

18 DECEMBER 2024

# KKOMO 센터 PROJECT





프로젝트 목표

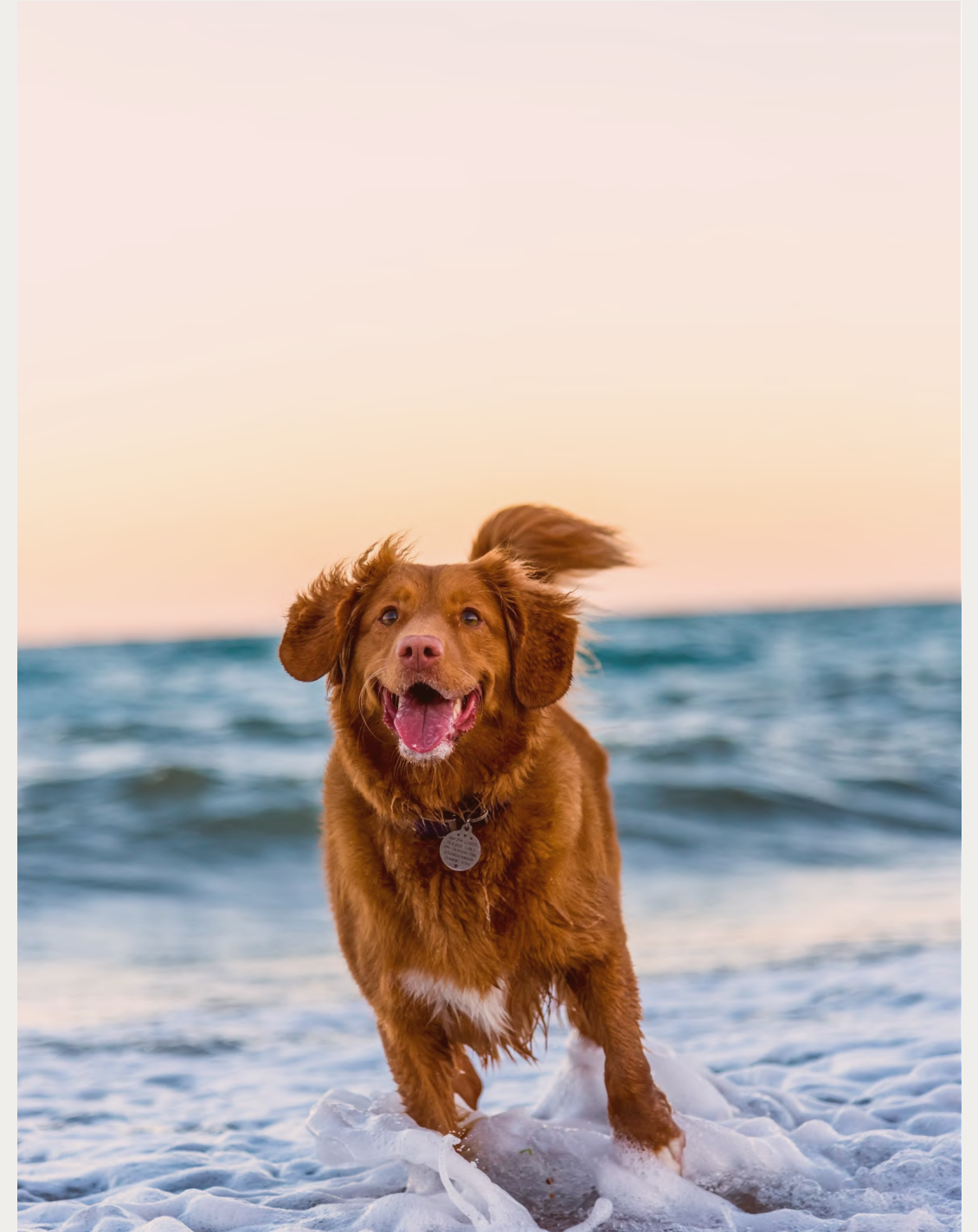
프로젝트 개발 환경

진행 과정

기술 스택 소개

결론 및 개선 사항

# 프로젝트 목표





01

## 사용자 친화적인 웹 애플리케이션 개발

직관적이고 쉽게 접근할 수 있는  
웹 애플리케이션을 구축

02

## 프론트엔드와 백엔드 연동 구현

프론트엔드와 백엔드를 연동,  
실시간 데이터 처리 및  
동적 웹사이트를 구현

03

## 데이터베이스 설계 및 최적화

효율적인 데이터베이스 설계,  
웹 애플리케이션의 성능을 최적화

# 프로젝트 개발 환경



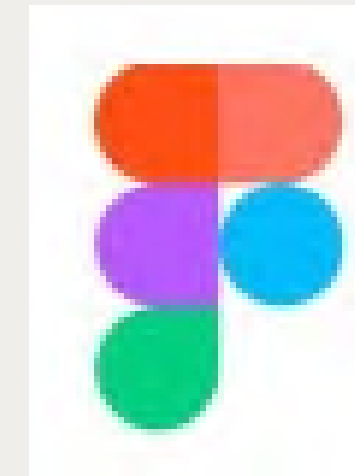
