



캠핑라운지

캠핑장 예약 서비스

김성우 | 양찬식 | 손수용 | 최수민



1

개요

1. 기획의도

- 1-1 주제선정 - 실태조사
- 1-2 주제선정 - 수요분석
- 1-3 사용자 분석
- 1-4 개발 목표

2. 브랜딩

- 2-1 브랜딩
- 2-2 로고

3. 팀원 소개

2

프로젝트 상세

- 1. 요구사항 정의서
- 2. 기능 정의서
- 3. 개발 일정
- 4. 개발 환경
- 5. 프로젝트 구조도
- 6. UML
 - 6-1 Use Case Diagram
 - 6-2 Sequence Diagram
 - 6-3 Class Diagram
- 7. ERD

3

화면 설계

- 1. 헤더, 푸터
- 2. 메인페이지
- 3. 로그인
- 4. 회원 가입
- 5. 마이페이지
- 6. 타회원 조회
- 7. 회원간 채팅
- 8. 캠핑장 게시판
- 9. 리뷰 게시판
- 10. 어드민 페이지

4

코드 리뷰

5

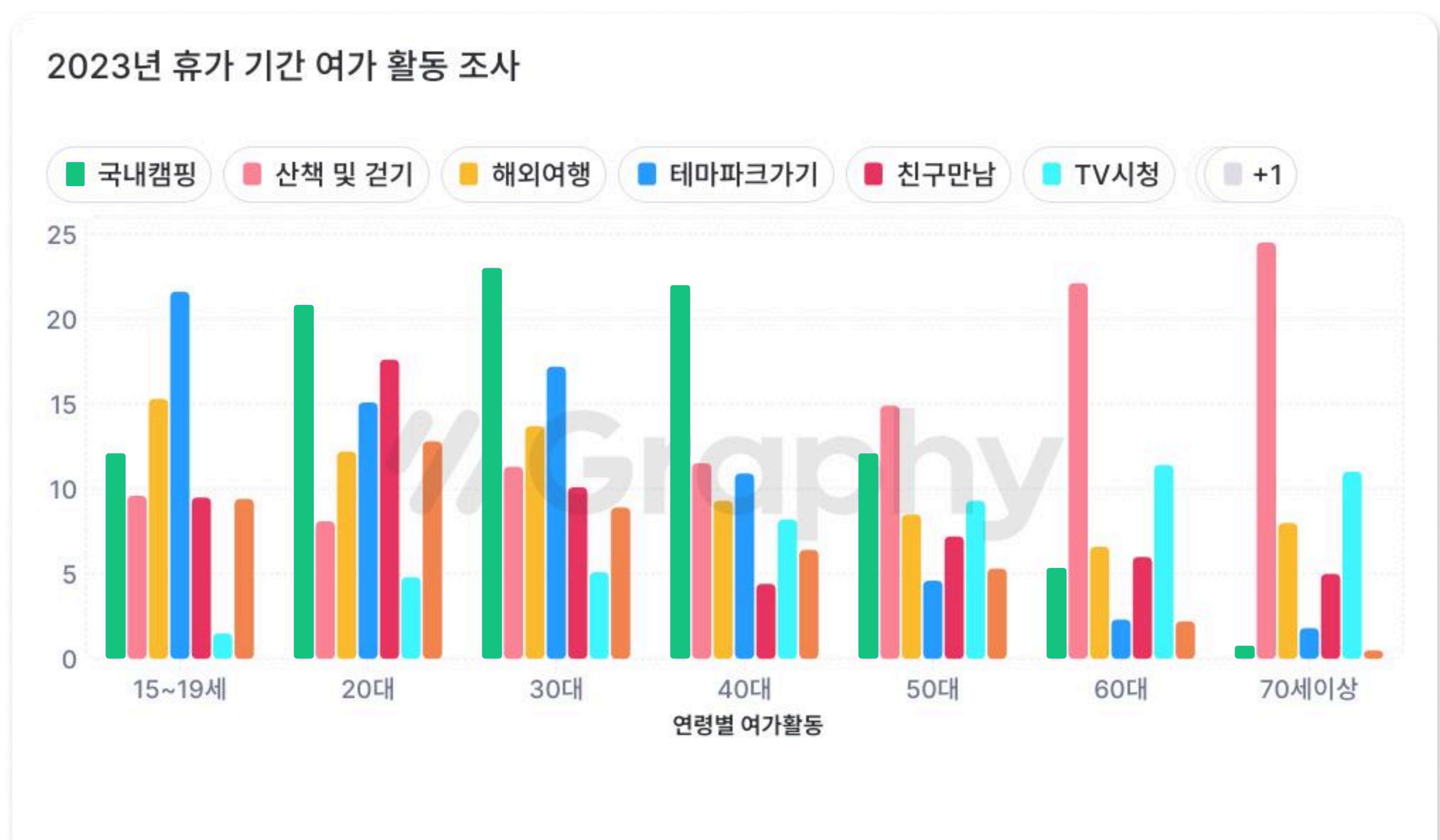
프로젝트 회고

기획 의도



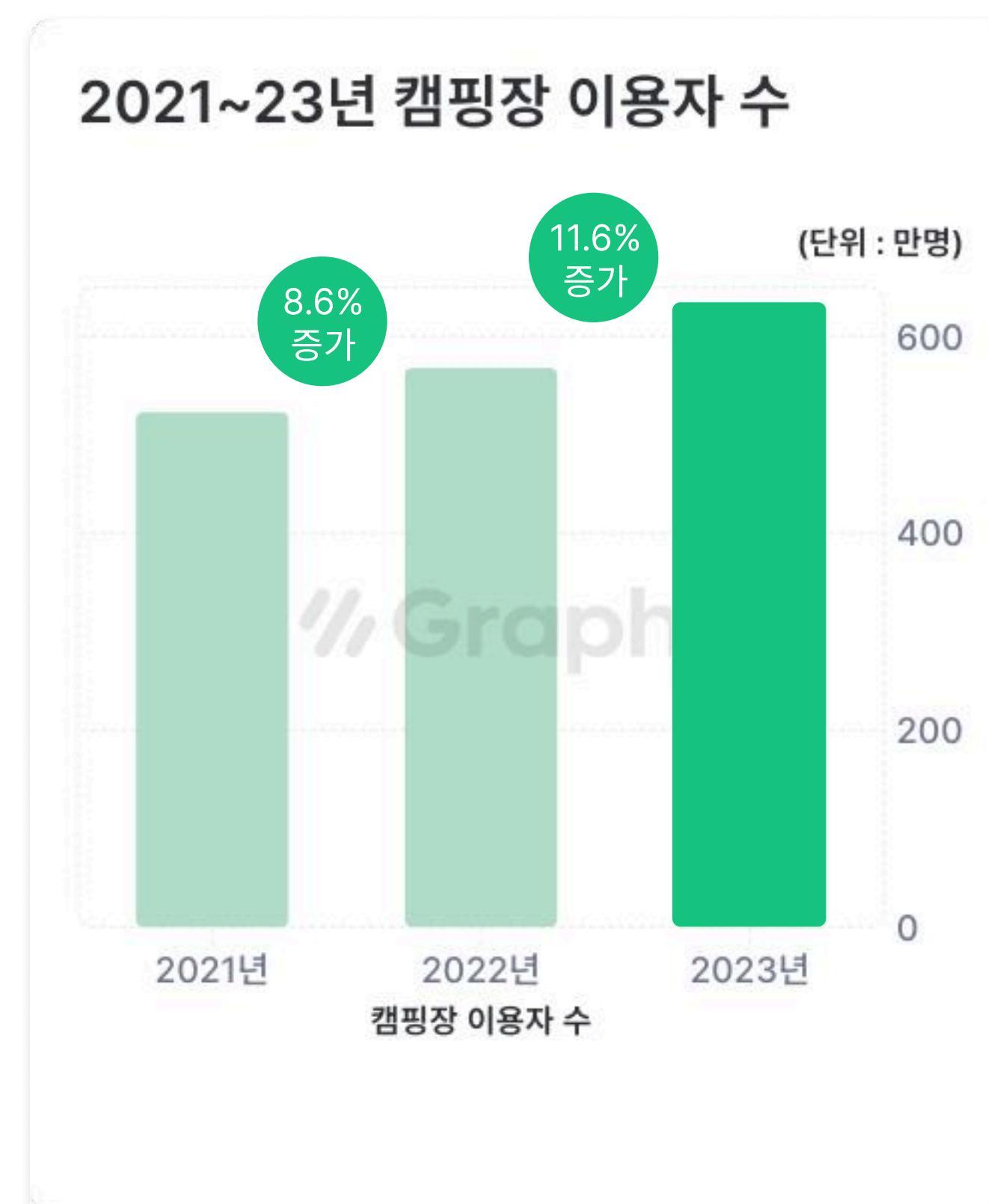
개요

- 20~50세대의 주요 여가활동 '캠핑'



출처 - 「국민여가활동조사」, 문화체육관광부

- 증가하는 캠핑 수요



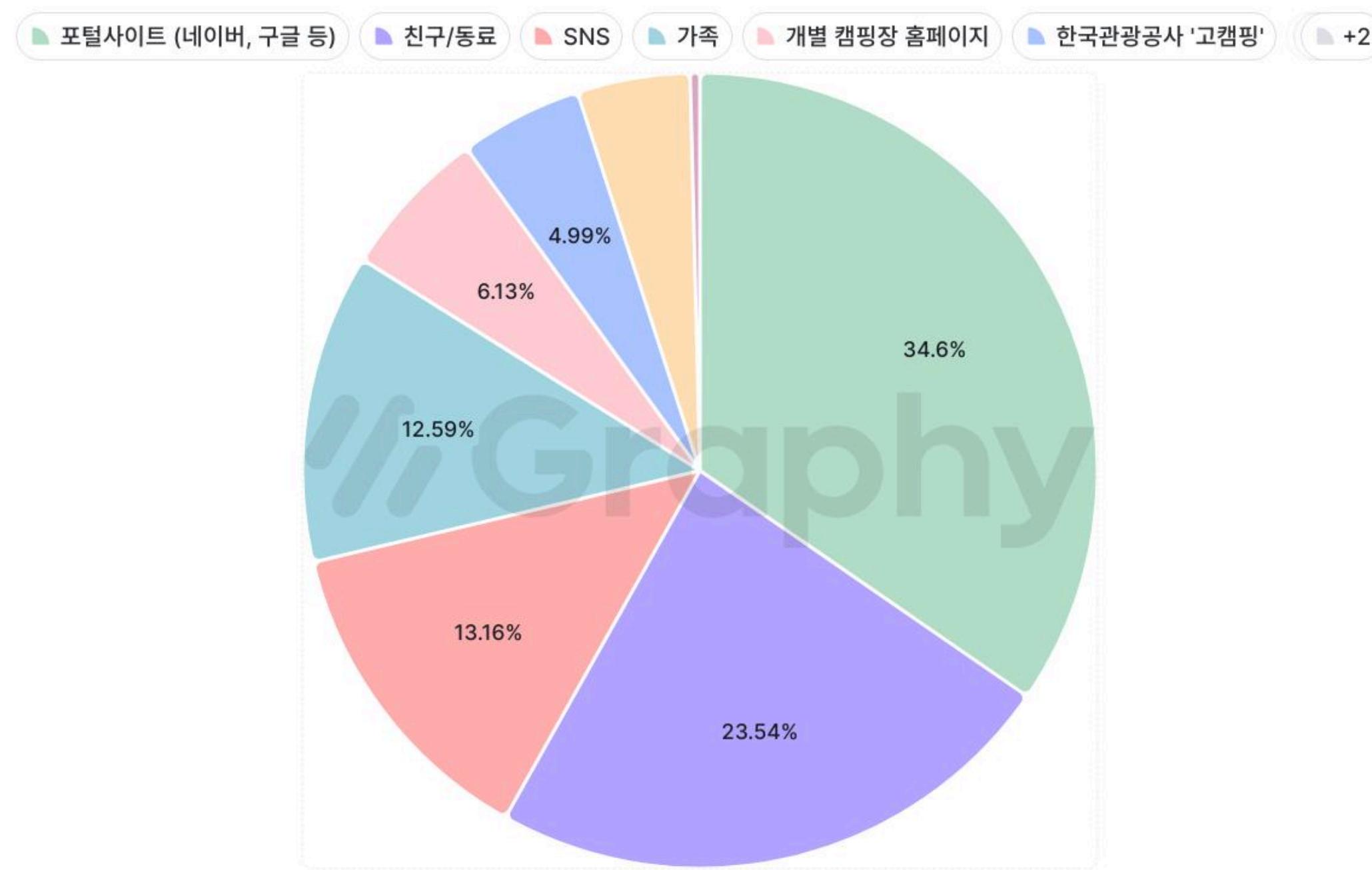
출처 - 「2023년 기준 캠핑 이용자 실태조사」, 문화체육관광부



개요

● 편리한 정보습득 경로의 부재

캠핑장 정보습득 경로



출처 : 2022년 기준 캠핑 이용자 실태조사 - 문화체육관광부

● 비교 분석 가능한 통합 사이트 필요

캠핑장 선택 시 고려사항

■ 1순위 ■ 1,2,3 순위



출처 : 2022년 기준 캠핑 이용자 실태조사 - 문화체육관광부



주제 선정
“캠핑장 통합 예약 사이트”



이름 : 최민수

나이 : 30대

성별 : 남성

월 소득 : 300~400만원

캠핑 입문 계기

펜션, 호텔 등 숙박업소와는 다른 자연에서의 휴식

기존 캠핑장 예약 시 불편했던 점

각각의 캠핑장마다 차이점을 비교하는 것이 어려움

새로운 캠핑장 웹에 바라는 점

기본시설 및 편의시설을 카테고리로 원하는 캠핑장을 찾을 수 있어 예약이 간편함



이름 : 김겨울

나이 : 40대

성별 : 여성

월 소득 : 200~300만원

캠핑 입문 계기

가족 단위 활동을 위해 아이들과 함께할 수 있는 경험을 원함

기존 캠핑장 예약 시 불편했던 점

가족 친화적인 캠핑장 정보 부족
직접 이용해본 사용자의 후기가 궁금함

새로운 캠핑장 웹에 바라는 점

놀이시설 유무 등 세부 정보를 쉽게 확인할 수 있어 유용함
사용자 후기 확인 및 유저간 문의 기능



이름 : 이유진

나이 : 20대

성별 : 여성

월 소득 : 100~200만원

캠핑 입문 계기

반려동물과 함께할 수 있는 여행을 원함

기존 캠핑장 예약 시 불편했던 점

반려동물 동반 가능 여부를 개별적으로 전화 등 기타 수단으로 확인해야 함

새로운 캠핑장 웹에 바라는 점

반려동물 동반 가능 캠핑장을 필터링할 수 있어 정보 탐색이 간편함



이름 : 정은우

나이 : 40대

성별 : 남성

월 소득 : 600~700만원

캠핑 입문 계기

고급스러운 글램핑을 통해 색다른 여가 생활을 즐기기 위함

기존 캠핑장 예약 시 불편했던 점

글램핑 시설의 상세 사진 및 제공 서비스 정보 부족

새로운 캠핑장 웹에 바라는 점

고급 캠핑장 정보를 한눈에 확인하고 예약까지 간편하여 효율적



개요



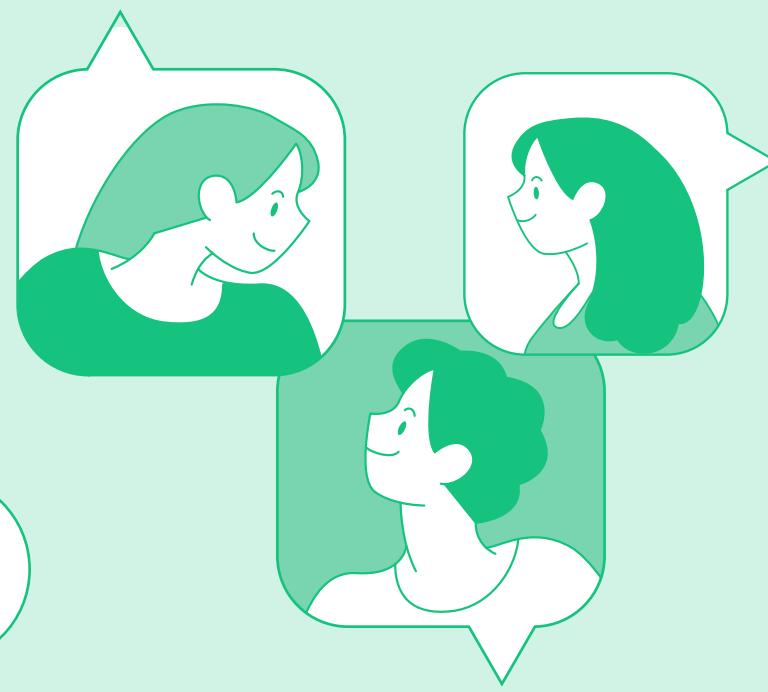
여러 캠핑장 비교

편의시설 검색



유저 친화적인 UIUX

리뷰 및 유저간 소통



브랜딩



Camping + Lounge

캠핑라운지

Font : 학교안심 둥근미소

Primary Color : #16C47F

Slogan

“예약부터 소통까지! 캠핑의 모든 것!”

휴식과 정보가 공존하는 캠핑러들의 안식처, 캠핑라운지!



캠핑라운지 (Camping Lounge)의
C와 A를 활용하여 텐트 모양을 형상화

Text Logo

캠핑라운지

Font : 학교안심 둥근미소

Color : #16C47F

#FFFFFF

팀원 소개



개요



김성우

프로젝트 팀장(PM)

디자인 및 퍼블리싱

프론트엔드 총괄

백엔드 총괄 및 DB설계

Admin(관리자) 기능 개발

Open API 연동

Query DSL 연동

캠핑장 게시판/게시글 CRUD 개발

예약 기능 개발



양찬식

프로젝트 리더(PL)

프론트엔드 ,백엔드 개발

인증시스템, JWT 개발

WebSocket 연동

채팅 기능 개발

DB 설계



손수용

프론트엔드 ,백엔드 개발

맴버 파트 개발

인증시스템 연동

소셜로그인(Google API) 연동

토스 API 테스트용 결제 연동

DB 설계



최수민

프론트엔드 ,백엔드 개발

리뷰 게시판/게시글 CRUD 개발

댓글 CRUD 개발

React Quill 연동 글쓰기 에디터

DB 설계

프로젝트 상세



주요 요구사항 정의

캠핑라운지

A조(김성우, 양찬식, 손수용, 최수임)

개요

공공데이터 API 중 한국관광공사 '고캠핑' 홈페이지에서 제공하는 정보를 활용하여 전국 캠핑장 정보를 제공하는 웹사이트를 제작한다.

'캠핑라운지'는 Camping과 lounge의 합성어로 편하게 휴식을 즐기는 라운지처럼 캠핑을 좋아하는 사용자들이 모여 캠핑에 관한 이야기를 나누고 정보를 주고받는 휴식공간이 되었으면 좋겠다는 사이트의 제작 지향점을 의미한다.

주요 기능은 야영장 정보를 사용자가 원하는 옵션에 맞게 필터링 하여 제공하는 것과 해당 야영장 사용을 예약하는 것, 사용자별 리뷰게시물 등록 및 댓글 등록 기능, 사용자간 실시간 채팅 기능이 있다.

주요 기능

본 웹사이트에서 제공할 기능은 다음과 같다.

1. 캠핑장 정보 제공
2. 캠핑장 정보 필터링 및 검색
3. 날짜 별 캠핑장 사용 가능 여부 확인
4. 특정 날짜의 사용 가능한 자리(사이트) 수 조회
5. 사용 가능한 자리(사이트) 예매
6. 유저 간 실시간 채팅
7. 유저 리뷰 및 후기 게시판
 - o 후기 등록
 - o 타 유저가 댓글 및 별점 등록 가능
 - o 사진 첨부 기능 지원
 - o 게시글 작성 에디터 제공 (가능할 경우)
8. 인기 게시글(후기) 메인 페이지 노출

기능 상세 요구사항

1. 캠핑장 정보 제공

- 한국관광공사 '고캠핑' API를 활용하여 전국 캠핑장 정보를 제공
- 캠핑장 기본 정보 표시

2. 캠핑장 정보 필터링 및 검색 기능

- 키워드 검색(캠핑장 이름, 지역 등)
- 필터링 기능 제공
 - o 화장실 여부
 - o 개수대 여부
 - o 온수 제공 여부
 - o 운동시설 여부
 - o 반려동물 출입 가능 여부
 - o 수영장 여부
 - o 최대 수용 인원 수
 - o 캠핑장 연락처
 - o 전력 제공 여부
- 날짜 별 캠핑장 사용 가능 여부 확인
- 사용자가 원하는 날짜를 선택하여 해당 날짜의 캠핑장 사용 가능 여부를 조회 가능

4. 특정 날짜의 사용 가능한 자리(사이트) 수 조회

- 선택한 캠핑장의 특정 날짜별 이용 가능한 자리(사이트) 수를 조회 가능

5. 사용 가능한 자리(사이트) 예매

- 특정 날짜와 사이트를 선택하여 예매 가능
- 결제 시스템 연동 (추후 논의)

6. 유저 간 실시간 채팅

- 웹소켓을 활용하여 유저 간 1:1 및 그룹 채팅 기능 제공
- 캠핑장별 채팅방 개설 기능 고려

7. 유저 리뷰 및 후기 게시판

- 유저당 한 개의 포스팅 공간 제공
- 후기 게시판 기능 제공
 - o 후기 등록 가능
 - o 타 유저가 댓글 작성 가능
 - o 별점 등록 가능 (1~5점)
 - o 사진 첨부 가능
 - o 게시글 작성 에디터 제공 (추후 구현 여부 논의)

8. 인기 게시글(후기) 메인 페이지 노출

- 좋아요, 조회수, 댓글 수 등을 기준으로 인기 게시글을 메인 페이지에 노출
-

시스템 요구사항

1. 기술 스택

- 프론트엔드: React (Next.js 고려 가능)
- 백엔드: Spring Boot + MyBatis

- 데이터베이스: MySQL
- 인증 및 보안: JWT 인증, OAuth2 로그인 (구글, 네이버, 카카오 등)
- 실시간 채팅: WebSocket 사용
- 결제 시스템(예매 관련): PG 사 연동 검토
- 호스팅: AWS 또는 국내 클라우드 서비스 고려

2. API 활용

- 한국관광공사 '고캠핑' API
- 기상청 날씨 API
- 기타 필요한 공공 데이터 API (추후 검토)

3. 사용자 권한 및 역할

- 일반 사용자: 캠핑장 검색 및 예매, 리뷰 작성, 댓글 및 채팅 참여 가능
- 관리자: 게시판 관리, 사용자 관리, 캠핑장 정보 등록 및 수정 가능



기능 정의서

0주
소

No	Location				Page ID	Page Name	Detail	
	1Depth	2Depth	3Depth	4Depth			Content	Description
1	로그인				MEM_01_01	로그인 페이지	이메일 입력, 비밀번호 입력	이메일, 비밀번호 입력 창이 있으며 로그인 버튼을 누를 시 입력된 정보와 DB의 회원 정보를 비교하여 로그인 성공 여부를 판단한다. DB에 없는 정보일 경우 알림(alert) 창을 띄운다.
		회원가입			MEM_01_02	회원가입 페이지	이메일, 비밀번호 및 개인정보 입력	로그인 페이지와 회원가입 버튼을 통하여 회원가입 페이지로 이동한다. 이메일, 비밀번호, 이름, 성별, 전화번호, 프로필 사진을 넣을 수 있다. 이메일 비밀번호는 필수 입력이다.
2	마이페이지	내 정보 조회			MEM_02_01	마이 페이지	유저 정보	회원 자신의 이메일, 이름, 성별, 전화번호, 프로필 사진을 볼 수 있다. 비밀번호는 안 보이게 처리 되어있다. 회원 정보 수정 버튼과 회원 탈퇴 버튼이 있다.
		회원 정보 수정			MEM_02_02	회원 정보 수정 페이지	새 유저 정보 입력	회원 자신의 이름, 이메일, 성별, 전화번호, 프로필 사진, 비밀번호를 변경 할 수 있다.
		회원 탈퇴			MEM_02_03	회원 탈퇴 페이지	회원 탈퇴 안내, 회원 탈퇴 버튼	회원 탈퇴 시 안내할 내용과 회원 탈퇴 버튼이 있다. 회원 탈퇴 버튼을 누르면 DB에 요청을 보내 회원을 비활성화 처리한다. 일정 시간이상 비활성화 된 회원은 자동으로 DB에서 삭제된다.
		내 예약 관리	예약 조회		MEM_02_04	예약 조회 페이지	내 예약 내역을 조회한다.	내 예약 내역을 조회한다.
			예약 삭제		MEM_02_05	예약 삭제 페이지	내 예약 내역을 삭제한다.	내 예약 내역을 삭제한다. 삭제 시 결제한 방식으로 환불처리된다.
3	타 유저 상세 조회				MEM_03	타 유저 마이페이지	타 유저 정보, 1:1 채팅 버튼	후기 페이지에서 작성자의 마이페이지로 이동할 수 있다. 작성자의 정보와 1:1 채팅 버튼이 있다.
4	1:1 채팅	메시지 발신			CH_01_01	채팅 페이지	메시지 입력	사용자가 1:1 채팅 페이지에서 텍스트 메시지를 입력하고 전송할 수 있다. 메시지는 실시간으로 상대방에게 전달된다.
		메시지 수신			CH_01_02		수신 한 메시지	상대방이 보낸 메시지를 실시간으로 수신하여 확인 할 수 있다. 수신된 메시지는 알림으로 확인 할 수 있다. 알림은 헤더 상단에 존재한다. (미정)

No	Location				Page ID	Page Name	Detail	
	1Depth	2Depth	3Depth	4Depth			Content	Description
1	관리자 인증	로그인			ADM_01_01	로그인 페이지	이메일 입력, 비밀번호 입력	이메일, 비밀번호 입력 창이 있으며 로그인 버튼을 누를 시 입력된 정보와 DB의 관리자 정보를 비교하여 로그인 성공 여부를 판단한다. DB에 없는 정보일 경우 알림(alert) 창을 띄운다. 관리자 계정 로그인 성공 시 관리자 페이지로 이동한다. 관리자 페이지는 효율적인 웹사이트 관리를 위해 일반 유저가 사용하는 페이지와는 별도로 제작된 페이지다.
		회원가입			ADM_01_02	회원가입(관리자 추가) 페이지	이메일, 비밀번호 및 개인정보 입력 관리자 추가 용 시리얼넘버 입력	일반 유저가 접근할 수 없는 별도의 로그인 페이지로 로그인한다. 회원가입 버튼을 통하여 회원가입 페이지로 이동한다. 이메일, 비밀번호, 이름, 성별, 전화번호, 프로필 사진을 넣을 수 있다. 일반 맴버 회원가입과는 달리 고유 시리얼 넘버를 입력해 값이 유효해야만 회원가입(관리자 추가)에 성공한다. 이메일의 비밀번호 시리얼넘버는 필수 입력이다.
3	관리자 페이지	관리자 정보 조회			ADM_02_01	관리자 조회 페이지	현재 로그인한 관리자 정보를 조회	현재 로그인한 관리자의 이메일, 이름, 프로필 사진 등 정보를 조회한다.
		관리자 정보 수정			ADM_02_02	관리자 정보 수정 페이지	현재 로그인한 관리자 정보를 수정	이메일을 제외한 관리자 정보를 수정한다.
		관리자 삭제			ADM_02_03	관리자 탈퇴 페이지	현재 로그인한 관리자 삭제	관리자 삭제 버튼 클릭 시 비밀번호와 시리얼 넘버를 확인 후 일치하면 현재 로그인한 관리자 정보를 DB에서 삭제한다.
4	게시물 관리	캠핑장 조회	캠핑장 상세	ADM_03_01	캠핑장 조회 페이지	모든 캠핑장 게시물을 조회	등록된 모든 캠핑장 게시물을 조회한다. 캠핑장 클릭 시 해당 캠핑장의 상세 페이지를 조회한다.	
		캠핑장 추가		ADM_03_02	캠핑장 등록 페이지	캠핑장 게시물을 신규 등록	캠핑장 게시물을 지정된 양식에 맞게 작성하여 신규 등록한다.	
		캠핑장 수정		ADM_03_03	캠핑장 수정 페이지	캠핑장 게시물을 수정	캠핑장 게시물의 내용을 수정한다.	
		캠핑장 삭제		ADM_03_04	캠핑장 삭제 페이지	캠핑장 게시물을 삭제	캠핑장 게시물을 삭제한다.	
		리뷰 조회	리뷰 상세	ADM_04_01	리뷰 조회 페이지	리뷰 게시물을 조회	유저가 등록한 모든 리뷰 게시물을 조회한다.	
			리뷰 삭제	ADM_04_02	리뷰 삭제 페이지	리뷰 게시물을 삭제	리뷰 게시물을 삭제한다.	

5	회원 관리	회원 조회		ADM_05_01	유저 조회 페이지	모든 유저 권한 멤버 조회	권한이 있는 유저(USER)인 모든 멤버를 조회한다.
		회원 비활성화		ADM_05_02	유저 삭제 페이지	유저 권한 멤버 비활성화	권한이 있는 유저(USER)인 멤버를 비활성화 처리한다.
6	예약 관리	예약 조회		ADM_06_01	예약 조회 페이지	모든 예약을 조회	모든 유저와 모든 캠핑장의 예약을 조회한다.
		예약 삭제		ADM_06_02	예약 삭제 페이지	예약을 삭제	예약을 삭제한다.

No	Location				Page ID	Page Name	Detail			
	1 Depth		2 Depth				Content			
							Description			
1	캠핑장	캠핑장 정보 조회	캠핑장 목록	캠핑장 전체 목록 조회	CA_01_01	캠핑장 목록 조회 페이지	캠핑장 전체 목록 조회	등록된 캠핑장 전체 목록 조회		
2				인기 캠핑장 목록 조회		유저 메인 페이지	인기 캠핑장 목록 조회	예약이 많은 순서로 노출되는 인기 캠핑 목록 조회		
3				캠핑장 검색	CA_01_03	캠핑장 목록 조회 페이지	캠핑장 검색	특정 키워드로 검색		
4				캠핑장 상세 정보 조회	CA_01_04	캠핑장 상세 정보 페이지	캠핑장 상세 정보 조회	캠핑장명, 캠핑장 사진, 예약 가능 여부, 전화번호, 위치, 화장실 유무, 주차장 유무, 전력 유무, 편의시설 유무 등 캠핑장 정보 조회		
5		캠핑장 예약 관리		캠핑장 예약	CA_01_04	캠핑장 상세 정보 페이지	캠핑장 예약	캠핑장, 날짜 등을 선택하여 캠핑장 예약		
6				캠핑장 예약 정보 변경		CA_02_01	캠핑장 예약 정보 수정 페이지	예약 캠핑장, 예약 날짜 등 예약 정보 변경		
7				캠핑장 예약 취소			캠핑장 예약 취소	예약을 취소		

No	Location				Page ID	Page Name	Detail			
	1 Depth		2 Depth				Content			
							Description			
1	리뷰	리뷰	리뷰 상세페이지		REV_01_01	메인 페이지	best 리뷰, 작성된 리뷰 조회	베스트 리뷰는 주전 수가 높은 리뷰 순으로 결정되며 메인 페이지에 일정 기간(7일) 동안 노출된다. 작성된 리뷰는 각각의 회원들이 최근에 작성한 리뷰 순으로 노출된다.		
2					REV_01_02	리뷰 상세 페이지	리뷰 내용	본문 내용과 추천, 댓글창으로 구성		
3				추천	REV_01_03		리뷰 추천	작성자 본인은 자신이 작성한 리뷰에 추천을 할 수 없고 각각의 회원들은 리뷰마다 추천을 한번만 할 수 있다.		
4				댓글	REV_01_04		댓글 작성	댓글은 모든 리뷰에 작성 가능하고 100자 이하로 작성할 수 있다.		
5				수정	REV_01_05		댓글 수정	수정 및 삭제는 작성자 본인과 관리자만 가능하고		
6				삭제	REV_01_06		댓글 삭제	삭제된 댓글은 복구할 수 없다.		
7		마이페이지	리뷰 작성		REV_02_01	리뷰 작성 페이지	리뷰 작성	리뷰는 이용했던 캠핑장에 대해 하나의 리뷰만 작성할 수 있으며 블로그 형식으로 작성할 수 있다. 메인 썸네일과 본문에 사용될 사진을 등록 할 수 있고, 본문은 텍스트와 사진, 링크 등으로 구성된다.		
8			리뷰 조회		REV_02_02	리뷰 조회 페이지	작성한 리뷰 조회	회원이 작성한 모든 리뷰를 조회할 수 있다.		
9			리뷰 삭제		REV_02_03		작성한 리뷰 삭제	리뷰 삭제는 작성자 본인과 관리자만 가능하고, 해당 리뷰를 정말 삭제할지 확인(ex: 모달창) 한 이후에 삭제 혹은 취소 처리된다. (협의 필요) 삭제된 리뷰는 다시 복구할 수 없다.		
10			리뷰 수정		REV_02_04	리뷰 수정 페이지	작성한 리뷰 수정	리뷰 수정은 리뷰 작성 페이지에 본문의 내용이 유지된 상태에서 내용, 사진 등을 추가하거나 삭제 할 수 있다.		

