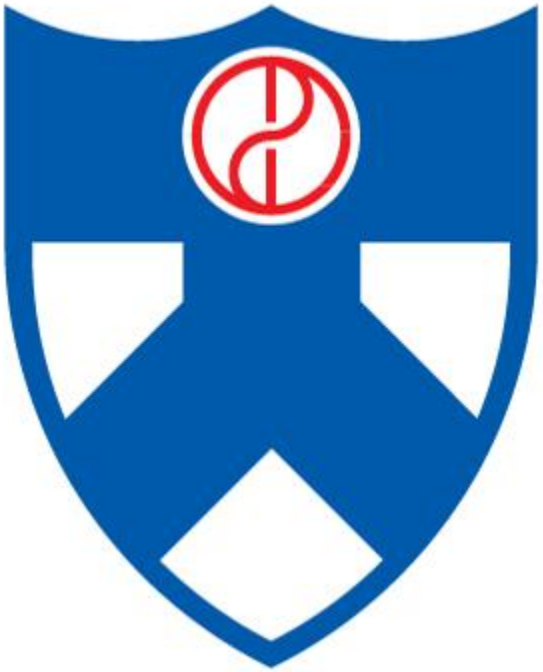
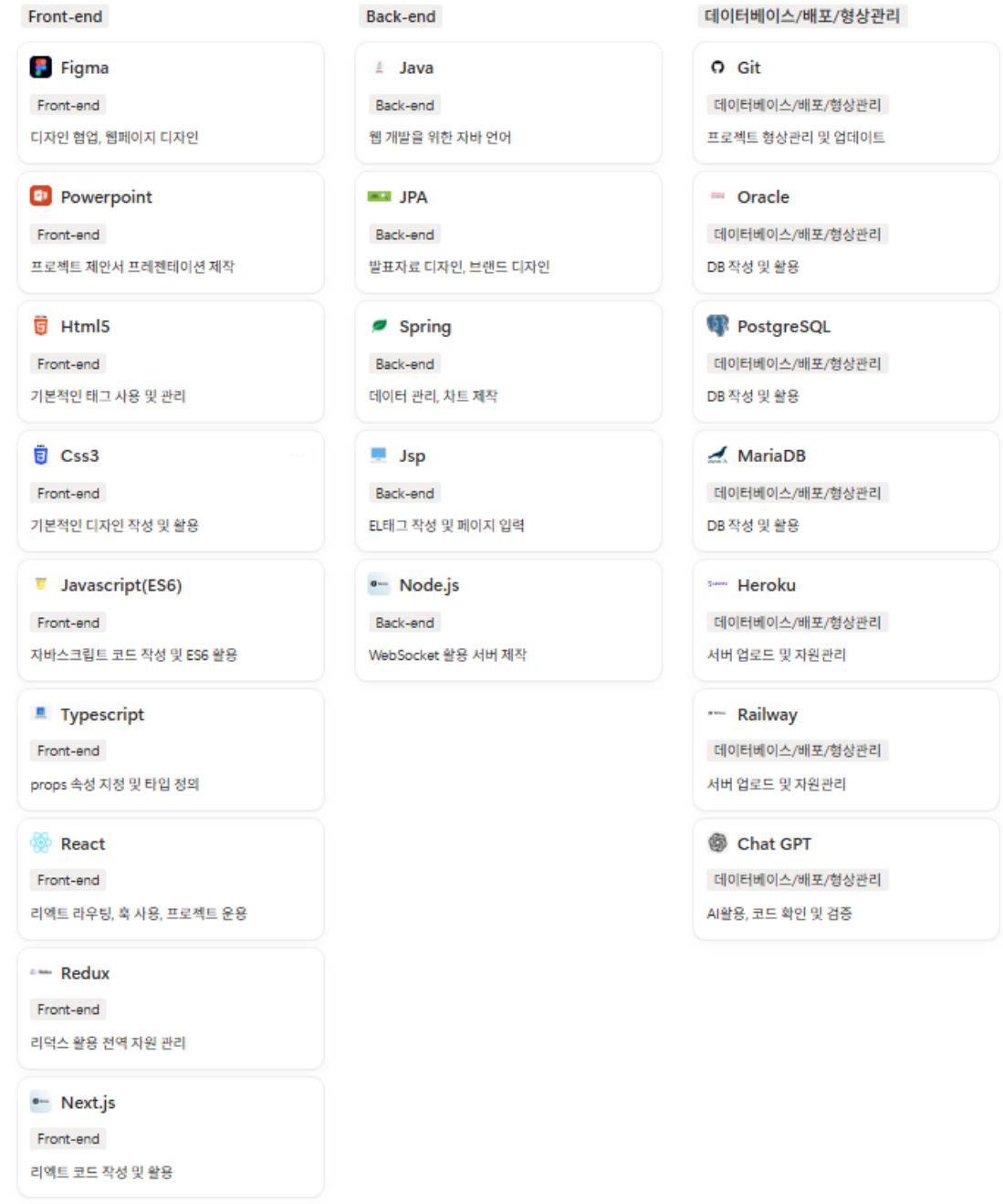
전산 포트폴리오

