED, CANCELREQUESTED
- 취소 요청을 했지만, 경기 시간이 지난 경우 자동 참가 처리

신청자

수정하기

재 요청

요청 취소

승인 취소 요청

작성자

참가 요청 승인

참가 요청 반려

취소 요청 승인

취소 요청 반려

주요 기능 - 매너 온도 평가

상대방	받은 매너 점수	받은 실력 점수	받은 후기
Member1	N	N	<button>받은 후기</button>
Member2	4	4	<button>받은 후기</button>

상대방	보낸 매너 점수	보낸 실력 점수	보낸 후기
Member1	5	5	<button>보낸 후기</button>
Member2	N	N	<button>후기 작성</button>

리뷰 작성 하기

리뷰(300자)
리뷰를 입력하세요.

보내기
창닫기

리뷰 상세 보기

후기
다음에도 같이 경기해요~

창닫기

```
@Lock(LockModeType.PESSIMISTIC_WRITE) 1 usage ↗ hyomin-dev
```

```
@Query(""
```

```
    select t1
    from Member t1
    where t1.id=:id and t1.isDeleted=false
    "")
```

```
Optional<Member> findByIdAndIsDeletedFalseWithLock(@Param("id") Long id);
```

```
/*
```

```
* 매너 온도 평가 등록, 마이페이지 업데이트
```

```
*/
```

```
@Transactional 2 usages ↗ hyomin-dev
```

```
public void registerMatchupRating(
```

```
    @Valid ReqMatchupRatingDto reqMatchupRatingDto,
    CustomUser user) {
```

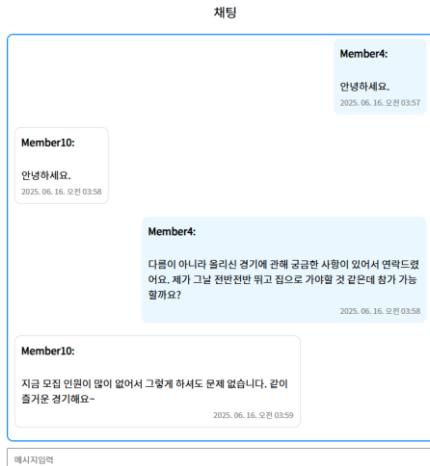
$$\text{온도 변화 값} = (\text{매너 점수} \times 0.14) + (\text{실력 점수} \times 0.06)$$

$$\text{반영 온도 변화} = (\text{온도 변화 값} - 0.4) \times 0.01$$

$$\text{갱신된 매너 온도} = \max(30, \text{기존 매너 온도} + \text{반영 온도 변화})$$

주요 기능 - STOMP 기반의 실시간 채팅

chat



group chat



입장 **차단** **→** **입장** **해제**

⚠️ 상대방에게 메시지를 보낼 수 없습니다.

메시지입력

전송

채팅 공통

- STOMP 기반의 실시간 채팅
- 채팅방별로 참가자들을 관리
- 주고 받은 메시지 DB에 저장
- 채팅방 재 입장 시 DB에서 기존 메시지 자동 조회