No	Location				Page ID	Page Name	Detail			
	1 Depth		2 Depth				Content			
							Description			
1	예약	결제			RES_01_01	캠핑장 예약 페이지	유저가 캠핑장을 예약한다	캠핑장 상세 조회 후 예약하기 버튼을 통해 접근한다. 날짜 별 예약 가능 날짜를 확인하고 해당 날짜에 예약하면 결제 시스템으로 연결된다. 결제가 정상적으로 처리되면 DB에 예약 정보를 저장한다.		
2			예약 조회		MEM_02_04	멤버 기능 정의의 내용과 동일	멤버 기능 정의의 내용과 동일	멤버 기능 정의의 내용과 동일		
	예약 관리	예약 취소			MEM_02_05	멤버 기능 정의의 내용과 동일	멤버 기능 정의의 내용과 동일	멤버 기능 정의의 내용과 동일		



상세

단계	작업	이름	번호	세부항목 및 산출물	시작일	종료일	작업 기간	
기획	현행업무 분석	공통	1	아이디어 구상	2025-02-06	2025-02-06	1	
			2	개발방향 설정, 자료조사 및 분석, 주제 선정				
			3	기획서 초안작성				
	의사소통 관리	김성우	4	기획서 중간발표 및 회의록 작성	2025-02-07	2025-02-07	1	
	요구사항 분석	공통	5	요구사항 정의서(SRS) 작성	2025-02-07	2025-02-07	1	
설계	기능 정의	공통	6	기능 정의서 작성	2025-02-10	2025-02-10	1	
	화면 설계		7	UML 및 플로우 차트 작성				
			8	와화면설계, UI 정보구조 설계, IA(메뉴 구조) 설계				
	프로그램 설계		9	S/W 개발 명세서 작성				
	데이터베이스 설계		10	테이블 명세서 작성 및 ERD 작성	2025-02-11	2025-02-11	1	
			11	공공데이터 API 서칭 및 선정				
			12	[MySQL] Schema 설계 및 관계형 데이터베이스 구축				
인터페이스 및 웹 디자인	김성우	김성우	13	디자인 컨셉 도출 및 레퍼런스 서칭	2025-02-11	2025-02-11	1	
			14	디자인 스타일가이드 작성				
			15	[피그마] 웹 페이지 디자인	2025-02-11	2025-02-12	2	
UI 구현	퍼블리싱	김성우	16	스타일 가이드 기반 공통 CSS 작성(변수 선언 및 클래스 네이밍)	2025-02-13	2025-02-13	1	
			17	[유저] 메인페이지, 로그인페이지 퍼블리싱	2025-02-13	2025-02-13	1	
			18	[유저] 캠핑장 목록, 캠핑장 상세 페이지 퍼블리싱	2025-02-14	2025-02-14	1	
			19	[유저] 유저 상세 정보 페이지 퍼블리싱	2025-02-14	2025-02-14	1	
			20	[유저] 리뷰 목록 및 리뷰 상세 페이지 퍼블리싱	2025-02-14	2025-02-14	1	
			21	[관리자] 메인페이지 퍼블리싱	2025-02-15	2025-02-15	1	
			22	[관리자] 유저 목록, 캠핑장 목록, 리뷰 목록, 예약 목록 페이지 퍼블리싱	2025-02-16	2025-02-16	1	
			23	[관리자] 캠핑장 등록 페이지 퍼블리싱	2025-02-16	2025-02-16	1	

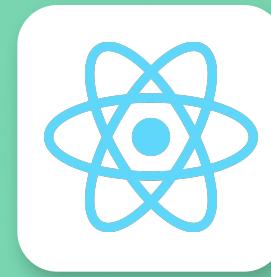


일정

개발 구현	Backend	김성우	24	[관리자] 캠핑장 등록, 조회, 수정, 삭제 기능 개발	2025-02-17	2025-02-18	2
			25	[캠핑장 예약 서비스] 캠핑장 목록 조회 및 상세 조회 기능 개발	2025-02-19	2025-02-19	1
			26	[캠핑장 예약 서비스] 캠핑장 예약 및 결제 기능 개발	2025-02-20	2025-02-23	4
			27	[관리자] 예약목록 조회 기능 개발	2025-02-24	2025-02-24	1
			28	[관리자, 사용자] 로그인 기능 연동	2025-02-25	2025-02-25	1
		양찬식	29	[시큐리티] 시큐리티 필터, 필터체인 등 설정 및 개발	2025-02-06	2025-02-10	5
			30	[JWT] JWT 생성, 로그인 요청 감시 필터, 인증 필터 개발	2025-02-10	2025-02-16	6
			31	[캠핑장 예약] 캠핑장 예약 기능 개발	2025-02-16	2025-02-19	4
		손수용	32	[유저] 로그인/회원가입 기능 개발	2025-02-06	2025-02-11	4
			33	[유저] 회원 정보 조회/ 회원 정보 수정/회원 탈퇴 기능 개발	2025-02-12	2025-02-14	3
			34	[1:1 채팅 서비스] 채팅 기능 개발	2025-02-17	2025-02-21	5
		최수민	35	[게시글 작성, 조회 기능] 사용자 리뷰 게시판 개발	2025-02-06	2025-02-10	5
			36	[추천 게시글 기능] 베스트 리뷰 노출 기능 개발	2025-02-11	2025-02-16	6
			37	[댓글] 댓글 기능 개발	2025-02-17	2025-02-21	5
테스트 및 보완	Frontend(React)	김성우	38	관리자 페이지 개발 (관리자 로그인, 캠핑장 CRUD, 관리자 기능) 사용자 페이지 개발 (캠핑장 조회, 캠핑장 예약)	2025-02-25	2025-02-28	4
			39	[회원가입] 유저 회원가입 페이지 개발	2025-02-19	2025-02-28	10
		양찬식	40	[로그인] 유저 로그인 페이지 개발	2025-02-24	2025-02-26	3
			41	회원페이지 개발: (로그인/회원가입)	2025-02-23	2025-02-27	5
		손수용	42	마이페이지 개발: (회원정보 조회/수정/탈퇴)	2025-02-23	2025-02-27	5
			43	[게시판] 게시판 페이지 개발	2025-03-03	2025-03-05	3
		최수민	44	[댓글] 댓글 창 개발	2025-03-06	2025-03-06	1
배포 및 공유	애플리케이션 테스트	공통	45	단위 테스트 및 오류 수정 및 보완	2025-03-03	2025-03-05	3
	애플리케이션 배포		46	통합 테스트 및 오류 수정 및 보완	2025-03-06	2025-03-06	1
			47	시스템 테스트 및 오류 수정 및 보완			
		김성우	48	최종 프로젝트 파일 배포 및 공유			



Front End 프론트 개발

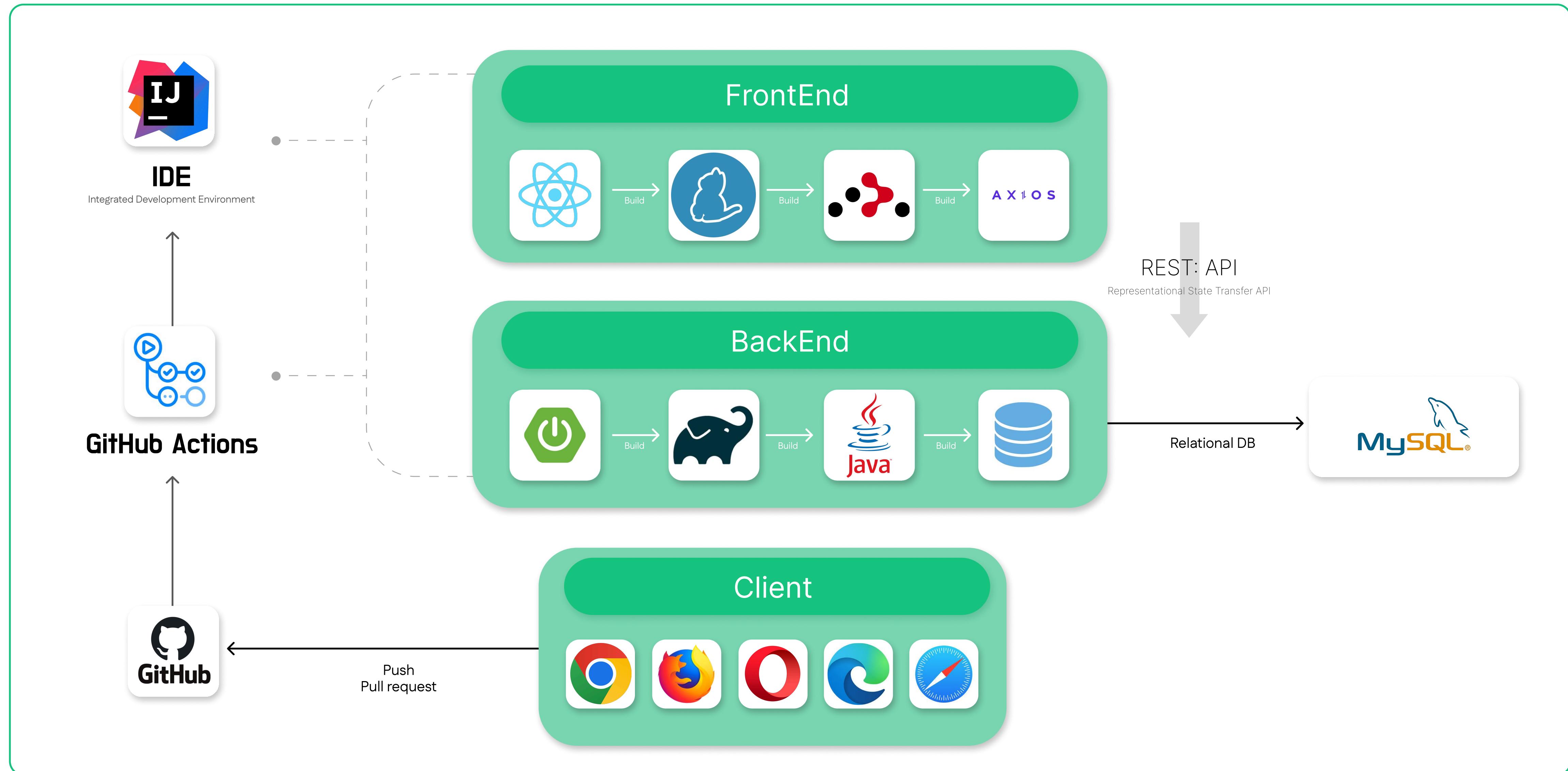


Back End 백엔드 개발



IDE 배포/개발 도구

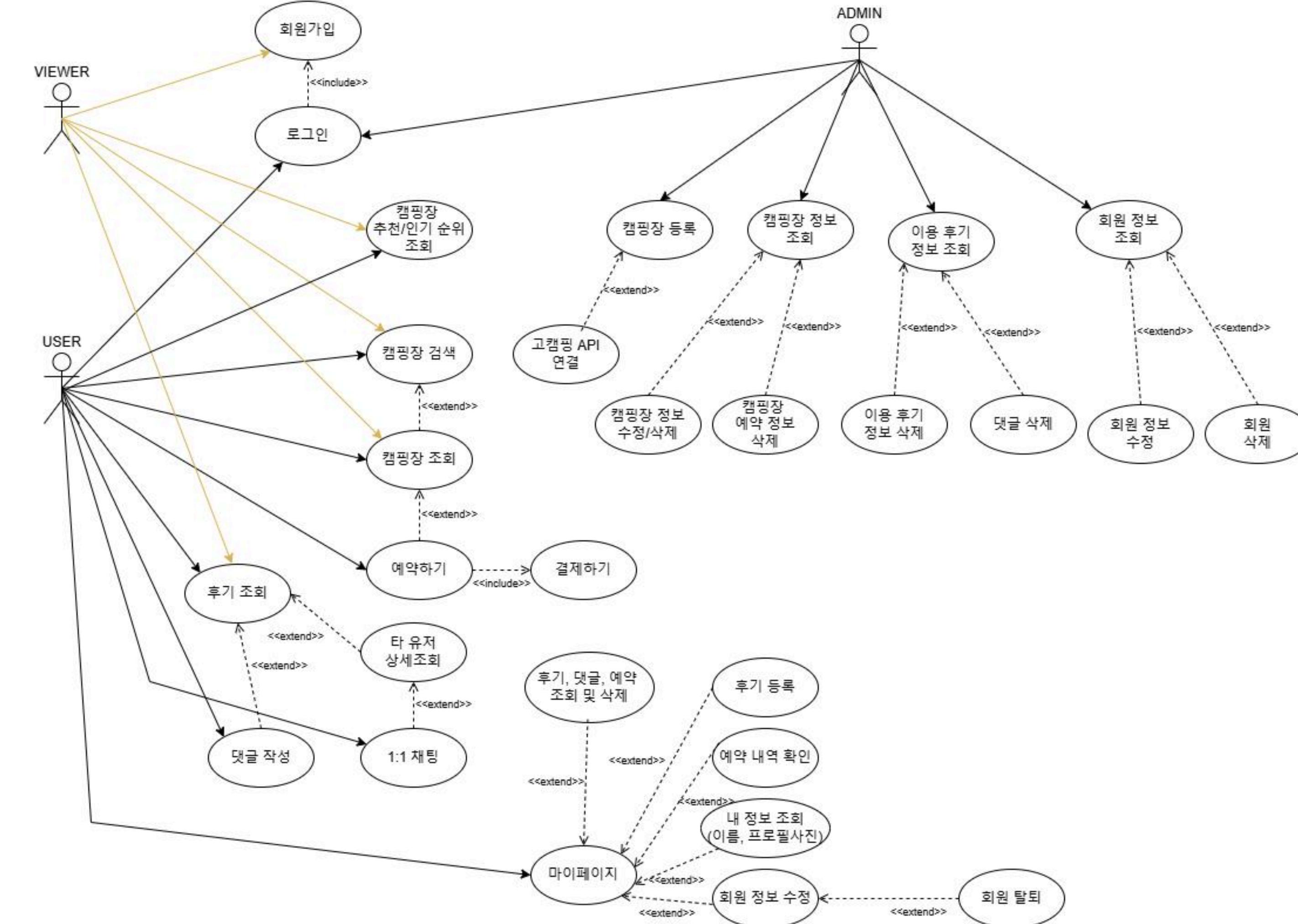






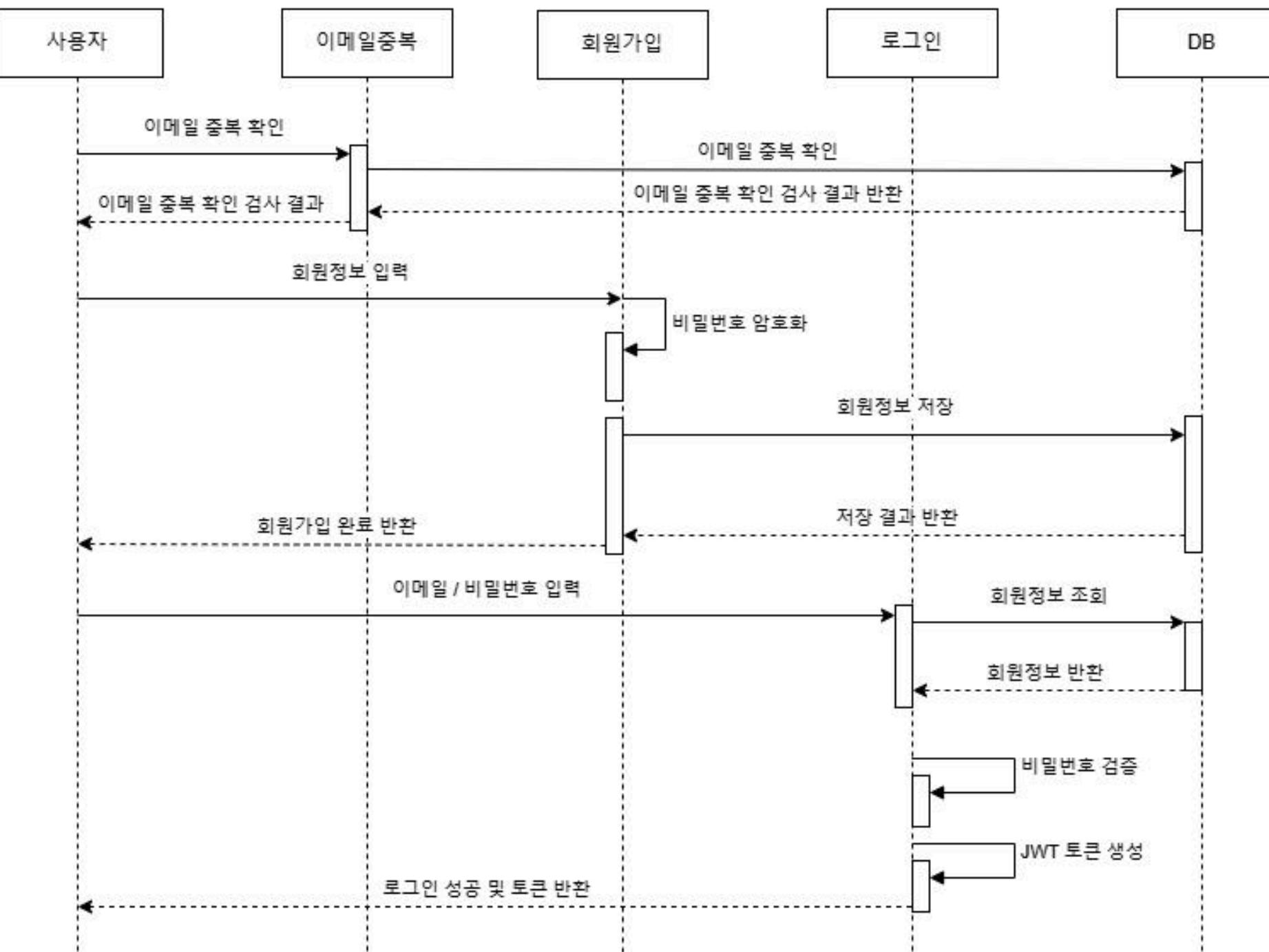
● 유즈케이스 관계 정의

- 연관 관계 (Association)
- 확장 관계 (Extend)
- ← 포함 관계 (Include)