(신승철)



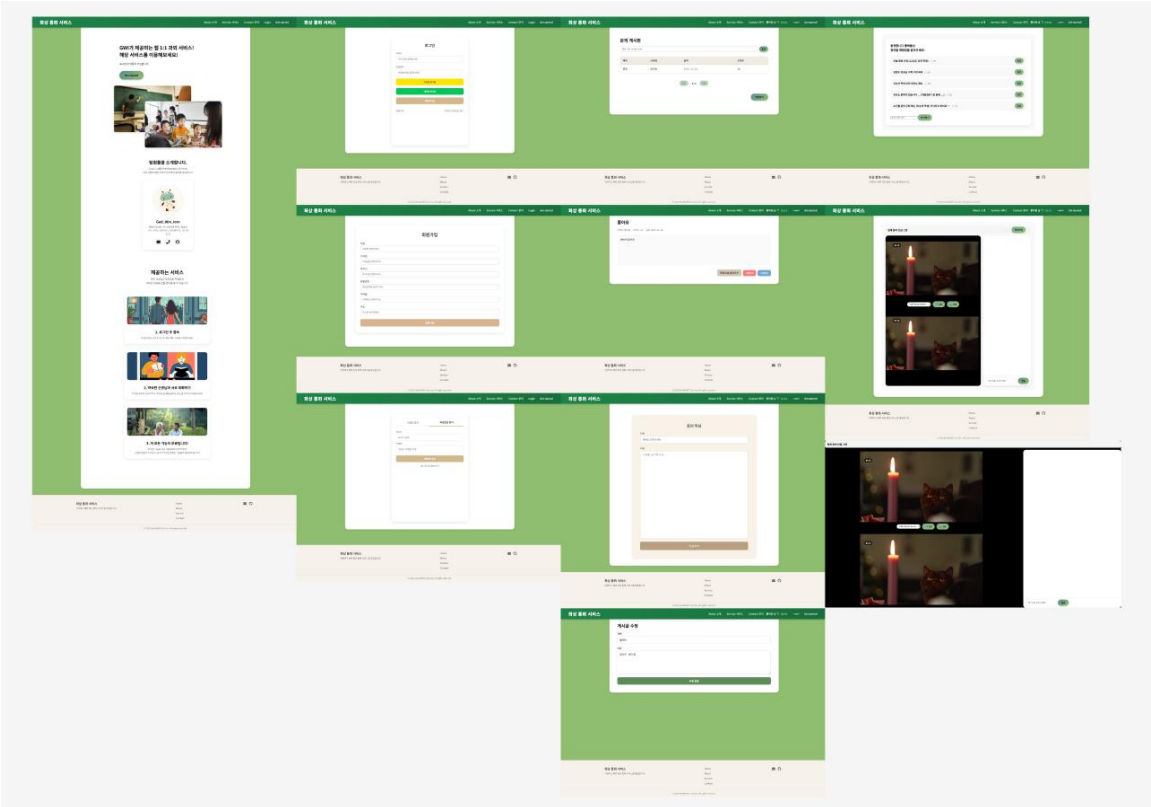
구분	상세 내용
Web 표준코딩 및 웹프로그래밍	- React 기반 SPA 개발 경험 - REST API 연동 및 비동기 통신 처리 - CORS / HTTPS 환경에서의 서비스 운영 경험
Database	- PostgreSQL 기반 테이블 설계 및 관계 모델링 - Spring Data JPA ORM 활용 데이터 처리 - Auditing 적용을 통한 생성·수정 이력 관리
시스템 운영 경험	- 사용자 / 관리자 권한 분리 설계 - HttpSession 및 JWT 인증 정책 설계 경험 - 배포 환경에서 발생한 인증·CORS 이슈 분석 & 해결 경험
학사·행정 시스템 연계 가능 역량	- 회원 관리, 권한 관리, 예약·신청 프로세스 구현 경험 -관리자 중심 워크플로우 설계 및 운영 관점 기능 개발

