APTOGETHER

아파트 통합관리 웹 솔루션

INDEX

APTOGETHER

아파트 통합관리 웹 솔루션

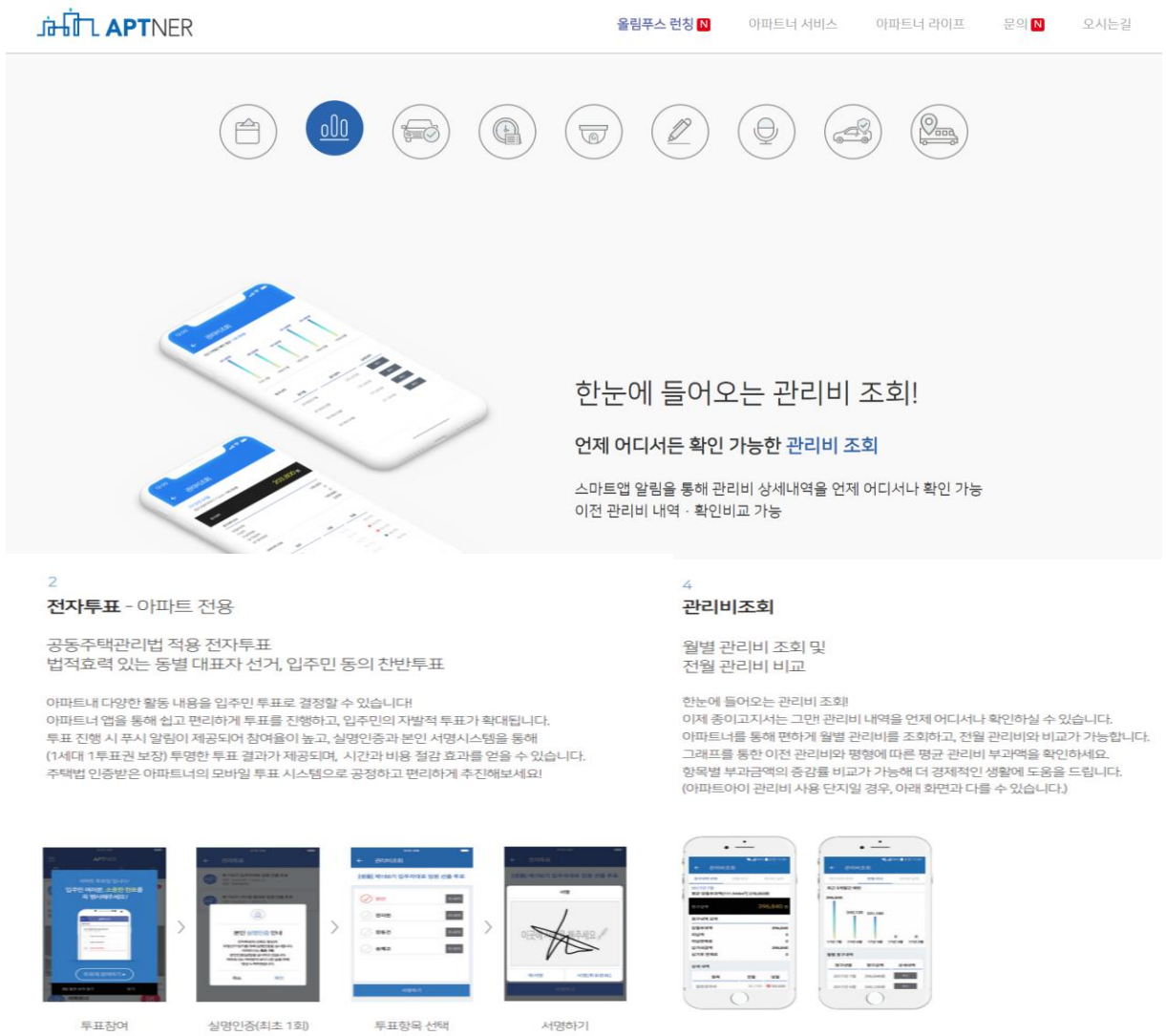
01. 주제선정

주제선정 – 배경 및 목적

기존 프로그램의 기능을 확장하고 새로운 기능을 추가함으로써
입주민의 편의를 제공

1 투명한 관리비 운영 **2** 편리한 정보 확인 **3** 스마트한 아파트 운영

벤치마킹



아파트너 : 모바일을 중심으로 한 통합형 아파트 생활 포털 플랫폼

관리비 조회(ex 전월 관리비 비교)

전자투표(ex 전자서명)

민원 등의 다양한 기능을 벤치마킹

APTOGETHER

아파트 통합관리 웹 솔루션

02. 개발환경

개발환경 – 개발환경 / 형상관리

JDK 1.8

HTML / CSS / JavaScript

Apache Tomcat 8.0

Bootstrap

Oracle Database 11g

SQL Developer

Spring Framework

Junit 4

Spring Tools Suite 3
/ Eclipse

GitHub / Git

개발환경 – 라이브러리 / API

JQuery

Signature Pad.js

DataTable

Chart.js

Full Calandar

moment.js

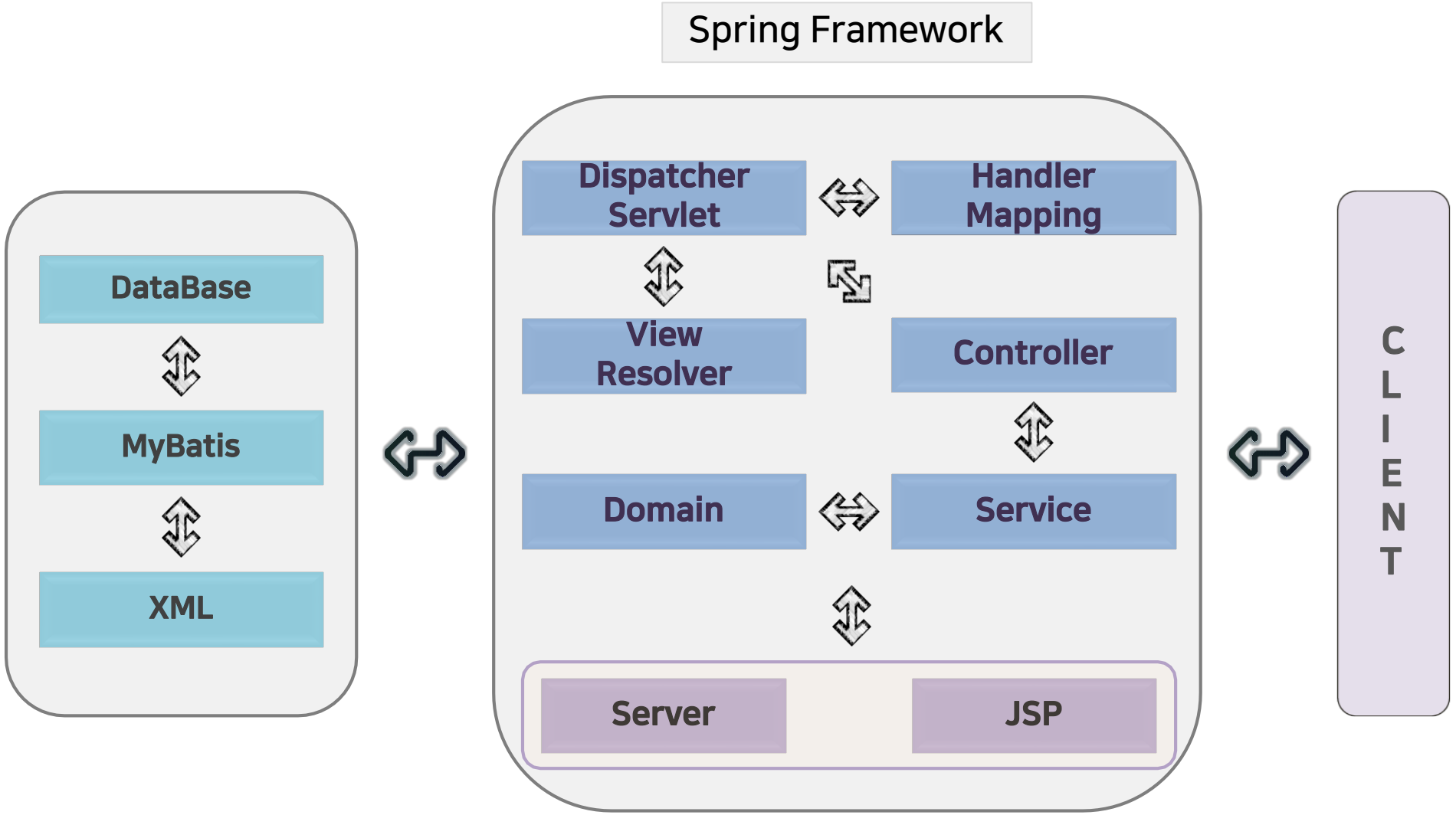
KAKAO MAP /
카카오 주소검색 / LOCAL

공공 데이터 포털 API

개발 일정

Task		기간	소요일정
주제선정	주제선정	04.02~04.07	6
	핵심 기능 설정	04.08~04.10	3
요구사항	조사	04.11~04.14	4
	취합 및 조율	04.15~04.16	2
	확정	04.17~04.18	2
	요구분석 정의서	04.19~04.22	4
	Back log작성	04.23~04.24	2
설계	비즈니스 로직 설정	04.25~04.29	5
	유스케이스	04.30~05.01	3
	ERD 작성	05.02~05.05	4
	DB 설계	05.06~05.12	7
	클래스다이어그램	05.13~05.14	2
	화면 설계도	05.15~05.18	4
환경설정		05.19~05.19	1
기능구현		05.20~06.09	21
Spring project 변환		06.10~06.16	7
Test		06.17~06.17	1

아키텍처 구조


Spring Framework
DataBase
MyBatis
XML
Dispatcher Servlet
Handler Mapping
View Resolver
Controller
Domain
Service
Server
JSP
CLIENT

프로젝트 구조

- Aptogether [kosta-197-project-spring HJ/Schedule]
  - src/main/java
    - org.aptogether.common
      - AuthInterceptor
    - org.aptogether.controller
      - AptRestController
      - FeeController
      - FeeRestController
      - 그 외 [기능]Rest/controller로 구성
    - org.aptogether.domain
      - HouseholdVO
      - LevyVO
      - NoticePageDTO
      - 그 외 [기능]VO,DTO,Criteria로 구성
    - org.aptogether.mapper
      - ScheduleMapper
      - AptMapper
      - 그 외 [기능]Mapper로 구성…
    - org.aptogether.security
      - ApiKeys
      - CustomLoginSuccessHandler
    - org.aptogether.service
      - LevyService
      - LevyServiceImpl
      - 그 외 [기능]service/serviceImpl로 구성
    - src/main/resources
      - org
        - aptogether
          - mapper
            - AptMapper.xml
            - 그 외 [기능]Mapper.xml로 구성
  - src/test/java
    - org.aptogether.controller
      - NoticeControllerTests
      - 그 외 [기능]ControllerTests로 구성
    - org.aptogether.mapper
      - LevyMapperTests
      - 그 외 [기능]MapperTests로 구성
    - org.aptogether.persistence
      - DataSourceTests
      - JDBCTests
    - org.aptogether.service
      - PollServcieTests
      - 그 외 [기능]ServiceTests로 구성
    - src
      - main
        - webapp
          - resources
            - css
              - jquery.dataTables.min.css
              - fullcalendar.min.css
              - sb-admin-2.css
              - 각 api, bootstrap.css로 구성
            - img
              - sort_asc.png
              - 각 css에 맞는 이미지 파일로 구성
            - js
              - demo
                - chart-area-demo.js
                - Chart.js에 필요한 js로 구성
    - fullcalendar
      - addEvent.js
      - addEventKeeper.js
      - Keeper와 입주민의 js를 분리하여 구성
    - poll.js
    - 각 [기능].js로 구성…
    - vendor
      - bootstrap
      - chart.js
      - datatables
      - fontawesome-free
      - jquery
      - jquery-easing
      - Bootstrap, api에 필요한 js,html로 구성
    - WEB-INF
      - spring
        - appServlet
          - servlet-context.xml
        - root-context.xml
        - security-context.xml
      - views
        - keeperSignin.jsp
        - keeperSignup.jsp
        - userDashBoard.jsp
        - 각 화면에 필요한 .jsp로 구성

APTOGETHER

아파트 통합관리 웹 솔루션

03. 설계

명명법

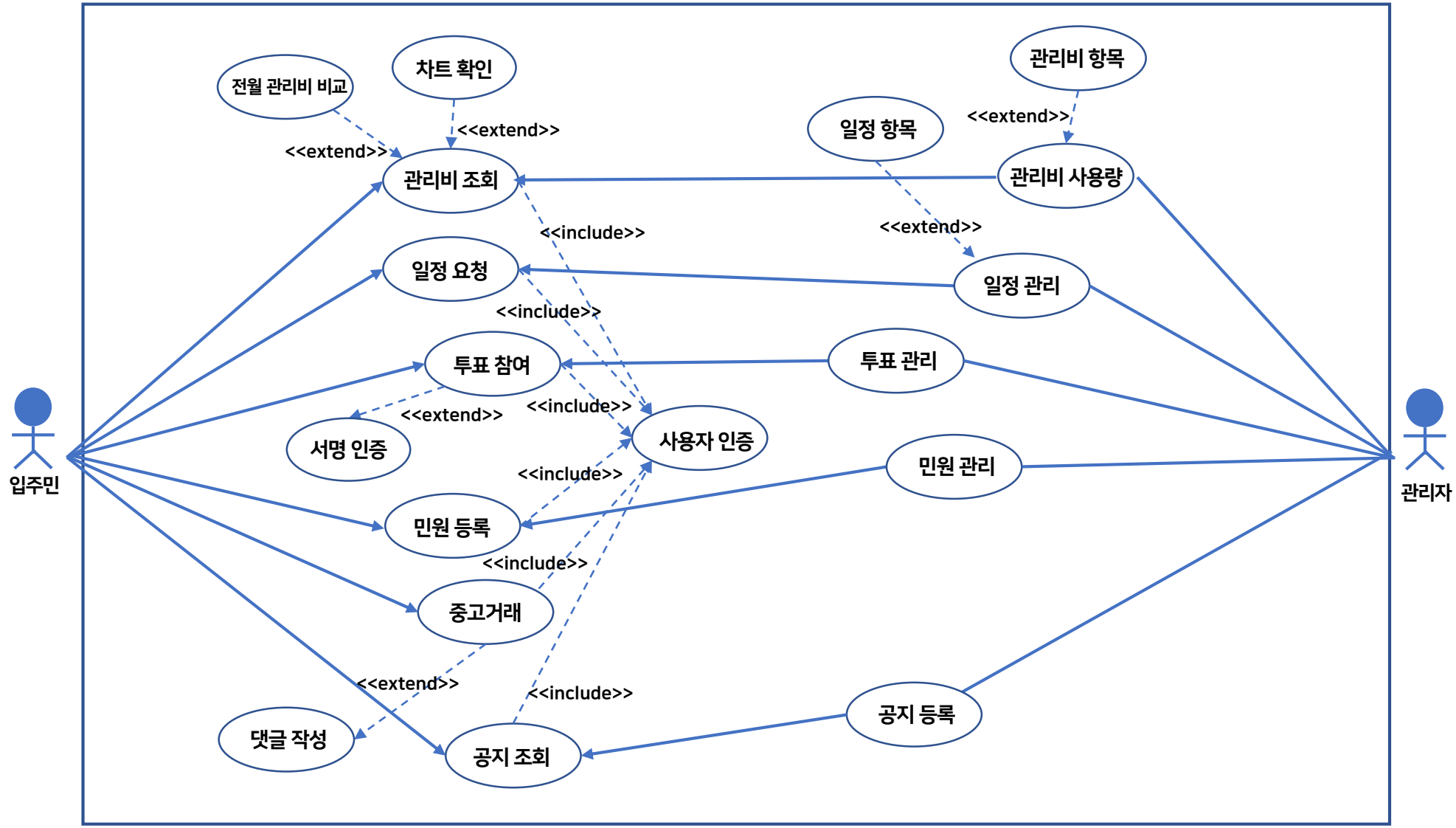
명명법

Column		Data Type	Key
Login	controller	org.aptogether.controller	- KeeperController
Fee			- TenantController
Schedule			- RestController
Poll	mapper	org.aptogether.mapper	- Mapper
Complain Board	domain	org.aptogether.domain	- name+VO
Market	service	org.aptogether.service	- Service
			-ServiceImpl

URL

Data Type	Key
Keeper Controller	/keeper/[모듈명]list
	/keeper/[모듈명]update
	/keeper/[모듈명]delete
	/keeper/[모듈명]get
	/keeper/[모듈명]create
	/keeper/[모듈명]기능명
Tenant Controller	/tenant/[모듈명]list
	/tenant/[모듈명]update
	/tenant/[모듈명]delete
	/tenant/[모듈명]get
	/tenant/[모듈명]create
	/tenant/[모듈명]기능명

유스케이스


입주민
관리자
전월 관리비 비교
차트 확인
관리비 항목
일정 항목
<<extend>>
<<extend>>
<<extend>>
관리비 조회
관리비 사용량
<<include>>
<<extend>>
일정 요청
일정 관리
<<include>>
투표 참여
투표 관리
<<extend>>
<<include>>
서명 인증
사용자 인증
민원 관리
<<include>>
민원 등록
<<include>>
중고거래
<<extend>>
<<include>>
댓글 작성
공지 조회
공지 등록

요구분석 정의서

순번	구분	요구사항
1	관리비	관리사무소는 관리비의 부과일자, 산출기간, 납부마감일, 부과작업상태에 대한 정보를 등록한다.
2		관리사무소는 세대별로 공동관리비 및 개별관리비를 산정 후 내역을 등록한다.
3		관리사무소는 부과한 관리비 내역에 대한 목록을 동,호수별로 조회한다.
4		관리사무소는 부과한 관리비 내역을 수정할 수 있다.
5		입주민이 자신의 동, 호수에 고지된 관리비 내역을 확인할 수 있다.
6		입주민은 전월과 당월에 대한 관리비 증감 정보가 표시를 확인할 수 있다.
7		입주민은 최근 6개월 간 고지된 관리비 납부내역을 차트(Line, Pie)로 조회할 수 있다.
8	일정	일정은 달력을 통해 요청 및 등록될 수 있어야한다.
9		해당 날짜를 마우스 오버하여 일정의 내용을 상세 확인할 수 있어야한다.
10		일주일간의 일정을 타임테이블을 통해 상세 확인할 수 있어야한다.
11		자신이 살고 있는 동의 일정만 선택하여 볼 수 있어야한다.
12		자신이 살고 있는 아파트의 일정을 선택하여 볼 수 있어야 한다.
13		입주민이 요청한 일정은 테이블로 그려져 관리자의 승인을 기다려야한다.
14		관리자는 색상을 통해 일정의 중요도를 나타낼 수 있어야한다.
15		달력의 크기는 일정해야 한다.(일정이 많은 시 +개수로 나타낸다.)

요구분석 정의서

순번	구분	요구사항
16	보안 및 회원가입 로그인	회원가입 후에 로그인 시에 암호화 된 비밀번호를 비교, 로그인 성공 시 스프링 시큐리티에 정보를 담고, 권한에 따른 페이지 접속
17		스프링 시큐리티를 적용하여, 권한에 따른 페이지 요청 시 에러와 승인 라우팅
18		회원가입에 필요한 여러가지 정보를 받아, 암호화 후 비밀번호 저장
19	아파트 등록	주소로 아파트를 검색하고 아파트 관련 api를 조회하여 관련 데이터(아파트 고유 코드번호, 위도 경도 등등)를 데이터베이스에 넣고, 등록한 아파트를 사용할 수 있게 한다. 관리사무소, 입주민이 가입할 때에 이 과정이 선행되어야 한다.
20	회원 승인	거주민이 회원가입을 했을 때에 관리사무소는 승인을 할 수 있다. 승인이 되기 전에는 사용할 수 없다
21	지도	회원 가입할 때에 등록한 아파트 기준 위치에 편의시설을 보여준다.

요구분석 정의서

순번	구분	요구사항
22	민원	민원 작성을 위해서는 우선 해당하는 태그를 선택하여야 한다.
23		민원 작성은 작성 도중에 언제든지 취소할 수 있어야 한다.
24		민원 제목 클릭 시 해당 민원 상세 내용 조회가 가능하여야 한다.
25		작성한 민원은 자신과 관리자만 조회할 수 있어야 한다.
26		작성한 민원은 취소가 가능하여야 한다.
27		작성한 민원은 나의 민원에 게시글로 남겨져 관리자의 답변을 기다려야한다.
28		민원 조회 화면에서 다시 작성 화면으로 넘어갈 수 있어야 한다.
29		한 페이지에서 조회할 수 있는 민원의 수는 동일해야 한다.
30		한 페이지에서 민원들은 글 번호에 따라 오름차순으로 정렬되어야 한다.
31	관리자 민원처리	관리자는 입주민들의 민원을 조회 가능하여야 한다.
32		관리자는 민원 하나당 하나의 답변을 남길 수 있다.
33		민원 삭제를 비롯하여 답변 수정 및 삭제가 가능하여야 한다.
34		답변이 아직 등록되지 않았거나 삭제 되었을 경우 이는 표현 되어야 한다.
35		관리자는 답변 작성 도중 취소가 가능하여야 한다.
36		관리자는 민원 조회 중 다시 나의 민원으로 돌아갈 수 있어야 한다.
37		답변 등록 혹은 수정할 경우 날짜는 현재 시각으로 갱신되어야 한다.