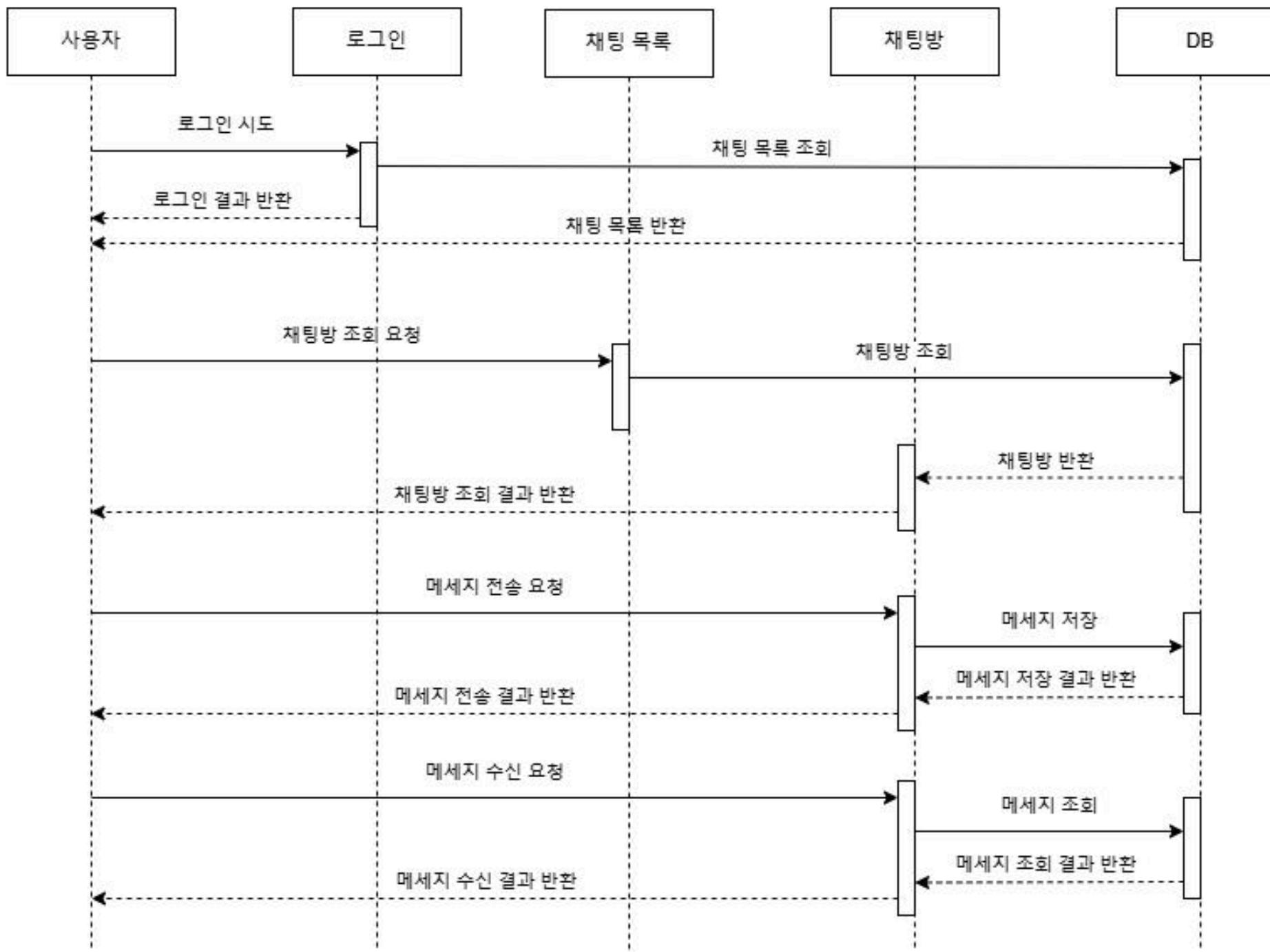




● 로그인/회원가입 페이지

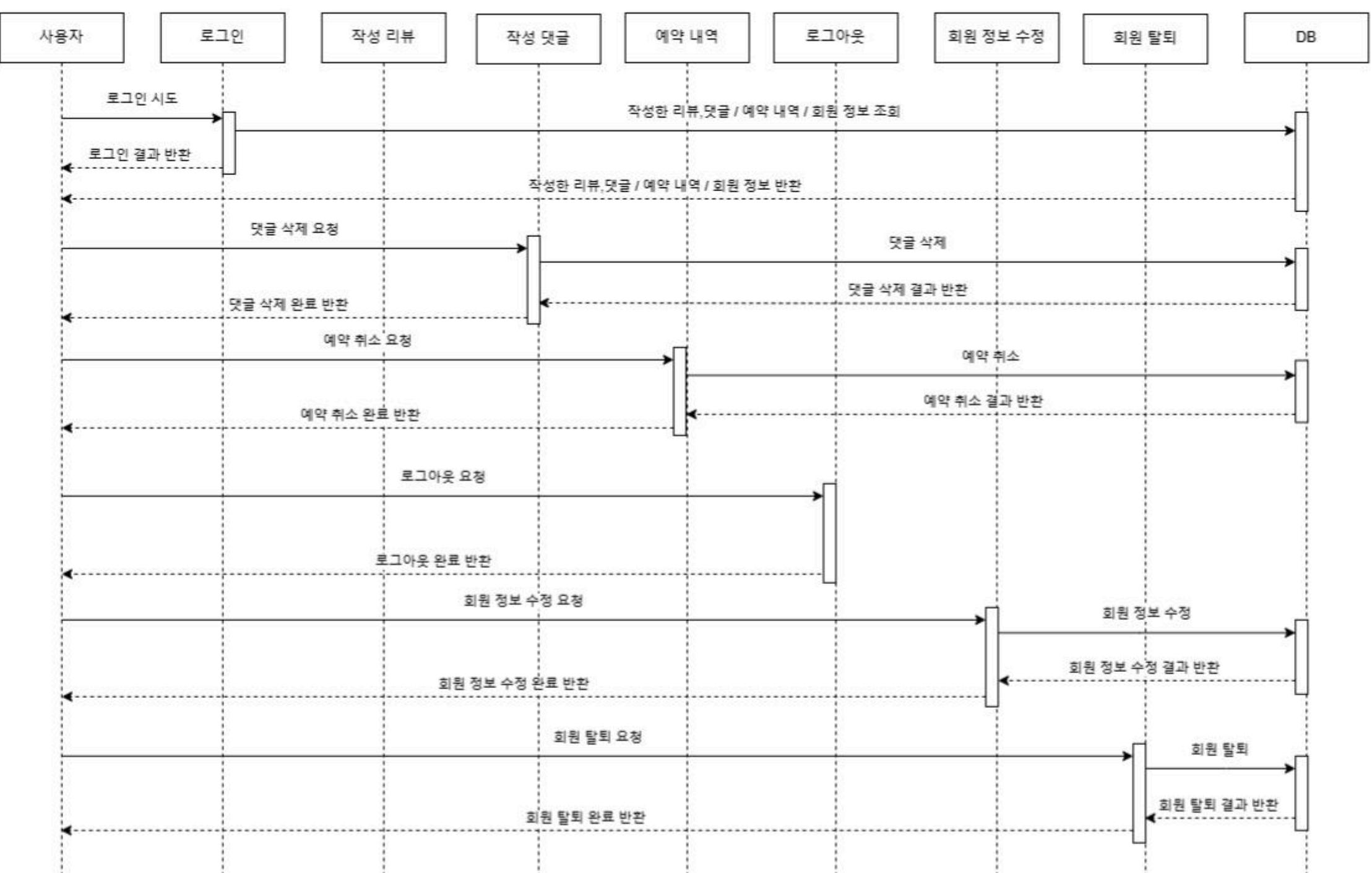


● 채팅 페이지

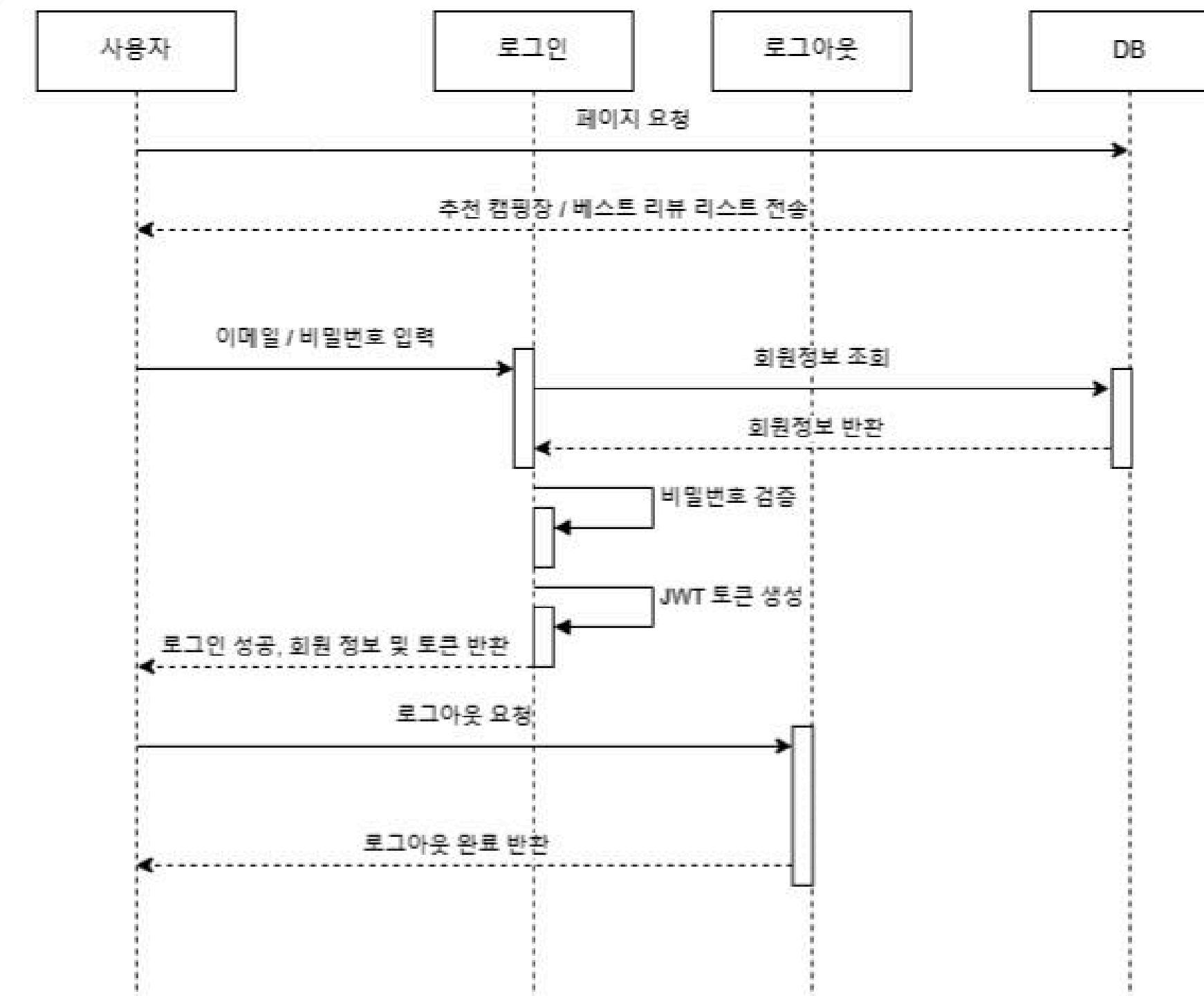




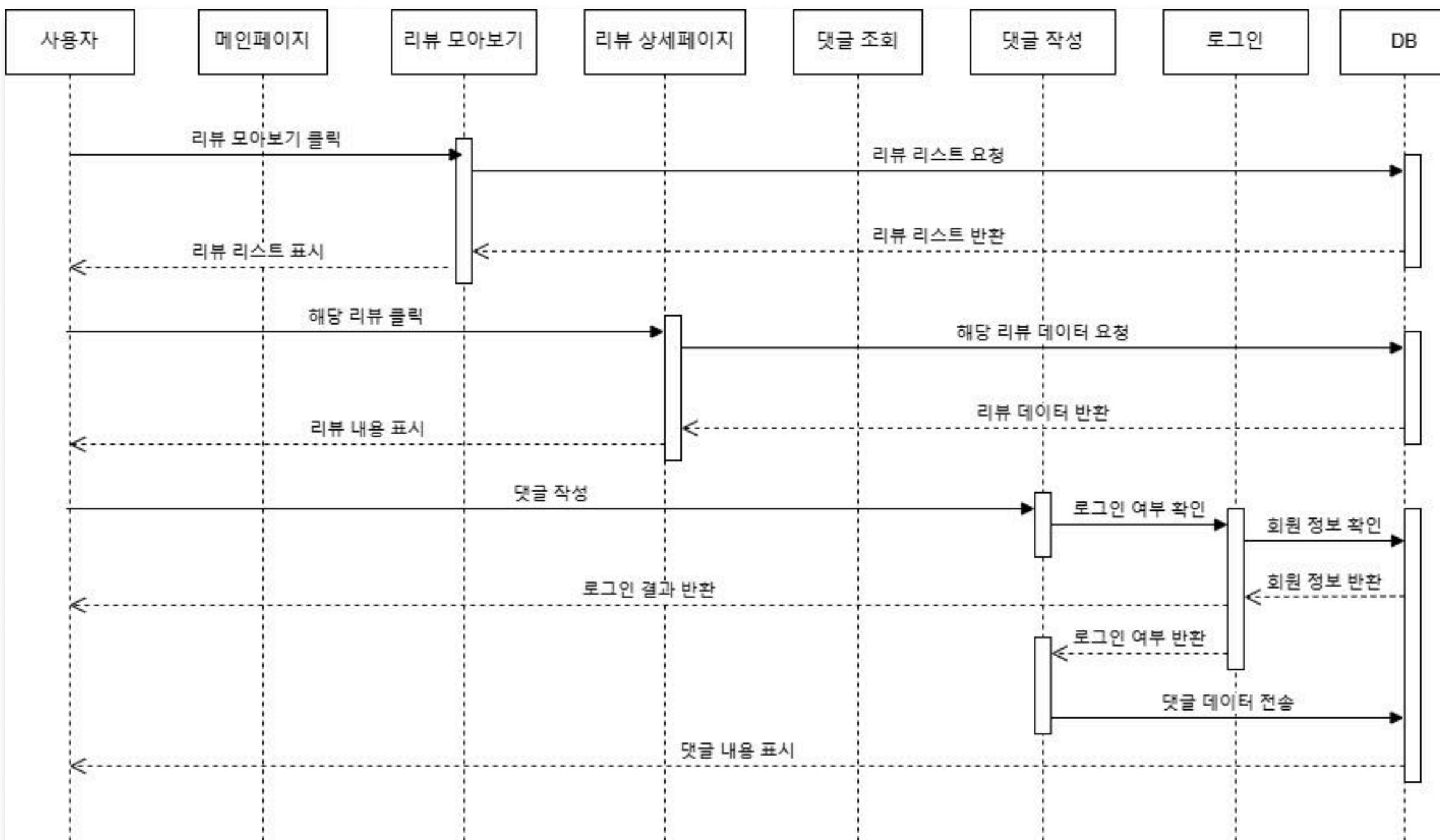
● 마이 페이지



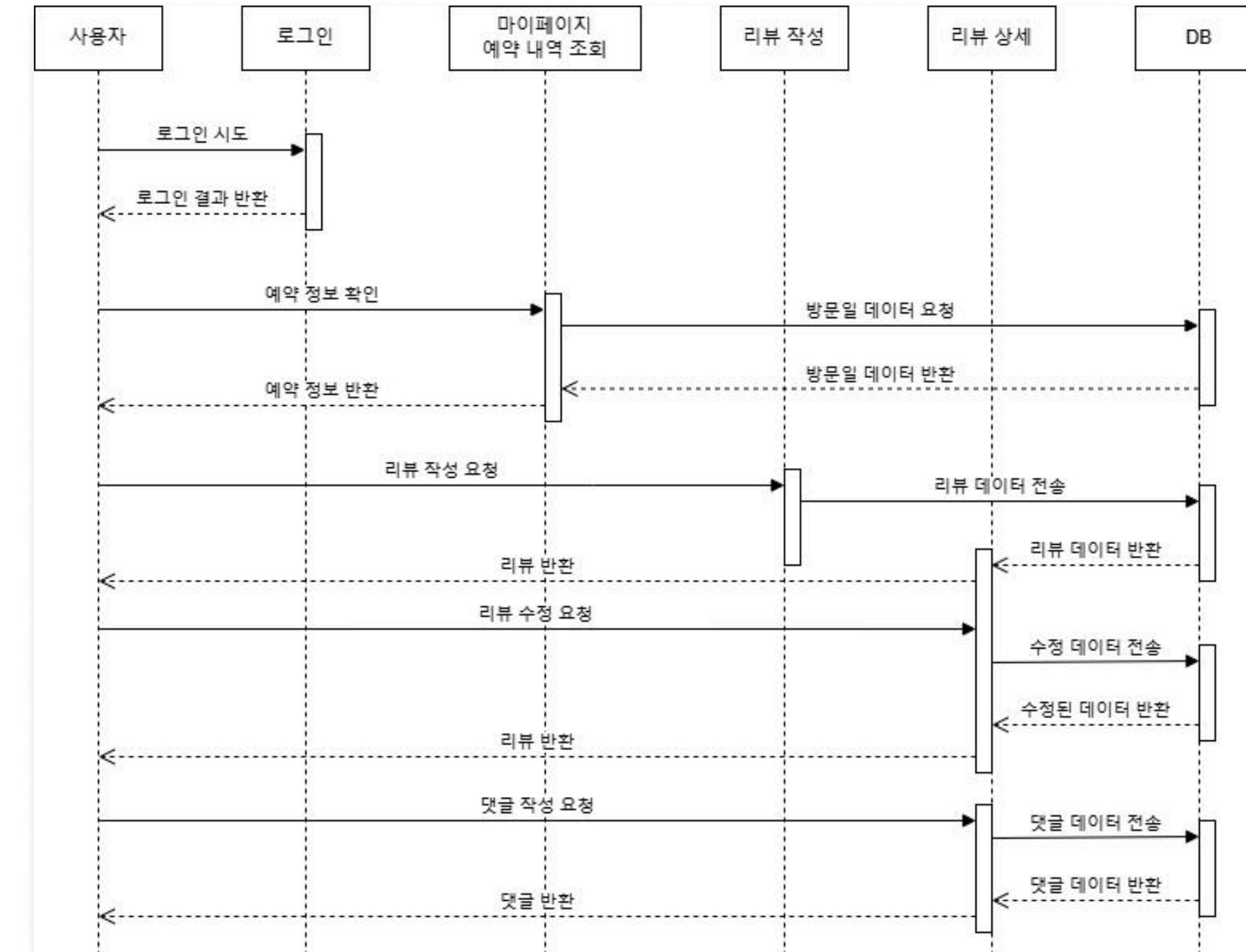
● 메인 페이지



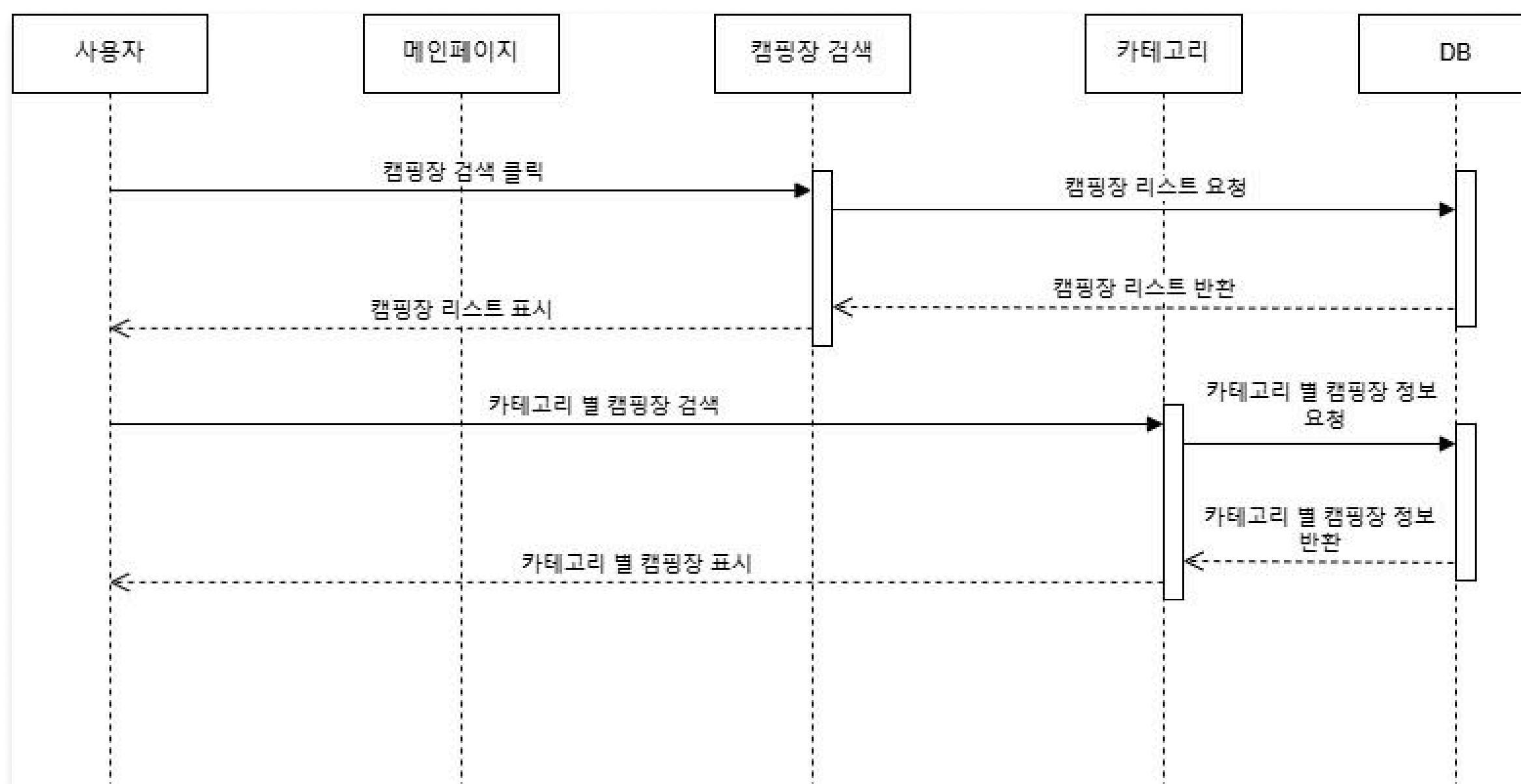
● 리뷰 리스트/디테일 페이지



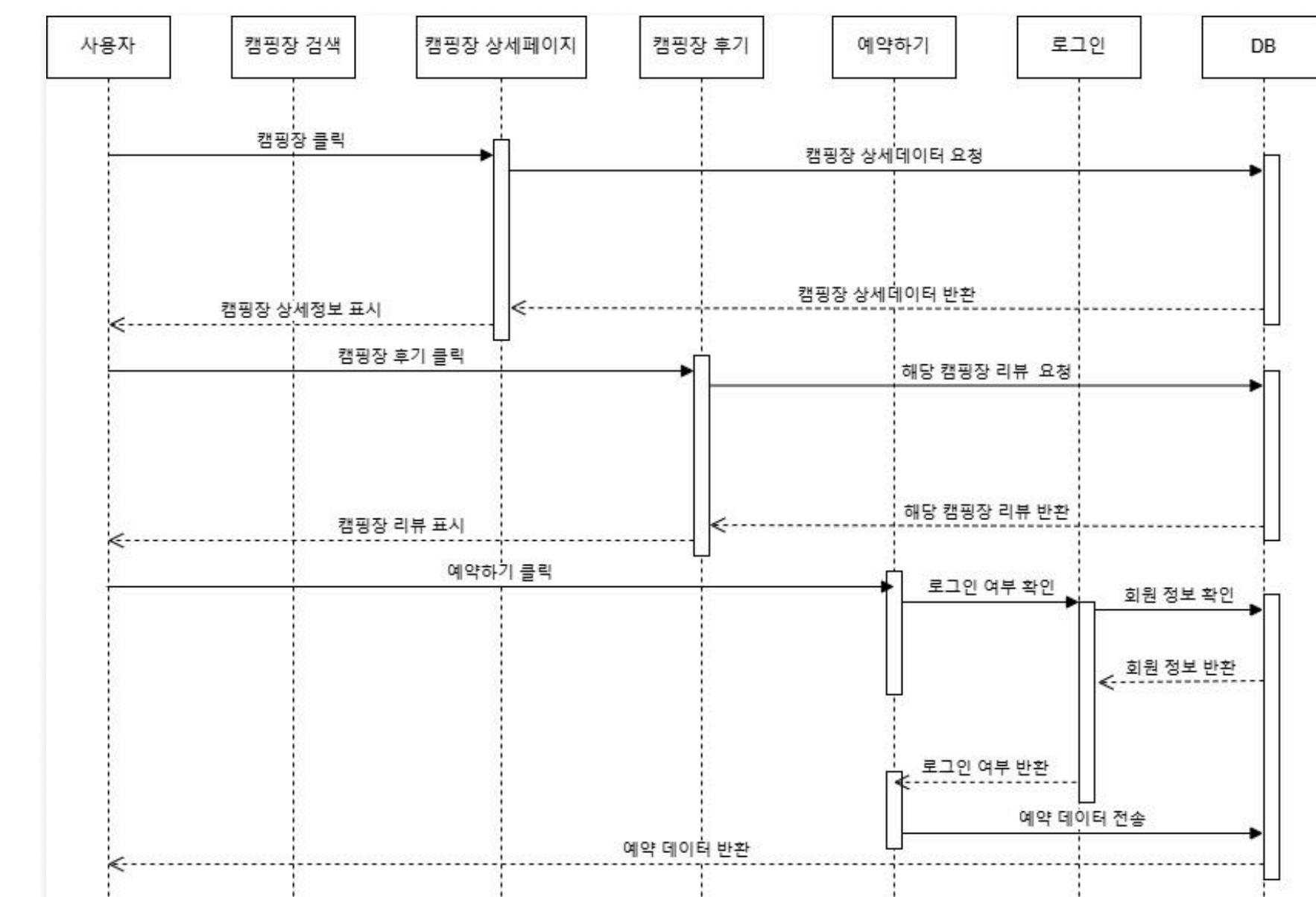
● 리뷰 작성 페이지



● 캠프 리스트 페이지

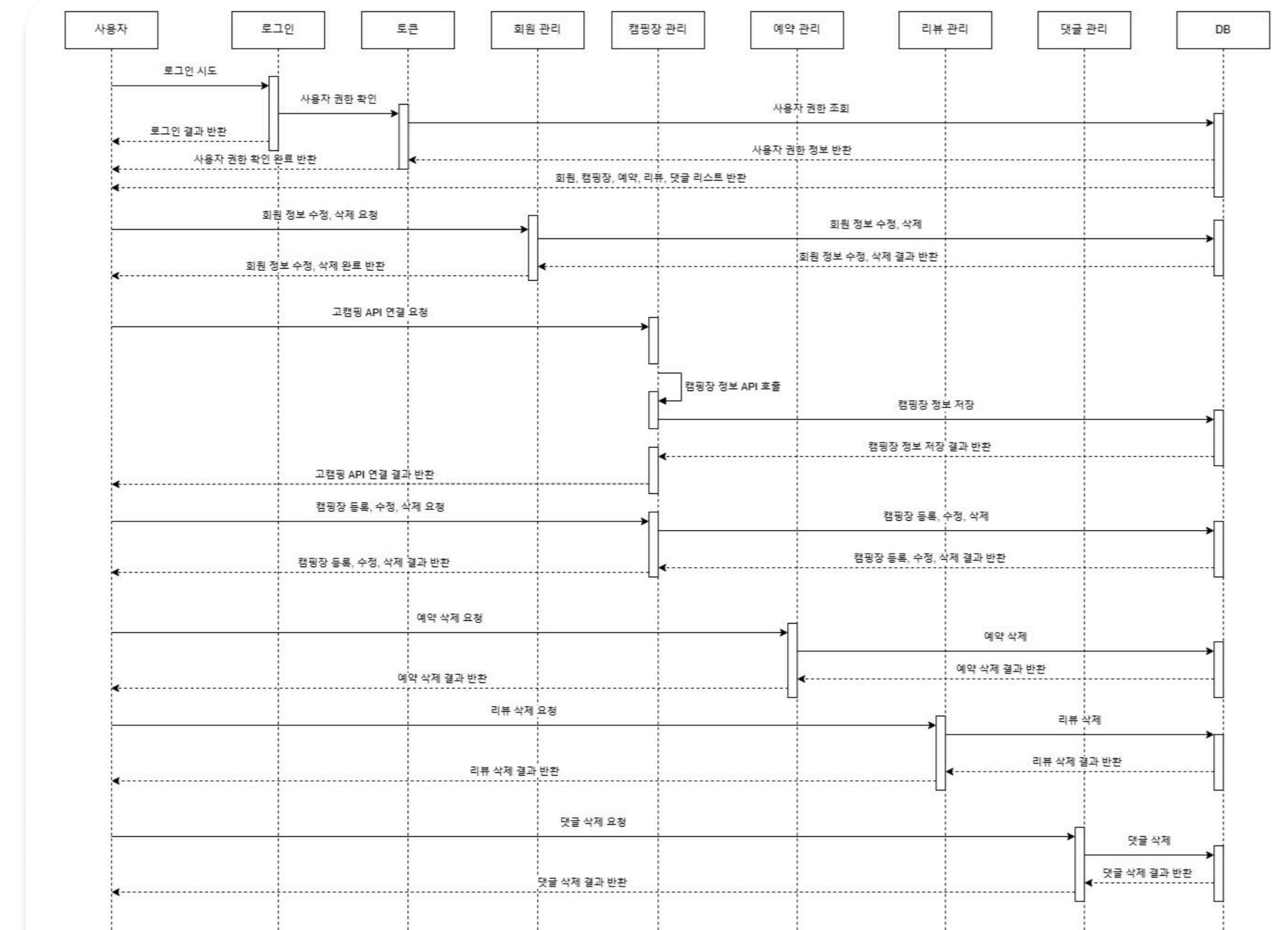


● 캠프 디테일 페이지

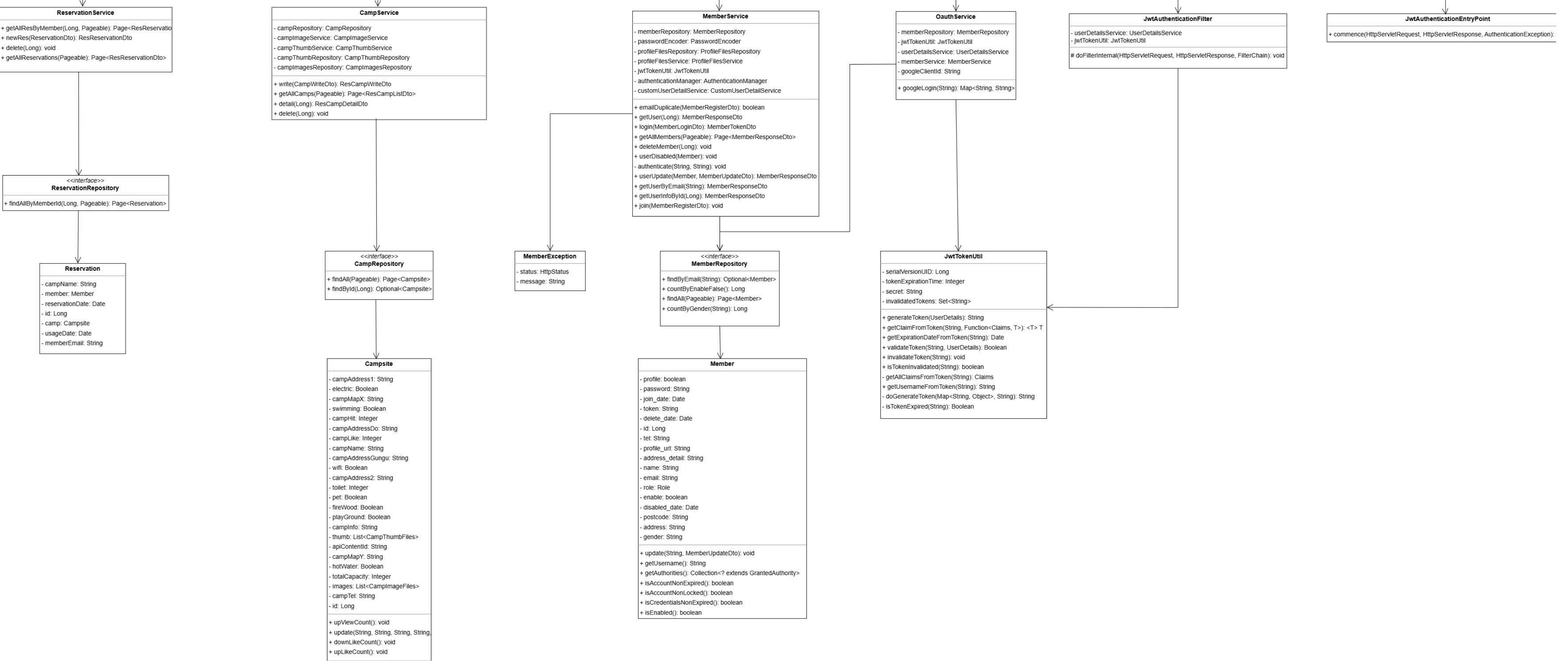


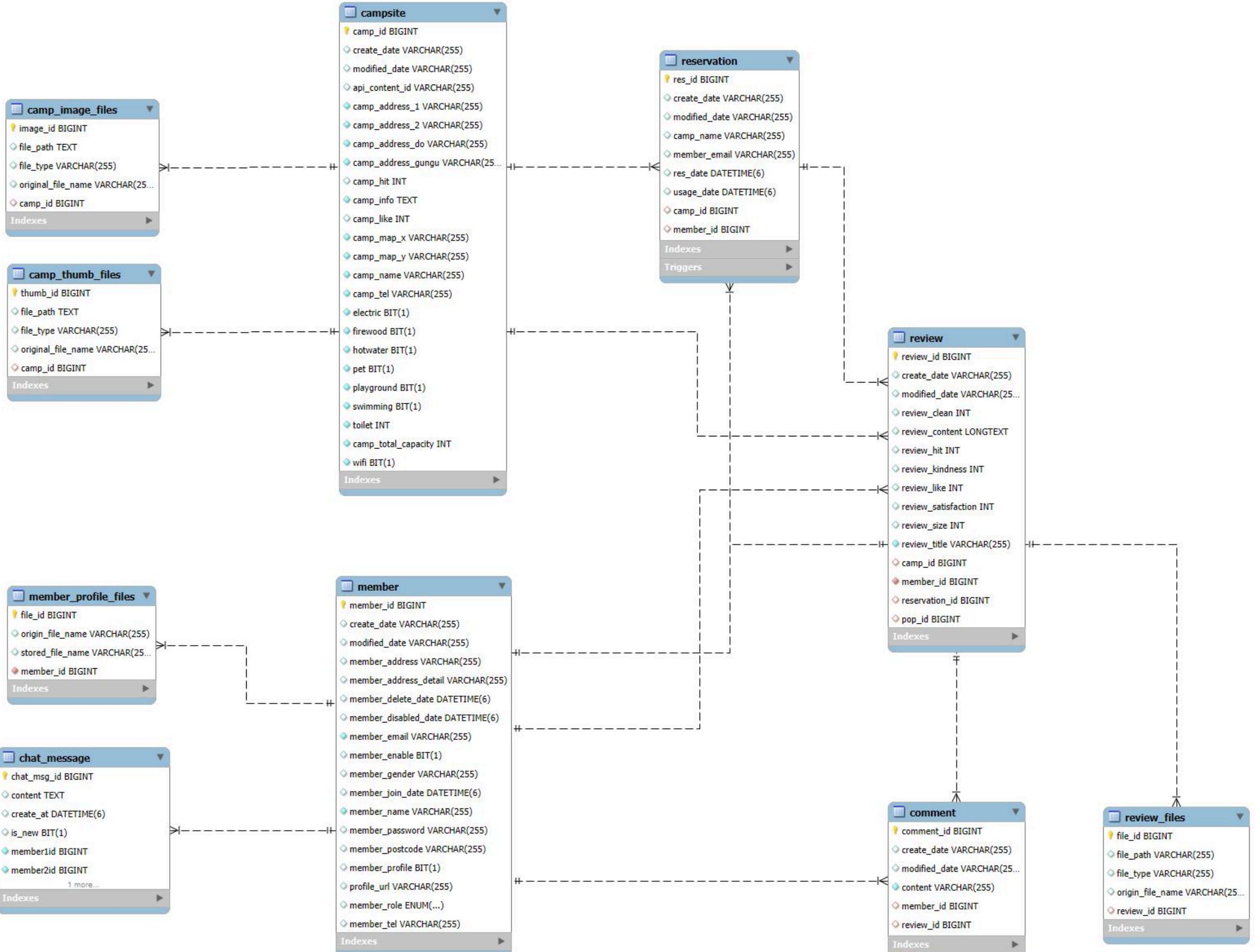


● 어드민 페이지







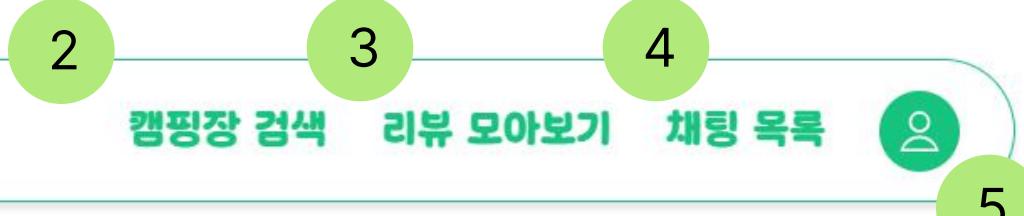
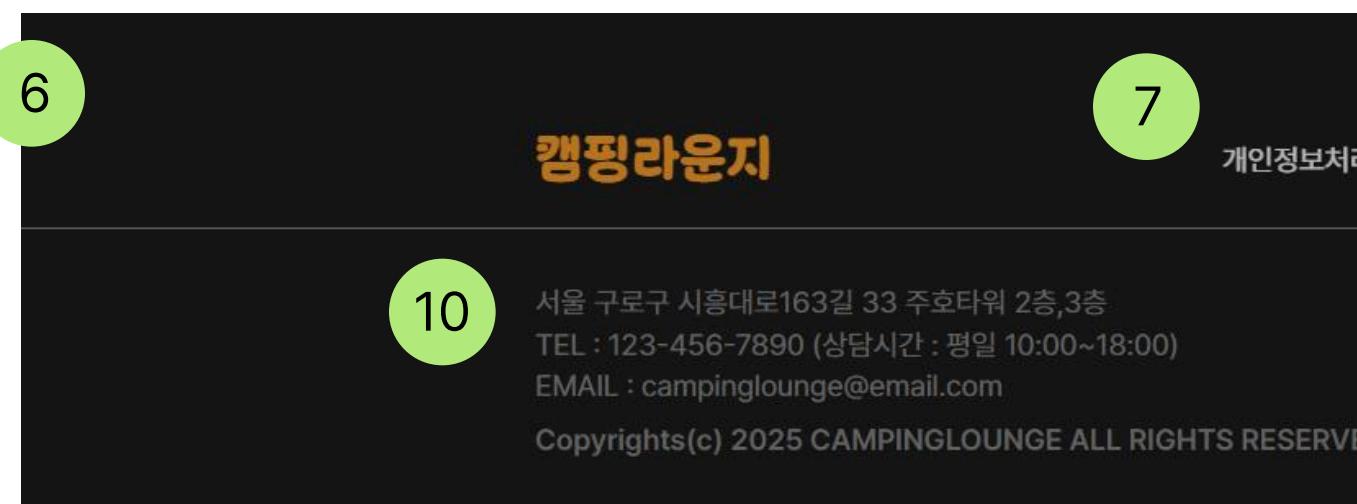
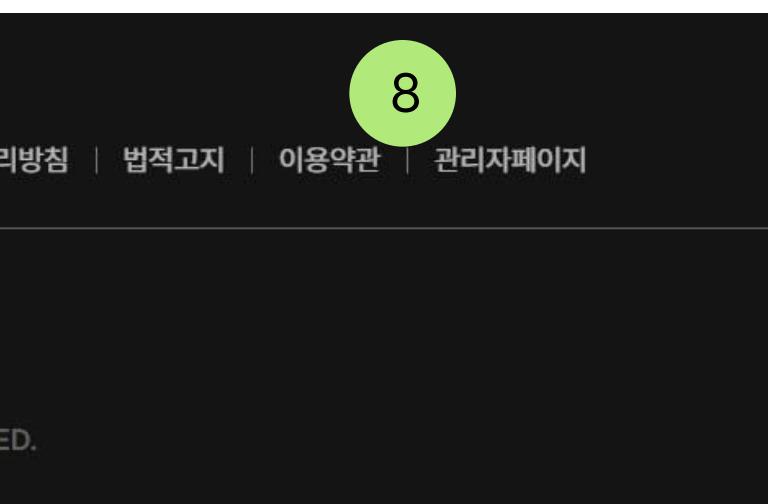
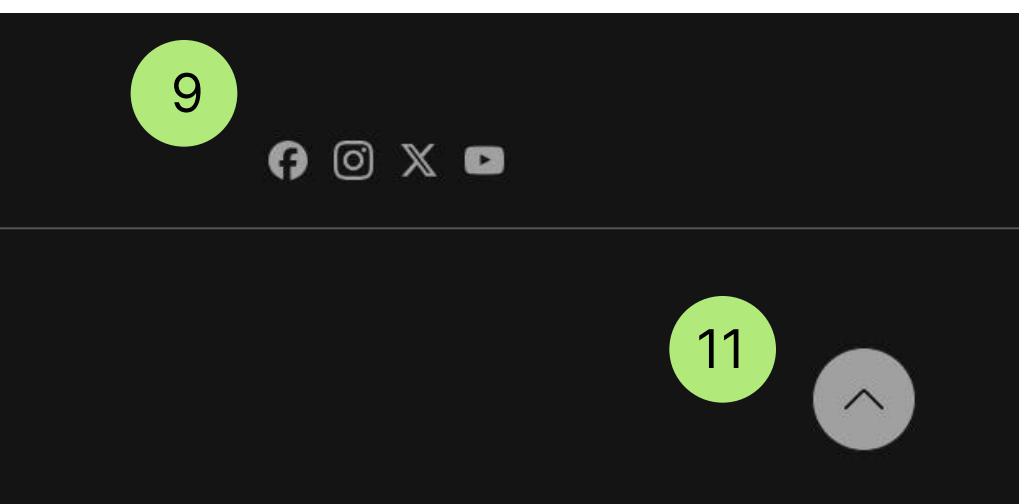
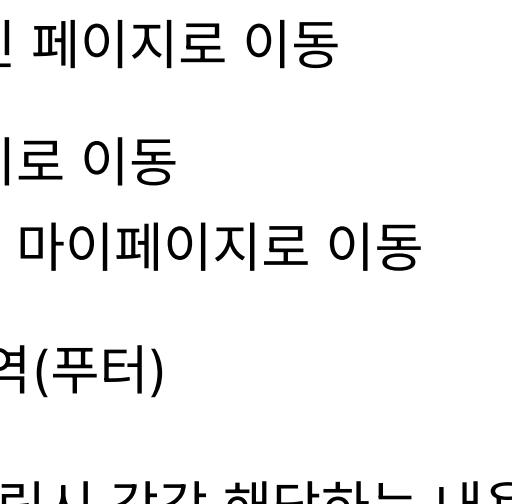
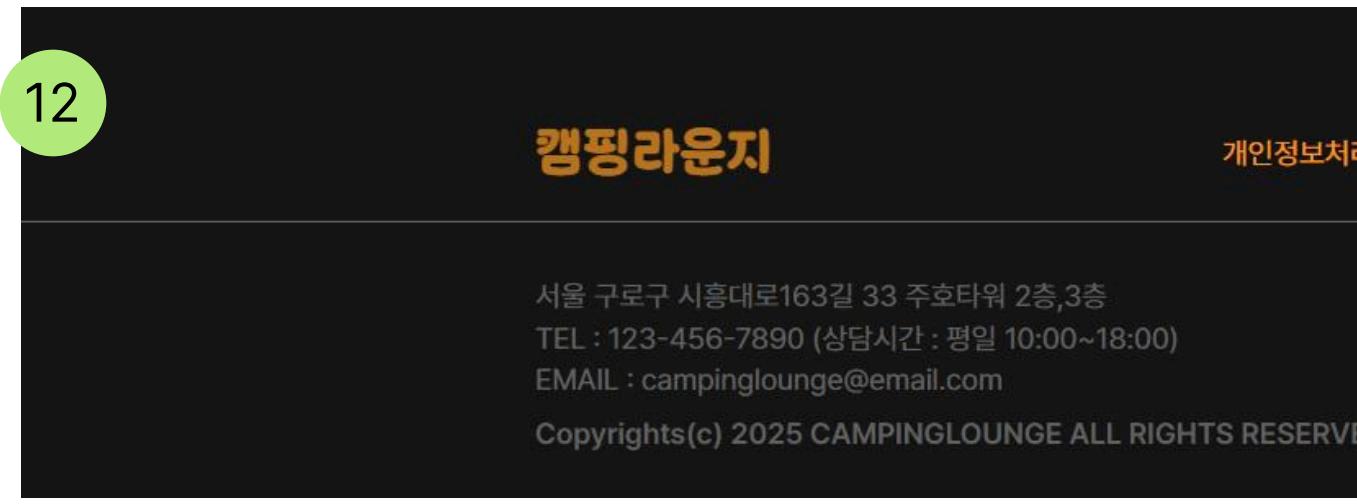
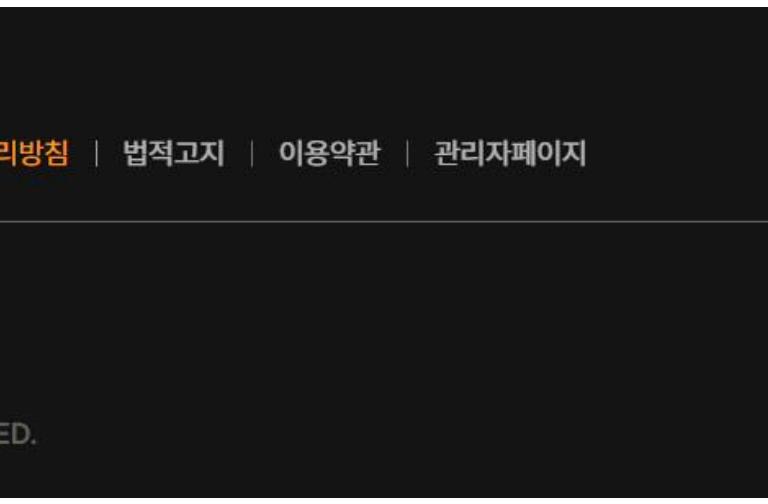
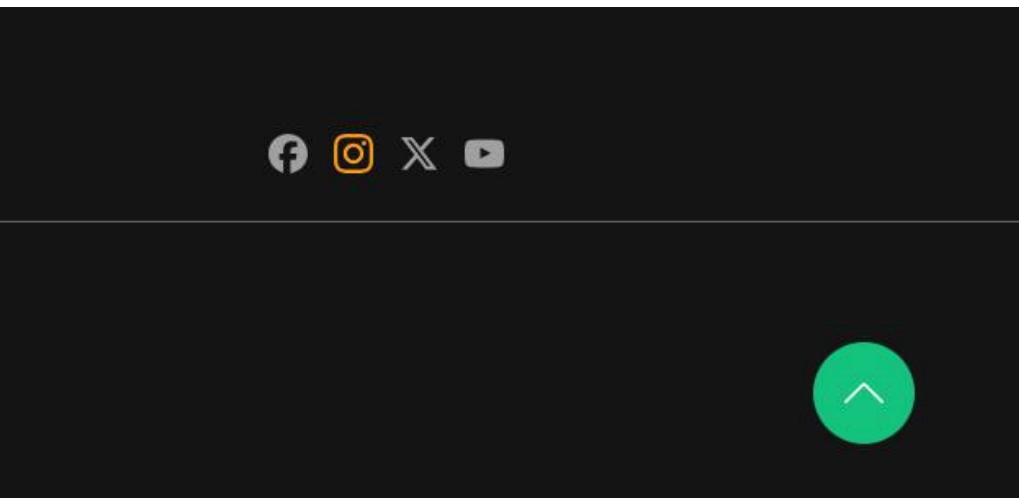
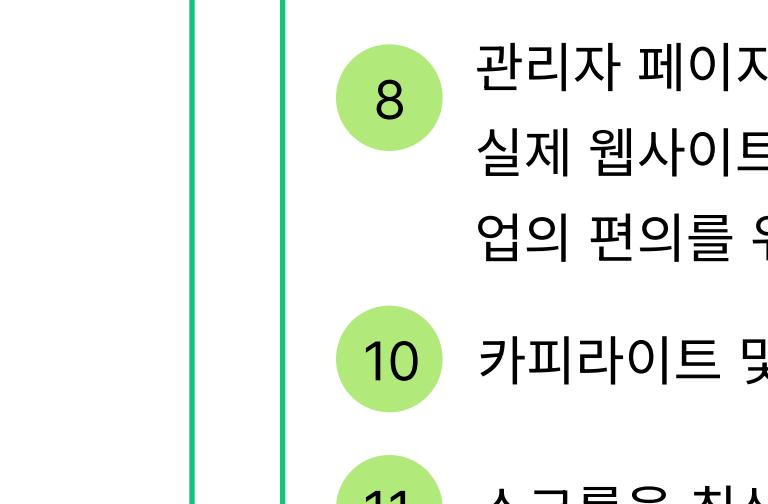


● 테이블 설명

No.	테이블 명	테이블 설명
1	campsite	캠프 정보를 저장하는데 사용되는 테이블
2	camp_image_files	캠프 이미지를 저장하는데 사용되는 테이블
3	camp_thumb_files	캠프 대표 이미지를 저장하는데 사용되는 테이블
4	member	사용자 정보를 저장하는데 사용되는 테이블
5	member_profile_files	사용자 프로필 사진을 저장하는데 사용되는 테이블
6	chat_message	채팅 메시지를 저장하는데 사용되는 테이블
7	review	리뷰를 저장하는데 사용되는 테이블
8	review_files	리뷰 이미지를 저장하는데 사용되는 테이블
9	reservation	예약 정보를 저장하는데 사용되는 테이블
10	comment	댓글을 저장하는데 사용되는 테이블

화면 설계



설명	상단 메뉴 영역, 하단 정보 영역	경로	모든 페이지 상단/하단	담당자	김성우
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
					
					
					

상세 설명

- 1 상단 메뉴 영역(헤더). 스크롤 시에도 위치가 상단에 고정되어있다.
- 2 캠핑장 목록으로 이동
- 3 리뷰 목록으로 이동
- 4 채팅 목록으로 이동. 로그인하지 않았을 경우엔 로그인 페이지로 이동
- 5 로그인 페이지로 이동
로그인시에는 마이페이지로 이동
- 6 하단 정보 영역(푸터)
- 7 푸터 메뉴. 클릭시 각각 해당하는 내용이 팝업 창으로 나타남.
- 8 관리자 페이지 이동 버튼.
실제 웹사이트라면 없을 버튼이지만, 작업의 편의를 위해 푸터에 위치시킴.
- 10 카피라이트 및 업체 정보
- 11 스크롤을 최상단으로 이동하는 버튼.
푸터에 위치한게 아닌, 화면 우측하단 고정
- 12 각 요소 호버 이펙트



설명

메인 페이지

경로

메인 페이지

담당자

김성우

상세 설명

CAMPING LOUNGE

The main page features a large, high-quality photograph of a campsite. The image shows several tents in a wooded area, with one tent's entrance open. In the foreground, there is a barbecue grill with smoke rising from it. To the left of the grill, there is a folding chair and a small table. The background is filled with dense green trees, and the overall atmosphere is bright and sunny.