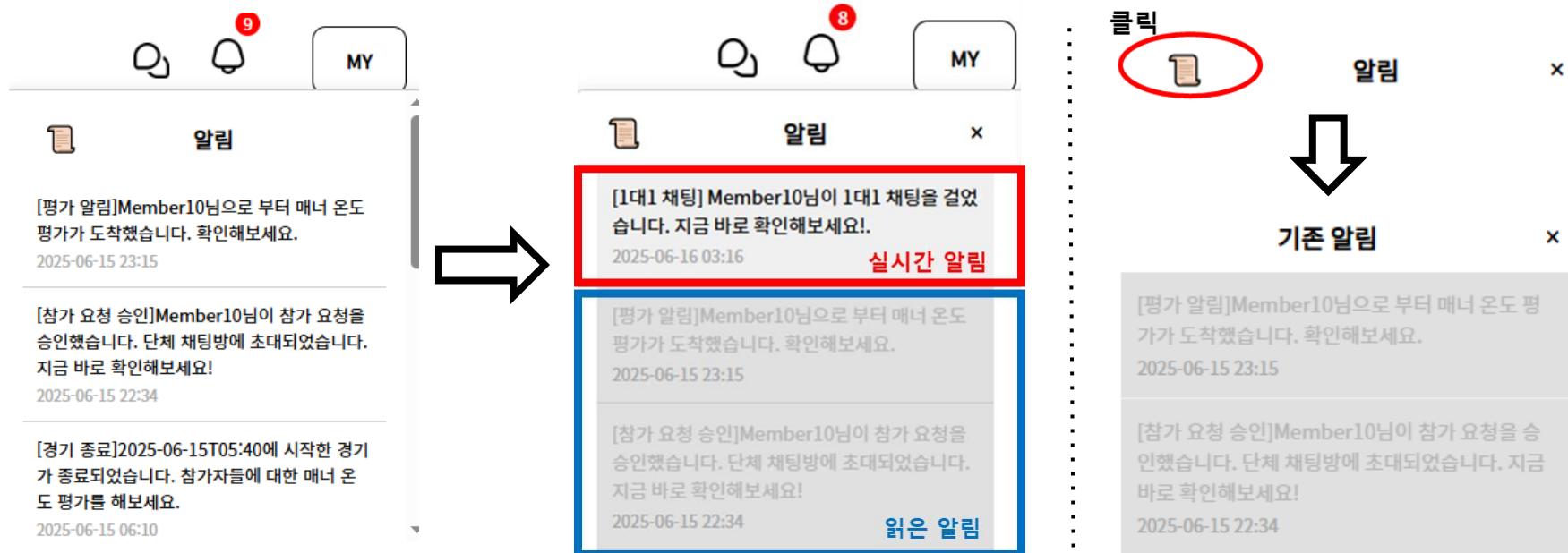
1대1 채팅

- 1대1 채팅에 차단/차단 해제 기능 도입
- 차단 당하면 상대방에게 메시지를 보낼 수 없고 보내면 “상대방에게 메시지를 보낼 수 없습니다”가 시스템이 답장
- chat_user_block table을 통해 1대1 채팅유저간에 차단여부를 관리

그룹 채팅

- 채팅방내에서 참가자 목록 열람
- 참가자들간에 자유롭게 1대1 채팅 가능

주요 기능 - STOMP 기반의 실시간 알림



이 알림과 관련된 페이지로 이동하시겠습니까?

예

아니요

- STOMP 기반의 실시간 알림
- notification table을 이용해 수신자, 생성자, 알림 내용, 이동 url, 읽음 여부를 관리
- 새로운 알림을 위한 사이드 패널과 그 동안 읽은 알림을 위한 사이드 패널을 이용
- 메시지 클릭 시 읽음 처리 및 페이지 이동 알림

주요 기능 - 커뮤니티

1. 자유게시판

2. 게시글 상세조회

3. 게시글 작성

4. 게시글 수정

5. 게시글 삭제

1. 게시글 목록

```

public String list(Model model) {
    List<Board> boards = boardRepository.findAll();
    model.addAttribute("boards", boards);
    return "list";
}

```

2. 게시글 상세보기

```

public String detail(@PathVariable long id, Model model) {
    Board board = boardRepository.findById(id);
    model.addAttribute("board", board);
    List<Attachment> attachments = attachmentRepository.findByBoardType(BOARD, board.getId());
    model.addAttribute("attachments", attachments);
    model.addAttribute("commentCount", new CommentRepository().getCommentCount(board));
    model.addAttribute("commentList", commentRepository.getCommentListByBoard(board));
    return "detail";
}

```

3. 게시글 작성

```

@PostMapping("/write")
public String write(@ModelAttribute Board board) {
    boardRepository.save(board);
    return "redirect:/list";
}

```

4. 게시글 수정

```

@PutMapping("/update")
public String update(@ModelAttribute Board board) {
    boardRepository.save(board);
    return "redirect:/list";
}

```

5. 게시글 삭제

```

@DeleteMapping("delete/{id}")
public String delete(@PathVariable long id, AuthenticationPrincipalCustomer user) {
    Attachment attachment = attachmentRepository.findById(id);
    attachment.setDeleted(true);
    attachmentRepository.save(attachment);
    Board board = boardRepository.findById(attachment.getBoardId());
    if (board.getAuthor().getId() == user.getId()) {
        return "redirect:/list";
    }
    attachment.setDeleted(true);
    attachmentRepository.delete(attachment);
    String fileName = attachment.getFileName();
    int fileIndex = fileName.lastIndexOf(".");
    String fileExt = fileName.substring(fileIndex + 1);
    String file = attachment.getOriginalFileName();
    String fileExt2 = file.substring(fileIndex + 1);
    if (!file.equals(fileName)) {
        attachment.setFileName(file);
        attachmentRepository.save(attachment);
    }
    return "redirect:/list";
}

```

1. 커뮤니티 전체조회

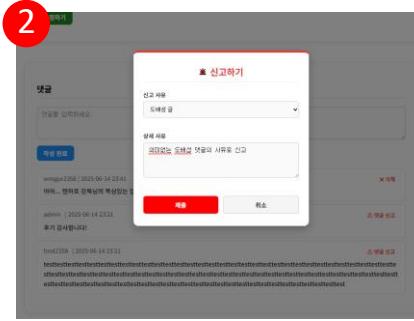
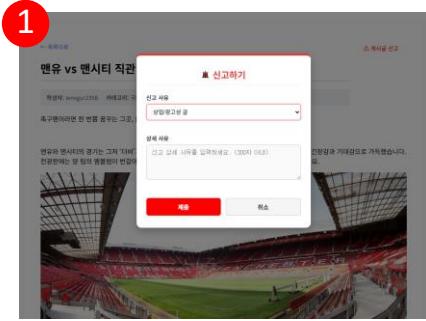
2. 게시글 상세조회