## Front



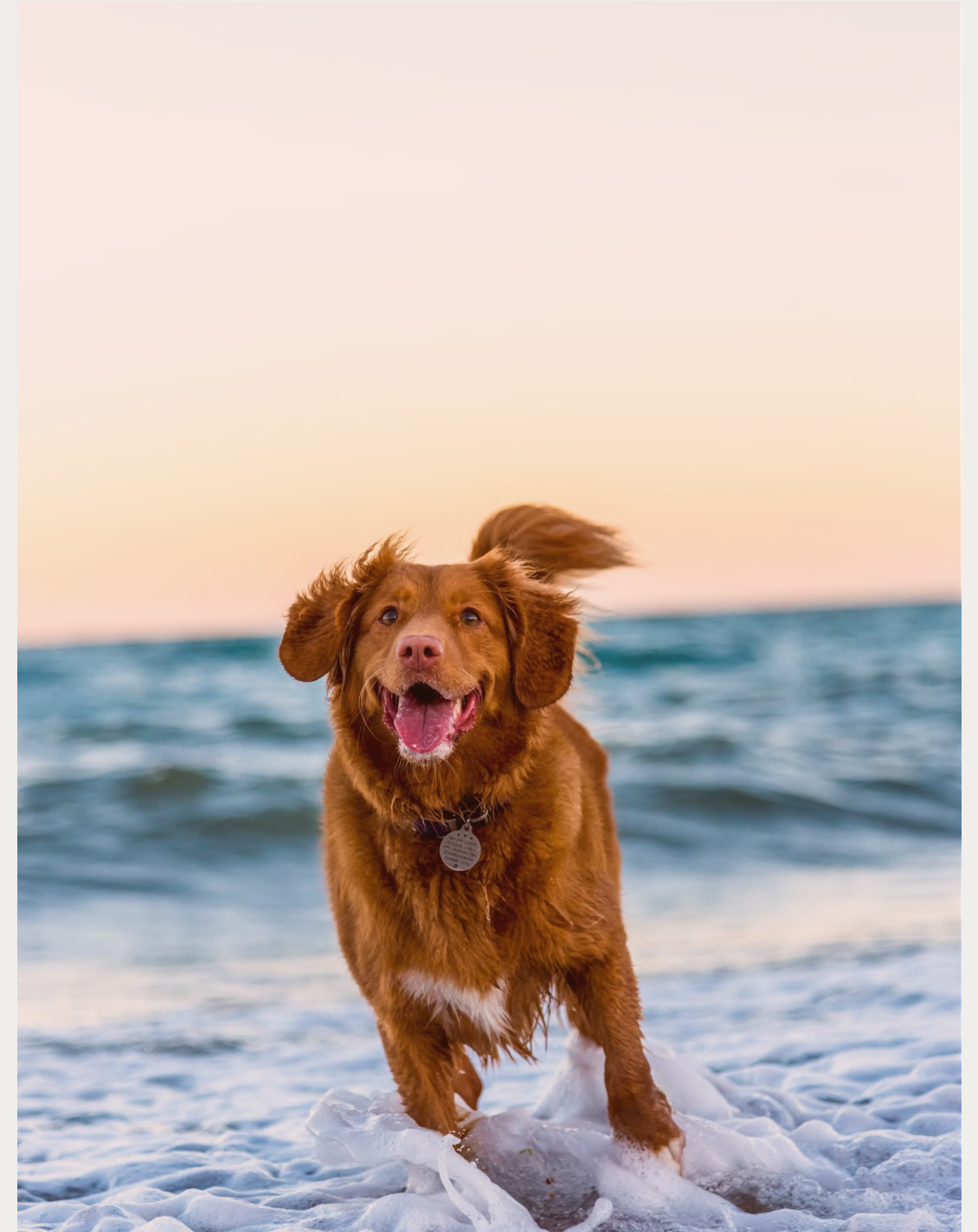
## Back



## Tools & Design



# 진행 과정

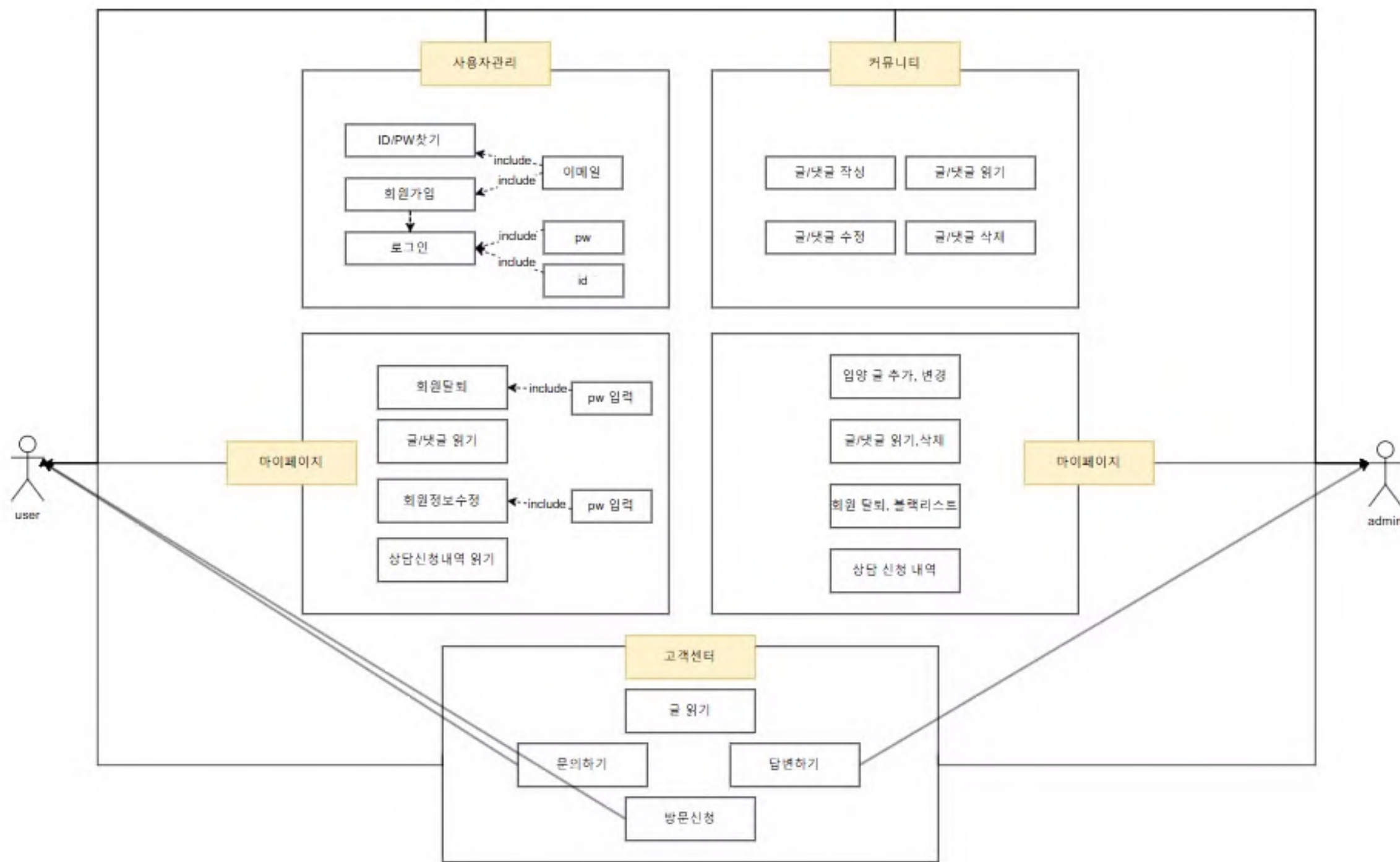


# 진행 과정

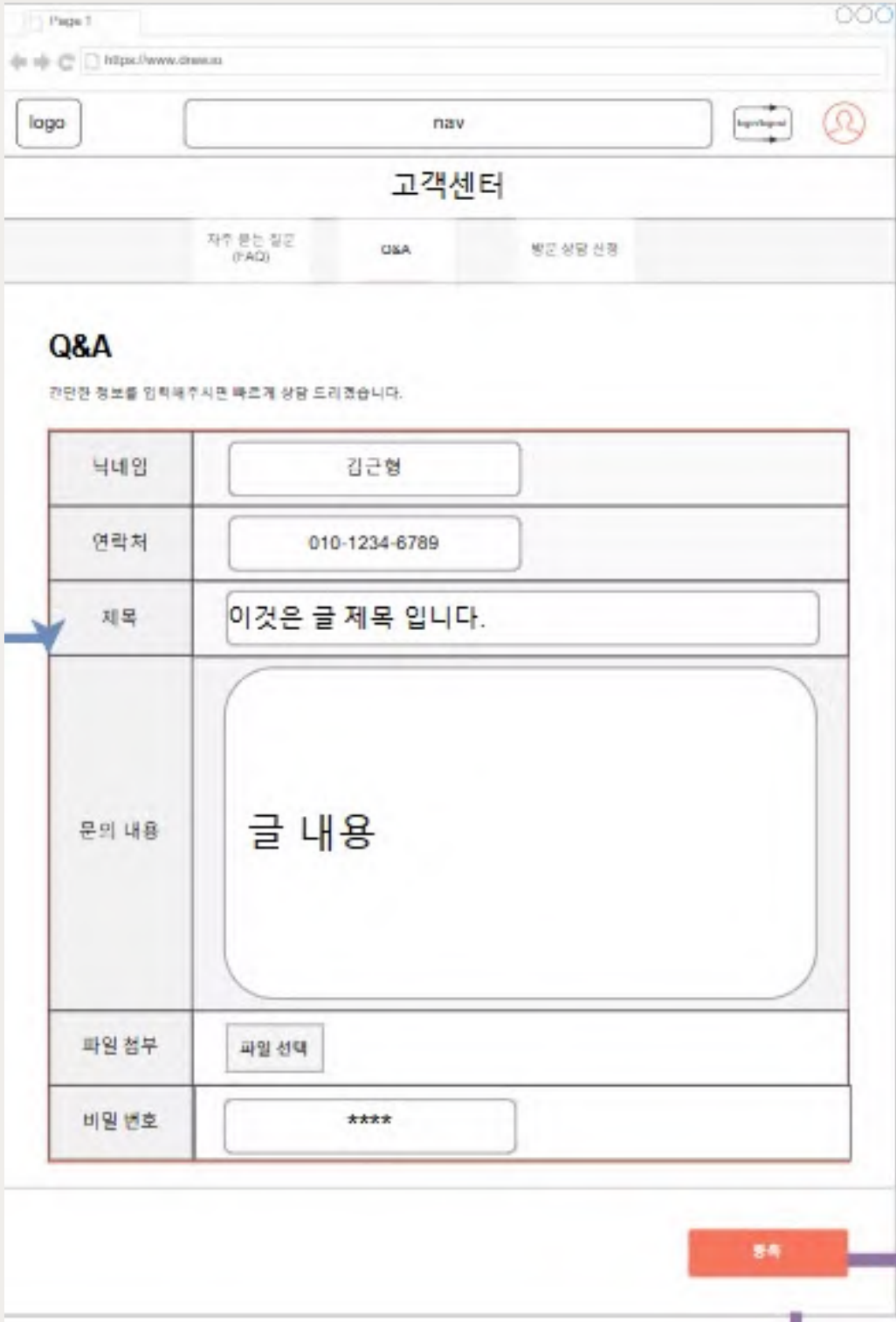
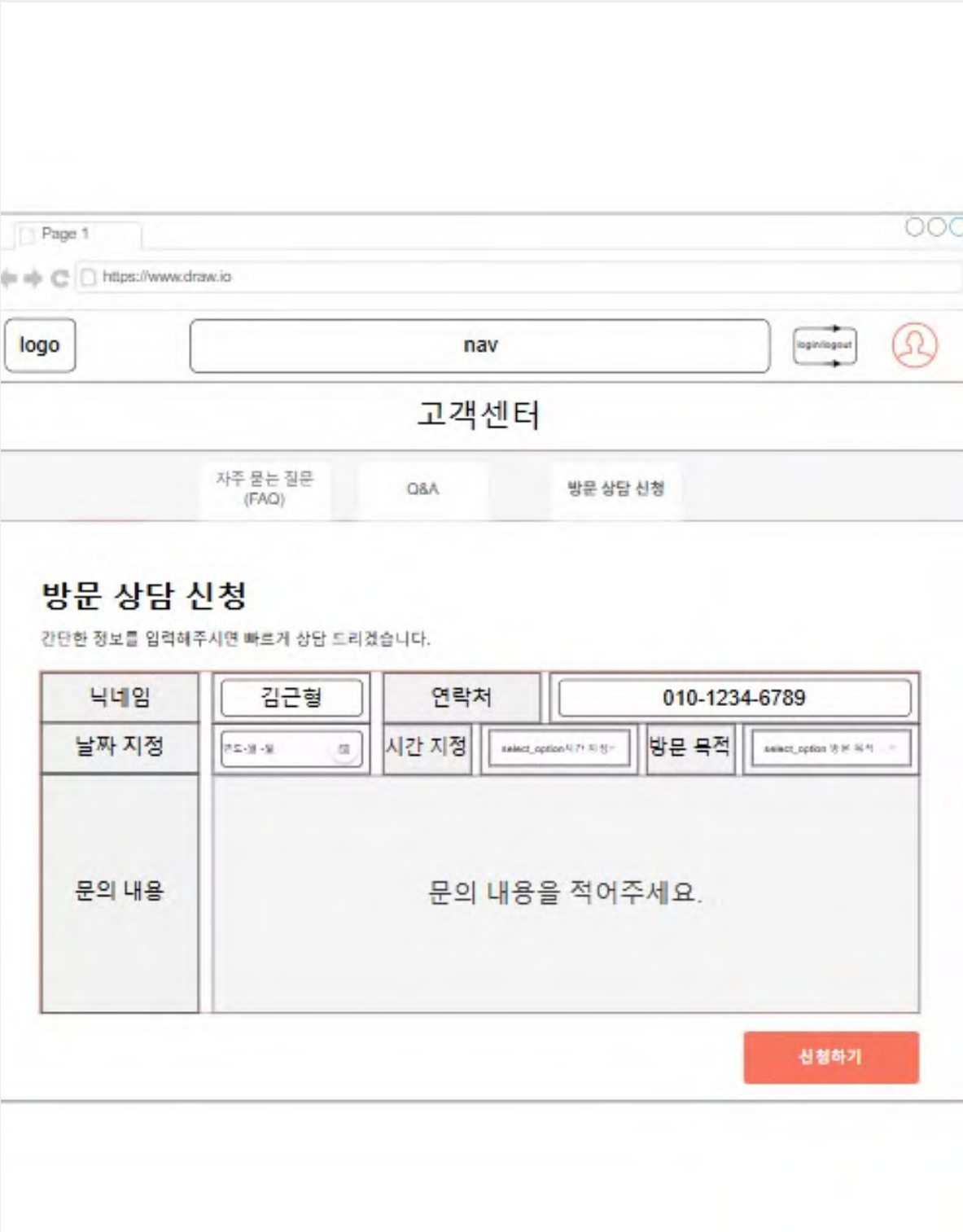


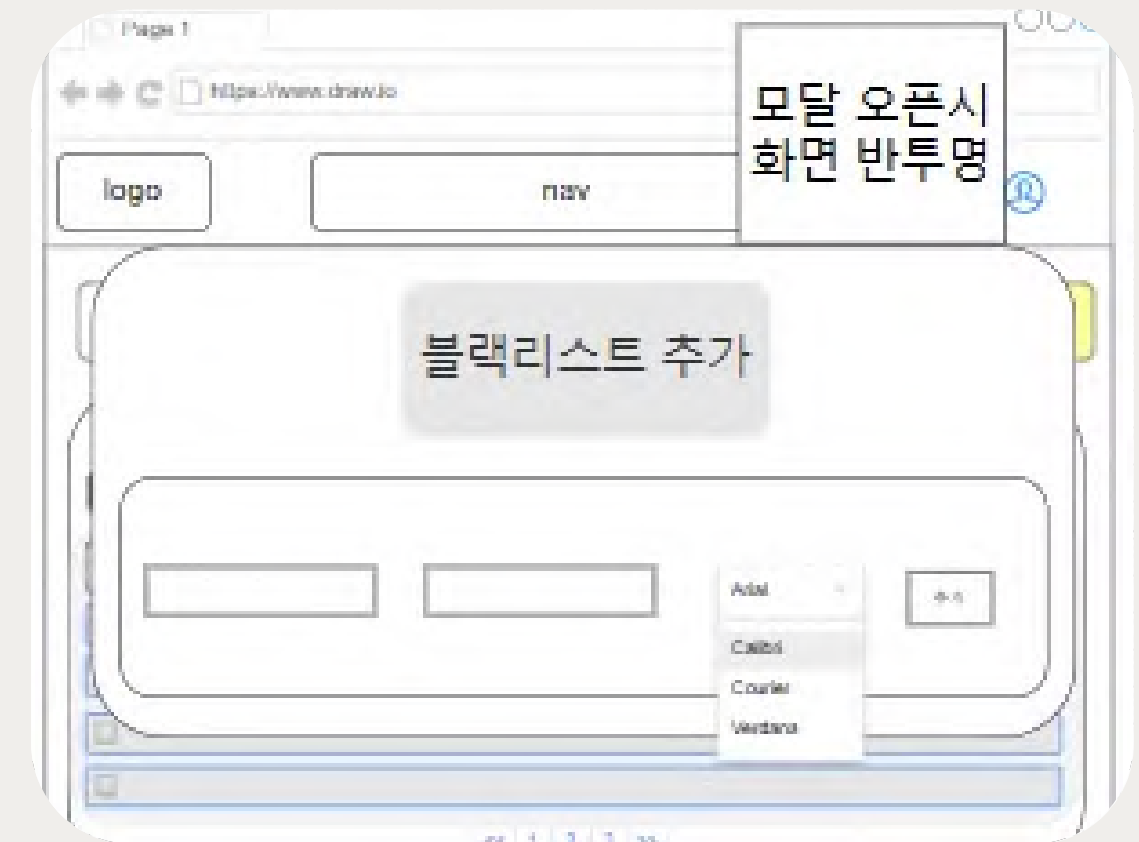
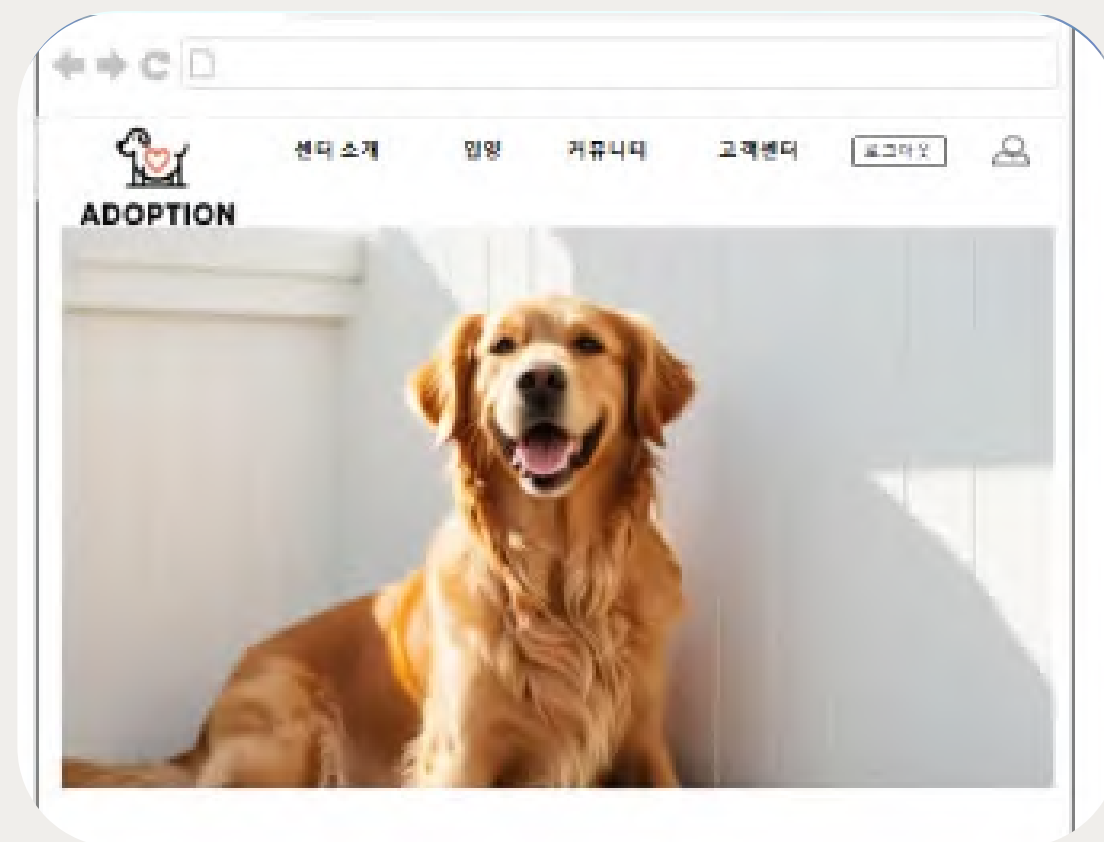
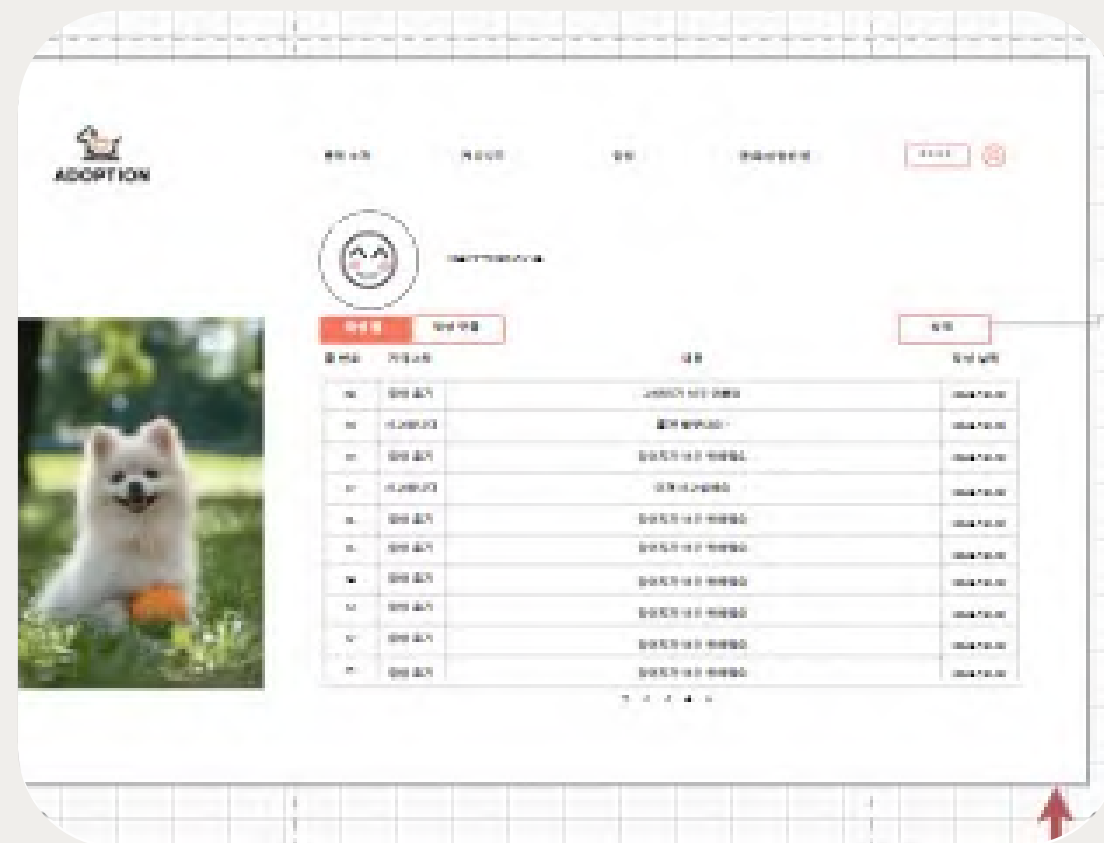