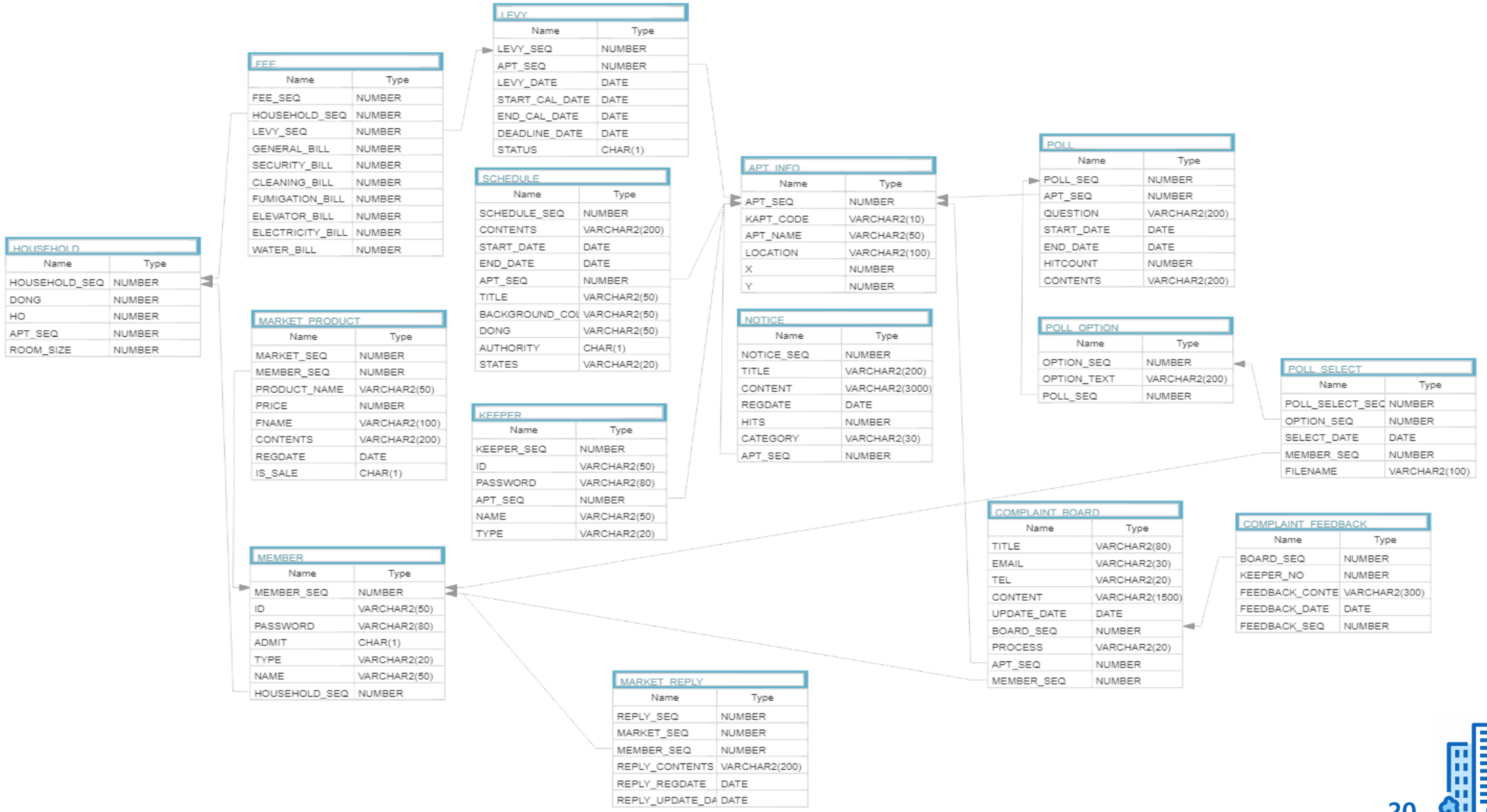
요구분석 정의서

순번	구분	요구사항
38	전자투표	관리자 페이지를 통해서 전자투표를 생성 할 수 있어야 한다.
39		투표 생성 시 여러 항목을 추가 할 수 있어야 한다.
40		생성된 전자투표에 사용자가 참여 할 수 있어야 한다.
41		사용자는 생성된 전자투표의 항목을 반드시 선택 해야만 한다.
42		사용자는 항목 선택 후 서명 할 수 있어야 한다.
43		사용자는 참여한 투표에 중복 투표 할 수 없어야 한다.
44		사용자는 전체,진행중, 종료 목록을 볼 수 있어야 한다.
45		사용자 페이지의 종료 된 투표는 막대그래프로 결과를 볼 수 있어야 한다.
46		관리자 페이지의 종료 된 투표는 참여자목록을 볼 수 있어야 한다.
47		관리자는 종료 된 투표의 참여자목록에 등록된 서명을 클릭 시 원본크기로 볼 수 있어야 한다.

요구분석 정의서

순번	구분	요구사항
48	중고거래	중고거래에 사진과 함께 글을 올릴 수 있어야 한다.
49		중고거래 글에 댓글을 달 수 있어야 한다.
50		중고거래에 글을 수정/삭제할 수 있다
51		댓글을 수정할 수 있다.
52	공지사항	관리자는 게시판에 공지사항을 등록할 수 있어야한다.
53		게시한 순서대로 공지를 확인할 수 있어야한다.
54		게시물을 삭제 및 수정할 수 있어야한다.
55		클릭 시 조회수가 증가되어야 한다.
56		게시물이 10개 이상이면 다음 페이지에 게시되어야 한다.

ERD


HOUSEHOLD
Name Type
HOUSEHOLD_SEQ NUMBER
DONG NUMBER
HO NUMBER
APT_SEQ NUMBER
ROOM_SIZE NUMBER
FEE
Name Type
FEE_SEQ NUMBER
HOUSEHOLD_SEQ NUMBER
LEVY_SEQ NUMBER
GENERAL_BILL NUMBER
SECURITY_BILL NUMBER
CLEANING_BILL NUMBER
FUMIGATION_BILL NUMBER
ELEVATOR_BILL NUMBER
ELECTRICITY_BILL NUMBER
WATER_BILL NUMBER
LEVY
Name Type
LEVY_SEQ NUMBER
APT_SEQ NUMBER
LEVY_DATE DATE
START_CAL_DATE DATE
END_CAL_DATE DATE
DEADLINE_DATE DATE
STATUS CHAR(1)
SCHEDULE
Name Type
SCHEDULE_SEQ NUMBER
CONTENTS VARCHAR2(200)
START_DATE DATE
END_DATE DATE
APT_SEQ NUMBER
TITLE VARCHAR2(50)
BACKGROUND_COL VARCHAR2(50)
DONG VARCHAR2(50)
AUTHORITY CHAR(1)
STATES VARCHAR2(20)
APT_INFO
Name Type
APT_SEQ NUMBER
KAPT_CODE VARCHAR2(10)
APT_NAME VARCHAR2(50)
LOCATION VARCHAR2(100)
X NUMBER
Y NUMBER
POLL
Name Type
POLL_SEQ NUMBER
APT_SEQ NUMBER
QUESTION VARCHAR2(200)
START_DATE DATE
END_DATE DATE
HITCOUNT NUMBER
CONTENTS VARCHAR2(200)
MARKET_PRODUCT
Name Type
MARKET_SEQ NUMBER
MEMBER_SEQ NUMBER
PRODUCT_NAME VARCHAR2(50)
PRICE NUMBER
FNAME VARCHAR2(100)
CONTENTS VARCHAR2(200)
REGDATE DATE
IS_SALE CHAR(1)
NOTICE
Name Type
NOTICE_SEQ NUMBER
TITLE VARCHAR2(200)
CONTENT VARCHAR2(3000)
REGDATE DATE
HITS NUMBER
CATEGORY VARCHAR2(30)
APT_SEQ NUMBER
POLL_OPTION
Name Type
OPTION_SEQ NUMBER
OPTION_TEXT VARCHAR2(200)
POLL_SEQ NUMBER
POLL_SELECT
Name Type
POLL_SELECT_SEQ NUMBER
OPTION_SEQ NUMBER
SELECT_DATE DATE
MEMBER_SEQ NUMBER
FILENAME VARCHAR2(100)
KEEPER
Name Type
KEEPER_SEQ NUMBER
ID VARCHAR2(50)
PASSWORD VARCHAR2(80)
APT_SEQ NUMBER
NAME VARCHAR2(50)
TYPE VARCHAR2(20)
MEMBER
Name Type
MEMBER_SEQ NUMBER
ID VARCHAR2(50)
PASSWORD VARCHAR2(80)
ADMIT CHAR(1)
TYPE VARCHAR2(20)
NAME VARCHAR2(50)
HOUSEHOLD_SEQ NUMBER
COMPLAINT_BOARD
Name Type
TITLE VARCHAR2(80)
EMAIL VARCHAR2(30)
TEL VARCHAR2(20)
CONTENT VARCHAR2(1500)
UPDATE_DATE DATE
BOARD_SEQ NUMBER
PROCESS VARCHAR2(20)
APT_SEQ NUMBER
MEMBER_SEQ NUMBER
COMPLAINT_FEEDBACK
Name Type
BOARD_SEQ NUMBER
KEEPER_NO NUMBER
FEEDBACK_CONTE VARCHAR2(300)
FEEDBACK_DATE DATE
FEEDBACK_SEQ NUMBER
MARKET_REPLY
Name Type
REPLY_SEQ NUMBER
MARKET_SEQ NUMBER
MEMBER_SEQ NUMBER
REPLY_CONTENTS VARCHAR2(200)
REPLY_REGDATE DATE
REPLY_UPDATE_DA DATE

데이터 베이스 구조

APT_INFO

Column		Data Type	Key	Not Null
APT_SEQ	아파트 번호	NUMBER	PRIMARY KEY	Y
KAPT_CODE	공공 아파트 코드	VARCHAR2(300byte)	·	Y
APT_NAME	아파트 이름	VARCHAR2(300byte)	·	Y
LOCATION	아파트 도로명 주소	VARCHAR2(500byte)	·	Y
X	경도	NUMBER	·	Y
Y	위도	NUMBER	·	Y

* KAPT_CODE : 공공데이터 포털에서 제공하는 대한민국 아파트 고유번호

HOUSEHOLD

Column		Data Type	Key	Not Null
HOUSEHOLD_SEQ	가구 번호	NUMBER	PRIMARY KEY	Y
APT_SEQ	아파트 번호	NUMBER	FOREIGN KEY	Y
DONG	동	NUMBER	·	Y
HO	호	NUMBER	·	Y
ROOM_SIZE	세대 면적	NUMBER	·	N

데이터 베이스 구조

KEEPER

Column		Data Type	Key	Not Null
KEEPER_SEQ	관리자 번호	NUMBER	PRIMARY KEY	Y
APT_SEQ	아파트 번호	NUMBER	FOREIGN KEY	Y
ID	관리자 아이디	VARCHAR2(100byte)	·	Y
PASSWORD	관리자 비밀번호	VARCHAR2(300byte)	·	Y
NAME	관리자 이름	VARCHAR2(100byte)	·	Y
TYPE	권한	VARCHAR2(20byte)	·	Y

* TYPE : Spring Security 에서 사용할 권한들이 들어 있습니다. (예로 ROLE_TENANT, ROLE_KEEPER)

MEMBER

Column		Data Type	Key	Not Null
MEMBER_SEQ	입주민 번호	NUMBER	PRIMARY KEY	Y
HOUSEHOLD_SEQ	가구 번호	NUMBER	FOREIGN KEY	Y
ID	입주민 아이디	VARCHAR2(100byte)	·	Y
PASSWORD	입주민 비밀번호	VARCHAR2(300byte)	·	Y
NAME	입주민 이름	VARCHAR2(100byte)	·	Y
ADMIT	승인여부	CHAR	·	Y
TYPE	권한	VARCHAR2(20byte)	·	Y

데이터 베이스 구조

FEE

Column		Data Type	Key	Not Null
FEE_SEQ	입주민 번호	NUMBER	PRIMARY KEY	Y
HOUSEHOLD_SEQ	가구 번호	NUMBER	FOREIGN KEY	Y
LEVY_SEQ	납부 번호	NUMBER	FOREIGN KEY	Y
GENERAL_BILL	일반관리비	NUMBER	·	Y
SECURITY_BILL	경비비	NUMBER	·	Y
CLEANING_BILL	청소비	NUMBER	·	Y
FUMIGATION_BILL	소독비	NUMBER	·	Y
ELEVATOR_BILL	승강기 유지비	NUMBER	·	Y
ELECTRICITY_BILL	전기료	NUMBER	·	Y
WATER_BILL	수도료	NUMBER	·	Y

데이터 베이스 구조

LEVY

Column		Data Type	Key	Not Null
LEVY_SEQ	납부 번호	NUMBER	PRIMARY KEY	Y
APT_SEQ	아파트 번호	NUMBER	FOREIGN KEY	Y
LEVY_DATE	납부일자	DATE	·	Y
START_CAL_DATE	산출 시작일	DATE	·	N
END_CAT_DATE	산출 종료일	DATE	·	N
DEADLINE_DATE	납부 마감일	DATE	·	Y

NOTICE

Column		Data Type	Key	Not Null
NOTICE_SEQ	공지 번호	NUMBER	PRIMARY KEY	Y
APT_SEQ	아파트 번호	NUMBER	FOREIGN KEY	Y
TITLE	제목	VARCHAR2(200byte)	·	Y
CONTENT	내용	VARCHAR2(3000byte)	·	Y
REGDATE	작성일	DATE	·	Y
HITS	조회수	NUMBER	·	Y
CATEGORY	구분	VARCHAR2(30byte)	·	Y

데이터 베이스 구조

SCHEDULE

Column		Data Type	Key	Not Null
SCHEDULE_SEQ	스케줄 번호	NUMBER	PRIMARY KEY	Y
APT_SEQ	아파트 번호	NUMBER	FOREIGN KEY	Y
TITLE	제목	VARCHAR2(50byte)	·	Y
CONTENTS	내용	VARCHAR2(200byte)	·	Y
START_DATE	일정 시작일	DATE	·	Y
END_DATE	일정 마감일	DATE	·	Y
BACKGROUND_COLOR	배경색	VARCHAR2(50byte)	·	Y
DONG	동	VARCHAR2(50byte)	·	Y
AUTHORITY	권한	CHAR	·	Y
STATES	일정 등록 상태	VARCHAR2(20byte)	·	Y

* **Authority** : 0과 1로 저장되며 사용자가 요청한 상태는 0, 등록된 상태는 1로 저장됩니다.
* States : 일정 등록 진행 상태를 나타냅니다.

데이터 베이스 구조

POLL

Column		Data Type	Key	Not Null
POLL_SEQ	투표 번호	NUMBER	PRIMARY KEY	Y
APT_SEQ	아파트 번호	NUMBER	FOREIGN KEY	Y
QUESTION	안건명	VARCHAR2(50byte)	·	Y
START_DATE	투표 시작일	DATE	·	Y
END_DATE	투표 마감일	DATE	·	Y
HITCOUNT	조회수	NUMBER	·	Y
CONTENTS	안건내용	VARCHAR2(500byte)	·	Y

POLL_OPTION

Column		Data Type	Key	Not Null
OPTION_SEQ	항목 번호	NUMBER	PRIMARY KEY	Y
OPTION_TEXT	항목명	VARCHAR2(50byte)	·	Y
POLL_SEQ	투표번호	NUMBER	FOREIGN KEY	Y

데이터 베이스 구조

POLL_SELECT

Column		Data Type	Key	Not Null
POLL_SELECT_SEQ	투표 선택 번호	NUMBER	PRIMARY KEY	Y
MEMBER_SEQ	입주민 번호	NUMBER	FK	Y
OPTION_SEQ	항목 번호	NUMBER	FOREIGN KEY	Y
SELECT_DATE	선택 날짜	DATE	·	Y
FILENAME	파일명	VARCHAR2(100byte)	·	Y

COMPLAINT_BOARD

Column		Data Type	Key	Not Null
BOARD_SEQ	민원 번호	NUMBER	PRIMARY KEY	Y
MEMBER_SEQ	작성자 번호	NUMBER	FOREIGN KEY	Y
TITLE	민원 제목	VARCHAR2(80byte)	·	Y
EMAIL	이메일	VARCHAR2(30byte)	·	N
TEL	연락처	VARCHAR2(20byte)	·	N
CONTENT	민원 내용	VARCHAR2(1500byte)	·	N
UPDATE_DATE	갱신일	DATE	·	Y
PROCESS	진행상태	VARCHAR2(20byte)	·	N

데이터 베이스 구조

COMPLAINT_FEEDBACK

Column		Data Type	Key	Not Null
FEEDBACK_SEQ	답변 번호	NUMBER	PRIMARY KEY	Y
BOARD_BNO	민원 번호	NUMBER	FOREIGN KEY	Y
FEEDBACK_CONTENT	답변 내용	VARCHAR2(300byte)	·	Y
FEEDBACK_DATE	답변일자	VARCHAR2(20byte)	·	Y

MARKET_PRODUCT

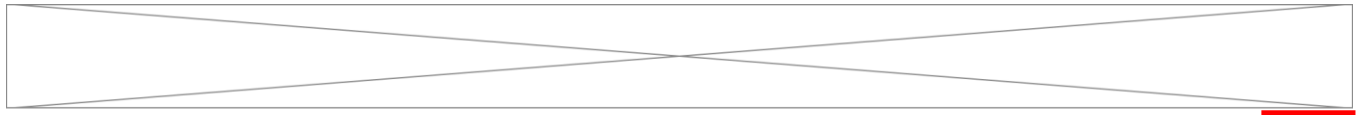
Column		Data Type	Key	Not Null
MARKER_SEQ	중고거래 번호	NUMBER	PRIMARY KEY	Y
MEMBER_SEQ	중고거래 작성자	NUMBER	FOREIGN KEY	N
PRODUCT_NAME	판매 물품명	VARCHAR2(30byte)	·	N
PRICE	판매 가격	NUMBER	·	N
FNAME	물품 이미지 파일명	VARCHAR2(100byte)	·	N
CONTENTS	판매 내용	VARCHAR2(200byte)	·	N
REGDATE	중고거래 게시글 작성일	DATE	·	N
ISSALE	판매여부	CHAR	·	N

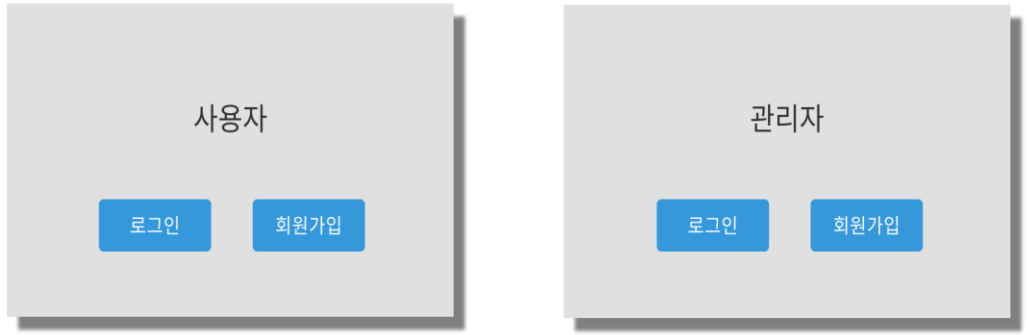
데이터 베이스 구조

MARKET_REPLY

Column		Data Type	Key	Not Null
REPLY_SEQ	댓글 번호	NUMBER	PRIMARY KEY	Y
MARKET_SEQ	중고거래 번호	NUMBER	FOREIGN KEY	Y
MEMBER_SEQ	댓글 작성자	NUMBER	FOREIGN KEY	N
REPLY_CONTENTS	댓글 내용	VARCHAR2(200byte)	·	N
REPLY_REGDATE	댓글 등록일	DATE	·	N
REPLY_UPDATEDATE	댓글 수정일	DATE	·	N

화면설계



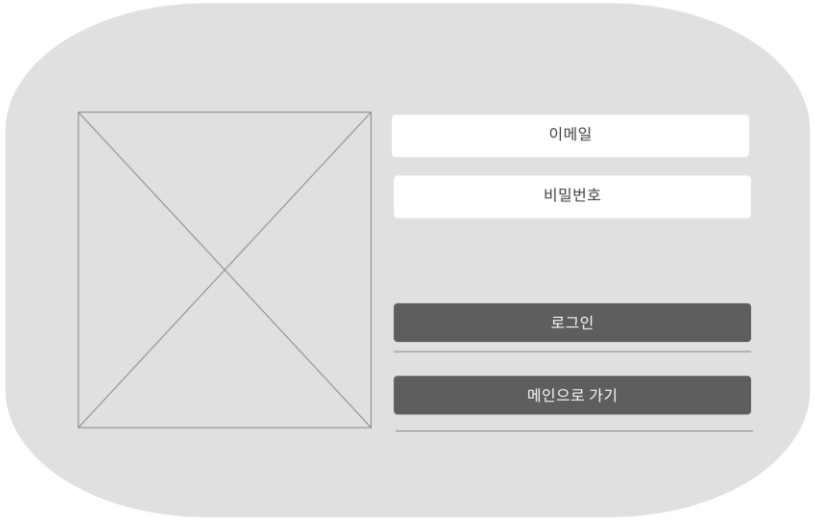

사용자
로그인
회원가입
관리자
로그인
회원가입

메인화면

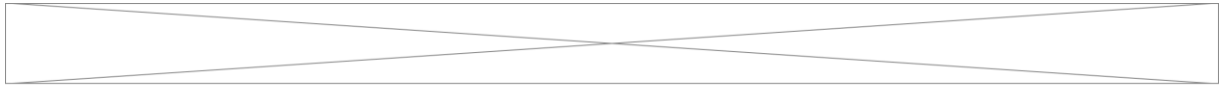

아파트등록
아파트 등록화면
Aptogether
메인
반갑습니다. 아파트 등록 페이지입니다.
아파트 등록
우편번호
주소
상세주소
법정동 코드를 기준으로 검색한 아파트 목록
풍림아이원(A22123)
선택
대방 노블랜드 2차(A2233231)
선택