3. 게시글 작성

4. 게시글 수정

5. 게시글 삭제

주요 기능 - 사용자 신고



```

1 // 신고 접수, 일반 사용자 (POST /community/reports)
2 @PostMapping("/community/reports")
3 @PreAuthorize("isAuthenticated()")
4 public ResponseEntity<?> reportContent(
5     @RequestParam ReportType type,
6     @RequestParam Long targetId,
7     @RequestParam String reason,
8     @RequestParam ReasonType reasonType,
9     @AuthenticationPrincipal CustomUser user
10 ) {
11     if (user == null) {
12         return ResponseEntity.status(401).body("로그인이 필요합니다.");
13     }
14
15     try {
16         reportService.report(type, targetId, reason, reasonType, user.getMember());
17         return ResponseEntity.ok("신고가 접수되었습니다.");
18     } catch (IllegalStateException e) {
19         return ResponseEntity.badRequest().body(e.getMessage());
20     }
21 }

```

신고 목록								
ID	유형	신고대상	신고자	신고분류	상세내용	신고일	보기	회원 처리
1	COMMENT	host2358	admin	도배싱 글	의미없는 도배싱 글의 자유로 제재를 요청	2025-06-14 23:57	개설	7회 30회 정우
2	COMMENT	wngsur2358	admin	상업/광고성 글	상업성의 자유로 신고함	2025-06-14 18:08	개설	7회 30회 정우
3	BOARD	wngsur2358	admin	욕설/비방/괴롭힘	신고테스트	2025-06-14 18:08	개설	7회 30회 정우
4	COMMENT	Member1	host2358	카테고리 취지에 어긋남	관리자가 아닌 회원이 끌린 공지사항을 단 댓글인데 신고 부탁드립니다!	2025-06-14 09:04	개설	7회 30회 정우
5	BOARD	Member1	host2358	카테고리 취지에 어긋남	관리자가 아닌데 공지사항에 게시글을 올렸어요	2025-06-14 09:03	개설	7회 30회 정우

1 2 3 4 5 다음

```

1 // 관리자를 신고 목록 페이지 (GET /admin/reports/page)
2 @GetMapping("/admin/reports/page")
3 @PreAuthorize("hasRole('ADMIN')")
4 public String reportManagePage(
5     @RequestParam("page") int page,
6     @RequestParam("defaultValue" = "0") ReportType reportType,
7     @RequestParam("required" = false) ReasonType reasonType,
8     @ModelAttribute Model model
9 ) {
10     Pageable pageable = PageRequest.of(page, 5).Sort by("createdDate").descending();
11     Page<Report> reportPage = reportService.getpagedReportsWithFilter(pageable, reportType, reasonType);
12
13     model.addAttribute("reportPage", reportPage);
14     model.addAttribute("reports", reportPage.getContent());
15     model.addAttribute("currentPage", page);
16     model.addAttribute("reportType", reportType != null ? reportType.name() : "");
17     model.addAttribute("reasonType", reasonType != null ? reasonType.name() : "");
18 }

```

1.게시글 신고

2.댓글 신고

3.신고내역 조회 (관리자 기능)

주요 기능 - 관리자 기능(커뮤니티)

1

2

2

3

2

3

4

4

1. 관리자 신고목록 조회
2. 신고 게시글/댓글 추적+ 관리자 삭제
3. 회원 정지 및 해제 처리
4. 사용자 시점 회원 정지 확인

주요 기능 - FAQ

FAQ

자주 묻는 질문

1:1 문의

No.	카테고리	제목	작성일
1	Team / Guest	팀 채팅 안내	2025-06-14
2	개정	개인정보 수정하는 법	2025-06-14
3	신고	신고 방법	2025-06-14
4	Team / Guest	Guest 이용 방법	2025-06-14
5	이용가이드	MatchON 이용 방법	2025-06-14
6	매너온도	매너온도 계산 기준 안내	2025-06-14
7	이용가이드	챗봇 이용 방법	2025-06-14
8	신고	멤버 신고 안내	2025-06-14

FAQ 등록

제목	<input type="text"/>
카테고리	<input type="text" value="– 선택하세요 –"/>
내용	<input type="text" value="내용을 입력해 주세요...."/>