# 기획 및 설계



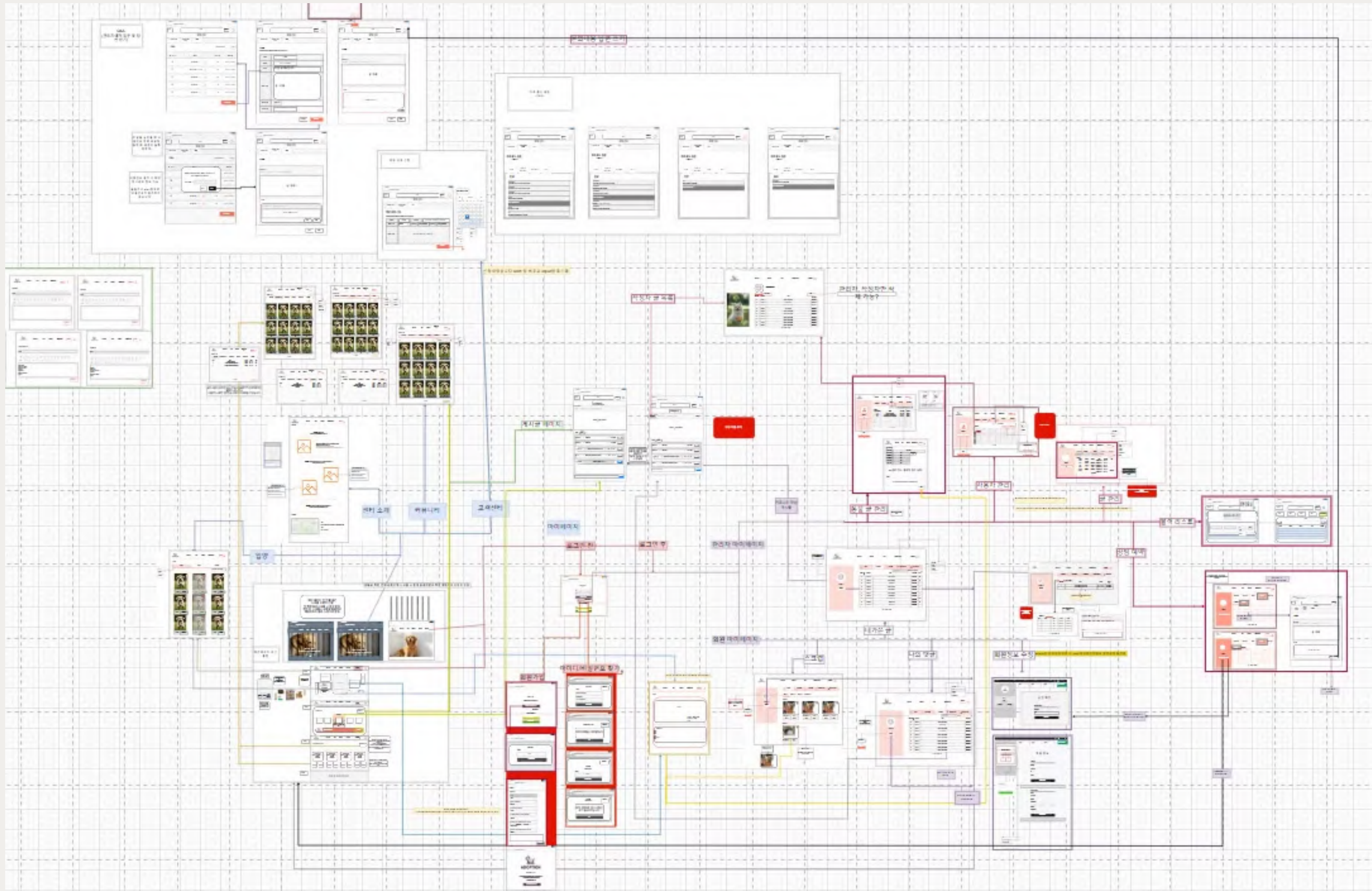
# 기획 및 설계







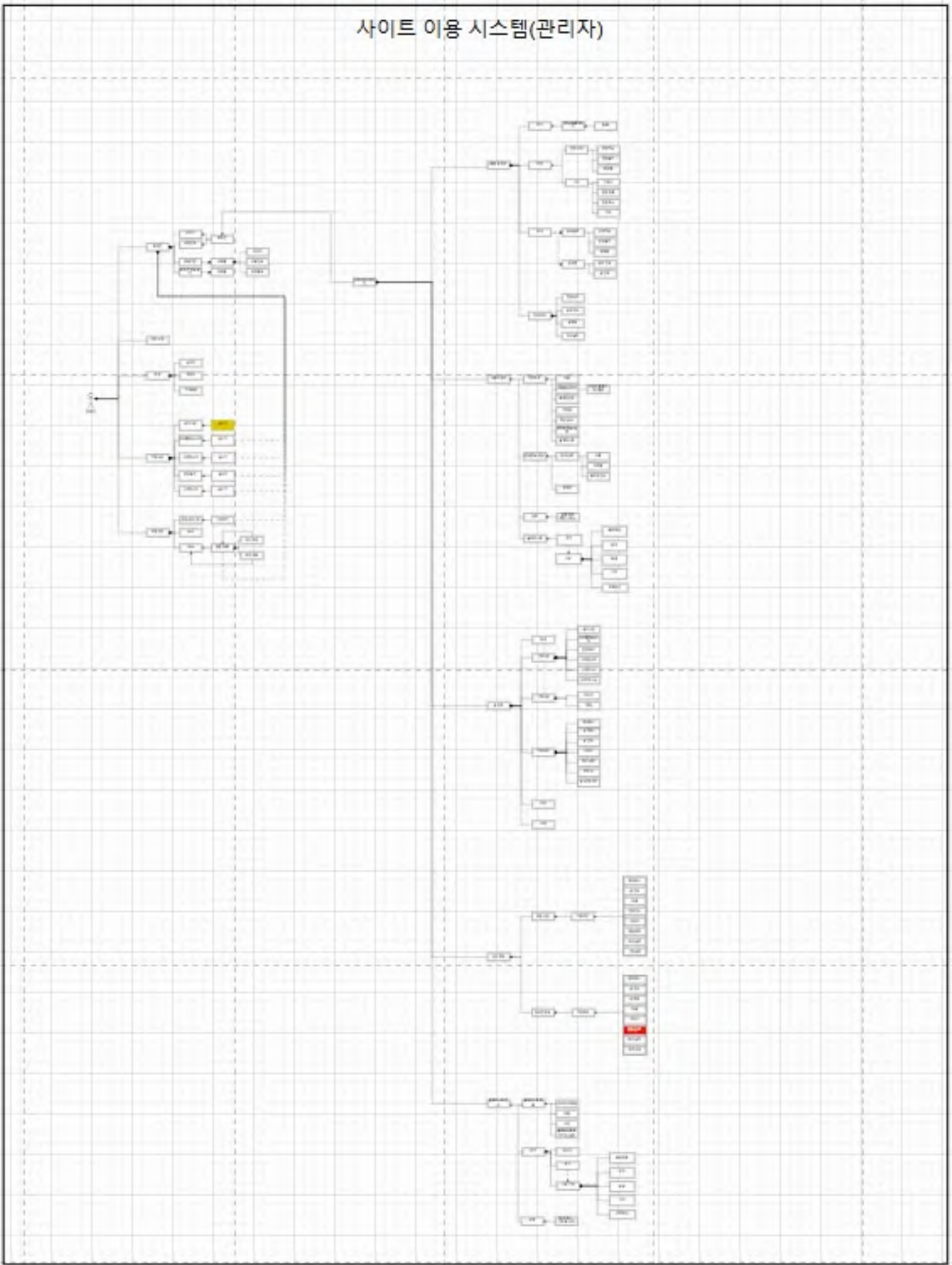
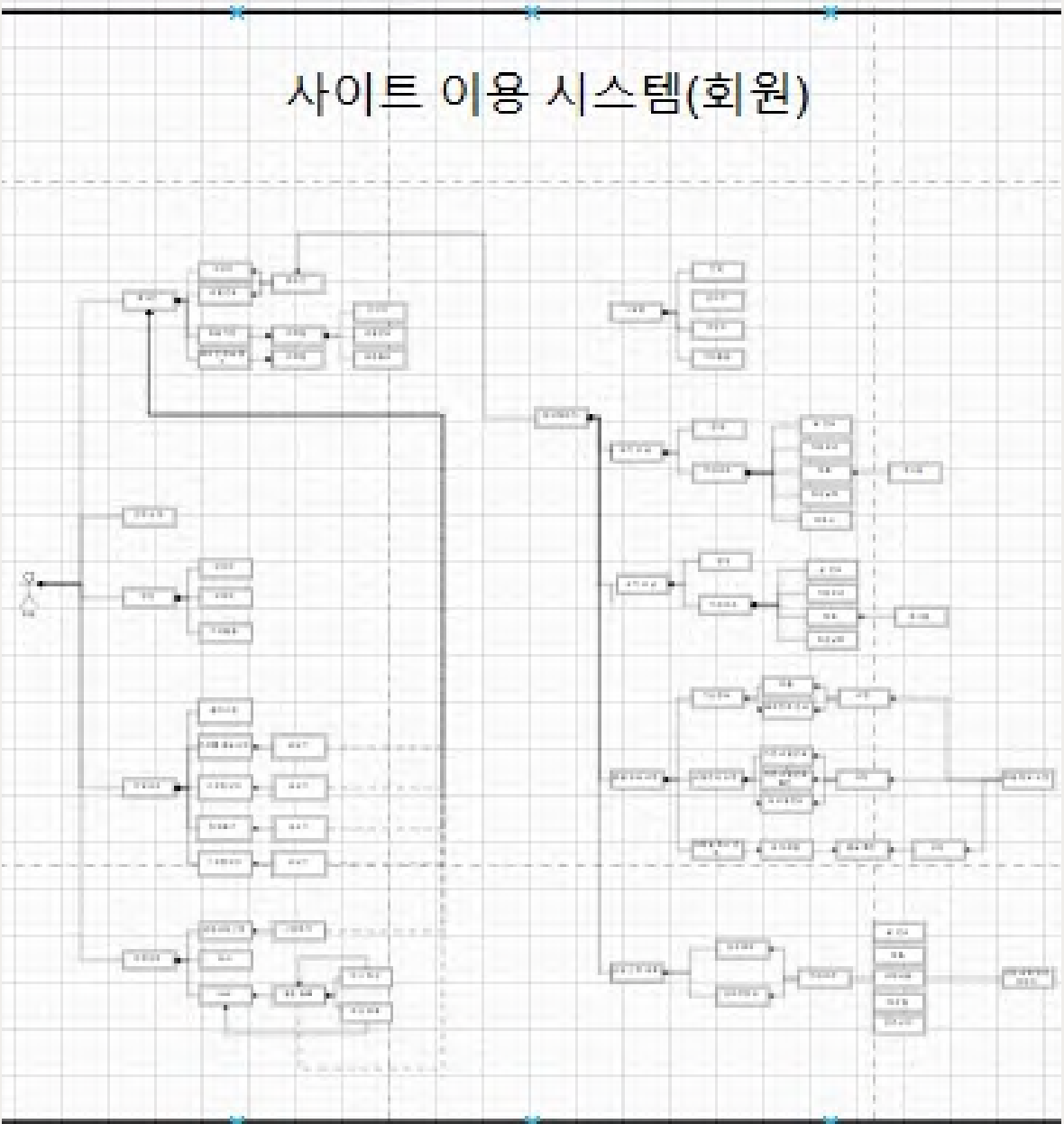
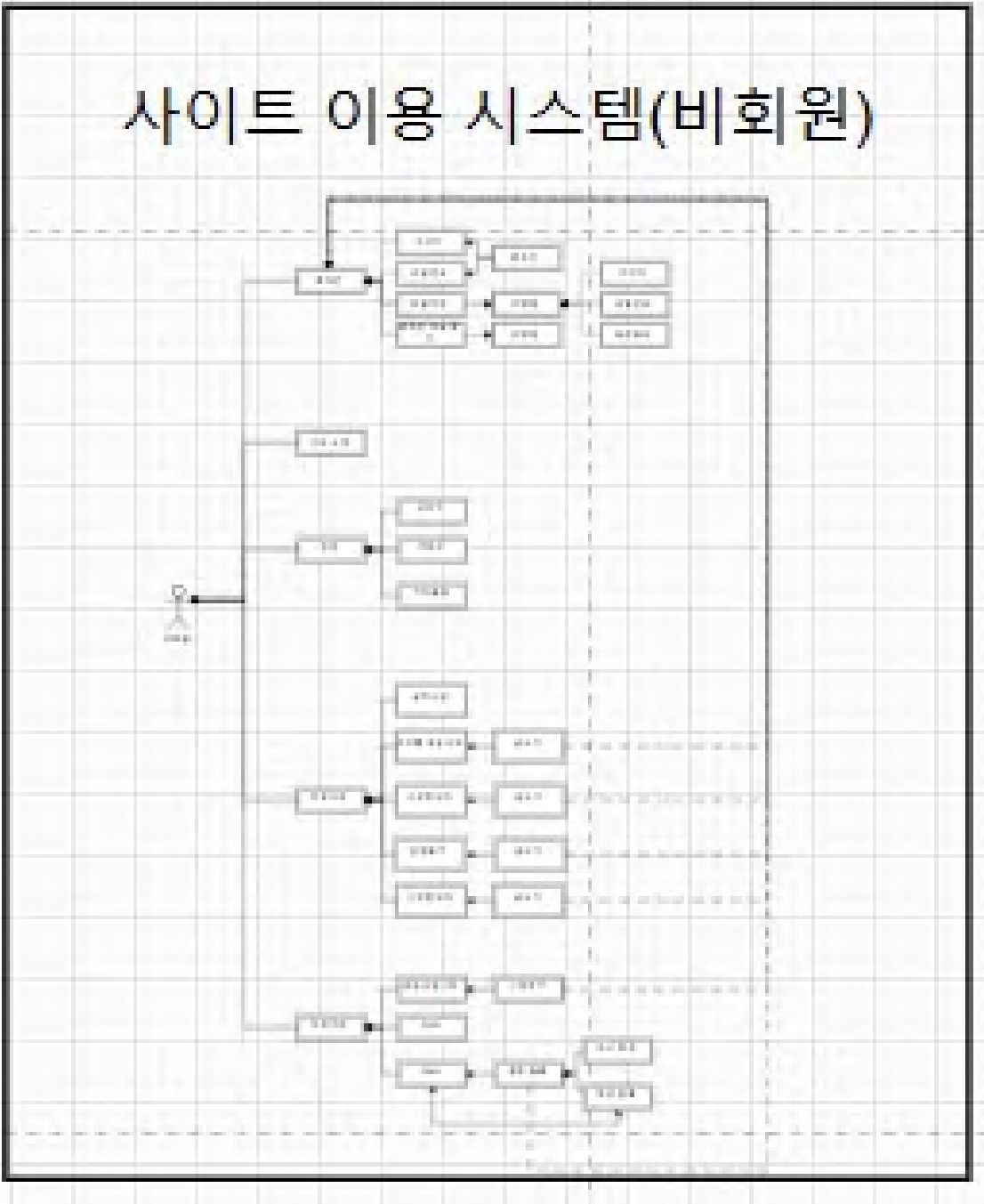
# 기획 및 설계





