

COMPAGN O

Discover a vibrant community where pet lovers connect, share, and learn. From photo contests to local meetups, find all the information and resources you need to give your pet the best life. Join us today and make every moment with your furry friend unforgettable!



Dana | Yebin | Hyunsoo | Minjeong | Sanghyun | Dongjun | Hoeyoung

INDEX

01

Team Member

02

Workflow

03

Project Tech Stack

04

Why COMPAGNO

05

Requirements Specification

06

DB Modeling

07

Project Structure

08

Menu Structure

09

Code Review

10

Project Review

01

Team Member



Yebin

반려동물 동반 게시판
질문 게시판
Q&A



Sanghyun

유저(회원가입/로그인)
마이페이지



Dana

동물등록 게시판, 우리동네 게시판, 시터 게시판
일정 및 문서관리, PPT 작성



Hyunsoo

메인페이지
원데이클래스 게시판



Minjeong

입양공고 게시판
실종공고 게시판
쪽지



Dongjun

콘테스트 게시판
광고



Hoeyoung

제품정보 공유 게시판
공지사항

02

Workflow - Monthly

1. 전체 프로젝트 진행일정 계획

| 2024년 5월 | | | | | | |
|----------|--------|--------|--------|--------|--------|--------|
| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
| 28 | 29 | 30 | 31 | 4월 1일 | 2 | 3 |
| 2개 더보기 | | | | | | |
| 5 | | | | | | |
| 3개 더보기 | | | | | | |
| 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 3개 더보기 | 6개 더보기 | 5개 더보기 | 6개 더보기 | 2개 더보기 | 2개 더보기 | 3개 더보기 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 3개 더보기 | | | | | | |
| 26 | 27 | 28 | 29 | 30 | 5월 1일 | 2 |
| | | | | | | |

2024년 4월

| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
|--------|--------|--------|--------|--------|--------|--------|
| 31 | 4월 1일 | 2 | 3 | 4 | 5 | 6 |
| 5개 더보기 | | | | | | |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 1개 더보기 | 6개 더보기 | 2개 더보기 | 2개 더보기 | 2개 더보기 | 3개 더보기 | 1개 더보기 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 1개 더보기 | 6개 더보기 | 3개 더보기 | 2개 더보기 | 2개 더보기 | 3개 더보기 | 1개 더보기 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 1개 더보기 | 4개 더보기 | 3개 더보기 | 5개 더보기 | 2개 더보기 | 3개 더보기 | 1개 더보기 |
| 28 | 29 | 30 | 5월 1일 | 2 | 3 | 4 |
| | | | | | | |

02

Workflow - Weekly

2. 업무 세분화(주간)

2. 업무 세분화(주간)

Final Weekly Report - 1st week

| | This Week (4.1 ~ 4.7) | Next Week (4.8 ~ 4.14) |
|-----|--|--|
| 설정문 | <ul style="list-style-type: none"> ▣ 2024-04-01~2024-04-07 주간 학습 목표 ▣ 2024-04-01~2024-04-07 주제 | <ul style="list-style-type: none"> ■ 어려운 주제 전환 ■ 개인화된 학습 ■ 대체 활동의 설정 시기 ■ 통합활동에 대한 구현 기획 |
| 설계안 | <ul style="list-style-type: none"> ▣ 2024-04-01~2024-04-07 주제 ▣ 2024-04-01~2024-04-07 주제 | <ul style="list-style-type: none"> ■ GAA 7주 기간 ■ GAA DB ■ 차별화 + 혼용교과와 같이 조건 설정 시스템 적용 ■ 혼용 교과에서 차이가 있는 곳 |
| 집행수 | <ul style="list-style-type: none"> □ 예상 학습지나 선별학부 분별 확립 □ 학습지 활용으로 학생별 수준설정 | <ul style="list-style-type: none"> ■ 혼용교과와 SameDB 내구현 ■ ONECLASSDB 7주 기간 및 적용 |
| 승인률 | <ul style="list-style-type: none"> ▣ 2024-04-01~2024-04-07 주간 학습 목표 → 기본 (A), 제작자별, 투표 업로드 | <ul style="list-style-type: none"> ■ 혼용 교과(영어, 수학, 과학) ■ 혼용 교과 기본 기준 ■ 혼용 교과(NOTEBOOK) |
| 파악방 | <ul style="list-style-type: none"> ▣ 2024-04-01~2024-04-07 주간 학습 목표 ▣ 2024-04-01~2024-04-07 주간 학습 목표 ■ 기본 기준 | <ul style="list-style-type: none"> ■ 혼용교과의 흐름 기록 단위 구현 |
| 정찰통 | <ul style="list-style-type: none"> ▣ 2024-04-01~2024-04-07 주간 학습 목표 ▣ 2024-04-01~2024-04-07 주간 학습 목표 ▣ 2024-04-01~2024-04-07 주간 학습 목표 | <ul style="list-style-type: none"> ■ Reach-Gui 초기 설정 시기 ■ 혼용 기준을 구현 시기 |
| 교류회 | <ul style="list-style-type: none"> ▣ 2024-04-01~2024-04-07 주간 학습 목표 | * |
| | <ul style="list-style-type: none"> ■ 1주 단위로 학습 목표에 서비스 활용법 적용 ■ 단기 학습지 설정 시기 기준 ■ 혼용교과와 혼용교과 분야, 초대 ■ 혼용 교과 설정 분야, 기준 | |

2004

Final Weekly Report - 3rd week

| | This Week (4.15 ~ 4.21) | Next Week (4.22 ~ 4.28) |
|-----|--|--|
| 교내전 | <ul style="list-style-type: none"> ▣ [교내전] 교내전 카페(온라인) ▣ [교내전] 교내전 카페(온라인), FAQ 기술 지원 ▣ [교내전] 교내전 카페(온라인) ▣ [교내전] 교내전 카페(온라인), FAQ 기술 지원 | <ul style="list-style-type: none"> ■ [교내전] 교내전 카페(온라인) ■ [교내전] 교내전 카페(온라인), FAQ 기술 지원 |
| 교외전 | <ul style="list-style-type: none"> ■ [교외전] 티켓 구매 사이트 | <ul style="list-style-type: none"> ■ [교외전] 티켓 구매 사이트 ■ [교외전] 티켓 구매 사이트 |
| 집행수 | <ul style="list-style-type: none"> □ OneDayClass 기술지원 사이트 □ OneDayClass 기술지원 사이트 □ OneDayClass 기술지원 지원 사이트 □ OneDayClass 기술지원 지원 사이트 | <ul style="list-style-type: none"> ■ OneDayClass 기술지원 사이트 |
| 운영팀 | <ul style="list-style-type: none"> □ 도록방문자수증 기술 구현 프로젝트 □ 도록방문자수증 기준화 구현 사이트 | <ul style="list-style-type: none"> ■ 도록방문자수증 기술 구현 프로젝트 ■ 도록방문자수증 기술 구현 프로젝트 |
| 제작팀 | <ul style="list-style-type: none"> □ [제작팀] 제작팀 카페(온라인) □ [제작팀] 제작팀 카페(온라인) □ [제작팀] 제작팀 카페(온라인) | <ul style="list-style-type: none"> ■ [제작팀] 제작팀 카페(온라인) ■ [제작팀] 제작팀 카페(온라인) |
| 평생동 | <ul style="list-style-type: none"> ■ [평생동] 평생동 카페(온라인) ■ [평생동] 평생동 카페(온라인) ■ [평생동] 평생동 카페(온라인) | <ul style="list-style-type: none"> ■ [평생동] 평생동 카페(온라인) |
| 체육팀 | <ul style="list-style-type: none"> □ [체육팀] 체육팀 카페(온라인) □ [체육팀] 체육팀 카페(온라인) | <ul style="list-style-type: none"> ■ [체육팀] 체육팀 카페(온라인) |
| 기획 | <ul style="list-style-type: none"> ■ [기획팀] 웹사이트-온라인 ■ [기획팀] 웹사이트-온라인 ■ [기획팀] 웹사이트-온라인 ■ [기획팀] 웹사이트-온라인 | <ul style="list-style-type: none"> ■ [기획팀] 웹사이트-온라인 ■ [기획팀] 웹사이트-온라인 ■ [기획팀] 웹사이트-온라인 ■ [기획팀] 웹사이트-온라인 |

2014.4.3

Final Weekly Report - 4th week

第11页

02

Workflow - Daily

3. 일일 진행업무 공유

| 5.6 ~ 5.10 Daily Report Summary | | | |
|---------------------------------|-----|-----|-----|
| 날짜 | 제작일 | 제작일 | 제작일 |
| 5.6.30 | | | |
| 5.7.01 | | | |
| 5.7.02 | | | |
| 5.7.03 | | | |
| 5.7.04 | | | |
| 5.7.05 | | | |
| 5.7.06 | | | |
| 5.7.07 | | | |
| 5.7.08 | | | |
| 5.7.09 | | | |
| 5.7.10 | | | |
| 5.7.11 | | | |
| 5.7.12 | | | |
| 5.7.13 | | | |
| 5.7.14 | | | |
| 5.7.15 | | | |
| 5.7.16 | | | |
| 5.7.17 | | | |
| 5.7.18 | | | |
| 5.7.19 | | | |
| 5.7.20 | | | |
| 5.7.21 | | | |
| 5.7.22 | | | |
| 5.7.23 | | | |
| 5.7.24 | | | |
| 5.7.25 | | | |
| 5.7.26 | | | |
| 5.7.27 | | | |
| 5.7.28 | | | |
| 5.7.29 | | | |
| 5.7.30 | | | |
| 5.7.31 | | | |
| 5.8.01 | | | |
| 5.8.02 | | | |
| 5.8.03 | | | |
| 5.8.04 | | | |
| 5.8.05 | | | |
| 5.8.06 | | | |
| 5.8.07 | | | |
| 5.8.08 | | | |
| 5.8.09 | | | |
| 5.8.10 | | | |
| 5.8.11 | | | |
| 5.8.12 | | | |
| 5.8.13 | | | |
| 5.8.14 | | | |
| 5.8.15 | | | |
| 5.8.16 | | | |
| 5.8.17 | | | |
| 5.8.18 | | | |
| 5.8.19 | | | |
| 5.8.20 | | | |
| 5.8.21 | | | |
| 5.8.22 | | | |
| 5.8.23 | | | |
| 5.8.24 | | | |
| 5.8.25 | | | |
| 5.8.26 | | | |
| 5.8.27 | | | |
| 5.8.28 | | | |
| 5.8.29 | | | |
| 5.8.30 | | | |
| 5.8.31 | | | |
| 5.9.01 | | | |
| 5.9.02 | | | |
| 5.9.03 | | | |
| 5.9.04 | | | |
| 5.9.05 | | | |
| 5.9.06 | | | |
| 5.9.07 | | | |
| 5.9.08 | | | |
| 5.9.09 | | | |
| 5.9.10 | | | |
| 5.9.11 | | | |
| 5.9.12 | | | |
| 5.9.13 | | | |
| 5.9.14 | | | |
| 5.9.15 | | | |
| 5.9.16 | | | |
| 5.9.17 | | | |
| 5.9.18 | | | |
| 5.9.19 | | | |
| 5.9.20 | | | |
| 5.9.21 | | | |
| 5.9.22 | | | |
| 5.9.23 | | | |
| 5.9.24 | | | |
| 5.9.25 | | | |
| 5.9.26 | | | |
| 5.9.27 | | | |
| 5.9.28 | | | |
| 5.9.29 | | | |
| 5.9.30 | | | |
| 5.9.31 | | | |
| 5.10.01 | | | |
| 5.10.02 | | | |
| 5.10.03 | | | |
| 5.10.04 | | | |
| 5.10.05 | | | |
| 5.10.06 | | | |
| 5.10.07 | | | |
| 5.10.08 | | | |
| 5.10.09 | | | |
| 5.10.10 | | | |
| 5.10.11 | | | |
| 5.10.12 | | | |
| 5.10.13 | | | |
| 5.10.14 | | | |
| 5.10.15 | | | |
| 5.10.16 | | | |
| 5.10.17 | | | |
| 5.10.18 | | | |
| 5.10.19 | | | |
| 5.10.20 | | | |
| 5.10.21 | | | |
| 5.10.22 | | | |
| 5.10.23 | | | |
| 5.10.24 | | | |
| 5.10.25 | | | |
| 5.10.26 | | | |
| 5.10.27 | | | |
| 5.10.28 | | | |
| 5.10.29 | | | |
| 5.10.30 | | | |
| 5.10.31 | | | |
| 5.11.01 | | | |
| 5.11.02 | | | |
| 5.11.03 | | | |
| 5.11.04 | | | |
| 5.11.05 | | | |
| 5.11.06 | | | |
| 5.11.07 | | | |
| 5.11.08 | | | |
| 5.11.09 | | | |
| 5.11.10 | | | |
| 5.11.11 | | | |
| 5.11.12 | | | |
| 5.11.13 | | | |
| 5.11.14 | | | |
| 5.11.15 | | | |
| 5.11.16 | | | |
| 5.11.17 | | | |
| 5.11.18 | | | |
| 5.11.19 | | | |
| 5.11.20 | | | |
| 5.11.21 | | | |
| 5.11.22 | | | |
| 5.11.23 | | | |
| 5.11.24 | | | |
| 5.11.25 | | | |
| 5.11.26 | | | |
| 5.11.27 | | | |
| 5.11.28 | | | |
| 5.11.29 | | | |
| 5.11.30 | | | |
| 5.11.31 | | | |
| 5.12.01 | | | |
| 5.12.02 | | | |
| 5.12.03 | | | |
| 5.12.04 | | | |
| 5.12.05 | | | |
| 5.12.06 | | | |
| 5.12.07 | | | |
| 5.12.08 | | | |
| 5.12.09 | | | |
| 5.12.10 | | | |
| 5.12.11 | | | |
| 5.12.12 | | | |
| 5.12.13 | | | |
| 5.12.14 | | | |
| 5.12.15 | | | |
| 5.12.16 | | | |
| 5.12.17 | | | |
| 5.12.18 | | | |
| 5.12.19 | | | |
| 5.12.20 | | | |
| 5.12.21 | | | |
| 5.12.22 | | | |
| 5.12.23 | | | |
| 5.12.24 | | | |
| 5.12.25 | | | |
| 5.12.26 | | | |
| 5.12.27 | | | |
| 5.12.28 | | | |
| 5.12.29 | | | |
| 5.12.30 | | | |
| 5.12.31 | | | |
| 5.13.01 | | | |
| 5.13.02 | | | |
| 5.13.03 | | | |
| 5.13.04 | | | |
| 5.13.05 | | | |
| 5.13.06 | | | |
| 5.13.07 | | | |
| 5.13.08 | | | |
| 5.13.09 | | | |
| 5.13.10 | | | |
| 5.13.11 | | | |
| 5.13.12 | | | |
| 5.13.13 | | | |
| 5.13.14 | | | |
| 5.13.15 | | | |
| 5.13.16 | | | |
| 5.13.17 | | | |
| 5.13.18 | | | |
| 5.13.19 | | | |
| 5.13.20 | | | |
| 5.13.21 | | | |
| 5.13.22 | | | |
| 5.13.23 | | | |
| 5.13.24 | | | |
| 5.13.25 | | | |
| 5.13.26 | | | |
| 5.13.27 | | | |
| 5.13.28 | | | |
| 5.13.29 | | | |
| 5.13.30 | | | |
| 5.13.31 | | | |
| 5.14.01 | | | |
| 5.14.02 | | | |
| 5.14.03 | | | |
| 5.14.04 | | | |
| 5.14.05 | | | |
| 5.14.06 | | | |
| 5.14.07 | | | |
| 5.14.08 | | | |
| 5.14.09 | | | |
| 5.14.10 | | | |
| 5.14.11 | | | |
| 5.14.12 | | | |
| 5.14.13 | | | |
| 5.14.14 | | | |
| 5.14.15 | | | |
| 5.14.16 | | | |
| 5.14.17 | | | |
| 5.14.18 | | | |
| 5.14.19 | | | |
| 5.14.20 | | | |
| 5.14.21 | | | |
| 5.14.22 | | | |
| 5.14.23 | | | |
| 5.14.24 | | | |
| 5.14.25 | | | |
| 5.14.26 | | | |
| 5.14.27 | | | |
| 5.14.28 | | | |
| 5.14.29 | | | |
| 5.14.30 | | | |
| 5.14.31 | | | |
| 5.15.01 | | | |
| 5.15.02 | | | |
| 5.15.03 | | | |
| 5.15.04 | | | |
| 5.15.05 | | | |
| 5.15.06 | | | |
| 5.15.07 | | | |
| 5.15.08 | | | |
| 5.15.09 | | | |
| 5.15.10 | | | |
| 5.15.11 | | | |
| 5.15.12 | | | |
| 5.15.13 | | | |
| 5.15.14 | | | |
| 5.15.15 | | | |
| 5.15.16 | | | |
| 5.15.17 | | | |
| 5.15.18 | | | |
| 5.15.19 | | | |
| 5.15.20 | | | |
| 5.15.21 | | | |
| 5.15.22 | | | |
| 5.15.23 | | | |
| 5.15.24 | | | |
| 5.15.25 | | | |
| 5.15.26 | | | |
| 5.15.27 | | | |
| 5.15.28 | | | |
| 5.15.29 | | | |
| 5.15.30 | | | |
| 5.15.31 | | | |
| 5.16.01 | | | |
| 5.16.02 | | | |
| 5.16.03 | | | |
| 5.16.04 | | | |
| 5.16.05 | | | |
| 5.16.06 | | | |
| 5.16.07 | | | |
| 5.16.08 | | | |
| 5.16.09 | | | |
| 5.16.10 | | | |
| 5.16.11 | | | |
| 5.16.12 | | | |
| 5.16.13 | | | |
| 5.16.14 | | | |
| 5.16.15 | | | |
| 5.16.16 | | | |
| 5.16.17 | | | |
| 5.16.18 | | | |
| 5.16.19 | | | |
| 5.16.20 | | | |
| 5.16.21 | | | |
| 5.16.22 | | | |
| 5.16.23 | | | |
| 5.16.24 | | | |
| 5.16.25 | | | |
| 5.16.26 | | | |
| 5.16.27 | | | |
| 5.16.28 | | | |
| 5.16.29 | | | |
| 5.16.30 | | | |
| 5.16.31 | | | |
| 5.17.01 | | | |
| 5.17.02 | | | |
| 5.17.03 | | | |
| 5.17.04 | | | |
| 5.17.05 | | | |
| 5.17.06 | | | |
| 5.17.07 | | | |
| 5.17.08 | | | |
| 5.17.09 | | | |
| 5.17.10 | | | |
| 5.17.11 | | | |
| 5.17.12 | | | |
| 5.17.13 | | | |
| 5.17.14 | | | |
| 5.17.15 | | | |
| 5.17.16 | | | |
| 5.17.17 | | | |
| 5.17.18 | | | |
| 5.17.19 | | | |
| 5.17.20 | | | |
| 5.17.21 | | | |
| 5.17.22 | | | |
| 5.17.23 | | | |
| 5.17.24 | | | |
| 5.17.25 | | | |
| 5.17.26 | | | |
| 5.17.27 | | | |
| 5.17.28 | | | |
| 5.17.29 | | | |
| 5.17.30 | | | |
| 5.17.31 | | | |
| 5.18.01 | | | |
| 5.18.02 | | | |
| 5.18.03 | | | |
| 5.18.04 | | | |
| 5.18.05 | | | |
| 5.18.06 | | | |
| 5.18.07 | | | |
| 5.18.08 | | | |
| 5.18.09 | | | |
| 5.18.10 | | | |
| 5.18.11 | | | |
| 5.18.12 | | | |
| 5.18.13 | | | |
| 5.18.14 | | | |
| 5.18.15 | | | |
| 5.18.16 | | | |
| 5.18.17 | | | |
| 5.18.18 | | | |
| 5.18.19 | | | |
| 5.18.20 | | | |
| 5.18.21 | | | |
| 5.18.22 | | | |
| 5.18.23 | | | |
| 5.18.24 | | | |
| 5.18.25 | | | |
| 5.18.26 | | | |
| 5.18.27 | | | |
| 5.18.28 | | | |
| 5.18.29 | | | |
| 5.18.30 | | | |
| 5.18.31 | | | |
| 5.19.01 | | | |
| 5.19.02 | | | |
| 5.19.03 | | | |
| 5.19.04 | | | |
| 5.19.05 | | | |
| 5.19.06 | | | |
| 5.19.07 | | | |
| 5.19.08 | | | |
| 5.19.09 | | | |
| 5.19.10 | | | |
| 5.19.11 | | | |
| 5.19.12 | | | |
| 5.19.13 | | | |
| 5.19.14 | | | |
| 5.19.15 | | | |
| 5.19.16 | | | |
| 5.19.17 | | | |
| 5.19.18 | | | |
| 5.19.19 | | | |
| 5.19.20 | | | |
| 5.19.21 | | | |
| 5.19.22 | | | |
| 5.19.23 | | | |
| 5.19.24 | | | |
| 5.19.25 | | | |
| 5.19.26 | | | |
| 5.19.27 | | | |
| 5.19.28 | | | |
| 5.19.29 | | | |
| 5.19.30 | | | |
| 5.19.31 | | | |
| 5.20.01 | | | |
| 5.20.02 | | | |
| 5.20.03 | | | |
| 5.20.04 | | | |
| 5.20.05 | | | |
| 5.20.06 | | | |
| 5.20.07 | | | |
| 5.20.08 | | | |
| 5.20.09 | | | |
| 5.20.10 | | | |
| 5.20.11 | | | |
| 5.20.12 | | | |
| 5.20.13 | | | |
| 5.20.14 | | | |
| 5.20.15 | | | |
| 5.20.16 | | | |
| 5.20.17 | | | |
| 5.20.18 | | | |
| 5.20.19 | | | |
| 5.20.20 | | | |
| 5.20.21 | | | |
| 5.20.22 | | | |
| 5.20.23 | | | |
| 5.20.24 | | | |
| 5.20.25 | | | |
| 5.20.26 | | | |
| 5.20.27 | | | |
| 5.20.28 | | | |
| 5.20.29</ | | | |