FAQ

자주 묻는 질문

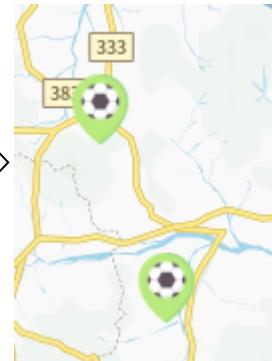
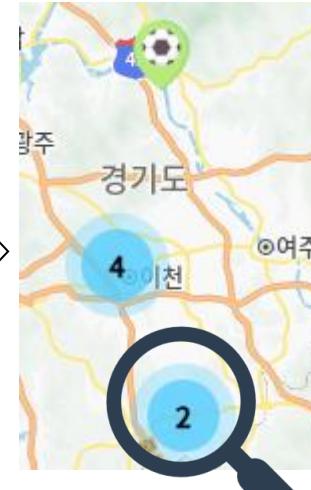
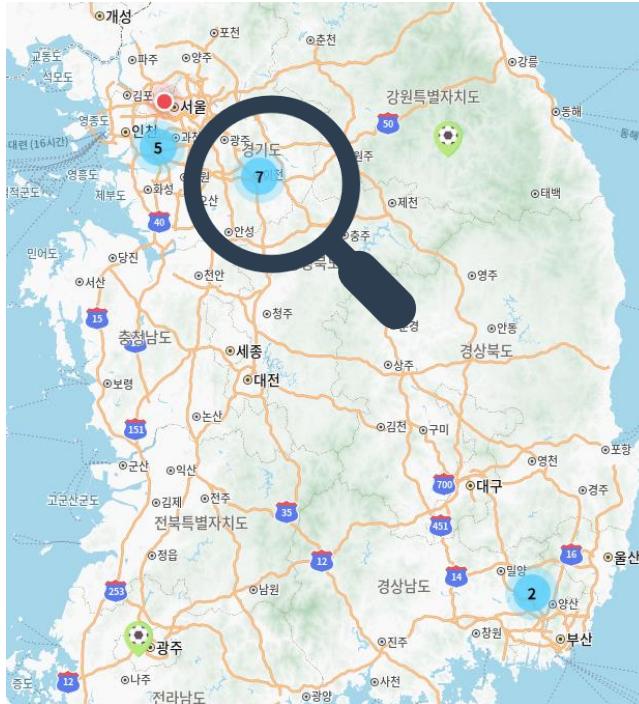
1:1 문의

등록하기

No.	카테고리	제목	작성일	수정일	삭제
1	Team / Guest	팀 채팅 안내	2025-06-14 11:06	2025-06-14 11:06	✖
2	개정	개인정보 수정하는 법	2025-06-14 11:06	2025-06-14 11:06	✖
3	신고	신고 방법	2025-06-14 11:06	2025-06-14 11:06	✖
4	Team / Guest	Guest 이용 방법	2025-06-14 11:06	2025-06-14 11:06	✖
5	이용가이드	MatchON 이용 방법	2025-06-14 11:06	2025-06-14 11:06	✖
6	매너온도	매너온도 계산 기준 안내	2025-06-14 11:06	2025-06-14 11:06	✖
7	이용가이드	챗봇 이용 방법	2025-06-14 11:06	2025-06-14 11:06	✖
8	신고	멤버 신고 안내	2025-06-14 11:06	2025-06-14 11:06	✖