1 CAMPING LOUNGE 메인 화면
각각 캠핑장 리스트 / 리뷰 게시판과
연결된 링크

2 자동 슬라이드 이미지는 메인 페이지의
중앙에 배치하여 주요 컨텐츠를 강조하는
역할을 한다.

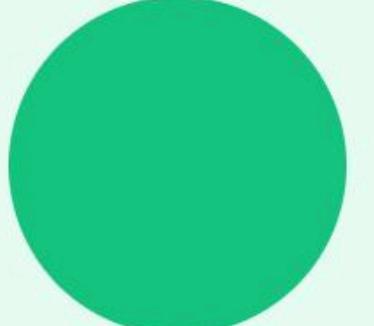
3 페이지에 들어서면 스크롤 애니메이션을
통해 사용자가 이용 방법을 직관적으로
안내받을 수 있다.

화면 설계

CAMPING LOUNGE



A screenshot of the Camping Lounge mobile application interface. At the top left is a circular profile picture placeholder labeled '캠핑라운지'. To its right are three menu items: '캠핑장 검색', '리뷰 모아보기', and '채팅 목록'. A green circular badge with the number '4' is positioned at the top right. The main content area features a large title 'CAMPING LOUNGE' above a photograph of a campsite. Below the image are two sections: '화면 설계' on the left and a numbered list of 8 steps on the right.

- 5  Login
- 6 이메일
- 6 비밀번호
- 6
- 7
- 8

- 4 헤더의 마이페이지 버튼을 클릭시 메인페이지에서는 로그인 탭이 활성화 되고 다른페이지에선 로그인페이지로 이동
- 5 로그인 탭이 활성화 되면서 페이지 레이아웃이 변경된다. 로그인을 다시 버튼 클릭시 비활성화 된다.
- 6 이메일과 비밀번호 입력 필드와 로그인버튼
- 7 구글 API를 연동하여 소셜 로그인 기능을 제공한다.
- 8 회원가입 페이지로 이동한다.

CAMPING LOUNGE



A photograph of a campsite featuring several tents, a wooden picnic table with chairs, and a campfire with smoke rising.



홍길동

9 ksoung140w@gmail.com
010-1234-5678
서울 구로구 시흥대로 585-3

10

11

[마이 페이지](#)

[로그아웃](#)

This block displays a user profile icon (a plant), the user's name (Hong Gil Dong), and their contact information (email: ksoung140w@gmail.com, phone: 010-1234-5678, address: Seoul Guro-gu Sihyeondae-ro 585-3). It also includes numbered callouts (9, 10, 11) and two buttons for 'My Page' and 'Logout'.

- 9 로그인 한 회원의 아이디와 연락처가 표시되는 영역
- 10 로그인 한 회원의 마이페이지로 이동한다.
- 11 로그아웃 버튼



설명 메인 페이지 경로 메인 페이지 스크롤 담당자 김성우

1 **추천 캠핑장**

2 캠핑장 더 보기

3

4

4

4 **캠핑라운지 오픈 이벤트!**
캠핑라운지 SNS를 팔로우하고 친구에게 공유하면 깜짝 선물이?!
[이벤트 참여](#)

상세 설명

- 1 캠핑라운지 추천 캠핑장 리스트, 회원들의 높은 평가를 받은 캠핑장을 소개한다.
- 2 캠핑라운지에 등록되어있는 모든 캠핑장을 검색할 수 있다.
- 3 스와이프 및 자동 슬라이드 기능을 지원하며, 좌우 버튼과 이미지 드래그를 통해 게시물을 확인할 수 있다.
- 4 캠핑라운지 예약 사이트 오픈 기념, SNS 친구 공유 이벤트와 함께 현재 이용자에게 특별 혜택을 제공한다.



설명 메인 페이지 경로 메인 페이지 스크롤 담당자 김성우

화면 설계

베스트 리뷰

1 그자리 야영장 후기
그자리 야영장

2 김천오토캠핑장 후기
김천실내테니스장 & 김천오토캠핑장

3 고래마을 후기
호미곶 고래마을 캠핑장

리뷰 모두 보기 00

< >

2 CampingLounge. made by Kim Seongwoo, Yang Chansik, Son Suyong, Choi Sumin. This webpage is not used for any purpose other than learning. | CampingLounge. made by Kim Se

상세 설명

- 조회수 높은 리뷰가 순차적으로 베스트 리뷰로 선정되어 메인 페이지에 노출된다.
- 실시간으로 확인할 수 있는 슬라이드형 공지 바



설명 로그인 페이지 경로 헤더 > 로그인 담당자 손수용

The screenshot shows a login form with the following elements:

- 1. Email input field labeled "이메일" (Email) with placeholder "이메일".
- 2. Password input field labeled "비밀번호" (Password) with placeholder "비밀번호".
- 3. A green button labeled "이메일로 로그인" (Log in with Email).
- 4. A button labeled "Google 계정으로 로그인" (Log in with Google account) featuring the Google logo.
- 5. An orange button labeled "회원가입" (Sign Up).

상세 설명

- 1 이메일 입력 필드
- 2 비밀번호 입력 필드
- 3 로그인 버튼
DB에서 회원 정보를 조회하여 이메일과 비밀번호 일치시 로그인 허용
- 4 구글 API를 연동하여 소셜 로그인 기능을 제공한다
- 5 회원가입 페이지로 이동



설명 일반 유저 회원가입 페이지 경로 헤더 > 로그인 > 회원 가입 담당자 손수용

회원 가입

* 은 필수 입력 항목입니다.

1 * 이메일
 중복 확인

* 비밀번호

* 비밀번호 확인

* 이름

* 전화번호

2 주소
우편번호 주소 찾기

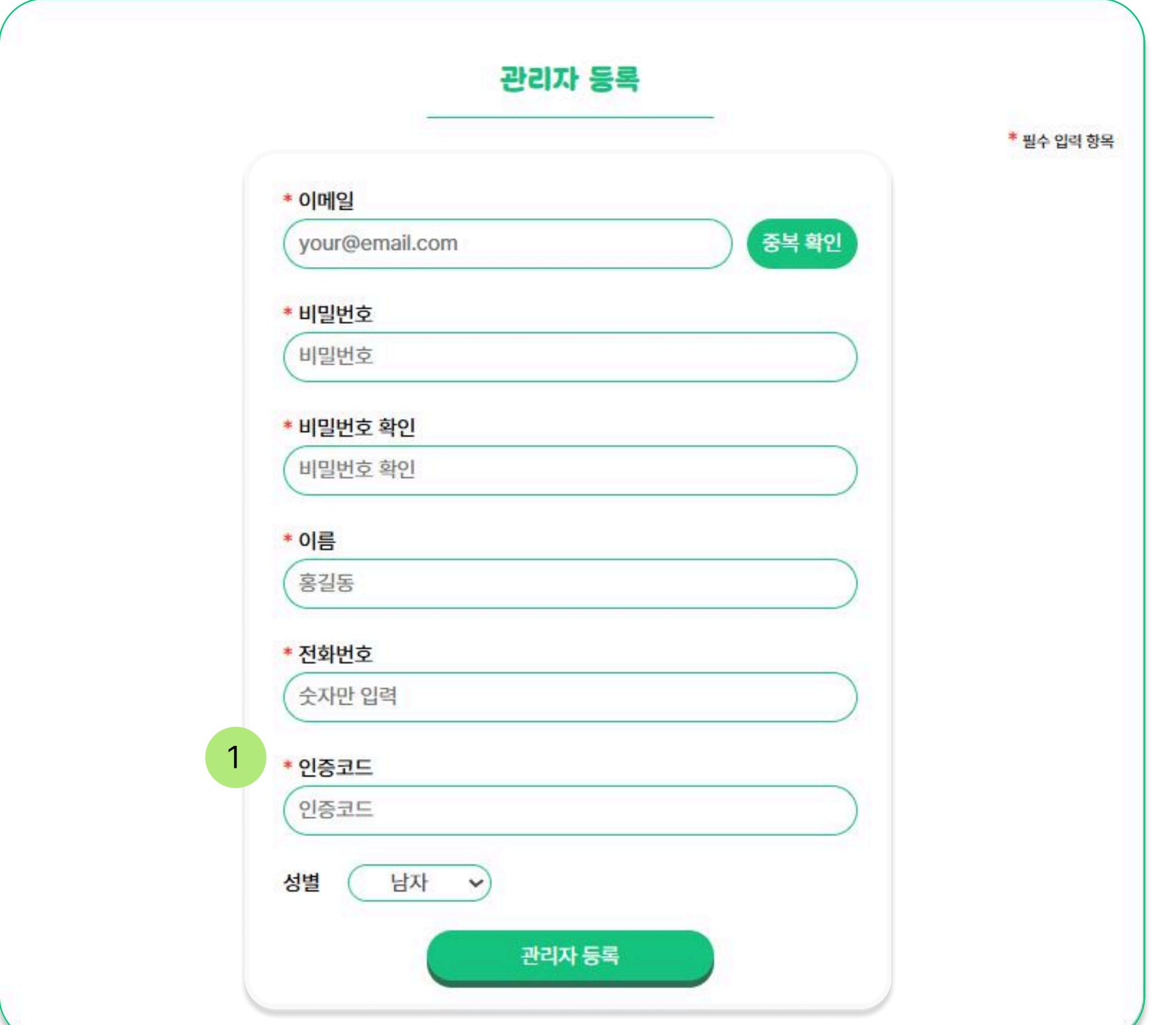
성별 남자

3 회원 가입

상세 설명

- 1 일반 사용자가 회원가입시 입력해야 할 필수 입력 항목.
- 2 회원가입시 주소를 입력할 수 있으며 선택 사항이다.
- 3 가입 시 필요한 정보들을 전부 입력하고 회원가입 버튼을 클릭하면 회원가입이 완료되고, 메인 페이지로 이동한다.



설명	관리자를 등록하는 페이지	경로	관리자 등록 경로	담당자	손수용
화면 설계					
	1				

상세 설명

- 1 관리자용 회원가입 페이지.
일반 사용자용 회원가입 페이지와는 다른 경로로 접근하며
미리 발급받은 인증코드를 사용해
가입이 가능하고
캠핑라운지의 모든 등록, 삭제, 수정의
권한을 가진 계정을 생성한다.

설명

자신의 회원정보를 조회 하는 페이지

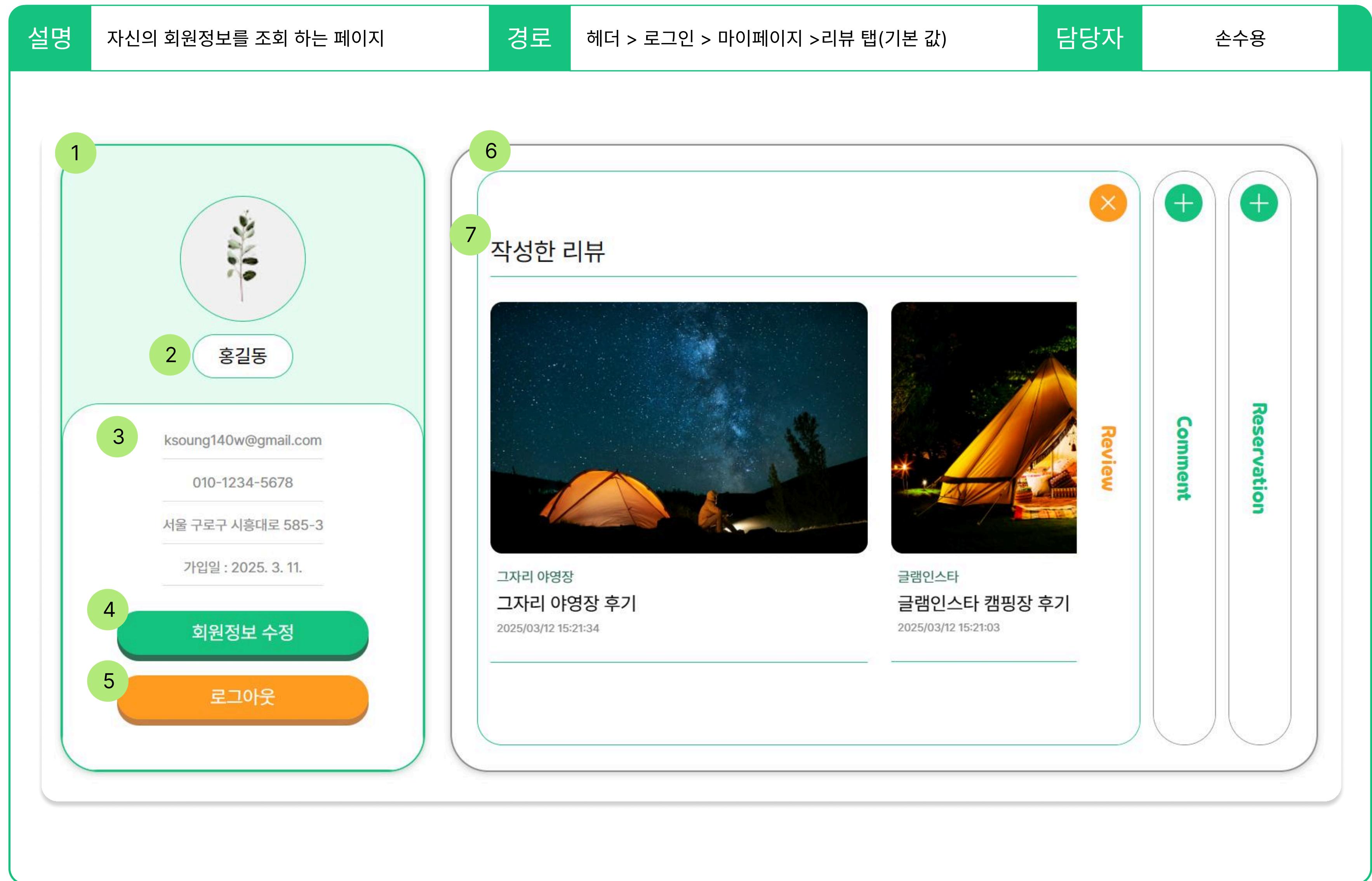
경로

헤더 > 로그인 > 마이페이지 >리뷰 텁(기본 값)

담당자

손수용

화면설계



상세 설명

- 회원 프로필 이미지. 등록된 이미지가 없으면 기본 이미지로 표시
 - 회원 이름
 - 회원 정보영역. 이메일, 전화번호, 주소, 가입일을 조회한다
 - 회원 정보 수정 페이지로 이동
 - 로그아웃 버튼
 - 회원 활동정보 영역
리뷰 탭, 댓글 탭, 예역내역 탭으로 나뉜다.
 - 리뷰탭
회원이 작성한 리뷰 목록을 조회한다.
게시물 목록을 슬라이드 메뉴로 확인 가능.
리뷰를 클릭하면 해당 리뷰 페이지로 이동.



설명

자신의 회원정보를 조회 하는 페이지

경로

헤더 > 로그인 > 마이페이지 > 댓글 탭

담당자

손수용

화면 설계

The screenshot shows the 'Comment' tab selected in the MyPage (2) interface. On the left, there's a sidebar with user information: profile picture (green circle with a plant), name (Hong Gil Dong), email (ksoung140w@gmail.com), phone number (010-1234-5678), address (Seoul Guro-gu Sihung-dong 585-3), and joining date (2025. 3. 11). Below this are two buttons: '회원정보 수정' (Edit Member Information) and '로그아웃' (Logout). The main content area has three tabs: 'Review' (selected), 'Comment' (orange), and 'Reservation'. The 'Comment' tab displays a list of comments with columns for content, author, date, and delete button. There are also '+' and '-' buttons for adding or removing comments. At the bottom is a pagination bar with icons for navigating between pages.

댓글 내용	작성글	작성일	삭제
멋진 캠핑장이군요	첫 캠핑 다녀왔어요~	2025/03/12 15:43:06	[삭제]
주변에 편의점이 있나요?	봉즈살롱 리얼 후기	2025/03/12 15:42:53	[삭제]
비용은 얼마나 들었나요?	캠핑장 후기입니다😊	2025/03/12 15:42:25	[삭제]
멋진 캠핑장이네요!	청라 캠핑장 솔직 후기	2025/03/12 15:42:11	[삭제]
추천합니다 🌟	그자리 야영장 후기	2025/03/11 16:04:00	[삭제]

상세 설명

1 댓글 탭

작성한 댓글 목록을 조회한다

2 댓글 테이블

작성한 댓글의 내용, 작성 위치, 작성일을 조회할 수 있다.

3 삭제 버튼

해당 댓글을 삭제한다

4 페이지네비게이션

다음/이전 댓글 목록을 조회한다



설명

자신의 회원정보를 조회 하는 페이지

경로

헤더 > 로그인 > 마이페이지 > 예약 내역 탭

담당자

손수용

화면 설계

The screenshot shows the 'Reservation' tab of the MyPage. On the left, there's a sidebar with a profile icon (green leaf), a placeholder for a photo ('홍길동'), and contact information: email (ksoung140w@gmail.com), phone number (010-1234-5678), address (서울 구로구 시흥대로 585-3), and joining date (가입일 : 2025. 3. 11.). Below these are two buttons: '회원정보 수정' (Edit Member Information) and '로그아웃' (Logout). The main content area has three tabs: 'Review' (Review), 'Comment' (Comment), and 'Reservation' (Reservation). The 'Reservation' tab is active, showing a list of bookings. The first booking is for '우니메이카 밀양점' on March 12, 2025, with a visit on March 12, 2025. It includes a green '후기 작성' (Write Review) button and a red '취소 불가' (不可取消) button. The second booking is for '글램인스타' on March 11, 2025, with a visit on June 12, 2025. It includes a green '후기 작성' button and a red '예약 취소' (Cancel Reservation) button. The third booking is for '글램인스타' on March 11, 2025, with a visit on March 31, 2025. It includes a green '후기 작성' button and a red '예약 취소' button. At the top of the reservation list are four numbered callouts: 1. 예약 내역 (Reservation List), 2. 예약 테이블 (Reservation Table), 3. 후기 작성 버튼 (Review Button), and 4. 예약 취소 버튼 (Cancel Reservation Button).

캠핑장	예약 확정일	방문일	후기	취소
우니메이카 밀양점	2025. 3. 12.	2025. 3. 12.	후기 작성	취소 불가
글램인스타	2025. 3. 11.	2025. 6. 12.	후기 작성	예약 취소
글램인스타	2025. 3. 11.	2025. 3. 31.	후기 작성	예약 취소

상세 설명

- 1 예약 내역 탭
예약한 내역을 조회한다

- 2 예약 테이블
예약한 캠핑장, 예약 확정일, 방문 예정일을 조회할 수 있다

- 3 후기 작성 버튼
해당 캠핑장의 방문 후기 작성 페이지로 이동한다

- 4 예약 취소 버튼
예약을 취소하는 버튼. 방문일 당일과 방문일 이후에는 취소할 수 없다.



설명

자신의 회원정보를 수정 하는 페이지

경로

헤더 > 로그인 > 마이페이지 > 회원 정보 수정

담당자

손수용

화면 설계

회원 정보 수정

1 프로필 이미지
2 프로필 이미지
파일 선택 선택된 파일 없음
3 *이름
홍길동
4 *전화번호
010-1234-5678
5 주소
08391 주소 찾기
서울 구로구 시흥대로 585-3
샘플아파트 123동 345호
6 회원 정보 수정
7 회원 탈퇴

상세 설명

- 1 현재 프로필 사진
사진을 변경하면 해당 사진으로 변경된다
- 2 프로필 이미지 수정 필드
수정할 이미지를 업로드한다
- 3 이름 수정 필드
- 4 전화번호 수정 필드
- 5 주소 수정 필드
카카오 주소찾기 API를 연동하여 주소를
검색할 수 있다.
- 6 회원 정보 수정버튼
변경한 내용으로 정보를 수정한다
- 7 회원 탈퇴 버튼
회원을 비활성화 처리한다.
비활성화 된 회원은 DB에서 인식하여
30일 뒤 정보가 삭제되는 트리거를 실행
한다



설명

다른 회원 정보를 조회하는 페이지

경로

헤더 > 리뷰 목록 > 리뷰 상세 > 회원 이름 > 회원 조회

담당자

손수용

화면 설계

돌이아빠

1

admin@admin.com

가입일 : 2025. 3. 11.

2

3

4

작성한 리뷰

청라캠핑파크
청라에도 이런 캠핑장이?
2025/03/11 18:04:50

청라캠핑파크
캠핑장 후기입니다😊
2025/03/11 18:04:15

Review

Review

채팅하기

상세 설명

- 다른 맴버의 프로필 영역
프로필 이미지, 이름 조회
- 다른 맴버의 정보
이메일, 가입일 조회 가능
- 채팅 버튼
해당 맴버와 채팅을 시작한다.
채팅 페이지로 이동
- 리뷰 조회 탭
해당 맴버가 작성한 리뷰를 조회한다



설명

맴버간 채팅을 조회하고 수신/발신하는 페이지

경로

헤더 > 채팅 목록

담당자

양찬식

화면 설계

캡핑라운지

캠핑장 검색 리뷰 모아보기 채팅 목록

채팅 목록

유다은
맞아요 주인분이 친절해서 좋았어요!
2025. 3. 13. 오후 2:46:09

한승호
혹시 비슷한 가격대의 캠핑장을..
2025. 3. 13. 오후 2:45:21

문자수
다음엔 같이 한번 방문해보시죠!
2025. 3. 13. 오후 2:44:29

돌이아빠
작성하신 리뷰를 보고 문의 드릴게있는데요
2025. 3. 13. 오후 2:39:13

홍길동
방문하셨던 캠핑장에 대해 궁금한게 있는데 대화 괜
2025. 3. 13. 오후 2:32:40

돌이아빠
안녕하세요?
2025. 3. 13. 오후 2:38:48

돌이아빠
작성하신 리뷰를 보고 문의 드릴게있는데요
2025. 3. 13. 오후 2:39:13

어떻게 궁금하신가요?
2025. 3. 13. 오후 2:46:49

메시지를 입력하세요.

발송

상세 설명

1 채팅 목록 영역

채팅을 주고받은 맴버 목록을 조회한다

2 해당 맴버의 이름, 마지막 채팅, 마지막 채팅 발송 날짜/시간 조회 가능
클릭 시 채팅방 전환3 채팅방 영역
선택한 맴버와의 채팅방
왼쪽엔 상대방의 메세지, 오른쪽엔 본인의 메세지가 표시된다.4 채팅 작성 필드
발송할 메세지를 작성하고 발송한다



설명

캠핑장을 검색하고 조회하는 페이지

경로

헤더 > 캠핑장 검색

담당자

김성우

화면 설계

1 캠핑장 필터
2 캠핑장 검색창
3 캠핑장 리스트
4 좋아요 버튼
5 URL 복사 버튼

캠핑장 이름	위치	설명
우니메이카 안동점	경상북도 안동시	우니메이카 안동점 깔끔하고 풍경 좋은 카라반&오토캠핑장 우니메이카 안동점은 경북 안동시 남후면 무릉리에 자리 잡았다. 안동시청...
그자리 야영장	강원도 홍천군	애견과 함께 놀수 있는 캠핑장 밖은 공기와 계곡이 바로 옆에 있는 그자리 야영장. 애견과 함께 놀수 있는 운동장도 있어 자유롭게 놀수...
명덕농원	경기도 포천시	
밀양 도깨비방망이 캠핑장	경상남도 밀양시	
카라반파크 아마존	경기부드 아마존	
관광농원금산만악리수목원오토캠핑장	충청남도 금산군	

상세 설명

- 1 캠핑장 필터
QueryDSL의 동적 쿼리 생성 기능을 활용하여 해당 정보를 포함한 캠핑장을 필터링해 조회하는 버튼
활성화된 필터는 색이 들어가 강조 됨
- 2 캠핑장 검색창
입력한 정보를 포함한 캠핑장을 조회하는 검색창
- 3 캠핑장 리스트
클릭 시 해당 캠핑장의 상세 정보로 이동하는 캠핑장 리스트
이름, 위치, 설명, 썸네일 정보를 조회
- 4 좋아요 버튼
해당 캠핑장의 좋아요 수를 1개 올리는 버튼
토글 버튼으로 1증가와 1감소 가능
- 5 URL 복사 버튼
해당 캠핑장의 상세 정보 페이지의 URL을 클립보드에 복사하는 버튼



설명

캠핑장을 검색하고 조회하는 페이지

경로

헤더 > 캠핑장 검색

담당자

김성우

화면 설계

6

밀양 도깨비방망이 캠핑장



경상남도 밀양시
공기 좋은 밀양 단장면의 시골마을에 위치한 폐교를 개조한 폐교캠핑장으로 털바꿈한 공간 밀양 도깨비방망이 캠핑장은 ...

카라반파크 아마존



전라북도 완주군
카라반파크 아마존은 완주에 있는 4,443m²의 대형 카라반 캠핑장입니다. 총 25동의 카라반 캠핑장과 부대시설로 이루어져 있으며 A동...

관광농원금산만악리수목원오토캠핑장



충청남도 금산군
대전 인근에 위치한 렉셔리 캠핑장, 2만여평의 수목원 보유, 카라반, 오토캠핑, 카페, 바베큐 등 자주 찾는 명소수목원에 위치해 더없이 ...

호두마루글램핑



경상남도 사천시
남해바다 노을맞집 캠핑장 일상에서 벗어나 아름다운 풍경과 여유로 움을 누릴 수 있는 호두두마루글램핑은 침대, 소파, 화장실 및 샤워...

경상북도교육청김천오토캠핑장



경상북도 김천시
경상북도 김천시 증산면에 위치한 김천오토캠핑장사이트 간격이 넓은 깔끔한 캠핑장 경상북도교육청김천오토캠핑장은 경북 김천시 증...

마을야영장, 침산추월



세종시 세종시
마을 야영장, 침산추월은 마을과 공생하는 침산추월 사회적협동조합을 설립하여, 주민 분들이 소통하고 협력하며 일하는 구조를 통해 ...

7

상세 설명

6

Hover 애니메이션
마우스 커서가 위치한 캠핑장에 Hover 애니메이션을 추가하여 강조

7

페이지 네비게이션
원하는 페이지로 이동하는 버튼



설명

캠핑장의 상세 정보를 확인하는 페이지

경로

헤더 > 캠핑장 검색 > 캠핑장 > 캠핑장 정보 탭(기본 값)

담당자

김성우

화면 설계



상세 설명

- 1 캠핑장의 대표 이미지가 상세 페이지 상단에 표시된다.
- 2 버튼 클릭 시 하단이 후기 목록으로 전환 됨 해당 캠핑장을 이용했던 회원들이 작성한 리뷰들을 확인할 수 있다.
- 3 캠핑장을 이용하기 위해선 예약을 해야하고, 예약은 캠핑라운지의 회원에게 제공되는 서비스이기 때문에 로그인이 되어있다면 예약 페이지로, 로그인이 되어있지 않으면 로그인 페이지로 이동한다.
- 4 제공된 정보 외 궁금한 사항을 등록한 캠핑장의 연락처로 직접 문의 할 수 있다. 등록된 연락처가 없다면 고객센터로 연결된다.

화면 설계

5

시설 정보

6



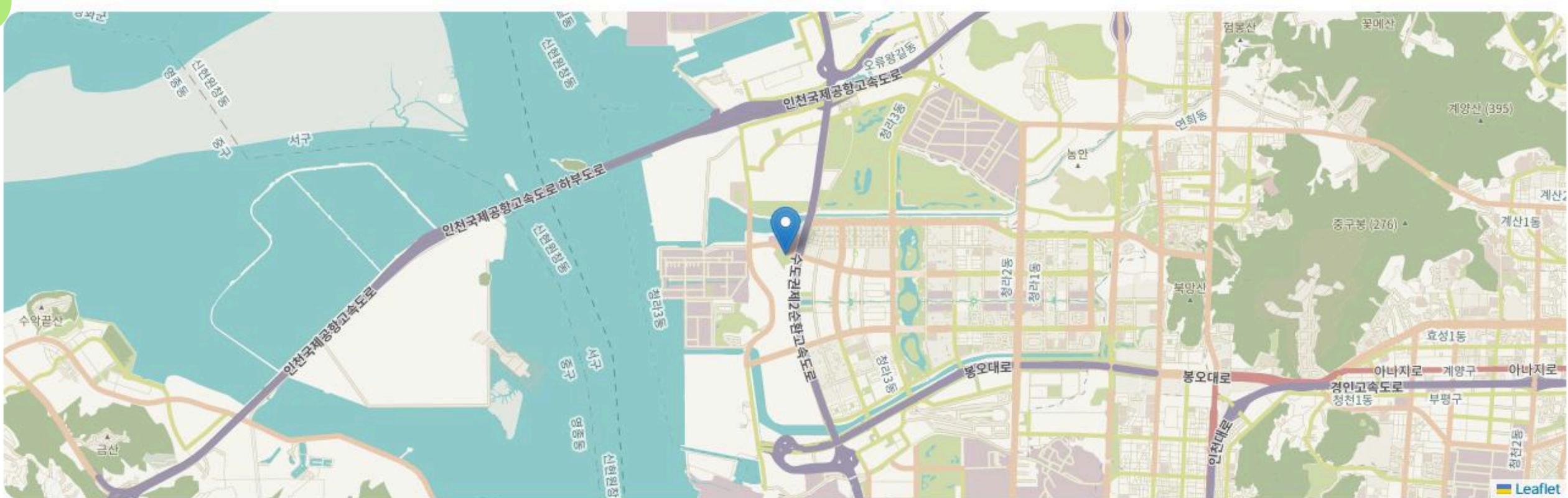
캠핑장 소개

7

청라국제도시의 해변공원 내에 위치한 도심 캠핑장 캠핑을 위해 이동하는 거리가 아까운 이들에게 최적인 도심지 캠핑장이다. 청라국제도시의 해변공원 내에 위치한 캠핑장으로 저렴한 가격과 일박 예약이 가능해서 당일 캠핑을 즐기기도 좋다. 캠핑장 옆으로 서해와 영종도가 바라 보이는 산책코스가 있는 노을공원이 있다. 도심 공원 특성상 캠핑장비와 준비물 없는 이들을 위해 캠핑장비와 취사시설등 일체를 대여해 주는 서비스도 마련되어 있다. 이외에도 바비큐 존, 텐트카라반존, 어린이 놀이시설 등을 갖추고 있다. 주요시설 : 바베큐존(14면) / 자동차야영장 사이트(38면) / 텐트카라반존(13면) 기타정보 : 개인 트레일러, 카라반 입장 가능(전장 규격 6M 이내로 제한, 견인차량 반드시 외부주차장 주차) / 반려동물 동반 불가능 카라반 내부시설 : 침대, 에어컨, WIFI (위치에 따라 불가능한 곳 있음), 난방기구 화로대 : 개인 화로 숯만 사용 가능(장작 사용 불가), 개인 장작 반입 금지

캠핑장 위치

8



5

썸네일과 버튼 하단은 두 가지 탭으로 구성되어 있다.