03

Project Tech Stack

Front-end



Back-end



Data-Base



Server



Tool



04

Why COMPAGNO

1. 반려인 증가 추세

다양한 반려동물 관련 정보 제공으로 반려동물 소유자들의 편의 도모

1. 유기동물 문제 해결

유기동물 입양 공고 및 정보 제공을 통해 유기동물 입양 촉진

1. 책임감 있는 반려문화 정착

반려동물 등록, 실종 신고 등 관련 정보를 제공하여 반려동물에 대한 인식 개선

1. 커뮤니티 형성 및 소통

반려동물 소유자들이 지역 주민들과 경험과 정보를 공유할 수 있는 플랫폼 제공

반려동물 용품 및 서비스에 대한 리뷰와 최신 정보를 공유

반려동물 동반 장소(카페, 공원 등) 소개로 여가 활동 지원

반려동물 양육에 대한 이해를 높이고, 반려동물을 키우는 사람들이 서로 연결되어 따뜻한 공동체

형성

책임 있는 반려문화를 조성하고 반려동물의 안전을 지키며, 유기동물 문제 해결에 기여

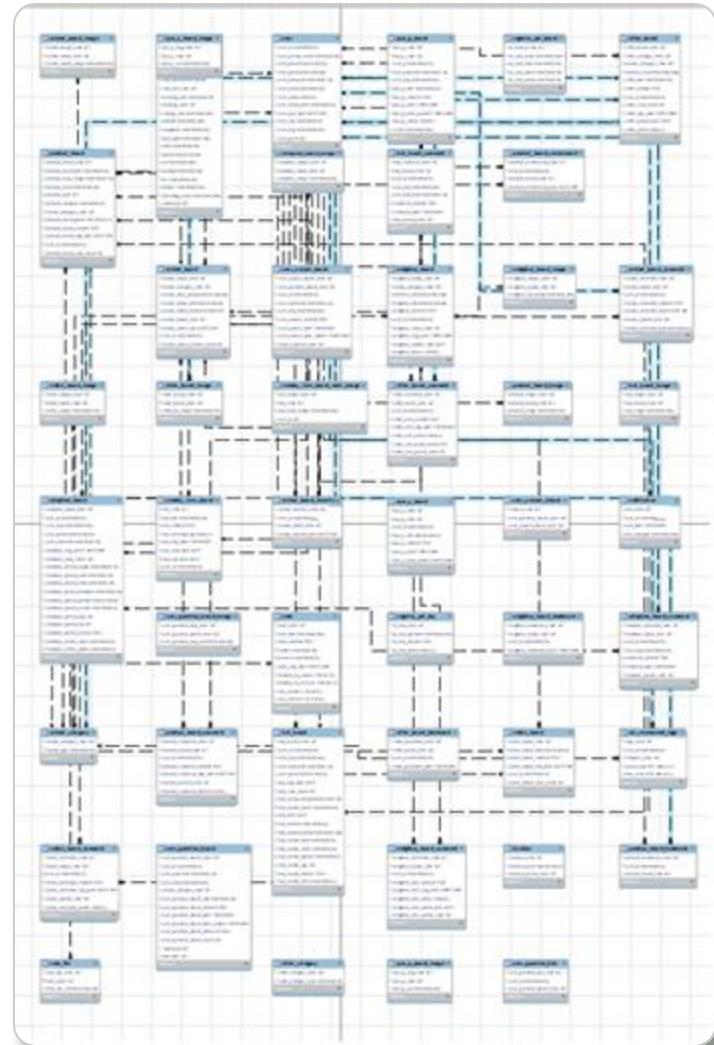
05

Requirements Specification

| No. | 구분 | 시작자 | Depth 1 | Depth 2 | Depth 3 | 발달자 | 종합도 | 기술주제 | 요구사항 내용 | 정립날짜 | 회고 |
|-----|--------------|--------------|---------|---------|---------|-----|-----|------|--|------------|----|
| 1 | 반려동물 출판 계약서 | 관리자, 운영, 마케팅 | 초록 | 설계 | 설계 | 설계 | 설계 | 설계 | 반려동물 출판 관리시스템은 관리 조건과 맞게 출력되는. | 2024. 4. 3 | |
| 4 | 관리자, 운영, 마케팅 | 초록 | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 인증코드는 운영자를 확인하는. | | |
| 5 | 관리자, 운영, 마케팅 | 초록 | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 관리자 혹은 사용자는 개인적인 정보를 등록하는. | | |
| 6 | 관리자, 운영, 마케팅 | 초록 | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 개인적인 접근 시 개인 코드로 세션 관리하고, 세션 만료로 인한 세션 해제 및 초기화 할 수 있다. | | |
| 8 | 관리자, 운영, 마케팅 | 초록 | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 출판된 책의 일부 영상을 활용 시 기기마다 접근, 출판자가 출판 시 기기마다 접근, 출판자가 출판 시 기기마다 접근. | | |
| 47 | 우편물등록 특사항 | 관리자, 운영, 마케팅 | 초록 | 설계 | 설계 | 설계 | 설계 | 설계 | 우편물 등록은 출판된 책과 함께 우편물을 등록하는. | | |
| 52 | 관리자, 운영, 마케팅 | 초록 | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 인증코드는 개인 정보 내용을 조정하는. | | |
| 53 | 관리자, 운영 | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 인증코드는 개인 정보 내용을 조정하는. | | |
| 54 | 관리자, 운영 | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 인증코드는 개인 정보 내용을 조정하는. | | |
| 57 | 관리자, 운영 | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 인증코드는 개인 정보 내용을 조정하는. | | |
| 70 | 제작제작자 | 운영, 마케팅 | | 설계 | 설계 | 설계 | 설계 | 설계 | 제작자는 제작자에게 출판된 책에 대한 사용권을 갖는다. 해당 책은 판권자에게 출판권을 갖는다. 해당 책은 출판권자에게 출판권을 갖는다. 해당 책은 출판권자에게 출판권을 갖는다. | | |
| 71 | 운영, 마케팅 | | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 제작자는 제작자에게 출판된 책에 대한 사용권을 갖는다. 해당 책은 판권자에게 출판권을 갖는다. 해당 책은 출판권자에게 출판권을 갖는다. | | |
| 73 | 운영, 마케팅 | | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 제작자는 제작자에게 출판된 책에 대한 사용권을 갖는다. 해당 책은 판권자에게 출판권을 갖는다. 해당 책은 출판권자에게 출판권을 갖는다. | | |
| 74 | 운영, 마케팅 | | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 제작자는 제작자에게 출판된 책에 대한 사용권을 갖는다. 해당 책은 판권자에게 출판권을 갖는다. 해당 책은 출판권자에게 출판권을 갖는다. | | |
| 119 | 책자 | 책자 | 책자 작성 | | | 설계 | 설계 | 설계 | 1. 제작된 책자 내용은 출판된 책과 일치해야 한다. 2. 제작자는 출판된 책과 일치하는 책자 내용을 갖는다. 해당 책은 출판권자에게 출판권을 갖는다. | | |
| 128 | 책자 | 책자 | 책자 제작 | 설계 | 설계 | 설계 | 설계 | 설계 | 3. 책자 제작자는 책자 내용을 조정하는. | | |
| 129 | 책자 | | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 4. 책자 제작자는 책자 내용을 조정하는. | | |
| 130 | 책자 | | 책자 보기 | 설계 | 설계 | 설계 | 설계 | 설계 | 1. 제작자는 책자 내용을 갖는다. 해당 책은 출판권자에게 출판권을 갖는다. 2. 출판권자는 책자 내용을 갖는다. 해당 책은 출판권자에게 출판권을 갖는다. | | |
| 132 | 책자 | | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 3. 출판권자는 책자 내용을 갖는다. 해당 책은 출판권자에게 출판권을 갖는다. 4. 제작자는 책자 내용을 갖는다. 해당 책은 출판권자에게 출판권을 갖는다. | | |
| 133 | 책장가격 | 비판형 | 책장가격 | | | 설계 | 설계 | 설계 | 제작자는 책장가격을 설정하는. | | |
| 140 | 제작판 | 비판형 | 제작판 | | | 설계 | 설계 | 설계 | 제작자는 책장가격을 설정하는. | | |
| 141 | 제작자 | 제작판 | 제작판 | 설계 | 설계 | 설계 | 설계 | 설계 | 제작자는 책장가격을 설정하는. | | |
| 142 | 제작자 | 제작판 | 제작판 | 설계 | 설계 | 설계 | 설계 | 설계 | 제작자는 책장가격을 설정하는. | | |
| 144 | 책장 | | 책장 정렬 | 설계 | 설계 | 설계 | 설계 | 설계 | 제작자는 책장을 정렬하는. | | |
| 156 | 자료제작판 | 운영, 마케팅 | 자료제작 | 설계 | 설계 | 설계 | 설계 | 설계 | 제작자는 책장을 정렬하는. | | |
| 157 | 운영, 마케팅 | 자료제작 | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 제작자는 책장을 정렬하는. | | |
| 158 | 운영 | | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 제작자는 책장을 정렬하는. | | |
| 169 | 판교 | 운영, 마케팅 | 판교 | 설계 | 설계 | 설계 | 설계 | 설계 | 1. 제작자는 판교 제품으로 출판되는 책을 출판하는. 2. 제작자는 판교 제품으로 출판되는 책을 출판하는. | | |
| 170 | 제작정보 제공 계약서 | 운영, 마케팅 | 판교 | 설계 | 설계 | 설계 | 설계 | 설계 | 제작자는 판교 제품으로 출판하는. | | |
| 177 | 운영, 마케팅 | | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 제작자는 판교 제품으로 출판하는. | | |
| 178 | 운영 | | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 제작자는 판교 제품으로 출판하는. | | |
| 179 | 운영 | | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 제작자는 판교 제품으로 출판하는. | | |
| 180 | 운영 | | 설계 | 설계 | 설계 | 설계 | 설계 | 설계 | 제작자는 판교 제품으로 출판하는. | | |

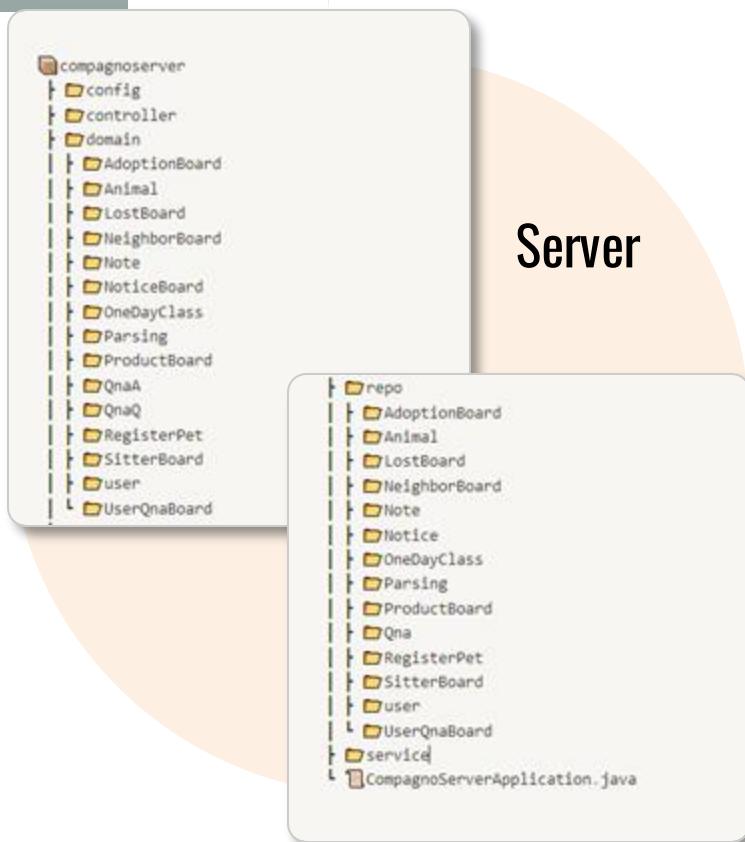
06

DB Modeling

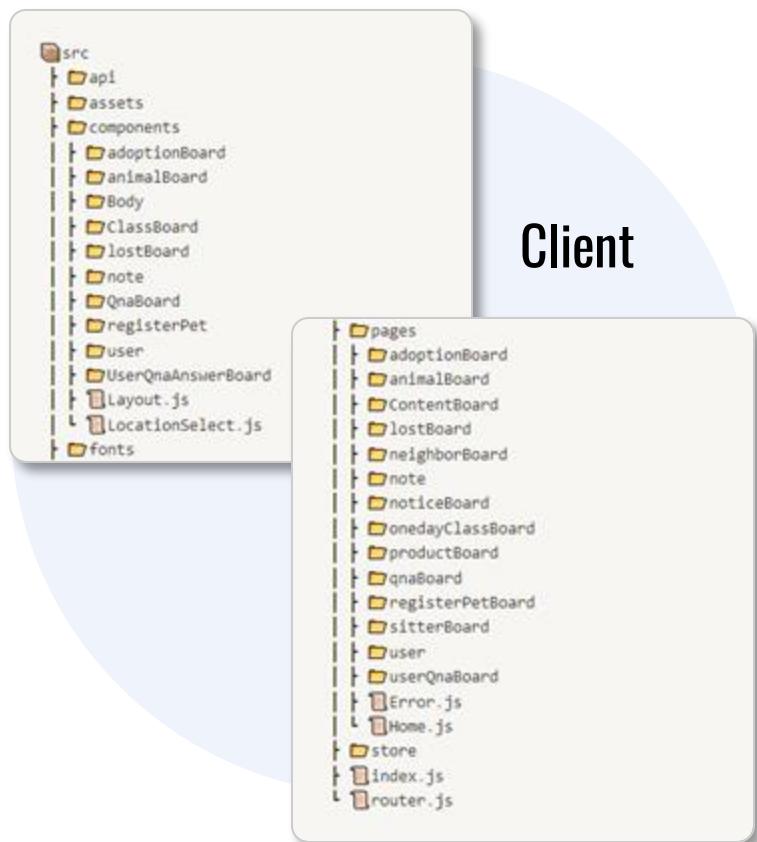


07

Project Structure



Server



Client

08

Menu Structure





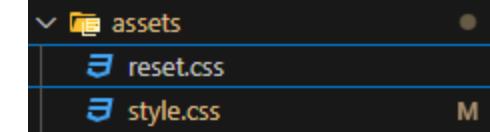
09

Code Review

메인페이지 UI & Classify



관심사 분리. 코드 재사용성 증가. 유지보수의 용이성을 향상 시키기 위해 분리



메인페이지

Section 1

section 1

Compagno

등용 등록 구조 등록 개시판 서비스 공지사항 signup login

```
.image-wrap {
    transition: 1s ease-out;
    transition-delay: 0.2s;
    position: relative;
    width: auto;
    overflow: hidden;
    clip-path: polygon(0 100%, 100% 100%, 100% 0, 0 100%);
    visibility: hidden;
}

.image-wrap img {
    transform: scale(1.3);
    transition: 1s ease-out;
}

.animating .image-wrap {
    clip-path: polygon(0 0, 100% 0, 100% 100%, 0 100%);
    visibility: visible;
    transform: skewY(0);
    height: 70vh;
}

.animating img {
    transform: scale(1);
    transition: 4s ease-out;
}
```

```
.fadeup {
    opacity: 0;
    transition: 0.4s ease-out;
    transform: translateY(40px);
}

.fading-up {
    opacity: 1;
    transition: 1.5s ease-out;
    transform: translateY(0px);
    transition-delay: 1.2s;
}
```

메인페이지

Section 1

```

const Section1 = () => {
  const [,] = useState();
  const options = {
    root: null,
    rootMargin: "0px",
    threshold: 0.4,
  };
  useEffect(() => {
    document.addEventListener(
      "DOMContentLoaded",
      []
    );
    let revealCallback = (entries) => {
      entries.forEach((entry) => {
        let container = entry.target;
        if (entry.isIntersecting) {
          container.classList.add("animating");
          return;
        }
        if (entry.boundingClientRect.top > 0) {
          container.classList.remove("animating");
        }
      });
    };
    let revealObserver =
      new IntersectionObserver(revealCallback, options);
    document.querySelectorAll(".reveal").forEach((reveal) => {
      revealObserver.observe(reveal);
    });
    let fadeupCallback = (entries) => {
      entries.forEach((entry) => {
        let container = entry.target;
        container.classList.add("not-fading-up");
        if (entry.isIntersecting) {
          container.classList.add("fading-up");
          return;
        }
        if (entry.boundingClientRect.top > 0) {
          container.classList.remove("fading-up");
        }
      });
    };
    let fadeupObserver =
      new IntersectionObserver(fadeupCallback, options);
    document.querySelectorAll(".fadeup").forEach((fadeup) => {
      fadeupObserver.observe(fadeup);
    });
  });
}

```

```

<section id="section1">
  <p className="fadeup">Compagno .</p>
  <h1 className="fadeup">우린 동물을 지키고 사랑합니다</h1>
  <div className="reveal">
    <div className="image-wrap">
      
      
      
    </div>
  </div>
</section>

```

Compagno

로그인 구조 등록 게시판 서비스 공지사항 signup login

메인페이지

Section 2

section 2



Font Event

글자 Div마다 각자의 영역을 변수와 함수형태로 잡아서
React Hook의 useRef를 사용해서 접근해 준다

```
<div ref={scrollRef1}
      className={`scroll_on type_bottom ${container1 ? "active" : ""}`}>
</div>

<div ref={scrollRef2}
      className={`scroll_on type_bottom ${container2 ? "active" : ""}`}>
</div>

<div ref={scrollRef3}
      className={`scroll_on type_bottom ${container3 ? "active" : ""}`}>
</div>
```

```
.scroll_wrap {
  overflow: hidden;
}

.scroll_on {
  padding: 50px 0;
  opacity: 0;
  transition: all 1.5s;
}

.scroll_on.active {
  opacity: 1 !important;
  transform: translate(0, 0) !important;
}

.scroll_on.type_bottom {
  transform: translate(0, -50px);
}
```

메인페이지

Section 2



```
<div ref={scrollRef1}
  className={`scroll_on type_bottom ${container1 ? "active" : ""}`}>
</div>

<div ref={scrollRef2}
  className={`scroll_on type_bottom ${container2 ? "active" : ""}`}>
</div>

<div ref={scrollRef3}
  className={`scroll_on type_bottom ${container3 ? "active" : ""}`}>
</div>
```

useState로 Container마다 초기값을 false로 하고
삼항 연산자를 이용해서 true일때 active가 돌아가도록



스크롤시 글자태그 영역이 화면에 보일시 Event 발생

```
const Section2 = () => {
  const scrollRef1 = useRef(null);
  const scrollRef2 = useRef(null);
  const scrollRef3 = useRef(null);
  const [container1, setContainer1] = useState(false);
  const [container2, setContainer2] = useState(false);
  const [container3, setContainer3] = useState(false);

  const handleScroll = () => {
    const rect1 =
      scrollRef1.current.getBoundingClientRect();
    const rect2 =
      scrollRef2.current.getBoundingClientRect();
    const rect3 =
      scrollRef3.current.getBoundingClientRect();

    const winHeight = window.innerHeight;

    const contentHeight1 = rect1.bottom - rect1.top;
    const contentHeight2 = rect2.bottom - rect2.top;
    const contentHeight3 = rect3.bottom - rect3.top;

    if (
      rect1.top <= winHeight - contentHeight1 &&
      rect1.bottom >= contentHeight1
    ) {
      setContainer1(true);
    } else {
      setContainer1(false);
    }
    if (
      rect2.top <= winHeight - contentHeight2 &&
      rect2.bottom >= contentHeight2
    ) {
      setContainer2(true);
    } else {
      setContainer2(false);
    }
    if (
      rect3.top <= winHeight - contentHeight3 &&
      rect3.bottom >= contentHeight3
    ) {
      setContainer3(true);
    } else {
      setContainer3(false);
    }
  };

  useEffect(() => {
    window.addEventListener("scroll", handleScroll);

    return () => {
      window.removeEventListener("scroll", handleScroll);
    };
  }, []);
}
```

메인페이지

Section 3

section 3

Compagno

로그인 회원가입 회원가입 회원가입 회원가입

로그인 회원가입 회원가입 회원가입 회원가입

저희는 반려동물과 함께하는 즐거움, 특별함의 서비스를 제공하고 이웃과의 다양한 모임을 통해 소중한 인연과 추억을 지속적인 유지를 이야기고 있습니다.

다양한 서비스와 함께 우리는 반려동물을 통해 행복을 이야기합니다.