- 자주 묻는 질문 목록
- 관리자는 자주 묻는 질문
수정, 삭제 및 등록 가능

주요 기능 - STADIUM



구장 보기

백사면

검색

백사면체육공원 축구장

경기도 이천시 백사면 원적로801...
031-644-4315

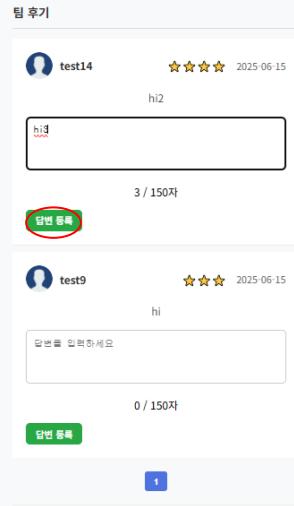
백사체육공원

- 구장 검색 기능

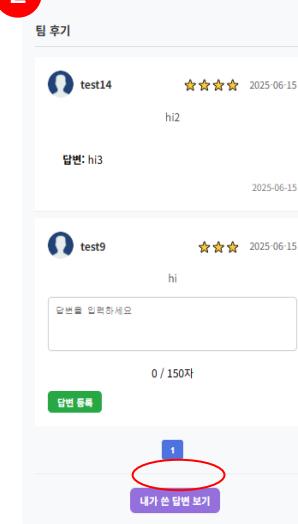
- 카카오 Map API 중 마커 클러스터러를 이용

주요 기능 - 팀 리뷰에 대한 팀장 답변 작성

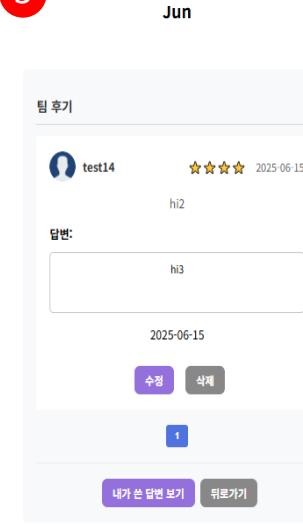
1



2



3



1

```
<div class="review-actions" th:if="${isTeamLeader}">
    <button id="myResponsesBtn" class="btn-secondary">내가 쓴 답변 보기</button>
    <button id="backToAllReviewsBtn" class="btn-cancel" style="border: 1px solid #ccc; padding: 0 5px;">뒤로 가기</button>
</div>
```

```
const isTeamLeader = document.getElementById('leader-flag')?.dataset?.leader === 'true';
const myResponsesBtn = document.getElementById('myResponsesBtn');
if (isTeamLeader && myResponsesBtn) {
    myResponsesBtn.addEventListener('click', (e) => {
        e.preventDefault();
        loadMyResponses(); // Reuse the new centralized logic
    });
}
```

2

```
reviewList.innerHTML = reviews.map(review => {
    const hasResponse = review.response != null;
    const respondedAt = hasResponse
        ? <div class="review-response">
            <strong>답변:</strong> ${review.response}
            <span class="response-date" style="float: right; color: #888; font-size: 0.85em;">
                ${review.respondedAt ? review.respondedAt.split('T')[0] : ''}
            </span>
        </div>
        : isTeamLeader
            ? <div class="response-form">
                <textarea class="response-text" data-id="${review.id}" placeholder="답변을 입력하세요"></textarea>
                <p class="response-char-count" data-id="${review.id}">0 / 150</p>
            </div>
        : null
    );
    return hasResponse
        ? <div class="review-item">
            <div>
                ${review.username} ${review.date}
                <span>${review.rating}</span>
            </div>
            <div>
                ${review.content}
            </div>
            <div>
                ${review.response}
            </div>
            <div>
                ${review.respondedAt}
            </div>
            <div>
                ${review.likes}
            </div>
            <div>
                ${review.comments}
            </div>
            <div>
                ${review.shareCount}
            </div>
            <div>
                ${review.isTeamLeader}
            </div>
        </div>
        : <div class="review-item">
            <div>
                ${review.username} ${review.date}
                <span>${review.rating}</span>
            </div>
            <div>
                ${review.content}
            </div>
            <div>
                ${review.shareCount}
            </div>
            <div>
                ${review.isTeamLeader}
            </div>
        </div>
    );
});
```

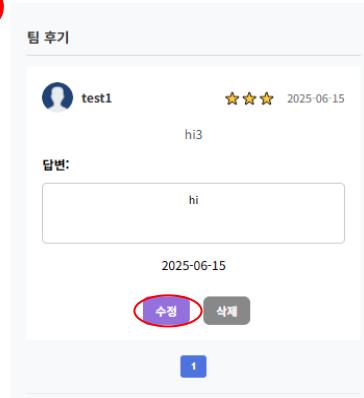
팀장만 후기 답변 작성 가능하도록
프론트&백엔드에서 역할 기반
제어를 이중 적용

이미 답변된 리뷰는 다시 작성
불가하며, UI에서는 작성 품
자체를 숨김

백엔드는 추가로 중복 작성 시도를
막고, 권한 없는 요청도 차단

주요 기능 - 답변 수정/삭제 권한 제어

1



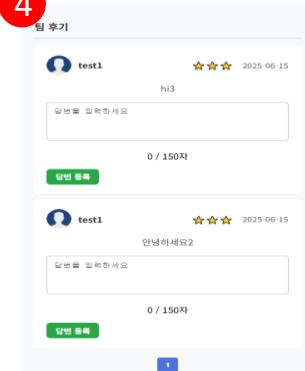
2



3



4



1

```
if (!isTeamLeader && myResponsesBtn) {
    myResponsesBtn.addEventListener('click', (e) => {
        e.preventDefault();
        fetch(`/team/${teamId}/answered-reviews`)
            .then(res => res.json())
            .then(data => {
                const reviews = data.data;
                reviewListView.style.display = 'block';
                reviewWriteView.style.display = 'none';
                backToAllReviewsBtn.style.display = 'inline-block';
            })
    })
}
```

```
@Transactional
public void updateReviewResponse(Long responseId, String updatedText, CustomUser user) {
    // 1. Fetch the response
    Response response = responseRepository.findById(responseId)
        .orElseThrow(() -> new CustomException("리뷰 대댓글을 찾을 수 없습니다."));

    // 2. Check that the user is the leader of the team related to the review
    Review review = response.getReview();
    Team team = review.getTeam();

    // Fetch the TeamMember entry for this user and team
    TeamMember teamMember = teamMemberRepository.findByIdAndTeam_Id(
        user.getId(), team.getId()
    ).orElseThrow(() -> new CustomException("해당 팀원은 팀장이 아닙니다."));

    if (!teamMember.getTeamLeaderStatus()) {
        throw new AccessDeniedException("해당 팀원은 팀장이 아닙니다.");
    }

    // 3. Update the content and timestamp
    response.updateContent(updatedText);

    // 4. Save (if not using dirty checking)
    responseRepository.save(response);
}
```

팀장이 작성한 리뷰 답변만을 조회할 수 있도록 역할 기반 조건(`isTeamLeader`)을 프론트에서 적용하고,

해당 버튼 클릭 시 서버에서 답변이 존재하는 리뷰 목록을 조회하여 동적으로 출력.