첫 번째 탭인 캠핑장 정보 탭

6

캠핑장 옵션을 아이콘을 활용해 시각적으로 표시함.

검색 시 필터링이 가능한 옵션이다.

7

캠핑장 소개 부분.

상세 정보 및 이용 팁이 포함되어 있다.

8

지도 API인 'Leaflet API'를 활용하여 캠핑장 위치를 지도에 표시.



설명

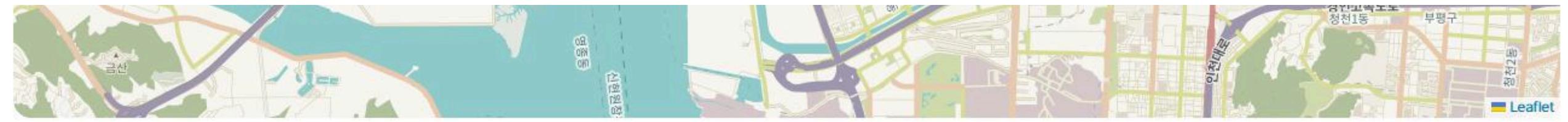
캠핑장 상세 정보를 조회하는 페이지

경로

헤더 > 캠핑장 검색 > 캠핑장 > 캠핑장 정보 탭(기본 값)

담당자

김성우



갤러리

9



시설명	사용기간	규격	비수기		비고
			평일	주말(공휴일)	
데트 캠핑 차갈	1동/1박	5x9	20,000	25,000	25,000 4인 기준 (7박2일)
오토 캠핑 차갈	1대/1박	5x8	30,000	40,000	40,000 4인 기준
오토 캠핑 데크	1대/1박	5x8	50,000	60,000	60,000 4인 기준
기린 박 차갈	1대/1박	6x15	60,000	70,000	70,000 4인 기준
트레일러	1대/1박	6x15	160,000	220,000	220,000 4인 기준

예약하기

상세 설명

9

캠핑장의 다양한 사진들을 제공하여 주변 경관과 캠핑장 전경을 방문 전에 미리 확인할 수 있다.



설명

캠핑장 상세 정보를 조회하는 페이지

경로

헤더 > 캠핑장 검색 > 캠핑장 > 캠핑장 리뷰 탭

담당자

김성우

화면 설계

10

캠핑장 후기

청라 캠핑장 솔직 후기
2025/03/11 18:05:16청라에도 이런 캠핑장이?
2025/03/11 18:04:50캠핑장 후기입니다😊
2025/03/11 18:04:15청라 캠핑장 솔직 후기
2025/03/11 18:03:10첫 캠핑 다녀왔어요~
2025/03/11 18:02:35청라캠핑파크 후기
2025/03/11 18:01:41

상세 설명

10

- ② 캠핑장 후기 버튼을 클릭하면 캠핑장 정보 탭에서 캠핑장 후기 목록 탭으로 전환 해당 캠핑장의 후기를 확인할 수 있다.



화면 설계

캠핑장 예약(1)

설명 캠핑장을 예약하는 페이지

경로 헤더 > 캠핑장 검색 > 캠핑장 > 예약하기

담당자 김성우

The screenshot shows a mobile application interface for camping reservations. On the left, there's a sidebar with sections like '캠핑라운지' (Camping Lounge), '시설 정보' (Facility Information), and '캠핑장 소개' (Camping Site Introduction). The main area features a large calendar for March 2025. A specific date, March 19, 2025, is highlighted in green and circled with number 7. Other dates are also circled with numbers 6, 8, and 9. A large orange button labeled '예약하기' (Reserve) is at the bottom right of the calendar. Callout numbers 1 through 8 are placed around the interface to point out specific UI elements.

- 1 캠핑장 예약 버튼을 클릭하면 예약창이 활성화 된다.
- 2 닫기 버튼
클릭하면 예약창이 비활성화 된다.
- 3 선택한 날짜
직접 수정이 불가능한 필드로 달력에서 날짜를 선택하면 자동으로 수정된다.
- 4 달력 영역
예약 날짜를 선택하는 달력
- 5 예약 월 선택
화살표로 월별 이동이 가능하다
- 6 현재 날짜
현재 날짜가 강조되어 표시된다
- 7 선택한 날짜
날짜를 클릭하면 현재 날짜보다 강조하여 표시된다. 해당 날짜가 입력 필드로 자동 입력된다.
- 8 예약 버튼
입력된 날짜로 예약을 진행하는 버튼.

상세 설명



화면 설계

캠핑장 예약(2)

설명 캠핑장을 예약하는 페이지

경로 헤더 > 캠핑장 검색 > 캠핑장 > 예약하기

담당자 김성우

화면 설계

상세 설명

- 1 토스 결제 API
예약 버튼 클릭 시 활성화 된다.
- 2 닫기 버튼
클릭하면 결제창이 비활성화 된다.
- 3 결제 방법 선택
원하는 결제 방법을 선택한다.
- 4 실제 결제가 이루어지지 않는 테스트용
API 임을 알린다.



설명

모든 리뷰를 조회하는 페이지

경로

헤더 > 리뷰 모아보기

담당자

최수민

화면 설계

캠핑라운지

캠핑장 검색 리뷰 모아보기 채팅 목록



1

👑 베스트 리뷰 👑

2



김천오토캠핑장 후기

김천실내테니스장 & 김천오토캠핑장
조회수: 59

3



그자리 야영장 후기

그자리 야영장
조회수: 51

고래마을 후기

호미곶 고래마을 캠핑장
조회수: 29

리뷰 전체 보기

그자리 야영장

글램인스타

청라캠핑파크

상세 설명

1

베스트 리뷰 영역

조회수가 높은 순으로 리뷰를 조회한다
해당 리뷰는 슬라이드로 표시된다

2

베스트 리뷰

조회수가 높은 순으로 표시된 리뷰
클릭하면 해당 리뷰로 이동한다

3

리뷰의 제목, 캠핑장 위치, 조회수



설명

모든 리뷰를 조회하는 페이지

경로

헤더 > 리뷰 모아보기

담당자

최수민

화면 설계

4

청라캠핑파크



청라에도 이런 캠핑장이?

조회수: 2

청라캠핑파크



캠핑장 후기입니다😊

조회수: 3

청라캠핑파크



청라 캠핑장 솔직 후기

조회수: 0

청라캠핑파크



첫 캠핑 다녀왔어요~

조회수: 3

청라캠핑파크



청라캠핑파크 후기

조회수: 2

봉즈살롱



봉즈살롱 리얼 후기

조회수: 3

5



상세 설명

4

전체 리뷰 영역
작성 순서가 최신순으로 표시된다
클릭하면 해당 리뷰로 이동

5

페이지 네비게이션
원하는 페이지로 이동하는 버튼



설명	이용한 캠핑장의 리뷰를 작성하는 페이지	경로	마이페이지 > 예약 내역 탭 > 후기 작성	담당자	최수민
----	-----------------------	----	-------------------------	-----	-----

화면 설계

The screenshot shows the 'Review Writing (1)' screen. At the top, there are tabs for '설명' (Description), '경로' (Path), '담당자' (Responsible Person), and '최수민' (Choi Soo-min). The main area contains a review form with the following elements:

- 등록 취소** (Registration Cancellation) button (1)
- 제목을 입력하세요** (Enter the title) input field (2)
- 캠핑장** (Camping Site) and **우니메이카 밀양점** (UNIMEAKI Milyang Branch) buttons (3)
- 위치** (Location) and **경남 밀양시 단장면 국전로 78-5** (Gyeongsangnam-do Milyang-si Dangjang-myeon Gyeongju-ro 78-5) buttons (3)
- 방문일** (Visit Date) and **2025. 3. 12.** (2025. 3. 12.) buttons (3)
- 만족도** (Satisfaction Level) section with radio buttons:
 - 매우 불만족해요
 - 불만족해요
 - 보통이에요
 - 만족해요
 - 완전 만족해요
- 사이트 크기** (Site Size) section with radio buttons:
 - 매우 좁아요
 - 좁아요
 - 충분해요
 - 넓어요
 - 완전 넓어요
- 청결도** (Cleanliness) section with radio buttons:
 - 더러워요
 - 지저분해요
 - 보통이에요
 - 깨끗해요
 - 완전 깨끗해요
- 친절함** (Friendliness) section with radio buttons:
 - 완전 불친절해요
 - 불친절해요
 - 보통이에요
 - 친절해요
 - 완전 친절해요

상세 설명

- 등록 취소버튼
취소 시 이전 페이지로 이동
- 리뷰 제목 작성 필드
- 캠핑장 정보 영역
후기를 작성할 캠핑장 정보가 자동으로 입력된다. Read only 인풋으로 수정이 불가능 하다.
- 만족도 체크 항목
라디오 버튼으로 만족도를 체크한다



설명 이용한 캠핑장의 리뷰를 작성하는 페이지

경로 마이페이지 > 예약 내역 탭 > 후기 작성

담당자 최수민

The screenshot shows a user interface for posting a review. At the top, there's a navigation bar with tabs for '설명' (Description), '경로' (Path), '담당자' (Responsible Person), and '최수민' (Choi Soo-min). Below the navigation is a large input area for the review content. Step 5 is highlighted with a green circle and points to a '파일 선택' (File Selection) button with the message '선택된 파일 없음' (No files selected). Step 6 is highlighted with a green circle and points to the main text input field with the placeholder '내용을 입력해주세요' (Please enter content). Step 7 is highlighted with a green circle and points to the top toolbar which includes font size (H1, H2, Sans Serif), bold (B), italic (I), underline (U), list (List), and link (A) buttons. Step 8 is highlighted with a green circle and points to a green '리뷰 등록' (Post Review) button at the bottom of the input area.

화면 설계

상세 설명

- 5 썸네일 등록 필드
썸네일로 등록할 이미지 파일을 업로드한다
- 6 React-Quill 연동 글쓰기 에디터
- 7 React-Quill 에디터 옵션
폰트 크기 선택, 폰트 스타일, 밑줄, 이미지 업로드 등이 가능하다
- 8 리뷰등록 버튼
입력한 내용으로 리뷰를 등록한다



설명

리뷰의 상세 내용을 확인하는 페이지

경로

헤더 > 리뷰 모아보기 > 리뷰

담당자

최수민

화면 설계

1 흥길동 2 수정하기 3 리뷰 삭제

4 2025/03/13 16:16:54
10 명이 이 글을 읽었어요!
1 명이 이 글을 좋아해요!

5 밀양에 캠핑 다녀왔어요~!

6

캠핑장 우니메이카 밀양점
위치 경남 밀양시 단장면 국전로 78-5
방문일 2025-03-12
만족도 완전 만족해요
사이트 크기 완전 넓어요
청결도 보통이에요
친절함 친절해요

상세 설명

- 리뷰를 작성한 회원의 이름이 표시되고, 클릭시 작성한 회원의 마이페이지로 이동한다.
- 리뷰 수정페이지로 이동하고 리뷰 내용을 수정할 수 있다. 작성자에게만 활성화 되는 버튼.
- 리뷰를 삭제할 수 있다. 작성자에게만 활성화 되는 버튼.
- 작성 날짜, 조회수, 좋아요 수가 표시되는 공간이고, 조회수는 베스트 리뷰 선정 기준이 된다. 조회 중인 리뷰를 좋아요 버튼을 통해 추천할 수 있다.
- 리뷰의 제목에 해당하는 영역
- 리뷰 작성자가 이용했던 캠핑장의 이름, 위치, 방문 날짜 및 작성자가 평가한 만족도가 표시된다.

7



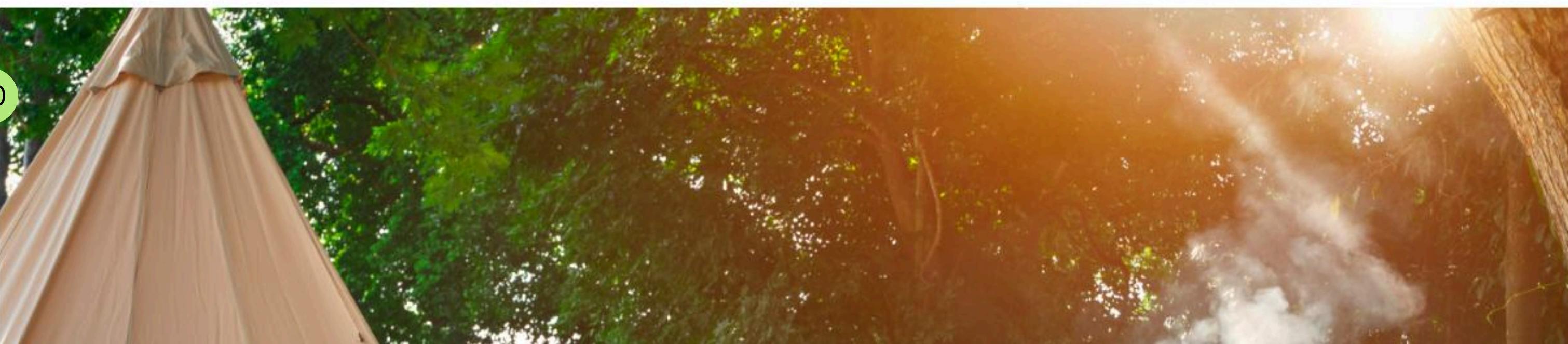
우니메이카 밀양점에 방문했습니다.

주인분도 친절하고 캠핑장도 깨끗하고 좋았습니다.

9

하지만 가는 길이 비포장 도로라 운전이 미숙하신 분이라면 조금 힘드실거 같네요.

10



7

리뷰 본문 영역

8

리뷰 썸네일

리뷰 등록시 업로드한 썸네일은 본문 최상단에 표시된다

9

리뷰 본문

React-Quill 에디터로 작성한 리뷰 내용

10

React-Quill 에디터로 업로드한 이미지

화면 설계

정말 즐거운 경험이었습니다.

우니메이카 밀양점 추천해요!

11



사진이 정말 멋지네요!

2025/03/13 16:24:40

똘이아빠

12



근처에 편의점이 있나요?

2025/03/13 16:23:17

캠핑러버



정말 멋진 캠핑장입니다! 추천해요

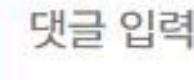
2025/03/13 16:21:43

홍길동

댓글 삭제

13

14



댓글 입력

15

댓글 등록

16

목록으로

11

회원들이 작성한 댓글이 표시되는 영역

12

각각의 회원의 프로필 이미지와 이름이 표시되고, 회원의 이름을 클릭하면 해당 회원의 마이페이지로 이동한다.

13

댓글을 삭제하는 버튼.
해당 댓글을 작성한 회원에게만 댓글 삭제 버튼이 표시된다.

14

댓글을 작성할 수 있는 공간. 텍스트로만 작성이 가능하다.

15

작성한 댓글을 등록하는 버튼.

16

리뷰 목록 페이지로 이동한다.



설명

자신이 작성한 리뷰를 수정하는 페이지

경로

헤더 > 리뷰 모아보기 > 리뷰 > 수정하기

담당자

최수민

화면 설계

1

수정 취소

2

밀양에 캠핑 다녀왔어요~!

3

캠핑장

우니메이카 밀양점

위치

경남 밀양시 단장면 국전로 78-5

방문일

2025. 3. 12.

만족도

 매우 불만족해요 불만족해요 보통이에요 만족해요 완전 만족해요

4

사이트 크기

 매우 좁아요 좁아요 충분해요 넓어요 완전 넓어요

청결도

 더러워요 지저분해요 보통이에요 깨끗해요 완전 깨끗해요

친절함

 완전 불친절해요 불친절해요 보통이에요 친절해요 완전 친절해요

상세 설명

1

수정 취소버튼

취소 시 이전 페이지로 이동

2

리뷰 제목 수정 필드

리뷰 제목이 자동으로 작성되어 있다

3

캠핑장 정보 영역

후기를 작성할 캠핑장 정보가 자동으로 입력된다. Read only 인풋으로 수정이 불가능 하다.

4

만족도 체크 항목

라디오 버튼으로 만족도를 체크한다



설명 자신이 작성한 리뷰를 수정하는 페이지

경로 헤더 > 리뷰 모아보기 > 리뷰 > 수정하기

담당자 최수민

화면 설계

5 파일 선택 선택된 파일 없음

6

H₁ H₂ Sans Serif B U A

우니메이카 밀양점에 방문했습니다.
주인분도 친절하고 캠핑장도 깨끗하고 좋았습니다.
하지만 가는 길이 비포장 도로라 운전이 미숙하신 분이라면 조금 힘드실거 같네요.

7

리뷰 수정

상세 설명

- 5 리뷰 썸네일 수정 필드
현재 썸네일을 볼 수 있고 파일을 선택해 교체할 수 있다
- 6 React-Quill 연동 글쓰기 에디터
현재 리뷰 내용이 자동으로 입력되어있다
- 7 리뷰 수정 버튼
입력 내용으로 리뷰를 수정한다



설명	모든 회원, 게시물, 예약, 댓글을 관리하는 페이지	경로	어드민 페이지	담당자	김성우
----	------------------------------	----	---------	-----	-----

캠핑라운지 관리자 페이지

관리 메뉴

- 대시보드**
- 회원 관리
- 캠핑장 관리
- 예약 관리
- 리뷰 관리
- 댓글 관리

전체 현황

전체 회원	15 명	등록된 캠핑장	212 개	진행된 예약	12 건
등록된 리뷰	3 개	등록된 댓글	3 개	전체 채팅 수	6 개

관리자 정보

이메일	admin@admin.com
이름	어드민 01
접속 시간	00:06:50

로그아웃

신규 회원

순번	이메일	이름	성별	전화번호	권한	상태	수정	삭제
15	bgfj656@example.com	문지수	여자	01043085712	USER	활성	수정	삭제
14	asdzc789@example.com	서준혁	남자	01098754021	USER	활성	수정	삭제
13	bangbang456@example.com	신예린	여자	01061342895	USER	비활성	수정	삭제
12	ksoung1402@example.com	한승호	남자	01027513968	USER	활성	수정	삭제
11	nkke99@example.com	유다은	남자	01084927560	USER	활성	수정	삭제
10	bvjiy544@example.com	오태양	남자	01036258714	USER	비활성	수정	삭제
9	kaksadk23@example.com	강서현	여자	01079031482	USER	활성	수정	삭제
8	yangtang96@example.com	정우진	남자	01052198643	USER	활성	수정	삭제

상세 설명

- 1 어드민 페이지용 헤더.
어드민 페이지에 접속하면 이를 인지하고 헤더가 어드민용으로 바뀜.
- 2 관리 메뉴 선택 버튼.
원하는 관리메뉴를 선택하는 버튼.
- 3 관리자 정보창.
현재 접속한 관리자의 이메일, 이름, 접속 시간을 확인할 수 있는 정보창.
- 4 로그아웃 버튼
현재 접속한 관리자를 로그아웃 하는 버튼.
- 5 대시보드 현황판
전체 회원 수, 캠핑장 수, 예약 수, 리뷰 수, 댓글 수, 채팅 수를 한눈에 확인할 수 있음.
- 6 대시보드 신규회원 테이블
최근 가입한 순으로 8명까지 회원을 조회 할 수 있는 테이블
- 7 수정, 삭제 버튼
해당 맴버를 수정, 삭제하는 버튼



설명 회원 정보를 조회, 수정, 삭제하는 페이지

경로 어드민 페이지 > 회원관리

담당자 김성우

The screenshot shows the '회원 관리' (Member Management) section of the Admin page. It includes a sidebar with '관리 메뉴' (Management Menu) containing links for Dashboard, Member Management (highlighted), Camping Site Management, Reservation Management, Review Management, and Tag Management. Below this is '관리자 정보' (Administrator Information) with fields for Email (admin@admin.com), Name (어드민 01), and Connection Time (00:17:03). A '로그아웃' (Logout) button is also present. The main area displays '회원 관리' (Member Management) with three summary boxes: '전체 회원' (15 명), '비활성화된 회원' (2 명), and '남여 성비' (61 : 38). Below these are filter buttons for sorting by 번호 순, 이메일 순, 이름 순, 성별 순, and 상태 순. The main content is a table listing 15 members with columns for 순번, 이메일, 이름, 성별, 전화번호, 권한, 상태, and buttons for 수정 and 삭제. The table includes dropdowns for gender and status, and buttons for 저장 and 취소. At the bottom are navigation buttons for page 1, 2, 3, etc., and a '선택 삭제' (Select Delete) button.

상세 설명

- 1 회원 관리 현황판 전체 회원 수, 비활성 회원, 성비를 확인
- 2 필터 버튼 오름/내림 차순 정렬, 필터 순 정렬 기능
- 3 전체 선택 버튼/ 선택삭제 버튼 모든 레이블을 선택하는 버튼과 선택한 레이블을 일괄 삭제하는 버튼
- 4 회원 관리 테이블 모든 회원을 조회하는 테이블. 한 페이지에 최대 15명 조회 가능
- 5 수정 / 삭제버튼 해당 회원 정보를 수정/ 삭제하는 버튼
- 6 회원 수정 활성화 수정 버튼 클릭 시 회원 수정 기능 활성화
- 7 페이지 네비게이션 원하는 페이지로 이동하는 버튼



설명 캠핑장을 등록(직접/API), 수정, 삭제하는 페이지

경로 어드민 페이지 > 캠핑장 관리

담당자 김성우

번호	캠핑장 이미지	캠핑장 이름	위치	조회수	좋아요	수정	삭제
212		하늘빛	인천시 옹진군	2	1	<button>수정</button>	<button>삭제</button>
211		우니메이카 밀양점	경상남도 밀양시	3	1	<button>수정</button>	<button>삭제</button>
209		우니메이카 안동점	경상북도 안동시	2	0	<button>수정</button>	<button>삭제</button>
206		우니메이카 당진점	충청남도 당진시	0	0	<button>수정</button>	<button>삭제</button>
205		김천맑은계곡 오토캠핑장	경상북도 김천시	0	0	<button>수정</button>	<button>삭제</button>
200		명덕농원	경기도 포천시	0	0	<button>수정</button>	<button>삭제</button>
199		밀양 도깨비방망이 캠핑장	경상남도 밀양시	0	0	<button>수정</button>	<button>삭제</button>
194		호두마루글램핑	경상남도 사천시	0	0	<button>수정</button>	<button>삭제</button>
192		마을야영장, 침산추월	세종시 세종시	2	0	<button>수정</button>	<button>삭제</button>

관리 메뉴

- 대시보드
- 회원 관리
- 캠핑장 관리**
- 예약 관리
- 리뷰 관리
- 댓글 관리

관리자 정보

이메일 admin@admin.com
이름 어드민 01
접속 시간 00:10:34

로그아웃

캠핑장 관리

등록된 캠핑장 212 개

1 등록된 캠핑장 (등록 버튼)
2 캠핑장 등록 (신규 등록 버튼)
3 고캠핑 API 연결 (연결 버튼)
4 정렬 및 필터 버튼
5 수정 / 삭제 버튼
6 캠핑장 검색 창
7 페이지 네비게이션

상세 설명

- 1 캠핑장 관리 현황판**
캠핑장 수 조회, 신규등록, API연결 기능
- 2 캠핑장 등록 버튼**
버튼 클릭시 캠핑장 등록 창 활성화
- 3 고캠핑 API 연결 버튼**
고캠핑 API에 연결하여 입력한 수 만큼의 캠핑장 정보를 받아옴
- 4 정렬 및 필터**
원하는 정렬 순서를 선택하는 버튼과 해당 옵션을 포함한 캠핑장을 조회하는 버튼
- 5 수정 / 삭제버튼**
해당 캠핑장 정보를 수정/ 삭제하는 버튼
수정 버튼 클릭 시 캠핑장 수정 창 활성화
- 6 캠핑장 검색 창**
입력한 정보를 포함한 캠핑장을 검색
- 7 페이지 네비게이션**
원하는 페이지로 이동하는 버튼



설명 캠핑장을 직접 등록하는 페이지

경로 어드민 페이지 > 캠핑장 관리 > 캠핑장 등록

담당자 김성우

1 캠핑장 등록 관리자 페이지

2 관리 메뉴 대시보드 회원 관리 캠핑장 관리 예약 관리 리뷰 관리 댓글 관리

3 캠핑장 등록 캠핑장 이름 캠핑장 설명 연락처 010 - 주소 도 시/군/구 주소 주소 1 주소 2 기타 x 좌표 : y 좌표 | 0 시설 정보 전기 화장실 온수 정작 판매 와이파이 운동 시설 동물 출입 물놀이 썸네일 파일 선택 선택된 파일 없음 캠핑장 사진 파일 선택 선택된 파일 없음 등록

4 캠핑장 등록 창 캠핑장 등록 버튼 클릭 시 활성화

5 등록 버튼 입력한 신규 캠핑장의 정보를 비동기 통신으로 백엔드에 전달하는 버튼

상세 설명

- 페이지 비활성화 캠핑장 등록 버튼 클릭 시 등록 창 외의 기능 비활성화. 어두운 영역 클릭 시 등록 취소
- 캠핑장 등록 창 캠핑장 등록 버튼 클릭 시 활성화
- 닫기 버튼 캠핑장 등록을 취소하고 등록 창을 비활성화 하는 버튼
- 캠핑장 정보 입력 필드 등록할 캠핑장의 정보를 입력하는 필드 이름, 설명, 연락처, 주소, 좌표, 수용인원, 옵션, 썸네일 이미지, 상세 이미지를 등록
- 등록 버튼 입력한 신규 캠핑장의 정보를 비동기 통신으로 백엔드에 전달하는 버튼



설명 캠핑장 정보를 수정하는 페이지

경로 어드민 페이지 > 캠핑장 관리 > 캠핑장 수정

담당자 김성우

1 관리 대시보드 회원 관리 캠핑장 관리 예약 관리 리뷰 관리 댓글 관리

2 캠핑장 설정 캠핑장 이름 우니메이카 안동점 캠핑장 설명 우니메이카 안동점은 경북 안동시 남후면 무릉리에 자리 잡았다. 안동시청을 기점으로 10km가량 떨어졌다. 자동차를 타고 경북대로와 무릉길을 번갈아 달리면 된다는 도착까지 걸리는 시간은 15분 안팎이다. 캠핑장에는 파쇄석으로 이뤄진 오토캠핑사이트 10면이 마련돼 있다. 사이트 크기는 가로 5m 세로 10m로 넓찍하다. 캠퍼는 개별 화장실과 샤워실을 이용할 수 있어 편리하다. 카라반 5대도 함께 운영 중이다. 내부에는 침대, 냉난방시설, 취사도구 등 일상생활이 가능할 정도의 시설이 갖춰져 있다. 주변에 안동민속촌과 안동문화관광단지가 있어 연계 여행에 나서기 수월하다.

3 고캠핑 API 연결 가져올 캠핑장 수 API 요청

온수 와이파이 운동시설 반려동물 수영장 조회수 좋아요 수정 삭제

조회수	좋아요	수정	삭제
2	1	수정	삭제
3	1	수정	삭제
1	0	수정	삭제
2	0	수정	삭제
12	4	수정	삭제
6	1	수정	삭제
0	0	수정	삭제
0	0	수정	삭제

4 캠핑장 수정 창 캠핑장 수정 버튼 클릭 시 활성화

5 단기 버튼 캠핑장 수정을 취소하고 수정 창을 비활성화 하는 버튼

6 캠핑장 정보 수정 필드 수정할 캠핑장 정보를 입력하는 필드 기본 값으로 선택한 캠핑장의 정보가 입력되어있음.

7 등록 버튼 입력한 캠핑장의 수정 정보를 비동기 통신으로 백엔드에 전달하는 버튼

상세 설명

- 페이지 비활성화 캠핑장 수정 버튼 클릭 시 수정 창 외의 기능 비활성화. 어두운 영역 클릭 시 수정 취소
- 캠핑장 수정 창 캠핑장 수정 버튼 클릭 시 활성화
- 닫기 버튼 캠핑장 수정을 취소하고 수정 창을 비활성화 하는 버튼
- 캠핑장 정보 수정 필드 수정할 캠핑장 정보를 입력하는 필드 기본 값으로 선택한 캠핑장의 정보가 입력되어있음.
- 등록 버튼 입력한 캠핑장의 수정 정보를 비동기 통신으로 백엔드에 전달하는 버튼



화면 설계

어드민 페이지 - 예약 관리

설명	예약을 조회, 삭제하는 페이지	경로	어드민 페이지 > 예약 관리	담당자	김성우
----	------------------	----	-----------------	-----	-----

캠핑라운지 관리자 페이지

관리 메뉴

- 대시보드
- 회원 관리
- 캠핑장 관리
- 예약 관리**
- 리뷰 관리
- 댓글 관리

예약 관리

1 진행된 예약
12 건
2 등록된 리뷰
3 개
3 등록된 댓글
3 개

▼
번호 순
회원 순
캠핑장 순
예약일 순
사용일 순

번호	회원 번호	회원 이메일	캠핑장 번호	캠핑장 이름	예약일	사용일	삭제
12	9	kaksadk23@example.com	133	청라캠핑파크	2025. 3. 11.	2025. 4. 10.	<button>삭제</button>
11	9	kaksadk23@example.com	209	우니메이카 안동점	2025. 3. 11.	2025. 6. 19.	<button>삭제</button>
10	9	kaksadk23@example.com	192	마을야영장, 침산추월	2025. 3. 11.	2025. 3. 26.	<button>삭제</button>
9	9	kaksadk23@example.com	193	경상북도교육청김천오토캠핑장	2025. 3. 11.	2025. 3. 21.	<button>삭제</button>
8	9	kaksadk23@example.com	193	경상북도교육청김천오토캠핑장	2025. 3. 11.	2025. 3. 26.	<button>삭제</button>
7	2	example@test.com	204	김천실내테니스장 & 김천오토캠핑장	2025. 3. 11.	2025. 3. 27.	<button>삭제</button>
6	2	example@test.com	207	호미곶 고래마을 캠핑장	2025. 3. 11.	2025. 3. 27.	<button>삭제</button>
5	2	example@test.com	211	우니메이카 밀양점	2025. 3. 11.	2025. 3. 26.	<button>삭제</button>
4	1	ksoung140w@gmail.com	188	글램인스타	2025. 3. 11.	2025. 6. 12.	<button>삭제</button>
3	1	ksoung140w@gmail.com	188	글램인스타	2025. 3. 11.	2025. 3. 31.	<button>삭제</button>
2	1	ksoung140w@gmail.com	208	그자리 야영장	2025. 3. 11.	2025. 3. 29.	<button>삭제</button>
1	1	ksoung140w@gmail.com	212	하늘빛	2025. 3. 11.	2025. 3. 19.	<button>삭제</button>

선택 삭제

« < 1 2 > »

화면 설계

상세 설명

- 1 전체 예약 현황판
전체 예약 수를 확인 가능
클릭 시 예약 관리 창으로 전환
- 2 전체 리뷰 현황판
전체 리뷰 수를 확인 가능
클릭 시 리뷰 관리 창으로 전환
- 3 전체 댓글 현황판
전체 댓글 수를 확인 가능
클릭 시 댓글 관리 창으로 전환
- 4 정렬 버튼
오름/내림 차순, 해당 내용 차순으로 정렬
- 5 예약 관리 테이블
전체 예약 정보를 조회할 수 있는 테이블
- 6 삭제 버튼
해당 예약을 삭제하는 버튼
- 7 페이지 네비게이션
원하는 페이지로 이동하는 버튼

69



설명 리뷰 게시물을 조회, 삭제하는 페이지

경로 어드민 페이지 > 리뷰 관리

담당자 김성우

캠핑라운지 관리자 페이지

관리 메뉴

- 대시보드
- 회원 관리
- 캠핑장 관리
- 예약 관리
- 리뷰 관리**
- 댓글 관리

관리자 정보

이메일	admin@admin.com
이름	어드민 01
접속 시간	00:00:14

로그아웃

리뷰 관리

1. 진행된 예약 16 건
2. 등록된 리뷰 7 개
3. 등록된 댓글 7 개

4. 정렬 버튼 (번호 순, 제목 순, 회원 순, 캠핑장 순, 조회수 순, 작성일 순)
5. 리뷰 목록 테이블
6. 리뷰 상세 정보 및 삭제 버튼
7. 페이지 네비게이션

순번	제목	회원 번호	회원 이름	캠핑장 번호	캠핑장 이름	조회수	작성일	바로가기	삭제
7	봉즈살롱 리얼 후기	3	어드민 01	3	봉즈살롱	2	2025. 3. 11.	바로가기	삭제
6	별보러 가기 좋은 캠핑장!	3	어드민 01	17	더예감스테이	3	2025. 3. 11.	바로가기	삭제
5	화순동아일랜드 후기	11	유다은	178	화순동아일랜드	0	2025. 3. 11.	바로가기	삭제
4	첫 캠핑 다녀와봤어요~	15	문지수	25	고래불해수욕장(병곡지구) 야영장	5	2025. 3. 11.	바로가기	삭제
3	김천오토캠핑장 후기	2	홍길동	204	김천실내테니스장 & 김천오토캠핑장	34	2025. 3. 11.	바로가기	삭제
2	고래마을 후기	2	홍길동	207	호미곶 고래마을 캠핑장	29	2025. 3. 11.	바로가기	삭제
1	그자리 야영장 후기	1	성우	208	그자리 야영장	46	2025. 3. 11.	바로가기	삭제