# 기획 및 설계




# 기획 및 설계





# API 명세서

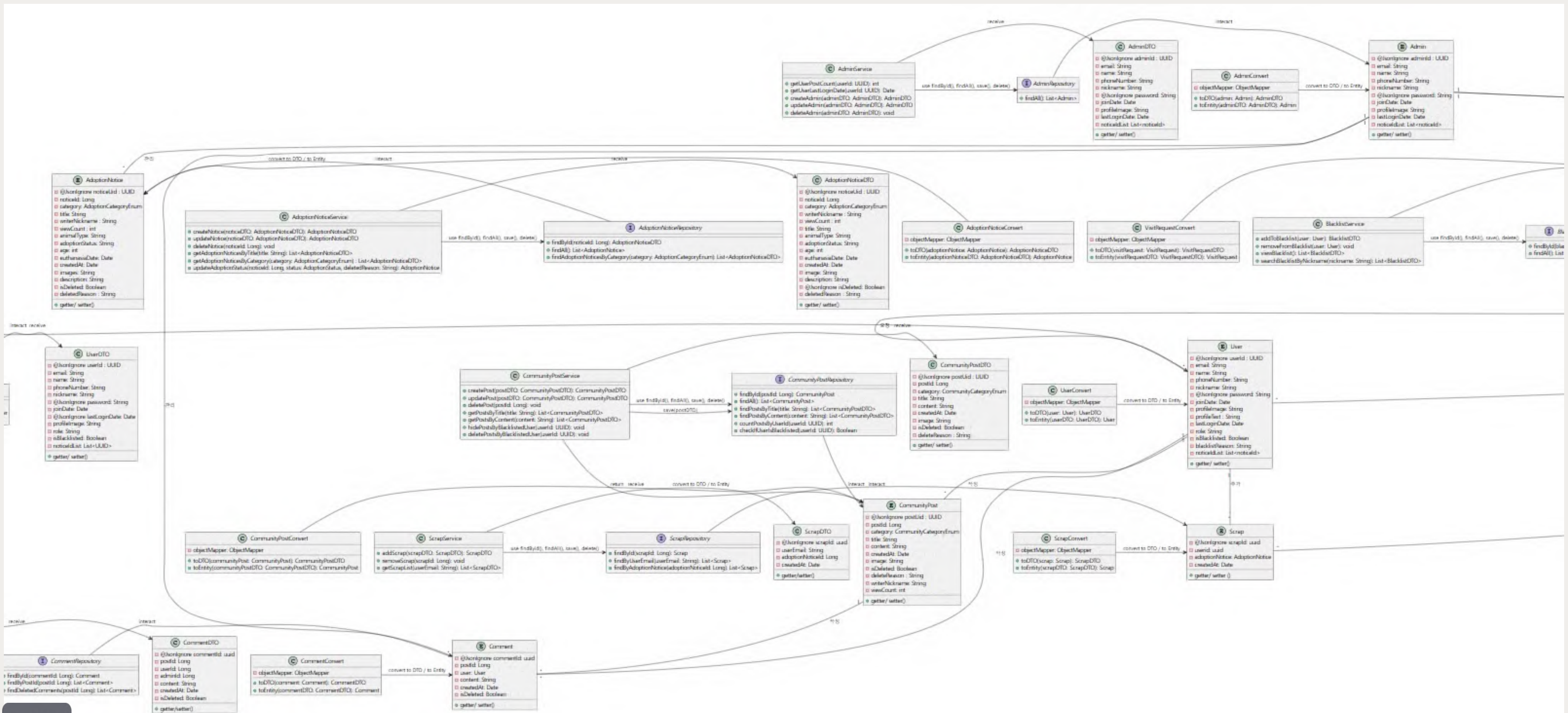
API 명세서

As 가능	Ⓜ HTTP 메서드	≡ API path	≡ parameter	≡ Request	≡ Response
 <b>계좌입금</b>					
 <b>로그인 페이지</b>					
<b>로그인</b>	POST	/api/user/login		{ "email": "susan.white456@gmail.com", "password": "susie4567" }	{ "message": "성공" }
<b>자동로그인</b>	GET	/api/auth/auto-login			
<b>로그아웃</b>	POST	/api/auth/logout			{ "message": "성공" }
 <b>회원가입</b>	POST	/api/user/register		{ "name": "코스모", "phone": "010-1234-5678", "email": "example@example.com", "nickname": "개냥이", "password": "Efg123!@" }	{ "message": "성공합니다." }
<b>회원탈퇴</b>	DELETE	/api/user/delete		{ "name": "코스모", "phone": "010-1234-5678", "email": "example@example.com", "nickname": "개냥이", "password": "Efg123!@" }	{ "message": "성공" }
<b>이메일 체크</b>	POST	/api/user/check/email		{ "email": "example@example.com", }	{ "message": "성공합니다." }
<b>닉네임 체크</b>	POST	/api/user/check/nickname		{ "nickname": "kosmo147" }	{ "message": "성공합니다." }
<b>아이디 찾기</b>	POST	/api/user/find/email		{ "name": "코스모", "phone": "010-1234-5678" }	{ "message": "성공합니다." }
<b>패스워드 찾기</b>	POST	/api/user/find/password		{ "email": "example@example.com", "number": "123456" }	{ "message": "성공합니다." }
<b>관리자 페이지 등록</b>	GET	/api/admin/adopt			{ "id": "123", "title": "3세/마"

As 가능	Ⓜ HTTP 메서드	≡ API path	≡ parameter	≡ Request	≡ Response
 <b>마이페이지</b>					
<b>프로필 변경 페이지</b>	GET	/api/user/profile			
<b>프로필 변경 페이지 수정</b>	PATCH	/api/user/profile/update		{ "img_src": "https://example.com/dogs/1.jpg", "text": "저는 김포모입니다" }	{ "img_src": "https://example.com/dogs/1.jpg", "text": "저는 김포모입니다" }
<b>비밀번호 재설정</b>	POST	/api/user/auth/password		{ "email": "example@example.com", "password": "Efg123!@" }	
<b>회원정보 수정 페이지</b>	GET	/api/user/info			
<b>회원정보 수정 기본 정보 수정</b>	PATCH	/api/user/info/update		{ "email": "example@example.com", "nickname": "개냥이", "phone": "010-1234-5678", "updated_at": "2024-11-07T12:00:00Z" }	{ "message": "성공" }
<b>회원가입 안내 및 홍보글</b>	POST	충북		{ "nickname": "개냥이" }	{ "message": "성공합니다." }
<b>사용자 정보 수정 및 내 정보 확인</b>	POST	충북			
<b>회원정보 수정 비밀번호 수정</b>	PATCH	/api/user/password/update		{ "password": "Efg123!@" }	{ "message": "성공" }
<b>회원정보 수정 프로필 수정</b>	PATCH	필요 x		{ "image_url": "https://example.com/dogs/1.jpg", "profile_intro": "안녕하세요 반갑습니다" }	{ "message": "성공" }
<b>사용자 스캔 페이지</b>	GET	/api/user/my-scan			{ "message": "성공" "data": { "noticeId": 1, "postId": 1, "title": "3세/마", "category": "id", "expire_date": "2024-11-07T12:00:00Z", "dogs_id": 123, "image_url": "https://example.com/dogs/1.jpg", "contents": "충북 : 픽스견 일 색 : 한강색 체중 : 2kg 나이 : 3세" }

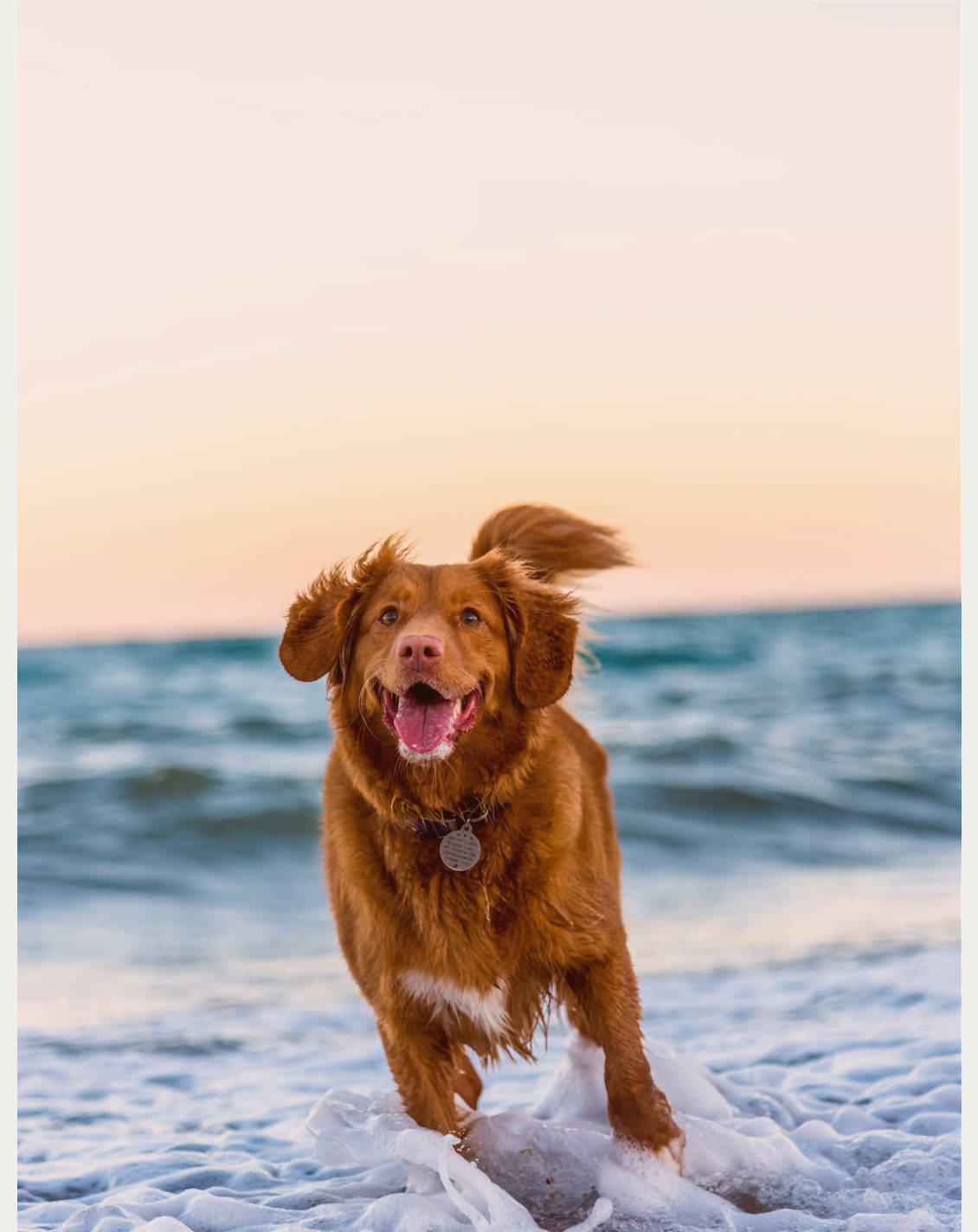
As 가능	Ⓜ HTTP 메서드	≡ API path	≡ parameter	≡ Request	≡ Response
				{ "created_at": "2024-11-06T10:00:00Z" }	
<b>커뮤니티 댓글쓰기</b>	POST	/api/community/comment/newcomment		{ "email": "example@example.com", "content": "이 글에 대해 좋은 의견입니다. 도움이 되었습니다.", "nickname": "qwer123", "created_at": "2024-11-06T12:00:00Z" // 댓글 작성 시간 (선택적, 서버에서 자동으로 처리할 수도 있음) }	{ "message": "성공" }
<b>커뮤니티 댓글보기</b>	GET	/api/community/{postId}/comment/{comment-id}			{ "content": "이 글입니다. 도움이 되었습니다.", "nickname": "qwer123", "created_at": "2024-11-06T12:00:00Z" }
<b>커뮤니티 글 수정</b>	PATCH	/api/community/{postId}		{ "email": "example@example.com", "postId": "3", "title": "수정된 제목", "content": "수정된 내용입니다. 추가 정보를 업데이트합니다.", "category": "missing", "image_url": "https://example.com/updated-image.jpg", "updated_at": "2024-11-07T12:00:00Z" }	{ "message": "성공" }
<b>커뮤니티 글 삭제</b>	DELETE	/api/community/{postId}		{ "email": "example@example.com", "category": "missing", "postId": "3" }	{ "message": "성공" }
<b>커뮤니티 댓글 삭제</b>	DELETE	/api/community/{postId}/{comment-id}		{ "email": "example@example.com", "category": "missing", "comment-id": "3" }	{ "message": "성공" "content": "이 글입니다. 도움이 되었습니다.", "nickname": "qwer123", "created_at": "2024-11-06T12:00:00Z" }
 <b>고견</b>					
 <b>고견</b>	GET	필요 x			{ "message": "성공" }

# 클래스 다이어그램





# 기술 스택 소개





# DataBase

## 1

### 테이블 설계 (ERD 포함)

```
CREATE TABLE IF NOT EXISTS `community_post` (  
  `post_uid` VARCHAR(36) NOT NULL COLLATE 'utf8mb3_general_ci',  
  `delete_reason` VARCHAR(255) NULL DEFAULT NULL COLLATE 'utf8mb3_general_ci',  
  `is_deleted` BIT(1) NOT NULL,  
  `post_category` ENUM('ANNOUNCEMENT', 'ADOPTREVIEW', 'BUYANDSELL', 'FINDCHILD', 'REPORT') NULL DEFAULT NULL COLLATE 'utf8mb3_general_ci',  
  `post_content` TEXT NULL DEFAULT NULL COLLATE 'utf8mb3_general_ci',  
  `post_created_at` DATETIME(6) NOT NULL,  
  `post_id` INT(11) NULL DEFAULT NULL,  
  `post_img_url` TEXT NULL DEFAULT NULL COLLATE 'utf8mb3_general_ci',  
  `post_title` VARCHAR(255) NULL DEFAULT NULL COLLATE 'utf8mb3_general_ci',  
  `post_updated_at` DATETIME(6) NOT NULL,  
  `post_view_count` INT(11) NOT NULL,  
  `user_id` VARCHAR(255) NULL DEFAULT NULL COLLATE 'utf8mb3_general_ci',  
  PRIMARY KEY (`post_uid`) USING BTREE,  
  UNIQUE INDEX `UK5djt4w7q6wk68ryo6xb716dp0` (`post_id`) USING BTREE,  
  INDEX `fk_postuser_id` (`user_id`) USING BTREE,  
  CONSTRAINT `fk_postuser_id` FOREIGN KEY (`user_id`) REFERENCES `user` (`user_id`) ON UPDATE CASCADE ON DELETE CASCADE  
)  
COLLATE='utf8mb3_general_ci'
```

```
blacklists`, `email`, `is_blacklisted`, `name`, `nickname`, `password`, `phone_number`, `profile_text`, `provider`,  
'[{ "blackedDate": [2024,11,10,9,30], "blackReason": "Abusive behavior" }]', 'john.doe982@gmail.com', b'1', '김진우', 'joh  
'[]', 'sophia.moon876@gmail.com', b'0', '이채은', 'sophia876', 'sophia8765', '010-6789-0123', '이채은입니다. 다양한 분야에  
'[]', 'janedoe1234@gmail.com', b'0', '이수진', 'sujin94', 'password4567', '010-3456-7890', '안녕하세요, 이수진입니다. 세상  
'[]', 'henry.kim842@gmail.com', b'0', '강지원', 'henry842', 'henry8421', '010-8425-6953', '강지원입니다. 시스템 개발에 많은  
'[]', 'emily.choi730@gmail.com', b'0', '이세연', 'emily730', 'emily7301', '010-3721-5849', '이세연입니다. 창의적 문제 해결  
'[]', 'lily.kim301@gmail.com', b'0', '정수빈', 'lily301', 'lily3017', '010-6548-9075', '정수빈입니다. 디자인과 UI/UX에 대한  
'[]', 'emily.lee999@gmail.com', b'0', '황지민', 'emily999', 'emily9998', '010-2468-1357', '황지민입니다. 디자인과 기술의 융  
'[]', 'jake.lee502@gmail.com', b'0', '이재호', 'jake502', 'jake5023', '010-2956-4832', '이재호입니다. 기술적 문제 해결에 대  
'[]', 'lucas.smith876@gmail.com', b'0', '김준호', 'lucas876', 'lucas8765', '010-3456-7890', '김준호입니다. 다양한 기술을 익  
'[]', 'kylie.johnson567@gmail.com', b'0', '김다영', 'kylie567', 'kylie5678', '010-9876-5432', '김다영입니다. 다양한 분야에  
'[]', 'steve.smith1980@gmail.com', b'0', '조준호', 'steve1980', 'steve1980pass', '010-5678-9012', '조준호입니다. 저는 창의  
'[]', 'lisa.choi760@gmail.com', b'0', '최지영', 'lisa760', 'lisa7601', '010-3762-8407', '최지영입니다. 디자인과 창의적 사고  
'[]', 'olivia.jung543@gmail.com', b'0', '정현주', 'olivia543', 'olivia5432', '010-8361-5278', '정현주입니다. 신기술과 혁신  
'[]', 'julia.kim567@gmail.com', b'0', '박주현', 'julia567', 'julia5678', '010-1234-5678', '박주현입니다. 새로운 기술과 도
```