각 답변 항목에 대해 수정 및 삭제 기능을 함께 제공하며, 프론트와 백엔드 양측에서 팀장 권한을 기준으로 접근을 제어함으로써

주요 기능 - 팀 가입 조건 필터링 및 중복 요청 차단

1. 팀 소개 화면. 팀 이름은 Jun. 팀 소개란에 팀 소개 내용이 있고, 모집 상태는 모집중입니다.

2. 팀 가입 신청 화면. 팀 이름은 JP. 팀 소개란에 팀 소개 내용이 있고, 모집 상태는 모집중입니다. 팀 활동 지역은 영남권입니다.

3. 팀 소개 화면. 팀 이름은 TEAM. 팀 소개란에 팀 소개 내용이 있고, 모집 상태는 모집중입니다. 팀 활동 지역은 영남권입니다.

4. 팀 소개 화면. 팀 이름은 TEAM. 팀 소개란에 팀 소개 내용이 있고, 모집 상태는 모집중입니다. 팀 활동 지역은 영남권입니다.

1

```
// Change starts here
// Strict check: any past request blocks a new one
boolean alreadyRequested = teamJoinRequestRepository.existsByMemberAndTeam(member, team);
if (alreadyRequested) {
    throw new IllegalArgumentExeption("이미 허가한 팀입니다.");
}

TeamJoinRequest joinRequest = TeamJoinRequest.builder()
    .member(member)
    .team(team)
    .joinRequestStatus(Status.PENDING)
    .isDeleted(false)
    .introduction(intro)
    .build();

teamJoinRequestRepository.save(joinRequest);

// Find the team leader
Member Leader = teamMemberRepository.findTeamAndTeamLeaderStatusTrue(team).Optional<TeamMember>
    .orElseThrow(() -> new IllegalStateException("팀 리더를 찾을 수 없습니다."));
TeamMember .getMemberO(); // FIX HERE

String message = "[[팀 이름]] " + member.getMemberName() + "이 팀에 가입 신청했습니다.";
String url = "/team/" + team.getId() + "/join-request/" + joinRequest.getId();
```

2

```
if (isDisabled) {
    if (userHasTeam) {
        alert("이미 소속된 팀이 있습니다.");
    } else if (!recruitmentStatus) {
        alert("이 팀은 더 이상 팀원을 모집하지 않습니다.");
    } else {
        alert("가입이 불가능한 상태입니다.");
    }
}
```

사용자가 이미 팀에 소속되어 있거나 팀이 모집 완료 상태일 아닐 경우, 프론트에서 신청 탭을 비활성화하고 경고창을 띄우며 신청을 막고, 백엔드에서도 중복 요청을 차단하여 안정성과 무결성을 함께 확보.

주요 기능 - 팀 가입 요청 승인/거절

1

2

3

4

```
TeamMember newMember = TeamMember.builder()
    .team(request.getTeam())
    .teamRequest(getTeamRequest())
    .introduction("안녕하세요!")
    .joinRequestStatus(false)
    .build();

teamMemberRepository.save(newMember);

// member는 팀원으로 team으로 등록되었음
memberRepository.save(member); // 팀원으로 등록되는 경우 팀장이 허가를 필요로 한다.

Member applicant = request.getMember();
String message = "[팀] " + team.getTeamName() + " 팀 가입 신청되었습니다.";
String url = "team/team/" + team.getId(); // or /my-team if you redirect them to their team page
notificationService.sendNotification(applicant, message, url);

ChatRoom teamGroupChatRoom = Optional.ofNullable(team.getChatRoom())
    .filter(ChatRoom::getTeamGroupChat)
    .orElseThrow(() -> new IllegalStateException("팀을 통해 팀원을 추가할 수 없습니다."));
chatService.addParticipantToRoom(teamGroupChatRoom, member); // add member to chat room
```

1

```
@Transactional
public void rejectJoinRequest(Long requestId) {
    TeamJoinRequest request = teamJoinRequestRepository.findById(requestId)
        .orElseThrow(() -> new IllegalArgumentException("존재하지 않는 팀원입니다."));

    if (request.getJoinRequestStatus() != Status.PENDING) {
        throw new IllegalStateException("승인된 팀원입니다.");
    }

    request.setDenied();
    Member applicant = request.getMember();
    Team team = request.getTeam();

    String message = "[팀] 거절됨 " + team.getTeamName() + " 팀 가입 신청이 거절되었습니다.";
    String url = "team/team/" + team.getId(); // optional: could just send them to team list
    notificationService.sendNotification(applicant, message, url);
}
```