화면설계

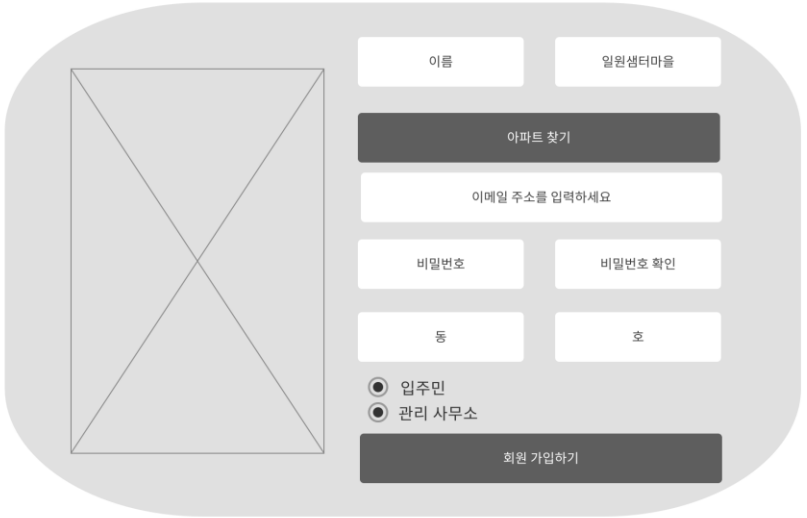
Aptogether - 로그인



로그인

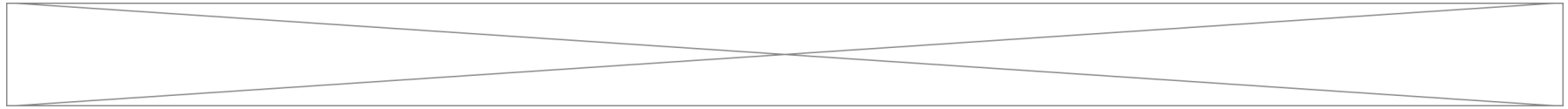


Aptogether - 회원가입



회원가입

화면설계



전자투표 등록하기
투표등록

관리비 청구하기
입주민조회

아파트 일정 등록하기
일정등록

민원처리 업무
민원확인

회원가입 대기중인 주민

show
10

search
검색

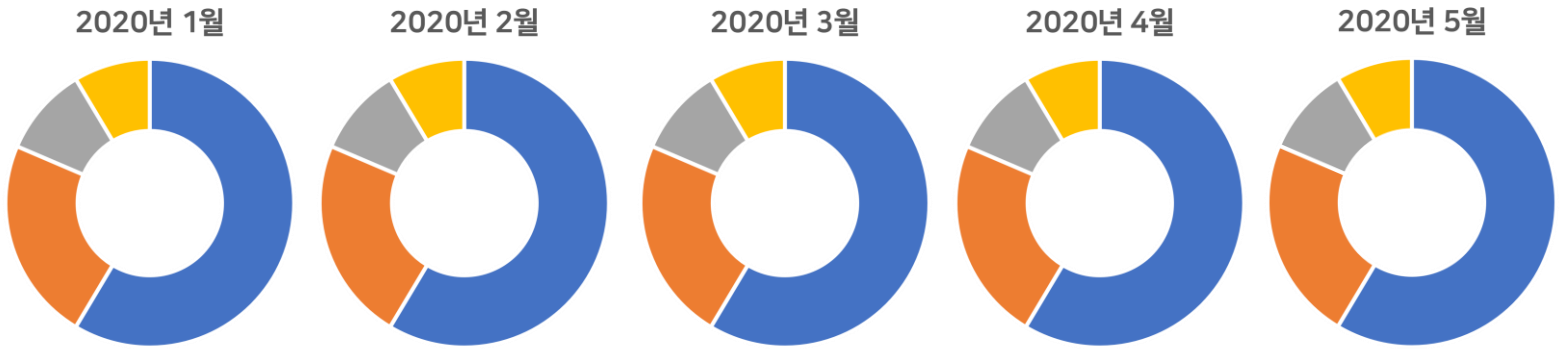
이름	동	호	이메일	승인
사용자	101동	201호	aa@aa.com	승인
사용자	102동	301호	bb@bb.com	승인

< 1 2 3 4 >

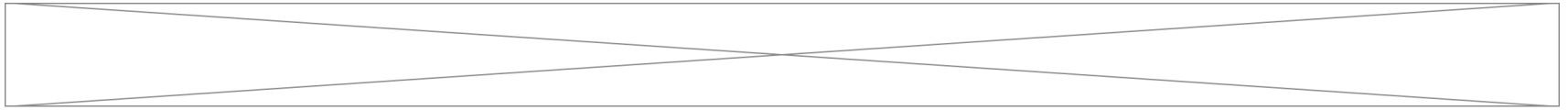
관리자 메인화면

화면설계

입주민 메인화면



화면설계



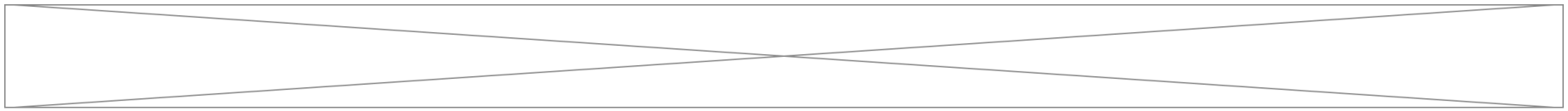
2020년 6월 관리비 부과

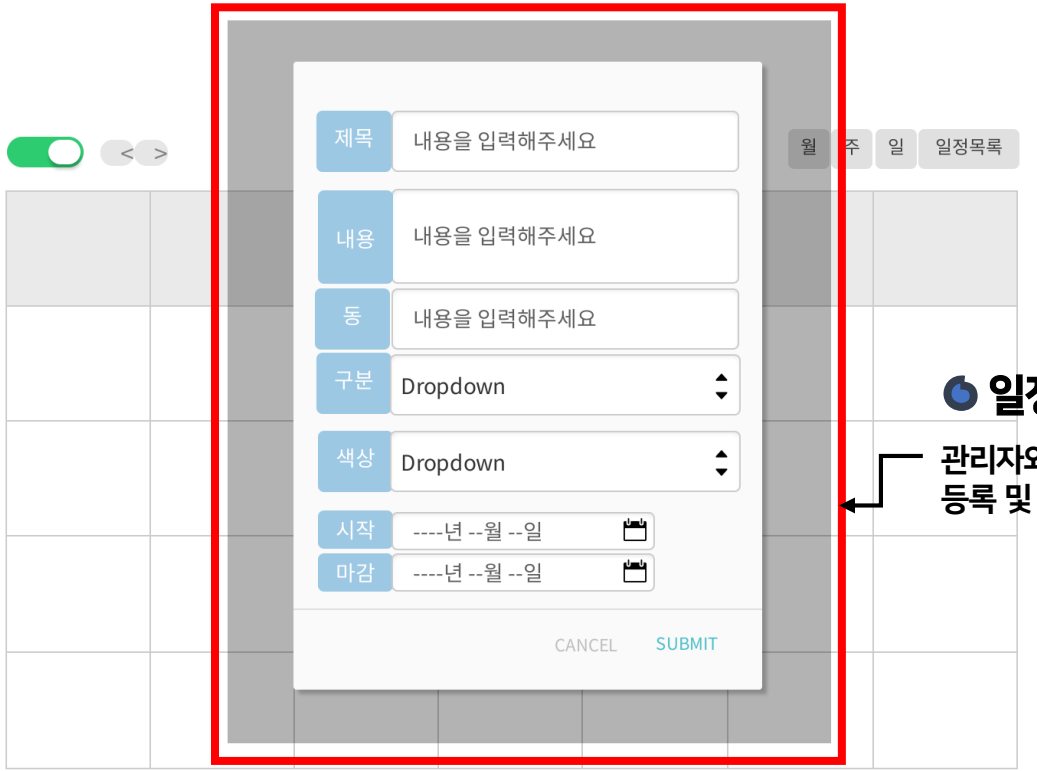
관리비 부과기초작업 110동

동	호	일반 관리비	경비비	소독비	승강기유지비	전기료	수도료	부과작업
110	1010	9,000	10,000	85,000	60,000	50,000	100,000	저장
	1011	8,500	20,000	85,000	60,000	80,000	54,000	저장
	1012	0	0	0	0	0	0	관리비조정
	1013	0	0	0	0	0	0	관리비조정
	1014	0	0	0	0	0	0	관리비조정

관리자 관리비 부과화면

화면설계

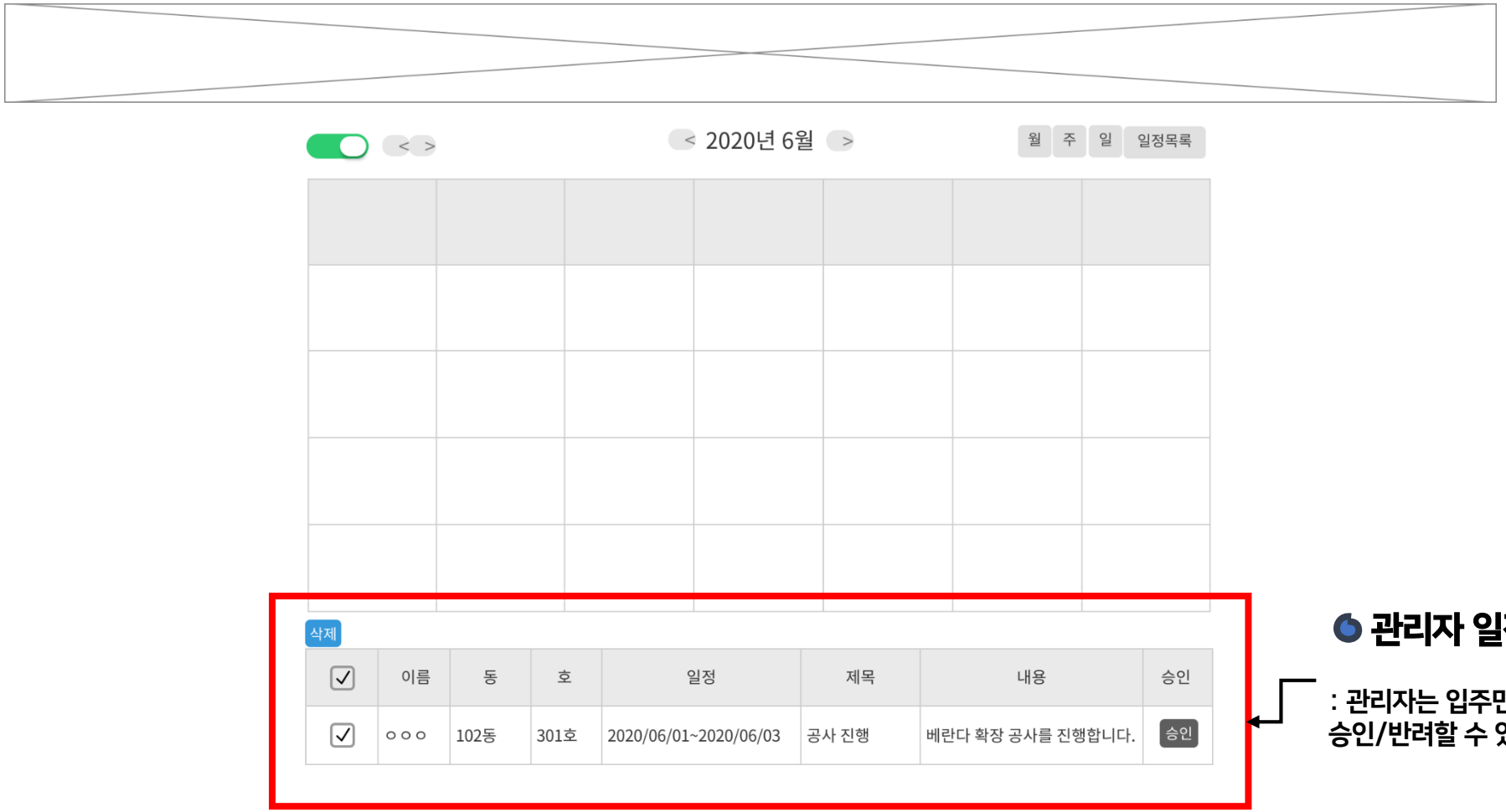




일정 등록 및 요청

관리자와 입주민은 달력을 클릭하여
등록 및 요청을 진행합니다.

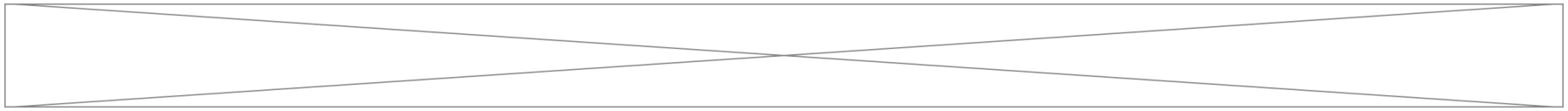
화면설계



관리자 일정화면

: 관리자는 입주민의 일정을 승인/반려할 수 있습니다.

화면설계



NOTICE

총 __ 개의 게시물이 있습니다.

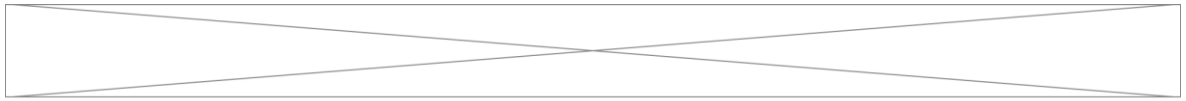
notice

글번호	제목	작성일
3	공지 사항	2020.06.16
2	공지 사항	2020.06.14
1	공지 사항	2020.04.12

< 1 2 3 4 >

공지사항

화면설계

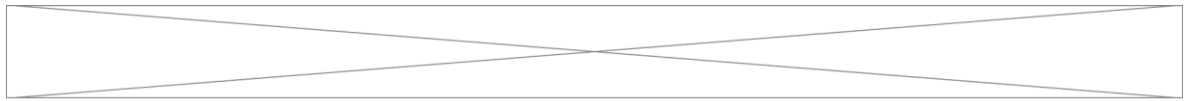


제목 작성일 | 조회수

내용

Modify List

공지 세부화면



제목 수정하세요 구분

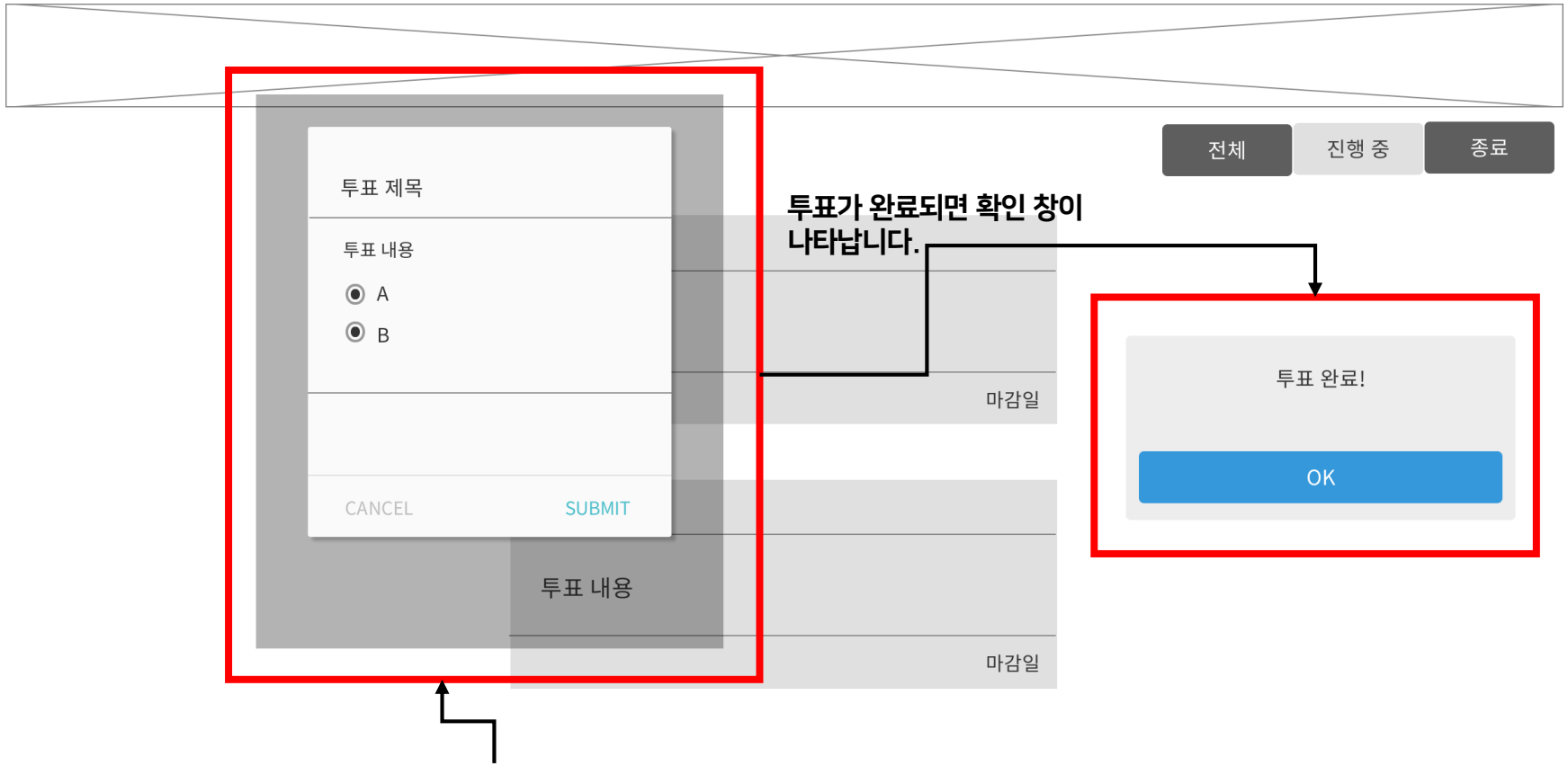
공지
일반

수정할 내용을 입력하세요.

Modify Delete List

공지 수정화면

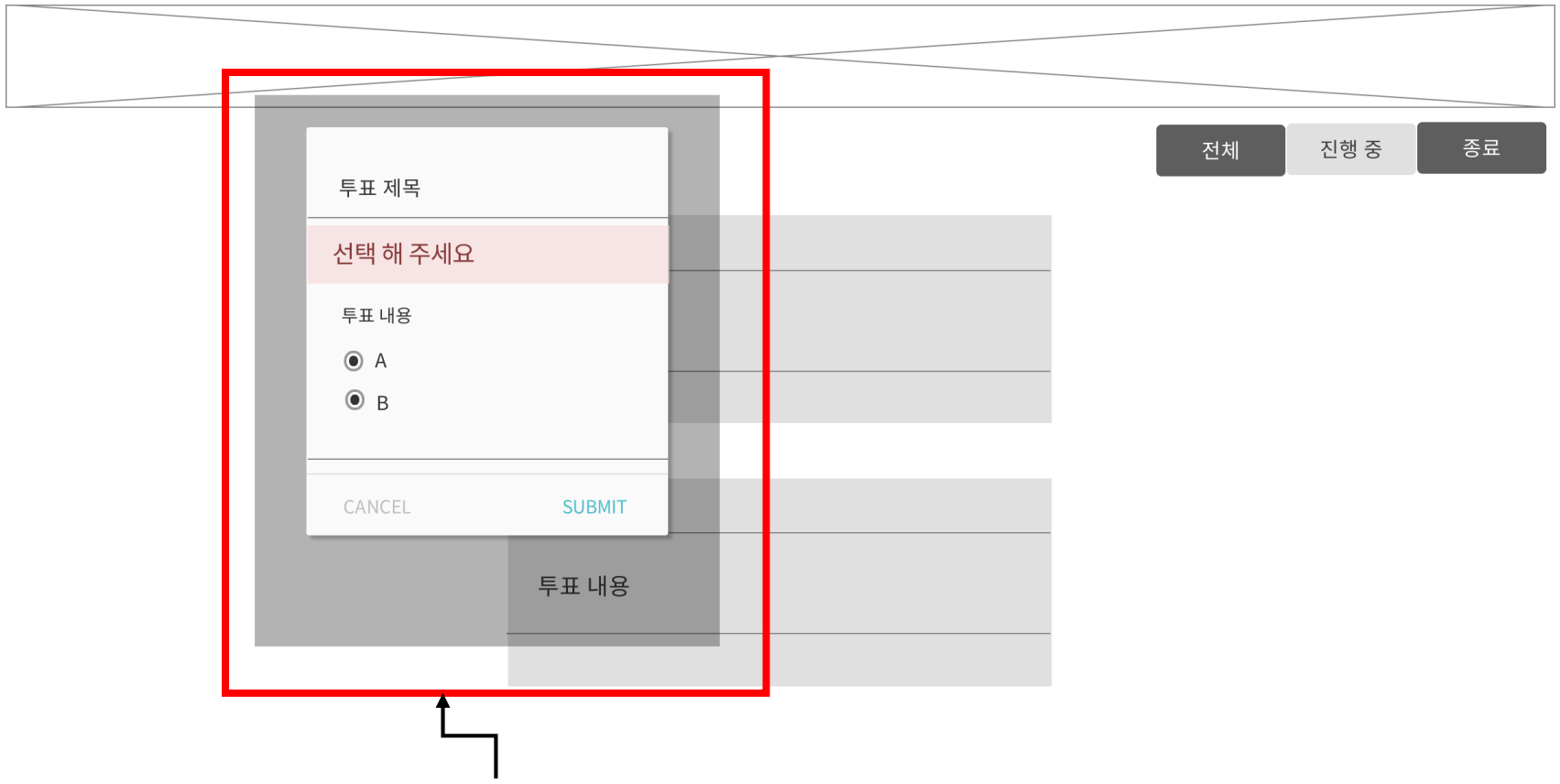
화면설계



입주민 투표화면

입주민은 modal 창을 이용해
투표를 진행합니다.