스킬



Notion 링크 : https://painted-pond-2ca.notion.site/2dd3ae9225c6804886a5fdc85e3be5dd

전산 포트폴리오 – HomeSchools 홈 스쿨 웹 서비스 [1]



사이트 링크 : https://home-schools-front-cfvk.vercel.app/

Git 페이지 링크 : https://github.com/shincharl/homeSchools?tab=readme-ov-file

학습 스터디 블로그 : https://chamchicoder.tistory.com/

1. 프로젝트 개요

HomeSchools(홈스쿨 프로그램) 은 실시간 채팅과 1:1 화상 수업 기능을 제공하는 웹 기반 학습 지원 서비스입니다. 사용자는 회원 가입 및 로그인 후, 선생님 또는 학생의 역할로 참여하여 실시간 소통을 기반으로 학습을 진행할 수 있습니다.

본 프로젝트는 Spring Boot 기반의 REST API 서버와 React + TypeScript 기반 SPA 구조로 Front-End 와 Back-End 를 분리하여 개발하였으며, 확장성과 유지보수를 고려한 시스템 아키텍처를 목표로 설계되었습니다.

특히 실시간성이 중요한 기능(화상 통신, 채팅)은 별도의 통신 구조를 적용하여 서버 부하 분산과 안정성을 확보하는 데 중점을 두었습니다.

2. 기획 배경 및 목표

최근에는 연령, 직업, 전공에 관계없이 누구나 지식을 공유하고 배울 수 있는 환경의 중요성이 점점 커지고 있습니다.

그러나 여전히 시간적 · 공간적 제약으로 인해 학습 기회에 접근하기 어려운 사용자들이 존재합니다.

이러한 문제를 해결하기 위해,

누구나 선생님이 될 수 있고, 누구나 학생이 될 수 있는 지식 공유 플랫폼을 목표로 본 프로젝트를 기획하였습니다.

(1) 기획 목표

- **사용자 역할에 제한이 없는 학습 참여 구조 제공**
- **실시간 화상 · 채팅을 통한 몰입도 높은 소통 환경 구현**
- **사용자 의견을 서비스 개선에 반영할 수 있는 구조 설계**

본 프로젝트는 단순한 기능 구현을 넘어,

지식이 순환되고 함께 성장하는 학습 환경 구축을 지향합니다.

3. 주요 기능

HomeSchools 는 1.0 버전 기준으로 핵심 기능 위주의 구조로 설계되었습니다.

(2) 주요 기능

·○ 회원 가입 / 로그인

- **JWT 기반 인증을 활용한 사용자 인증 기능**

·○ 1:1 화상 통신

- **WebRTC 기반 실시간 화상 수업 지원**

·○ 실시간 채팅

- **WebSocket 을 이용한 실시간 메시지 송수신**

·○ 건의 및 버그 신고 게시판

- **사용자 의견 수렴 및 서비스 개선을 위한 커뮤니케이션 창구**

사용자는 상황에 따라 선생님 또는 학생의 역할을 자유롭게 선택할 수 있으며, 서비스 개선 과정에 직접 참여할 수 있는 구조를 갖추고 있습니다.

4. 기술 스택

·○ Front-end

- **React: 컴포넌트 기반 UI 설계**
- **TypeScript: 정적 타입을 통한 안정적인 코드 작성**
- **Redux Toolkit: 로그인 상태, 사용자 정보, 화상 · 채팅 상태 관련 전역 관리**
- **Vite: 빠른 개발 환경 및 빌드 속도**
- **CSS Module : 컴포넌트 단위 스타일 관리**

·○ Back-end (Main API Server)

- **Java 17 (LTS)**
- **Spring Boot : REST API 서버 구축**
- **Spring Security : JWT 기반 인증 및 인가 처리**
- **JPA (Hibernate) : ORM 기반 데이터 처리**

·○ RealTime Server

- **Node.js**
- **WebSocket(Socket.IO) : 실시간 채팅 처리**

·○ Communication

- **WebRTC : 1:1 화상 통신 (P2P 구조)**

·○ Database

- **PostgreSQL : 사용자, 게시판 데이터 관리**

·○ Infra / Deployment

- **Heroku : Spring Boot API 서버 배포**
- **Vercel : React 프론트엔드 및 WebSocket 서버 배포**

5. 시스템 아키텍처

HomeSchools 는 API 서버와 실시간 통신 서버를 분리한 구조를 기반으로 설계되었습니다.