Compagno는 반려동물

Scroll Event

```

<section id="section3" className="page-start">
  <div className="sec3">
    <div className="son1">
      <h2>01</h2>
      <p> ... </p>
      
    </div>
    <div className="son1">
      <h2>02</h2>
      <p> ... </p>
      
    </div>
    <div className="son1">
      <h2>03</h2>
      <p> ... </p>
      
    </div>
    <div className="son1">
      <h2>04</h2>
      <p style={{ position: "relative", top: "20px" }}> ...
        
      </p>
    </div>
  </div>
</section>
```

메인페이지

Section 3

sec3의 위치를 상단에 위치를 좌측상단에서 고정을 하고
sec3 부분에서 부터 float left로 이동하겠끔

```
const Section3 = () => {
  const [scroll, setScroll] = useState(0);

  useEffect(() => {
    function handleScroll() {
      const scrollTop =
        window.pageYOffset || document.documentElement.scrollTop;
      setScroll(scrollTop);

      const section2Top = document
        .getElementById("section2")
        .getBoundingClientRect().top;
      const offset = scrollTop - section2Top;

      if (scrollTop > section2Top) {
        document.querySelectorAll(".sec3").forEach((element) => {
          element.style.left = -offset + "px";
        });
      }
    }

    window.addEventListener("scroll", handleScroll);

    return () => {
      window.removeEventListener("scroll", handleScroll);
    };
  }, []);
}
```

```
.sec3 {
  position: fixed;
  left: 0;
  top: 0;
  overflow: hidden;
  color: white;
}

.sec3 > div {
  background-color
    : rgb(70, 92, 88);
  width: 2000px;
  position: relative;
  right: 1409px;
  height: 100vh;
  top: 68px;
  float: left;
}
```

회원가입

회원가입의 핵심! BCryptPasswordEncoder 암호화

| | |
|-------------|---|
| 아이디 | <input type="text" value="q"/> |
| | 중복확인 |
| | 영어 대소문자 및 숫자 포함 8~15자 (단, 첫 글자는 영문자로) |
| 비밀번호 | <input type="password"/> |
| | 영어 대소문자, 숫자 및 특수문자 포함 8~16자(영어, 특수문자, 숫자 각각 1개 이상) |
| 비밀번호 확인 | <input type="password"/> |
| 이름 | <input type="text" value="q"/> \$2a\$10\$.Vq1H4ymwWnGsgUss0NWI.F845VKSGn3OEO4c1IveJeHCAriB4BhC |
| | 영어 대소문자 및 한글 2~20자 |
| 닉네임 | <input type="text" value="q"/> \$2a\$10\$f7jqngZBfP.snk7TUIQf.FH9j.pjLuR5AE39Wy0aqCoto10fac/2 |
| | 영어 대소문자 및 한글 3~20자 |
| 이메일 주소 | <input type="text" value="q"/> 올바른 이메일 양식 입력 요망 |
| 전화번호 | <input type="text" value="0"/> 하이픈(-) 포함해서 입력 요망 |
| 회원가입 | |

```
// 회원가입
@PostMapping("/signUp")
public ResponseEntity create(@RequestBody User vo) {
    LocalDateTime localDateTime = LocalDateTime.now();
    String nowDate = java.sql.Timestamp.valueOf(localDateTime).toString();

    String uuid = UUID.randomUUID().toString();

    User user = User.builder()
        .userId(vo.getUserId())
        .userPwd(passwordEncoder.encode(vo.getUserPwd()))
        .userPersonName(result.getUserName())
        .userPersonName(result.getPersonName())
        .userPhone(vo.getUserPhone())
        .userEnrollDate(nowDate)
        .userStatus("n")
        .userRole("ROLE_USER")
        .userImg("user" + File.separator + "defaultImage.png") // LostB
        .build();

    User result = userService.create(user);

    userService.setLogic(user);

    UserDTO dto = UserDTO.builder()
        .userId(result.getUserId())
        .userPersonName(result.getUserPersonName())
        .build();

    return ResponseEntity.ok().body(dto);
}
```



로그인

```

@.Autowired
private TokenProvider tokenProvider;

no usages: ± SHLee334
@Override
protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain) throws ServletException, IOException {
    String token = parseBearerToken(request);

    if(token!= null && !token.equalsIgnoreCase( anotherString: "null")) {
        User user = tokenProvider.validateGetUser(token);

        AbstractAuthenticationToken authentication = new UsernamePasswordAuthenticationToken(user, user.getPassword(), user.getAuthorities());
        authentication.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

        SecurityContext securityContext = SecurityContextHolder.createEmptyContext();
        securityContext.setAuthentication(authentication);

        SecurityContextHolder.setContext(securityContext);
    }
    filterChain.doFilter(request, response);
}

1 usage: ± SHLee334
private String parseBearerToken(HttpServletRequest request) {
    String bearerToken = request.getHeader(± "Authorization");

    if(StringUtils.hasText(bearerToken) && bearerToken.startsWith("Bearer")) {

        return bearerToken.substring( beginIndex: 7);
    }
    return null;
}
}
return ResponseEntity.badRequest().build();
}

```

로그인

Securityconfig에서 앞서 발급한 토큰의 유무를 활용해 URL 별 접근 제한 / 허용에 활용

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    return http.csrf(csrf -> csrf.disable())
        .httpBasic(basic -> basic.disable())
        .sessionManagement(session -> session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
        .authorizeHttpRequests(authorize -> authorize
            .requestMatchers( ...patterns: "/compagno/login").permitAll()
            .requestMatchers( ...patterns: "/compagno/signUp").permitAll()
            .requestMatchers( ...patterns: "/compagno/signUp/**").permitAll()
            .requestMatchers( ...patterns: "/compagno/public/**").permitAll()
            .requestMatchers( ...patterns: "/compagno/register-pet/**").permitAll()
            .anyRequest().authenticated()
        )
        .addFilterAfter(jwtAuthenticationFilter, CorsFilter.class)
        .build();
}
```

프론트 사용자 정보

Redux 란?

- 여러 컴포넌트가 공유하는 상태를 관리하는 라이브러리
- 상태를 컴포넌트에 종속시키지 않고 바깥(store)에서 관리
- Provider 컴포넌트가 최상위로 동작



```
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <Provider store={store}>
    <RouterProvider router={router} />
  </Provider>
);
```

프론트 사용자 정보

Redux Store

```

export const asyncLogin = createAsyncThunk("user/login", async (data) => {
  const Response = await loginUser(data);
  return Response.data;
});

const user = createSlice({
  name: "user",
  initialState: {},
  reducers: {
    userSave: (state, action) => {
      // 새로고침해도 로그인 정보 유지할수있게
      return action.payload;
    },
    userLogout: (state, action) => {
      return {};
    },
  },
  extraReducers: (builder) => {
    builder.addCase(asyncLogin.fulfilled, (state, action) => {
      const result = action.payload;

      return result;
    });
    builder.addCase(asyncLogin.rejected, (state, action) => [
      alert("회원 정보가 존재하지 않습니다.");
    ]);
  },
});
export default user;
export const { userSave, userLogout } = user.actions;

```

userSave : 로그인 성공 시 사용자 정보 저장

userLogout : 사용자 정보 비우며 로그아웃

로그인 성공 시 (asyncLogin.fulfilled)

- 사용자 정보 담겨있는 Token 리턴

로그인 실패 시 (asyncLogin.rejected)

- 로그인 실패 문구 안내

프론트 사용자 정보

Header와 LocalStorage

token이 null이 아닌 경우(== 로그인 인증에 성공한 경우)
LocalStorage에 저장되어 있는 'user' 정보를 전역 처리
React Hook의 useEffect와 활용해 컴포넌트 접근 시 해당 로직 실행

```
// 유저 정보 가져오기 (유저관련)
const user = useSelector((state) => {
  return state.user;
});
```

```
// dispatch를 통해 Reducer에 주문 전달
const token = localStorage.getItem("token");

if (token !== null) {
  dispatch(userSave(JSON.parse(localStorage.getItem("user"))));
}
[, []);
```

프론트 사용자 정보

Login과 LocalStorage

```

1 const info = useSelector((state) => {
2   return state.user;
3 });
4
5 // 로그인 버튼 눌렀을때
6 const submit = () => {
7   dispatch(asyncLogin(user));
8 };
9
10 useEffect(() => {
11   if (Object.keys(info).length !== 0) {
12     localStorage.setItem("token", info.token);
13     localStorage.setItem("user", JSON.stringify(info));
14     const currentUserStatus = info.userStatus;
15     if (currentUserStatus === "y") {
16       localStorage.removeItem("token");
17       localStorage.removeItem("user");
18       dispatch(userLogout());
19       alert("회원 정보가 존재하지 않습니다.");
20     } else {
21       navigate("/compagno");
22     }
23   }
24 }, [info]);

```

로그인 버튼 클릭 시

store의 유저정보가 존재하면

- > localStorage에 UserDTO 타입의 정보 및 토큰 저장

userStatus(가입 상태)가 'y(탈퇴)' 이면

- > 로그인 제한

프론트 사용자 정보

Component와 사용자 정보



```
1 const [user, setUser] = useState({});  
2 // 유저정보 가지고 오기  
3 const info = useSelector((state) => {  
    return state.user;  
});  
4  
5 useEffect(() => {  
    if (Object.keys(info).length === 0) {  
        setUser(JSON.parse(localStorage.getItem("user")));  
    } else {  
        setUser(info);  
    }  
}, [ ]);
```

두 가지의 방법으로 사용자 정보 전역처리

1. store의 사용자 정보 가져오기
2. store에 정보 없을 시 localStorage의 사용자 정보 가져오기

이를 통해 사용자 정보가 필요한 컴포넌트에서 정보 활용 가능

입양공고&실종신고

전체보기 > 페이지당 게시물 12개 + 페이징 처리



```

1 display: grid;
2   grid-template-columns: 1fr 1fr 1fr 1fr;
3   grid-template-rows: 540px 540px 540px;
4   grid-row-gap: 20px;
5   grid-column-gap: 30px;
6   width: 67%;
7   margin-top: 40px;
  
```

입양공고&실종신고

전체보기 > 페이지당 게시물 12개 + 페이징 처리

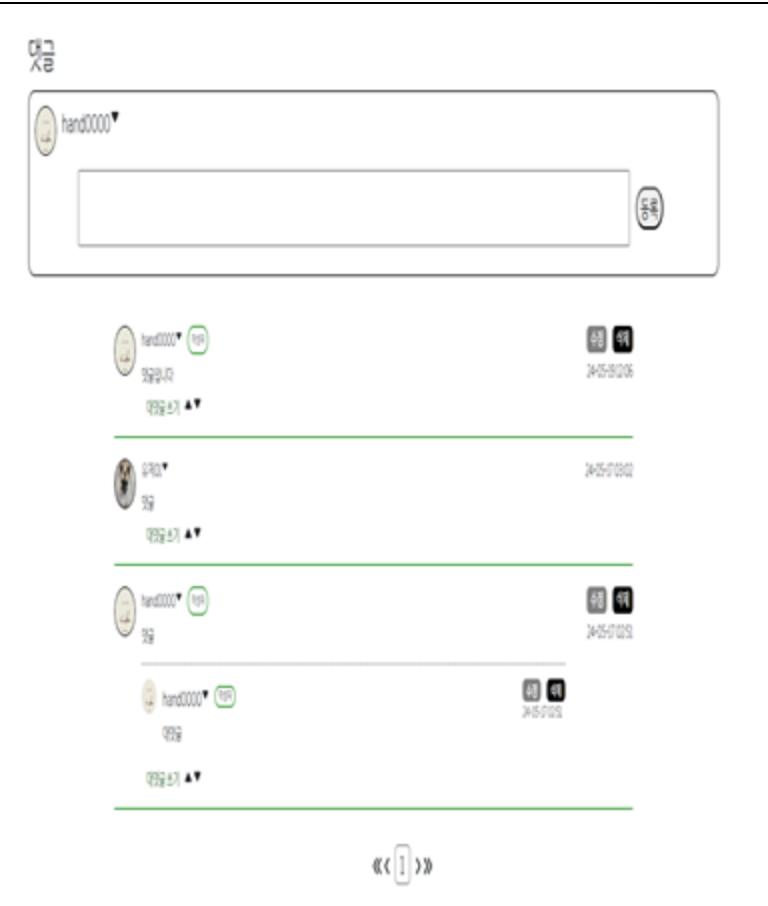
```
1 <div className="paging">
2     <FaAngleLeft className="iconPaging" onClick={() => setPage(1)} />
3     <FaAngleLeft
4         className="iconPaging"
5         onClick={() => (page > 1 ? setPage(page - 1) : setPage(1))}>
6     />
7     {pages.map((num, index) => (
8         <button
9             key={index}
10            value={num}
11            onClick={(e) => setPage(Number(e.target.value))}>
12            {num}
13        </button>
14    ))}
15     <FaAngleRight
16         className="iconPaging"
17         onClick={
18             () => (page < totalPage ? setPage(page + 1) : setPage(totalPage))
19         }
20     />
21     <FaAngleRight
22         className="iconPaging"
23         onClick={() => setPage(totalPage)}
24     />
25 </div>
```

```
1 const [adops, setAdops] = useState([]);
2 const [page, setPage] = useState(1);
3 const [totalPage, setTotalPage] = useState(0);
4 const [pages, setPages] = useState([]);

1 const adopsAPI = async () => {
2     let response = await viewAllAdopBoard(
3         page +
4             "&adopAnimalKind=" +
5             searchKind +
6             "&adopAnimalGender=" +
7             searchGender +
8             "&adopAnimalNeuter=" +
9             searchNeuter +
10            "&adopAnimalFindplace=" +
11            searchFindplace +
12            "&adopCenterName=" +
13            searchCenterName +
14            "&sort=" +
15            sort
16        );
17     setAdops(response.data.content);
18     setAllCount(response.data.totalElements);
19     setTotalPage(response.data.totalPages);
20 };
21
22 let lastPage = 0;
23 let firstPage = 0;
24 let pagelist = [];
25
26 useEffect(() => {
27     adopsAPI();
28 }, [page, sort]);
29
30 useEffect(() => {
31     lastPage = Math.ceil(page / 5) * 5;
32     firstPage = lastPage - 4;
33     if (totalPage < lastPage) {
34         lastPage = totalPage;
35     }
36     for (let i = firstPage; i <= lastPage; i++) {
37         pagelist.push(i);
38     }
39     setPages(pagelist);
40 }, [totalPage]);
```

입양공고&실종신고

상세보기 > 댓글&대댓글 + 페이징



<삼항연산자 사용>

[조건1]
회원 / 비회원 => 댓글&대댓글 작성창 유무

```
1 {Object.keys(user).length == 0 ? <></>:<></>}
```

[조건2]

'작성자' 표시 => 게시물 작성자 아이디==댓글/대댓글 아이디

[조건3]

댓글/대댓글 수정&삭제 => 유저 아이디== 댓글/대댓글 아이디

```
1 {user.userId == comment.user.userId ? <></>:<></>}
```

[조건4]

대댓글 => 댓글 코드와 대댓글의 상위 댓글 코드 비교

```
1 {comment.adopCommentCode != bottomComments.adopParentCode ? <></>:<></>}
```

입양공고&실종신고 상세보기 > 댓글&대댓글 + 페이징

```

1 // 댓글 수정
2 const [edit, setEdit] = useState({});
3 const updateBtn = async (comment) => {
4   setEdit({
5     commentContent: comment.commentContent,
6     adopBoardCode: code,
7     adopCommentCode: comment.adopCommentCode,
8     adopParentCode: 0,
9     commentDate: moment().format("YYYY-MM-DD hh:mm:ss"),
10    user: {
11      userId: user.userId,
12      userNickname: user.userNickname,
13      userImg: user.userImg,
14    },
15  });
16 };
17 const updateComment = async () => {
18   await updateCommentAdop(edit);
19   setEdit({}); // 초기화
20   commentsAPI();
21 };

```

```

1 // 대댓글 작성
2 const [bottomComments, setBottomComments] = useState({
3   userImg: user.userImg,
4   userNickname: user.userNickname,
5   userId: user.userId,
6   commentContent: "",
7   adopBoardCode: code,
8   adopParentCode: "",
9 });

```

입양공고&실종신고

상세보기 > 댓글&대댓글 + 페이징

```

1 const [page, setPage] = useState(1);
2 const [totalComments, setTotalComments] = useState(0);
3 const [totalPage, setTotalPage] = useState(0);
4 const [pages, setPages] = useState([]);
5
6 const [comments, setComments] = useState([]);
7 const commentsAPI = async () => {
8   const response = await viewCommentAdop(code, page);
9   const responseAll = await viewAllCommentAdop(code);
10  setTotalComments(responseAll.data.length);
11  setTotalPage(Math.ceil(responseAll.data.length / 5));
12  setComments(response.data);
13};
14
15 useEffect(() => {
16   commentsAPI();
17 }, []);
18
19 useEffect(() => {
20   commentsAPI();
21 }, [page]);
22
23 let lastPage = 0;
24 let firstPage = 0;
25 let pageList = [];
26
27 useEffect(() => {
28   lastPage = Math.ceil(page / 5) * 5;
29   firstPage = lastPage - 4;
30
31   if (totalPage < lastPage) {
32     lastPage = totalPage;
33   }
34   for (let i = firstPage; i <= lastPage; i++) {
35     pageList.push(i);
36   }
37   setPages(pageList);
38 }, [totalPage]);

```

```

1 <div
2   id="pagingBtn"
3   style={({
4     width: "65%",
5     textAlign: "center",
6     margin: "20px 0px",
7     position: "absolute",
8     paddingBottom: "20px",
9   })
10  >
11    <aAngleLeft
12      className="iconPaging"
13      onClick={() => setPage(1)}
14      style={({ cursor: "pointer" })
15    />
16    <aAngleLeft
17      style={({ cursor: "pointer" })
18      className="iconPaging"
19      onClick={() => (page > 1 ? setPage(page - 1) : setPage(1))
20    />
21    (pages.map((num, index) => (
22      <>
23        <button
24          id="btnPage"
25          key={index}
26          value={num}
27          onClick={(e) => setPage(Number(e.target.value))}
28          style={({
29            fontWeight: "bold",
30            width: "25px",
31            height: "28px",
32            borderRadius: "5px",
33            border: "1px solid gray",
34            backgroundColor: "white",
35            color: "black",
36            margin: "5px",
37          })
38        >
39          {num}
40        </button>
41      </>
42    )));
43    <aAngleRight
44      style={({ cursor: "pointer" })
45      className="iconPaging"
46      onClick={
47        () => (page < totalPage ? setPage(page + 1) : setPage(totalPage))
48      }
49    />
50    <aAngleRight
51      style={({ cursor: "pointer" })
52      className="iconPaging"
53      onClick={() => setPage(totalPage)}
54    />
55  </div>
56 </div>

```

입양공고&실종신고

게시글 수정 > 기존 작성값 고정

입양등록정보
등록일자

성별

고양이

생산자

개체

성별

미수컷

수컷

모름

중성화여부

예

아니오

아시아우리민족

```
1 // 축종 기준 선택
2 const defaultKind = () => {
3   const list = document.getElementsByClassName("animalKind");
4   for (let i = 0; i < list.length; i++) {
5     if (list[i].value == adopAnimalKind) {
6       list[i].selected = true;
7     }
8   }
9 };
10 // 축종 변경
11 const selectKind = (e) => {
12   setAdopAnimalKind(e.target.value);
13 };
14 
```

```
1 // 성별 기준 선택
2 const defaultGender = () => {
3   const checkboxes = document.getElementsByClassName("gender");
4   for (let i = 0; i < checkboxes.length; i++) {
5     if (checkboxes[i].value == adopAnimalGender) {
6       checkboxes[i].checked = true;
7     }
8   }
9 };
10 // 성별 선택
11 const genderCheck = (gender) => {
12   const checkboxes = document.getElementsByClassName("gender");
13   for (let i = 0; i < checkboxes.length; i++) {
14     if (checkboxes[i].value != gender) {
15       checkboxes[i].checked = false;
16     } else if (checkboxes[i].value == gender) {
17       checkboxes[i].checked = true;
18       setAdopAnimalGender(gender);
19     }
20   }
21 };
```

게시글 수정 > 기존 이미지 삭제&수정, 이미지 미리보기



```
<label>사진첨부</label>
{imgSrc.length == 0 ? (
  <div id="existingImg">
    {existImages?.map((image) => (
      <img
        alt=""
        key={image.adopImageCode}
        src={image.adopImage?.replace(
          "\\\\"DESKTOP-U0CNG13\\upload\\adoptionBoard",
          "http://192.168.10.28:8081/adoptionBoard/"
        )}
        onClick={() => deleteImage(image.adopImageCode)}
      />
    )))
  </div>
) : (
  <div className="images">
    {imgSrc.map((img, i) => (
      <div key={i}>
        <img src={img} />
      </div>
    )))
  </div>
)
)
<div id="imgList">
  <div id="images">
    <label>
      <input
        type="file"
        accept="image/*"
        multiple
        onChange={imageCreate}
      />
      <p>사진 업로드 추가 (최대 3장)</p>
    </label>
  </div>
</div>
</div>
```

```
1 // 이미지 입력 및 미리보기
2 const [imgSrc, setImgSrc] = useState([]);
3 const imageCreate = (e) => {
4     const files = Array.from(e.target.files);
5     if (files.length > 3) {
6         alert("최대 사진 갯수를 초과하였습니다. 다시 선택하여주세요.");
7     } else {
8         setImages(files);
9
10        let file;
11        for (let i = 0; i < files.length; i++) {
12            file = files[i];
13            const reader = new FileReader();
14            reader.onload = () => {
15                images[i] = reader.result;
16                setImgSrc([...images]);
17            };
18            reader.readAsDataURL(file);
19        }
20    }
21};
22
23
24
25 // 새로 받은 것
26 images.forEach((image, index) => {
27     formData.append(`images[${index}]`, image);
28 });
29 // 기존 것
30 existImages.forEach((existImg, index) => {
31     formData.append(`image[${index}]`, existImg.adopImage);
32 });
33
34
35 // 기존 사진 클릭 삭제
36 const deleteImage = (code) => {
37     const imagesss = existImages.filter(
38         (image) => image.adopImageCode !== code
39     );
40     setExistImages(imagesss);
41 };
42
```

검색, 정렬, 페이징



```
@GetMapping("/public/adoptionBoard")
public ResponseEntity<Page<AdoptionBoard>> viewAll(@RequestParam(name="page", defaultValue = "1")int page,
@RequestParam(name="adopAnimalKind", required = false)String adopAnimalKind,
@RequestParam(name="adopAnimalGender", required = false)String adopAnimalGender,
@RequestParam(name="adopAnimalNeuter", required = false)String adopAnimalNeuter,
@RequestParam(name="adopAnimalFindplace", required = false)String
adopAnimalFindplace,
@RequestParam(name="adopCenterName", required = false)String adopCenterName,
@RequestParam(name="sort", defaultValue = "#")int sortNum){

    QAdoptionBoard qAdoptionBoard = QAdoptionBoard.adoptionBoard;
    BooleanBuilder builder = new BooleanBuilder();
    BooleanExpression expression = null;
    if(adopAnimalKind!=null){
        expression = qAdoptionBoard.adopAnimalKind.contains(adopAnimalKind);
        builder.and(expression);
    }

    if(adopAnimalGender!=null){
        expression = qAdoptionBoard.adopAnimalGender.contains(adopAnimalGender);
        builder.and(expression);
    }
    if(adopAnimalNeuter!=null){
        expression = qAdoptionBoard.adopAnimalNeuter.contains(adopAnimalNeuter);
        builder.and(expression);
    }
    if(adopAnimalFindplace!=null){
        expression = qAdoptionBoard.adopAnimalFindplace.contains(adopAnimalFindplace);
        builder.and(expression);
    }
    if(adopCenterName!=null){
        expression = qAdoptionBoard.adopCenterName.contains(adopCenterName);
        builder.and(expression);
    }