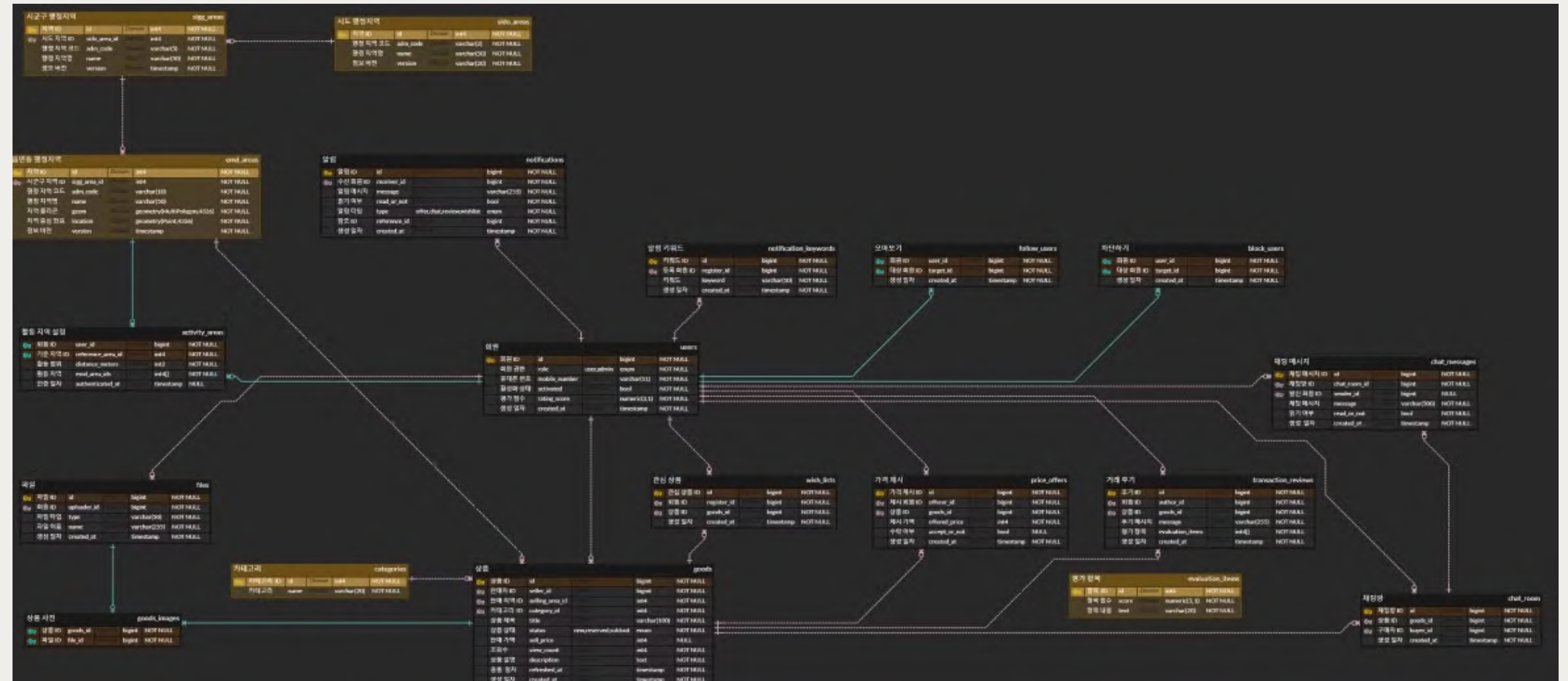
```
dto  
  AdoptMypageDTO  
  AdoptNoticeDTO  
  AdoptNoticeListDTO  
  BlacklistDTO  
  CommentDTO  
  CommentListDTO  
  CommunityDTO
```



# DataBase

## 복잡하다는 단점

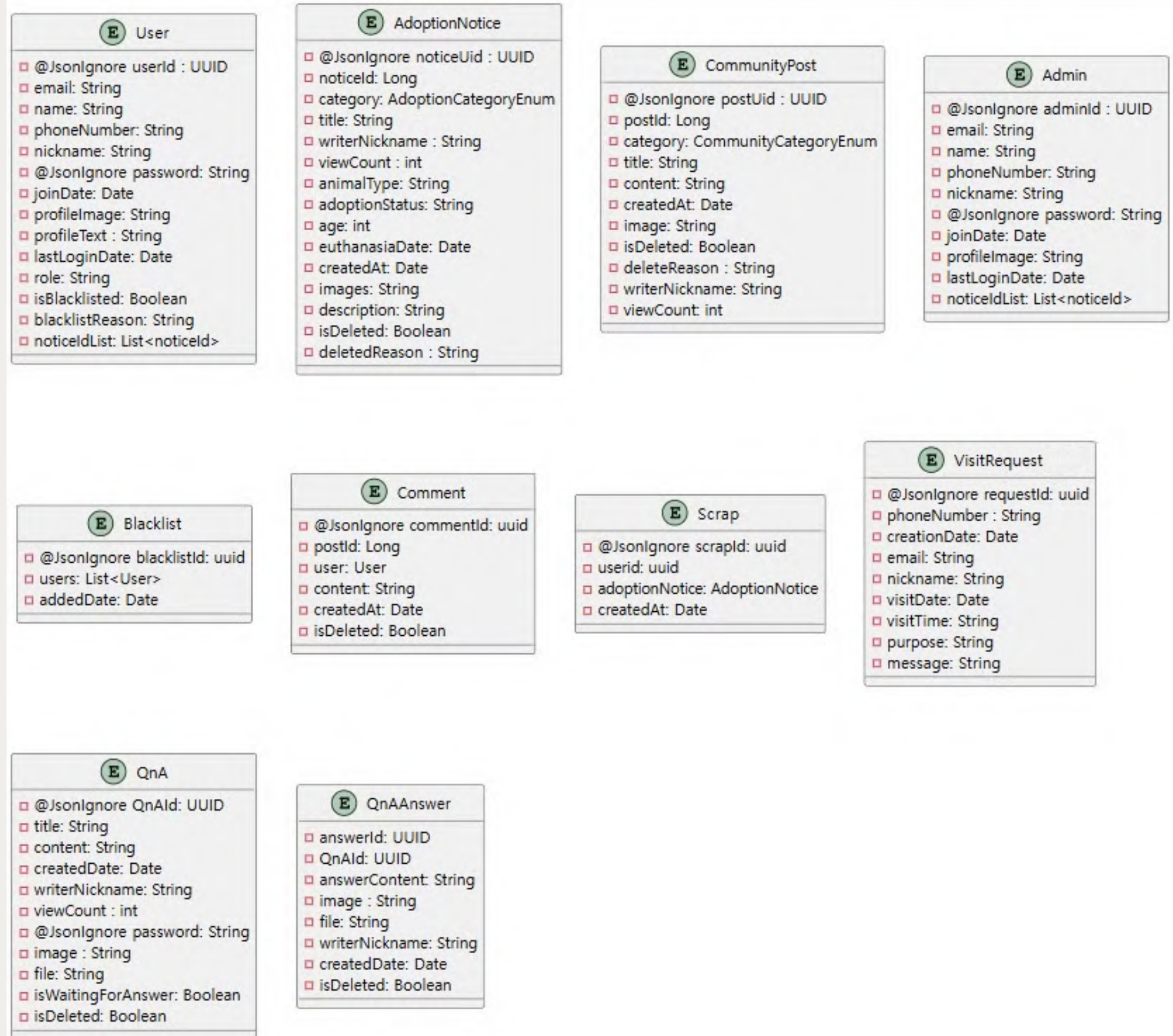
수평 확장성, 유동적 확장의 어려움



# DataBase

객체 간 관계를 자연스럽게 표현


데이터의 깊이가 깊어져 관리의 어려움





# Kick!

converter 사용

- ✓  enums
  - ⓔ AdoptStatus
  - ⓔ Authority
  - ⓔ NoticeCategory
  - ⓔ PostCategory
  - ⓔ Provider
  - ⓔ QnAState
  - ⓔ VisitPurpose
  - ⓔ VisitTime

# Kick!

converter 사용

```
package com.kosmo.kkomoado

public enum VisitTime { 6
    TIME1, // 09:30 ~ 10:30
    TIME2, // 10:30 ~ 11:30
    TIME3, // 11:30 ~ 12:30
    TIME4, // 14:00 ~ 15:00
    TIME5, // 15:00 ~ 16:00
    TIME6 // 16:00 ~ 17:00
}
```

# Kick!

converter 사용

converter

© AdoptStatusConverter

© AuthorityConverter

© BlacklistConverter

© NoticeCategoryConverter

© PostCategoryConverter

© ScrapConverter

© UrlConverter

© UserConverter



# Kick!

converter 사용

```
@Converter 2개 사용 위치 ㄹ poqazpoqaz
public class BlacklistConverter implements AttributeConverter<List<BlacklistDTO>, String> {

    private final ObjectMapper objectMapper; 5개 사용 위치

    // 생성자에서 ObjectMapper를 초기화하고 JavaTimeModule을 등록
    public BlacklistConverter() { 0개의 사용위치 ㄹ poqazpoqaz
        this.objectMapper = new ObjectMapper();
        this.objectMapper.registerModule(new JavaTimeModule()); // LocalDateTime 처리
    }

    @Override 0개의 사용위치 ㄹ poqazpoqaz
    public String convertToDatabaseColumn(List<BlacklistDTO> attribute) {
        try {
            if (attribute == null) {
                return null; // null 처리 (예: DB에 null 저장)
            }
            // List<BlacklistDTO>를 JSON 문자열로 변환
            return objectMapper.writeValueAsString(attribute);
        } catch (IOException e) {
            throw new IllegalArgumentException("Error converting List<BlacklistDTO> to JS
        }
    }

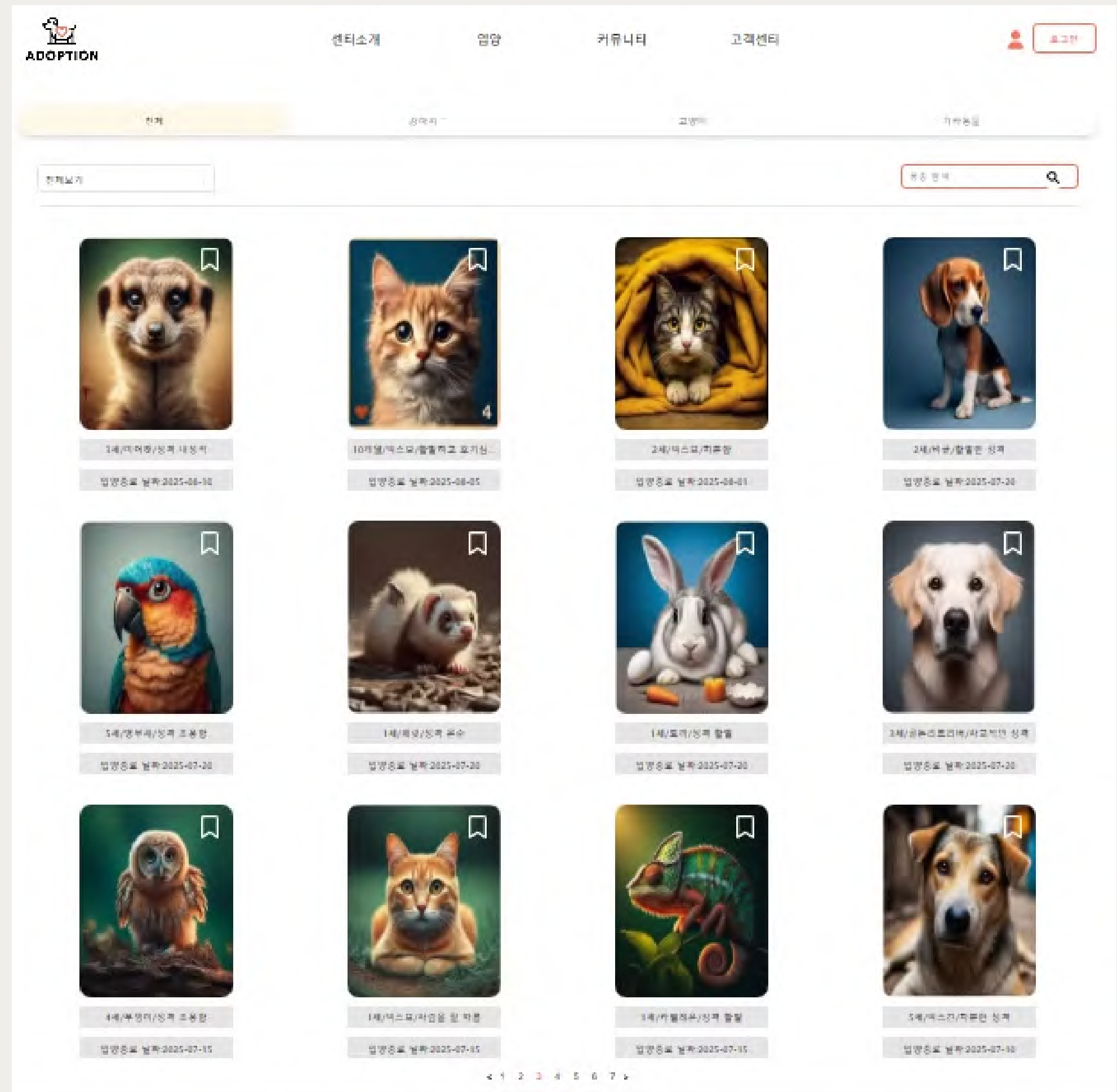
    @Override 0개의 사용위치 ㄹ poqazpoqaz
    public List<BlacklistDTO> convertToEntityAttribute(String dbData) {
        try {
            if (dbData == null) {
                return null; // null 처리 (예: DB에서 null 값 처리)
            }
            // JSON 문자열을 List<BlacklistDTO>로 변환
            return objectMapper.readValue(dbData, objectMapper.getTypeFactory().construct
        } catch (IOException e) {
            throw new IllegalArgumentException("Error converting JSON to List<BlacklistDT
        }
    }
}
```