(3) 아키텍처 구성

·○ Spring Boot API 서버

- **회원 관리, 인증, 게시판 등 주요 비즈니스 로직 처리**
- **JWT 기반 인증 및 인가 담당**

·○ Node.js WebSocket 서버

- 실시간 채팅 전용 서버
- API 서버와 역할 분리를 통해 성능 및 확장성 확보

·○ WebRTC

- 클라이언트 간 P2P 방식의 1:1 화상 통신

·○ Redux Toolkit

- 인증 상태 및 실시간 기능 관련 전역 상태 관리

·○ PostgreSQL

- 서비스 전반의 데이터 영속성 관리

6. ERD 및 DB 구조

본 프로젝트는 사용자 중심 구조를 기반으로

Member(회원) 와 Contact(문의/건의 게시글) 엔티티를 중심으로 설계되었습니다.

(4) Member 엔티티

·○ Local 및 소셜 로그인 확장을 고려한 구조

·○ PROVIDER 필드를 통해 로그인 방식 구분

·○ TEMP_PASSWORD 필드를 활용한 보안 정책 적용

(5) Contact 엔티티

·○ 기능 건의, 버그 신고, 문의 사항 관리

·○ 회원과의 1:N 관계 설정

이를 통해 사용자 활동 이력 관리와 확장 가능한 인증 구조를 확보하였습니다.

7. 핵심 구현 내용

(6) JWT 기반 인증 및 인가

- SPA 구조에 적합한 Stateless 인증 방식 적용
- 로그인 시 토큰 발급, 요청 시 필터에서 토큰 검증
- 서버 상태 저장 없이 인증 유지 가능