    Pageable pageable = PageRequest.of(page-1, 12);
    Sort aggregated = Sort.by("adopBoardCode").descending();
    Sort aRegiDate = Sort.by("adopRegiDate");
    if(sortNum==0){
        pageable = PageRequest.of(page-1, 12, aRegiDate);
    }
    if(sortNum==1){
        pageable = PageRequest.of(page-1, 12, aRegiDate);
    }

    Sort aViewCount = Sort.by("adopViewCount");
    Sort aViewCountD = Sort.by("adopViewCount").descending();

    if(sortNum==2){
        pageable = PageRequest.of(page-1, 12, aViewCount);
    }
    if(sortNum==3){
        pageable = PageRequest.of(page-1, 12, aViewCountD);
    }

    return ResponseEntity.ok(service.viewAll(pageable, builder));
}quest.of(page-1, 12, aViewCountD);
}

return ResponseEntity.ok(service.viewAll(pageable, builder));
} 
```

이미지 수정

```

for(AdoptionBoardImage image : list){
    if(dto.getImage() != null && !dto.getImage().contains(image.getAdopImage()) || dto.getImage() == null){
        File file = new File(image.getAdopImage());
        file.delete();
        service.delImg(image.getAdopImageCode());
    }else{
        mainImage.add(image.getAdopImage());
    }
}

if(dto.getImages() != null){
    for(AdoptionBoardImage image: list){
        File file = new File(image.getAdopImage());
        file.delete();
        service.delImg(image.getAdopImageCode());
        mainImage.remove(0);
    }
    for(MultipartFile file : dto.getImages()){
        String fileName = file.getOriginalFilename();
        String uuid = UUID.randomUUID().toString();

        String saveName = uploadPath + File.separator + "adoptionBoard" + File.separator + uuid + "_" + fileName;
        Path savePath = Paths.get(saveName);
        file.transferTo(savePath);

        service.createImg(AdoptionBoardImage.builder()
            .adopImage(saveName)
            .adopBoardCode(AdoptionBoard.builder().adopBoardCode(dto.getAdopBoardCode()).build())
            .build());
        mainImage.add(saveName);
    }
}
adop.setAdopAnimalImage(mainImage.getFirst());
service.create(adop);
return ResponseEntity.ok().build();
}

```

[이미지 수정]

- 기존 이미지 있을 경우
=> 삭제
- 새로 삽입된 이미지 있을 경우
=> 이미지 중 첫 번째를 메인 이미지로 설정
=> 새로 삽입된 이미지 추가

댓글&대댓글

queryFactory 사용

```
// 상위 댓글 조회
public List<AdoptionBoardComment> topComments(int adopBoardCode, int page){
    return queryFactory
        .selectFrom(qAdoptionBoardComment)
        .where(qAdoptionBoardComment.adopParentCode.eq(0))
        .where(qAdoptionBoardComment.adopBoardCode.eq(adopBoardCode))
        .orderBy(qAdoptionBoardComment.commentDate.desc())
        .offset(5*(page-1))
        .limit(5)
        .fetch();
}

// 하위 댓글 조회
public List<AdoptionBoardComment> bottomComments(int parent, int adopBoardCode){
    return queryFactory
        .selectFrom(qAdoptionBoardComment)
        .where(qAdoptionBoardComment.adopParentCode.eq(parent))
        .where(qAdoptionBoardComment.adopBoardCode.eq(adopBoardCode))
        .orderBy(qAdoptionBoardComment.commentDate.desc())
        .fetch();
}
```

service

```
// (상위 100개 댓글 조회)
@GetMapping("/public/adoptionBoard/comment/{adopBoardCode}")
public ResponseEntity<List<AdoptionBoardCommentDTO>> viewComment(@PathVariable(name="adopBoardCode")int adopBoardCode,
@RequestParam(name="page")int page){
    List<AdoptionBoardComment> topList = comment.topComments(adopBoardCode, page);
    List<AdoptionBoardCommentDTO> response = commentDetailList(topList, adopBoardCode);
    return ResponseEntity.ok(response);
}

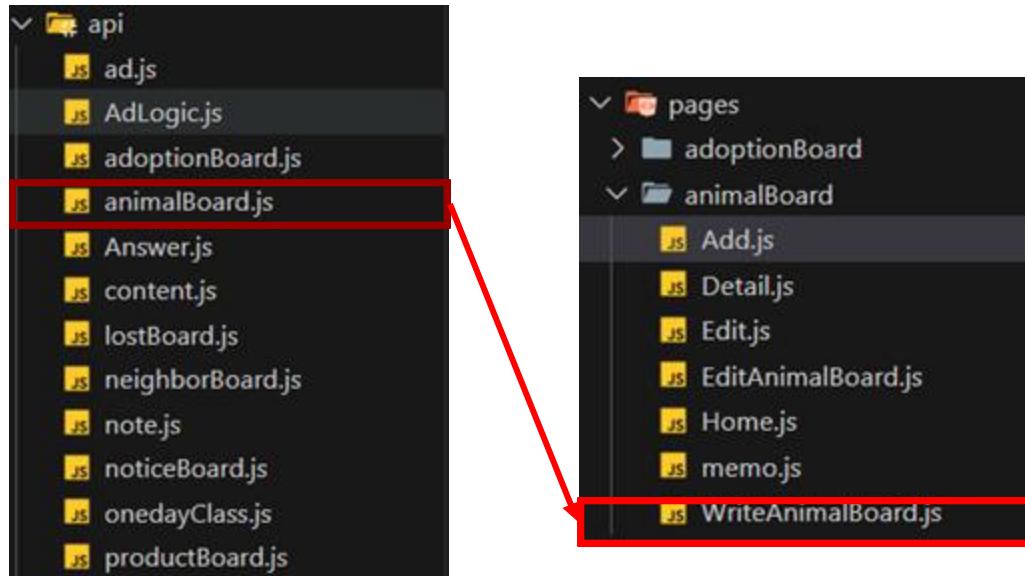
// (하위 댓글 조회)
public List<AdoptionBoardCommentDTO> commentDetailList(List<AdoptionBoardComment> comments, int adopBoardCode){
    List<AdoptionBoardCommentDTO> response = new ArrayList<>();
    for(AdoptionBoardComment item : comments){
        List<AdoptionBoardComment> replies = comment.bottomComments(item.getAdopCommentCode(), adopBoardCode);
        List<AdoptionBoardCommentDTO> repliesDTO = commentDetailList(replies, adopBoardCode);
        AdoptionBoardCommentDTO dto = commentDetail(item);
        dto.setReplies(repliesDTO);
        response.add(dto);
    }
    return response;
}

// builder.build 활용하기
public AdoptionBoardCommentDTO commentDetail(AdoptionBoardComment vo){
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
    return AdoptionBoardCommentDTO.builder()
        .adopBoardCode(vo.getAdopBoardCode())
        .commentDate(sdf.format(vo.getCommentDate()))
        .commentContent(vo.getCommentContent())
        .adopCommentCode(vo.getAdopCommentCode())
        .user(UserDTO.builder()
            .userId(vo.getUser().getUserId())
            .userNickname(vo.getUserNickname())
            .userImg(vo.getUserImg())
            .build())
        .build();
}
```

controller

자유게시판 1. 글쓰기 File Structure

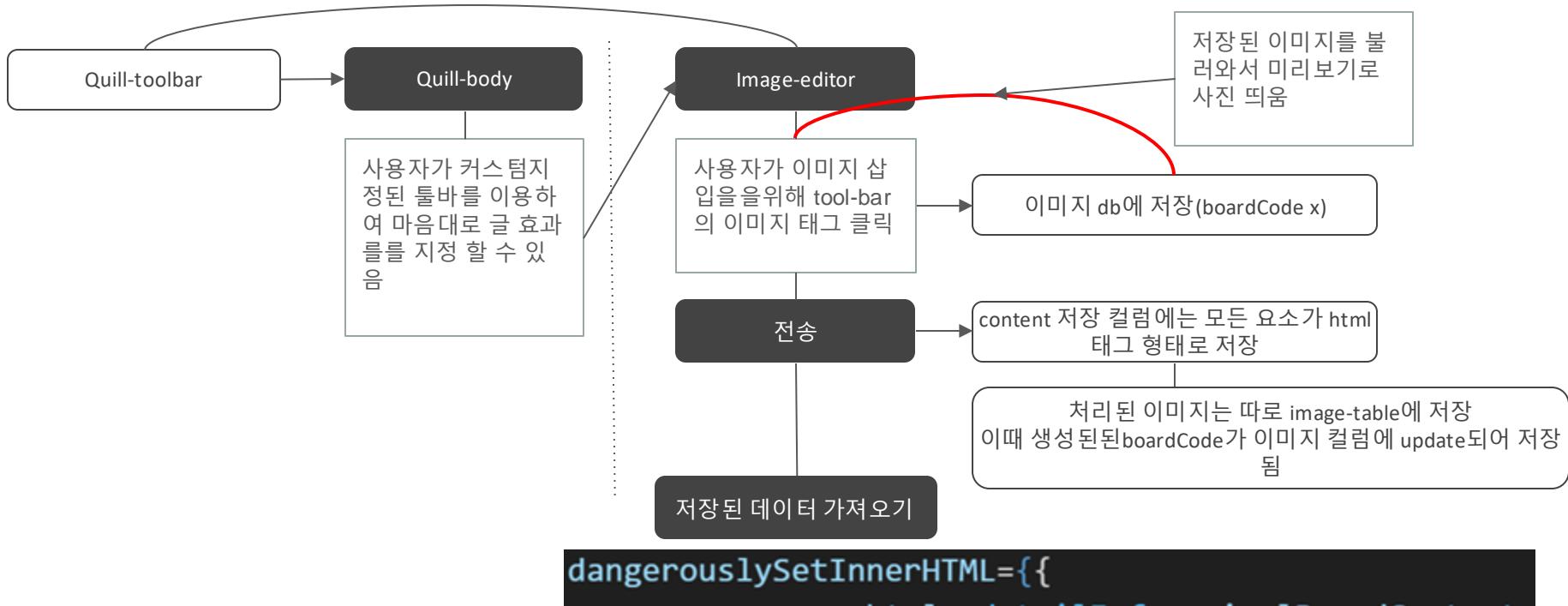
개발 목표 : react-quill을 사용하여, 사용자가 더욱 편리하게 이미지 수정, 삽입 글 효과 주기 등을 할 수 있다.



자유게시판 3.글쓰기 - react-quill

react-quill 이란?

React-Quill은 React 애플리케이션에서 사용할 수 있는 WYSIWYG(What You See Is What You Get) 텍스트 에디터. Quill이라는 오픈 소스 텍스트 에디터를 React 환경에서 쉽게 사용할 수 있도록 만든 라이브러리.



COMPAGNO

© 2024 COMPAGNO. All rights reserved.

자유게시판 3. 글쓰기 - react-quill - front

```
imageUploader: {  
    upload: (file) => {  
        return new Promise((resolve, reject) => {  
            const formData = new FormData();  
            // console.log(animalBoardCode);  
            // formData.append("animalBoardCode", animalBoardCode); 이부분은 단순히  
            formData.append("file", file);  
            // console.log(formData);  
  
            fetch("http://localhost:8080/compagno/public/animal-board/image", {  
                method: "POST",  
                body: formData,  
            })  
        });  
    };  
};
```

1. imageUploader:

이미지 업로더 기능 객체

2. upload 메서드:

file을 인자로 받아, 이를 서버에 업로드하는 비동기 작업을 수행

return 값: Promise 성공하면 resolve/실패하면 reject

3. FormData 객체 생성:

FormData 객체를 생성하고, 파일을 formData에 추가

4. fetch를 통한 이미지 업로드:

fetch API를 사용하여 POST처리 해당 formData 전송

자유게시판 3.글쓰기 - react-quill - front

```
.then((response) => {
  if (!response.ok) {
    throw new Error("Network response was not ok");
  }
  return response.json();
})
.then((result) => {
  const image =
    "http://192.168.10.28:8081" + result.animalBoardImage;
  resolve(image);
  // setImages((prev) => [...prev, file]);
})
.catch((error) => {
  console.error(
    "There was a problem with the fetch operation:",
    error
  );
  reject(error);
});
```

5. 응답 처리:

=> 응답이 성공적이면 JSON 형식으로 변환

6. 이미지 URL 생성:

=> 서버에서 반환된 결과에서 이미지 URL을 생성

사전에 정해진 파일 위치 + result에 담긴 image위치

=> 해당 URL을 resolve를 통해 반환

자유게시판 3. 글쓰기 - react-quill - controller

```
// 이미지 저장 - boardCode == null 인 상태로 시작
@PostMapping("public/animal-board/image") no usages ▲ jungdj221
public ResponseEntity<AnimalBoardImage> saveImages(AnimalBoardDTO dto) throws IOException {
    String fileName = dto.getFile().getOriginalFilename();
    String uuid = UUID.randomUUID().toString();
    String saveName = uploadPath + File.separator + "animalBoard" + File.separator + uuid + "_" + fileName;

    Path savePath = Paths.get(saveName); // image 저장 경로
    dto.getFile().transferTo(savePath);

    AnimalBoardImage image = AnimalBoardImage.builder()
        .animalBoardImage(saveName.substring( beginIndex: 24)) // uploadPath 재확인
        .build(); // \\DESKTOP-U0CNG13\\ upload\\여기까지가 25글자 animalBoard전에 딱 끊김

    AnimalBoardImage response = animalBoardService.saveImages(image);
    log.info("controller response : " + response);
    return ResponseEntity.ok(response); // 제일 중요한거! 사진 미리보기등 특정 부분의 결과값 확인을 위해서
}
```

자유게시판 3. 글쓰기 - react-quill - controller

```
@PostMapping("/animal-board") no usages ▲ jungdj221 +1
public ResponseEntity<AnimalBoard> writeBoard(@RequestBody AnimalBoardDTO dto){
    Date nowDate = currentDate();

    log.info("write dto : " + dto);
    Object principal = Authentication();
    if(principal instanceof User) {
        User user = (User) principal;

        // 이제 animal_board에 데이터를 추가함으로써 board_code가 생성됨
        AnimalBoard response = animalBoardService.write(AnimalBoard.builder()
            .animalBoardContent(dto.getAnimalBoardContent())
            .animalBoardTitle(dto.getAnimalBoardTitle())
            .animalBoardDate(nowDate)
            .animalCategory(AnimalCategory.builder()
                .animalCategoryCode(dto.getAnimalCategoryCode())
                .build())
            .user(User.builder()
                .userId(user.getUserId()) // 오류나면 dto로 바꾸기
                .build())
            .build());
    }
}
```

dto로 받아온 글 정보로 boardCode 생성.

=> 글 추가시에 auto_increment 속성인 animal_board_code가 추가됨.

이를 통해서 이미지에 해당하는 boardCode를 반환해줄 수 있음

자유게시판 3. 글쓰기 - react-quill - controller

```
/*
 * 정규표현식 패턴 설정
 String regex = "<[^>]+>";

 // 패턴 매칭을 위한 Pattern 객체
 Pattern pattern = Pattern.compile(regex);

 // 패턴과 일치하는 부분 찾기 위한 Matcher 객체
 Matcher matcher = pattern.matcher(dto.getAnimalBoardContent());

 List images = new ArrayList(); // 글쓰기에서 최종적으로 남은 이미지 리스트 담을 공간
 // 매칭된 문자열 추출
 while(matcher.find()) {
     if(matcher.group().startsWith("<img")) {
         String image = matcher.group().substring(35, matcher.group().length() - 2);
         if(image.contains("\\")){
             String[] finalizedArr = image.split( regex "\\" );
             String finalizedImage = finalizedArr[0];
             log.info("db에 저장될 이미지 : " + image);
         }
     }
 }
```

'<'로 시작하고'>'로 끝나는 문자열을 매칭

dto로 받아온 content의 내용을 Matcher함수에 넣은후, 정규표현식 Pattern함수와 비교를 통해 일치하는 부분에 대한 정의를 하기위해 사용

이미지 처리전url : <p></p>

```
    // image-resize가 적용될 경우 split을 통해서 이미지 문자열만 2차 재추출 해야함.
 }else{
     images.add(image); // 매칭된 이미지를 images에 추가
     // matcher 가 정규표현식이 매칭되는 이미지태그의 이미지를 images 리스트에 담아줌
     // 애가 DB에 저장되어야할 아이
 }
```

Windows

자유게시판

3. 글쓰기 - react-quill - controller

```
imageResize: {  
    displaySize: true,  
    parchment: Quill.import("parchment"),  
    modules: ["Resize", "DisplaySize", "Toolbar"],
```

이미지 처리전 url : <p></p>

```
if(matcher.group().startsWith("<img")) {  
    String image = matcher.group().substring(35, matcher.group().length() - 2);  
    if(image.contains("\\")){  
        String[] finalizedArr = image.split(regex "\\\"");  
        String finalizedImage = finalizedArr[0];  
        log.info("db에 저장될 이미지 : " + image);  
        log.info("2차 처리 이미지 배열: " + finalizedArr[0]);  
        log.info("2차 처리 이미지 최종본: " + finalizedImage);  
        images.add(finalizedImage);  
        // image-resize가 적용될 경우 split을 통해서 이미지 문자열만 2차 재추출 해야함.  
    }else{  
        images.add(image); // 매칭된 이미지를 images에 추가  
        // matcher 가 정규표현식이 매칭되는 이미지태그의 이미지를 images 리스트에 담아줌  
        // 예가 DB에 저장되어야할 아이  
    }  
}
```

image-resize가 적용된 경우,

“”로 따옴표로 끝나는 부분을 기준으로 split후, 2차가공후 이미지 저장

자유게시판 3. 글쓰기 - react-quill - controller

```
/*
// animal_board_code 가 null 인 이미지 가져오기
List<AnimalBoardImage> list = animalBoardService.viewImages();
// 일단 그냥 막 들어와진 이미지를

for(AnimalBoardImage image : list){
    if(images.contains(image.getAnimalBoardImage())){
        // 확실이 들어가야하는 이미지리스트와 막무가내 리스트 비교, 두 비교값이 같다면?
        // 즉, DB에 저장되어야 하는 값을 발견했을때는 true
        // 해당 이미지들에게 animal_board_code 추가
        animalBoardService.saveImages(AnimalBoardImage.builder()
            .animalImageCode(image.getAnimalImageCode())
            .animalBoardImage(image.getAnimalBoardImage()) // url
            .animalBoard(AnimalBoard.builder()
                .animalBoardCode(response.getAnimalBoardCode())
                .build())
            .build()); // 미리 선언된 response의 boardCode를 여기에 넣어줌
    }else{
        // 실제 파일 삭제 --> animal_board_code 가 null인 이미지의 실제 사진 삭제
        File file = new File( pathname:uploadPath + image.getAnimalBoardImage());
        file.delete();
    }
}
```

db에 저장된 boardCode가 아직 null인 image들

contain으로 비교

dto에서 받아온 이미지 가공처리가 완료될 실질적으로 db에 저장되어야 할 이미지 리스트

서로 매칭이 된 이미지들만 boardCode가 추가되고, 나머지 파일들은 삭제

자유게시판 3. 글쓰기 - react-quill - controller

```
// 마지막으로 DB상에서 animal_board_Code가 null인 image, image 테이블에서 삭제  
animalBoardService.deleteAnimalNoImages();  
  
// 완성된 이미지 리스트 불러오기 - 추후에 리스트로 변경하여 사용자에게 선택권한을 줌  
AnimalBoardImage thumbnail = animalBoardService.getThumbnailList(response.getAnimalBoardCode());  
log.info("thumbnail : " + thumbnail);  
if(thumbnail!=null){  
    animalBoardService.saveThumbnail(thumbnail.getAnimalBoardImage(), response);  
} else{  
    animalBoardService.saveThumbnail( image: "animalDefault.jpg", response);  
}  
  
return ResponseEntity.ok().build();  
}  
return ResponseEntity.badRequest().build();  
}
```

제품 정보 게시판

Compagno

등록일자 | 조회수 | 게시판 | 서비스 | 공지사항 | 정회원이벤트

제품정보 공유 게시판

| 제품명 | 제작사 | 제작일 |
|--------------------|-----------|------------|------------|------------|------------|------------|------------|------------|------------|
| 소니 알파1000 줌렌즈 | SonyMMall | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 |
| 리거온라인 목사자 흑색 1000g | 리거온라인 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 |
| 리거온라인 목사자 흑색 1000g | 리거온라인 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 |
| 리거온라인 목사자 흑색 1000g | 리거온라인 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 |
| 리거온라인 목사자 흑색 1000g | 리거온라인 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 |
| 리거온라인 목사자 흑색 1000g | 리거온라인 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 |
| 리거온라인 목사자 흑색 1000g | 리거온라인 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 |
| 리거온라인 목사자 흑색 1000g | 리거온라인 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 |
| 리거온라인 목사자 흑색 1000g | 리거온라인 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 |
| 리거온라인 목사자 흑색 1000g | 리거온라인 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 | 2024.05.01 |

제품 정보 게시판 페이지

```

1 const getProductBoards = async () => {
2   const result = await searchProductBoard(filter, page);
3   setProductBoards(result.data);
4   setTotalPage(result.data.totalPages);
5 };
6 let lastPage = 0;
7 let firstPage = 0;
8 let pagelist = [];
9 // totalPage가 바뀔 때마다 실행
10 const paging = () => {
11   lastPage = Math.ceil(page / 5) * 5;
12   firstPage = lastPage - 4;
13
14   if (totalPage < lastPage) {
15     lastPage = totalPage; // 전체 페이지가 마지막 페이지보다 작은 경우엔 전체 페이지 수가 마지막 페이지 수랑 같음
16   }
17   setPrev(firstPage > 1);
18   setNext(lastPage < totalPage);
19   for (let i = firstPage; i <= lastPage; i++) {
20     pagelist.push(i); // 처음 i는 firstPage, 뒷위는 lastPage로 반복문 돌려서 i값을 넣은 list 만들기
21   }
22   setPages(pagelist); // 해당 list 배열을 setPages에 담기
23 };

```