# Frontend

1

UI/UX 디자인

반응형 웹으로 디자인

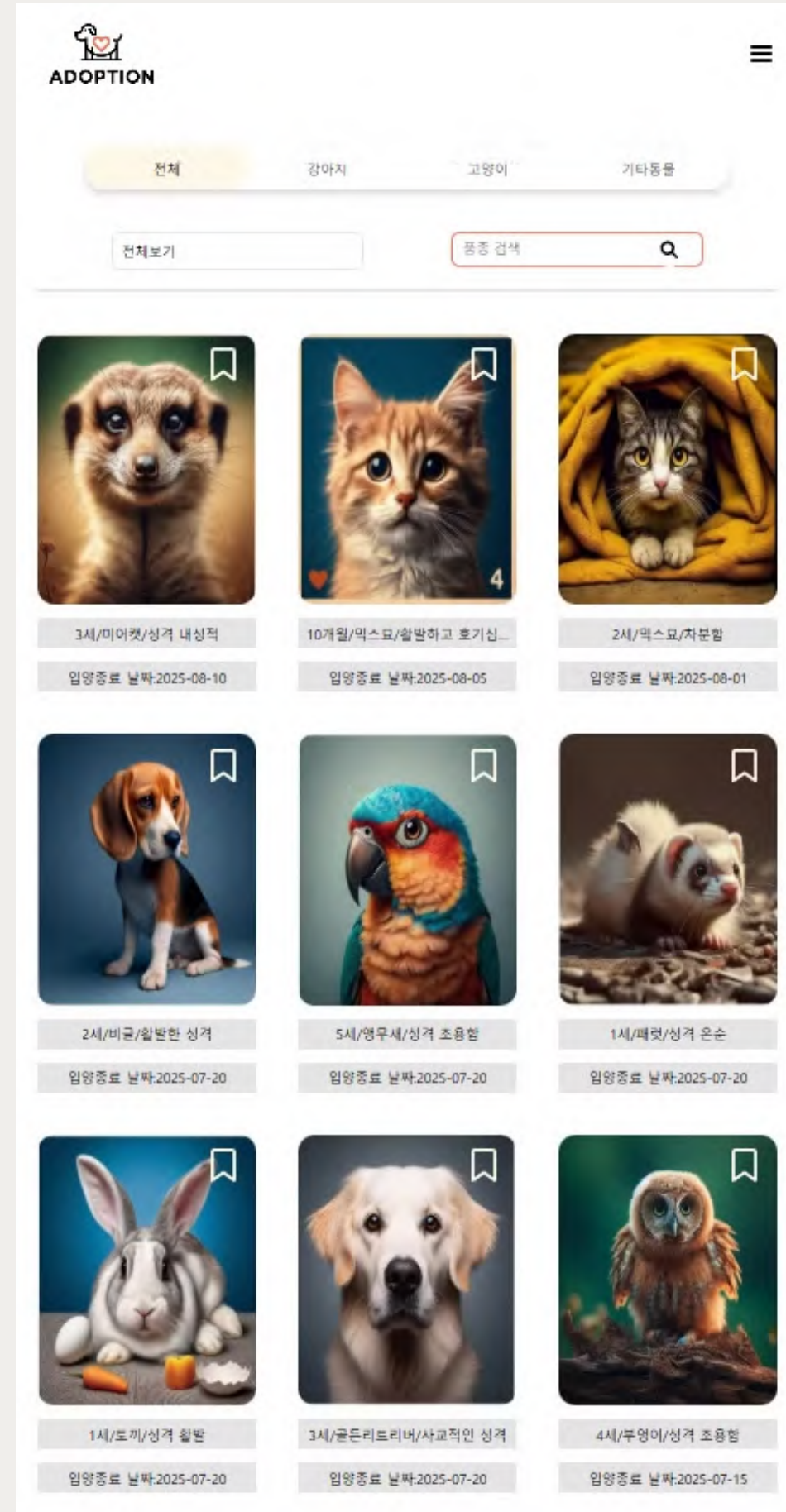


# Frontend

1

UI/UX 디자인

반응형 웹으로 디자인



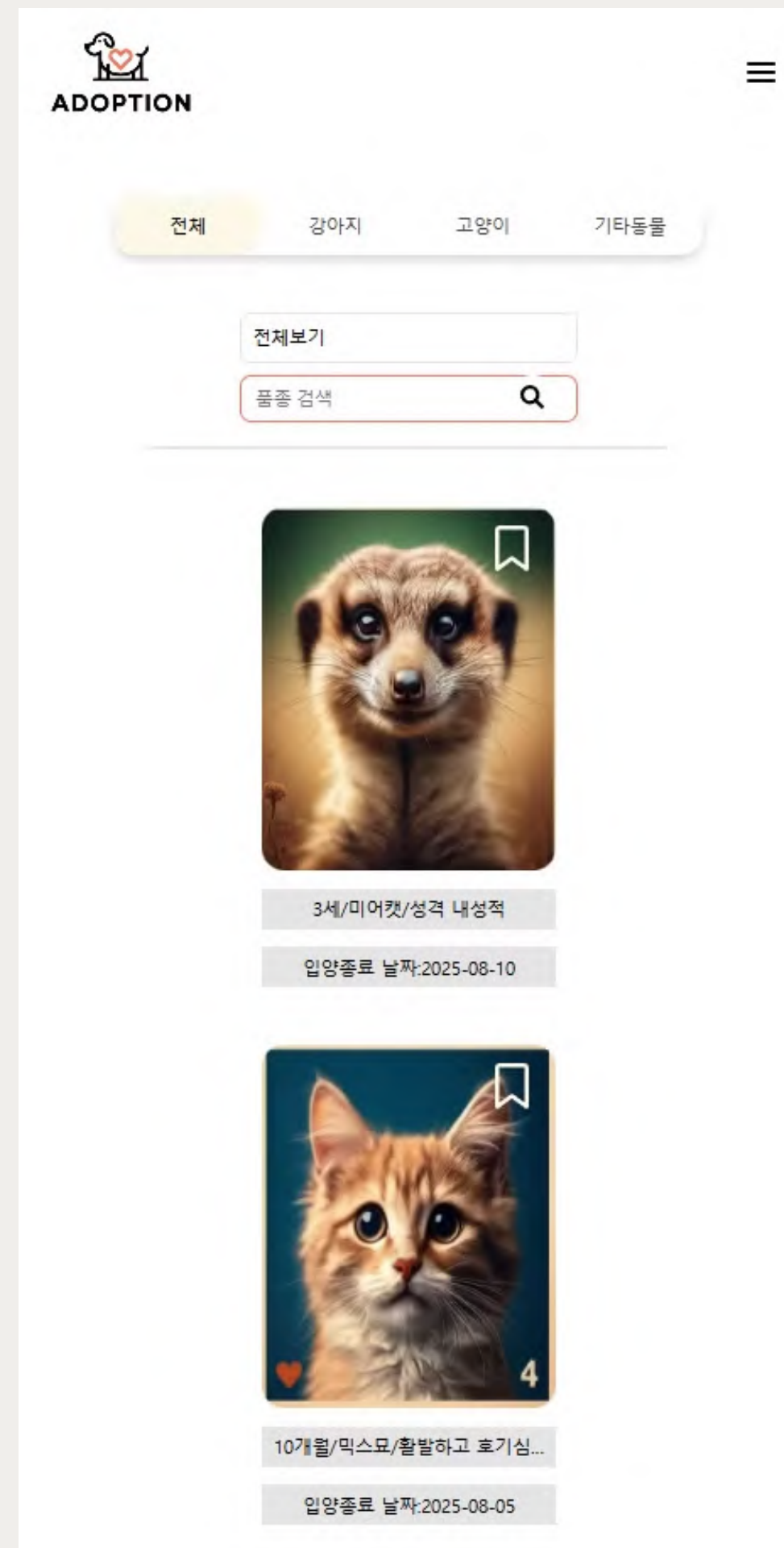


# Frontend

## 1

### UI/UX 디자인

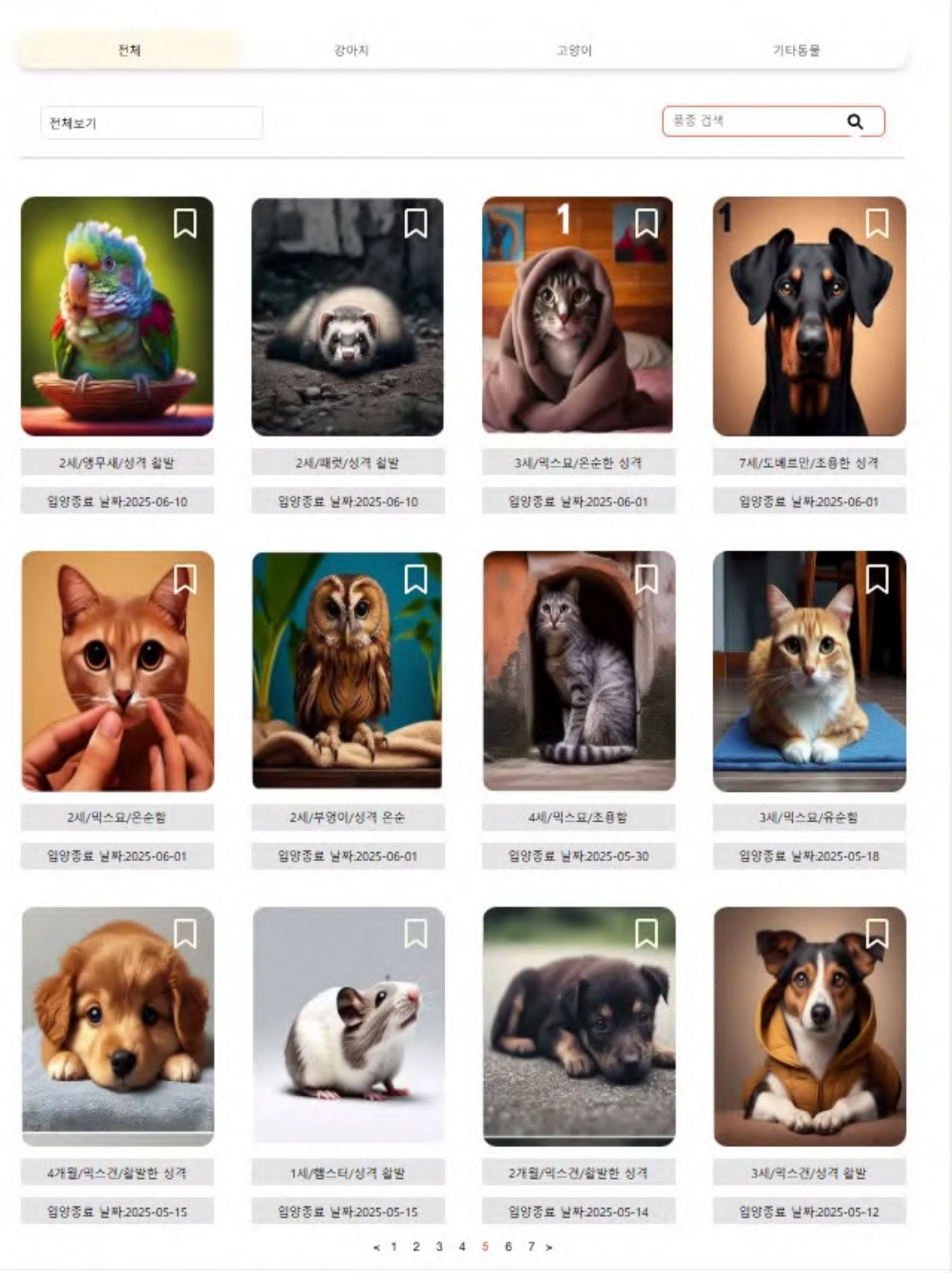
반응형 웹으로 디자인



# Frontend

## 2 주요 페이지 및 화면 흐름

네비게이션 기능을 중심으로 구현



# Frontend

## 2 주요 페이지 및 화면 흐름

네비게이션 기능을 중심으로 구현



공지사항	아이를 찾습니다	입양 후기	사고판다	신고합니다
<div>선택해주세요</div> <div>글 내용 &amp; 글 제목</div>				
번호	제목	작성자	작성일	조회수
1	유기견 보호소 신규 입양권 소개	관리자	2024-12-04T19:00:00	180
2	유기견 보호소 새벽 산책 봉사자 모집	관리자	2024-12-04T23:00:00	150
3	유기견 병원비 후원 캠페인 안내	관리자	2024-12-04T13:00:00	120
4	유기견 보호소 외부 봉사활동 안내	관리자	2024-12-04T15:00:00	130
5	유기견 보호소 겨울철 안전 관리 안내	관리자	2024-12-04T20:00:00	90
6	유기견 보호소 입양 후 관리 팀 안내	관리자	2024-12-04T21:00:00	110
7	유기견 보호소 후원자 감사 행사 안내	관리자	2024-12-04T12:00:00	150
8	유기견 보호소 기부 물품 안내	관리자	2024-12-05T00:00:00	160
9	유기견 입양 신청서 제출 안내	관리자	2024-12-04T18:00:00	170
10	유기견 입양 후기 모집	관리자	2024-12-04T14:00:00	100
11	유기견 보호소 자원봉사 활동 일정	관리자	2024-12-04T17:00:00	160
12	유기견 보호소 정기 후원 캠페인 안내	관리자	2024-12-04T22:00:00	140
13	유기견 보호소 자원봉사자 모집 안내	관리자	2024-12-04T10:00:00	180
14	유기견 입양 프로그램 안내	관리자	2024-12-04T11:00:00	200
15	유기견 보호소 임시 보호자 모집	관리자	2024-12-04T16:00:00	140