(7) 소셜 로그인 확장 구조

- Provider Enum 및 Provider Id 분리 설계
- 기존 테이블 변경 없이 로그인 방식 확장 가능

(8) WebSocket 기반 실시간 채팅

- Node.js 기반 전용 서버 구축
- JWT 인증 기반 사용자 식별
- 비동기 처리에 강한 구조 적용

(9) WebRTC 기반 화상 통신

- 서버 부하 최소화를 위한 P2P 방식 채택
- 시그널링과 미디어 스트림 분리 처리
- 네트워크 환경을 고려한 연결 구조 설계

8. 트러블 슈팅 및 해결 경험

·○ WebRTC 연결 불안정 문제

- NAT 환경에서 ICE Candidate 교환 문제 발생
- 시그널링 순서 개선을 통해 안정적인 연결 확보

·○ SPA 환경에서 Socket 생명주기 문제

- 렌더링 시 Socket 객체 중복 생성
- 전역 Provider 구조 도입으로 단일 인스턴스 유지

·○ DataChannel 메시지 송수신 문제

- DataChannel 양쪽 저장 구조로 개선
- 메시지 처리 기준을 수신자 기준으로 통일

·○ 배포 환경 CORS 문제

- Spring Security CORS 설정 명확화
- 인증 헤더 허용 처리

·○ JWT 인증 구조 이해 개선

- 토큰 발급과 검증 역할 분리
- 로그인 시에만 ID/PW 검증 수행

9. 프로젝트를 통해 얻은 역량

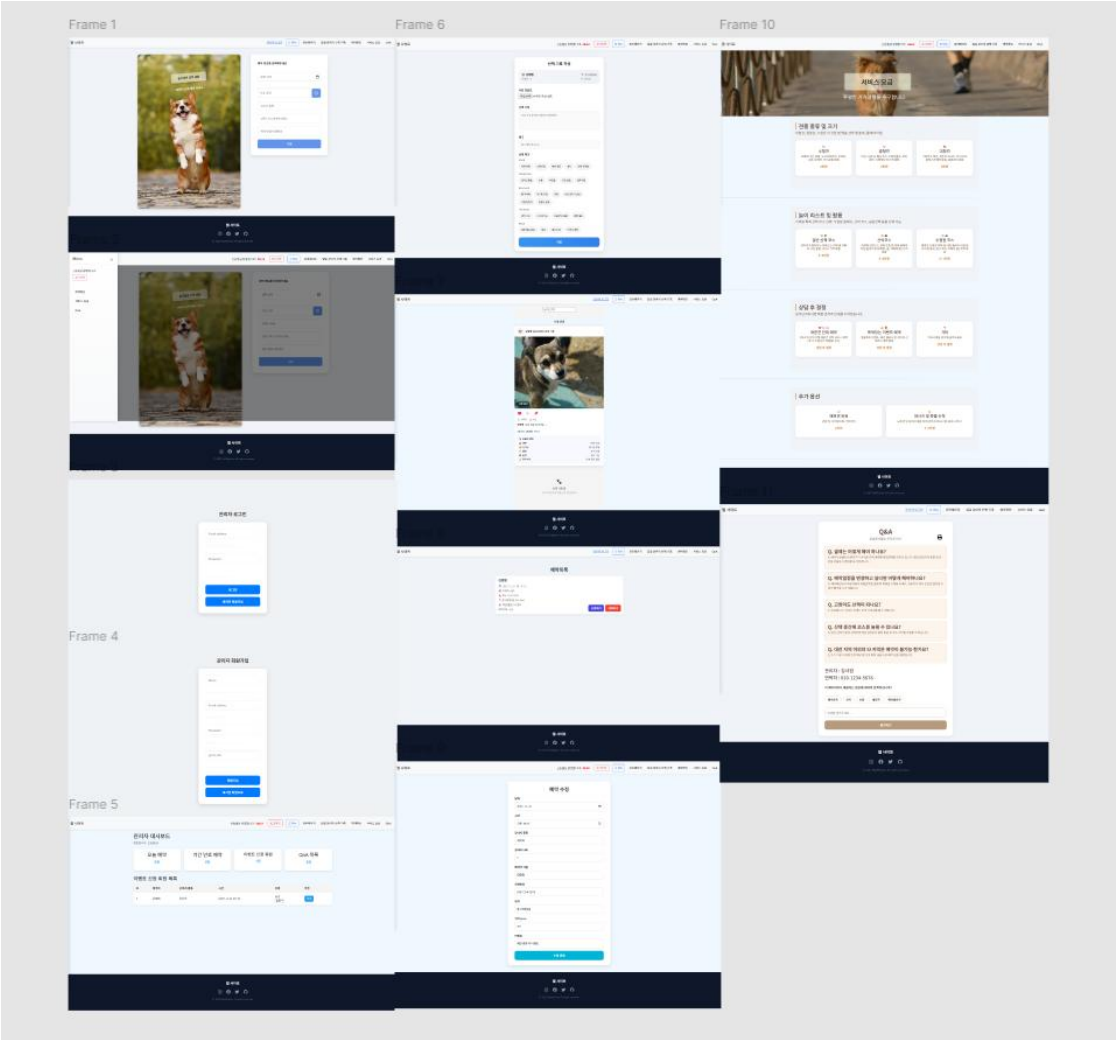
·○ 인증·보안 구조에 대한 이해

·○ 실시간 통신 및 네트워크 기반 서비스 설계 경험

·○ 역할 분리와 확장성을 고려한 시스템 아키텍처 설계

·○ 운영 환경에서 발생하는 문제 분석 및 해결 능력

전산 포트폴리오 – Dog-Go 강아지 산책 예약 서비스 [2]



사이트 링크 : https://home-schools-front-cfvk.vercel.app/

Git 페이지 링크 : https://github.com/shincharl/Dog_Go_Frontend (Front 코드)

Git 페이지 링크 : https://github.com/shincharl/Dog_Go_Frontend (Back 코드)

학습 스터디 블로그 : https://chamchicoder.tistory.com/

1. 프로젝트 개요

Dog-Go 는 반려견 보호자와 산책 서비스를 연결하는 웹 기반 예약 및 관리자 관리 시스템입니다.

사용자와 관리자의 역할을 명확히 분리하고, Spring Security 기반 HttpSession 인증 방식을 적용하여 실제 서비스 환경에서의 인증 흐름과 보안 구조를 고려해 구현하였습니다.

Front-end 와 Back-end 를 분리하여 배포하였으며, 배포 환경에서 발생하는 세션 인증 유지 문제, CORS 및 쿠키 이슈를 직접 분석, 해결한 경험을 담은 프로젝트입니다.

2. 기획 배경 및 목표

반려동물 서비스는 신뢰성과 관리 체계가 중요한 영역이며, 단순 예약 기능뿐 아니라 관리자의 운영 관점 기능이 반드시 필요하다고 판단했습니다.

이에 따라 본 프로젝트는 다음과 같은 목표를 가지고 기획되었습니다.