```

1 <nav className="paging">
2   (prev && (
3     <i>fa-angle-left</i> onClick={() => setPage(Math.ceil(page / 5) * 5 - 4)} />
4   ))
5   (page === 1 && (
6     <i>fa-angle-left</i>
7       onClick={() => (page > 1 ? setPage(page - 1) : setPage(1))} // 현재 페이지에서 한칸 앞으로
8     )
9   ))
10  (pages.map(
11    (
12      num,
13      index // 배열 담은 pages를 map으로 만들어서 반복문 페이지번호, 생성
14    ) => (
15      <button
16        className="pageNo@tn"
17        key={index}
18        value={num}
19        onClick={(e) => setPage(Number(e.target.value))}
20        style={num === page ? { backgroundColor: "#94829B" } : {}}
21      >
22        (num)
23      </button>
24    )
25  ))
26  (page === totalPage && totalPage !== 0 && (
27    <i>fa-angle-right</i>
28    onClick={() => (page < totalPage ? setPage(page + 1) : setPage(totalPage))} // 현재 페이지에서 한칸 뒤로
29  )
30 )
31 )
32 )
33 (next && (
34   <i>fa-angle-right</i> onClick={() => setPage(Math.ceil(page / 5) * 5 + 1)} />
35 ))
36 </nav>

```

제품 정보 게시판 검색, 페이징

제품정보공유 게시판



```

1  const [filter, setFilter] = useState({
2    productName: '',
3    productCategory: '',
4    minPrice: 0,
5    maxPrice: 0,
6    animal: 0,
7    minGrade: 0,
8    title: '',
9    select: '',
10   keyword: '',
11   sort: '',
12 });

```



```

1  export const searchProductBoard = async (filter, page) => {
2    function filterNonNull(obj) {
3      return Object.fromEntries(Object.entries(obj).filter(([k, v]) => v));
4    }
5
6    const url = `?${qs.stringify(filterNonNull(filter))}`;
7
8    if (page === 1 || page === 0) {
9      return await instance.get("productBoard" + url);
10   } else {
11     return await instance.get("productBoard" + url + "&page=" + page);
12   }
13 };

```

제품 정보 게시판 검색, 페이징

```

@Data @NoArgsConstructor @AllArgsConstructor
public class ProductBoardSearchDTO {
    private String productName;
    private Integer minPrice;
    private Integer maxPrice;
    private String productCategory;
    private Float grade;
    private Integer animal;
    private String select;
    private String keyword;
    private String sort;
}

```

```

@GetMapping("/public/productBoard")
public ResponseEntity<Page<ProductBoard>> searchBoard(
    @ModelAttribute ProductBoardSearchDTO dto,
    @RequestParam(name = "page", defaultValue = "1") int page) {
    Pageable pageable = PageRequest.of(page - 1, 12);

    Page<ProductBoard> list = productBoard.searchProductBoard(dto, pageable);
    return ResponseEntity.ok().body(list);
}

```

```

public Page<ProductBoard> searchByProductBoard(ProductBoardSearchDTO dto, Pageable pageable) {
    String sort = pageable.getSort();
    BoardBuilder builder = new BoardBuilder();
    if (dto.getAnimal() != null && dto.getAnimal() != 0) {
        builder.andIfProductBoard.productCategory_animalCategoryCode_eq(dto.getAnimal());
    }
    if (dto.getGrade() != null) {
        builder.andIfProductBoard.productGrade_gt(dto.getGrade());
    }
    if (dto.getProductCategory() != null && dto.getProductCategory().length() < 2) {
        builder.andIfProductBoard.productCategory_in(dto.getProductCategory());
    }
    if (dto.getProductName() != null && !dto.getProductName().contains("%")) {
        builder.andIfProductBoard.productName_like(dto.getProductName());
    }
    if (dto.getGrade() != null && dto.getGrade() > 0) {
        builder.andIfProductBoard.productGrade_lt(dto.getGrade());
    }
    if (dto.getAnimal() != null && dto.getAnimal() > 0) {
        builder.andIfProductBoard.productCategory_animalCategoryCode_gt(dto.getAnimal());
    }
    switch (sort.getDirection()) {
        case "ASC":
            break;
        case "DESC":
            builder.andIfProductBoard.productCategory_in(dto.getAnimal());
            break;
        case "ASC2":
            builder.andIfProductBoard.productBoardContent_contentIn(dto.getAnimal());
            break;
        case "DESC2":
            builder.andIfProductBoard.productBoardContent_contentIn(dto.getAnimal());
            break;
    }
    if (dto.getSelect() != null && dto.getKeyword() != null && dto.getKeyword().length() > 0) {
        builder.andIfProductBoard.productBoardContent_contentIn(dto.getKeyword());
    }
}

List<ProductBoard> list = queryFactory.selectFrom(productBoard)
    .where(builder)
    .offset(dto.getPageable().getOffset())
    .limit(dto.getPageable().getPageSize())
    .fetch();

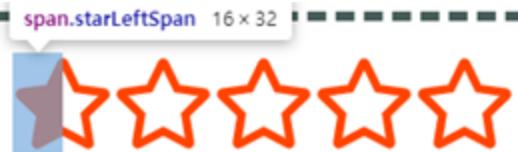
int count = queryFactory.selectFrom(productBoard)
    .where(builder)
    .fetchCount();

return new PageImpl<Pageable>(list, pageable, count);

private OrderSpecification<String> productBoardSort(String sort) {
    List<OrderSpecification<String>> orderSpecifications = new ArrayList<OrderSpecification<String>>();
    if (sort == null || sort.equals("none")) {
        orderSpecifications.add(new OrderSpecification<String>("productBoard.productId", false));
    } else if (sort.equals("sort")) {
        orderSpecifications.add(new OrderSpecification<String>("productBoard.productId", true));
    } else {
        orderSpecifications.add(new OrderSpecification<String>("productBoard.productId", false));
    }
    return orderSpecifications.stream().reduce((a, b) -> a.withOrder(b.getOrder()))
        .orElse(OrderSpecification.of("productBoard.productId", false));
}

```

제품 정보 게시판 별점



```

1 .starLeftSpan {
2   position: absolute;
3   width: 16px;
4   height: 32px;
5   overflow: hidden;
6   left: 0px;
7 }
8 .starRightSpan {
9   position: absolute;
10  width: 16px;
11  height: 32px;
12  overflow: hidden;
13  right: 0px;
14  padding: 0px;
15 }
16 .starDiv {
17  width: 32px;
18  height: 32px;
19  position: relative;
20  display: flex;
21  cursor: pointer;
22 }
23 .star {
24  font-size: 2rem;
25  color: orangered;
26 }
27 .starHalf {
28  font-size: 2rem;
29  color: orangered;
30  padding-top: 3px;
31  padding: 1px 1px;
32 }
33 .gradeDiv {
34  display: inline-flex;
35  flex-direction: row;
36  gap: 0px;
37  width: 250px;
38 }

```

```

1 (filter.grade >= 1.5) {
2   <div className="starDiv">
3     <span>
4       className="starLeftSpan"
5       onClick={() => setFilter((prev) => ({ ...prev, grade: 0.5 }))}
6     </span>
7     <span>
8       className="starRightSpan"
9       onClick={() => setFilter((prev) => ({ ...prev, grade: 1 }))})
10    </span>
11    <faStar className="star" />
12  </div>
13  > : filter.grade === 0.5 ? {
14    <div className="starDiv">
15      <span>
16        className="starLeftSpan"
17        onClick={() => setFilter((prev) => ({ ...prev, grade: 0.5 }))}
18      </span>
19      <span>
20        className="starRightSpan"
21        onClick={() => setFilter((prev) => ({ ...prev, grade: 1 }))})
22      </span>
23      <faStarHalfAlt className="starHalf" />
24    </div>
25  > : (
26    <div className="starDiv">
27      <span>
28        className="starLeftSpan"
29        onClick={() => setFilter((prev) => ({ ...prev, grade: 0.5 }))}
30      </span>
31      <span>
32        className="starRightSpan"
33        onClick={() => setFilter((prev) => ({ ...prev, grade: 1 }))})
34      </span>
35      <faFlagStar className="star" />
36    </div>
37  )
38  > : filter.grade >= 2 ? (
39    <div className="starDiv">
40      <span>
41        className="starLeftSpan"
42        onClick={() => setFilter((prev) => ({ ...prev, grade: 1.5 }))}
43      </span>
44      <span>
45        className="starRightSpan"
46        onClick={() => setFilter((prev) => ({ ...prev, grade: 2 }))})
47      </span>
48      <faStar className="star" />
49    </div>
50  )
51  > : filter.grade === 1.5 ? {
52    <div className="starDiv">
53      <span>
54        className="starLeftSpan"
55        onClick={() => setFilter((prev) => ({ ...prev, grade: 1.5 }))}
56      </span>
57      <span>
58        className="starRightSpan"
59        onClick={() => setFilter((prev) => ({ ...prev, grade: 2 }))})
60      </span>
61      <faStarHalfAlt className="starHalf" />
62    </div>

```

제품 정보 게시판 상세보기

Compagno

포함되는 구조물을 제시한 서비스 품질 평가 및 추천

세종성모 공유 게시판

강아지 이동장



포함 4-1 작성일 2024-05-20 04:07



사용한 훈련과 이용점

사용 중인 개

사용 품목: 케이지

가격: 300,000원

인기도: ★★★★☆ 5

정말 훌륭해요! 3개월 된 강아지들도 딱 훈련한 걸까요? 단점은 훈련한 만큼 무겁다!



작성

댓글 2

작성

정회장아빠님▶ 4주 전

여기 가면에도 좋았고 만약 충분한 거 같아요

리액션 댓글입니다.

작성 1

정회장아빠님▶ 4주 전

여기가 정말 좋았어요

작성 1

제품 정보 게시판 조회수

```
1 export const getProductBoard = async (no) => {
2   return await instance.get("productBoard/" + no, {
3     withCredentials: true,
4   });
5 };
6
```

```
@CrossOrigin (origins = {"http://localhost:3000"}, maxAge = 6000, allowCredentials = "true")
```

```
// 조회수
public void viewCountUp(int code, HttpServletRequest req, HttpServletResponse res) {
    Cookie[] cookies = Optional.ofNullable(req.getCookies()).orElseGet(() -> new Cookie[]{});
    Cookie cookie = Arrays.stream(cookies)
        .filter(c -> c.getName().equals("productBoard"))
        .findFirst()
        .orElseGet(() -> {
            productBoard.viewCountUp(code);
            return new Cookie("productBoard", "[" + code + "]");
        });
    if(cookie.getValue().contains("(" + code + ")")) {
        productBoard.viewCountUp(code);
        cookie.setValue(cookie.getValue() + "(" + code + ")");
    }
    cookie.setPath("/");
    cookie.setMaxAge(60*60*24);
    res.addCookie(cookie);
}
```

```
// 조회수
@Transactional
public void viewCountUp(int code) {
    queryFactory.update(qProductBoard)
        .set(qProductBoard.productBoardViewCount, qProductBoard.productBoardViewCount.add(1))
        .where(qProductBoard.productBoardCode.eq(code))
        .execute();
}
```

제품 정보 게시판 댓글 삭제

```
// 댓글 삭제  
@Transactional  
public void delete(int code) {  
    if(dao.existsById(code)) {  
        queryFactory.update(qProductBoardComment)  
            .set(qProductBoardComment.productCommentDelete, 'Y')  
            .where(qProductBoardComment.productCommentCode.eq(code))  
            .execute();  
    }  
}
```

```
1  {comments?.map((comment) =>  
2      comment.productCommentDelete === "N" ?
```

제품 정보 게시판 수정, 작성

Compagno

로그인회원 구독등록 게시판 서비스 글쓰기 게시판 목록

제품정보 공유 게시판

최신

급여직 사로 여기 편찮아요



내용

사람이 먹지도 될 것 같은 것 같아요

이미지 업로드
파일 선택: 파일선택



취소

설정

제품 정보 게시판 이미지 처리

이미지 업로드

파일 선택

선택된 파일 없음



기존 이미지

```
1 <div className="imagesDiv">
2   {imgSrc.map((img, i) => (
3     <span className="imageSpan" key={i}>
4       <img src={img} />
5       <FaRegCircleXmark
6         onClick={() => {
7           deleteImgSrc(i);
8           imageChange(i);
9         }}
10      />
11    </span>
12  ))
13  {prevImgSrc.map((img, i) => (
14    <span className="imageSpan" key={img.productImageCode}>
15      <img src={"http://192.168.10.28:8081/" + img.productImage} />
16      <FaRegCircleXmark
17        onClick={() => deletePrevSrc(img.productImageCode)}
18      />
19    </span>
20  ))}
21 </div>
22 {prevImg.length !== 0 && (
23   <Button
24     className="prevImgBtn"
25     variant="secondary"
26     onClick={() => {
27       imgCancel();
28     }}
29   >
30     기존 이미지
31   </Button>
32 )
33 </div>
```

제품 정보 게시판 이미지 처리

```

1 const imageCreate = (e) => {
2   const images = Array.from(e.target.files);
3
4   if (images.length + prevImgSrc.length > 4) {
5     alert("이미지는 최대 4장 까지 등록 가능합니다");
6     e.target.value = "";
7     setImgSrc([]);
8   } else {
9     setFiles(images);
10
11   let file;
12   if (files.length > images.length) {
13     files.splice(images.length);
14   }
15   for (let i = 0; i < images.length; i++) {
16     file = images[i];
17     const reader = new FileReader();
18     reader.onloadend = () => {
19       files[i] = reader.result;
20       setImgSrc([...files]);
21     };
22     reader.readAsDataURL(file);
23   }
24
25   if (images.length === 0) {
26     setImgSrc([]);
27     setFiles([]);
28   }
29 }
30 };

```

```

1 const imageChange = (i) => {
2   const dataTransfer = new DataTransfer();
3   files
4     .filter((file) => file !== files[i])
5     .forEach((file) => {
6       dataTransfer.items.add(file);
7     });
8
9   selectImage.current.files = dataTransfer.files;
10  setFiles(Array.from(selectImage.current.files));
11 };

```

```

1 const imgCancle = () => {
2   selectImage.current.value = "";
3   setFiles([]);
4   setImgSrc([]);
5   setPrevImgSrc(prevImg);
6 };

```

```

1 const deletePrevSrc = (code) => {
2   const images = prevImgSrc.filter(
3     (image) => image.productImageCode !== code
4   );
5   setPrevImgSrc(images);
6 };
7
8 const selectImage = useRef("");
9 const deleteImgSrc = (no) => {
10  let newImgSrc = [...imgSrc];
11  newImgSrc.splice(no, 1);
12  setImgSrc(newImgSrc);
13 };

```

반려동물 동반

1.openAPI 파싱

```
public ResponseEntity<JSONObject> fetch() throws Exception {
    int no = 0;
    HttpURLConnection conn = null;
    InputStream stream = null;
    JSONObject result = null;

    for (int n = 1; n <= 8; n++) {
        no = n;
        URL url = new URL("https://api.odcloud.kr/api/15111389/v1/uddi:41944402-8249-4e45-9e9d-
a52d0a7db1cc" + "?serviceKey="
        + "ServiceKey"
        + "&page=" + no + "&perPage=3000" + "&returnType=JSON");

        conn = (HttpURLConnection) url.openConnection();
        stream = getNetworkConnection(conn);
        result = readStreamToString(stream);
    }
    return ResponseEntity.status(HttpStatus.OK).body(result);
}
```

약 2만 4천 개의 데이터 처리로 인한 runtime 오류 발견
테스트를 통해 최대 3000개까지 처리 가능 확인 후 반복문 처리

반려동물 동반

1. openAPI 파싱

HttpURLConnection을 통해 네트워크 연결 설정 및 데이터 가져오기

```
private InputStream getNetworkConnection(HttpURLConnection conn) throws Exception {
    conn.setConnectTimeout(3000);
    conn.setReadTimeout(3000);
    conn.setRequestMethod("GET");
    conn.setDoInput(true);

    if (conn.getResponseCode() != HttpURLConnection.HTTP_OK) {
        throw new IOException("HTTP error code : " + conn.getResponseCode());
    }
    return conn.getInputStream();
}
```

연결 및 읽기 타임아웃을 설정하고,
HTTP 응답 코드가 200(OK)이 아니면 예외처리

파싱한 데이터를 DB 컬럼에 맞춰 저장

```
private JSONObject readStreamToString(InputStream stream) {
    StringBuilder result = new StringBuilder();

    try {
        BufferedReader br = new BufferedReader(new InputStreamReader(stream, "UTF-8"));

        String readLine;
        StringBuffer responseBuffer = new StringBuffer();

        while ((readLine = br.readLine()) != null) {
            responseBuffer.append(readLine);
            result.append(readLine + "\n\r");
        }

        String responseData = responseBuffer.toString();
        JSONObject jsonResponse = new JSONObject(responseData);
        JSONArray parsing = jsonResponse.getJSONArray("data");

        for (int i = 0; i < parsing.length(); i++) {
            JSONObject vo = parsing.getJSONObject(i);
            String name = vo.getString("시작점");
            ...
            Parsing loca = new Parsing();

            loca.setName(name);
            ...
            loca.setOperatingHours(operatingHours);

            service.create(loca);
        }
        return null;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

반려동물 동반

2. 메인 카테고리 선택에 따라 다른 서브 카테고리 출력

메인 카테고리

카테고리 선택

```

1 const selectMainCat = [
2   { value: 0, name: "카테고리 선택" },
3   { value: 1, name: "반려의료" },
4   { value: 2, name: "반려동반여행" },
5   { value: 3, name: "반려동물 식당 카페" },
6   { value: 4, name: "반려동물 서비스" },
7 ];
8
9 const selectSubCat = [
10  { value: 0, name: "서브 카테고리 선택" },
11  { value: 1, name: "동물약국", parent: 1 },
12  { value: 2, name: "동물병원", parent: 1 },
13  { value: 3, name: "미술관", parent: 2 },
14  { value: 4, name: "문화회관", parent: 2 },
15  { value: 5, name: "랜선", parent: 2 },
16  { value: 6, name: "여행자", parent: 2 },
17  { value: 7, name: "박물관", parent: 2 },
18  { value: 8, name: "카페", parent: 3 },
19  { value: 9, name: "식당", parent: 3 },
20  { value: 10, name: "반려동물용품", parent: 4 },
21  { value: 11, name: "미용", parent: 4 },
22  { value: 12, name: "위탁관리", parent: 4 },
23];

```

메인 카테고리를 선택해야
서브 카테고리 선택창이 보이도록 설정

```

1 <div id="select">
2   <span>메인 카테고리</span>
3   <select
4     onChange={handleSelectMainCat}
5     value={mainCat}
6     id="mainCat"
7   >
8     {selectMainCat.map((item) => {
9       return (
10         <option value={item.value} key={item.value}>
11           {item.name}
12         </option>
13       );
14     ))}
15   </select>
16 </div>
17 <div id="select">
18   {mainCat === 0 ? (
19     <>
20     <span>서브 카테고리</span>
21     <select
22       onChange={handleSelectSubCat}
23       value={subCat}
24       id="subCat"
25     >
26       {filterCat.map((item) => {
27         return (
28           <option value={item.value} key={item.value}>
29             {item.name}
30           </option>
31         );
32       }));
33     </select>
34   </>

```

반려동물 동반

3. 출력되는 리스트에 따른 지도 매핑

| 기관명 | 주소 | 조회수 |
|----------|--------------------|-----|
| 비전액국 | 경기도 평택시 비전동 1012-3 | 0 |
| 비전하늘액국 | 경기도 평택시 비전동 1012-2 | 0 |
| 항상액국 | 경기도 평택시 비전동 995 | 0 |
| 365세설당액국 | 경기도 안성시 인자동 414-1 | 0 |
| 참사랑액국 | 경기도 안성시 인자동 352-12 | 0 |
| 서안액국 | 경기도 안성시 서안동 14 | 0 |
| 안성시창액국 | 경기도 안성시 인자동 8-3 | 0 |
| 안성바른액국 | 경기도 안성시 서안동 78 | 0 |
| 현대액국 | 경기도 안성시 동분동 45 | 0 |
| 안성종로액국 | 경기도 안성시 서안동 82-1 | 0 |

« < 1 2 3 4 5 6 7 8 9 10 > »



출력되는 content가 변경될 때마다 list 새로 생성
생성된 리스트를 지도에 매핑

```

1  useEffect(() => {
2    let maplist = [];
3    content.forEach((item) => {
4      maplist.push({
5        title: item.name,
6        latlng: { lat: item.latitude, lng: item.longitude },
7      });
8    });
9    setList(maplist);
10   setLat(maplist[0]?.latlng.lat);
11   setLng(maplist[0]?.latlng.lng);
12 }, [content]);
  
```

```

<Map
  center={{
    // 지도의 중심좌표
    lat: lat,
    lng: lng,
  }}
  level={3}
>
{list.map((position, index) => (
  <MapMarker
    key={`${position.title}-${position.latlng}`}
    position={position.latlng}
    image={
      src: "https://t1.daumcdn.net/localimg/localimages/07/mapapidoc/markerStar.png",
      sizes: {
        width: 24,
        height: 35,
      },
    },
    title={position.title}
  />
))
</Map>
  
```

질문 게시판

1. '좋아요'한 글만 보기

정렬
작성일 최신순
동물
전체
채택 답변 유무
전체
검색 조건
제목
검색어를 입력하세요

전체 0건
 좋아요한 글만 보기

| 질문 번호 | 채택 여부 | 제목 | 작성자 | 작성일 | 좋아요수 | 조회수 |
|---------|-------|----|-----|-----|------|-----|
| « « » » | | | | | | |

```
● ● ●

public Page<UserQnaQuestionBoard> viewliked(List<UserQnaQuestionBoard> list, Pageable pageable) {
    int start = (int)pageable.getOffset();
    int end = Math.min((start + pageable.getPageSize()), list.size());
    List<UserQnaQuestionBoard> sublist = list.subList(start, end);

    // 리스트를 역순으로 정렬
    Collections.reverse(sublist);

    return new PageImpl<>(sublist, pageable, list.size());
}
```

좋아요 버튼을 누른 순서대로 목록 출력

질문 게시판

1. '좋아요'한 글만 보기

```

1  {user.userId !== undefined && user.userRole !== 'ROLE_ADMIN' ? (
2    <div id="like">
3      <div className="form-check form-switch">
4        <input
5          className="form-check-input"
6          type="checkbox"
7          id="flexSwitchCheckDefault"
8          onChange={(e) => filtering(e)}
9        />
10       <label className="form-check-label" htmlFor="flexSwitchCheckDefault">
11         좋아요한 글만 보기
12       </label>
13     </div>
14   </div>
15 ) : (
16   <></>
17 )

```

좋아요 목록 출력 여부를
e.target.checked를 통해 확인

로그인한 유저에게만
좋아요한 글만 보기 버튼이 출력

```

const filtering = async (e) => {
  const isChecked = e.target.checked;
  setLiked(isChecked);
  if (isChecked) {
    const response = await getliked(page, true);
    setQuestions(response.data);
    setTotalPage(response.data?.totalPages);
  } else {
    const response = await getliked(page,
false);
    setQuestions(response.data);
    setTotalPage(response.data?.totalPages);
  }
};

```

질문 게시판

1. '좋아요'한 글만 보기

```
@GetMapping("/userQuestion")
public ResponseEntity<Page<UserQnaQuestionBoard>> viewliked(@RequestParam(name="liked", defaultValue = "false") boolean liked,
                                                               @RequestParam(name="page", defaultValue = "1") int page){

    Sort sort = Sort.by("userQuestionBoardCode").descending();
    Pageable pageable = PageRequest.of(page-1, 10, sort);

    QUserQnaQuestionBoard qUserQnaQuestionBoard = QUserQnaQuestionBoard.userQnaQuestionBoard;
    QUserQnaQuestionLike qUserQnaQuestionLike = QUserQnaQuestionLike.userQnaQuestionLike;
    ...

    if(principal instanceof User) {
        // 유저의 정보를 담아 넘겨가야요한 목록 출력
        User user = (User) principal;
        if(liked){
            List<UserQnaQuestionBoard> list = queryFactory
                .selectFrom(qUserQnaQuestionBoard)
                .join(qUserQnaQuestionLike).on(qUserQnaQuestionBoard.userQuestionBoardCode.eq(qUserQnaQuestionLike.userQuestionBoardCode))
                .where(qUserQnaQuestionLike.userId.eq(user.getUserId()))
                .fetch();

            Page<UserQnaQuestionBoard> likedlist = service.viewliked(list, pageable);
            return ResponseEntity.status(HttpStatus.OK).body(likedlist);
        } else{
            // 기본적인 목록 출력
            return ResponseEntity.status(HttpStatus.OK).body(service.viewAll(builder, pageable));
        }
    }
    return null;
}
```

유저가 선택한 좋아요 목록 출력 여부를 liked 변수로 받아 좋아요 목록 출력

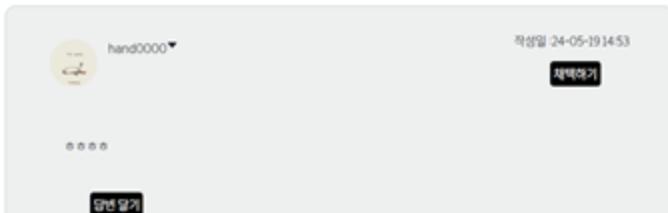
질문 게시판 2. 채택하기

채택하기 테스트를 위한

작성일: 24-05-19 02:52



질문합니다.