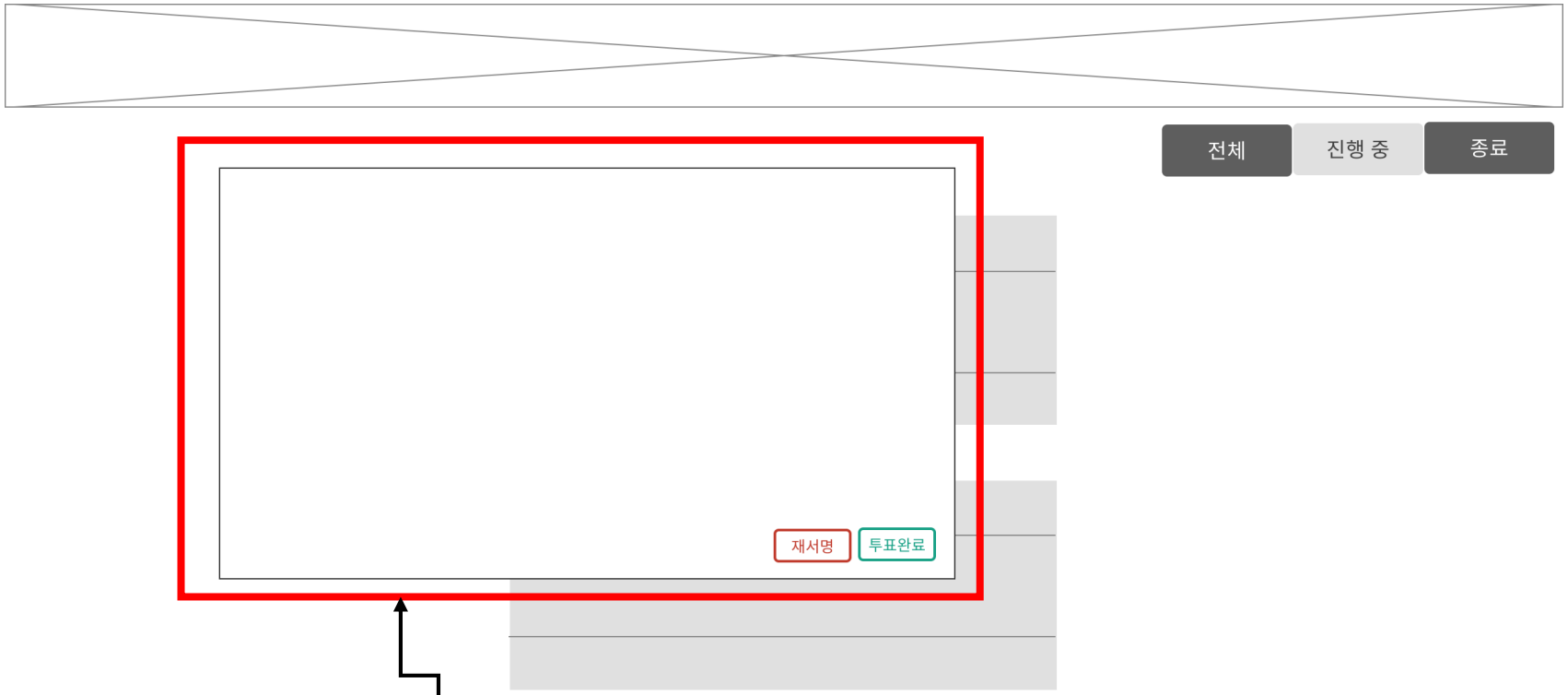
화면설계



입주민 투표화면

미선택 시 경고 화면이 나타납니다.

화면설계



입주민 전자서명

투표를 완료하면 전자서명 창이 나타납니다.

화면설계

투표 생성

관리자 투표 생성

관리자는 modal 창을 이용하여
투표를 생성합니다.

화면설계

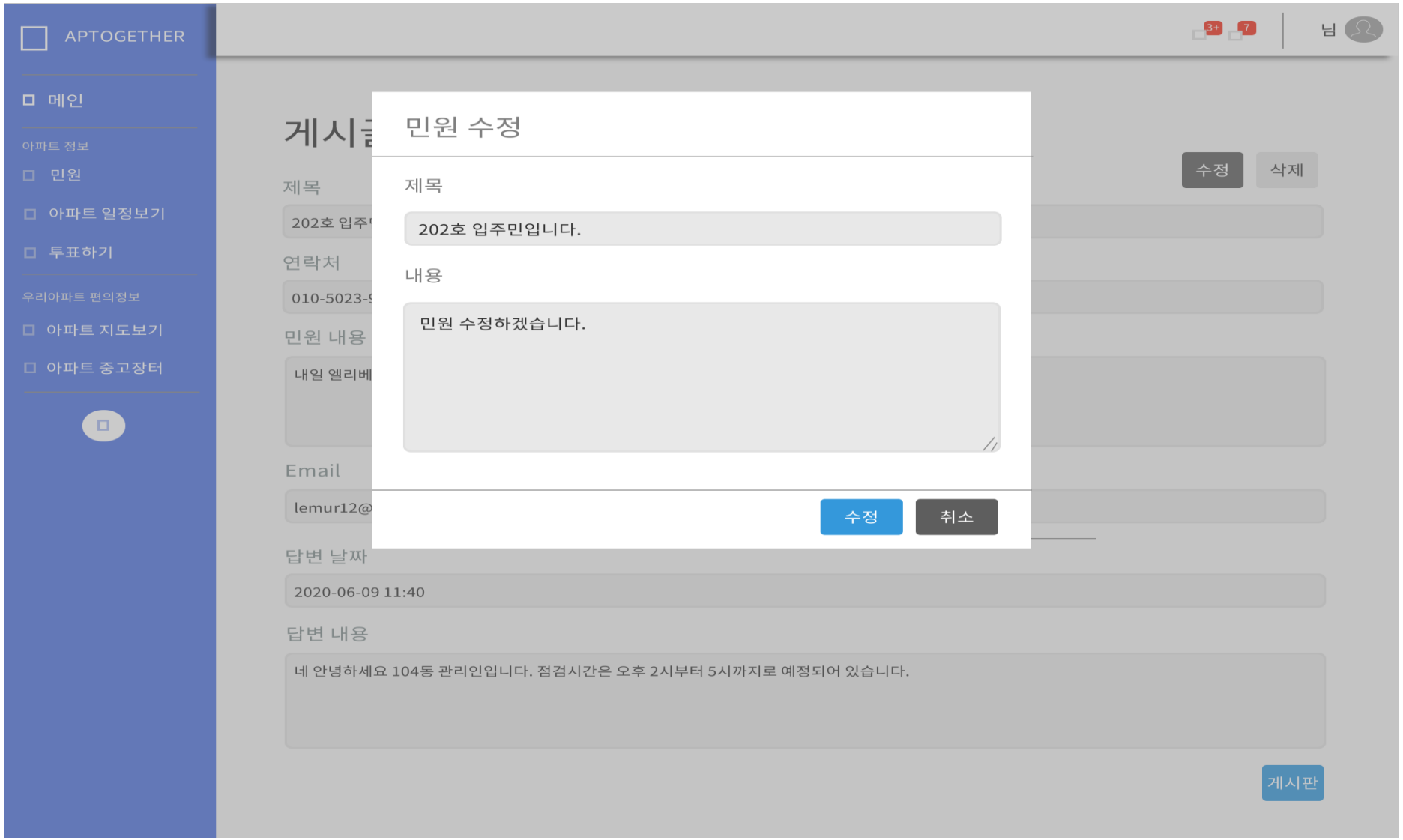

APTOGETHER
메인
아파트 정보
민원
아파트 일정보기
투표하기
우리아파트 편의정보
아파트 지도보기
아파트 중고장터
3+
7
님
나의 민원
민원작성
글 번호
제목
작성자
작성일
진행상태
1
202호 입주민입니다.
입주민1
2020-06-09
답변완료
2
테스트2
입주민1
2020-06-09
진행중
3
테스트3
입주민1
2020-06-10
진행중
1

입주민 글 목록

화면설계

입주민 글 상세조회

화면설계


APTOGETHER
메인
아파트 정보
민원
아파트 일정보기
투표하기
우리아파트 편의정보
아파트 지도보기
아파트 중고장터
님
게시글
수정
삭제
제목
202호 입주
연락처
010-5023-
민원 내용
내일 엘리베
Email
lemur12@
답변 날짜
2020-06-09 11:40
답변 내용
네 안녕하세요 104동 관리인입니다. 점검시간은 오후 2시부터 5시까지로 예정되어 있습니다.
게시판
민원 수정
제목
202호 입주민입니다.
내용
민원 수정하겠습니다.
수정
취소

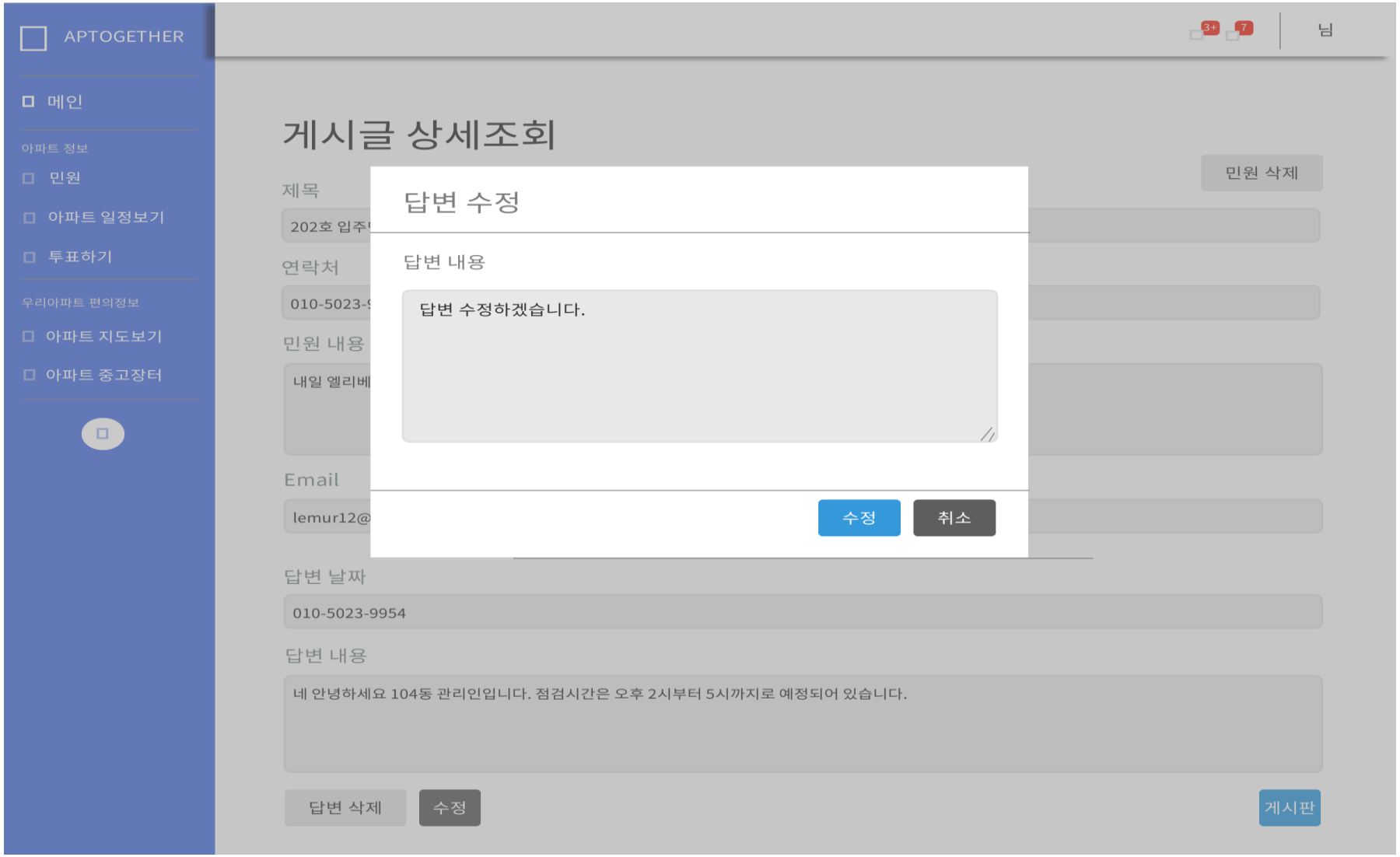
입주민 글 수정

화면설계


APTOGETHER
메인
아파트 정보
민원
아파트 일정보기
투표하기
우리아파트 편의정보
아파트 지도보기
아파트 중고장터
3+
7
님
나의 민원
대기 민원: 2
해결 민원: 1
글 번호
제목
작성자
작성일
진행상태
1
202호 입주민입니다.
입주민1
2020-06-17
답변완료
2
테스트2
입주민1
2020-06-17
진행중
3
테스트3
입주민1
2020-06-18
진행중
1

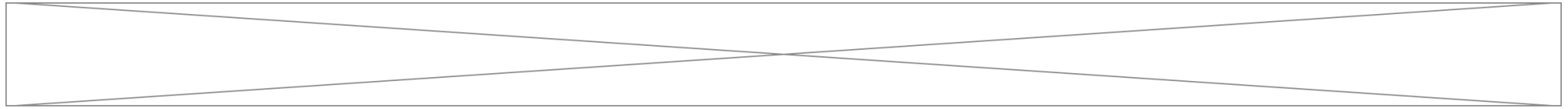
관리자 글 목록

화면설계



관리자 글 상세보기

화면설계



작성자 read only

판매물품 물품을 입력하세요

가격 가격을 입력하세요

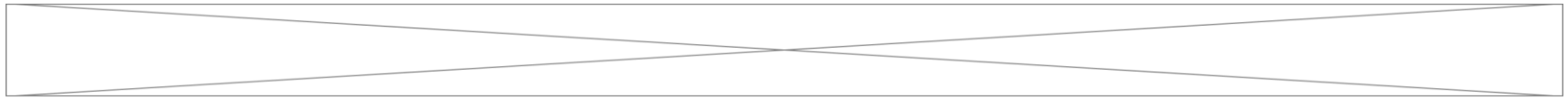
판매현황 내용을 입력해주세요

첨부 파일선택 선택된 파일이 없습니다

내용 내용을 입력하세요

중고거래 등록

화면설계

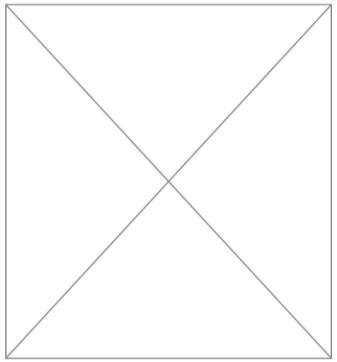


작성자 A

판매물품 청소기

가격 20,000원

내용



청소기 판매합니다.
댓글로 문의 주세요!

수정 삭제

댓글쓰기

내용을 입력해주세요 등록

댓글목록

남겨진 댓글입니다. 2020.06.26 09:21

남겨진 댓글입니다. 2020.06.26 09:21

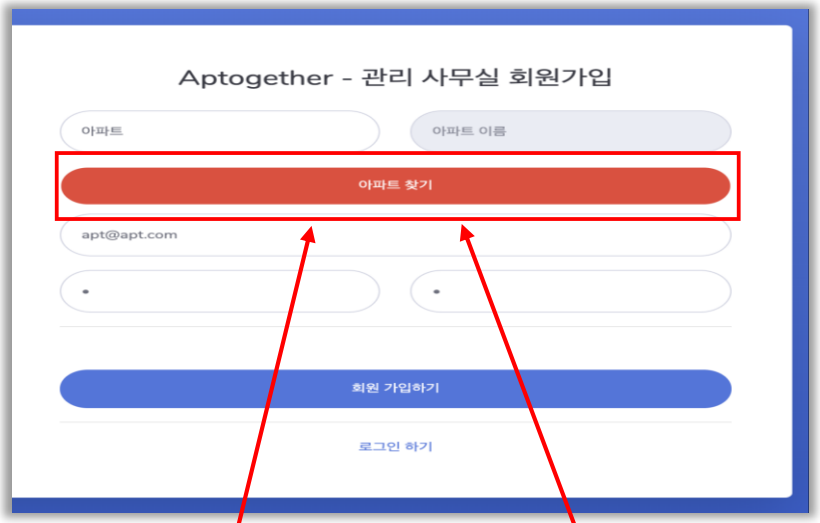
중고거래 상세보기

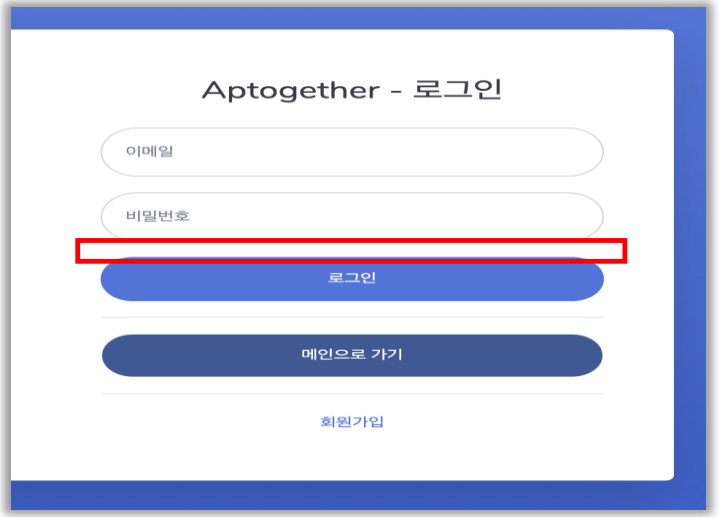
APTOGETHER

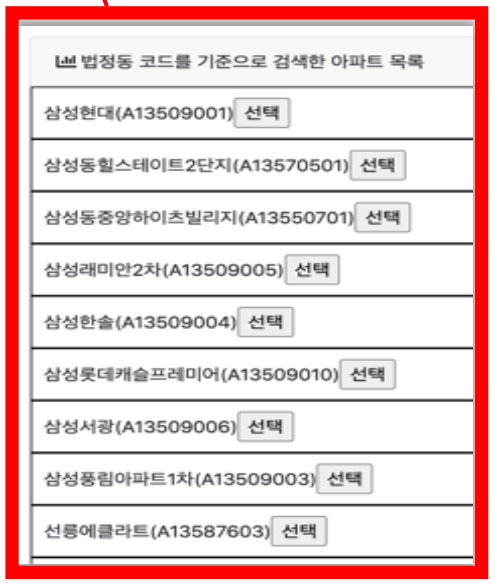
아파트 통합관리 웹 솔루션

04. 기능

회원가입 & 로그인 페이지 – 관리사무소 & 입주민







- **아파트 찾기 버튼**
  - 등록된 아파트를 검색하는 모달창이 띄웁니다.
  - 주소 클릭을 하면 법정동 코드를 가져와 아파트 목록을 나열합니다.
- **아파트 선택**
  - 선택된 아파트를 DB에 저장, 이미 값이 있다면 중복 메시지가 나오고 DB에 저장되지 않습니다.
- **로그인 실패 시, 에러메세지 보여줌**

회원가입 & 로그인 페이지 – 관리사무소 & 입주민

주요코드 : 아파트 찾기 & 등록 기능

```java
@GetMapping(value = "/showAptList/{bjdCode}", produces = "application/json; charset=utf8")
public String showAptList(@PathVariable("bjdCode") String bjdCode, Authentication auth) {
    final String USER_AGENT = "Mozilla/5.0";
    String url1 = "http://apis.data.go.kr/1611000/AptListService/getLegaldongAptList?bjdCode=";
    String url2 = "&numOfRows=100&ServiceKey=" + ApiKeys.getAptListKey();

    URL targetUrl;
    log.info(bjdCode);
    try {
        targetUrl = new URL(url1 + bjdCode + url2);
        log.info(url1 + bjdCode + url2);
        HttpURLConnection con = (HttpURLConnection) targetUrl.openConnection();
        con.setRequestMethod("GET");
        con.setRequestProperty("User-Agent", USER_AGENT);
        DocumentBuilder builder;
        Document doc = null;
        int responseCode = con.getResponseCode();
        BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream(), "utf-8"));
        String inputLine;
        String responseString = new String();

        while ((inputLine = in.readLine()) != null) {
            responseString += inputLine;
        }
        in.close();
        if (responseCode == 200) {
            InputSource is = new InputSource(new StringReader(responseString));
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            doc = dBuilder.parse(is);
            XPathFactory xpathFactory = XPathFactory.newInstance();
            XPath xpath = xpathFactory.newXPath();

            XPathExpression expr = xpath.compile("//items/item");
            NodeList nodeList = (NodeList) expr.evaluate(doc, XPathConstants.NODESET);
            List<AptCodeVO> list = new ArrayList<AptCodeVO>();
            Gson gson = new Gson();
            for (int i = 0; i < nodeList.getLength(); i++) {
                NodeList child = nodeList.item(i).getChildNodes();
                AptCodeVO apt = new AptCodeVO();
                for (int j = 0; j < child.getLength(); j++) {
                    Node node = child.item(j);
                    log.info(node.getNodeName() + node.getTextContent());
                    if (node.getNodeName().equals("kaptCode")) {
                        apt.setKaptcode(node.getTextContent());
                    } else if (node.getNodeName().equals("kaptName")) {
                        apt.setAptName(node.getTextContent());
                    }
                }
                list.add(apt);
            }
            Map<String, List> map = new HashMap<String, List>();
            map.put("aptInfo", list);
            return gson.toJson(map);
        }
        System.out.println(responseString);

    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return "{\"status\": \"false\" }";
}
```

- 카카오 주소검색 API
  - 원하는 법정동 코드를 구하여 RestFul 형태로 아파트 목록을 요청합니다.
- 공공데이터 포털 API
  - 법정동 코드에 속한 아파트 목록을 제공해줍니다.
  - 공공데이터포털에서는 XML로 응답이 오기 때문에, JSON으로 변환하여 보내는 코드입니다.