# Frontend

## 2

### 주요 페이지 및 화면 흐름

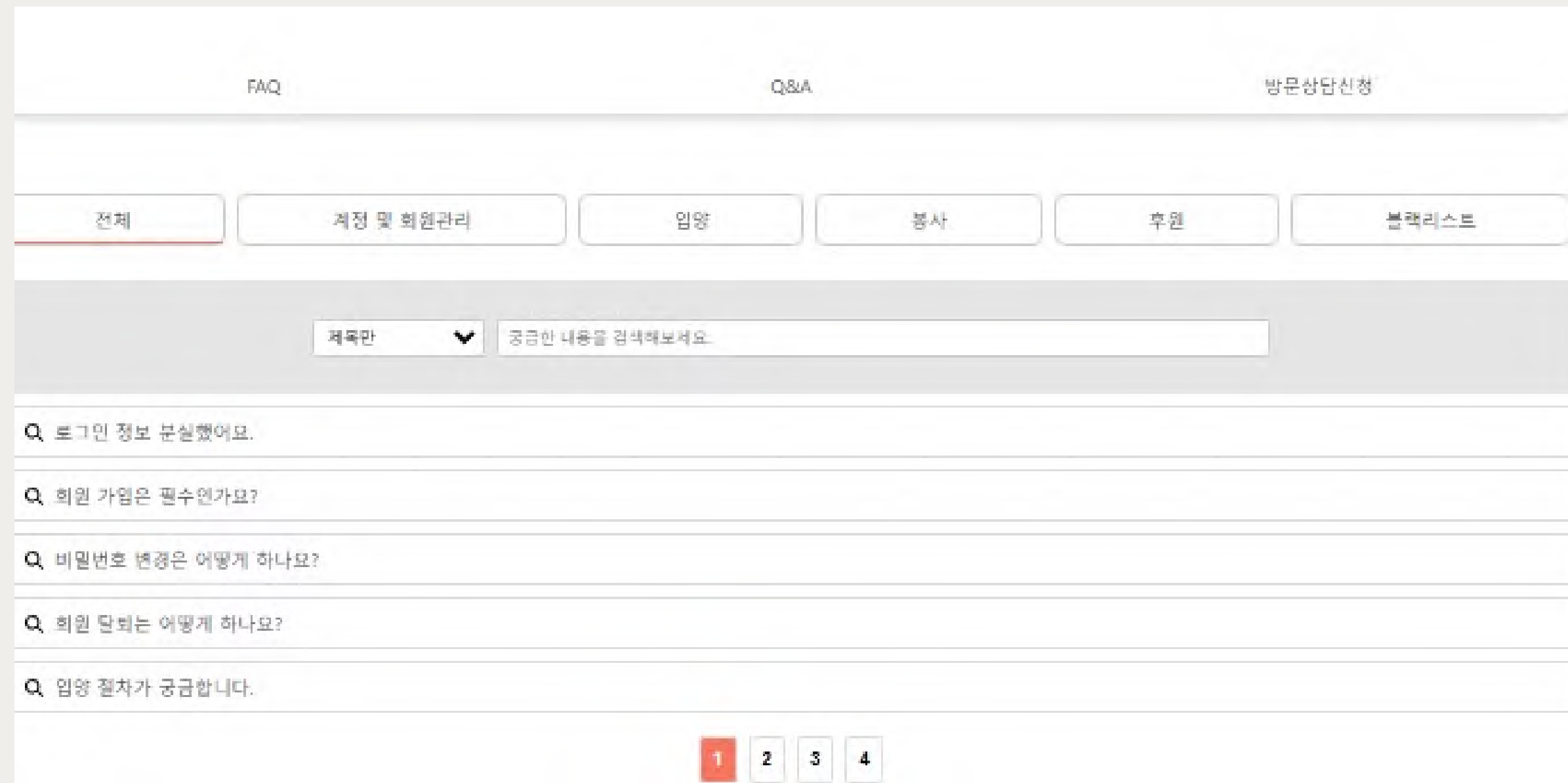
네비게이션 기능을 중심으로 구현

센터소개

입양

커뮤니티

고객센터



# Backend

## 1

### 서버 설정 및 API 구현

CRUD 기능을 중심으로 API를 구현

```
// 입양공지 조건별 검색(12개)
@GetMapping("/list") 0개의 사용위치 1 poqazpoqaz +1
public ResponseEntity<AdoptNoticeListDTO> getLists(
    @RequestParam(name = "page", defaultValue = "1") int page,    // 페이지 번호 (디폴트: "1")
    @RequestParam(name = "noticeCategory") NoticeCategory noticeCategory,    // 카테고리
    @RequestParam(name = "sortBy", defaultValue = "euthanasiaDate") String sortBy,
    @RequestParam(name = "sortOrder", defaultValue = "desc") String sortOrder,
    HttpServletRequest request) {    // 정렬 순서 (디폴트: "asc")

    // 페이지와 정렬 설정
    Sort sort = Sort.by(Sort.Order.asc(sortBy));    // 기본 정렬은 오름차순
    if ("desc".equalsIgnoreCase(sortOrder)) {
        sort = Sort.by(Sort.Order.desc(sortBy));
    }

    // size를 고정값 12로 설정
    int size = 12;

    Pageable pageable = PageRequest.of(page, size, sort);

    AdoptNoticeListDTO result;
    // 카테고리 필터링 처리
    if (NoticeCategory.ALL.equals(noticeCategory)) {
        result = adoptionNoticeService.getNotices(pageable);    // 카테고리 없이 전체 마이컬
    } else {
        result = adoptionNoticeService.getNoticesByCategory(noticeCategory, pageable);
    }

    result = adoptionNoticeService.getUserScrapMappingList(result, request);

    // 상태 코드 200 OK와 함께 반환
    return ResponseEntity.ok(result);
}

// 검색어로 공고 조회
```



# Backend

## 2

### 인증 및 권한 관리

사용자 권한 분리 (어드민 / 유저)

```
// 댓글 업데이트 메서드
public boolean updateComment(CommentDTO commentDTO, String authority, String sessionId) {

    // 댓글 ID로 DB에서 해당 댓글 조회
    Optional<CommentEntity> existingCommentOptional = commentRepository.findById(commentDTO.getId());

    if (existingCommentOptional.isEmpty()) {
        return false; // 댓글이 존재하지 않으면 업데이트 실패
    }

    CommentEntity commentEntity = existingCommentOptional.get();

    // USER 권한인 경우: 본인이 작성한 댓글만 수정 가능
    if (authority.equals("USER")) {
        if (!commentEntity.getUserId().equals(sessionUserId)) {
            return false; // 본인이 작성한 댓글이 아니면 업데이트 실패
        }

        // 댓글 내용과 수정 시간을 업데이트
        commentEntity.setCommentContent(commentDTO.commentContent());
        commentEntity.setCommentUpdatedAt(LocalDate.now()); // 수정 시간을 현재 시간으로 설정
    }

    // ADMIN 권한인 경우: 모든 댓글 수정 가능
    else if (authority.equals("ADMIN")) {
        commentEntity.setIsDeleted(true); // 댓글 삭제 상태로 변경
        commentEntity.setCommentDelReason(commentDTO.commentDelReason()); // 삭제 이유
        commentEntity.setCommentUpdatedAt(LocalDate.now()); // 수정 시간을 현재 시간으로 설정
    }

    // 수정된 댓글 저장
    commentRepository.save(commentEntity);

    return true; // 댓글 업데이트 성공
}
```

# Backend

## 3

JPA 사용

데이터베이스와의 연동을 최적화

```
@Entity 13개 사용 위치 1 poqazpoqaz +3
@Getter
@Setter
@Table(name = "comment")
public class CommentEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.UUID)
    @Column(name = "comment_id", nullable = false, length = 36)
    private String commentId;
    // 댓글번호

    @Lob
    @Column(name = "comment_content", nullable = false)
    private String commentContent;
    // 댓글컨텐츠

    @Column(name = "comment_created_at", nullable = false, updatable = false)
    private LocalDateTime commentCreatedAt;
    // 댓글작성일

    @Column(name = "comment_updated_at", nullable = false)
    private LocalDateTime commentUpdatedAt;
    // 댓글업데이트날짜

    @Column(name = "is_deleted", nullable = false)
    private Boolean isDeleted = false; // 댓글 삭제 여부를 나타내는 필드
    // 댓글 삭제 사유

    @Column(name = "comment_del_reason", nullable = false)
    private String commentDelReason = null;
    // 삭제여부(관리자)

    // communitypost의 post id
    @Column(name = "post_uid", nullable = false)
    private String postUid; // FK
    // 커뮤니티 글번호
    // comment 작성자
    @Column(name = "user_id", nullable = false)
    private String userId; // FK
    // 작성자

    @Override 1 poqazpoqaz +2
    public String toString() {
        return "CommentEntity{" +
            "commentId='" + commentId + '\'' +
            ", commentContent='" + commentContent + '\'' +
            ", commentCreatedAt=" + commentCreatedAt +
```

# Backend

## 3

JPA 사용

데이터베이스와의 연동을 최적화

```
package com.kosmo.kkomoadopt.repository;

import ...

@Repository 4개 사용 위치 1 KOSMO +2
public interface CommentRepository extends JpaRepository<CommentEntity, String> {

    // 댓글 ID와 작성자 ID로 댓글 존재 여부를 확인
    boolean existsByCommentIdAndUserId(String commentId, String userId); 1개 사용 위치 1 kosmo

    // 게시물 댓글 조회 메서드
    List<CommentEntity> findByPostUid(String postUid); 1개 사용 위치 1 KOSMO
}
```



# Backend

## 3

### JPA 사용

데이터베이스와의 연동을 최적화

// 댓글 업데이트 메서드

```
public boolean updateComment(CommentDTO commentDTO, String authority, String sessionId)
```

// 댓글 ID로 DB에서 해당 댓글 조회

```
Optional<CommentEntity> existingCommentOptional = commentRepository.findById(commentId)
```

```
if (existingCommentOptional.isEmpty()) {
```

```
    return false; // 댓글이 존재하지 않으면 업데이트 실패
```

```
}
```

```
CommentEntity commentEntity = existingCommentOptional.get();
```

// USER 권한인 경우: 본인이 작성한 댓글만 수정 가능

```
if (authority.equals("USER")) {
```

```
    if (!commentEntity.getUserId().equals(sessionUserId)) {
```

```
        return false; // 본인이 작성한 댓글이 아니면 업데이트 실패
```

```
    }
```

// 댓글 내용과 수정 시간을 업데이트

```
commentEntity.setCommentContent(commentDTO.commentContent());
```

```
commentEntity.setCommentUpdatedAt(LocalDateTime.now()); // 수정 시간을 현재 시간으로
```

```
}
```

// ADMIN 권한인 경우: 모든 댓글 수정 가능

```
else if (authority.equals("ADMIN")) {
```

```
    commentEntity.setIsDeleted(true); // 댓글 삭제 상태로 변경
```

```
commentEntity.setCommentDelReason(commentDTO.commentDelReason()); // 삭제 이유 저장
```

```
commentEntity.setCommentUpdatedAt(LocalDateTime.now()); // 수정 시간을 현재 시간으로
```

```
}
```

// 수정된 댓글 저장

```
commentRepository.save(commentEntity);
```

```
return true; // 댓글 업데이트 성공
```

```
}
```

# Backend

## 3

JPA 사용

데이터베이스와의 연동을 최적화

```
package com.kosmo.kkomoadopt.dto;

import java.time.LocalDateTime;

public record CommentDTO( 8개 사용 위치 📍 kosmo +1
    String commentId, 2개 사용 위치
    String commentContent, 2개 사용 위치
    String postUid, 1개 사용 위치
    String userId, 1개 사용 위치
    String commentDelReason 1개 사용 위치
) {
    public void setUserId(String sessionUserId) { 📍 KOSMO
    }
}
```