채택된 답변 하위에 채택된 답변 제외 출력

```
//...顶部右侧 顶部右侧 顶部右侧  
{topChoose.length === 0 ? (  
    <div key={topChoose.userAnswerBoardCode}>  
        <div id="topanswer">  
            ...  
            <div>  
                {user.userId === question.userId ? (  
                    //...顶部右侧 顶部右侧 顶部右侧  
                    <>  
                        <button  
                            onClick={() =>  
                                chooseDelete(  
                                    topChoose.userQuestionBoardCode  
                                )  
                            }  
                        >  
                        删除 顶部右侧  
                    </button>  
                ) : (  
                    <div>  
                        <button id="choosebutton" onClick={() => choose(answer)}>  
                            顶部右侧</button>  
                    </div> : (  
                        {user.userId === answer.user.userId && editA === null ? (  
                            <div>
```

작성자가 접속한 경우

채택한 답변이 없을 경우 '채택하기' 버튼 출력

채택한 답변이 있을 경우 채택된 답변에만 '채택 취소하기' 출력

질문 게시판 2. 채택하기

```
// 6-1. 선택 처리
@PostMapping("/userQuestion/answerChoose")
public ResponseEntity<UserQnaAnswerChoose> chooseAnswer(UserQnaAnswerChoose vo){
    service.chooseAnswer(vo);

    // 질문 시장에 상태 'Y'로 업데이트
    UserQnaQuestionBoard result = service.view(vo.getUserQuestionBoardCode());
    result.setUserQuestionBoardStatus('Y');
    service.update(result);
    return null;
}

// 6-2. 선택 취소하기
@DeleteMapping("/userQuestion/answerChoose/{code}")
public void deleteChoose(@PathVariable(name="code") int code){
    UserQnaAnswerChoose chooseAnswer = service.getChoose(code);

    // 퀴즈 시장에 상태 'N'으로 변경
    UserQnaQuestionBoard result = service.view(code);
    result.setUserQuestionBoardStatus('N');
    service.update(result);
    service.deleteChoose(chooseAnswer.getChooseCode());
}
```

채택하는 경우

answerChoose 객체 형태로 질문 코드, 답변 코드 정보를 담아 저장

```
// 6-3. 채택된 답변 보기
@GetMapping("/public/userQuestion/answerChoose/{code}")
public ResponseEntity<UserQnaAnswerBoardDTO> getAnswer(@PathVariable(name="code") int code){
    // questionBoardCode로 채택 답변 정보 찾아서 그의 연결된 answer 조회
    UserQnaAnswerChoose choose = service.getChoose(code);
    UserQnaAnswerBoard chooseAnswer;
    if(choose != null){
        chooseAnswer = answerService.viewAnswer(choose.getUserAnswerBoardCode());
    } else {
        chooseAnswer= null;
        return ResponseEntity.status(HttpStatus.OK).build();
    }
    ...
    // 같은 상위 답변 코드로 하위 답변 가져오기
}
```

질문 게시판

3. 댓글, 대댓글 등 접속자 조건에 따른 버튼 권한

```
// 채택된 답변이 있는 경우  
{topChoose.length !== 0 ? (<></>) : (<></>)}  
  
// 접속자가 질문 작성자인 경우  
{user.userId === question.userId ? (<></>):( <></>)}  
  
// 질문 작성자가 하위 답변 작성자인 경우  
{question.userId === reanswer.user.userId ? (<></>):( <></>)}  
  
// 비회원의 경우  
{user.userId !== undefined ? (<></>) : (<></>)}
```

원데이 클래스

Back

사용자가 입력하고 데이터를 받는 부분은 DTO

```

@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@DynamicInsert
@Builder
@Table(name="oneday_class_board")
public class ClassBoard {

    @Id
    @Column(name = "odc_code")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int odcCode;

    @Column(name = "odc_title")
    private String odcTitle;

    @Column(name = "odc_content")
    private String odcContent;

    @Column(name = "odc_accompanying")
    private char odcAccompanying;

    @Column(name = "odc_regi_date")
    private Timestamp odcRegiDate;

    @Column(name = "odc_start_date")
    private Date odcStartDate;

    @Column(name = "odc_last_date")
    private Date odcLastDate;

    @ManyToOne
    @JoinColumn(name="user_id")
    private User user;

    @OneToMany(mappedBy = "classBoard")
    private List<ClassBoardMainImage> images;
}

```

```

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class ClassBoardDTO {

    private int odcCode;
    private String odcTitle;
    private String odcAccompanying;
    private String odcContent;
    private String odcRegiDate;
    private String odcStartDate;
    private String odcLastDate;
    private MultipartFile file;
    private int imageCode;
    private String imageURL;
    private String userId;
}

```

```

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
@DynamicInsert
@Table(name = "oneday_class_board_main_image")
public class ClassBoardMainImage {

    @Id
    @Column(name = "odc_image_code")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int odcImageCode;

    @ManyToOne
    @JoinColumn(name = "odc_code")
    @JsonIgnore
    private ClassBoard classBoard;

    @Column(name = "odc_main_image")
    private String odcMainImage;
}

```

메인 이미지를 받기 위한 테이블과 클래스 구분

원데이 클래스 등록에 대한 필요한 정보들 컬럼

원데이 클래스 DAO & service

```
public interface ClassBoardDAO extends JpaRepository<ClassBoard, Integer> {
}
```

ClassBoardDAO

```
public interface ClassBoardMainImageDAO extends JpaRepository<ClassBoardMainImage, Integer> {
    @Query(value="SELECT * FROM oneday_class_board_main_image
                 WHERE odc_code = :code", nativeQuery = true)
    ClassBoardMainImage findByCode(@Param("code")Integer code);
}
```

ClassBoardMainImageDAO

service

```
@Service
public class OneDayClassService {
    @Autowired
    private ClassBoardDAO dao;
    @Autowired
    private ClassBoardMainImageDAO img;

    public ClassBoardMainImage createImg(ClassBoardMainImage vo){
        return img.save(vo);
    }

    public ClassBoardMainImage viewImg(int odcCode){
        return img.findById(odcCode);
    }

    public ClassBoardMainImage updateImages(ClassBoardMainImage vo){
        if (img.existsById(vo.getOdcImageCode())){
            return img.save(vo);
        }
        return null;
    }

    public void deleteImg(int odcCode){
        img.deleteById(odcCode);
    }

    public ClassBoard insert(ClassBoard vo){
        return dao.save(vo);
    }

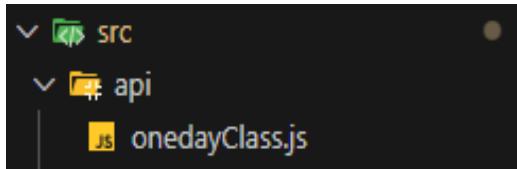
    public List<ClassBoard> viewAll(){
        return dao.findAll();
    }

    public ClassBoard view(int odcCode){
        return dao.findById(odcCode).orElse(null);
    }

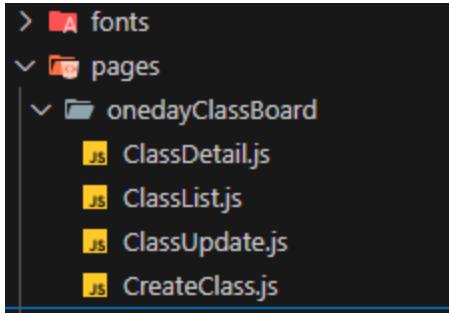
    public ClassBoard update(ClassBoard vo){
        if (dao.existsById(vo.getOdcCode())){
            return dao.save(vo);
        }
        return null;
    }

    public ClassBoard delete(int odcCode){
        if (dao.existsById(odcCode)){
            dao.deleteById(odcCode);
        };
        return null;
    }
}
```

원데이 클래스 API & pages



Axios



Security
Tooken

```
const getToken = () => {
  return localStorage.getItem("token");
};

const authorize = axios.create
  ({ baseURL: "http://localhost:8080/compagno/" });

authorize.interceptors.request.use((config) => {
  const token = getToken();
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});
```

원데이클래스 ClassList

Compagno



OnedayClassList 기능 관련 매팅

```
@GetMapping("/public/ClassBoard")
public ResponseEntity<List<ClassBoard>> viewAll() {
    List<ClassBoard> list = service.viewAll();
    return ResponseEntity.status(HttpStatus.OK).body(list);
}
```

SpringBoot에서의 수정관련 기능을 리액트 API로 연결

```
export const viewAllClass = async () => {
    return await instance.get("ClassBoard");
};
```

원데이 클래스 ClassCreate

Compagno



```
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@Builder  
public class ClassBoardDTO {  
  
    private int odcCode;  
    private String odcTitle;  
    private String odcAccompanying;  
    private String odcContent;  
    private String odcRegiDate;  
    private String odcStartDate;  
    private String odcLastDate;  
    private MultipartFile file;  
    private int imageCode;  
    private String imageURL;  
    private String userId;  
}
```

DTO를 통해 입력 받을 코드들을
분리하여 관리

```
export const addOnedayClass = async (data) => {  
    console.log(data);  
    return await authorize.post("ClassBoard", data);  
};
```

원데이클래스 ClassCreate



```

@RestController
@RequestMapping("/compagno/*")
@CrossOrigin(origins = {"*"}, maxAge = 6000)
public class OneDayClassController {

    @Autowired
    private OneDayClassService service;

    @Value("${spring.servlet.multipart.location}")
    private String uploadPath;

    @PostMapping("/ClassBoard")
    public ResponseEntity<ClassBoard> insert(ClassBoardDTO dto) throws IOException, ParseException {
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        Date startDate = sdf.parse(dto.getOdcStartDate());
        Date lastDate = sdf.parse(dto.getOdcLastDate());

        ClassBoard vo = ClassBoard.builder()
            .odcTitle(dto.getOdcTitle())
            .odcContent(dto.getOdcContent())
            .odcStartDate(startDate)
            .odcLastDate(lastDate)
            .odcAccompanying(dto.getOdcAccompanying().charAt(0))
            .user(userInfo())
            .build();

        ClassBoard result = service.insert(vo);

        if (dto.getFile() != null) {
            ClassBoardMainImage imgVo = new ClassBoardMainImage();
            String fileName = dto.getFile().getOriginalFilename();
            String uuid = UUID.randomUUID().toString();
            String saveName = uploadPath + File.separator + uuid + "_" + fileName;
            Path savePath = Paths.get(saveName);
            dto.getFile().transferTo(savePath);

            imgVo.setOdcMainImage(saveName);
            imgVo.setClassBoard(result);
            service.createImg(imgVo);
        }

        if (result != null) {
            return ResponseEntity.status(HttpStatus.CREATED).body(result);
        }
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).build();
    }
}

```

원데이클래스 ClassCreate

SpringBoot에서의 수정관련 기능을 리액트 API로 연결

Compagno



```
export const viewClass = async (code) => {
  return await instance.get("ClassBoard/" + code);
};
```

OnedayClassView 기능관련 매팅

```
@GetMapping("/public/ClassBoard/{odcCode}")
public ResponseEntity view(@PathVariable(name = "odcCode") int odcCode) {
    ClassBoard vo = service.view(odcCode);
    return ResponseEntity.status(HttpStatus.OK).body(vo);
}
```

ClassBoard에 내가 입력 받을 컬럼들과 User와 조인하여 가져올 컬럼들을 정보기제

원데이클래스 Security

Compagno

동록 등록 게시판 서비스 글자시험 디비전과별 logout

유저 닉네임 : 편과자
이메일 : kwanjaja23@naver.com

등록 날짜 : Invalid date

클래스 시작 날짜 : 24-05-20
클래스 마지막 날짜 : 24-05-28

동반 가능 여부 : X

제목
도 개인사이트(광고지도 예약연락 있다-->)
상세 내용
우리의 예관동물들이 어떤 날 어느 상황에 따라 우리는 마음을 조심스레 다가가다가 가끔 마음을 받고 읊씩해진적이 많다. 그래서 우리는 우리의 반려동물들의 심리상황과 간접상태를 항상 면밀히 알고 사랑해줘야 한다 두동역 ~~~

수정 취소

```
{user.userId == odcClass.user?.userId ||
  user.userRole == "ROLE_ADMIN" ? (
    <button onClick={onUpdate}>수정</button>
  ) : (
    <></>)
  <button onClick={onBack}>취소</button>
```

비회원과 회원 / 등록회원과 관리자에 대한
관한을 Cecurity를 삼항연산자를
통해 유무를 설정

원데이클래스 ClassDelete

SpringBoot에서의 수정관련 기능을 리액트 API로 연결

Compagno



```
export const deleteClass = async (code) => {
  return await authorize.delete("ClassBoard/" + code);
};
```

```
@DeleteMapping("/ClassBoard/{odcCode}")
public ResponseEntity delete(@PathVariable(name = "odcCode") int odcCode) {

    ClassBoard prev = service.view(odcCode);
    ClassBoardMainImage image = service.viewImg(odcCode);

    File file = new File(image.getOdcMainImage());
    file.delete();

    service.deleteImg(image.getOdcImageCode());
    service.delete(odcCode);

    return ResponseEntity.ok().build();
}

public User userInfo() {
    SecurityContext securityContext = SecurityContextHolder.getContext();
    Authentication authentication = securityContext.getAuthentication();
    Object principal = authentication.getPrincipal();

    if (principal instanceof User) {
        User user = (User) principal;
        return user;
    }
    return null;
}
```

원데이클래스 ClassUpdate

SpringBoot에서의 수정관련 기능을 리액트 API로 연결

Compagno



```
export const updateClass = async (data) => {
  return await authorize.put("ClassBoard", data);
};
```

```
@PutMapping("/ClassBoard")
public ResponseEntity update(ClassBoardDTO dto) throws ParseException, IOException {
  if(dto.getFile() != null) {
    File file = new File(dto.getImageURL());
    file.delete();
    service.deleteImg(dto.getOdcCode());
    ClassBoardMainImage imgVo = new ClassBoardMainImage();

    String fileName = dto.getFile().getOriginalFilename();
    String uuid = UUID.randomUUID().toString();
    String saveName = uploadPath + File.separator + "ClassBoard"
      + File.separator + uuid + "-" + fileName;
    Path savePath = Paths.get(saveName);

    dto.getFile().transferTo(savePath);
    imgVo.setOdcMainImage(saveName);
    imgVo.setClassBoard(ClassBoard.builder().odcCode(dto.getOdcCode()).build());
    service.createImg(imgVo);
  }

  SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
  Date startDate = sdf.parse(dto.getOdcStartDate());
  Date lastDate = sdf.parse(dto.getOdcLastDate());

  ClassBoard vo = ClassBoard.builder()
    .odcCode(dto.getOdcCode())
    .user(User.builder()
      .userId(userInfo().getUserId()).build())
    .odcTitle(dto.getOdcTitle())
    .odcContent(dto.getOdcContent())
    .odcRegiDate(Timestamp.valueOf(dto.getOdcRegiDate()))
    .odcStartDate(startDate)
    .odcLastDate(lastDate)
    .odcAccompanying(dto.getOdcAccompanying().charAt(0))
    .build();
  service.insert(vo);
}

return ResponseEntity.ok().build();
}
```

Q&A 게시판

1. 비밀글 설정 버튼 누를 때 비밀번호 입력 폼 가시화



```

1 <div id="secret" class="form-check form-switch">
2   <input class="form-check-input" type="checkbox" id="flexSwitchCheckDefault" onChange={(e) => setPassword(e)} />
3   <label class="form-check-label" for="flexSwitchCheckDefault">비밀글 설정</label>
4 {pwd === 1 ? (<
5   <Form.Control
6     type="password"
7     placeholder="비밀글 비밀번호"
8     value={secret}
9     onChange={(e) => setSecret(e.target.value)}
10    /></>) : (<></>)}
11 </div>

```



```

1 const [pwd, setPwd] = useState(0);
2
3 const setPassword = async (e) => {
4   const isChecked = e.target.checked;
5   if (isChecked) {
6     setPwd(1);
7   } else {
8     setPwd(0);
9     setSecret("");
10  }
11 };

```

비밀글 비밀번호 입력 후
비밀글 설정 해제 시 입력된 비밀번호 초기화

Q&A 게시판

2. 비밀글 조회 시 비밀번호 확인

```
// 모달을 구현
const [modalShow, setModalShow] = useState(false);

const CenterModal = (props) => {
  const [pwd, setPwd] = useState("");

  const pwdCheck = () => {
    if (secretPwd === pwd) {
      navigate("/compagno/question/detail/" + code);
    } else {
      alert("비밀번호가 일치하지 않습니다!");
      document.querySelector("#password").focus();
    }
  };
}
```

비밀글이고, 관리자가 아닌 경우
글에 접근 시 modal을 통해 비밀번호 입력창으로 연결

```
<td id="title">
  {question.secret === "" || question.secret == null ? (
    // 비밀번호가 걸려있지 않을 때
    ...
  ) : (
    <>
      {user.userRole === "ROLE_ADMIN" ? (
        // 관리자일 때
        ...
      ) : (
        <>
          // 비밀번호가 걸려있고, 관리자가 아닐 때
          <a href={`/compagno/question/detail/${question.qnaQCode}`} onClick={(e) => {
            e.preventDefault();
            setModalShow(true);
            setSecret(question.secret);
          }}>
            </a>
            ...
          </td>
        
```

Q&A 게시판

2. 비밀글 조회 시 비밀번호 확인

[동물 등록](#) [구조 등록](#) [게시판](#) [서비스](#) [공지사항](#)

전체 38건

| 질문 번호 | 답변 여부 | 제목 | 작성자 | 작성일 |
|-------|-------|------------------------|----------|----------------|
| 42 | N | 비밀글이지용 | user0002 | 24-05-19 03:08 |
| 41 | N | 이미지를 수정하기 위해 사진을 넣겠습니다 | user0002 | 24-05-19 02:44 |
| 40 | N | 비밀번호 | user0002 | 24-05-19 02:43 |
| 39 | N | 다시~!! | user0002 | 24-05-19 12:59 |
| 38 | Y | 답변 등록되면 | user0001 | 24-05-19 12:07 |
| 37 | N | ? | user0002 | 24-05-18 04:06 |
| 36 | N | 로그 | user0002 | 24-05-18 01:07 |
| 35 | N | ? | user0002 | 24-05-18 12:26 |
| 34 | N | | user0002 | 24-05-18 12:25 |
| 33 | Y | 등록합니다.. | user0002 | 24-05-18 12:18 |

Q&A 게시판

3. 관리자 답변 작성, 삭제 시 상태변화



```
// 답변 등록 시 status 업데이트  
QnaQBoard update = questionService.view(dto.getQnaQCode());  
update.setQnaACode(result);  
update.setQnaQStatus("Y");  
update.setQnaQDateUpdate(date);
```



```
// 답변 삭제 시 status 업데이트  
QnaQBoard update = questionService.view(target.getQnaQCode().getQnaQCode());  
update.setQnaQStatus("N");  
questionService.update(update);
```

Q&A 게시판

3. 관리자 답변 작성, 삭제 시 상태변화

전체 36건

제목

검색어를 입력하세요

Q. 조회

질문 등록

| 질문 번호 | 답변 여부 | 제목 | 작성자 | 작성일 |
|-------|-------|---------|----------|----------------|
| 40 | N | 비밀번호 | user0002 | 24-05-19 02:43 |
| 39 | N | 다시~!! | user0002 | 24-05-19 12:59 |
| 38 | Y | 답변 등록되면 | user0001 | 24-05-19 12:07 |
| 37 | N | ? | user0002 | 24-05-18 04:06 |
| 36 | N | 로그 | user0002 | 24-05-18 01:07 |
| 35 | N | ? | user0002 | 24-05-18 12:26 |
| 34 | N | | user0002 | 24-05-18 12:25 |
| 33 | Y | 등록합니다.. | user0002 | 24-05-18 12:18 |
| 32 | N | | user0002 | 24-05-18 11:59 |
| 31 | N | | user0002 | 24-05-18 11:58 |