회원가입 & 로그인 페이지 – 관리사무소 & 입주민

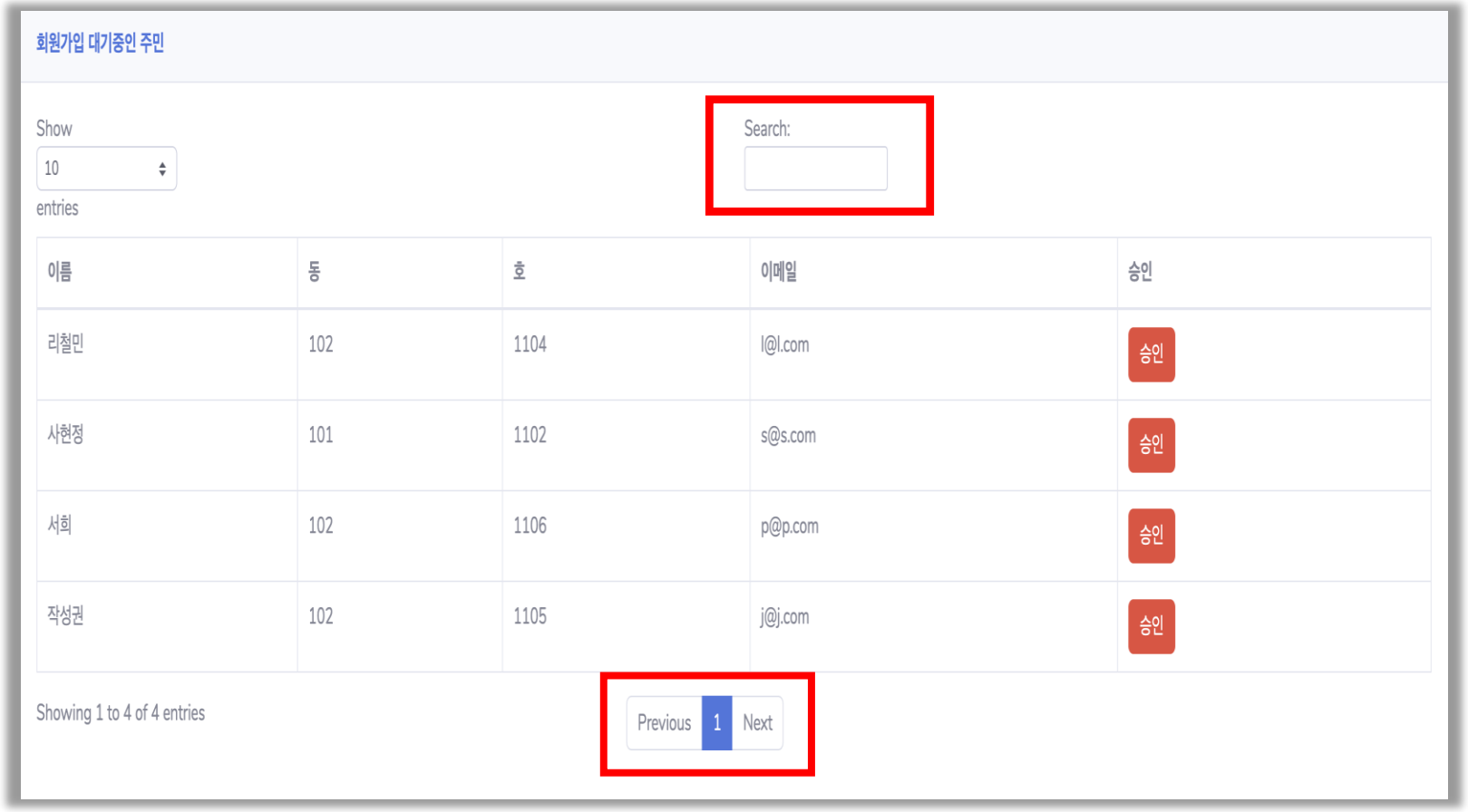
주요코드 : 아파트 찾기 & 등록 기능

```
<security:http pattern="/keeper/**" auto-config="true"
    use-expressions="true" authentication-manager-ref="customKeeperAuth">
    <security:csrf disabled="true" />

    <security:intercept-url pattern="/error/*"
        access="permitAll" />
    <security:intercept-url pattern="/apt/*"
        access="permitAll" />

    <security:intercept-url
        pattern="/keeper/signup" access="permitAll" />
    <security:intercept-url
        pattern="/keeper/signin" access="permitAll" />
    <security:intercept-url pattern="/keeper/**"
        access="hasRole('ROLE_KEEPER')" />

    <security:form-login login-page="/keeper/signin"
        default-target-url="/keeper/dashboard"
        login-processing-url="/keeper/signin" username-parameter="email"
        password-parameter="password" />

    <security:logout logout-url="/logout"
        invalidate-session="true" delete-cookies="JSESSION_ID" />
    <security:access-denied-handler
        error-page="/error/401" />
</security:http>
```

```
<security:http pattern="/tenant/**" auto-config="true"
    use-expressions="true" authentication-manager-ref="customTenantAuth">
    <security:csrf disabled="true" />

    <security:intercept-url pattern="/error/*"
        access="permitAll" />
    <security:intercept-url pattern="/apt/*"
        access="permitAll" />

    <security:intercept-url
        pattern="/tenant/signin" access="permitAll" />
    <security:intercept-url
        pattern="/tenant/signup" access="permitAll" />
    <security:intercept-url pattern="/tenant/**"
        access="hasRole('ROLE_TENANT')" />


    <security:form-login login-page="/tenant/signin"
        authentication-success-handler-ref="customLoginSuccessHandler"
        login-processing-url="/tenant/signin" username-parameter="email"
        password-parameter="password" />

    <security:logout logout-url="/logout"
        invalidate-session="true" delete-cookies="JSESSION_ID" />

    <security:access-denied-handler
        error-page="/error/401" />
</security:http>
```

- **카카오 주소검색 API**
  - 원하는 법정동 코드를 구하여 RestFul 형태로 아파트 목록을 요청합니다.
- **공공데이터 포털 API**
  - 법정동 코드에 속한 아파트 목록을 제공해줍니다.
  - 공공데이터포털에서는 XML로 응답이 오기 때문에, JSON으로 변환하여 보내는 코드입니다.
- **스프링 시큐리티**
  - 관리자와 입주민의 테이블이 다르기 때문에, 상황에 맞는 시큐리티의 user를 각 기능에 맞게 서비스를 재정의합니다.

관리자 페이지 – 입주민 승인



DATATABLES API

└ 자바스크립트 라이브러리인 Datatables API를 활용하여 검색과 페이징 처리 구현

입주민 사용 승인하기

└ 입주민 회원가입 시, 승인을 받아야 사용이 가능하도록 했습니다.

아파트 주변 편의시설 페이지 – 카카오 지도

도곡삼성래미안 근처 편의시설 입니다

좌표 정보 활용

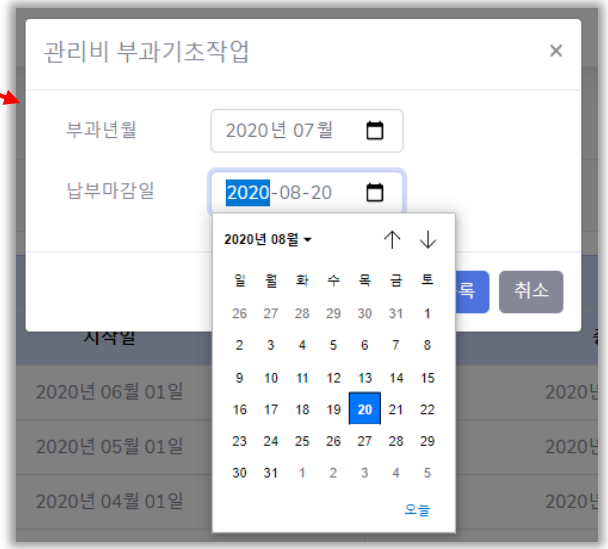
└ 회원가입 시, 아파트 정보에 등록되어 있는 아파트 좌표를 기준으로 주변 편의시설 지도를 그려줍니다.

카테고리 별 편의시설 조회

└ 카테고리 별로 선택을 하면 해당하는 편의시설들이 지도에 표시됩니다.

└ 지도에 표시된 편의시설을 클릭하면 해당 시설의 상세정보를 보여줍니다.

관리비 부과 페이지 – 부과년월 등록 & 조회기능



- 부과년월을 내림차순으로 정렬한 목록 정보제공
- [추가]버튼을 눌러 부과년월과 납부마감일을 선택 후, 부과일자 정보등록

관리비 부과 페이지 - 부과년월 등록 & 조회기능

주요코드 : 관리비 납부년월 부과 & 목록 조회

```
@PostMapping(value = "/addLevy", consumes = "application/json", produces = {MediaType.TEXT_PLAIN_VALUE})
public ResponseEntity<String> addLevy(@RequestBody LevyVO levy, Authentication auth){

    log.info("LevyVO : " + levy);

    CustomKeeper keeper = (CustomKeeper) auth.getPrincipal();
    levy.setAptSeq(keeper.getAptSeq());

    int insertCount = service.addLevy(levy);

    log.info("Levy Add COUNT : " + insertCount);

    return insertCount == 1
            ? new ResponseEntity<>("success", HttpStatus.OK)
            : new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
}
```

```
@GetMapping(value = "/listLevy", produces = { MediaType.APPLICATION_XML_VALUE,
                                              MediaType.APPLICATION_JSON_UTF8_VALUE })
public ResponseEntity<List<LevyVO>> listLevy(Authentication auth){

    LevyVO levy = new LevyVO();

    CustomKeeper keeper = (CustomKeeper) auth.getPrincipal();

    levy.setAptSeq(keeper.getAptSeq());

    return new ResponseEntity<>(service.listLevy(levy), HttpStatus.OK);
}
```

- 관리비 추가 시 어노테이션 @requestBody 사용

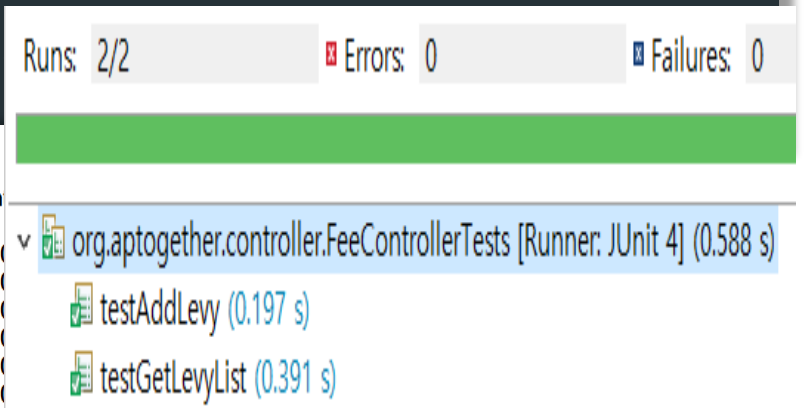
- 스프링시큐리티 활용
  - 관리자 & 입주민의 주체를 인증(Authentication)하여 SEQ값을 파라미터로 넘겨줍니다.

관리비 부과 페이지 – 부과년월 등록 & 조회기능

Junit Test : 관리비 납부년월 부과 & 목록 조회

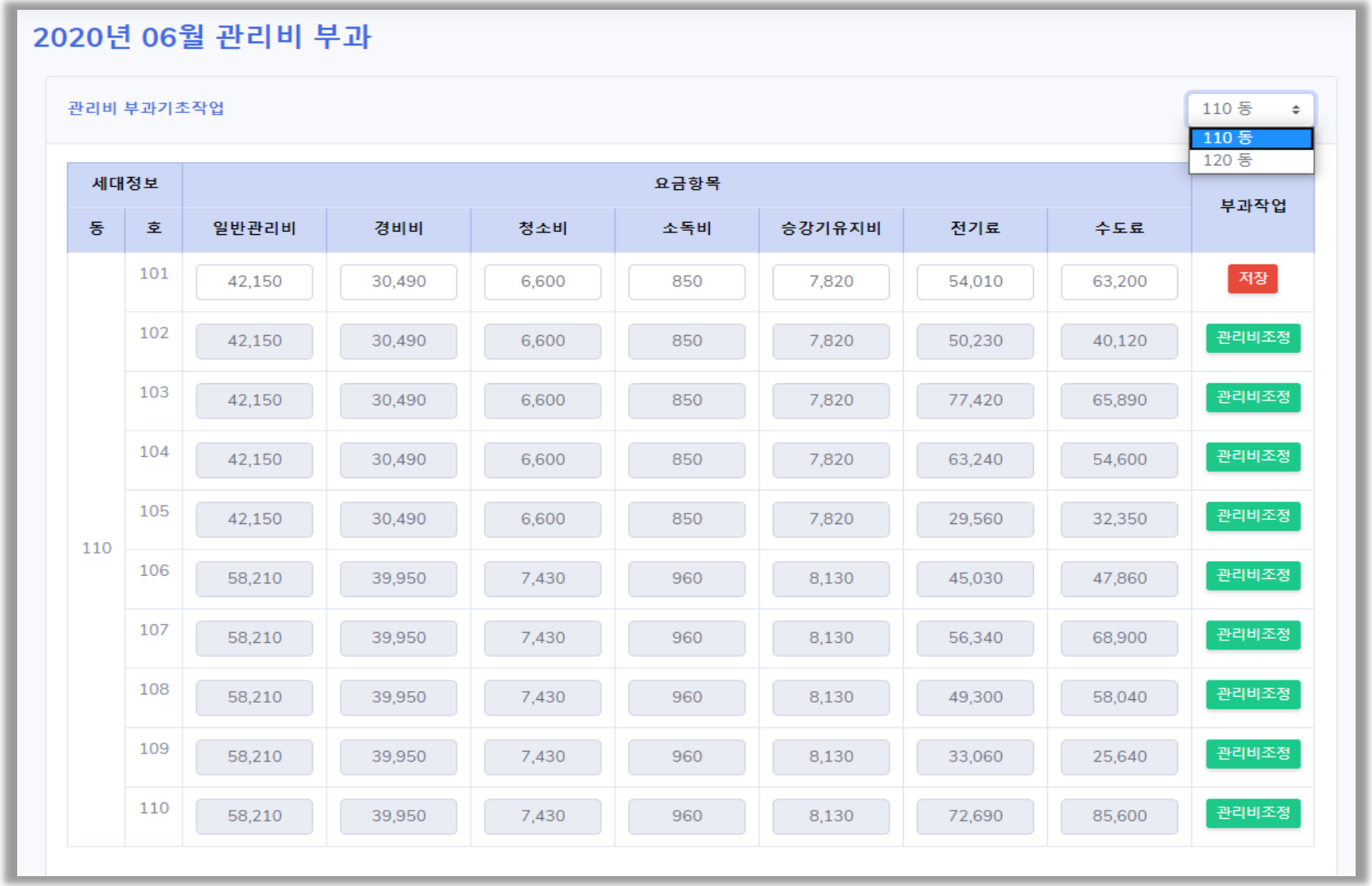
```java
@Test
public void testAddLevy() throws Exception{

    log.info(mockMvc.perform(MockMvcRequestBuilders.get("/keeper/addLevy"))
                                    .andReturn()
                                    .getModelAndView());
}

@Test
public void testGetLevyList() throws Exception{

    log.info(mockMvc.perform(MockMvcRequestBuilders.get("/keeper/listLevy"))
                    .andReturn()
                    .getModelAndView());
}
```

```
INFO : jdbc.resultsettable -
|--------|-------|---------------------|---------------------|---------------------|---------------------|------|
|levy_seq |apt_seq |levy_date          |start_cal_date       |end_cal_date         |deadline_date        |sta
|--------|-------|---------------------|---------------------|---------------------|---------------------|------|
|86      |9      |2020-06-01 00:00:00.0 |2020-06-01 00:00:00.0 |2020-06-30 00:00:00.0 |2020-07-2(
|85      |9      |2020-05-01 00:00:00.0 |2020-05-01 00:00:00.0 |2020-05-31 00:00:00.0 |2020-06-2(
|84      |9      |2020-04-01 00:00:00.0 |2020-04-01 00:00:00.0 |2020-04-30 00:00:00.0 |2020-05-2(
|83      |9      |2020-03-01 00:00:00.0 |2020-03-01 00:00:00.0 |2020-03-31 00:00:00.0 |2020-04-2(
|82      |9      |2020-02-01 00:00:00.0 |2020-02-01 00:00:00.0 |2020-02-29 00:00:00.0 |2020-03-2(
|81      |9      |2020-01-01 00:00:00.0 |2020-01-01 00:00:00.0 |2020-01-31 00:00:00.0 |2020-02-2(
|--------|-------|---------------------|---------------------|---------------------|---------------------|------|
```



- TestAddLevy
  - 관리비 납부년월 입력
- testGetLevyList
  - DB에 저장된 납부일자 리스트 호출

관리비 부과 페이지 – 관리비 부과목록 & 수정기능



세대정보		요금항목							부과작업
동	호	일반관리비	경비비	청소비	소독비	승강기유지비	전기료	수도료	
110	101	42,150	30,490	6,600	850	7,820	54,010	63,200	저장
	102	42,150	30,490	6,600	850	7,820	50,230	40,120	관리비조정
	103	42,150	30,490	6,600	850	7,820	77,420	65,890	관리비조정
	104	42,150	30,490	6,600	850	7,820	63,240	54,600	관리비조정
	105	42,150	30,490	6,600	850	7,820	29,560	32,350	관리비조정
	106	58,210	39,950	7,430	960	8,130	45,030	47,860	관리비조정
	107	58,210	39,950	7,430	960	8,130	56,340	68,900	관리비조정
	108	58,210	39,950	7,430	960	8,130	49,300	58,040	관리비조정
	109	58,210	39,950	7,430	960	8,130	33,060	25,640	관리비조정
	110	58,210	39,950	7,430	960	8,130	72,690	85,600	관리비조정

- 동 별로 호수와 관리비 나열
  - 동을 Group by 하여 옵션에서 선택한 동의 호수들만 보여줍니다.
- 관리비 내역 수정하기

관리비 부과 페이지 – 관리비 수정기능

주요코드 : 관리비 내역 수정

```
$("#modFeeSave"+householdSeq).on("click", function() {
    $("#feeUpdateContent").html($("#selectDong option:selected").val() +"동 " + ho + "호의 관리비 내역을 수정합니다.");
    $("#save").on("click", function () {

        $(".fee"+householdSeq).attr("readonly", true);
        $("#modFeeTd"+householdSeq).attr("hidden", false);
        $("#modFeeSaveTd"+householdSeq).attr("hidden", true);

        var feeItem = $(".fee"+householdSeq);
        //관리비 조정 버튼을 누르고, 입력한 수정 값을 담은 객체
        var fee = {     //숫자 세 자리수 마다 붙는 쉼표를 정규표현식으로 제거하여 Int형으로 입력
                generalBill : feeItem.eq(0).val().replace(/[^0-9]/g,''),
                securityBill : feeItem.eq(1).val().replace(/[^0-9]/g,''),
                cleaningBill : feeItem.eq(2).val().replace(/[^0-9]/g,''),
                fumigationBill : feeItem.eq(3).val().replace(/[^0-9]/g,''),
                elevatorBill : feeItem.eq(4).val().replace(/[^0-9]/g,''),
                electricityBill : feeItem.eq(5).val().replace(/[^0-9]/g,''),
                waterBill : feeItem.eq(6).val().replace(/[^0-9]/g,''),
                householdSeq : householdSeq,
                levySeq : data[0].feeList[0].levySeq
        };
        //관리비 업데이트 서비스 호출
            feeService.updateFee(fee, function() {
            });

            $("#UpdateCheck").modal("hide");
            $("#feeUpdateContent").html('');
    });

    });
```

```
function updateFee(fee, callback, error) {

    $.ajax({
        type : 'put',
        url : '/keeper/updateFee/',
        data : JSON.stringify(fee),
        contentType : "application/json; charset=utf-8",
        success : function(result, status, xhr) {
            if(callback){
                callback(result);
            }
        },
        error : function(xhr, status, er) {
            if(error){
                error();
            }
        }
    });
}
```

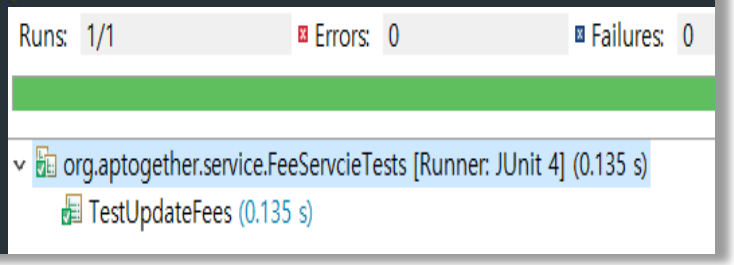
- fee 객체에 수정한 관리비를 key : value 로 담습니다.
  - 자바스크립트의 모듈패턴을 사용하여 fee객체를 JSON으로 변환 후 업데이트 서비스를 요청합니다.