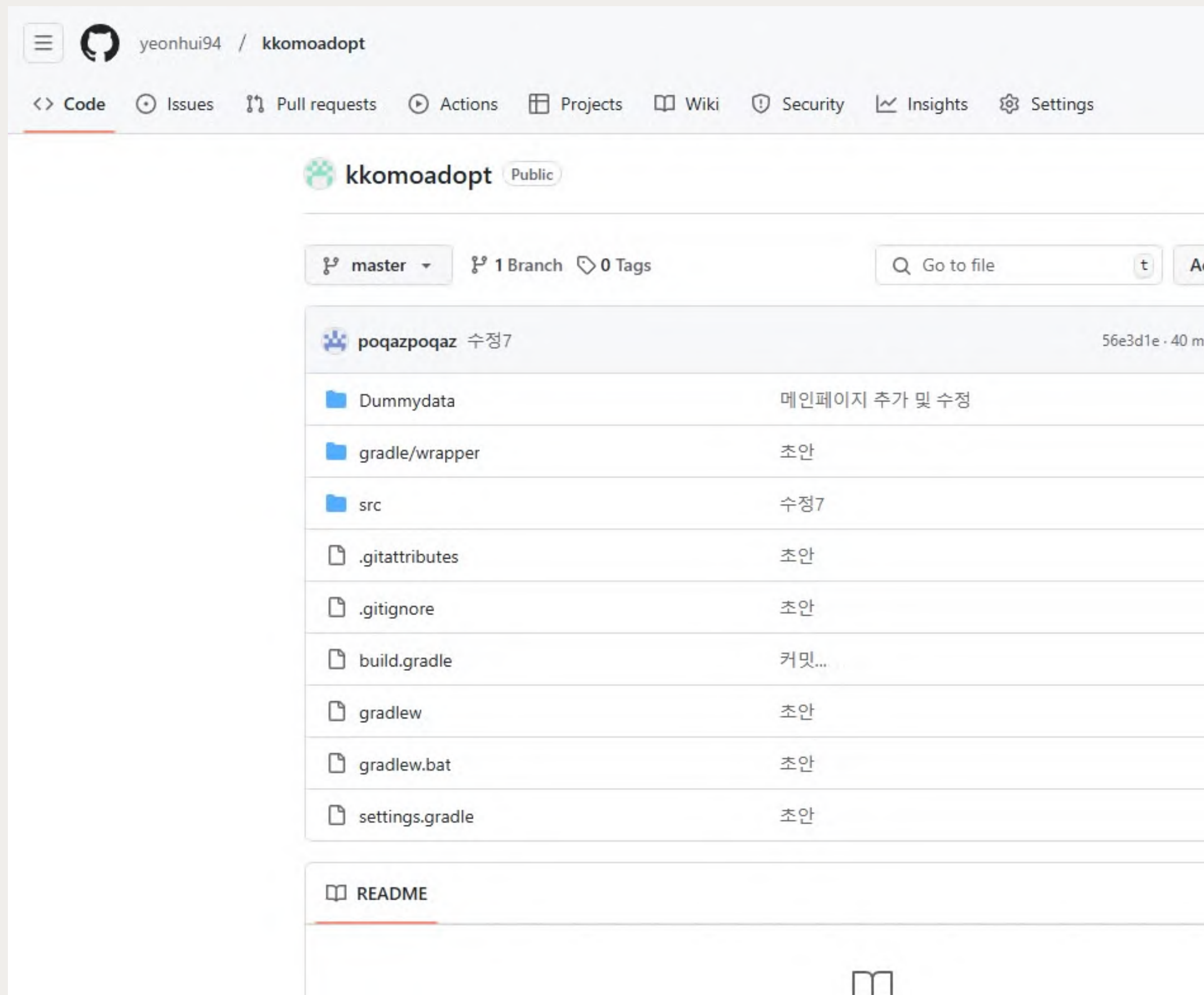
# 기타도구

1

Figma 활용

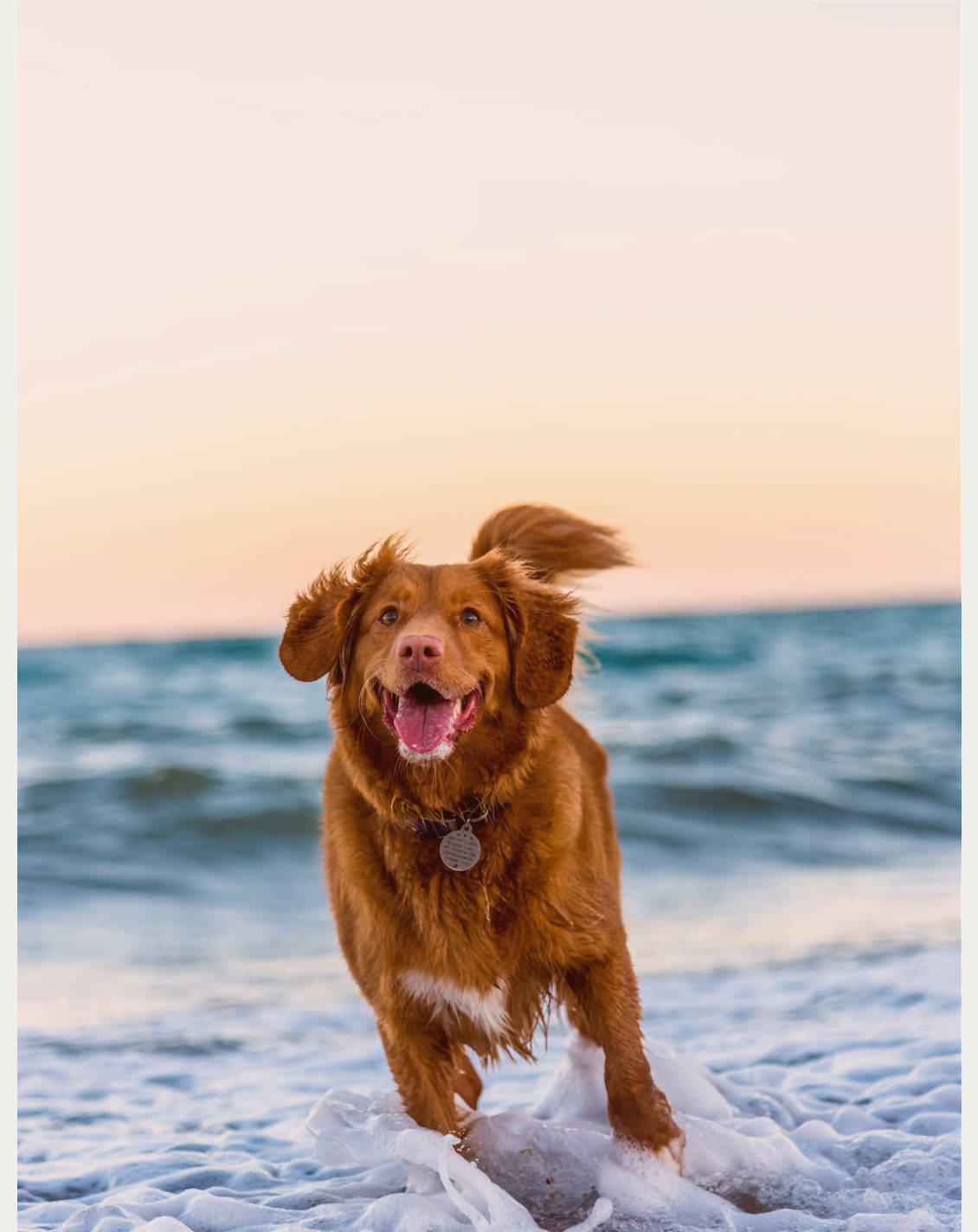
2

GitHub



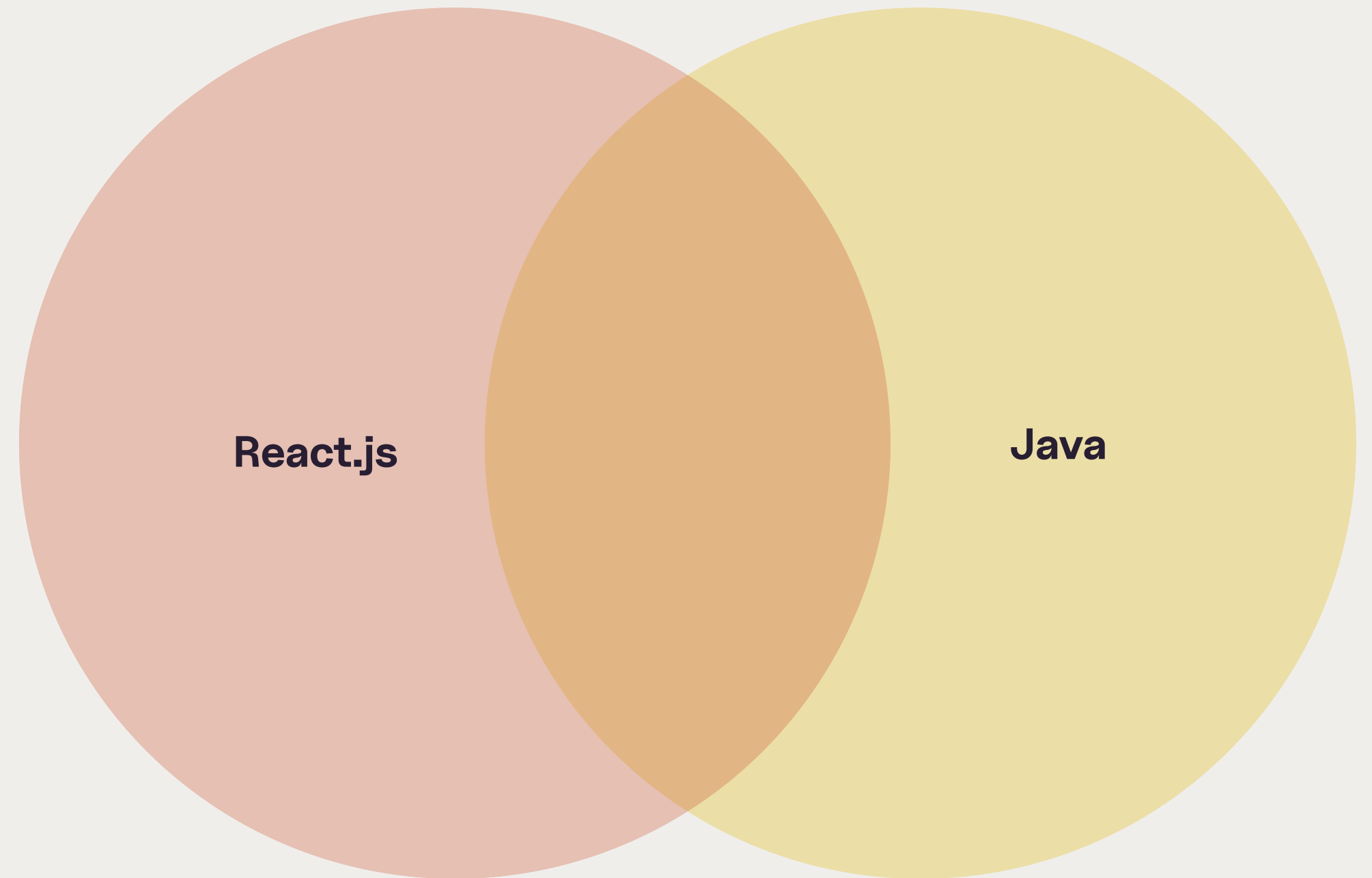


# 결론 및 개선 사항



# 기술적 성과

프론트엔드와 백엔드 간의 원활한 통신을 실현



# 학습과 성장

웹 애플리케이션 개발의 전반적인 흐름을 이해

팀워크와 협업의 중요성

데이터베이스 설계 및 API 연동에 대한 깊은 이해





# 개선사항

기능적 개선

UI/UX 개선

성능 최적화

테스트 및 오류 처리

추가 기능 제안

# 기능적 개선

1 민감 내용 암호화

2 댓글의 댓글

댓글

닉네임	내용
인삼밭	금방 찾을 수 있을 거예요!
체리콕	빨리 찾으면 좋겠네요 ㅠㅠ

댓글 쓰기

심한 욕설, 폭언, 음담패설, 광고성 글은 블랙리스트 대상이 될수있습니다.



상호 : (주)KKOMO 센터 | 주소 : 서울 금천구 가산디지털 2로 101 한라

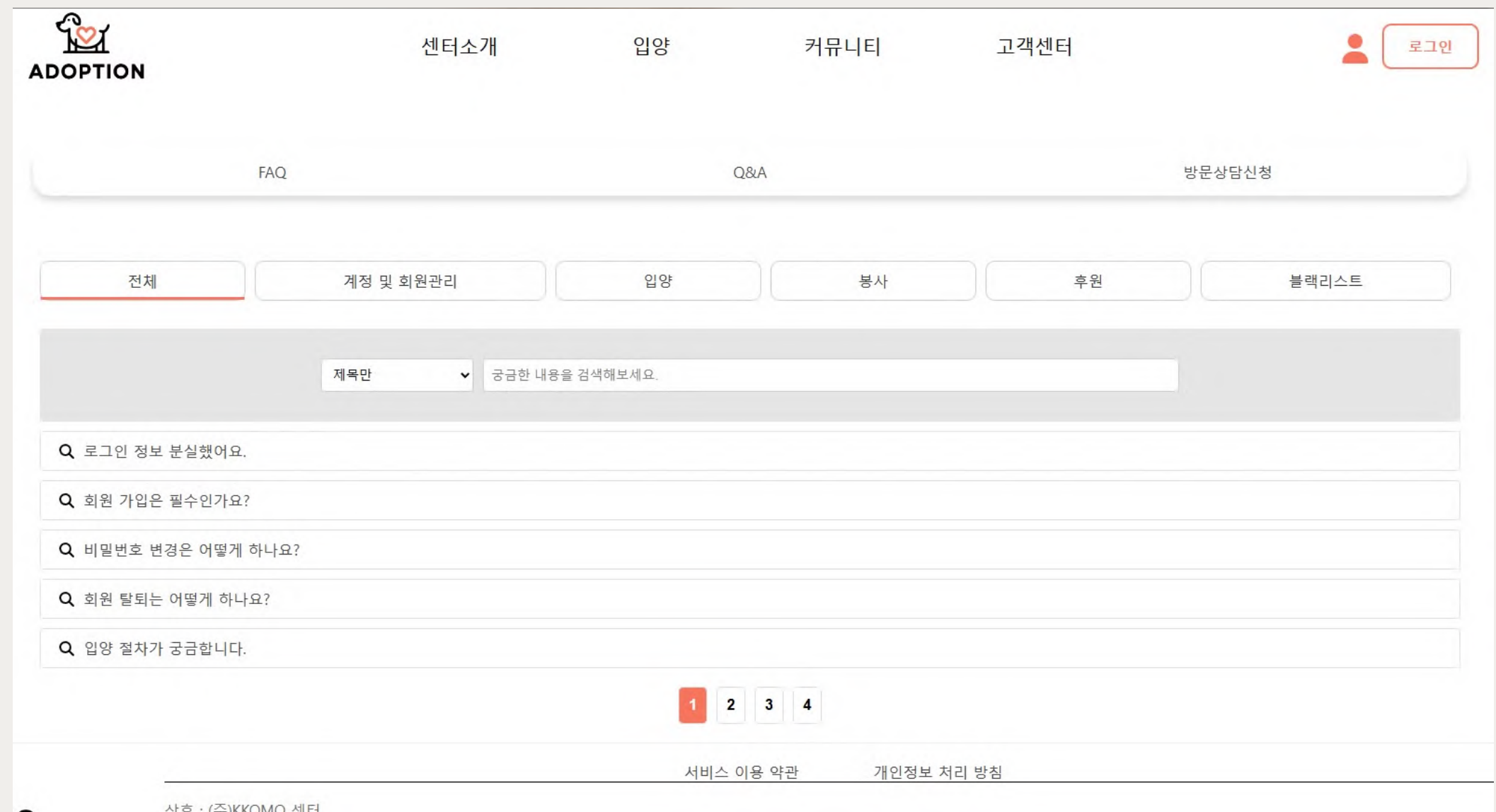
사업자번호 : 000-00-00000 | 이메일 : abcd@naver.com

Copyright© 2024 KKOMO센터 All Rights Reserved

# UI / UX 개선

## 1 다크모드 추가

## 2 조화로운 색상과 폰트 사용





# 성능 최적화

1 쿼리 속도 개선

---

2 로딩 속도 최적화

---

# 테스트 및 오류 처리

1 오류 처리 메시지

---

2 오류 알림 메시지 추가

---

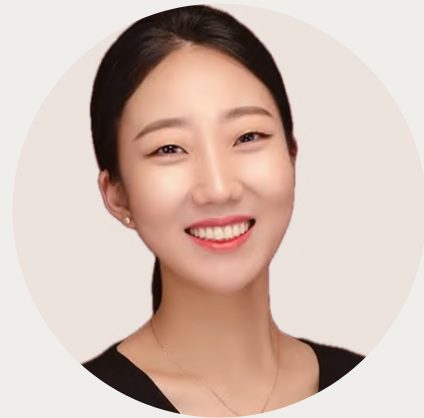
# 추가 기능 제안

1 실시간 상태 알림 기능 추가

---

2 사용자 간 소통할 수 있는 페이지

---



조연희

### 기획 및 요구사항 분석

- 와이어프레임 설계 및 프로토타입 제작

### 프론트엔드 개발

- UI/UX 설계 및 구현
- API 연동
- 사용자 인터페이스 개발
- 애니메이션 요소 추가

### 백엔드 개발

- API 설계 및 엔드포인트 구현



곽대훈

### 기획 및 요구사항 분석

- 와이어프레임 설계 및 프로토타입 제작

### API 명세서 작성 및 공유

### 프론트엔드 개발

- UI/UX 설계 및 구현
- API 연동
- 사용자 인터페이스 개발
- 애니메이션 요소 추가

### 백엔드 개발

- API 설계 및 엔드포인트 구현
- 데이터 생성, 수정, 삭제 로직 개발



김진혁

### 백엔드 개발

- Spring Boot 기반 서버 개발
- 데이터 생성, 수정, 삭제 로직 개발
- 데이터 처리 속도 개선
- RESTful API 설계 및 엔드포인트 구현
- HeidiSQL을 활용한 데이터 조작 및 테스트

### 프론트엔드 개발

- API 연동



지소엽

### 기획 및 요구사항 분석

- 와이어프레임 설계 및 프로토타입 제작

### 프론트엔드 개발

- UI/UX 설계 및 구현
- 사용자 인터페이스 개발



문상일

### 기획 및 요구사항 분석

- 와이어프레임 설계 및 프로토타입 제작

### 프론트엔드 개발

- UI/UX 설계 및 구현
- 사용자 인터페이스 개발

### 백엔드 개발

- API 설계 및 엔드포인트 구현
- 데이터 생성, 수정, 삭제 로직 개발



오재현

### 기획 및 요구사항 분석

- 와이어프레임 설계 및 프로토타입 제작

### 프론트엔드 개발

- UI/UX 설계 및 구현
- API 연동
- 컴포넌트 상태 관리
- 사용자 인터페이스 개발
- 애니메이션 요소 추가

### DB

- ERD 다이어그램 생성
- 클래스 다이어그램 생성

Thank you ♥





# Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It's free)