2

-팀 리더가 아닌 경우 상세 정보를 볼 수 없도록 서버에서 권한 체크 수행

-승인 시 팀 멤버 등록
member.team 필드에 팀 설정
 팀 단체 채팅방 자동 참여

-처리 상태가 PENDING인지 확인

joinRequest.denied()
 호출

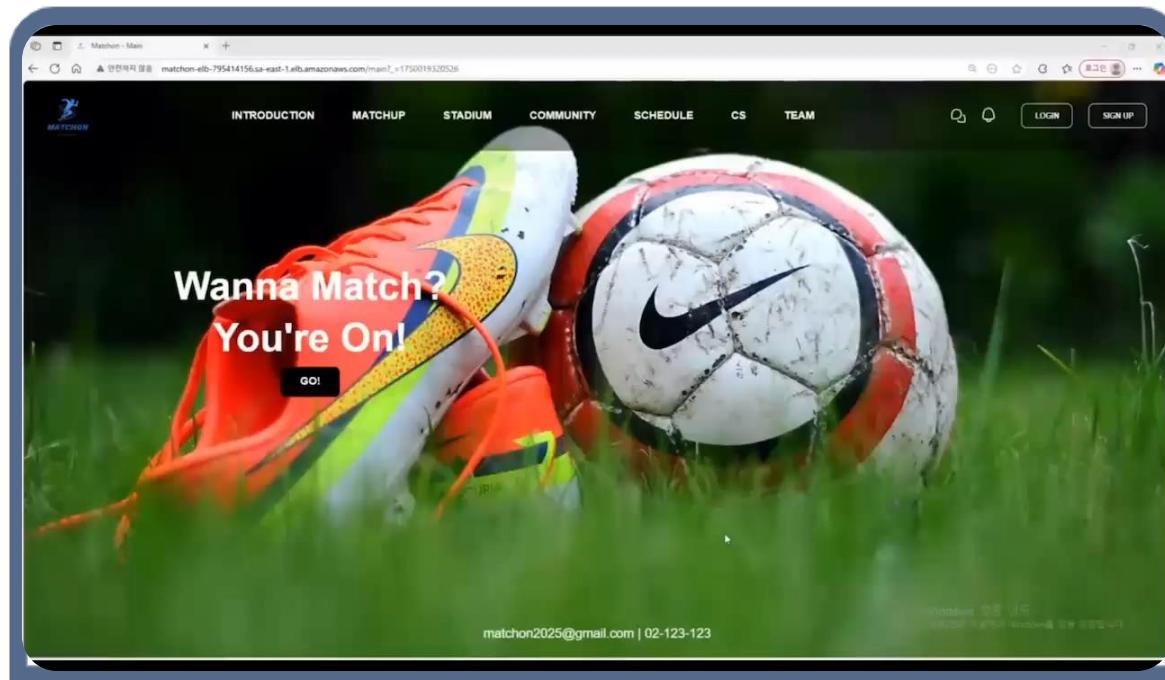
사용자에게 알림 전송



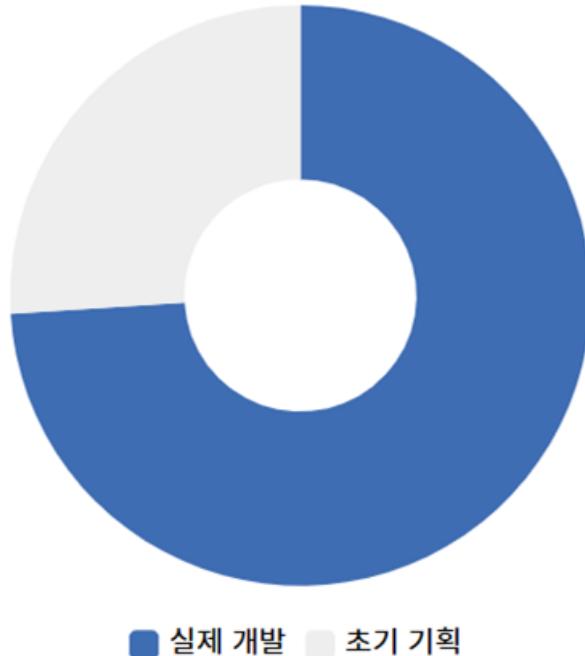
단위 테스트

구글드라이브 링크

기능 시연



성과



→ 개발률 **74.3%**

- ❖ 총 기획된 기능 : 35개
- ☑ 구현 완료 : **26개** (약 74.3%)

💡 전략적 범위 조정

- 초기 기획 단계에서 전 기능을 정의했으나,
프로젝트 목적과 사용자 중심으로 핵심 기능
위주로 우선 구현
- 우선순위가 낮은 9개의 기능은 제외