Q&A 게시판

4. 이미지 수정

```
if (dto.getImages() != null) {
    List<String> imagesList = dto.getImages()
        .stream().map(image -> image.getQnaQUrl()).collect(Collectors.toList());

    List<QnaQBoardImage> list = service.viewImg(dto.getQnaQCode());

    // 기본으로해서 사용한 모든 이미지 삭제
    for (QnaQBoardImage image : list) {

        if ((dto.getImages() != null && !imagesList.contains(image.getQnaQUrl())) ||
            (dto.getImages() == null)) {
            File file = new File(image.getQnaQUrl());
            file.delete();

            service.deleteImg(image.getQnaQImgCode());
        }
    }
} else {
    // 기본으로서 모든 이미지를 모두 삭제하는 경우 모든 이미지 삭제
    List<QnaQBoardImage> list = service.viewImg(dto.getQnaQCode());
    for (QnaQBoardImage image : list) {
        File file = new File(image.getQnaQUrl());
        file.delete();

        service.deleteImg(image.getQnaQImgCode());
    }
}
```

```
if (dto.getFiles() != null) {
    // 기본으로서 사용한 모든 파일 삭제
    for (MultipartFile file : dto.getFiles()) {
        QnaQBoardImage img = new QnaQBoardImage();

        String fileName = file.getOriginalFilename();
        String uuid = UUID.randomUUID().toString();
        String saveName = uploadPath + File.separator + "QnaQ" + File.separator + uuid + "_"
            + fileName;

        Path savePath = Paths.get(saveName);
        file.transferTo(savePath);

        img.setQnaQUrl(saveName.substring(38));
        img.setQnaQCode(dto.getQnaQCode());

        service.createImg(img);
    }
}
```

수정 시 추가하려는 이미지 추가

이전 이미지와 클라이언트가 넘긴 이전 이미지 비교하여 삭제

Q&A 게시판

4. 이미지 수정

이미지를 수정하기 위해 사진을 넣겠습니다



유개02

안녕하세요



작성일: 24-05-19 02:26

수정 삭제

이전 이미지 클릭 시 클릭한 이미지를 삭제하고
삭제하지 않을 시 이미지를 유지

수정할 때 추가하고자 하는 이미지 추가

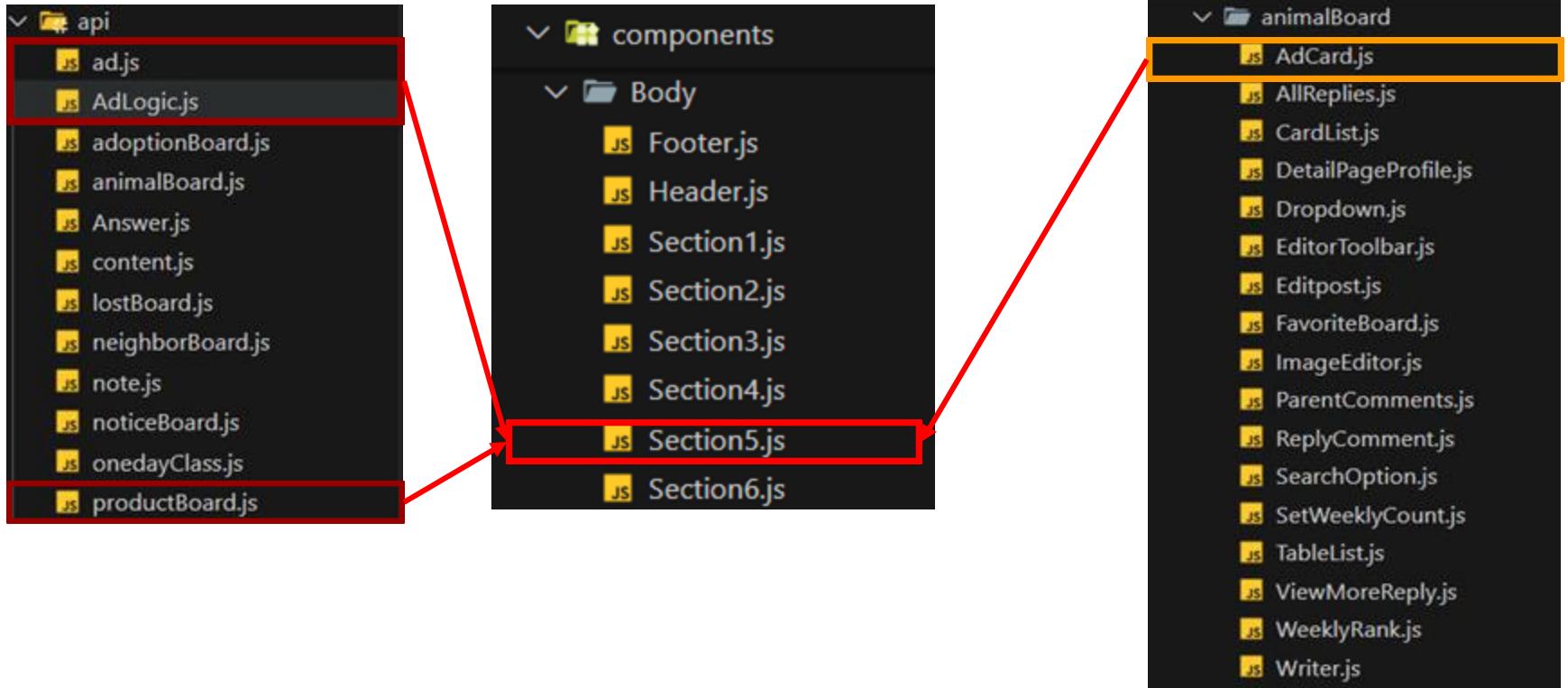
```
// 이미지 삭제 시 이미지 헤더
const deleteImage = (code) => {
  setEditQ((prev) => {
    const images = prev.images.filter((image) => image.qnaQImgCode !== code);
    return { ...prev, images: images };
  });
};

// 수정 후의 이미지 파일 추가 및 관리
const preview = (e) => {
  const files = Array.from(e.target.files);
  setShowImages((prev) => [...prev, ...files]);

  const imageLists = e.target.files;
  let imageUrlLists = [...showImages];

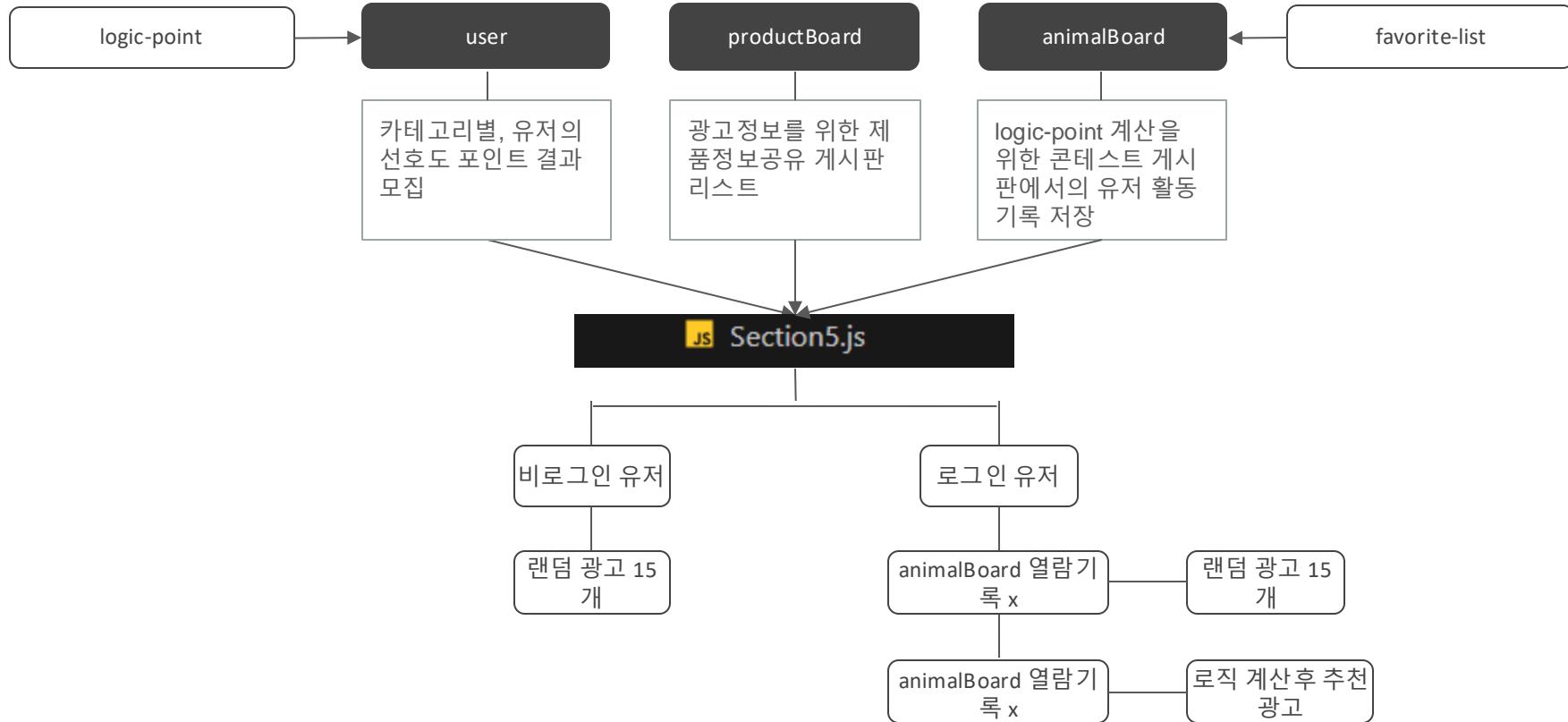
  for (let i = 0; i < imageLists.length; i++) {
    const currentImageUrl = URL.createObjectURL(imageLists[i]);
    imageUrlLists.push(currentImageUrl);
  }
  setShowImages(imageUrlLists);
};

// 수정 적용(선택자)
const handleDeleteImage = (id) => {
  setShowImages(showImages.filter((_, index) => index !== id));
  setImages(images.filter((_, index) => index !== id));
};
```



광고

1.logic structure



광고

2-1. 로직 구현 : 초기값 세팅

```
const [ads, setAds] = useState([]);  
const getProductsAPI = async () => {  
  const response = await showProducts();  
  // console.log(response.data);  
  setAds(response.data);  
}; // 기존 productBoard 정보
```

| logic_code | user_id | category_code | input_value | total_score |
|------------|----------|---------------|-------------|-------------|
| 1 | sang0001 | 1 | 0.00 | 0.00000 |
| 2 | sang0001 | 2 | 0.00 | 0.00000 |
| 3 | sang0001 | 3 | 0.40 | 0.69641 |
| 4 | sang0001 | 4 | 0.00 | 0.00000 |

/ 현재 카테고리 포인트 가져오기

```
const [points, setPoints] = useState([]);  
const currentPointAPI = async () => {  
  if (Object.keys(user).length !== 0) {  
    const response = await getCurrentPoint(user.userId);  
    setPoints(response.data);  
  }  
};
```

광고 2-1. 로직 구현 : 비로그인 유저

```
const [randomAds, setRandomAds] = useState([]); // 비로그인 광고
// 로그인 안했을때
const nonFilter = () => {
  if (Object.keys(user).length === 0) {
    console.log(ads);
    const shuffleArray = (array) => {
      return array.sort(() => Math.random() - 0.5);
    };
    const noneFilteredArr = shuffleArray(
      ads.filter((ad) => ad.productBoardGrade >= 3.5)
    ).slice(0, 15);
    setRandomAds(noneFilteredArr);
  }
};
```

광고

2-1. 로직 구현 : 로그인 유저

```
// 로그인 했을때
const adListFilter = () => {
  console.log("로그인함");
  //카테고리 포인트가 가장 높은 카테고리
  // 1. 가장 높은 점수
  const maxTotalScore = Math.max(...points.map((point) => point.totalScore));
```



현재 로그인한 유저의 카테고리 포인트리스트중에서 column total_value의 max 값을 가져온다

```
if (maxTotalScore === 0) {
  // 포인트 최댓값이 0일때
  // 사실상 비로그인상태와 같음
  const shuffleArray = (array) => {
    return array.sort(() => Math.random() - 0.5);
  };
  const noneFilteredArr = shuffleArray(ads);
  setNewAds(noneFilteredArr);
  //=====
} else {
  setNewAds([]);
```

1. **maxTotalScore === 0**
=> 유저가 아직 animalBoard게시판 열람을 하지 않음
=> 사실상 비로그인 유저와 같음

1. **maxTotalScore !== 0**
=> 유저가 최소 하나 이상의 animalBoard를 열람 함.
=> 기존 배열 삭제 후 다음 로직 실행

광고

2-1. 로직 구현 : 로그인 유저

```
console.log(maxTotalScore);
// 2. 이 요소 가져오기
const highestScoreCate = points.find(
  (point) => point.totalScore === maxTotalScore
);
// console.log(maxTotalScore); // 초기값이 0이라서 필터링으면 없어짐

// 두 번째로 큰 요소(가장 큰 요소 제거)리스트
const exceptHighest = points.filter(
  (point) => point.totalScore !== maxTotalScore
);
// 그 중에서 다시 가장 큰 값 뽑기
const secondMaxTotalScore = Math.max(
  ...exceptHighest.map((exception) => exception.totalScore)
);
if (secondMaxTotalScore === 0) {
  // 가장 관심있는 첫순위는 뽑아짐, 근데 두번째가.
  // 제일 큰 요소만 반환
  const highestOnlyArr = ads
    .filter(
      (ad) =>
        ad.animalCategory.animalCategoryCode ===
        highestScoreCate.animalCategory.animalCategoryCode
    )
    .slice(0, 12);
  setNewAds(highestOnlyArr);
} else {
  setNewAds([]);
}
```

1. **point.totalScore === maxTotalScore**
=> 해당 유저의 total_score중 가장 높은 값을 기준으로 해당 카테고리 요소를 가져온다.

1. **exceptHighest**

=> 두 번째로 높은 totalScore의 요소를 가져오기 위해 가장 큰 요소를 리스트에서 제거한다.
=> 동일 과정을 반복하여 두번째로 큰 요소를 가져온다.

1. **secondMaxTotalScore === 0**

=> 유저가 단 하나의 카테고리와 관련된 animalBoard글을 열람함
=> 제일 큰요소만이 담긴 광고리스트 반환

광고

2-1. 로직 구현 : 로그인 유저

```
{  
    setNewAds([]);  
    // console.log(secondMaxTotalScore);  
    // console.log(points);  
    const secondHighestScoreCate = points.find(  
        (point) => point.totalScore === secondMaxTotalScore  
    ),  
    console.log(secondHighestScoreCate);  
  
    // 세 번째로 큰 요소(두 번째로 큰 요소가 있는 리스트에서 개만 삭제) 리스트  
    const exceptSecondHighest = exceptHighest.filter(  
        (point) => point.totalScore !== secondMaxTotalScore  
    );  
    // 그중 큰값 뽑기  
    const thridMaxTotalScore = Math.max(  
        ...exceptSecondHighest.map((exception) => exception.totalScore)  
    );  
    const thridHighestScoreCate = points.find(  
        (point) => point.totalScore === thridMaxTotalScore
```

1. `secondMaxTotalScore !== 0`

=> 기존 리스트를 비운후, 같은 로직을 반복하여 두 번째로 높은 카테고리 요소를 가져옴

=> 세 번째로 높은 카테고리 요소를 찾기 위해 첫번째 요소가 제거된 `exceptHighest` 리스트에서 두번째 요소도 제거

1. 최종적으로 세번째 요소까지 가져옴

광고

2-1. 로직 구현 : 로그인 유저

```
// 포인트 가장 높음
const adsCate01 = ads
  .filter(
    (ad) =>
      ad.animalCategory.animalCategoryCode ===
        highestScoreCate.animalCategory.animalCategoryCode
  )
  .filter((ad) => ad.productBoardGrade >= 3.5)
  .slice(0, 7);
// 포인트 두번째로 높음
console.log(secondHighestScoreCate);
const adsCate02 = ads
  .filter(
    (ad) =>
      ad.animalCategory.animalCategoryCode ===
        secondHighestScoreCate.animalCategory.animalCategoryCode
  )
  .filter((ad) => ad.productBoardGrade >= 3.5)
  .slice(0, 5);
// // 포인트 세 번째로 높음
const adsCate03 = ads
  .filter(
    (ad) =>
      ad.animalCategory.animalCategoryCode ===
        thridHighestScoreCate.animalCategory.animalCategoryCode
  )
  .filter((ad) => ad.productBoardGrade >= 3.5)
  .slice(0, 3);
```

1. 가장 total_score 높은 카테고리 순

a. adsCate01

=> 7개

a. adsCate02

=> 5개

a. adsCate03

=> 3개

+ filter (평점이 3.5이상)

광고

2-1. 로직 구현 : 로그인 유저

```
// 새롭게 뽑힌 배열
const filteredArr = [...adsCate01, ...adsCate02, ...adsCate03];

const shuffleArray = (array) => {
    return array.sort(() => Math.random() - 0.5);
};

const randomArr = shuffleArray(filteredArr);
console.log(randomArr);
setNewAds(randomArr);
```

```
{Object.keys(user).length !== 0 ? (
    <>
    {newAds.map((ad, index) => (
        <AdCard adDetail={ad} key={index} />
    )))
    </>
) : (
    <>
    {randomAds.map((ad, index) => (
        <AdCard adDetail={ad} key={index} />
    )))
    </>
)}
```

1. filteredArr

=> 필터 처리된 3개의 리스트를 최종 리스트에 담아줌

1. 내부 배열 요소를 랜덤하게 섞은 filteredArr을 randomArr에 담아줌

1. setNewAds(randomArr)에 담은 후 return

Object.keys(user).length !==0,

=> 유저 반환값이 없으면 randomAdd 리스트 반환

=> 있으면 newAds 리스트 반환

광고

2-2. 로직포인트 구현 : 로그인 유저만

```
const pointFilter = async () => {
    // console.log(detailInfo);
    // 증가시킬 것
    const target = points.filter(
        (point) =>
            point.animalCategory.animalCategoryCode ===
            detailInfo.animalCategory.animalCategoryCode
    );
    console.log(target);
    await fluctuationByDetailP(target);
}
```

```
// 특정 글에 들어갔을때 받는 값 증가 : animalCategoryCode userId
export const fluctuationByDetailP = async (target) => {
    // 배열에 감싸져있기에 첫번째 배열에서 꺼내오기
    if (target.length !== 0) {
        let x_value = target[0].inputValue + hitPoint;
        const response = reciprocalFunction(x_value),
        let data = { target: target[0], response: response, inputValue: x_value }
        console.log(data);
        await instance.put(`logic-point/positive`, data);
    }
}
```

1. logic total_score를 기준
=> 로그인 유저한테만 해당
1. animalBoard 글 열람시 해당
하는 카테고리에 대한 포인트
증가

광고 2-2. 로직포인트 구현 : 로그인 유저만

```
// 감소시킬것
console.log(detailInfo.animalCategory.animalCategoryCode);
const exceptionList = points.filter(
  (point) =>
    point.animalCategory.animalCategoryCode !==
      detailInfo.animalCategory.animalCategoryCode
);
console.log(exceptionList);
// 가장 높은 totalScore 값을 가진 요소를 찾기
const maxTotalScore = Math.max(
  ...exceptionList.map((point) => point.totalScore)
);
console.log(maxTotalScore); // 값이 여러개?
if (maxTotalScore >= 0.05) {
  // 가장 높은 totalScore 값을 가진 요소를 가져오기
  const maxScoreInException = exceptionList.find(
    (point) => point.totalScore === maxTotalScore
);
  console.log(maxScoreInException);
  // setException(maxScoreInException);
  await fluctuationByDetailM(maxScoreInException);
```

```
// 특정 글에 들어갔을때 받는 값 감소 : animalCategoryCode userId
export const fluctuationByDetailM = async (exception) => {
  if (exception.length !== 0) {
    console.log(exception);
    let x_value = exception.inputValue + hitLostPoint;
    const response = reciprocalFunction(x_value),
      let data = {
        exception: exception,
        response: response,
        inputValue: x_value,
      };
    await instance.put("logic-point/negative", data);
  }
},
```

감소의 경우, 해당 유저의 카테고리 total_score가 현재 열람중인 글의 카테고리 제외 가장 높은 카테고리 요소에 대한 포인트 차감



```
1 const reciprocalFunction = (x) => {  
2     return 100 / (1 + Math.exp(-0.1 * (x - 50)));  
3 };
```

마이페이지

마이페이지와 useLocation



```
1 // 페이지 경로에서 정보 따오기
2 const location = useLocation();
3 const nowLoca = location.pathname.substring(17);
```



북마크한 제품 목록

| 동물 카테고리 | 글 제목 | 상품명 | 가격 | 작성일 |
|---------|-------------------------|--------------------|---------|------------|
| 비둘기 | 애완 비둘기에 대한 빠악빠악 게이지 | 빠악빠악 게이지 | 60000 | 2024-05-20 |
| 고양이 | 고양이 승승장 | 승승장 | 300000 | 2024-05-20 |
| 고양이 | 캣타워 | 민농 캣타워 | 400000 | 2024-05-20 |
| 고양이 | 백백 스크래처 | 이용 백백 스크래처 | 20000 | 2024-05-20 |
| 기타 | 소니 경박단소 총액조 | 소니 알파 24-50mm G 렌즈 | 1799000 | 2024-05-20 |
| 고양이 | 카에라 | a7c2 | 2390000 | 2024-05-19 |
| 비둘기 | 암조용으로 추천. 파나소닉 루믹스 G9M2 | 루믹스 G9M2 | 2090000 | 2024-05-19 |