선택 삭제

상세 설명

- 전체 예약 현황판
전체 예약 수를 확인 가능
클릭 시 예약 관리 창으로 전환
- 전체 리뷰 현황판
전체 리뷰 수를 확인 가능
클릭 시 리뷰 관리 창으로 전환
- 전체 댓글 현황판
전체 댓글 수를 확인 가능
클릭 시 댓글 관리 창으로 전환
- 정렬 버튼
오름/내림 차순, 해당 내용 차순으로 정렬
- 리뷰 관리 테이블
전체 리뷰 정보를 조회할 수 있는 테이블
- 바로가기/삭제 버튼
해당 리뷰로 이동하는 버튼
해당 리뷰를 삭제하는 버튼
- 페이지 네비게이션
원하는 페이지로 이동하는 버튼



설명 댓글을 조회, 삭제하는 페이지

경로 어드민 페이지 > 댓글 관리

담당자 김성우

캠핑라운지 관리자 페이지

관리 메뉴

- 대시보드
- 회원 관리
- 캠핑장 관리
- 예약 관리
- 리뷰 관리
- 댓글 관리**

관리자 정보

이메일	admin@admin.com
이름	어드민 01
접속 시간	00:00:53

로그아웃

댓글 관리

1. 진행된 예약 16 건
2. 등록된 리뷰 7 개
3. 등록된 댓글 7 개

4. 정렬 버튼 (번호 순, 리뷰 순, 회원 순, 작성일 순)
5. 리뷰 댓글 테이블
6. 삭제 버튼
7. 페이지 네비게이션

번호	회원 번호	회원 이름	리뷰 번호	리뷰 제목	댓글 내용	작성일	삭제
9	11	유다은	4	첫 캠핑 다녀와봤어요~	사진찍은 곳이 혹시 어디인지 여쭤봐도 괜찮을까요?	2025. 3. 11.	삭제
8	11	유다은	4	첫 캠핑 다녀와봤어요~	첫 캠핑이라니 정말 즐거워졌겠어요~	2025. 3. 11.	삭제
6	15	문지수	3	김천오토캠핑장 후기	비용은 얼마나 들었나요?	2025. 3. 11.	삭제
4	15	문지수	3	김천오토캠핑장 후기	야경이 참 예뻐요	2025. 3. 11.	삭제
3	1	성우	1	그자리 야영장 후기	추천합니다 🌟	2025. 3. 11.	삭제
2	3	어드민 01	1	그자리 야영장 후기	날씨는 어땠나요?	2025. 3. 11.	삭제
1	3	어드민 01	1	그자리 야영장 후기	정말 멋진 풍경이네요!	2025. 3. 11.	삭제

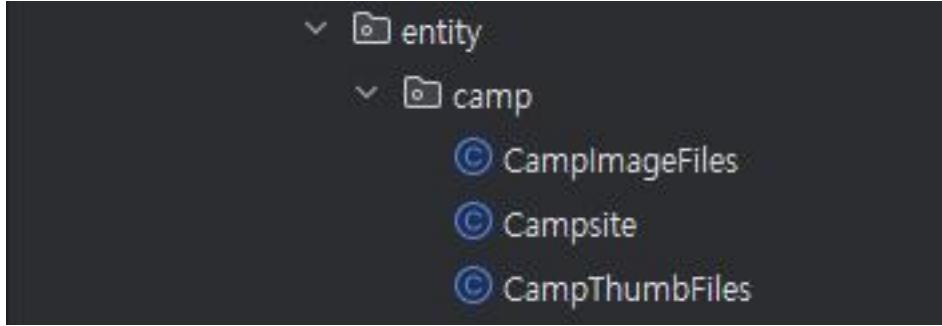
상세 설명

- 전체 예약 현황판
전체 예약 수를 확인 가능
클릭 시 예약 관리 창으로 전환
- 전체 리뷰 현황판
전체 리뷰 수를 확인 가능
클릭 시 리뷰 관리 창으로 전환
- 전체 댓글 현황판
전체 댓글 수를 확인 가능
클릭 시 댓글 관리 창으로 전환
- 정렬 버튼
오름/내림 차순, 해당 내용 차순으로 정렬
- 리뷰 댓글 테이블
전체 댓글 정보를 조회할 수 있는 테이블
- 삭제 버튼
해당 댓글을 삭제하는 버튼
- 페이지 네비게이션
원하는 페이지로 이동하는 버튼

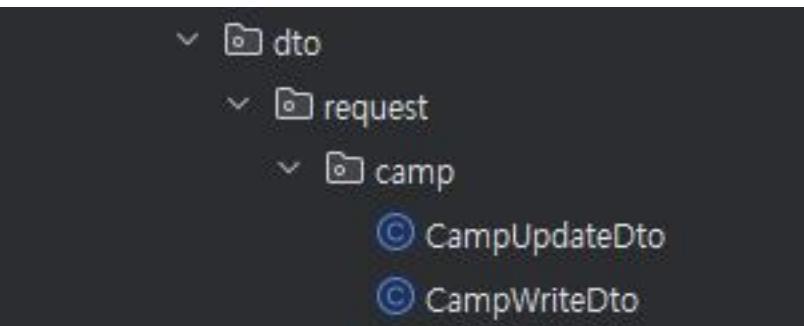
코드 리뷰

코드

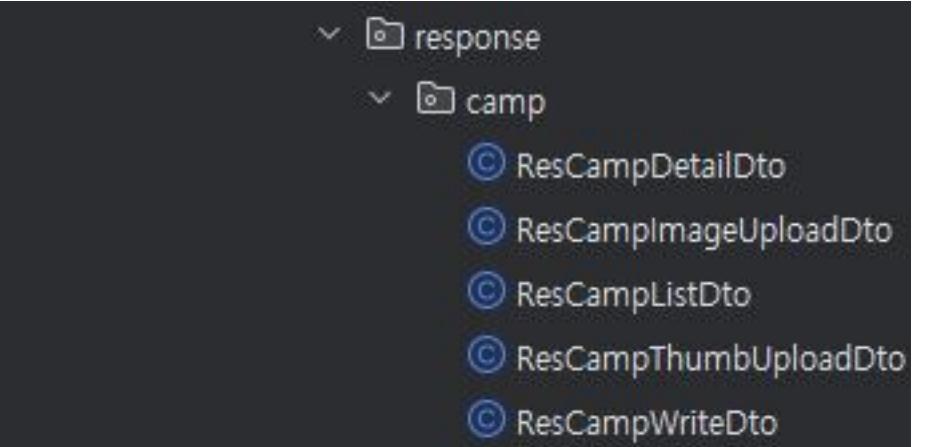
① Entity



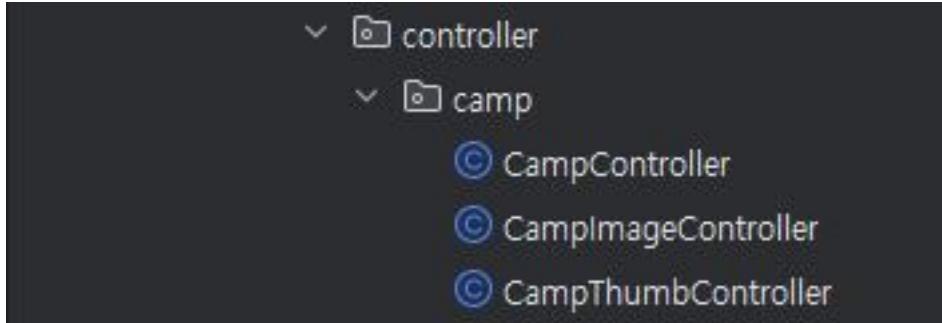
② DTO - Request



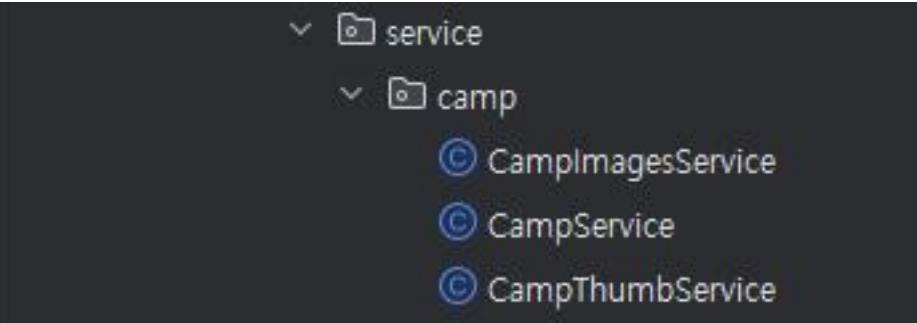
③ DTO - Response



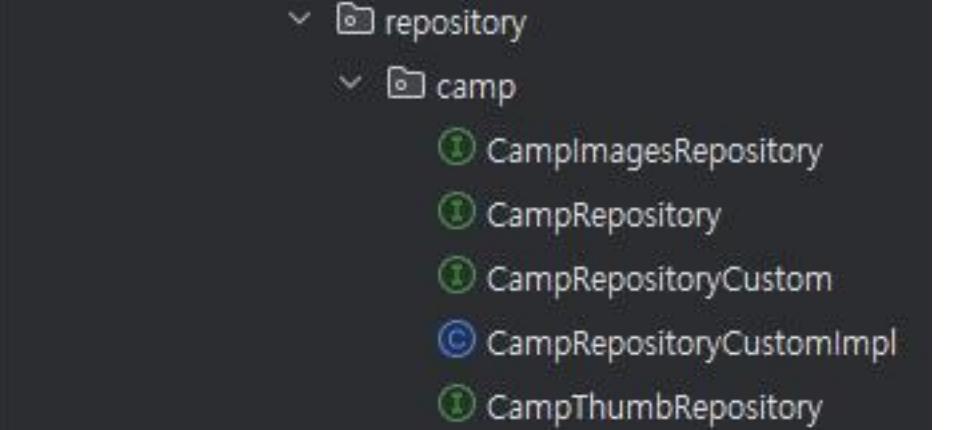
④ Controller



⑤ Service



⑥ Repository



설명

- 데이터베이스의 테이블과 매핑되는 Entity
- 프론트엔드에서 발송한 Request에 대응하는 DTO
- 프론트엔드에서 요청한 Response에 대응하는 DTO
- REST API 컨트롤러. 엔드포인트를 제공해 프론트엔드에서 호출할 수 있도록 함
- 컨트롤러와 리포지토리 사이에서 비즈니스 로직을 처리
- 데이터베이스와 상호작용

코드

CampList.js

```

const [searchTerm :string , setSearchTerm] = useState( initialState: "" );
const [selectedFilters :any[], setSelectedFilters] = useState( initialState: [] );
const filterList :string[] = ["전기", "화장실", "장작 판매", "온수", "와이파이", "운동시설", "반려동물", "수영장"];

5 const [changeSearch = (e) :void => setSearchTerm(e.target.value);

const toggleFilter = (filter) :void => { Show usages 🔍 seongwoo
4 setSelectedFilters( value: (prevFilters :any[]) => {
    const newFilters :any[] = prevFilters.includes(filter)
        ? prevFilters.filter((f) :boolean => f !== filter) // 선택 해제
        : [...prevFilters, filter]; // 추가 선택

    setPage( value: 1);
    return newFilters;
});};

<div className="search-filter">
    <ul className="filter">
        {filterList.map(function (filter :string ) {
            return (
                <li
                    className={selectedFilters.includes(filter) ? "item active" : "item"}
                    key={filter}
                    2 onClick={() :void => toggleFilter(filter)}
                >
                    #{{filter}}
                </li>
            );
        })
    </ul>
</div>
<div className="search-area mt_md">
    <input type="text" className="search-input" placeholder="캠핑장 검색"
        value={searchTerm}
        3 onChange={changeSearch}/>
    <input type="submit" value="" className="search-btn"/>
</div>

```

설명

- 1 검색어, 필터 리스트, 선택된 필터 선언
- 2 JSX에서 필터 버튼을 클릭하면 toggleFilter() 함수를 호출
- 3 JSX에서 검색어가 변경되면 changeSearch() 함수를 호출
- 4 해당 함수를 호출한 필터 내용을 파라미터로 받아 selectedFilter에 일치하는 내용이 있으면 선택을 해제하고 없다면 추가
- 5 변경된 검색어의 값을 searchTerm에 저장

코드

CampList.js

```
useEffect(() => {
  campSearch(searchTerm, page);
}, [selectedFilters, searchTerm, page]);
```



```
const campSearch = async (search, page) => {
  try {
    console.log("검색어:", searchTerm, "필터:", selectedFilters);
    const response: AxiosResponse<any> = await axios.get(`http://localhost:8080/camp/search`, config: {
      params: {
        search: searchTerm,
        filters: selectedFilters,
        page: page - 1,
        sort: "id,desc",
      },
      paramsSerializer: (params) => qs.stringify(params, {arrayFormat: "repeat"}),
    });

    console.log("[CampList.js] campSearch() success.");
    console.log(response.data);

    setCampList(response.data.content);
    setTotalCnt(response.data.totalElements); // 수정된 부분

  } catch (error) {
    console.log("[CampList.js] campSearch() error.");
    console.log(error.response?.data);
    console.log(error);
  }
};
```

설명

- ① useEffect 흑으로 선택된 필터나 검색어가 달라질때마다 campSearch() 함수 호출
- ② AXIOS 비동기 통신으로 해당 엔드포인트에 GET 매서드 요청
- ③ 파라미터로 검색어와 선택된 필터 전달
- ④ AXIOS 쿼리 파라미터를 문자열로 변환
*변환 이유는 다음 장에 추가 설명
- ⑤ 백엔드에서 받아온 Response의 데이터를 campList에 저장하여 활용



코드

Console

The screenshot shows the Chrome DevTools Console tab with the following log entries:

- React Router Future Flag Warning: React Router will begin wrapping state updates in 'React.startTransition' in v7. You can use the 'v7_startTransition' future flag to opt-in early. For more information, see https://reactrouter.com/v6/upgrading/future#v7_starttransition.
- React Router Future Flag Warning: Relative route resolution within Splat routes is changing in v7. You can use the 'v7_relativeSplatPath' future flag to opt-in early. For more information, see https://reactrouter.com/v6/upgrading/future#v7_relativesplatpath.
- [CampList.js] campSearch() success.
- [CampList.js] getCampList() success.
- Error while trying to use the following icon from the Manifest: <http://localhost:3000/logo192.png> (Download error or resource isn't a valid image)
- Access to XMLHttpRequest at '[http://localhost:8080/camp/search?search=&filters\[\]=%ED%99%94%EC%9E%A5%EC%8B%A4&page=0](http://localhost:8080/camp/search?search=&filters[]=%ED%99%94%EC%9E%A5%EC%8B%A4&page=0)' from origin '<http://localhost:3000>' has been blocked by CORS policy: camp:1 No 'Access-Control-Allow-Origin' header is present on the requested resource.
- [CampList.js] campSearch() error.
- undefined
- AxiosError {message: 'Network Error', name: 'AxiosError', code: 'ERR_NETWORK', config: {...}, request: XMLHttpRequest, ...}
- GET [http://localhost:8080/camp/search?search=&filters\[\]=%ED%99%94%EC%9E%A5%EC%8B%A4&page=0](http://localhost:8080/camp/search?search=&filters[]=%ED%99%94%EC%9E%A5%EC%8B%A4&page=0) net::ERR_FAILED 400 (Bad Request)
- Access to XMLHttpRequest at '[http://localhost:8080/camp/search?search=&filters\[\]=%ED%99%94%EC%9E%A5%EC%8B%A4&filters\[\]=%EC%98%A8%EC%98%98&page=0](http://localhost:8080/camp/search?search=&filters[]=%ED%99%94%EC%9E%A5%EC%8B%A4&filters[]=%EC%98%A8%EC%98%98&page=0)' from origin '<http://localhost:3000>' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.
- [CampList.js] campSearch() error.
- undefined
- AxiosError {message: 'Network Error', name: 'AxiosError', code: 'ERR_NETWORK', config: {...}, request: XMLHttpRequest, ...}
- GET [http://localhost:8080/camp/search?search=&filters\[\]=%ED%99%94%EC%9E%A5%EC%8B%A4&filters\[\]=%EC%98%A8%EC%98%98&page=0](http://localhost:8080/camp/search?search=&filters[]=%ED%99%94%EC%9E%A5%EC%8B%A4&filters[]=%EC%98%A8%EC%98%98&page=0) net::ERR_FAILED 400 (Bad Request)

1

설명

1 paramsSerializer 사용 이유

프론트엔드에서 발송하는 데이터와 후술할 백엔드의 코드에서 받아야 할 데이터의 형식이 일치한다고 여겨졌음에도 'Bad Request' 오류가 발생

console.log로 selectedFilter의 내용을 확인했을 땐 의도한대로 State가 저장됨을 확인

그러나 요청 URL에서 이를 하나의 Array로 인식하지 못하고 개별 파라미터로 발송하고 있음을 확인

문제 파악 후 paramsSerializer: (params) → qs.stringify(params, {arrayFormat: "repeat"}) 구문으로 해당 파라미터를 배열의 형태로 데이터를 직렬화하여 발송

코드

CampController.java

```

1 //서치, 필터
@GetMapping("/camp/search") ▲ seongwoo
public ResponseEntity<Page<ResCampListDto>> searchCamp(
    @RequestParam(required = false) String search,
    @RequestParam(required = false) List<String> filters,
    @PageableDefault(size = 9, sort = "id", direction = Sort.Direction.DESC) Pageable pageable){
    Page<ResCampListDto> listDto; 3 campService.getFilteredCamps(search, filters, pageable);
    return ResponseEntity.status(HttpStatus.OK).body(listDto);
}

```

CampService.java

```

//검색 필터 기능
public Page<ResCampListDto> getFilteredCamps(String search, List<String> filters, Pageable pageable) { 1 usage ▲ seongwoo
    4 return campRepository.findFilteredCamps(search, filters, pageable);
}

```

CampRepository

```

public interface CampRepository extends JpaRepository<Campsite, Long>, CampRepositoryCustom {
    5
}

```

CampRepositoryCustom.java

```

public interface CampRepositoryCustom { 2 usages 2 implementations ▲ seongwoo
    Page<ResCampListDto> findFilteredCamps(String search, List<String> filters, Pageable pageable);
}

```

설명

- ① 프론트에서 보낸 요청을 컨트롤러에서 수신
- ② 파라미터로 발송한 검색어, 필터를 각각 String, List<String>으로 받음
- ③ 검색, 필터링 기능을 활용하기 위해 서비스 레이어로 파라미터를 전달
- ④ 서비스 레이어에서 리포지토리 레이어로 전달
- ⑤ 일반적인 JPA Repository로 처리하기 어렵고, 코드의 리펙토링을 위해 CampRepositoryCustom 인터페이스를 따로 생성하고 상속
- ⑥ CampRepositoryCustom에 findFilterdCamps 매서드를 선언



코드

CampRepositoryCustomImpl.java

```
@Repository * seongwoo
@RequiredArgsConstructor
public class CampRepositoryCustomImpl implements CampRepositoryCustom {
    private final JPAQueryFactory queryFactory;

    @Override 1 usage * seongwoo
    public Page<ResCampListDto> findFilteredCamps(String search, List<String> filters, Pageable pageable) {
        QCampsite camp = QCampsite.campsite;
        BooleanBuilder builder = new BooleanBuilder();

        //검색기능
        if (search != null && !search.isBlank()) {
            BooleanBuilder searchCondition = new BooleanBuilder();
            searchCondition.or(camp.campName.containsIgnoreCase(search));
            searchCondition.or(camp.campAddressDo.containsIgnoreCase(search));
            searchCondition.or(camp.campAddressGungu.containsIgnoreCase(search));

            builder.and(searchCondition);
        }

        //필터 기능
        if (filters != null && !filters.isEmpty()){
            for(String filter : filters){
                switch (filter){
                    case "화장실": builder.and(camp.toilet.gt( right: 0)); break;
                    case "온수": builder.and(camp.hotWater.eq( right: true)); break;
                    case "전기": builder.and(camp.electric.eq( right: true)); break;
                    case "장작 판매": builder.and(camp.fireWood.eq( right: true)); break;
                    case "와이파이": builder.and(camp.wifi.eq( right: true)); break;
                    case "운동시설": builder.and(camp.playGround.eq( right: true)); break;
                    case "반려동물": builder.and(camp.pet.eq( right: true)); break;
                    case "물놀이": builder.and(camp.swimming.eq( right: true)); break;
                }
            }
        }
    }
}
```

설명

- 1 CampRepositoryCustom 인터페이스를 구현
- 2 동적 쿼리 구현을 위해 QueryDSL 라이브러리를 의존성 추가
JPA의 Entity를 QueryDSL로 제어하기 위해 JAPQueryFactory 객체를 선언
- *QueryDSL 사용 이유
기준에 배운 방식인 JPQL로 쿼리를 작성하게 되면 검색어에 포함될 데이터의 종류 만큼의 쿼리문과, 8개의 필터 조합의 모든 경우의 수 만큼의 쿼리문을 따로 작성해야하는 문제가 있었다.
따라서 동적쿼리를 구현하기위해 QueryDSL 을 따로 공부하여 적용해보았다.
- 3 findFilteredCamps 매서드를 오버라이드
- 4 QueryDSL의 QEntity 를 활용해 camp 객체 선언
- 5 동적 where절을 만들기 위한 BooleanBuilder
- 6 검색어가 비어있지 않을시 캠핑장 이름, 주소 가 검색어에 포함되어있는지 확인 후 where 절에 추가
- 7 selectedFilter를 switch문으로 반복하여 해당필터에 대응하는 필드가 true인 camp를 선별 후 where절에 추가

코드

CampRepositoryCustomImpl.java

```
//페이지
1 OrderSpecifier<Long> orderById = camp.id.desc();
List<Campsite> campList = queryFactory
    .select(camp)
    .where(builder)
    .orderBy(orderById)
    .offset(pageable.getOffset())
    .limit(pageable.getPageSize())
    .from(camp)
    .fetch();

long total = queryFactory
    .selectFrom(camp)
    .where(builder)
    .fetchCount();

//DTO 전환
2 List<ResCampListDto> dtoList = campList.stream() Stream<Campsite>
    .map(c -> ResCampListDto.builder()
        .id(c.getId())
        .campName(c.getCampName())
        .campInfo(c.getCampInfo())
        .campAddressDo(c.getCampAddressDo())
        .campAddressGungu(c.getCampAddressGungu())
        .thumb(c.getThumb().stream() Stream<CampThumbFiles>
            .map(ResCampThumbUploadDto::fromEntity) Stream<ResCampThumbUploadDto>
            .collect(Collectors.toList()))
        .build()) Stream<ResCampListDto>
    .collect(Collectors.toList());

3 return new PageImpl<>(dtoList, pageable, total);
}
```

설명

- 1 QueryFactory로 쿼리문 생성
where절에 필터링한 BooleanBuilder 매핑
Id 역순으로 정렬한 데이터를 Page<>로 반환하기 위해 List<Campsite> 엔티티 배열에 반환
전체 캠프 페이지 수를 Long타입으로 반환하기 위해 .fetchCount() 함수로 카운트
- 2 엔티티로 반환한 데이터를 프론트로 발송하기 위해 DTO로 전환
- 3 전환한 DTO를 Page객체에 파라미터로 전송
DTO와 페이지 정보를 담은 객체를 반환

코드

① 한국관광공사_고캠핑 정보 조회서비스_GW

https://www.data.go.kr/data/15101933/openapi.do#/tab_layer_recommend_data

AdminCampGetApi.js

```
const [rowNum : number , setRowNum] = useState( initialState: 0);
const getCampApi = async (rowNum) : Promise<void> =>{ Show usages  ↗ seongwoo
    try {
        2  const response : AxiosResponse<any> = await axios.get( url: `http://localhost:8080/admin/camp/openapi/${rowNum}` );
        console.log("[AdminCampGetApi.js] getCampApi() success.");
        alert(`캠핑장 ${rowNum}개를 가져왔습니다.`);
    }catch (error){
        console.log("[[AdminCampGetApi.js] getCampApi() error.]");
        console.error("에러 메시지:", error.response?.data || error.message);
    }
}
```

CampController.java

```
@GetMapping(value = @"/admin/camp/openapi/{rowNum}") ↗ seongwoo
public String CampDataFromOpenApi(@PathVariable Long rowNum) {
    3 campService.CampDataFromOpenApi(rowNum);
    return "고캠핑 공공 API 연동에 성공했습니다.";
}
```

설명

① 공공데이터포털(data.go.kr)에서 제공하는 “한국관광공사_고캠핑 정보 조회서비스_GW” Open API를 연결해 캠핑장 정보 활용

② 프론트엔드에서 요청할 캠핑장 수를 파라미터로 백엔드에 Get 요청

③ 컨트롤러에서 요청 수신 후 서비스 단에 전달

코드

CampService.java

```
public void CampDataFromOpenApi(Long rowNum){ 1 usage  ▲ seongwoo
----- 캠핑장, 썸네일 가져오기
    //api 연결
    1 //서비스키를 인코딩하는 과정에서 계속 인코딩 오류가 생겨 Uri빌더 활용
    //참고 https://life.photogrammer.me/openapi-servicekey-encoding-problem/
    System.out.println();
    2 final var builder = new DefaultUriBuilderFactory();
    builder.setEncodingMode(DefaultUriBuilderFactory.EncodingMode.NONE);
    final String uriString = builder.builder() UriBuilder
        .scheme("http")
        .host("apis.data.go.kr")
        .path("B551011/GoCamping/basedList")
        .queryParam( name: "serviceKey", serviceKey)
        .queryParam( name: "numOfRows", rowNum)
        .queryParam( name: "pageNo", ...values: 1)
        .queryParam( name: "MobileOS", ...values: "ETC")
        .queryParam( name: "MobileApp", ...values: "CampingLounge")
        .queryParam( name: "_type", ...values: "json")
        .build() URI
        .toString();
    3 final URI uri = URI.create(uriString);

    try {
        RestTemplate restTemplate = new RestTemplate();
        ResponseEntity<String> response = restTemplate.getForEntity(uri, String.class);
    }
```

설명

- 1 API 요청을 위해 처음에는 String 문자열에 Http경로를 작성하고 시크릿 키를 변수로 넣었으나 시크릿키가 일치하지 않아 요청이 거부되는 오류가 발생하였다.

해당 오류의 원인이 시크릿키를 .properties에서 @Value 어노테이션을 활용해 변수로 받아오는 과정에서 인코딩이 일어난다는 것임을 확인하고 해결방법을 서칭하였다.

주석의 블로그에서 같은 문제를 발견하고 해결법을 참고하여 URL 빌더를 활용하였다.

- 2 URL 빌더를 활용해 API 요청 URL생성 서비스키와 가져올 캠핑장 수를 파라미터로 전달

- 3 RestTemplate 객체를 활용해 해당 URL을 외부 API에 요청
반환값을 String으로 저장

코드

CampService.java

```
1 String jsonString = response.getBody();
System.out.println(response.getBody());

2 //json 데이터를 자바 객체(캠프엔티티)로 맵핑
ObjectMapper objectMapper = new ObjectMapper();
JsonNode root = objectMapper.readTree(jsonString);

//고캠핑 제이슨데이터의 구성요소(메뉴얼 참고)
JsonNode items = root
    .path( s: "response")
    .path( s: "body")
    .path( s: "items")
    .path( s: "item");

3 List<Campsite> campsiteList = new ArrayList<>();
List<CampThumbFiles> thumbList = new ArrayList<>();
List<CampImageFiles> imageList = new ArrayList<>();
```

설명

- 1 String 문자열로 받아온 JSON 데이터 중 필요한 부분인 Body만 따로 저장
- 2 String 문자열로 받아온 JSON데이터에 접근하기 위해 ObjectMapper로 JsonNode 객체로 변환
JSON데이터의 구성요소는 매뉴얼을 참고해 필드 값을 추출하였다
- 3 캠핑장 정보, 썸네일 경로, 이미지 경로를 저장할 Array 선언

코드

CampService.java

```
1  for(JsonNode item : items){  
    Random random = new Random();  
  
    //데이터 일부 가공  
    String sbrsCl = item.path("sbrsCl").asText(defaultValue: "");  
    String posblFclyCl = item.path("posblFclyCl").asText(defaultValue: "");  
    String animalCmgCl = item.path("animalCmgCl").asText(defaultValue: "");  
    String lineIntro = item.path("lineIntro").asText();  
    String intro = item.path("intro").asText();  
    String featureNm = item.path("featureNm").asText();  
    String tooltip = item.path("tooltip").asText();  
    String campInfo = lineIntro + intro + featureNm + tooltip;  
  
    //해당 정보들이 문자열로 되어있어서 boolean으로 변환  
    boolean hotWater = sbrsCl.contains("온수");  
    boolean electric = sbrsCl.contains("전기");  
    boolean firewood = sbrsCl.contains("장작판매");  
    boolean wifi = sbrsCl.contains("무선인터넷");  
    boolean playGround = posblFclyCl.contains("운동장");  
    boolean pet = animalCmgCl.equals("가능");  
    boolean swimming = posblFclyCl.contains("물놀이");
```

설명

- ① 캠핑장 정보가 저장된 item 필드를 for문으로 하나씩 맵핑
- ② API에 저장된 데이터를 Entity에 맞게 가공 예) 한줄설명, 상세설명, 이용팁 등으로 나눈 캠핑장 소개를 String campInfo 필드에 한번에 저장하기 위해 하나로 합침
- ③ 텍스트로 저장되어있는 캠핑장 옵션 확인하고 Boolean값으로 전환

코드

CampService.java

```
//캠프엔티티에 빌더로 맵핑
1 Campsite camp = Campsite.builder()
    .campName(item.path("facltNm").asText())
    .campInfo(campInfo)
    .campTel(item.path("tel").asText())
    .campAddressDo(item.path("doNm").asText())
    .campAddressGungu(item.path("sigunguNm").asText())
    .campAddress1(item.path("addr1").asText())
    .campAddress2(item.path("addr2").asText())
    .campMapX(item.path("mapX").asText())
    .campMapY(item.path("mapY").asText())
    .toilet(item.path("toiletCo").asInt())
    .hotWater(hotWater)
    .electric(electric)
    .fireWood(firewood)
    .wifi(wifi)
    .playGround(playGround)
    .pet(pet)
    .swimming(swimming)
2 .totalCapacity(random.nextInt(bound)+30) //총 자리수가 json에 없어서 랜덤으로 생성 30~100
    .campHit(0)
    .campLike(0)
    .apiContentId(item.path("contentId").asText())
    .build();
3 campsiteList.add(camp);
```

설명

- 1 가공이 완료된 데이터를 Entity에 Builder 패턴으로 맵핑
- 2 총 자릿수는 API에 없어서 랜덤으로 넣었다
조회수와 좋아요 수는 디폴트 값인 0
- 3 맵핑된 Entity는 선언해둔 List에 저장

코드

CampService.java

```
//썸네일 저장
1 if(!item.path(s: "firstImageUrl").asText().isEmpty()){
    //확장자 추출
    String fileUrl = item.path(s: "firstImageUrl").asText();
    int typeIndex = fileUrl.lastIndexOf(str: ".");
    String fileType = fileUrl.substring(beginIndex: typeIndex+1);

    CampThumbFiles thumb = CampThumbFiles.builder()
        .originalFileName("Thumbnail_" + item.path(s: "facltNm").asText())
        .filePath(fileUrl)
        .fileType(fileType)
        .build();
    thumb.setMappingCamp(camp);
    thumbList.add(thumb);
}
```

설명

- 1 썸네일 URL도 JSON데이터에 따로 저장되어 있어서 썸네일 Entity에 맞게 가공후 맵핑 선언해둔 List에 저장

코드

CampService.java

```
final var imageUrlbuilder = new DefaultUriBuilderFactory();
imageUrlbuilder.setEncodingMode(DefaultUriBuilderFactory.EncodingMode.NONE);
final String ImageUriString = builder.builder() UriBuilder
    .scheme("http")
    .host("apis.data.go.kr")
    .path("B551011/GoCamping/imageList")
    .queryParam( name: "serviceKey", serviceKey)
    .queryParam( name: "numOfRows", ...values: 8)
    .queryParam( name: "pageNo", ...values: 1)
    .queryParam( name: "MobileOS", ...values: "ETC")
    .queryParam( name: "MobileApp", ...values: "CampingLounge")
    .queryParam( name: "contentId", item.path( s: "contentId").asText())
    .queryParam( name: "_type", ...values: "json")
    .build() URI
    .toString();

final URI ImageUri = URI.create(ImageUriString);

try {
    RestTemplate imageRestTemplate = new RestTemplate();
    ResponseEntity<String> imageResponse = imageRestTemplate.getForEntity(ImageUri, String.class);

    String imageJsonString = imageResponse.getBody();

    //json 데이터를 자바 객체(캠프엔티티)로 매팅
    ObjectMapper imageObjectMapper = new ObjectMapper();
    JsonNode imageRoot = imageObjectMapper.readTree(imageJsonString);
```

설명

- 상세이미지 파일은 별도의 API 요청이 필요하여 이전과 같은 방법으로 요청 후 문자열로 데이터를 받아 JSON 데이터로 매팅

코드

CampService.java

```
for(JsonNode imageItem : imageItems){  
    //확장자 추출  
    String imageUrl = imageItem.path("imageUrl").asText();  
    int typeIndex = imageUrl.lastIndexOf(str ".");  
    String fileType = imageUrl.substring(beginIndex, typeIndex+1);  
  
    CampImageFiles images = CampImageFiles.builder()  
        .originalFileName("image_" + item.path("facltNm").asText() + imageItem.path("serialnum").asText())  
        .filePath(imageItem.path("imageUrl").asText())  
        .fileType(fileType)  
        .build();  
    images.setMappingCamp(camp);  
    imageList.add(images);  
}  
}catch (Exception e) {  
    e.printStackTrace();  
    System.err.println("고캠핑 공공 API 이미지 목록 조회에 실패했습니다.");  
}  
}  
  
② campRepository.saveAll(campsitelist);  
campThumbRepository.saveAll(thumbList);  
campImagesRepository.saveAll(imageList);
```

설명

- ① 데이터를 Entity에 맞게 가공 후 저장
- ② 각각 List에 저장된 객체는 각각의 Repository의 saveAll() 매서드를 활용하여 DB에 저장

ReviewWrite.js

```
1 import ReactQuill, { Quill } from 'react-quill';
2 import 'react-quill/dist/quill.snow.css';
3 import ImageResize from 'quill-image-resize-module-react';
4 Quill.register('modules/ImageResize', ImageResize);
```

코드 리뷰

상세 설명

- 1 리액트 퀼 에디터를 사용하려면
리액트 퀼 라이브러리를 설치하고
(npm install react-quill)
import 해야한다.
- 2 'react-quill/dist/quill.snow.css' ;
에디터의 기본 css를 import 해야
글 작성폼이 생긴다.
- 3 퀼 에디터로 업로드 한 이미지의 사이즈를
조절하기 위해서는
(npm install quill-image-resize-
module-react) 를 설치하고, import
- 4 퀼 모듈을 등록해서 사용한다.