관리비 부과 페이지 – 관리비 수정기능

Junit Test : 관리비 내역 수정

```
@Test
public void TestUpdateFees(){

    FeeVO fees = new FeeVO();
    fees.setFeeSeq(101);//관리비 번호
    fees.setLevySeq(81);//납부년월 번호
    fees.setHouseholdSeq(43);//가구 번호
    fees.setGeneralBill(23000);//일반관리비
    fees.setSecurityBill(21500);//경비비
    fees.setCleaningBill(14700);//청소비
    fees.setFumigationBill(350);//소독비
    fees.setElevatorBill(10200);//승강기유지비
    fees.setElectricityBill(46800);//전기료
    fees.setWaterBill(56000);//수도료

    service.updateFee(fees);
}
```

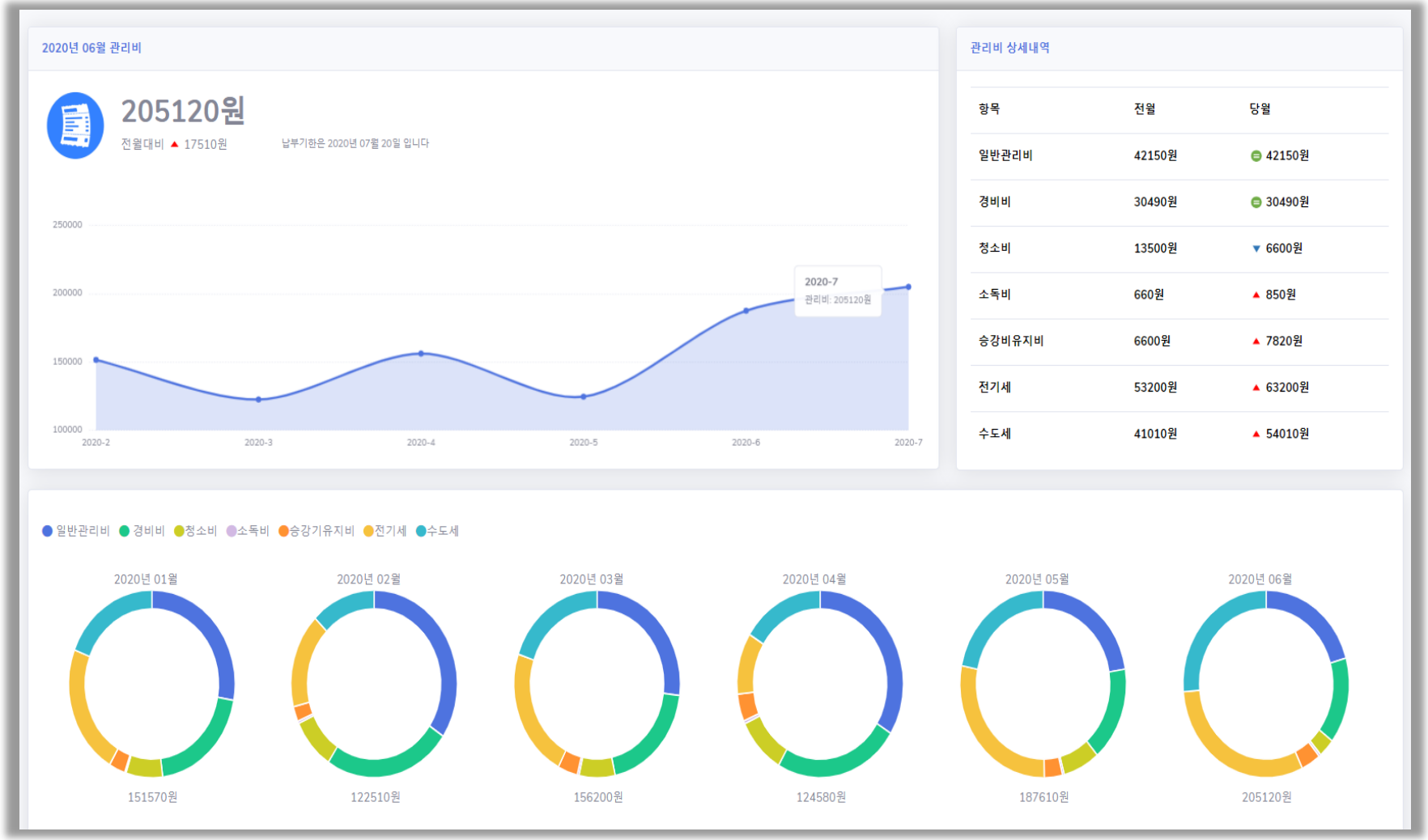
```
FeeServcieTests - 관리비 번호 : 101
FeeServcieTests - 납부년월 번호 : 81
FeeServcieTests - 가구 번호 : 43
FeeServcieTests - 일반관리비 : 23000
FeeServcieTests - 경비비 : 21500
FeeServcieTests - 청소비 : 14700
FeeServcieTests - 소독비 : 350
FeeServcieTests - 승강기유지비 : 10200
FeeServcieTests - 전기료 : 46800
FeeServcieTests - 수도료 : 56000
```



TestUpdateFees

└ 관리비 수정값 update 테스트

관리비 조회 페이지[입주민]



- 당월 관리비 내역 확인
  - 항목별 관리비와 총 금액
- 차트로 관리비 정보 제공
  - line 차트 : 최근 6개월 간의 관리비 총 금액 추이
  - pie 차트 : 최근 6개월의 간의 관리비 세부 항목 추이
- 전월대비 관리비 비교기능

관리비 조회 페이지[입주민]

주요코드 : 최근 6개월 간의 관리비 청구내역 조회

```java
@GetMapping("/tenant/dashboard")
public String tenantHome(Authentication auth, Model model) {
    CustomUser user = (CustomUser) auth.getPrincipal();
    //스프링 시큐리티가 입주민의 고유번호 인증값을 불러옴
    //Rownum은 6으로 설정하여 최근 6개월 간의 관리비 데이터를 조회
    List<TenantFeeInfoVO> tenantFeeInfoVO =  feeService.tenantFeeInfo(user.getMemberSeq(),6);
    model.addAttribute("tenantFeeInfo", tenantFeeInfoVO);

    return "userDashBoard";
}
```

```xml
<select id="tenantFeeInfo" resultType="org.aptogether.domain.TenantFeeInfoVO">
<![CDATA[
    SELECT * FROM(

                        SELECT h.apt_seq, m.member_seq, h.household_seq, h.dong, h.ho, m.name,
                               f.fee_seq, l.levy_seq, l.levy_date, l.deadline_date,
                               f.general_bill, f.security_bill, f.cleaning_bill, f.fumigation_bill,
                               f.elevator_bill, f.electricity_bill, f.water_bill

                        FROM  member m, household h, levy l, fee f

                        WHERE m.household_seq = h.household_seq
                        AND h.household_seq = f.household_seq
                        AND f.levy_seq = l.levy_Seq
                        AND m.member_seq = #{memberSeq}
                        ORDER BY levy_date desc
                        )
    WHERE ROWNUM <= #{rownum}
]]>
</select>
```

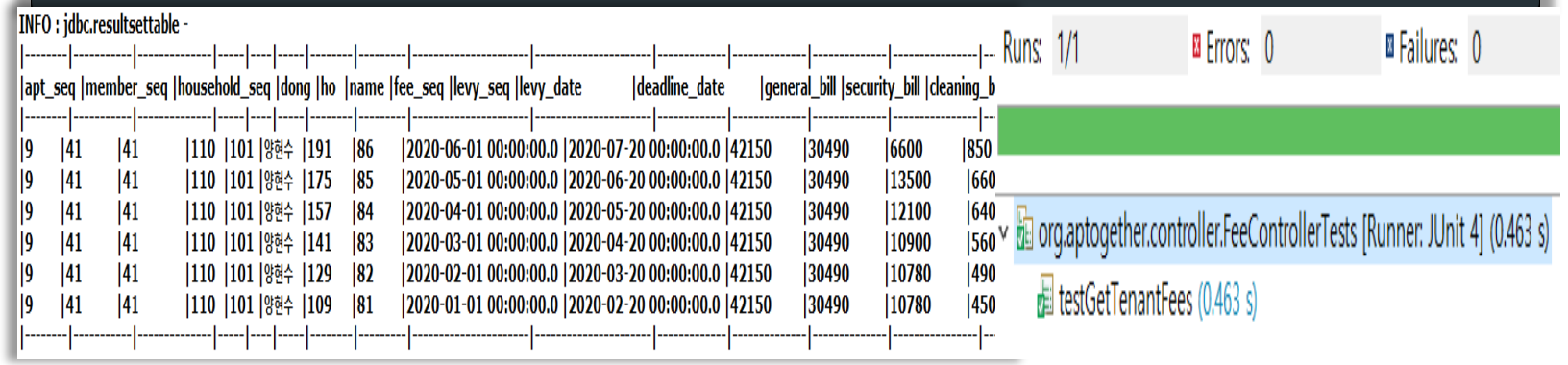
입주민에게 청구된 관리비 조회

- Levy(납부), Household(가구), Fee(관리비) 3개의 테이블 조인
- Rownum을 6으로 설정하여 최근 6개월 간의 관리비 목록을 호출합니다.

관리비 조회 페이지[입주민]

Junit Test : 최근 6개월 간의 관리비 청구내역 조회

```
@Test
public void testGetTenantFees() throws Exception{

    log.info(mockMvc.perform(MockMvcRequestBuilders.get("/tenant/dashboard"))
                                    .andReturn()
                                    .getModelAndView());

}
```



```
INFO : jdbc.resultsettable -
|-------|----------|------------|----|----|-----|-------|--------|-------------------|-------------------|-----------|------------|------------|-------------|--
|apt_seq |member_seq |household_seq |dong |ho  |name |fee_seq |levy_seq |levy_date         |deadline_date      |general_bill |security_bill |cleaning_b
|-------|----------|------------|----|----|-----|-------|--------|-------------------|-------------------|-----------|------------|------------|-------------|--
|9    |41     |41       |110 |101 |양현수 |191   |86     |2020-06-01 00:00:00.0 |2020-07-20 00:00:00.0 |42150     |30490      |6600       |850
|9    |41     |41       |110 |101 |양현수 |175   |85     |2020-05-01 00:00:00.0 |2020-06-20 00:00:00.0 |42150     |30490      |13500      |660
|9    |41     |41       |110 |101 |양현수 |157   |84     |2020-04-01 00:00:00.0 |2020-05-20 00:00:00.0 |42150     |30490      |12100      |640
|9    |41     |41       |110 |101 |양현수 |141   |83     |2020-03-01 00:00:00.0 |2020-04-20 00:00:00.0 |42150     |30490      |10900      |560
|9    |41     |41       |110 |101 |양현수 |129   |82     |2020-02-01 00:00:00.0 |2020-03-20 00:00:00.0 |42150     |30490      |10780      |490
|9    |41     |41       |110 |101 |양현수 |109   |81     |2020-01-01 00:00:00.0 |2020-02-20 00:00:00.0 |42150     |30490      |10780      |450
|-------|----------|------------|----|----|-----|-------|--------|-------------------|-------------------|-----------|------------|------------|-------------|--
```


- TestGetTenantFees
  - 3개 테이블의 조인 값과 6개월 분의 관리비 정보를 호출합니다.

공지사항 페이지

공지사항 글 목록

- 총 게시물 수를 나타내줍니다.
- 10개의 게시물 당 1페이지 씩 부가됩니다.

글 세부보기

- 클릭 시 조회수가 1씩 증가합니다.

공지사항 페이지

주요코드 : 공지 글목록과 상세보기 기능

```
//Keeper의 공지 목록을 불러오는 Controller
    @GetMapping("/listNoticeKeeper")
    public String list(NoticeCriteria cri,Authentication auth, Model model){
        CustomKeeper keeper = (CustomKeeper) auth.getPrincipal();
        int keeperAptSeq = keeper.getAptSeq();
        log.info("list : " + cri);
        log.info(keeperAptSeq);

    model.addAttribute("list", service.getList(cri, keeperAptSeq));
//  model.addAttribute("pageMaker", new NoticePageDTO(cri, 60));
    int total = service.getTotal(cri,keeperAptSeq);
    log.info("total: " + total);
    log.info(model);

    model.addAttribute("pageMaker", new NoticePageDTO(cri, total));
    return "/listNoticeKeeper";
    }
```

```
   @GetMapping("/getNoticeKeeper")
   public String get(@RequestParam("noticeSeq")int noticeSeq, @ModelAttribute("cri") NoticeCriteria cri,
                     Model model){

       log.info("/get");
       log.info(cri);

       model.addAttribute("notice", service.get(noticeSeq));
       service.plusCnt(noticeSeq); //조회수를 올려주기 위함

       return "/getNoticeKeeper";
   }
```

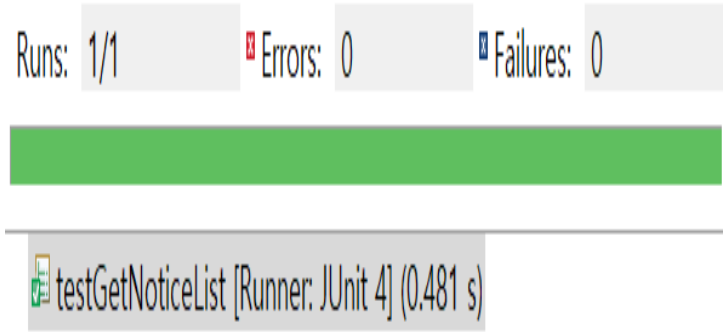
- 글 목록
  - 스프링 시큐리티 Authentication을 통해 로그인 된 관리자의 정보 조회
  - Paging처리와 함께 아파트 번호에 해당하는 공지 목록 조회
- 글 상세보기
  - 글 번호를 받아와 상세 조회

공지사항 페이지

Junit Test : 공지 글목록과 상세보기 기능

```
public void testGetNoticeList()throws Exception {
    log.info(mockMvc.perform(MockMvcRequestBuilders.get("/keeper/listNoticeKeeper"))
            .andReturn()
            .getModelAndView()
            .getModelMap());
}
```

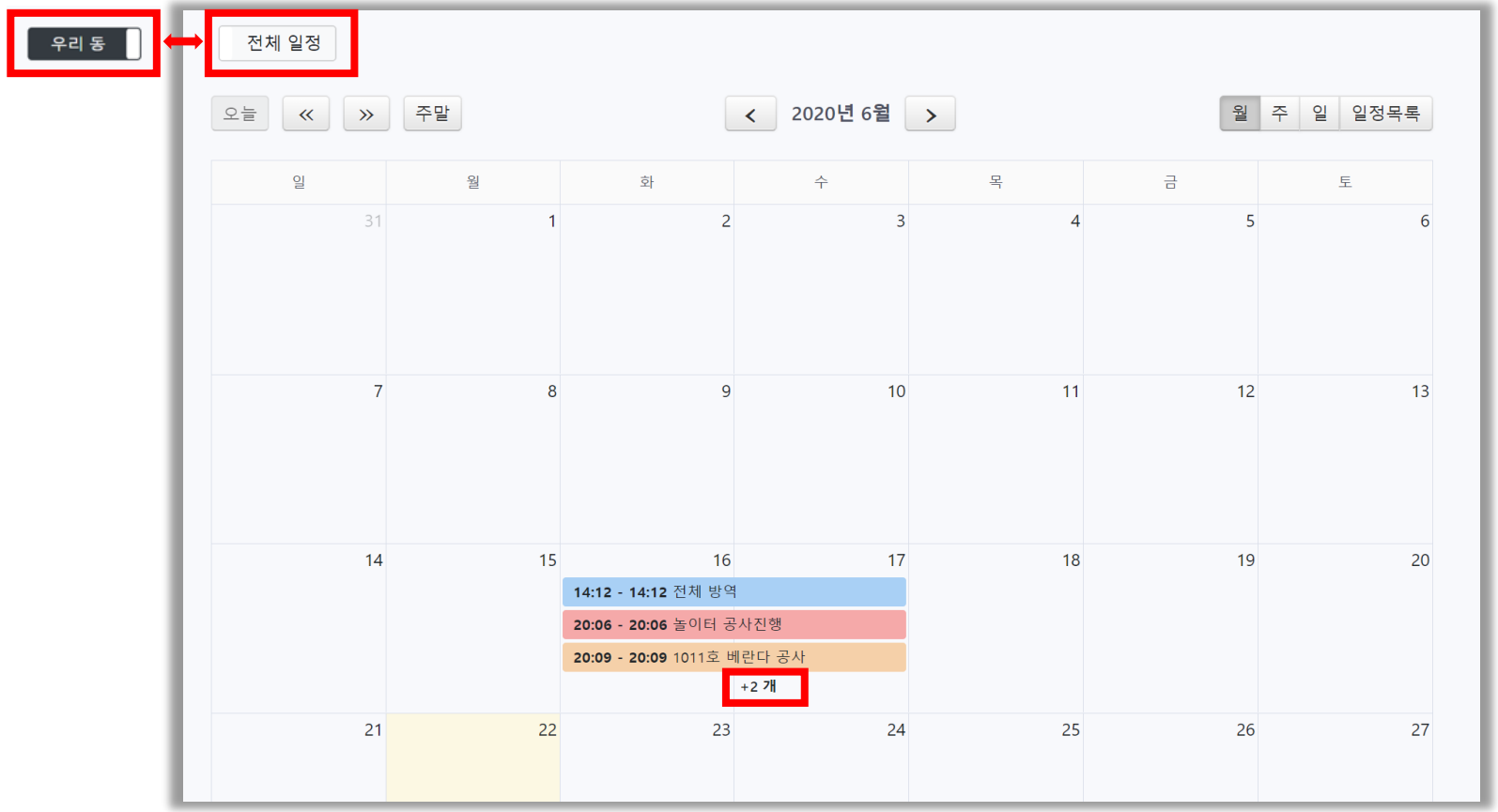
```
|----------|------------------|--------------------|
|notice_seq |title            |regdate             |
|----------|------------------|--------------------|
|1         |테스트 새로운 글 제목      |2020-06-22 19:23:00.0 |
|2         |테스트 새로운 글 제목      |2020-06-22 19:23:00.0 |
|3         |테스트 새로운 글 제목      |2020-06-22 19:23:00.0 |
|4         |놀이터 공사 진행          |2020-06-22 19:23:00.0 |
|5         |아파트 화단을 아름답게!     |2020-06-22 19:23:19.0 |
|6         |1010호 베란다 확장 공사 진행 |2020-06-22 19:25:14.0 |
|7         |유기견 보호 안내          |2020-06-22 19:28:40.0 |
|8         |세스코 설치합니다.         |2020-06-22 19:29:32.0 |
|9         |부녀회장 선거            |2020-06-22 19:30:23.0 |
|10        |층간 소음에 주의해주세요!    |2020-06-22 19:32:06.0 |
|----------|------------------|--------------------|
```



TestGetNoticeList

└ 공지사항 게시물 리스트를 가져옵니다.

일정 조회 필터 기능[입주민]



일정 조회 [토글 버튼]

- 토글 버튼을 통해 입주민의 일정 조회
- 입주민의 동 일정 ⇔ 전체 아파트 일정

일정 표기 방식

- 하나의 달력 행에 5개 이상의 일정이 등록되면 "+ n개" 로 표시됩니다.
- 색상 별로 일정의 종류를 표시합니다. (동 일정, 전체 일정, 주요일정)

일정조회 페이지[입주민]

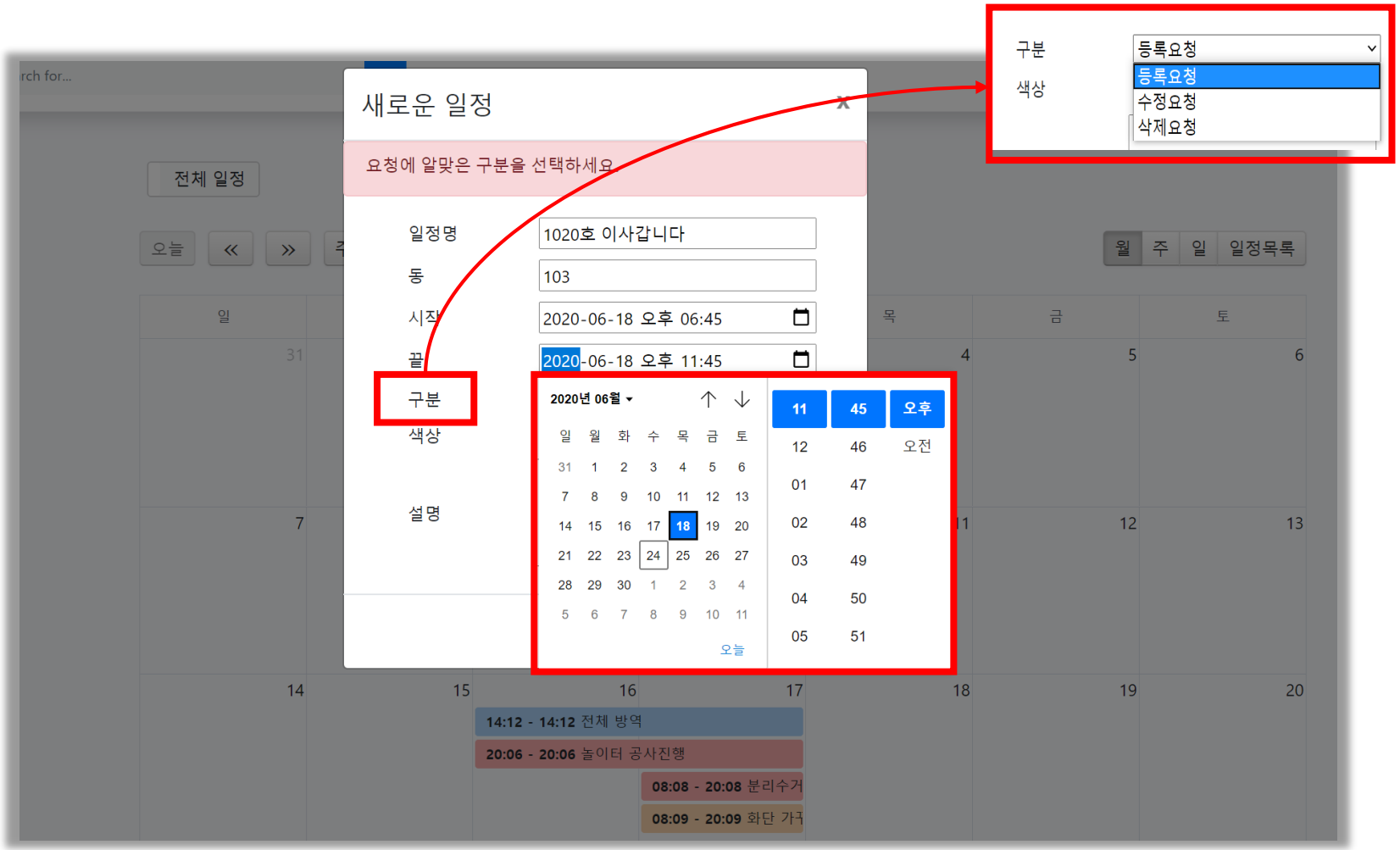
주요코드 : 토글 버튼을 사용하여 로그인 된 사용자의 동 정보 글만 조회

```
function filtering(event) {
    if($("#dong_toggle").prop('checked')){
        return (sessionDong == event.dong) ? true : false;
     }else{
        console.log("unclick");
        return true;
     }
}
```

입주민 동 정보와 같은 일정 조회

└ 삼항 연산자를 이용하여 입주민의
동 정보와 같은 글만 조회,
해제 시 아파트 전체 일정 조회

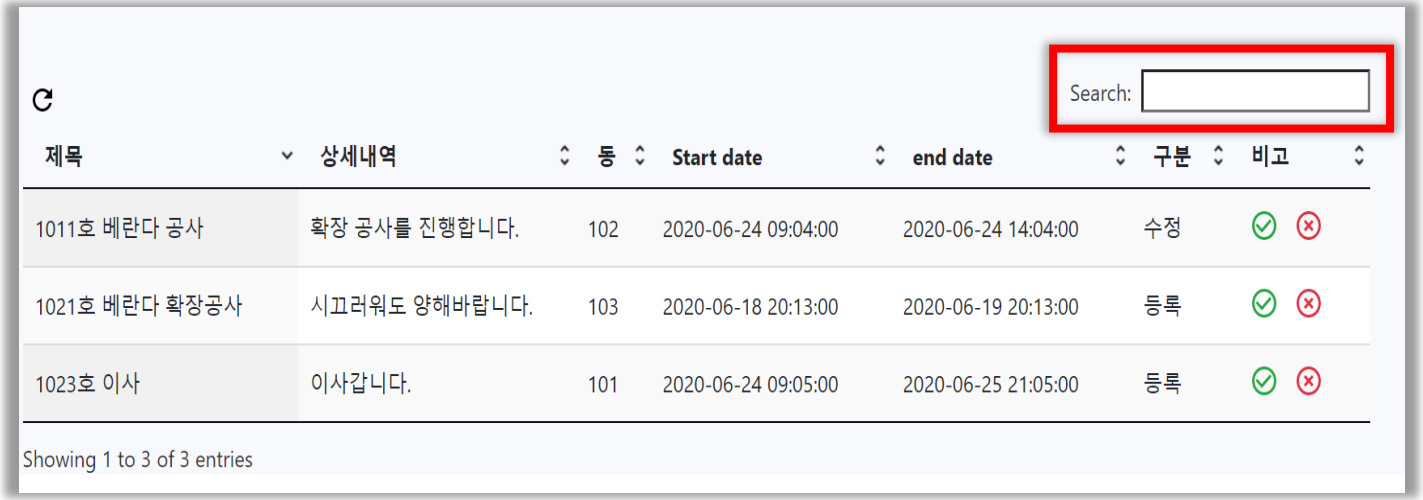
일정조회 페이지[입주민]