(1)기획 목표

- **사용자와 관리자의 역할을 명확히 분리한 서비스 구조 설계**
- **예약 -> 관리 -> 결과 기록으로 이어지는 운영 중심 흐름 구현**
- **실 배포 환경에서 동작 가능한 인증 및 보안 구조 적용**

3. 주요 기능

- **사용자 기능**

- **강아지 산책 예약 신청**
- **개인 예약 상태 조회/변경**

- **일자별 산책 결과 조회**
- **QnA 및 의견 등록**

·○ 관리자 기능 (ROLE_ADMIN)

- **오늘 예약 / 만료 예약 조회**
- **이벤트 및 산책 예약 관리**
- **전체 예약 상태 조회/변경**
- **일자별 산책 결과 작성 및 저장**
- **사용자 QnA 관리**

관리자 기능은 Spring Security 권한(Role) 기반 접근 제어를 통해 보호됩니다.

4. 기술 스택

·○ Front-end

- **HTML5 / CSS3**
- **JavaScript (ES6)**
- **React**
- **Axios**
- **Vite**
- **Vercel**

·○ Back-end

- **Java 21**
- **Spring Boot 3.5**
- **Spring Security**
- **JPA (Hibernate)**

·○ Database

- **PostgreSQL (Railway)**

·○ **Infra / Deployment**

- **Railway (Backend & Database)**
- **Vercel (Frontend)**
- **HTTPS 환경 구성**

5. 인증 및 보안 구조

본 프로젝트는 JWT 가 아닌 HttpSession 기반 인증 방식을 사용합니다.

이는 서버 중심의 인증 관리와 관리자 기능 보호에 적합하다고 판단하여 선택하였습니다.

(2) 인증 흐름

1. 사용자가 로그인 요청

2. AuthenticationManager 를 통한 사용자 인증

3. 인증 성공 시 SecurityContext 생성

4, HttpSessionSecurityContextRepository 를 통해 인증 정보 세션 저장

5. 서버에서 JSESSIONID 쿠키 반환

6. 이후 요청은 세션 쿠키 기반으로 인증 처리

(3) 보안 설정

·○ **Spring Security 기반 접근 제어**

·○ **관리자 API 는 ROLE_ADMIN 권한 필수**

·○ **CORS 설정 시 allowCredentials = true 적용**

·○ **프론트엔드 요청 시 withCredentials 옵션 적용**

6. DB 설계 및 엔티티 구조

(4) 공통 엔티티 (BaseEntity)

모든 주요 엔티티에서 상속받아 생성일, 수정일을 자동 관리합니다.

- **@MappedSuperclass 적용**
- **Spring Data JPA Auditing 사용**
- **중복 코드 제거 및 일관된 시간 관리**

(5) Member(회원)

사용자 및 관리자 계정을 관리하는 핵심 엔티티입니다.

- **이메일 기반 로그인**
- **EnumType.STRING 기반 권한(USER / ADMIN) 관리**
- **Spring Security 인증 대상 엔티티**

(6) QnA(문의)

사용자 문의 및 서비스 피드백을 관리합니다.

- **단순 문의 및 만족도 관리 목적**
- **Auditing 적용**

(7) Reservation (산책 예약)

강아지 산책 예약 정보를 관리하는 핵심 엔티티입니다.

- BaseEntity 상속
- @PrePersist 를 통한 예약 상태 기본값 설정
- Tracking 엔티티와 연관관계 구성

(8) Tracking (산책 기록)

산책 진행 중 작성되는 기록을 관리합니다.

- 대용량 텍스트 저장을 위한 @Lob 사용
- 위치, 상태 JSON 데이터 저장
- Cascade 및 orphanRemoval 적용
- 연관관계 편의 메서드 제공

(9) TrackingPhoto / TrackingTag

- 산책 사진 및 태그 관리
- JSON 순환 참조 방지 처리
- Tracking 과 1:N 관계 구성

7. 시스템 아키텍처 및 설계 포인트

- 프론트엔드 / 백엔드 완전 분리
- 세션 기반 인증을 통한 서버 중심 보안 관리
- 실 배포 환경에서의 인증 유지 문제 해결
- 관리자 권한 분리 및 접근 제어 구조 설계

8. 프로젝트를 통해 얻은 경험

- Spring Security 인증 흐름에 대한 실질적인 이해
- SecurityContext 와 HttpSession 동작 방식 분석
- CORS + 세션 쿠키 문제 해결 경험
- Railway / Vercel 기반 실 서비스 배포 경험
- 관리자 중심 시스템 설계 및 운영 관점 개발 역량