◀ ▶ 1

```
1 const MyPageTab = (onClickMenu) => {
2   const [menu1, setMenu1] = useState("");
3   const [menu2, setMenu2] = useState("");
4   const [menu3, setMenu3] = useState("");
5   const [menu4, setMenu4] = useState("");
6   const [menu5, setMenu5] = useState("");
7   const [menu6, setMenu6] = useState("");
8   const [menu7, setMenu7] = useState("");
9   const [menu8, setMenu8] = useState("");
10  const [menu9, setMenu9] = useState("");
11
12  const nowMenu = Object.values(onClickMenu).toString();
13
14  const setTabCss = () => {
15    if (nowMenu == "myadoption") {
16      setMenu1("selectedMenu");
17    } else if (nowMenu == "mylost") {
18      setMenu2("selectedMenu");
19    } else if (nowMenu == "myminimalfav") {
20      setMenu3("selectedMenu");
21    } else if (nowMenu == "myproductfav") {
22      setMenu4("selectedMenu");
23    } else if (nowMenu == "myneighbor") {
24      setMenu5("selectedMenu");
25    } else if (nowMenu == "myuserqna") {
26      setMenu6("selectedMenu");
27    } else if (nowMenu == "mysitter") {
28      setMenu7("selectedMenu");
29    } else if (nowMenu == "myonedayclass") {
30      setMenu8("selectedMenu");
31    } else if (nowMenu == "myqa") {
32      setMenu9("selectedMenu");
33    }
34  };
35}
```

쪽지

전체 쪽지함, 받은 쪽지함, 보낸 쪽지함, 중요 쪽지함

컴포넌트로 분리하여
사용

전체

받은

보낸

중요

```

1 const NoteHeaderTap = () => {
2   const location = useLocation();
3   const [nowLoca, setNowLoca] = useState("");
4   useEffect(() => {
5     setNowLoca(location.pathname);
6   }, [location]);
7
8   const [click, setClick] = useState("");
9   const navigate = useNavigate();
10  const noteNevi = (data) => {
11    setClick(data);
12    if (data == "all") {
13      navigate("/compagno/mypage/mynote");
14    }
15    if (data == "receive") {
16      navigate("/compagno/mypage/mynote/viewReceiveBox");
17    }
18    if (data == "send") {
19      navigate("/compagno/mypage/mynote/viewSendBox");
20    }
21    if (data == "star") {
22      navigate("/compagno/mypage/mynote/viewStar");
23    }
24  };

```

```

29   return (
30     <Div>
31       <div className="noteHeader">
32         <button
33           id="viewAll"
34           className={`${nowLoca === "/compagno/mypage/mynote" ? "select" : ""}`}
35           onClick={() => noteNevi("all")}
36         >
37           <BsEnvelopePaper
38             style={({ marginRight: "15px", fontSize: "1.3rem" })} />
39           전체 쪽지함
40         </button>
41
42         <button
43           id="receive"
44           className={`${nowLoca === "/compagno/mypage/mynote/viewReceiveBox" ? "select" : ""}`}
45           onClick={() => noteNevi("receive")}
46         >
47           <RiFolderReceivedLine
48             style={({ marginRight: "15px", fontSize: "1.3rem" })} />
49           받은 쪽지함
50         </button>
51
52         <button
53           id="send"
54           className={`${nowLoca === "/compagno/mypage/mynote/viewSendBox" ? "select" : ""}`}
55           onClick={() => noteNevi("send")}
56         >
57           <RiFolderSharedLine
58             style={({ marginRight: "15px", fontSize: "1.3rem" })} />
59           보낸 쪽지함
60         </button>
61
62         <button
63           id="star"
64           className={`${nowLoca === "/compagno/mypage/mynote/viewStar" ? "select" : ""}`}
65           onClick={() => noteNevi("star")}
66         >
67           <FaStar style={({ marginRight: "15px", fontSize: "1.3rem" })} />
68           중요 쪽지함
69         </button>
70       </div>
71     </div>
72   );
73 
```

쪽지

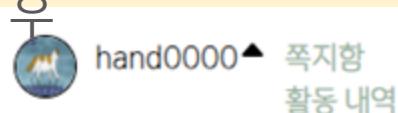
로그인 유저 / 다른 유저

<MyToggleButton name={comment.user.userNickname} />



hand0000▼

본인 계정 선택 경



hand0000▲ 쪽지함
활동 내역

다른 유저 선택 경



유저01▲ 쪽지 보내기

댓글

```
1 const MyToggleButton = (props) => {
2   const [isOpen, setIsOpen] = useState(false);
3   const handleToggle = () => {
4     setIsOpen(!isOpen);
5   };
6
7   // 유저정보 가지고온다
8   const dispatch = useDispatch();
9   const user = useSelector((state) => {
10     return state.user;
11   });
12   useEffect(() => {
13     const token = localStorage.getItem("token");
14     if (token !== null) {
15       dispatch(userSave(JSON.parse(localStorage.getItem("user"))));
16     }
17   }, []);
18
19   const [modalIsOpen, setModalIsOpen] = useState(false);
20   const navigate = useNavigate();
21   const sendNote = () => {
22     if (Object.keys(user).length === 0) {
23       navigate("/compagno/login");
24     } else {
25       setModalIsOpen(!modalIsOpen);
26     }
27   };
}
```

```
10   <div>
11     <div style={{ display: "flex" }>
12       {props.name === "" || user.userNickname === props.name ? (
13         <>(user.userNickname)</>
14       ) : (
15         <>(props.name)</>
16       )}
17
18       {isOpen ? (
19         <BsCaretnUpFill onClick={handleToggle}>
20       ) : (
21         <BsCaretnDownFill onClick={handleToggle}>
22       )}
23     </div>
24   </div>
25   <div id="noteToggle" style={{
26     display: isOpen ? "flex" : "none",
27   }}>
28     {props.name === "" || user.userNickname === props.name ? (
29       <a href="#">
30         <img alt="쪽지함 icon" style={{ color: "#99A9B9", fontWeight: "bold" }}>
31       </a>
32       <a href="#">
33         <img alt="활동 내역 icon" style={{ color: "#99A9B9", fontWeight: "bold" }}>
34       </a>
35     ) : (
36       <button onClick={sendNote}>
37         <img alt="쪽지 보내기 icon" style={{ border: "none", borderRadius: "10px", color: "black", fontWeight: "bold", backgroundColor: "#99A9B9", fontSize: "0.7rem" }}>
38       </button>
39     )}
40   </div>
41   <ModalContainer>
42     {modalIsOpen ? (
43       <ModalContent>
44         <div>
45           <input type="text" value={nickName} onChange={(e) => setNickName(e.target.value)} />
46           <button onClick={onRequestClose}>닫기</button>
47         </div>
48         <NoteCreate nickName={nickName} />
49       </ModalContent>
50     ) : null}
51   </ModalContainer>
52 </div>
53 </div>
54 </div>
55 </div>
56 </div>
57 </div>
58 </div>
59 </div>
60 </div>
61 </div>
62 </div>
63 </div>
64 </div>
65 </div>
66 </div>
67 </div>
68 </div>
69 </div>
70 </div>
71 </div>
72 </div>
73 </div>
74 </div>
75 </div>
76 </div>
77 </div>
78 </div>
79 </div>
80 </div>
81 </div>
82 </div>
83 </div>
84 </div>
85 </div>
86 </div>
87 </div>
88 </div>
89 </div>
90 </div>
91 </div>
92 </div>
93 </div>
94 </div>
95 </div>
```

쪽지 쪽지함 / 쪽지 보내기

전체쪽지 받은쪽지 보낸쪽지 ★ 중요쪽지

보내는 사람 받는 사람
제목 날짜

총 22개

| 번호 | 보내는 사람 | 제목 | 내용 | 받는 사람 | 날짜 | 첨부파일수 |
|----|----------|---------|--------------|----------|----------------|-------|
| ☆ | hand0000 | 8888 | 88888 | 우재02 | 24-05-19 03:41 | |
| ☆ | hand0000 | 88888 | | 성봉02 | 24-05-19 02:02 | |
| ☆ | hand0000 | 888888 | 8888888 | 성봉02 | 24-05-18 06:51 | |
| ★ | hand0000 | 봉이 사진 풍 | 보내자고 | 우재02 | 24-05-17 07:20 | 0 |
| ☆ | hand0000 | 화나온 | 도연재7 | 우재02 | 24-05-17 07:18 | |
| ☆ | hand0000 | 88888 | 888888888888 | 우재02 | 24-05-17 07:17 | 0 |
| ☆ | hand0000 | 88888 | | 우재02 | 24-05-17 07:16 | 0 |
| ☆ | 우재02 | 밀당드려용~ | 밀당밀당 | hand0000 | 24-05-17 06:53 | |
| ☆ | hand0000 | 위 | 사진 | 우재02 | 24-05-17 06:52 | |
| ☆ | hand0000 | 언성 | 언성언성언성 | 우재02 | 24-05-17 06:52 | |

« « 1 2 3 » »

자정 임 처 정보

쪽지 보내기

보내는 사람 hand0000 받는 사람 성봉02

제목
제목을 입력해주세요

내용
내용을 입력해주세요

파일 선택 선택된 파일 없음

쪽지 검색, 페이지

```
// viewAll(쪽지 목록) - 쪽지 검색
@GetMapping("/note/viewAll/{nickName}")
public ResponseEntity<Page<Note>> viewAll(@PathVariable(name="nickName")String nickName, @RequestParam(name="page", defaultValue = "1") int page, @RequestParam(name="sender", required = false)String sender, @RequestParam(name="receiver", required = false) String receiver, @RequestParam(name="noteTitle", required = false)String noteTitle, @RequestParam(name="noteRegiDate", required = false) String noteRegiDate){

    Sort sort = Sort.by("noteCode").descending();
    Pageable pageable = PageRequest.of(page-1, 10, sort);
    QNote qNote = QNote.note;
    BooleanBuilder builder = new BooleanBuilder();
    BooleanExpression expression = null;
    if(nickName!=null){
        builder.or(qNote.sender.eq(nickName));
        builder.or(qNote.receiver.eq(nickName));
    }

    if(sender!=null){
        expression = qNote.sender.contains(sender);
        builder.and(expression);
    }

    if(receiver!=null){
        expression = qNote.receiver.contains(receiver);
        builder.and(expression);
    }
    if(noteTitle!=null){
        expression = qNote.noteTitle.contains(noteTitle);
        builder.and(expression);
    }
    if(noteRegiDate!=""){
        String start = noteRegiDate+" 00:00:00";
        String end = noteRegiDate+" 23:59:59";

        expression = qNote.noteRegiDate.between(Timestamp.valueOf(start), Timestamp.valueOf(end));
        builder.and(expression);
    }

    return ResponseEntity.ok(service.viewAll(pageable, builder));
}
}
```

[controller]

@RequestParam으로 검색어 받기
-> 조건 충족 시 builder 처리

모든 쪽지함

-> sender나 receiver가 해당 닉네임일 경우

보낸 쪽지함

-> sender가 해당 닉네임일 경우

받은 쪽지함

-> receiver가 해당 닉네임일 경우

쪽지

삭제 로직 분류 및 쪽지함 별 쪽지수 확인 분리

```

@Column(name="deleted_by_sender")
private Boolean deletedBySender; // 기본 false

@Column(name="deleted_by_receiver")
private Boolean deletedByReceiver;

public Note updateDeleteSender(int noteCode) {
    if(noteDAO.existsById(noteCode)){
        Note vo = noteDAO.findById(noteCode).orElse(null);
        vo.setDeletedBySender(true);
        return noteDAO.save(vo);
    }
    return null;
}

public Note updateDeleteReceiver(int noteCode){
    if(noteDAO.existsById(noteCode)){
        Note vo = noteDAO.findById(noteCode).orElse(null);
        vo.setDeletedByReceiver(true);
        return noteDAO.save(vo);
    }
    return null;
}

public void delete(int noteCode){
    Note vo = noteDAO.findById(noteCode).orElse(null);
    if(vo!=null){
        noteDAO.delete(vo);
    }
}

```

service

받는 이/보낸 이의 경우 삭제 로직 분리

-> 삭제 클릭 시 삭제 컬럼 update
삭제 컬럼 둘 다 true 일 경우 : DB 삐

```

@DeleteMapping("/note/sender/{noteCode}")
public ResponseEntity<Note> deletedSender(@PathVariable(name="noteCode")int noteCode){

    Note note = service.updateDeleteSender(noteCode);

    if(note.getDeletedBySender() && note.getDeletedByReceiver()){
        service.delete(noteCode);
    }

    return (note!=null)?
        ResponseEntity.status(HttpStatus.ACCEPTED).body(note):
        ResponseEntity.status(HttpStatus.BAD_REQUEST).build();
}

@DeleteMapping("/note/receiver/{noteCode}")
public ResponseEntity<Note> deletedReceiver(@PathVariable(name="noteCode")int noteCode){

    Note note = service.updateDeleteReceiver(noteCode);

    if(note.getDeletedBySender() && note.getDeletedByReceiver()){
        service.delete(noteCode);
    }

    return (note!=null)?
        ResponseEntity.status(HttpStatus.ACCEPTED).body(note):
        ResponseEntity.status(HttpStatus.BAD_REQUEST).build();
}

```

controller

쪽지 중요 쪽지 기능

받는 이/보낸 이의 경우 로직 분리
-> 각자 중요 체크 따로 관리
DAO에 QUERY문으로 컬럼 boolean값 수정

```
@Column(name="star_sender") VO
private Boolean starSender;

@Column(name="star_receiver")
private Boolean starReceiver;
```

service

```
public void updateStarSender(int noteCode){noteDAO.updateStarSender(noteCode);}
public void updateStarReceiver(int noteCode){noteDAO.updateStarReceiver(noteCode);}
```

```
@Modifying
@Transactional
@Query(value="UPDATE note SET star_sender=(!star_sender) WHERE note_code =:noteCode",nativeQuery = true )
void updateStarSender(@Param("noteCode")int noteCode);
```

dao - Query

```
@Modifying
@Transactional
@Query(value="UPDATE note SET star_receiver=(!star_receiver) WHERE note_code =:noteCode",nativeQuery = true )
void updateStarReceiver(@Param("noteCode")int noteCode);
```

```
@PutMapping("/note/starSender")
public ResponseEntity<Note> updateStarSender(@RequestParam(name="noteCode")int noteCode){
    service.updateStarSender(noteCode);
    return ResponseEntity.status(HttpStatus.OK).build();
}
@PutMapping("/note/starReceiver")
public ResponseEntity<Note> updateStarReceiver( @RequestParam(name="noteCode")int noteCode ){
    service.updateStarReceiver(noteCode);
    return ResponseEntity.status(HttpStatus.OK).build();
}
```

controller

쪽지 답장



```

{note.sender == user.userNickname ? (
    <></>
) : (
    <>
    <button
        style={{
            border: "none",
            borderRadius: "10px",
            margin: "0px 10px",
            width: "50px",
            fontWeight: "bold",
            backgroundColor: "#CBD9CE",
        }}
        onClick={sendNote}
    >
        답장
    </button>
    </>
)}
```

10

Project Review

- 각자 담당 페이지를 정해서 개별적으로 진행하다 보니 팀으로 함께 진행한 작업이 부족했던 점과 테스트를 더 해 볼 시간이 부족했던 점이 아쉽습니다.
 - 여러 팀원들과 프로젝트를 진행하면서 비슷하거나 같은 기능도 다양한 방법으로 구현한 점이 흥미로웠고, 제가 구현한 방법이 아닌 다른 방법을 학습할 수 있었습니다.
 - 세미 프로젝트에서 게시판을 구현해보지 못해서 아쉬웠는데, 이번 프로젝트를 통해서 게시판을 만들어 볼 수 있어서 좋았습니다.
 - 개인적인 일로 처음에 계획했던 일정으로 마무리하기가 어려워져서 기획했던 기능들을 다 구현해보지 못해서 많이 아쉽습니다.
-
- 컴포넌트를 분리하지 못해서 중복되는 코드가 있었고 유지보수 시에도 가독성이 떨어졌습니다.
 - 오류 개선 시 로직 마지막 부분에서만 찾으려고 해서 더 오래 걸렸던 것 같습니다. 혼자서 오류를 개선하는 연습의 필요성을 느꼈습니다.

10

Project Review

- 각자 작업에 집중하다 보니 피드백을 주고 받는 시간이 부족했던 것 같습니다. 각자의 진행 상황이나 다른 사람의 관점에서 테스트를 진행하다가 발견할 수 있었을 오류들을 공유하면서 조금 더 수월하게 프로젝트 마무리를 할 수 있지 않았을까 싶습니다.
- 기능 개발 중 막히는 경우, 그 문제에 너무 오래 매달리기보다는 다른 부분의 작업을 진행하면서 해결 방법을 생각하는 습관을 들여야겠다고 생각했습니다.
- 맡은 게시판의 댓글 부분에서는 유저가 변경한 프로필을 그대로 반영하지만, 게시글 부분에서는 반영하지 못하는 점을 수정하지 못한 것이 아쉽습니다. 조금 더 다양한 방식으로 테스트를 해보면서 인지하지 못한 오류를 발견할 수 있었다면 좋았을 것 같아 아쉽습니다.
- 다양한 사람들과 함께 작업을 진행하면서 같은 기능을 구현하면서도 이렇게 다양한 방식으로 접근할 수 있음을 알게 되었고, 이를 통해 다양한 접근 방식을 배울 수 있었습니다.
- TIMESTAMP를 사용하는 과정에서 발생된 문제들을 구멍난 둑을 막듯이 해결하면서 훗날 유지보수에 신경을 제대로 쓰지 못한 것 같아서 아쉽습니다.

1. 프로젝트 기간동안 메인페이지 부분이랑 기능관련 부분을 잘 분배해서 작업을 했어야하는데 막히는 부분에서 시간을 잡다 보니깐 처음 기능구현 관련 목표에 도달하지 못한 것 같아서 아쉬웠다.
1. UI관련 디자인을 생각하는 시간과 수정을 반복하다 보니 기능구현에 잡아야 될 시간들을 다 잡아 아쉬움이 남았다.
1. 수업과 프로젝트를 같이 하는 시간일때 기능을 먼저 일부 잡아두고 갔었다면 프런트를 다잡고 다시 백을 잡았을때 기억이 드물게 생각나서 아쉬웠다.
1. 7명의 팀원들 각자의 분량이 많다 보니깐 진행상황에 대한 회의를 자주 하지 못한 것 같아서 슬펐다 ㅠㅠ
1. 부족한 부분을 채워 기간동안의 시간을 잘 분배하여 막힘이 있어도 잘 넘어갈 수 있도록 해야겠다.

10

Project Review

1. 세미 프로젝트 때보다 다인원으로 구성된 팀이라 조금 더 규모 있는 프로젝트를 진행할 수 있어 좋았습니다. 또한, 더욱 다양한 코드를 접할 수 있어 좋았고 세미 프로젝트 때는 게시판 위주로 진행하였는데 이번에는 '쪽지' 기능을 추가로 구현할 수 있어 좋았고 다음에는 초기 목표였던 '채팅'을 구현해보고 싶다 생각 하였습니다.
-
1. [프로젝트] 각자가 맡은 업무를 하는데 너무 집중되어서 서로의 결과물을 보고 발생 할 수 있는 오류들을 확인 하는 것이 늦어졌던 점이 아쉬웠다.
 1. [게시판] 유저 프로필 이미지 변경 시, 기존 작성되었던 게시물과 댓글 유저 이미지가 자동 변경되지 못한 점이 아쉬웠다.
 1. [쪽지] 검색 필터를 거쳐 출력되는 쪽지 갯수 구현을 진행하지 못한 점이 아쉬웠다.
 1. [쪽지] 한쪽에서만 삭제된 쪽지가 페이징 처리에서 제외되도록 해야하는데 이를 해결하지 못한 점이 아쉬웠다.

10

Project Review

1. 현대의 디지털 환경에서 개인 정보는 정말 중요한 자산이다. 이 자산을 암호화를 통해 지키는 것이 얼마나 중요한 것인지 다시 실감하게 되었다.
1. 개발을 진행하며 가장 크게 느낀 것은 나의 개발 편의성만을 생각하고 진행하면 속도는 빠르게 진행할 수 있지만, 정작 애써 적용한 Security가 무색하게 개인 정보가 유출되는 모습을 보고 보안 사고가 왜 발생하는지 알게 되었다는 점이다.
이 프로젝트를 통해 보안에 대한 인식이 더욱 강화되었고, 앞으로의 개발도 이런 기본적인 책임을 명심하며 진행해야겠다고 생각했다.

10

Project Review

1. 실제로 사용했던 사이트들의 자잘한 문제점들이 생긴 이유, 그리고 그것을 고치지 않고 방치할 수 밖에 없었던 그들의 노고를 간접적으로 이해할수있었던 시간이었습니다.
1. 기능개발에 경중을 조금더 자세히 두어서 진행하지 못했던 점에 아쉬움을 느꼈습니다. 대표적으로 글 ↪ 정시에 대한 이미지처리를 우선적으로 하지 않았던 점. 부분적으로 자잘한 css나 디테일적인 기능 마감, 랭킹 선정기능에 기간 계산이 들어감에도 db에서 서버로 가져오는 시간대에 일정하지 않았다는점을 늦게 인지했다는 점 등, 여러가지로 아쉬운 점이 있었습니다.
1. 추가로, 컴포넌트를 너무 과도하게 분리해도 진행도에 도움이 되지 않는다는점!!, 분리는 적재적소 필요 할때만 해야한다는것을 뼈저리게 느꼈습니다.

10

Project Review

1. 여러 사람들과 프로젝트를 진행하면서 다양한 코드를 보게 되었고, 이를 학습하게 되는 좋은 경험이 되었던 것 같습니다.
1. 당시에는 충분하다 생각되었지만 프로젝트를 마치고보니 서로의 코드 리뷰, 회의 같은 피드백을 주고받는 시간이 부족해 서로의 정보 공유가 잘 되지 않은 것 같아 아쉬웠습니다.
1. React를 사용하게 되면서 컴포넌트를 분리하지 않고 진행하다 보니 코드가 너무 길어져 가독성이 떨어지고 재사용, 유지보수가 힘들다는 것을 느끼게 되었습니다. 이를 통해 컴포넌트 분리의 중요성을 느끼게 되었습니다.