입주민 일정 수정 요청

- 달력을 클릭하면 Modal창을 통해 CRUD 작업을 합니다.
- 구분 미 선택 시 다음 단계로 진행 불가
- DataPicker를 사용하여 시작과 마감 일정 선택

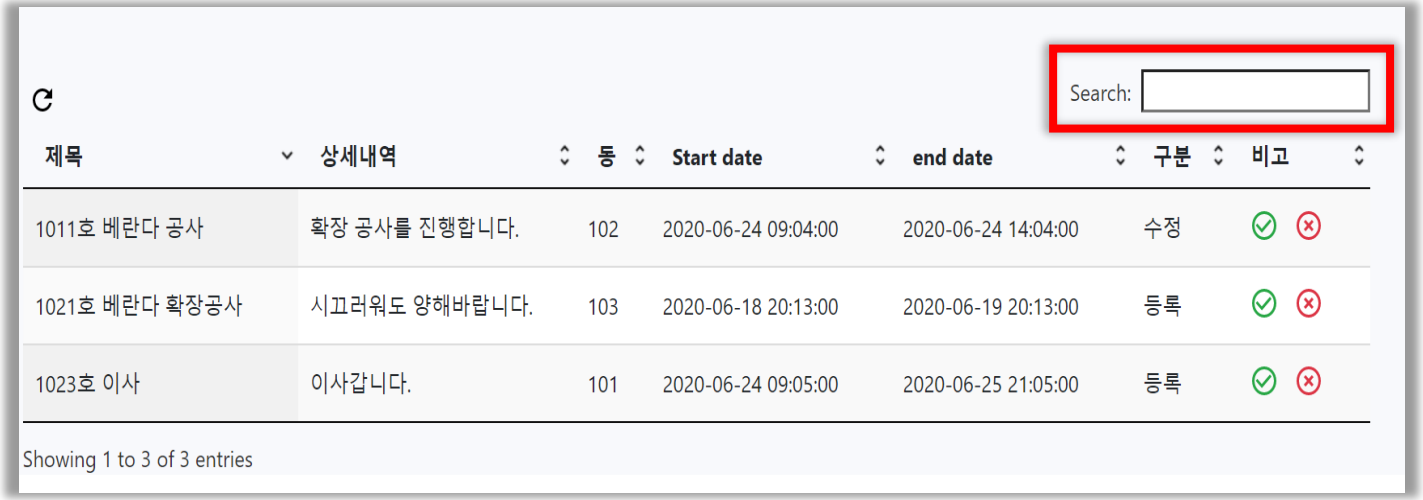
일정조회 페이지[입주민]



관리자 요청 테이블 조회

- 입주민이 요청한 일정은 테이블로 조회되어 검토 후 일정 등록이 됩니다.
- dataTable의 스크롤 페이징과 검색을 사용하여 관리자의 편리 고려

일정조회 페이지[입주민]



관리자 요청 테이블 조회

- 입주민이 요청한 일정은 테이블로 조회되어 검토 후 일정 등록이 됩니다.
- dataTable의 스크롤 페이징과 검색을 사용하여 관리자의 편리 고려

일정조회 페이지[입주민]

주요코드 : 일정 수정 요청 & 테이블 조회 기능

```
@RequestMapping(method={RequestMethod.PUT, RequestMethod.PATCH},
           value="/{scheduleSeq}",  consumes="application/json", produces={MediaType.APPLICATION_JSON_UTF8_VALUE})
    public String tenant_UpdateSchedule( @RequestBody ScheduleVO vo, @PathVariable("scheduleSeq")int scheduleSeq){
        log.info("schedule update seq :" + scheduleSeq);
        vo.setScheduleSeq(scheduleSeq);
        vo.setAuthority("0");
        log.info("schedule update : "+ vo);
        int modify = service.updateSchedule(vo);

        JsonObject obj = new JsonObject();
        Gson gson = new Gson();

        if(modify == 1 )
                obj.addProperty("status", "true");
        else
                obj.addProperty("status", "false");
        return gson.toJson(obj);
    }
```

```
@GetMapping("/scheduleKeeper")
    public String keeper_Schedule(Model model, Authentication auth){
    //승인 되지 않은 일정을 보여줍니다.
        log.info("admit list");

        CustomKeeper keeper = (CustomKeeper) auth.getPrincipal();
        System.out.println(keeper.getName());
        int aptSeq =    keeper.getAptSeq();
        System.out.println(aptSeq);

        model.addAttribute("list", service.listSchedule(aptSeq, "0"));
        return "scheduleKeeper";
    }
```

관리자 요청 테이블 조회

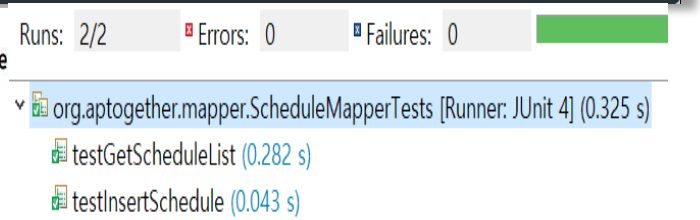
- 입주민이 요청한 일정은 테이블로 조회되어 검토 후 일정 등록이 됩니다.
- dataTable의 스크롤 페이징과 검색을 사용하여 관리자의 편리 고려

일정조회 페이지[관리자]

Junit Test : 일정 등록 테스트[관리자]

```java
@Test
public void testInsertSchedule(){

    IntStream.range(1,5).forEach(i->{
        ScheduleVO vo = new ScheduleVO();

        vo.setContents("test"+i);
        vo.setStartDate("2020/02/01 12:12");
        vo.setEndDate("2020/03/03 12:16");
        vo.setAptSeq(1);
        vo.setTitle("test title" + i);
        vo.setBackgroundColor("#wqe21");
        vo.setDong("201");
        vo.setAuthority("1");
        vo.setStates("공지");
        mapper.insertSchedule(vo);
    });

}
```

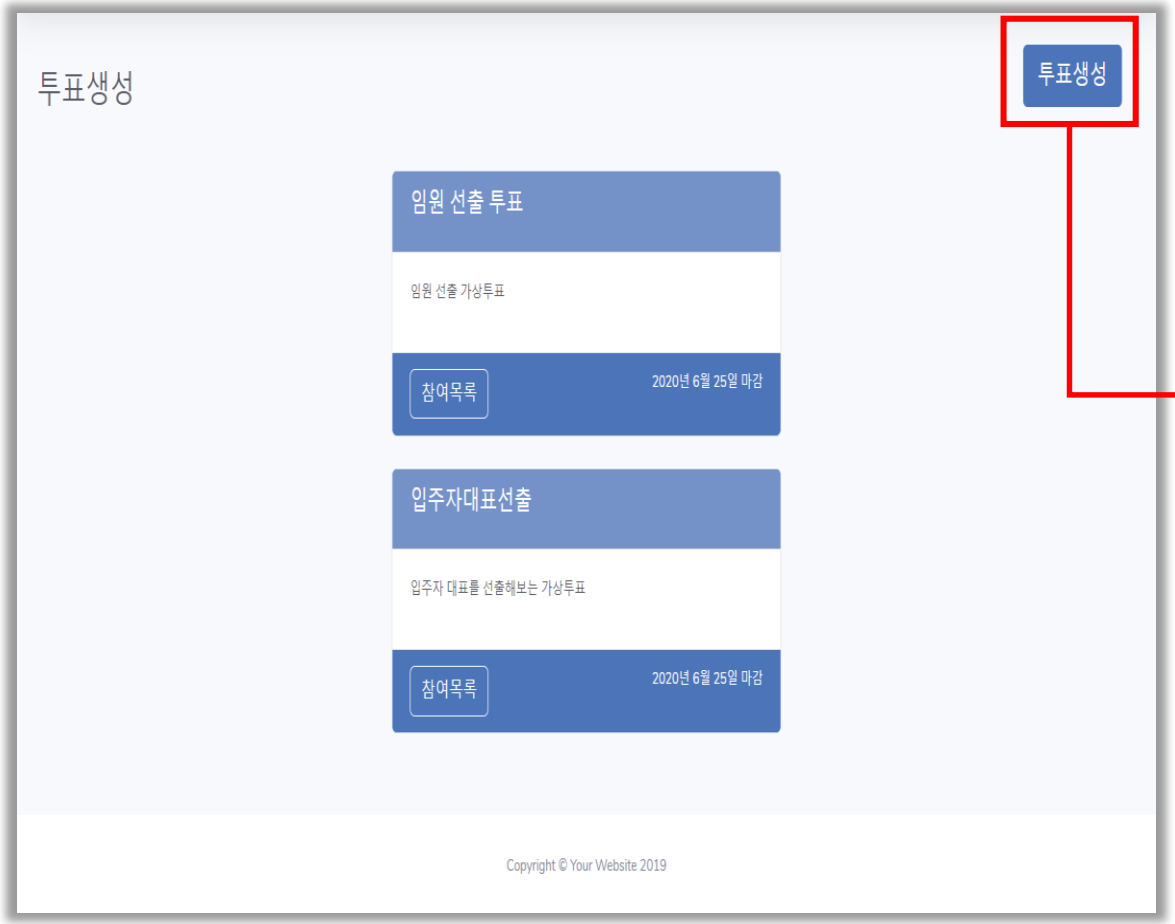
schedule_seq	contents	start_date	end_date	apt_seq	title	background_color	dong	authority	state
2	test1	2020-02-01 12:12:00	2020-03-03 12:16:00	1	test title1	#wqe21	201	1	공지
3	test2	2020-02-01 12:12:00	2020-03-03 12:16:00	1	test title2	#wqe21	201	1	공지
4	test3	2020-02-01 12:12:00	2020-03-03 12:16:00	1	test title3	#wqe21	201	1	공지
5	test4	2020-02-01 12:12:00	2020-03-03 12:16:00	1	test title4	#wqe21	201	1	공지

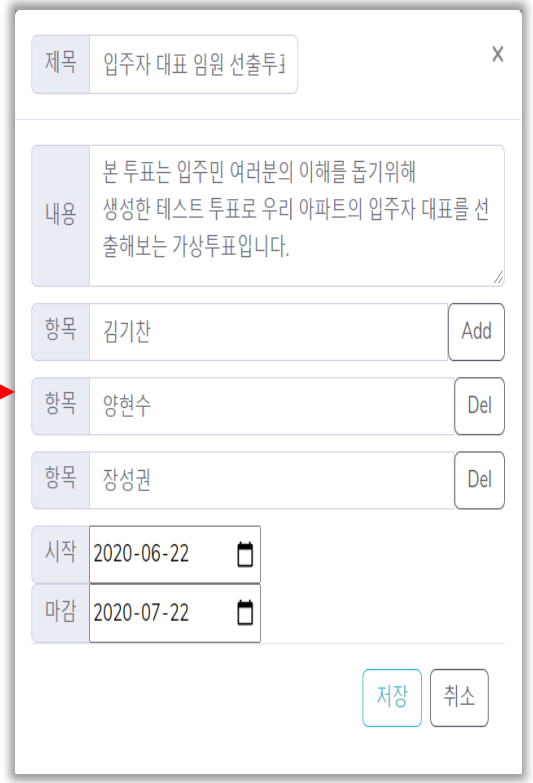


TestInsertSchedule

└ 원하는 날짜를 선택하고 일정을 추가

투표 등록 및 조회 페이지[관리자]





- 등록된 투표 조회
- 투표생성 버튼 클릭 후, 세부 내용을 입력하여 새로운 투표를 생성
- 항목의 Add 버튼과 Del 버튼으로 항목 추가, 제거

투표 등록 및 조회 페이지[관리자]

주요코드 : 투표 등록 & 조회 기능

```
@PostMapping("/pollInsert")
    public String keeper_insert(PollVO poll, RedirectAttributes rttr, Authentication auth) {
        CustomKeeper keeper = (CustomKeeper)auth.getPrincipal();
        int aptseq = keeper.getAptSeq();
        log.info("insert:" + poll);
        poll.setAptSeq(aptseq);

        service.PollInsert(poll);
        ArrayList<PollOptionVO> options = new ArrayList<>();
        for (int i = 0; i < poll.getOptions().size(); i++) {
            PollOptionVO option = new PollOptionVO();
            option.setOptionText(poll.getOptions().get(i));
            option.setPollSeq(poll.getPollSeq());
            options.add(option);
        }

        service.PollOptionInsert(options);

        rttr.addFlashAttribute("result", poll.getPollSeq());

        return "redirect:/keeper/pollList";

    }
```

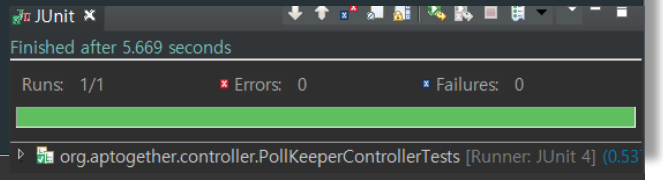
```
    @GetMapping("/pollList")
    public String keeper_list(Model model, Authentication auth) {
        CustomKeeper keeper = (CustomKeeper)auth.getPrincipal();
        int aptseq = keeper.getAptSeq();
        log.info("list");
        model.addAttribute("list", service.PollList(aptseq));

        return "pollList";
    }
```

- spring security 에서 로그인 한 관리자의 고유 인증값 을 가져옴
- 투표를 생성 할 때 입력한 항목을 함께 insert함

투표 등록 및 조회 페이지[관리자]

Junit Test : 아파트 번호로 투표목록 조회

```
@Test
    public void testpollList() throws Exception{
        log.info(
                mockMvc.perform(MockMvcRequestBuilders.get("/keeper/pollList"))
                .andReturn()
                .getModelAndView()
                .getModelMap());
    }
```

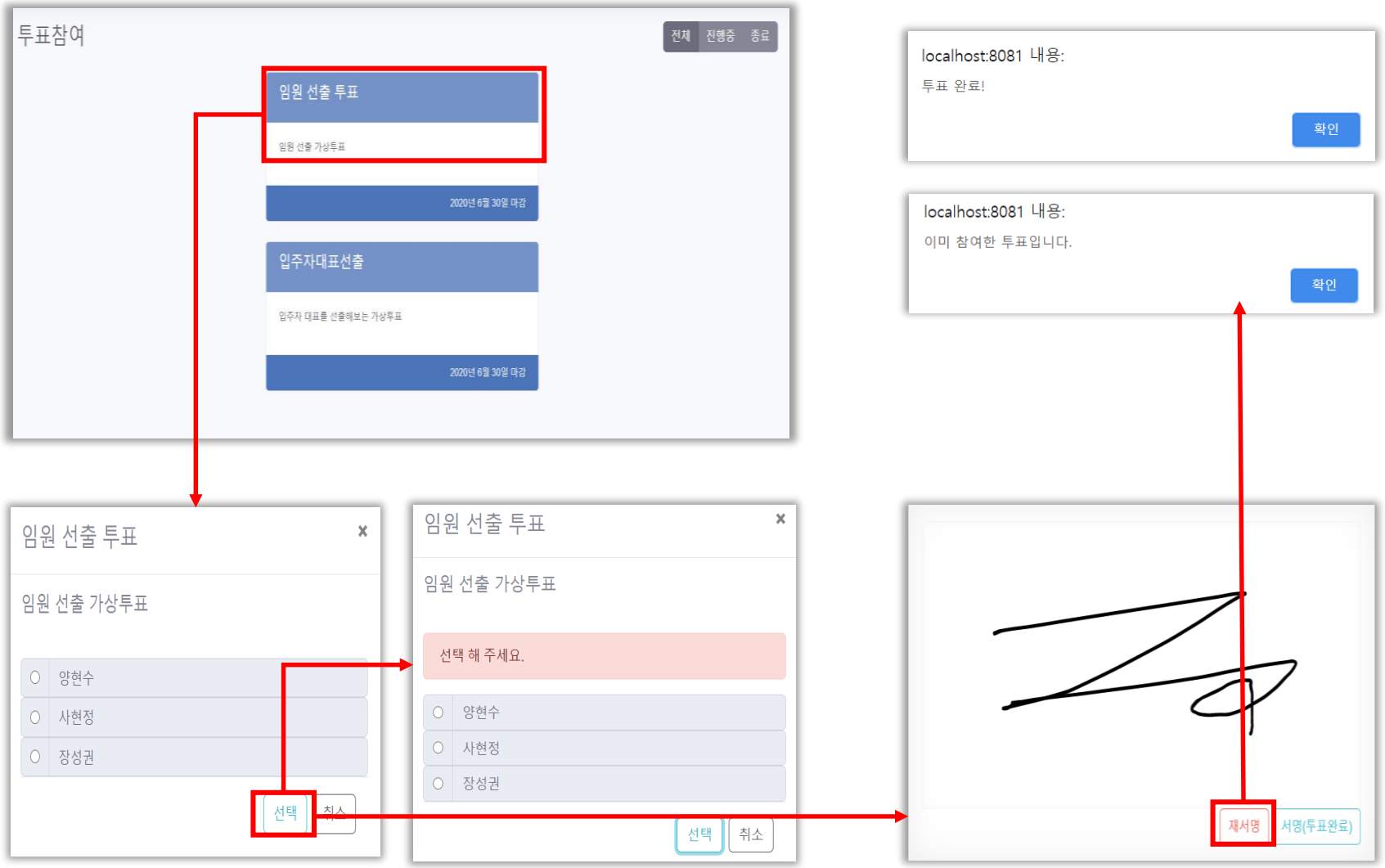



```
INFO : jdbc.resultsettable -
|---------|---------|-------------------|-------------------|---------|-------------------|--------|
|poll_seq |question |start_date         |end_date           |hitcount |contents           |apt_seq |
|---------|---------|-------------------|-------------------|---------|-------------------|--------|
|61       |임원 선출 투표 |2020-06-23 00:00:00 |2020-06-30 00:00:00 |0        |임원 선출 가상투표        |2       |
|60       |입주자대표선출  |2020-06-23 00:00:00 |2020-06-30 00:00:00 |0        |입주자 대표를 선출해보는 가상투표 |2       |
|---------|---------|-------------------|-------------------|---------|-------------------|--------|
```

TestPollList

└ 관리자의 아파트 번호로 목록 조회

투표 참여 및 조회 페이지[입주민]



- 투표 등록 순서를 기준으로 목록 출력
- 투표 클릭 시 항목 선택 모달 노출
- 항목 체크없이 선택 버튼 클릭 시 alert 발생
- 항목 선택 후 선택 버튼 클릭 시 서명 패드 노출
- 이미 참여한 투표에 서명 버튼 클릭 시 alert 발생

투표 참여 및 조회 페이지[입주민]

주요코드 : 투표 목록 조회 기능

```
@GetMapping("/pollTenantList")
    public String tenant_List(Model model, Authentication auth) {
        CustomUser user = (CustomUser) auth.getPrincipal();
        int aptseq = user.getAptSeq();
        log.info("list");
        model.addAttribute("list", service.PollOnList(aptseq));
        return "pollTenantList";
    }
```

```
@PostMapping(value = "/pollOptionList", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
    public ResponseEntity<List<PollOptionVO>> tenant_pollOptionList(
      @RequestBody Map<String, Object> seq) {
        log.info("option");
        int pollSeq = (int) seq.get("pollSeq");
        log.info(pollSeq);
        return new ResponseEntity<>(service.PollOptionList(pollSeq), HttpStatus.OK);
    }
```

- spring security 에서 로그인 한 입주민의 고유 인증값 을 가져옴
- 아파트번호로 투표 목록을 조회
- 참여하는 투표 번호로 DB에 저장된 항목을 불러옴

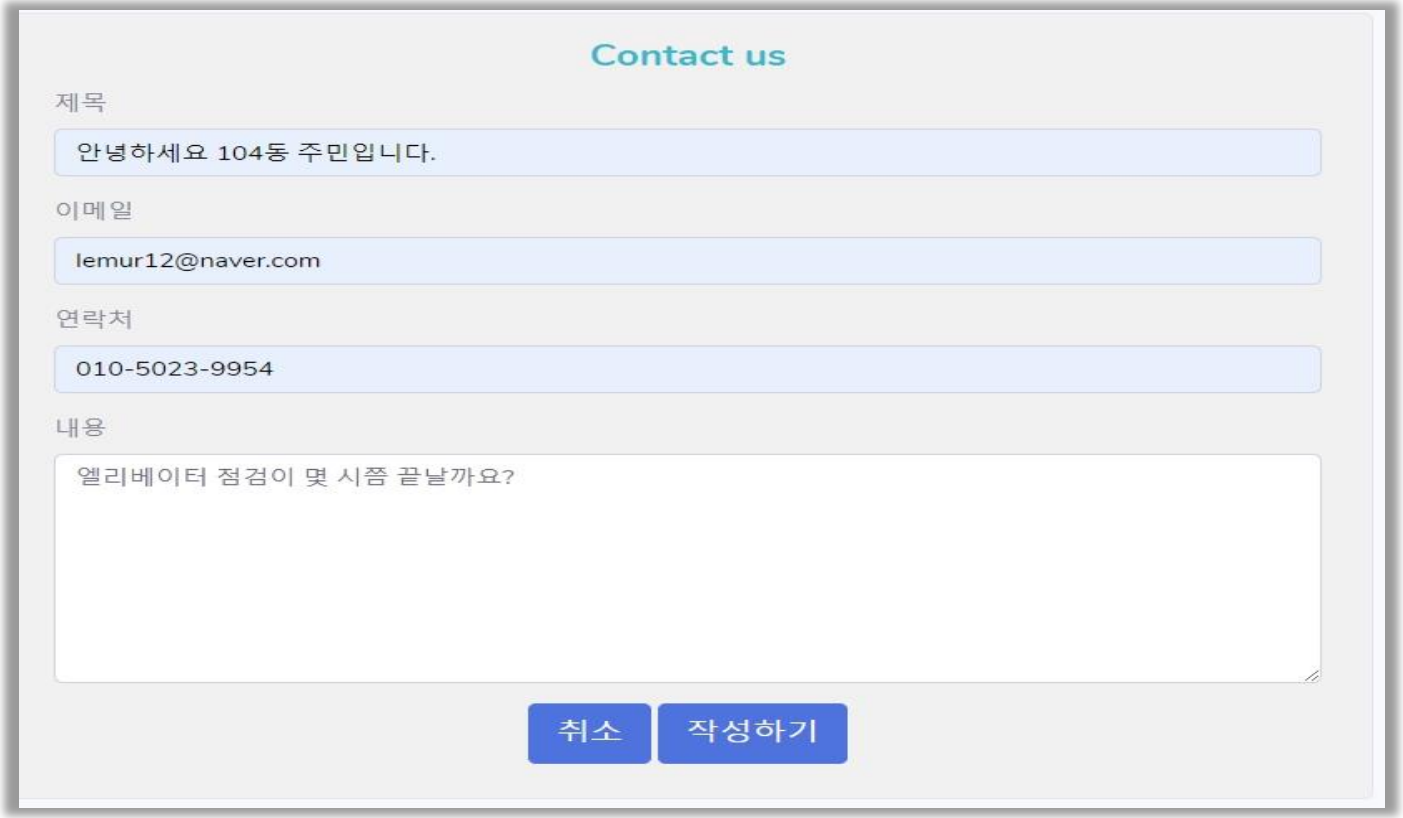
투표 참여 및 조회 페이지[입주민]