ReviewWrite.js

```
1 const modules = {  
  toolbar: [  
    { header: '1' }, { header: '2' }, { font: [] },  
    ['bold', 'underline'],  
    2 ['image'],  
    [{ color: [] }],  
  ],  
  3 ImageResize: {  
    modules: ['Resize', 'DisplaySize'],  
  },  
  clipboard: {  
    matchVisual: false,  
  },  
};
```

임고준

상세 설명

- 1 컴포넌트 내부에 모듈을 선언하고 속성을 추가해주는데,
- 2 기본적으로 이미지 업로드 기능이 없는 상태이기 때문에 [`image`] 를 추가해야 이미지 업로드 버튼이 나타난다.
- 3 ImageResize 속성도 추가해야 이미지에 사이즈를 조절하는 박스가 생긴다.

ReviewWrite.js

1

```
<ReactQuill  
  ref={quillRef}  
  value={reviewContent}  
  onChange={changeContent}  
  placeholder="내용을 입력해주세요"  
  theme="snow"  
  modules={modules}  
  style={{  
    height: '500px',  
  }}  
/>
```

상세 설명

- 1 <ReactQuill /> 을 return 문 내부에 작성폼이 필요한 곳에 배치한다.

내부 속성에 modules={modules} 를 넣어야 등록한 모듈을 사용 할 수 있다.

value={reviewContent} 는 에디터의 초기 상태값.

onChange={changeContent} 는 에디터에 입력할 때 실행되는 함수. 입력한 내용이 value에 업데이트 된다.

ReviewWrite.js

1

```
const satisOptions = [
  { value: 1, label: '매우 불만족해요' },
  { value: 2, label: '불만족해요' },
  { value: 3, label: '보통이에요' },
  { value: 4, label: '만족해요' },
  { value: 5, label: '완전 만족해요' },
];
```

```
const sizeOptions = [
  { value: 1, label: '매우 좁아요' },
  { value: 2, label: '좁아요' },
  { value: 3, label: '충분해요' },
  { value: 4, label: '넓어요' },
  { value: 5, label: '완전 넓어요' },
];
```

```
const cleanOptions = [
  { value: 1, label: '더러워요' },
  { value: 2, label: '지저분해요' },
  { value: 3, label: '보통이에요' },
  { value: 4, label: '깨끗해요' },
  { value: 5, label: '완전 깨끗해요' },
];
```

2

```
<ul className="radio_list radio-lg">
  {satisOptions.map((option) => (
    <li key={option.value}>
      <input
        type="radio"
        name="radio_01"
        id={`radio_01_${option.value}`}
        value={option.value}
        onChange={() =>
          handleRadioChange(
            setReviewSatisfaction,
            option.value
          )
        }
      />
      <label htmlFor={`radio_01_${option.value}`}>
        {option.label}
      </label>
    </li>
  ))}
</ul>
```

상세 설명

리뷰 작성시 이용했던 캠핑장에 대해
만족도를 평가할 수 있다.

- | | | | | | |
|--------|--------------------------------|-----------------------------|-----------------------------|----------------------------|-------------------------------|
| 만족도 | <input type="radio"/> 매우 불만족해요 | <input type="radio"/> 불만족해요 | <input type="radio"/> 보통이에요 | <input type="radio"/> 만족해요 | <input type="radio"/> 완전 만족해요 |
| 사이트 크기 | <input type="radio"/> 매우 좁아요 | <input type="radio"/> 좁아요 | <input type="radio"/> 충분해요 | <input type="radio"/> 넓어요 | <input type="radio"/> 완전 넓어요 |
| 청결도 | <input type="radio"/> 더러워요 | <input type="radio"/> 지저분해요 | <input type="radio"/> 보통이에요 | <input type="radio"/> 깨끗해요 | <input type="radio"/> 완전 깨끗해요 |
| 친절함 | <input type="radio"/> 완전 불친절해요 | <input type="radio"/> 불친절해요 | <input type="radio"/> 보통이에요 | <input type="radio"/> 친절해요 | <input type="radio"/> 완전 친절해요 |

1 평가 항목을 하나의 변수로 묶고,
세부 항목에 value 와 label 로 각각
번호와 평점을 지정한다.

2 {satisOptions.map((option) => (...))}
→ satisOptions 배열을 반복, 각각의
라디오 버튼을 생성

value={option.value}
→ 라디오 버튼 선택 시 저장되는 값
(예: 1, 2, 3, ...)

onChange = { () => handleRadioChange
(setReviewSatisfaction, option.value) }
→ 클릭 시 실행될 함수 (handleRadioChange)

<label htmlFor={`radio_01_\${option.value}`}>
>{option.label}</label>
→ 라벨을 클릭하면 해당 라디오 버튼이 선택되도록
연결

1

만족도

- 매우 불만족해요
- 불만족해요
- 보통이에요
- 만족해요
- 완전 만족해요

사이트 크기

- 매우 좁아요
- 좁아요
- 충분해요
- 넓어요
- 완전 넓어요

청결도

- 더러워요
- 지저분해요
- 보통이에요
- 깨끗해요
- 완전 깨끗해요

친절함

- 완전 불친절해요
- 불친절해요
- 보통이에요
- 친절해요
- 완전 친절해요

파일 선택 선택된 파일 없음

H1 H2 Sans Serif : B U I A

내용을 입력해주세요

2

상세 설명

1 이용한 캠핑장의 리뷰를 작성하면서 만족도를 같이 평가할 수 있다.

2 리액트 퀘일 에디터가 적용되어 이미지 업로드, 글 입력 등 글 작성이 가능해졌다.

회원은 만족도를 나타내는 라디오 버튼 중 하나를 선택한다.

<input type="radio"> 태그를 사용하여 각 점수에 해당하는 버튼을 렌더링한다.

React의 useState를 활용하여 선택된 값이 상태(state)에 저장된다.

Review.java (Entity)

```
@Entity  
@Getter  
@Setter  
@NoArgsConstructor  
public class Review extends BaseTimeEntity{  
  
    @Column(name = "review_satisfaction")  
    private Integer reviewSatisfaction;  
  
    @Column(name = "review_size")  
    private Integer reviewSize;  
  
    @Column(name = "review_clean")  
    private Integer reviewClean;  
  
    @Column(name = "review_kindness")  
    private Integer reviewKindness;
```

@Getter

@NoArgsConstructor

```
public class ReviewWriteDto {  
    private String reviewTitle;  
    private String reviewContent;  
    private Integer reviewSatisfaction;  
    private Integer reviewSize;  
    private Integer reviewClean;  
    private Integer reviewKindness;
```

@Builder no usages

```
public ReviewWriteDto( String reviewTitle, String  
    this.reviewTitle = reviewTitle;  
    this.reviewContent = reviewContent;  
    this.reviewSatisfaction = reviewSatisfaction;  
    this.reviewSize = reviewSize;  
    this.reviewClean = reviewClean;  
    this.reviewKindness = reviewKindness;  
    this.reviewImages = reviewImages;  
    this.memberId = memberId;  
    this.reservationId = reservationId;  
    this.campId = campId;  
    this.reviewHit = reviewHit;  
    this.reviewLike = reviewLike;
```

ReviewWriteDto.java

상세 설명

입력된 데이터는 ReviewWriteDto를 통해 Review 엔티티로 변환되고, 이 엔티티는 데이터베이스에 저장된다.

- reviewTitle → 리뷰 제목
- reviewContent → 리뷰 본문
- reviewSatisfaction → 선택한 만족도 점수 (1~5)
- reviewSize → 사이트 크기 평가 (1~5)
- reviewClean → 청결도 평가 (1~5)
- reviewKindness → 친절도 평가 (1~5)

ReviewService.java

```
// Review Write
public ResReviewWriteDto writeReviews(ReviewWriteDto reviewWriteDto) throws IOException {
    Member member = memberRepository.findById(reviewWriteDto.getMemberId())
        .orElseThrow(() -> new IllegalArgumentException("Member not found"));

    Campsite campsite = campRepository.findById(reviewWriteDto.getCampId())
        .orElseThrow(() -> new IllegalArgumentException("Camp not found"));

    Reservation reservation = reservationRepository.findById(reviewWriteDto.getReservationId())
        .orElseThrow(() -> new IllegalArgumentException("존재하지 않는 예약입니다."));

    Review review = reviewWriteDto.toEntity(member, campsite, reservation);

    Review savedReview = reviewRepository.save(review);

    return ResReviewWriteDto.fromEntity(savedReview);
}
```

상세 설명

- 1 reviewWriteDto에서 전달받은 memberId를 통해 memberRepository를 사용해 해당 회원을 데이터베이스에서 조회한다.
- 2 reviewWriteDto에서 받은 campId를 사용하여 campRepository를 통해 해당 캠핑장을 데이터베이스에서 조회한다.
- 3 리뷰 작성시 캠핑장 예약 정보를 불러와야 하기 때문에 reservationRepository에서 필요한 데이터를 조회한다.
- 4 reviewWriteDto 객체에 담긴 데이터를 사용하여 toEntity() 메서드를 호출해 Review 엔티티 객체를 생성한다. 이때 생성된 Review 객체는 member, campsite, reservation 정보도 포함한다.
- 5 데이터베이스에 저장된 savedReview 엔티티 객체를 ResReviewWriteDto.fromEntity() 메서드를 통해 ResReviewWriteDto DTO로 변환하고, 이를 반환하여 클라이언트에 응답을 보낸다.

ResReviewDetailDto.java

```
@Getter 25 usages sumhaa
@NoArgsConstructor
public class ResReviewDetailDto {

    private Integer reviewSatisfaction;
    private Integer reviewSize;
    private Integer reviewClean;
    private Integer reviewKindness;

    public static ResReviewDetailDto fromEntity (Review review){
        return ResReviewDetailDto.builder()
            .reviewSatisfaction(review.getReviewSatisfaction())
            .reviewSize(review.getReviewSize())
            .reviewClean(review.getReviewClean())
            .reviewKindness(review.getReviewKindness())
    }
}
```

상세 설명

사용자가 특정 리뷰의 상세 페이지를 요청하면, 백엔드는 reviewId를 기반으로 해당 리뷰 데이터를 조회한다.

조회된 Review 엔티티 데이터를 ResReviewDetailDto로 변환하여 프론트엔드에 전달한다.

변환 과정에서 엔티티의 필드 값을 가공하거나, 추가적인 데이터(예: 작성자의 닉네임, 캠핑장 이름 등)를 포함할 수 있다.

ReviewService.java

```
// Review Detail
public ResReviewDetailDto detailReviews(Long reviewId) { 2 usages  ↳ sumhaa
    // 리뷰 조회
    Review review = reviewRepository.findById(reviewId)
        .orElseThrow(() -> new ResourceNotFoundException("Review", "Review Id", String.valueOf(reviewId)));
    // 조회수 증가
    review.incrementReviewHit();

    // 댓글 불러오기
    List<Comment> comment = review.getComment();

    return ResReviewDetailDto.fromEntity(review);
}
```

상세 설명

reviewRepository.findById(reviewId)를 통해 Review 엔티티를 조회하고 필요한 데이터를 추출한다.

조회된 review 엔티티를 ResReviewDetailDto로 변환하는데, 이 과정에서 reviewSatisfaction, reviewSize, reviewClean, reviewKindness 값을 DTO에 맵핑한다.

ReviewDetail.js

```
1 // 리뷰 데이터를 가져오는 함수
2 const fetchReviewData = async () => {
3   if (isFetched.current) return; // 이미 데이터 가져왔으면 중단
4   isFetched.current = true;
5
6   try {
7     const reviewResponse = await axios.get(
8       `http://localhost:8080/review/${reviewId}`
9     );
10    setReview(reviewResponse.data); // 리뷰 데이터 상태 업데이트
11
12    console.log('리뷰 데이터', JSON.stringify(reviewResponse.data));
13  } catch (error) {
14    console.error('Error fetching review data:', error);
15  }
16}
```

```
    "reviewTitle": "테스트",
    "reviewContent": "#u00d7",
    "reviewHit": 75,
    "reviewLikes": 3,
    "reviewSatisfaction": 5,
    "reviewSize": 4,
    "reviewClean": 4,
    "reviewKindness": 4,
```

- fetchReviewData 함수는 특정 리뷰 데이터를 백엔드에서 가져와 화면에 표시 할 수 있도록 상태를 업데이트하는 것이다. 중복 요청을 방지하여 불필요 한 네트워크 호출을 줄이고, 가져온 데이터를 setReview를 통해 저장함으로써 화면에서 해당 리뷰 정보를 활용할 수 있도록 한다.
 - reviewId를 기반으로 특정 리뷰의 상세 정보를 백엔드에서 가져와 React의 상태로 저장한 후, 이를 화면에 렌더링할 수 있도록 하는 로직.
 - axios.get([http://localhost:8080/review/\\${reviewId}](http://localhost:8080/review/${reviewId}))를 사용하여 백엔드 API에 GET 요청을 보내고 reviewId에 해당하는 리뷰 데이터를 조회한 후 JSON 형태로 응답을 반환한다.
 - 이 데이터에는 리뷰 내용뿐만 아니라 dto를 통해 전송된 만족도 관련 데이터도 포함되어 있다.

OauthService

```

public class OauthService {
    @Value("${spring.security.oauth2.client.registration.google.client-id}")
    private String googleClientId; ①

    // 1. 구글 토큰이 신뢰할 수 있는 토큰인지 검증
    // 2. 정보가 없으면 회원가입
    // 3. 토큰 발급
    public Map<String, String> googleLogin(String token) { ②
        try {
            // GoogleIdTokenVerifier : 구글에서 제공하는 토큰 검증 클래스
            // NetHttpTransport : HTTP 통신을 처리
            // JacksonFactory : JSON 데이터를 읽고 분석함
            // setAudience : 클라이언트 ID가 인증화면 id와 같은지 검증
            GoogleIdTokenVerifier verifier = new GoogleIdTokenVerifier.Builder(new NetHttpTransport(), JacksonFactory.getDefaultInstance())
                .setAudience(Collections.singletonList(googleClientId))
                .build();

            GoogleIdToken idToken = verifier.verify(token);
            if (idToken == null) {
                System.out.println("유효하지 않은 토큰");
                throw new AuthorizationServiceException("유효하지 않은 토큰");
            }
        } ③

        GoogleIdToken.Payload payload = idToken.getPayload();
        String email = payload.getEmail();
        String name = payload.get("name").toString();
        String profile_url = payload.get("picture").toString();

        Optional<Member> member = memberRepository.findByEmail(email); ④
    }
}

```

- 1 차적으로 프로퍼티에서 구글 클라이언트Id를 받아와 googleClientId에 넣는다.
- 2 Token을 받은뒤 GoogleIdTokenVerifier를 이용하여 토큰을 검증할 것이다. 그 전에 GoogleIdTokenVerifier를 Build 해야한다. NetHttpTransport 를 이용해 HTTP 요청이 가능하게 설정하고 JacksonFactory, getDefaultInstance를 이용해 JSON 파싱을 가능하게 한다. 그 후 SetAudience를 이용해 클라이언트 ID가 인증화면 클라이언트 ID와 같은지 검증한다. SetAudience는 List 값만 받기에 List로 변환후 값을 넣어준다. 그 후 verifier.verify에 토큰을 집어넣어 토큰을 검증하고 검증된 토큰을 idToken에 넣어준다.
- 3 검증된 토큰 값의 payload에는 구글 로그인 한 회원이 제공한 정보가 담겨있다. payload에서 이메일과 이름, 그리고 프로필url을 받아온다.
- 4 그 후 email을 이용해 memberRepository에서 member를 찾는다.

```

1
if(member.isEmpty()) {
    Member newMember = Member.builder()
        .email(email)
        .name(name)
        .role(Role.USER)
        .enable(true)
        .join_date(new Date())
        .profile_url(profile_url)
        .build();

    System.out.println(newMember);

    memberRepository.save(newMember);
} else {
    System.out.println(member);
}

```

```

2
UserDetails userDetails = userDetailsService.loadUserByUsername(email);

String jwtToken = jwtTokenUtil.generateToken(userDetails);

Map<String, String> map = new HashMap<>();
map.put("email", email);
map.put("token", jwtToken);

MemberResponseDto User = memberService.getUserByEmail(email);
map.put("id", User.getId().toString());

return map;

3
} catch (Exception e) {
    System.out.println("токен 검증 중 예외 발생 : " + e.getMessage());
    throw new AuthorizationServiceException("신뢰할 수 없는 토큰");
}
}

```

1 Member가 비어있다는 걸 확인 한다면 이메일, 이름, USER권한, 가입날짜, 프로필url을 이용해 build 후 memberRepository에 저장한다.

2 generateToken 함수에서 UserDetails 값을 받기 때문에 userDetailsService.loadUserByUsername을 이용해 userDetials 값을 얻고 그 값을 generateToken에 넣어 토큰을 발급받는다. 그 이후 HashMap을 만들고 그 안에 email, token, member Id 값을 넣어 return 한다.

3 만약 앞에 있는 과정 중 문제가 발생하면 예외 발생을 일으킨다.

ProfileFileService

```
@Service 4 usages new *
public class ProfileFileService {
    private final MemberRepository memberRepository; 2 usages
    private final ProfileFilesRepository profileFilesRepository; 5 usages
    private final String UPLOAD_DIR = Paths.get(System.getProperty("user.dir"), ...more: "uploads").toString(); 1 usages

    public ProfileFileService(MemberRepository memberRepository, ProfileFilesRepository profileFilesRepository) { new *
        this.memberRepository = memberRepository;
        this.profileFilesRepository = profileFilesRepository;
    }

    // 기존 프로필 있으면 삭제 ( 파일 + DB )
    public void deleteFile(Long id) { 3 usages new *
        Optional<MemberProfileFiles> existingProfile = profileFilesRepository.findById(id);
        if (existingProfile.isPresent()) {
            Path oldFilePath = Paths.get(UPLOAD_DIR, existingProfile.get().getStored_file_name());
            File oldFile = oldFilePath.toFile();
            if (oldFile.exists()) {
                if (oldFile.delete()) {
                    System.out.println("이전 프로필 삭제 성공: " + oldFile.getAbsolutePath());
                } else {
                    System.out.println("이전 프로필 삭제 실패: " + oldFile.getAbsolutePath());
                }
            }
            profileFilesRepository.delete(existingProfile.get());
        }
    }
}
```

1 업로드 될 폴더의 경로를 지정한다.
System.getProperty("user.dir")는 현재 작업 폴더의 root를 의미하며 "uploads"는 만들어질 폴더 이름이다.
Paths.get은 안의 파라미터 값을 경로로 만들어 준다. 즉 root/uploads 형식으로 실행된 OS에 맞게 경로를 작성해준다.

2 id를 파라미터로 받아서 profileFileRepository에 파일이름이 존재하는지 확인한 후 존재할 경우, Paths.get을 이용해 저장된 파일이름을 경로로 만들고 그 경로로 File을 불러온다.

그 후 파일을 삭제하고 repository에서도 데이터를 삭제한다.

```

public String uploadFile(Long id, MultipartFile file) { 1 usage new +
    Optional<Member> member = memberRepository.findById(id);

    if (member.isEmpty()) {
        return "회원을 찾지 못했습니다.";
    }

    String filename = file.getOriginalFilename();

    // 폴더 없으면 폴더 생성
    File directory = new File(UPLOAD_DIR);
    if (!directory.exists()) {
        directory.mkdirs();
    }

    deleteFile(id); 2

    // 새 파일 저장
    String storedFileName = UUID.randomUUID().toString() + "_" + filename;
    Path filePath = Path.of(UPLOAD_DIR, storedFileName);

    try {
        file.transferTo(filePath.toFile());
    } catch (IOException e) {
        return "파일 저장 중 오류 발생 : " + e.getMessage();
    }

    // DB에 파일 정보 저장
    MemberProfileFiles memberProfileFiles = new MemberProfileFiles();

    memberProfileFiles.setMember(member.get());
    memberProfileFiles.setOrigin_file_name(filename);
    memberProfileFiles.setStored_file_name(storedFileName);

    profileFilesRepository.save(memberProfileFiles);

    return "프로필 업로드 성공";
}

```

- 1 member 를 id로 찾고 받은 file의 이름도 찾는다.
- 2 앞에서 지정한 UPLOAD_DIR를 이용해 폴더가 없을 경우 mkdirs() 함수로 폴더를 만든다. deleteFile() 함수를 이용해 이전 파일이 존재할 경우 삭제 한다.
- 3 파일이름이 겹치지 않도록 randomUUID() 함수를 이용해 저장용 파일이름을 새로 만든 후 그 이름으로 파일을 transferTo() 함수를 이용해 폴더에 저장한다.
- 4 파일이 잘 저장되었다면 profileFilesRepository에 멤버id, 파일이름, 저장용 파일이름을 저장한다.

● MemberService

```
public void join(MemberRegisterDto memberRegisterDto) { 1 usage new *
    // 비밀번호 암호화 하기
    String encodedPassword = passwordEncoder.encode(memberRegisterDto.getPassword()); 1

    Member member = new Member();
    member.setEmail(memberRegisterDto.getEmail());
    member.setName(memberRegisterDto.getName());
    member.setPassword(encodedPassword);
    member.setGender(memberRegisterDto.getGender());
    member.setTel(memberRegisterDto.getTel());
    member.setJoin_date(memberRegisterDto.getJoin_date());
    member.setRole(memberRegisterDto.getRole());
    member.setEnable(true);
    member.setAddress(memberRegisterDto.getAddress());
    member.setAddress_detail(memberRegisterDto.getAddress_detail());
    member.setPostcode(memberRegisterDto.getPostcode());

    // 데이터 베이스에 저장
    memberRepository.save(member); 2
}
```

```
public MemberTokenDto login(MemberLoginDto memberLoginDto) { 1 usage new *
    //UserDetailsService : Spring Security에서 사용자의 정보를 가져올 때 사용하는 서비스 3
    UserDetails userDetails = userDetailsService.loadUserByUsername(memberLoginDto.getEmail());

    authenticate(memberLoginDto.getEmail(), memberLoginDto.getPassword()); 4

    String token = jwtTokenUtil.generateToken(userDetails);
    return MemberTokenDto.fromEntity(userDetails, token); 5
}
```

- 1 멤버의 회원가입 정보를 프론트에서 받아온 후, passwordEncoder.encode 를 이용해 비밀번호를 암호화 한다.
- 2 그 이후 암호화 한 비밀번호와 정보들을 member 객체에 넣은 후 memberRepository에 저장한다.
- 3 받아온 로그인 정보 중 이메일을 userDetailsService.loadUserByUsername에 넣어서 userDetail을 받아온다.
*userDetailService 는 Spring Security와 연동되어 인증을 관리해준다.
- 4 그 이후 authenticate 를 이용해 로그인 인증을 한다.
authenticate에서 내부적으로 PasswordEncoder.matches 를 사용해 해시된 비밀번호를 검증해준다.

- 5 인증이 완료되었다면 generateToken 을 이용해 토큰을 만들고 userDetails와 token 을 객체로 만든 후 return 한다.

```
1  
private void authenticate(String email, String pwd) { 1 usage new *  
    try {  
        authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(email, pwd));  
    } catch (DisabledException e) {  
        throw new MemberException("인증되지 않은 아이디입니다.", HttpStatus.BAD_REQUEST);  
    } catch (BadCredentialsException e) {  
        throw new MemberException("비밀번호가 일치하지 않습니다.", HttpStatus.BAD_REQUEST);  
    }  
}
```

- 1 앞에서 사용한 authenticate 함수. 사용자가 로그인할 때 email, pwd를 이용해 인증하는 역할을 한다.
- 2 UsernamePasswordAuthenticationToken 객체를 만들고 authenticate 함수에 넣어 인증을 한다.
`authenticationManager.authenticate()` 함수가 내부적으로 여러 검증을 수행한다.
 - 1 - `UserDetail` 을 가져오고
 - 2 - 비밀번호를 검증하고
 - 3 - 계정이 비활성화 되었는지 아닌지 계정상태를 확인한다.이러한 내부 검증을 통해 인증을 한다.

● MemberController

```

1 @PutMapping("*/update") new *
2 public ResponseEntity<MemberResponseDto> userUpdate(@AuthenticationPrincipal Member member, @RequestBody MemberUpdateDTO memberUpdateDTO) {
3     memberUpdateDTO.setEnable(true); // 2
4     MemberResponseDto memberUpdate = memberService.userUpdate(member, memberUpdateDTO);
5     return ResponseEntity.status(HttpStatus.OK).body(memberUpdate);
}

```

```

// 구글 로그인
1 @PostMapping("*/auth/google") new *
2 public ResponseEntity<?> googleLogin(@RequestBody Map<String, String> request) { // 4
3     String googleToken = request.get("token");
4
5     Map<String, String> res = oauthService.googleLogin(googleToken);
6     String jwtToken = res.get("token");
7     if(res.get("enable").equals("false")) {
8         return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("탈퇴한 회원입니다.");
9     }
10
11     return ResponseEntity.status(HttpStatus.OK).header(jwtToken).body(res);
}

```

- 1 @AuthenticationPrincipal 을 이용해 현재 로그인 중인 회원 정보와 @RequestBody 를 이용해 업데이트 된 정보를 가져온다.
- 2 boolean타입인 enable은 null을 반환할 수 없어 타입을 Boolean으로 바꾸거나 다른 부분에서 값을 넣어줘야한다.
따라서 setEnable(true)를 작성하여 기본값을 true로 넣었다.
- 3 그 이후 memberService.userUpdate에 정보를 담아 보내 업데이트를 한다.
그 후 response 값을 반환한다.
- 4 requestBody로 받아온 값에서 토큰을 얻고 oauthService.googleLogin에 넘긴다.
- 5 그 후 받아온 response 값을 이용해 다시 반환된 토큰을 얻어오고 만약 response.enable 값이 false 라면 탈퇴한 회원이라고 반환한다.
그렇지 않다면 header에 토큰값을 담고 body 엔 response 값을 담아 반환한다.

WebConfig

```
@Configuration new +
public class WebConfig implements WebMvcConfigurer {
    @Override no usages new *
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        1 // toUri = OS에 따라 적절한 URI로 변환
        // Paths.get = OS에 따라 적절한 경로로 변환
        // System.getProperty("user.dir") = 프로젝트의 루트 경로를 가져옴
        String uploadPath = Paths.get(System.getProperty("user.dir"), ...more: "uploads").toUri().toString();

        2 // addResourceHandler = URL 요청 패턴 처리
        // addResourceLocations() = 요청된걸 어떤거에서 가져올지 정해줌
        // /uploads/ 로 시작하는 모든 요청을 uploadPath로 변환
        registry.addResourceHandler(...pathPatterns: "/uploads/**")
            .addResourceLocations(uploadPath);
    }
}
```

- 1 System.getProperty("user.dir") 와 "uploads"를 Paths.get에 넣어서 OS에 따라 적절한 경로로 변환하도록 한다.
System.getProperty("user.dir")는 프로젝트의 루트 경로를 가져온다.
toUri는 경로를 OS에 따라 적절한 URI로 변경해 준다.
변경된 경로를 uploadPath에 넣는다.
- 2 Registry.addResourceHandler를 통해 "/uploads/"로 시작하는 모든 요청을 uploadPath로 변경한다.

Address.js

```
function Address({setAddress, setPostcode}) { Show usages

  useEffect( effect: () : void  => {
    if (!window.daum) {
      const script : HTMLScriptElement = document.createElement( tagName: "script");
      script.src = "https://t1.daumcdn.net/mapjsapi/bundle/postcode/prod/postcode.v2.js";
      script.async = true;
      document.body.appendChild(script);
    }
  }, [deps: []]);

  const openPostcode = () : void  => { Show usages
    new window.daum.Postcode({
      oncomplete: function (data) : void  {

        setAddress(data.address);
        setPostcode(data.zonecode);
      }
    }).open();
  };

  return (
    <div>
      <button type="button" className="btn btn-p-f btn-sm" onClick={openPostcode}>주소 찾기</button>
    </div>
  );
}
```

1

2

3

1 카카오 우편번호 검색 API를 활용하기 위해 스크립트를 동적으로 추가한다.

document.createElement를 이용해 script를 만들고 script에 src 값 async 값을 넣는다.

이후, document.body.appendChild를 이용해 script를 집어 넣는다.

2 우편번호 검색창을 여는 함수

new window.daum.Postcode를 통해 우편번호 검색창을 띄운다.

oncomplete는 사용자가 주소를 선택하면 실행되는 콜백함수이다.

받은 data를 이용해 주소 및 우편번호를 설정한다.

3 이후 버튼을 누르면 openPostcode 함수가 실행되는 html 을 return 한다.

● GoogleLoginBtn.js

```
return (
  <GoogleOAuthProvider clientId={process.env.REACT_APP_GOOGLE_CLIENT_ID} >
    <GoogleLogin onSuccess={handleOnSuccess} onError={handleOnError} />
  </GoogleOAuthProvider>
)

const handleOnSuccess = async (response: Promise<void>) => {
  const token: string = response.credential;

  try {
    const response: AxiosResponse<any> = await axios.post(url: "http://localhost:8080/member/auth/google", data: {token: token});

    if (response.status === 200 || response.status === 201) {
      alert("로그인 성공");

      localStorage.setItem("CL_access_token", response.data.token);
      localStorage.setItem("email", response.data.email);
      localStorage.setItem("id", response.data.id);
      setCookie(name: "loggedIn", value: true);

      setAuth(response.data.email);
      setHeaders({ "Authorization": `Bearer ${response.data.token}` });

      navigate('/memberDetail/${response.data.id}`);
    } else {
      alert("로그인 실패: " + response.data.message);
    }
  } catch (e) {
    alert(e.response?.data?.error || e.response?.data || "로그인 실패");
    console.error("로그인 오류: ", e);
  }
}
```

1
2

3

4

- 1 Google의 인증기능을 제공하는 최상위 Provider 클라이언트 ID를 제공해야 Google API와 연동이 가능하기에 .env 파일에 클라이언트 ID 값을 적고 불러와서 clientId를 넣어줬다.
- 2 Google 로그인 버튼을 제공하는 컴포넌트 onSuccess 는 로그인 성공 시 실행할 함수가 들어간다. onError 는 로그인 실패 시 실행할 함수가 들어간다.
- 3 member/auth/google로 response로 받은 token 값과 함께 post 요청을 보낸 후 요청이 성공 하였을 경우, 로컬스토리지에 토큰과 이메일 아이디 값을 넣고, 브라우저 종료시 로그아웃을 위한 loggedIn 쿠키를 설정한다.
- 4 이후 Auth, Headers 를 Set 한 후 마이페이지로 이동시킨다.