추후 보완할 기능

01



병렬 & 직렬 승인

관리자 승인 시스템 향상

02



레슨 시스템 연계

축구 및 풋살 코치와
레슨 연계 시스템

03



팀 탈퇴

팀 탈퇴

04



주변 의료시설 찾기

경기 중 위급사항
발생시 빠른 대처

05



쇼핑몰

공동 구매 + 판매

소감

전준혁 이번 프로젝트를 통해 팀원들과 협동하며 실무 수준 비슷한 설계와 구현 경험을 해볼수 있어 좋은 경험이었습니다. 커뮤니티 기능은 단순한 CRUD를 넘어서 권한, 파일 처리, 예외 처리 등 다양한 요소를 고려해야 해 생각보다 어려웠고, 데이터베이스 구조와 배포환경에 대한 이해의 중요성을 절실히 느꼈습니다.

정준열 이번 프로젝트에서는 백엔드 전반의 흐름을 직접 구현하고 구조를 정리하면서 실제 서비스 수준의 설계와 코드를 고민해볼 수 있었습니다. 단순한 데이터 처리뿐 아니라 예외 처리, 보안, 상태 관리 등 여러 요소를 직접 다루며 전체 시스템을 안정적으로 연결하는 데 집중할 수 있었고, 팀원들과 함께 프로젝트를 완성해나가는 과정 자체가 값진 경험이었습니다.

최성은 이번 프로젝트에서는 해야할 기능을 구현 완료해서 기분이 좋습니다. 저번 프로젝트에서는 저의 한계로 인해 못했던 것들을 이번에는 구현할 수 있게 되어서 많은 걸 배운것 같습니다. 개발자는 단순히 코드만 잘 짜고 구현만 잘하는건 줄 알았는데 이번 프로젝트를 통해 개발자는 팀원과의 협업과 소통도 중요하다고 느꼈습니다. 앞으로 이번 프로젝트에서 한 다짐을 잊지 않고 팀원과의 소통과 협업을 중요시하는 개발자가 될것 같습니다.

최효민 프로젝트의 기획부터 설계, 개발, 배포까지 전 과정을 경험하며 웹 개발의 흐름과 실제 개발 프로세스를 배울 수 있었습니다. 협업 과정에서는 Git의 사용의 신중함과 코드 전반적으로 네이밍의 일관성 유지가 얼마나 중요한지 다시 한 번 체감했습니다. 팀원들과 원활히 소통하며 함께 프로젝트를 완성할 수 있어 뜻깊은 시간이었습니다.

홍주희 프로젝트를 처음부터 끝까지 직접 설계하며 실무 감각을 키울 수 있었고, 복잡한 테이블과 다양한 기능을 다루며 성장도 크게 느꼈습니다. 특히 JWT 인증방식이 까다로웠지만 서비스 로직에 대해 더 깊이 생각해볼 수 있는 시간이었습니다. 팀원들과의 협업을 통해 소통과 협력의 중요성을 깊이 배운 경험이었습니다.

출처

- **git** <https://meetup.nhncloud.com/posts/122>
- **awesome-css** <https://github.com/awesome-css-group/awesome-css?tab=readme-ov-file>
- **ui buttons** <https://github.com/youneslaaroussi/ui-buttons>
- **flaticon** <https://www.flaticon.com/>
- **jwt** https://m.blog.naver.com/71_stars/223758295582?isFromSearchAddView=true&recommendTrackingCode=0
- **Token** <https://changha-dev.tistory.com/164>
- **Websocket** WebSockets :: Spring Framework
- **sql filter** https://m.blog.naver.com/71_stars/223758260604
- **redis 연동** <https://jindeveloptravel0919.tistory.com/m/394>
- **s3 버킷** <https://jun-codinghistory.tistory.com/687?category=1149595>
- **jenkins 배포** <https://terianp.tistory.com/190>
- **시스템 설계** https://docs.aws.amazon.com/ko_kr/prescriptive-guidance/latest/data-storage-decoupling-sas-fsx/architecture-design.html,
<https://seongduck.tistory.com/149>
- **faq** <https://velog.io/@max9106/Spring-Boot-JPA-MySQL-게시글-수정-삭제>
- **Gmail API** <https://cloud.google.com/appengine/docs/standard/services/mail/sending-mail-with-mail-api?hl=ko&tab=java>
- **우편번호 API** <https://postcode.map.daum.net/guide>
- **alert API** <https://sweetalert2.github.io/>
- **카카오맵 API** <https://apis.map.kakao.com/web/guide/>
- **구장** <https://www.data.go.kr/data/15113986/openapi.do>, <https://apis.map.kakao.com/web/sample/basicMarkerImage/>
- **OpenApi를 csv로 저장** <https://romworld.tistory.com/229>

Q&A

자유롭게 질문해주세요!

감사합니다

Thank you

3팀

MatchON

matchon2025@gmail.com