● Login.js

```
function Login() { Show usages
  const [password : string , setPassword] = useState(initialState: "");
  const [email : string , setEmail] = useState(initialState: "");

  const navigate : NavigateFunction = useNavigate();
```

1

```
useEffect( effect: () : void => {
  // 이미 로그인 한 상태면 마이페이지로 이동
  if(auth) {
    navigate(`memberDetail/${localStorage.getItem(key: "id")}`);
  }
}, deps: []); // 컴포넌트가 처음 렌더링될 때 실행
```

2

- 1 로그인에 필요한 정보를 받기위해 useState를 이용해 password,email 을 만든다.
또한 로그인 후 마이페이지로 이동하기 위해 navigate를 선언해둔다.
- 2 페이지를 열었을 때, 이미 로그인 한 상태라 auth 값이 존재한다면 로그인 할 때 저장된 localStorage의 id 값을 받아와 마이페이지로 이동한다.

```

const login = async (e) : Promise<void> => {
  e.preventDefault();

  const data : {email: string, password: string} = { email, password };

  try {
    const response : AxiosResponse<any> = await axios.post(
      url: "http://localhost:8080/member/login",
      data
    );
  }

  if (response.status === 200 || response.status === 201) {
    alert("로그인 성공");
    const user_role = response.data.role;

    localStorage.setItem("CL_access_token", response.data.token);
    localStorage.setItem("email", response.data.email);
    localStorage.setItem("id", response.data.id);
    localStorage.setItem("role", response.data.role);
    localStorage.setItem("name", response.data.name);
    setCookie( name: "loggedIn", value: true);

    setAuth(response.data.email);
    setHeaders({ "Authorization": `Bearer ${response.data.token}` });

    if (user_role === "ADMIN") {
      navigate("/admin");
    } else {
      navigate(`/_memberDetail/${response.data.id}`);
    }
  } else {
    alert("로그인 실패: " + response.data.message);
  }
} catch (error) {
  alert(error.response?.data?.error || error.response?.data || "로그인 실패");
  console.error("로그인 오류:", error);
}
};

```

- 1 email과 password 값을 data로 묶어 member/login에 post 한다.
- 2 만약 response가 정상적으로 들어왔다면 localStorage에 토큰값, 이메일, 아이디, 권한, 이름을 넣는다. 그리고 세션 유지를 위하여 쿠키 또한 설정한다.
Auth와 Headers도 설정해준다.
- 3 만약 user의 권한이 ADMIN이라면 관리자 페이지로, 아니라면 마이페이지로 이동시킨다.
- 4 Response.status 값이 200 또는 201 이 아니라면 로그인 실패, 그리고 오류가 났을때도 로그인 실패를 띄운다.

MemberUpdate.js

```
const updateMember = async (e) : Promise<void> => { Show usages
  e.preventDefault();

  const data :{} = { name, tel, password, postcode, address, address_detail, profile: file ? true : !!profilePath };
  try {
    const response : AxiosResponse<any> = await axios.put( url: "http://localhost:8080/member/update", data, config: { headers });
    if (file) await uploadFile();
    if (response.status === 200 || response.status === 201) {
      setProfileChange(true);
      alert("회원 정보 수정 성공");
      navigate(-1);
    } else {
      alert("회원 정보 수정 실패");
    }
  } catch (error) {
    alert("서버 오류 발생");
  }
};
```

1

```
const uploadFile = async () : Promise<void> => { Show usages
  if (!file) return;
  const formData = new FormData();
  formData.append( name: "file", file);
  try {
    await axios.post( url: "http://localhost:8080/member/upload", formData, config: { headers });
  } catch (error) {
    alert("파일 업로드 실패: " + error);
  }
};
```

2

- 1 유저가 정보를 업데이트 할 경우
setProfileChange를 true 로 하여 상위 컴포넌트
에 정보를 전달한다. 이로써 헤더에서도 프로필 사
진이 바뀐 것이 적용된다.
- 2 유저가 파일을 업로드 할 경우 formData 형식에
집어넣어 post 한다.

● ActivitiesArea.js

```
<div className="tap-btn" onClick={() : void => setActiveTab( value: 0)}>
  <ul className="bar-wrap">
    <li className="bar"></li>
    <li className="bar"></li>
  </ul>
</div>
{activeTab === 0 && <ReviewArea />
<div className="title-area">
  <p className="title">
    Review
  </p>
</div>
```

1

```
[activeTab === 0 && <ReviewArea />
[activeTab === 1 && <CommentArea/>]
[activeTab === 2 && <ReservationArea/>]
```

2

```
{user.id == localStorage.getItem( key: "id" ) ? <li
  <section className="act">
    <div className="tap-btn" onClick={() : void => setActiveTab( value: 1)}>
      <ul className="bar-wrap">
        <li className="bar"></li>
        <li className="bar"></li>
      </ul>
    </div>
```

3

- 1 onClick={() : void => setActiveTab(0)} 을 이용하여 활성화 되어잇는 탭이 무엇인지 설정한다.
- 2 activeTab의 값에 따라 어떠한 Area 컴포넌트를 보여줄지 정해진다.
- 3 이러한 삼항연산자를 이용해, 현재 로그인한 회원 자신의 페이지가 아닐 경우, 댓글과 예약내역은 보이지 않게 가린다.

ReviewArea.js

```
<Swiper  
  modules={[Autoplay]}  
  className='review-swiper'  
  spaceBetween={30}  
  slidesPerView={1}  
  autoplay={{  
    delay: 3000,  
    disableOnInteraction: true,  
  }}  
>  
  
{reviews.map((review, index : number) => (  
  <SwiperSlide key={`${review.id}-${index}`}>  
    {' '}  
    {/* 고유한 key 값 설정 */}  
    <Link to={`/review/${review.reviewId}`}>  
      <div className="image-area">  
        <img  
          src={  
            review.reviewImages && review.reviewImages.length > 0  
              ? review.reviewImages[0].filePath  
              : '/images/review/noimage.png'  
          }  
          alt="썸네일"  
        />  
      </div>  
      <div className="txt-area">  
        <p className="camp-name">{review.campName}</p>  
        <p className="review-title">{review.reviewTitle}</p>  
        <div className="review-date">{review.createdDate}</div>  
      </div>  
    </Link>  
  </SwiperSlide>  
)})  
</Swiper>
```

- 1 SpaceBetween={30} 은 각 슬라이드 사이 거리를 30 준다는 뜻,
slidesPerView={1} 는 각 뷰마다 슬라이드를 한 개 보여준다는 뜻,
autoplay의 delay 값은 3초마다 슬라이드가 재생되며 disableOnInteraction은 사용자가 조작할 시 멈춘다는 뜻이다.
- 2 review.map을 이용해 각 리뷰를 SwiperSlide에 넣어서 하나의 슬라이드로 만든다.

ReservationArea.js

```
// 상태 관리: 현재 페이지와 한 페이지당 표시할 항목 수
const [currentPage : number, setCurrentPage] = useState(initialState: 1);
const itemsPerPage : number = 3; // 한 페이지에 보여줄 개수

// 총 페이지 수 계산
const totalPages : number = Math.ceil(reservations.length / itemsPerPage);

// 현재 페이지에 해당하는 데이터 추출
const indexOfLastItem : number = currentPage * itemsPerPage;
const indexOfFirstItem : number = indexOfLastItem - itemsPerPage;
const currentReservations : any[] = reservations.slice(
  indexOfFirstItem,
  indexOfLastItem
);
```

1

2

```
{currentReservations.map((reservation) => (
  <tr key={reservation.id}>
    <td>{reservation.campName}</td>
    <td>
      {new Date(reservation.reservationDate).toLocaleDateString()}
    </td>
    <td>
      {new Date(reservation.usageDate).toLocaleDateString()}
    </td>
    <td>
      <button
        className={`btn btn-${reservation.hasReview ? 's' : 'p'}
        }-f btn-xsm mlr-a`}
      >
        <Link
          to={`/review/write`}
          state={{memberId: reservation.memberId,
            campId: reservation.campId,
            reservationId: reservation.id,
            reservationDate: reservation.reservationDate,
          }}
        >
          {reservation.hasReview ? '후기 보기' : '후기 작성'}
        </Link>
      </button>
    </td>
  </tr>
))}
```

3

- 1 currentPage 및 itemsPerPage 를 선언해준다.
- 2 현재 페이지에 해당하는 예약만 currentReservations에 넣어준다.
- 3 currentReservations.map 을 이용해 각 예약을 리스트로 만들어 보여준다.

```
// 페이지 변경 함수
const goToNextPage = () : void => { Show usages
    if (currentPage < totalPages) setCurrentPage( value: currentPage + 1);
};

const goToPrevPage = () : void => { Show usages
    if (currentPage > 1) setCurrentPage( value: currentPage - 1);
};
```

1

```
<button
    className="fs_md"
    onClick={goToPrevPage}
    disabled={currentPage === 1}
>
    &lt;
</button>
<p className="fs_md">
    &nbsp;&nbsp; {currentPage} / {totalPages} &nbsp;&nbsp;
</p>
<button
    className="fs_md"
    onClick={goToNextPage}
    disabled={currentPage === totalPages}
>
    &gt;
</button>
```

2

- 1 currentPage를 변경하는 함수를 만든다.
- 2 페이지네이션 버튼을 이용해 current Page를 바꾸면 자동적으로 현재 currentReservations 도 바뀌며 페이지에 맞는 예약 리스트가 나오게 된다.

```
@Bean no usages new *
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    return http
        .httpBasic(httpBasic -> httpBasic.disable())
        .csrf(csrf -> csrf.disable())
        .cors(cors -> cors.configurationSource(corsConfigurationSource))
        .authorizeHttpRequests(authorize
            -> authorize
            1 .requestMatchers(
                ...patterns: "/*", "/*/*", "/admin/**", "/app/**", "/camp/**", "/chat/**"
                , "/member/login", "/member/auth/google", "/review/**", "/topic/**", "/uploads/**"
            ).permitAll()
            .requestMatchers(...patterns: "/admin/**").hasRole("ADMIN")
            .requestMatchers(...patterns: "/camp/**").hasAnyRole(...roles: "ADMIN", "USER")
            2 .requestMatchers(...patterns: "/chat/**").hasAnyRole(...roles: "ADMIN", "USER")
            .requestMatchers(...patterns: "/member/**").hasAnyRole(...roles: "ADMIN", "USER") // 로그인 제외, 나머지는 인증 필요
            .requestMatchers(...patterns: "/memberDetail/**").hasAnyRole(...roles: "ADMIN", "USER")
            .requestMatchers(...patterns: "/review/**").hasAnyRole(...roles: "ADMIN", "USER")
            .anyRequest().authenticated()
        )
        3 .sessionManagement(session -> session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
        4 .exceptionHandling(excep -> excep.authenticationEntryPoint(jwtAuthenticationEntryPoint))
        .addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class)
        .build();
}
```

- 1 특정 경로에 대한 접근 정책 설정 : 로그인 관련 경로(member/login, member/auth/google)는 permitAll로 인증 없이 접근 가능하게 설정한다.
- 2 특정 경로에 대한 접근 정책 설정 : admin 페이지는 ADMIN만, 그 외의 페이지는 ADMIN 혹은 USER만 접근 가능하도록 설정한다.
- 3 JWT 기반 Stateless 인증을 사용해서 세션 관리 안하도록 설정한다.
- 4 JWT 인증 필터(JwtAuthenticationFilter)를 UsernamePasswordAuthenticationFilter 전에 실행한다..

```
@Override no usages new*
protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)
    throws ServletException, IOException {
    Thread currentThread = Thread.currentThread();
    log.info("현재 실행 중인 스레드: " + currentThread.getName());

    // get token
    1 String header = request.getHeader(HEADER_STRING); //헤더에서 키값이 Authorization인 헤더를 가져온다
    String username = null;
    String authToken = null;
    Ctrl+L to Chat, Ctrl+I to Command
    //토큰이 있는 경우
    2 if (header != null && header.startsWith(TOKEN_PREFIX)) {
        authToken = header.replace(TOKEN_PREFIX, replacement: " ");
        try {
            username = this.jwtTokenUtil.getUsernameFromToken(authToken); //JWT로 부터 사용자 아이디(username)를 추출
        } catch (Exception e) {
            log.error("JWT 토큰 추출 실패: " + e.getMessage());
        }
    }
}
```

- 1 request 객체의 키 값이 Authorization인 값을 가져온다.
- 2 JWT로 부터 사용자 아이디를 추출하고 검증한다.

```
if (username != null && SecurityContextHolder.getContext().getAuthentication() == null) { //아직 인증되지 않은 상태일 때  
    //log.info(SecurityContextHolder.getContext().getAuthentication().getName()); //현재 사용자 아이디  
    UserDetails userDetails = this.userDetailsService.loadUserByUsername(username); //DB에서 사용자 정보 조회  
    1 if (this.jwtTokenUtil.validateToken(authToken, userDetails)) { //JWT가 유효한지 확인  
        //인증 토큰(권한 정보 가짐)  
        2 UsernamePasswordAuthenticationToken authenticationToken =  
            new UsernamePasswordAuthenticationToken(userDetails, credentials: null, userDetails.getAuthorities());  
  
        3 authenticationToken  
            .setDetails(new WebAuthenticationDetailsSource().buildDetails(request));  
        log.info("authenticated user " + username + ", setting security context");  
        SecurityContextHolder.getContext().setAuthentication(authenticationToken); //매우 중요 : SecurityContextHolder
```

1 아직 인증되지 상태인 경우에 request 객체로부터 추출한 JWT 토큰이 유효한지 확인한다.

2 사용자 권한 정보
(userDetails.getAuthorities())를 통해서 사용자 인증 객체에 정보를 담는다.

3 Spring Security의 SecurityContext에 인증 정보를 저장한다.
이후 요청에서는 인증된 사용자로 인식한다.

```
localStorage.setItem("CL_access_token", response.data.token);
localStorage.setItem("email", response.data.email);
localStorage.setItem("id", response.data.id);
localStorage.setItem("role", response.data.role);
localStorage.setItem("name", response.data.name);
```

2 const [headers, setHeaders] = useState({
 "Authorization": `Bearer \${localStorage.getItem("CL_access_token")}`
});

3 await axios.delete(`http://localhost:8080/admin/camp/\${campId}/delete`, {headers: headers})

- 1 로그인 완료시 localStorage에 사용자 정보를 저장한다.

localStorage는 브라우저의 로컬 저장소로 페이지가 새로고침되더라도 기존에 저장된 토큰이 유지된다.

- 2 headers라는 상수를 이용해서 HTTP 요청 시 사용할 헤더에 JWT 토큰을 포함한 정보를 저장한다.

- 3 HTTP 요청시 headers 상수를 이용해서 요청에 인증 헤더를 포함하여 요청한다.

```
@Configuration no usages new *
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

    @Override no usages new *
    2 public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint( ...paths: "/chat")
            .setAllowedOrigins("http://localhost:3000")
            .withSockJS(); // React 클라이언트와 연결
    }

    @Override no usages new *
    3 public void configureMessageBroker(MessageBrokerRegistry registry) {
        registry.enableSimpleBroker( ...destinationPrefixes: "/topic");
        registry.setApplicationDestinationPrefixes("/app");
    }
}
```

- 1 WebSocket 메시지 브로커(클라이언트의 메시지를 구독자에게 전달하는 역할) 활성화함으로써, 클라이언트 메시지 송수신을 가능하게 한다.
- 2 클라이언트가 WebSocket에 연결할 엔드포인트를 설정한다.
<http://localhost:3000>에서 오는 요청을 허용한다.
- 3 구독자가 메시지를 받을 경로를 지정하고, 클라이언트가 서버로 메시지를 보낼 때 사용할 경로를 설정한다.

```
1 const socket = new SockJS(`http://localhost:8080/chat?token=${token}`);
const client = new Client({
  webSocketFactory: () => socket,
  debug: (str) => console.log(str),
  reconnectDelay: 5000,
});

client.onConnect = () => {
  console.log("WebSocket 연결 성공!");
2 client.subscribe(`/topic/messages/${userId}/${selectedChat}`, (message) => {
  const newMessage = JSON.parse(message.body);
  setMessages((prevMessages) => {
    const isDuplicate = prevMessages.some(
      (msg) => msg.content === newMessage.content && msg.member1Id === newMessage.member1Id
    );
    return isDuplicate ? prevMessages : [...prevMessages, newMessage];
  });
};

3 client.onWebSocketClose = () => {
  console.error(" WebSocket 연결이 종료되었습니다. 다시 연결 시도 중...");
  setTimeout(() => client.activate(), 5000);
};

4 client.activate();
```

- 1 SockJS를 사용하여 서버(http://localhost:8080/chat)에 웹소켓 연결한다.
- 2 특정 채팅방의 메시지를 구독(subscribe) 한다.
- 3 client.onWebSocketClose()에서 연결이 끊어지면 자동 재연결한다.
- 4 WebSocket 연결을 생성하고, STOMP 프로토콜을 사용하여 서버와 통신을 시작 한다.

```
1 stompClient.publish({
  destination: "/app/send",
  body: JSON.stringify(chatMessage),
});

2 @MessageMapping("/send") new *
public ChatResponseDto sendMessage(ChatRequestDto message) {
  return chatService.sendMessage(message);
}

3 public ChatResponseDto sendMessage(ChatRequestDto messageDto) { new *
  // DTO → Entity 변환
  ChatMessage savedMessage = chatMessageRepository.save(messageDto.toEntity());

  // Entity → DTO 변환
  ChatResponseDto responseDto = ChatResponseDto.fromEntity(savedMessage);

  // 메시지를 양방향으로 전송
  messagingTemplate.convertAndSend(destination: "/topic/messages/" + responseDto.getMember1Id() + "/" + responseDto.getMember2Id(), responseDto);
  messagingTemplate.convertAndSend(destination: "/topic/messages/" + responseDto.getMember2Id() + "/" + responseDto.getMember1Id(), responseDto);

  return responseDto;
}

4 client.subscribe(`/topic/messages/${userId}/${selectedChat}`, (message) => {
  const newMessage = JSON.parse(message.body);
  setMessages((prevMessages) => {
```

- 1 /app/send를 구독하고 있는 WebSocket 엔드포인트로 메시지를 보낸다.
- 2 @MessageMapping 어노테이션을 통해 /app/send 경로의 요청을 받고, ChatService의 sendMessage 메서드를 호출한다.
- 3 채팅 메시지 DB에 저장한다. 보낸 사람과 받는 사람 모두에게 실시간으로 메시지를 전송한다.
- 4 React에서 메시지를 수신하고, 해당 메시지를 화면에 표시한다.

프로젝트 회고



김성우

프로젝트 팀장(PM)

디자인 및 퍼블리싱

프론트엔드 총괄

백엔드 총괄 및 DB설계

Admin(관리자) 기능 개발

Open API 연동

Query DSL 연동

캠핑장 게시판/게시글 CRUD 개발

예약 기능 개발

회고

이번 팀 프로젝트에서 저는 팀장이자 디자인, 퍼블리싱, 프론트엔드, 백엔드 모든 분야의 총괄을 맡았습니다. 강사님과 팀원들의 과분한 신뢰를 받았기에 이를 증명해 보이고 싶은 마음과 동시에, 제가 잘 해내지 못하면 프로젝트 전체에 악영향을 미칠 것이라는 부담감을 안고 한 달간의 과제를 수행했습니다. 그동안 배웠던 내용을 복습하고 응용하며 웹 개발과 데이터의 흐름에 대해 더 깊게 이해하고, 개발 언어를 체득 한 것도 물론 중요한 소득이었지만, 이 프로젝트를 통해 제가 깨달은 것은 크게 두 가지로 소통과 팀워크의 중요성입니다.

[소통의 중요성]

프로젝트를 시작하기 앞서 저는, 그간 열심히 배우고 공부한 내용을 토대로 프론트엔드, 백엔드 기능을 만들고, 퍼블리셔 경력을 살려 웹기획, 디자인, 퍼블리싱을 해내는 것은 자신이 있었습니다. 때문에 그저 배운것을 껌데기만 바꿔서 재현하는 것 보다 한 단계 나아가고 싶다는 욕심이 있었고, 작업을 진행하면서 새로운 코드와 기술을 공부하며 남들과는 다른 프로젝트를 만드는 것을 목표로 정했습니다.

의도는 좋았으나, 시간이 갈수록 처음 기획했던 것보다 프로젝트의 볼륨이 자꾸 커져 팀원들의 작업 부담이 늘어갔고, 처음 하는 팀 단위 개발이라 그런지 예상하지 못한 이슈가 종종 발생해 일정은 자주 딜레이되었습니다.

이 과정에서 저는 팀장으로서 팀원 개개인의 부담감, 역량 차이, 작업 이해도를 세심하게 파악하지 못했고, 이런 소통의 부재는 결국 1차 작업 취합 시점에 이르러, 여러 기능이 정상적으로 작동하지 않는 상황을 낳았습니다.

그동안 정기적으로 회의를 진행하긴 했지만, 돌이켜보면 이는 팀원들과의 소통을 위한 것이 아니라, 제가 의견을 전달하기 위한 방식에 불과했습니다. 팀원들이 어떤 부분에서 어려움을 겪고 있는지 제대로 파악하지 못한 채 진행한 회의였기 때문에, 실질적인 해결책이 되지 못했던 것입니다.

이러한 문제를 인식한 후, 저는 팀장으로서의 역할을 다시 고민하기 시작했습니다. 먼저, 기존의 코드를 전부 분석하며 팀원들이 어려움을 겪고 있는 부분을 파악했습니다. 이후 매일 팀원 개개인의 작업 현황을 확인하고, 막힌 부분을 지원하는 방식으로 소통을 강화했습니다. 또한, 회의의 목적을 단순한 의견 전달이 아닌, 서로의 작업을 공유하고 지원하는 방향으로 전환했습니다.

그 결과, 이후 작업은 더욱 원활하게 진행되었으며, 큰 이슈 없이 프로젝트를 마무리할 수 있었습니다. 이를 통해 저는 팀 단위 프로젝트에서 원활한 소통이 얼마나 중요한지를 실감할 수 있었습니다.

[팀워크의 중요성]

처음 프로젝트를 시작할 때, 저는 강한 책임감을 가지고 개발 외의 모든 부분까지 맡고자 했습니다. 기획, 문서 작성, 코드 취합 및 관리 등 가능한 모든 작업을 제가 처리해야 한다고 생각했습니다. 그러나 이미 맡고 있던 개발 업무가 많았기에, 다른 작업까지 신경 쓰다 보니 개발에 집중할 시간이 부족했습니다. 반대로, 개발에 집중하면 비개발 업무가 소홀해지는 악순환이 반복되었습니다.

결국 무리한 일정과 과한 책임감으로 인해 과부하 상태에 빠졌고, 어느 쪽도 원하는 완성도에 도달하지 못하는 한계를 경험했습니다. 이때 강사님의 조언과 팀원들의 자발적인 지원이 큰 도움이 되었습니다. 팀원들은 코드 취합 및 일부 작업을 맡겨달라고 제안했으며, 이를 통해 저는 역할 분담의 중요성을 깨닫게 되었습니다.

혼자서 모든 걸 해내려는 것은 오만한 생각이었으며, 팀원들과 협력했을 때 일이 훨씬 수월하게 진행된다는 것을 체감했습니다. 결국 팀 프로젝트에서 중요한 것은 개인의 역량이 아니라, 팀원들과의 협력을 통해 최상의 결과를 만들어내는 것이라는 점을 배우게 되었습니다.

이번 [캠핑라운지] 프로젝트는 저에게 단순히 그동안의 성취를 평가받기 위한 과제가 아닌, 개발자로서 한 단계 성장할 수 있는 소중한 경험이 되어주었습니다.

이 과정을 통해 현재의 제 역량을 객관적으로 확인할 수 있었고, 프로젝트 작업은 단순히 코드를 잘 만드는 것이 다가 아니라는 걸 깨달아 앞으로 어떤 노력을 기울여야 할지에 대한 방향성을 찾을 수 있었습니다. 또한, 팀의 일원으로서 어떤 마음가짐을 가져야 하는지도 배울 수 있었습니다.

앞으로도 이러한 경험을 바탕으로 더욱 성장하여, 팀에 꼭 필요한 개발자가 될 수 있도록 노력하겠습니다.



양찬식

프로젝트 리더(PL)

프론트엔드, 백엔드 개발

인증시스템, JWT 개발

WebSocket 연동

채팅 기능 개발

DB 설계

[프로젝트 주제와 설계의 중요성]

프로젝트를 진행하면서 프로젝트 주제와 테이블 설계의 중요성을 실감했다.

프로젝트의 주제를 선정하는 과정에서는 처음에 다양한 의견이 나왔고, 공공 API 연동과 최근 사람들의 관심사를 결합한 주제를 찾는 데 시간이 좀 걸렸는데, 다행히 팀원들과 충분히 의견을 나누는 시간을 가지면서 적절하고 흥미로운 주제를 선택할 수 있었다. 일단 프로젝트 명확히 주제가 정해지고, 그 주제가 흥미로운 덕분에 그 후는 순조롭게 진행됐던 것 같다.

또한 처음 설계한 구조에서 몇 가지 변화가 있었지만, 테이블이 완성되고 ERD를 작성하면서 프로젝트의 큰 그림이 명확해졌다.

프로젝트 주제와 DB 설계 등이 끝나니까 ERD, UML, 클래스 다이어그램 등 문서 작성이 이루어 졌는데, 이를 통해 프로젝트 주제 선정과 DB 설계 하는 것이 얼마나 중요한지 깨달았다.

[아쉬운 점]

리액트 컴포넌트를 더 작은 단위로 나누었다면, 유지보수가 훨씬 용이했을 것 같다는 점이 아쉬운 부분이다.

또한 웹소켓을 활용한 실시간 채팅 기능을 개발할 때, 기능 구현에만 집중한 나머지 예상보다 많은 실수가 발생했다.

결과적으로 개발 기간이 일주일 정도 더 되었는데, PL로서 같이 하기로 한 다른 기능을 팀원들에게 맡기는 것이 너무 미안했다.

또한 개발자로서, 앞으로는 시간에 쫓기지 않고 좀 더 여유를 가지고, 테이블 설계와 화면 설계서를 기반으로 개발을 진행하는 것의 중요성을 더욱 실감할 수 있었다.

[발전한 점]

프로젝트를 진행하면서 스프링부트, JPA, 리액트에 대한 이해도가 더 깊어졌고, 수업 시간에 이해하지 못했던 부분들이 뒤늦게 풀리는 경험도 있었다.

특히, 리액트에 대한 관심이 커졌고, 컨트롤러에서 요청과 응답에 사용할 DTO를 ENTITY로 변환해 사용하는 방식은 처음에는 이해할 수 없었으나, DB 연결에 있어 이점이 있다는 것을 깨닫고 JPA에 대한 흥미도 더 생겼다.

[느낀 점]

프로젝트 주제 선정부터 진행, 완료 단계까지 가장 중요한 것은 팀원 간의 원활한 의사소통과 팀워크였다고 생각한다.

일주일에 최소 두 번 이상 회의를 하면서 각자 하고 싶은 것들과 다양한 의견을 나누었고, 그 의견들이 나와는 너무 달라서 신기하고 흥미로웠다.

의견을 나누는 것뿐만 아니라, 개발 진행 중에 막힌 부분을 팀원에게 도움을 청하고 함께 해결 방법을 찾는 과정에서 내가 놓쳤던 부분을 발견하며 뿌듯한 경험을 할 수 있었다.

이번 프로젝트는 수업 시간에 배웠던 프로젝트 문서 작성과 개발 언어를 다시 다루면서 깊이 있는 공부가 되었다는 점에서 매우 의미가 있었다.

또한 팀 프로젝트이기 때문에 의사소통의 중요성을 실감했고, 팀원들에게 많은 도움을 받았기에 정말 감사하고 행운이었다고 느낀다.

한 달 동안 힘들기도 했고 아쉬운 점도 있었지만, 그만큼 많이 성장할 수 있었던 계기가 되었다.



손수용

- 프론트엔드, 백엔드 개발
- 맴버 파트 개발
- 인증시스템 연동
- 소셜로그인(Google API) 연동
- 토스 API 테스트용 결제 연동
- DB 설계

회고

[작업 전] 2025년 2월 초부터 3월 초까지 약 한 달간 진행한 팀 프로젝트 "캠핑라운지"는 프로그래밍 학원에서 배운 내용을 실제 환경에 적용하고 복습하여 보다 깊이 있게 이해하는 것과 팀원과의 효율적인 소통 및 협업 능력을 키우는데 목표를 두고 시작하게 되었습니다. 저는 특히 회원 관련 기능을 맡아 회원정보의 CRUD 및 상세페이지를 구현하는 역할을 수행했습니다.

"캠핑라운지"는 캠핑을 즐기고자 하는 20~40대 사용자를 타겟으로 기획된 웹사이트로, 캠핑장 예약뿐만 아니라 유저 간 1:1 채팅을 통한 캠핑장 정보 공유와 리뷰 작성 등의 기능을 제공하여 사용자가 캠핑장을 사전에 충분히 파악할 수 있도록 지원하는 데 중점을 두었습니다.

[작업 후] 프로젝트를 진행하며 여러 어려움도 겪었는데, 그중 가장 큰 난관은 프로필 사진 업로드 기능과 서로 다른 컴포넌트 간 상태 동기화 문제였습니다. 예를 들어, 마이페이지에서 프로필 사진을 변경했음에도 헤더의 프로필 사진은 변경되지 않는 동기화 문제가 발생하여 당황했던 적이 있었습니다. 분명 학원에서 다뤘던 내용이었지만 실제 구현 단계에서는 막막하게 느껴졌습니다.

이러한 문제를 해결하기 위해 저는 인터넷의 다양한 자료와 게시물을 적극적으로 활용했습니다. 프로필 사진 업로드 기능 구현 방법을 찾아 여러 방법을 테스트했고, 프로필 사진 URL을 관리하는 법, 컴포넌트 간 변수 공유를 위한 효과적인 상태 관리 기법 등을 배워 적용하여 문제를 극복할 수 있었습니다.

결과적으로 목표했던 핵심 기능들을 성공적으로 구현하였으며, 작업을 진행하면서 미처 생각하지 못했던 관리자 인증키 생성 기능과 프로필 사진 업로드 기능까지 추가로 개발하였습니다.

이번 프로젝트를 통해 "에러를 마주했을 때 당황하지 않고 에러 메시지를 분석하면 오히려 문제의 원인을 더 빠르게 파악할 수 있다"는 중요한 경험을 얻었습니다. 또한 팀원들과의 원활한 의사소통과 정확한 업무 분담이 얼마나 중요한지도 깨닫게 되었습니다. 초반에는 제가 요구사항을 잘못 이해해 불필요한 기능을 만들기도 했으나, 이후 소통을 강화하여 프로젝트의 방향성을 바로잡을 수 있었습니다.

한 달간의 짧지 않은 작업을 마치며 느낀 가장 큰 감정은 성취감이었습니다. 예상하지 못한 버그와 기능상의 문제들이 계속 나타났지만, 매번 그 원인을 찾아 해결하는 과정에서 오히려 프로젝트의 재미와 만족감을 느꼈습니다. 첫 프로젝트임에도 불구하고 팀원들과 함께 의미 있는 결과물을 만들어냈다는 점에서 큰 보람과 자신감을 얻을 수 있었던 경험이었습니다.



최수민

프론트엔드, 백엔드 개발

리뷰 게시판/게시글 CRUD 개발

댓글 CRUD 개발

React Quill 연동 글쓰기 에디터

DB 설계

이번 팀 프로젝트는 단순한 CRUD 기능을 구현하는 것으로 시작했다. 요구사항을 분석하고 기본적인 기능을 개발하는 과정은 비교적 수월했다. 하지만 프로젝트가 진행될수록 기능이 복잡해지고, 유지보수성과 확장성을 고려해야 하는 단계에 접어들면서 새로운 고민이 생겼다. 단순한 기능 구현만으로는 충분하지 않았고, 보다 체계적인 설계가 필요하다는 점이었다.

개발을 진행하면서 발목을 블집은건 바로 데이터 흐름의 이해도였다. 프론트엔드에서 요청이 들어와 백엔드에서 처리되고, 다시 데이터베이스와 상호작용한 후 응답이 반환되는 과정이 직관적으로 보이지 않는 경우가 많았다. 또한, DTO와 Entity 변환 과정에서 발생하는 오류도 골칫거리였다. 데이터를 변환하는 과정에서 필드가 누락되거나, 잘못된 값이 매핑되는 문제가 발생했다.

이를 해결하기 위해 변환 로직을 보다 심도있게 분석, 정리하면서 데이터 흐름을 더욱 직관적으로 이해하게 되었고, 이 과정에서 코드 가독성이 미치는 영향도 크게 체감했다. 기능 구현에만 집중하면서 네이밍 규칙이 제각각이거나, 복잡한 로직이 한 함수에 몰려 있는 경우가 많았다.

하지만 함수를 분리하며 유지보수성을 높이는 방식으로 개선해 나갔다. 그리고 단순히 동작하는 코드가 아니라, 동작을 하면서도 가독성이 좋은 코드가 협업에서 얼마나 중요한지를 느낄 수 있었다.

향후 개선해야 할 점도 분명해졌다. 팀원 간 지속적인 코드 리뷰로 다른 팀원이 어떤 의도를 가지고 작성한 코드인지 알 수 있었는데, 좋은 코드를 작성 하려면 다른 사람의 코드를 읽을 줄 알고, 많이 읽어봐야 한다는 말이 떠올랐다. 코드 리뷰를 통해 다른 사람의 접근 방식, 해결 방법을 이해하면서 나의 코드 작성 방식에도 변화가 생겼으며 팀원들의 다양한 문제 해결 방식을 배우고, 서로의 피드백을 반영할 수 있는 문화가 형성된 것은 프로젝트 전반에 긍정적인 영향을 미쳤다. 내 작업 방식에서 개선할 점을 발견해 더 효율적이고 가독성 높은 코드의 중요성을 알았고 이는 단순히 나의 코드 품질을 높이는 것 뿐만 아니라, 팀 내에서 더 나은 협업 환경을 구축하는 데 큰 도움이 될 것이라 생각한다.

이번 프로젝트는 단순한 기능 구현을 넘어 설계, 협업, 문제 해결 능력을 키울 수 있는 값진 경험이었다. 특히, 데이터 흐름을 이해하고 이를 서비스에 적용하는 과정에서 큰 성장을 경험했다. 처음에는 어려웠던 개념들이 점점 자연스럽게 와닿았고, 프로젝트를 진행할수록 코드 구조와 설계의 중요성을 더욱 깊이 깨닫게 되었고, 프로젝트에서 배운 것들을 바탕으로 더 나은 결과물을 만들어낼 수 있겠다는 확신이 들었다.

감사합니다