주요코드 : 투표 서명 & 중복 투표 확인 기능

```
@PostMapping(value = "/pollSignUpload", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
    public String tenant_signUpload(PollSelectVO select, PollLookupVO lookup, Authentication auth,
            @RequestParam(value = "file") MultipartFile[] file,
            @RequestParam("optionSeq") String optionSeq,
            @RequestParam("pollSeq") int pollSeq) throws Exception {
        CustomUser user = (CustomUser) auth.getPrincipal();
        int memberseq = user.getMemberSeq();
        int householdseq = user.getHouseholdSeq();
        Gson gson = new Gson();
        JsonObject obj = new JsonObject();
        select.setMemberSeq(memberseq);

        String uploadFolder = "C:\\upload\\sign";
        String s_uploadFolder = "C:\\upload\\s_sign";

        File uploadPath = new File(uploadFolder);
        File s_uploadPath = new File(s_uploadFolder);

        if (uploadPath.exists() == false) {
            uploadPath.mkdirs();
        }

        for (MultipartFile multipartFile : file) {
            String uploadFileName = optionSeq;
            UUID uuid = UUID.randomUUID();
            uploadFileName = uploadFileName + "_" + uuid.toString() + ".png";
            try {
                if (s_uploadPath.exists() == false) {
                    s_uploadPath.mkdirs();
                }
                File saveFile = new File(uploadPath, uploadFileName);
                multipartFile.transferTo(saveFile);

                FileOutputStream thumbnail = new FileOutputStream(
                  new File(s_uploadPath, "s_" + uploadFileName));
                Thumbnailator.createThumbnail(
                  multipartFile.getInputStream(), thumbnail, 100, 100);
                thumbnail.close();
            } catch (Exception e) {
                e.printStackTrace();
                obj.addProperty("status", "false");
                return gson.toJson(obj);
            }
            select.setOptionSeq(Integer.parseInt(optionSeq));
            select.setFileName("s_" + uploadFileName);
            lookup.setHouseholdSeq(householdseq);
            lookup.setPollSeq(pollSeq);
        }
        if (0 == service.PollSelectLookup(lookup)) {
            service.PollSelectInsert(select, lookup);
            obj.addProperty("status", "true");
            System.out.println("obj : " + obj);
            return gson.toJson(obj);
        } else {
            obj.addProperty("status", "false");
            System.out.println("obj : " + obj);
            return gson.toJson(obj);
        }
    }
```

- 서명 원본과 썸네일이 저장될 경로 폴더가 없으면 폴더를 생성
- UUID로 고유한 파일 이름 생성
- 이미 투표에 참여 했는지 조회 후 참여하지 않았다면 성공적으로 투표참여가 완료되고, 중복투표일시 alert 발생

민원 페이지



- 민원 종류 별로 선택
- 선택한 종류의 민원에 대한 글쓰기 모달창이 나옵니다.

민원 페이지

주요코드 : 민원 등록 & 관리자 답변 기능

```
//민원 작성
@PostMapping("/write")
public String tenantWrite(ComplaintVO vo, RedirectAttributes rttr) {

    log.info("==========================");

    log.info("register: " + vo);

    log.info("==========================");
    service.registerComplaint(vo);

    return "redirect:/tenant/complaint/getlist";
}
```

```
//답변 등록
@PostMapping(value = "/new", consumes = MediaType.APPLICATION_JSON_UTF8_VALUE, produces = {
        MediaType.TEXT_PLAIN_VALUE })
public ResponseEntity<String> create(@RequestBody Map<String, String> param) throws Exception {
    ComplaintKeeperVO keepervo = new ComplaintKeeperVO();
    ComplaintVO vo = new ComplaintVO();

    keepervo.setFeedbackContent(param.get("feedbackContent"));
    vo.setProcess("답변완료");
    int insertCount = service.writeFeedback(keepervo);
    if (insertCount == 1) {
        service.updateProcess(Integer.parseInt(param.get("boardNo")));

    }
    // log.info("Reply INSERT COUNT: " + insertCount);
    return insertCount == 1 ? new ResponseEntity<>(HttpStatus.OK)
            : new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
}
```

- 민원 등록이 끝나면
  민원 게시판으로 이동

- 관리자 답변 등록 기능
  - HTTP 요청 헤더가
    JSON 객체일때만 처리
  - 데이터 검색을 용이하기 위해
    Map 자료구조와 삼항연산자 사용

민원 페이지

주요코드 : 민원 게시판 이동

```
//나의 민원 이동
@GetMapping("/getlist")
public String tenantGetlist(ComplaintCriteria cri, Model model) {
    log.info("list: " + cri);
    model.addAttribute("list", service.getComplaintList(cri));
    int total = service.getTotalComplaint(cri);
    log.info("total:" + total);
    model.addAttribute("pageMaker", new ComplaintPageDTO(cri, total));
    return "tenantBoard";
}
```

```
// 관리자 나의민원 조회
@GetMapping("/getlist")
public String keeperGetlistWithPaging(ComplaintCriteria cri, ComplaintVO vo, Model model) throws
Exception {
    log.info("list: " + cri);
    model.addAttribute("list", service.getComplaintList(cri));
    int total = service.getTotalComp(cri);
    log.info("total:" + total);
    model.addAttribute("pageMaker", new ComplaintPageDTO(cri, total));
    return "keeperBoard";
}
```

민원 게시판 이동

└ 전체 게시글과 페이징 처리 로직이 담겨있는 DTO 객체를 통해 pageMaker란 이름으로 모델에 담아서 가져갑니다.

민원 페이지

- RowNum을 통한 오름차순 정렬
- 페이징 처리
- 자바스크립트를 사용한 페이지간 이동

민원 페이지

주요코드 : 페이징 처리 기능

```java
@Getter
@ToString
public class ComplaintPageDTO {
    private int startPage;
    private int endPage;
    private boolean prev, next;

    private int total;
    private ComplaintCriteria cri;

    public ComplaintPageDTO(ComplaintCriteria cri, int total){
        this.cri = cri;
        this.total = total;

        this.endPage = (int)(Math.ceil(cri.getPageNum()/10.0)) * 10;
        this.startPage = this.endPage - 9;

        int realEnd = (int)(Math.ceil((total * 1.0) / cri.getAmount()));

        if(realEnd < this.endPage) {
            this.endPage = realEnd;
        }
        this.prev = this.startPage > 1;
        this.next = this.endPage < realEnd;
    }
}
```

페이징 처리 기능

└ 다음 10개의 페이지 번호로 넘어가거나 다시 이전으로 돌아갈 수 있습니다.

민원 페이지

답변 미등록시

답변 등록시

- 답변 등록 or 미등록시 <c:if>를 사용한 영역 처리

민원 페이지

주요코드 : 민원 조회 기능

```
//상세 민원 조회
    @GetMapping("/get")
    public String tenantGet(@RequestParam("boardNo") Long boardNo, Model model) throws Exception {
        log.info("/get");
        model.addAttribute("board", service.readComplaint(boardNo));
        model.addAttribute("feedback", service.readFeedback(boardNo));

        return "tenantGet";
    }
```

```
// 상세 게시글 조회
    @GetMapping("/get")
    public String keeperGet(ComplaintKeeperVO vo, @RequestParam("boardNo") long boardNo,
            @RequestParam("feedbackRno") long feedbackRno, Model model) throws Exception {
        log.info("/get");

        model.addAttribute("board", service.readComplaint(boardNo));

        model.addAttribute("feedback", service.readFeedback(feedbackRno));

        return "keeperGet";

    }

}
```

민원 조회[입주민 / 관리자]

└ 민원 게시글과 관리자 답변을 모델에 담아서 가져갑니다.

중고거래 페이지

중고거래 게시판

- 입주민들의 중고거래 커뮤니티입니다.
- 중고물품의 정보와 사진을 업로드하여 중고품을 확인 할 수 있습니다.

중고거래 페이지

주요코드 : 중고물품 사진 파일 업로드 기능

```
<script type="text/javascript">
$(document).ready(function(){
    var filename='';
    $("#uploadBtn").on("click",function(e){
        var formData=new FormData();
        var inputFile=$("input[name='fnames']");
        console.log(inputFile)
        var files=inputFile[0].files;
        /* console.log(files); */
        for(var i=0;i<files.length;i++){
            /* console.log(files[i]); */
            formData.append("fnames",files[i]);
        }
        console.log("밑:"+formData.get("fnames"));
        $.ajax({
            url:'/tenant/uploadAjaxAction',
            processData:false,
            contentType:false,
            enctype:'multipart/form-data',
            data:formData,
            type:'POST',
            success:function(result){
                alert('upload');
            }
        });
```

```
@PostMapping("/uploadAjaxAction" )
public void uploadAjaxPost(MultipartHttpServletRequest request) {
    String uploadFolder = "C:\\upload";
   List<MultipartFile> fileList = request.getFiles("fnames");
for (MultipartFile multipartFile : fileList) {

log.info("-----------------------------------");
log.info("Upload File Name: " + multipartFile.getOriginalFilename());
log.info("Upload File Size: " + multipartFile.getSize());

String uploadFileName = multipartFile.getOriginalFilename();

uploadFileName = uploadFileName.substring(uploadFileName.lastIndexOf("\\") + 1);
log.info("only file name: " + uploadFileName);

File saveFile = new File(uploadFolder, uploadFileName);

try {
multipartFile.transferTo(saveFile);
} catch (Exception e) {
log.error(e.getMessage());
} // end catch

} // end for
```

MultipartHttpServletRequest

- 경로는 c드라이브의 upload폴더로 지정
- transferTo메소드를 이용하여 저장

중고거래 페이지

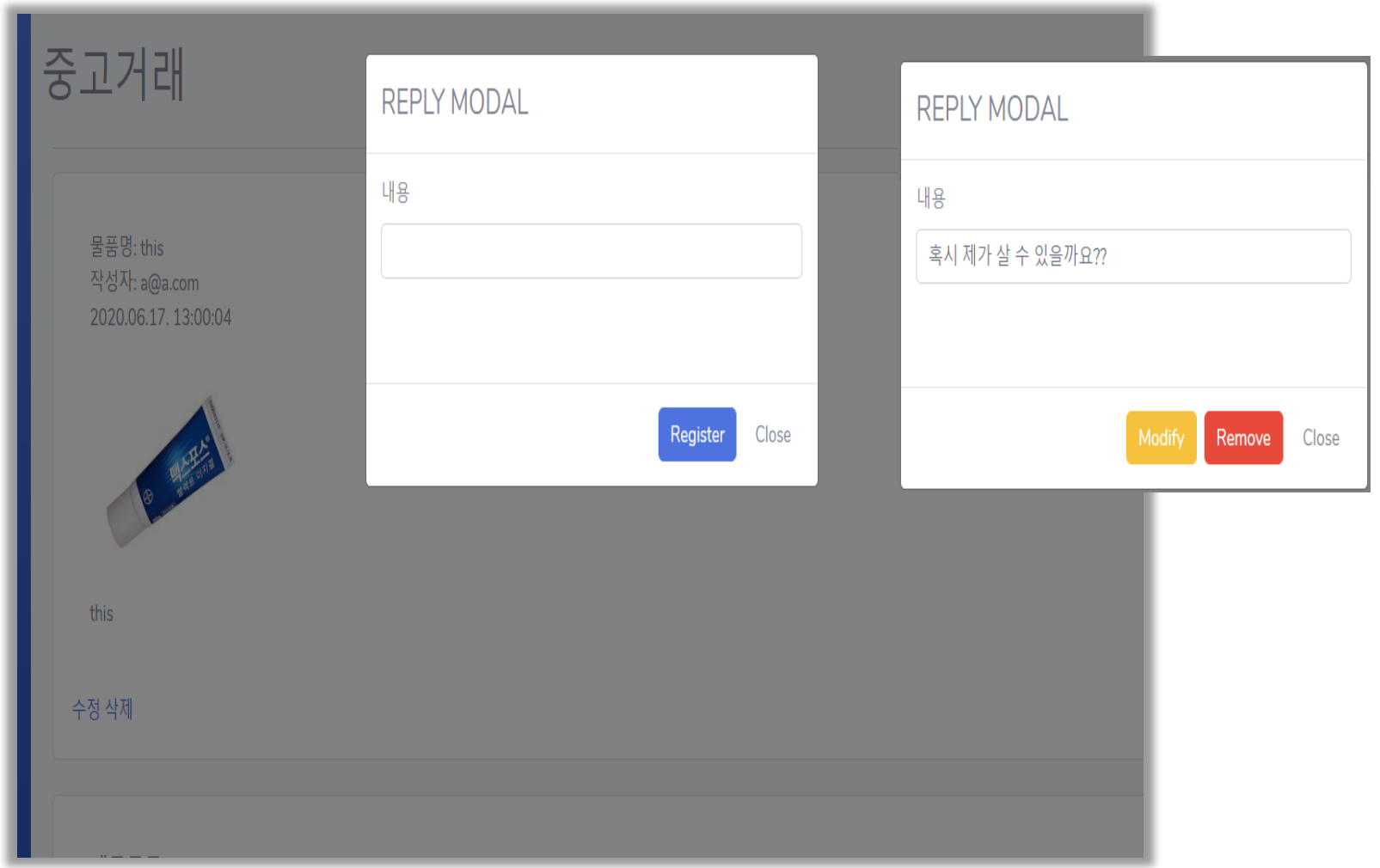
주요코드 : 중고물품 목록 조회 & 상세 보기 기능

```
<div class="card mb-4">
                    <div class="card-body">
                        <div id="columns">
                            <c:forEach var="board" items="${list}">
                                <figure>
                                <img src="<spring:url value='/tenant/download?
fileName=${board.fname}'/>">
                                    <a href="get?seq=${board.seq}">
                                        <figcaption>물품명:${board.productName}</figcaption>
                                    </a>
                                </figure>
                            </c:forEach>
```

```
<sec:authentication property="principal" var="principal" />
                        <sec:authorize access="isAuthenticated()">
                            <c:if test="${principal.username eq product.writer}">

                                <a href="modify?seq=${ product.seq }">수정</a>
                                <a href="remove?seq=${ product.seq }">삭제</a>
                            </c:if>
                        </sec:authorize>
                    </div>
```

- 마크업 언어로 중고물품 목록 조회
  - 태그 라이브러리 <c:forEach>
- 스프링 시큐어리티
  - 본인의 글만 수정,삭제 가능하도록 만든 view부분의 코드입니다.

중고거래 페이지



- 모달을 활용한 댓글달기 기능
  - 작성된 댓글의 수정, 삭제 기능

중고거래 페이지

주요코드 : 중고물품 댓글 달기 기능

```
@PostMapping(value="/new",consumes="application/json",produces={MediaType.TEXT_PLAIN_VALUE})
    public ResponseEntity<String> marketreplycreate(@RequestBody MarketReplyVO vo, Authentication auth)
{

        CustomUser user = (CustomUser) auth.getPrincipal();
        vo.setReplyWriter(user.getMemberSeq());
        int insertCount=service.ReplyRegister(vo);
        log.info("reply insert count:"+insertCount);
        return insertCount==1? new ResponseEntity<>("success",HttpStatus.OK):new ResponseEntity<>
(HttpStatus.INTERNAL_SERVER_ERROR);
    }
```

- 댓글 Insert 기능 컨트롤러
  - 삼항연산자를 사용하여 통신성공과 실패일 경우를 나누었습니다.

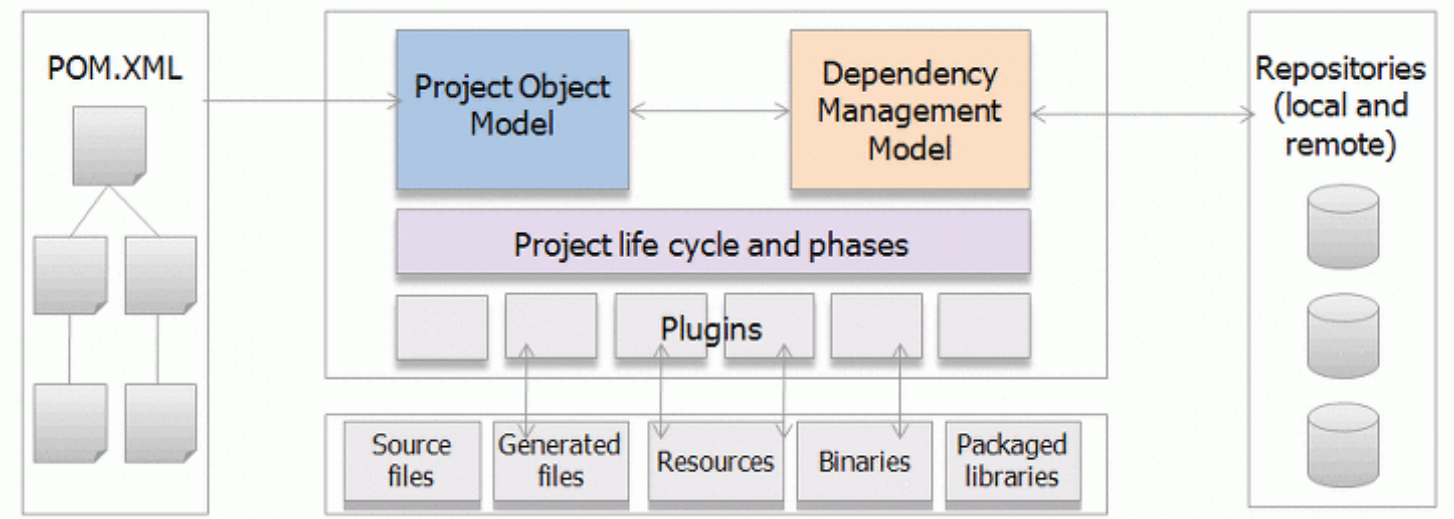
APTOGETHER

아파트 통합관리 웹 솔루션

05. 사용기술

사용 기술

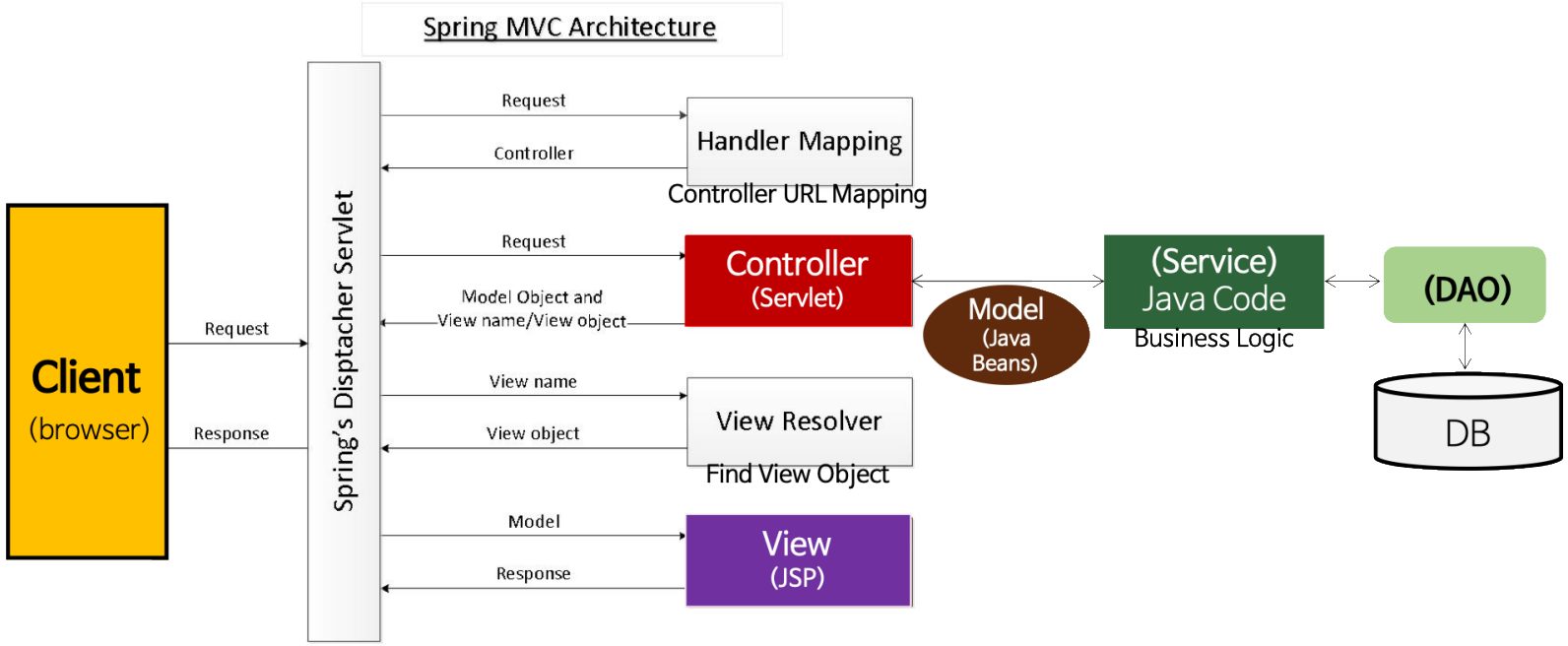
Maven™ : 주로 자바에서 프로젝트 빌드 및 관리를 하는 툴



프로젝트에서 Maven을 통해 다 같은 라이브러리 버전을 사용할 수 있었고,
라이브러리 PATH등을 따로 지정할 필요가 없어 협업 시에 정말 유용하게 사용할 수 있었습니다.

사용 기술

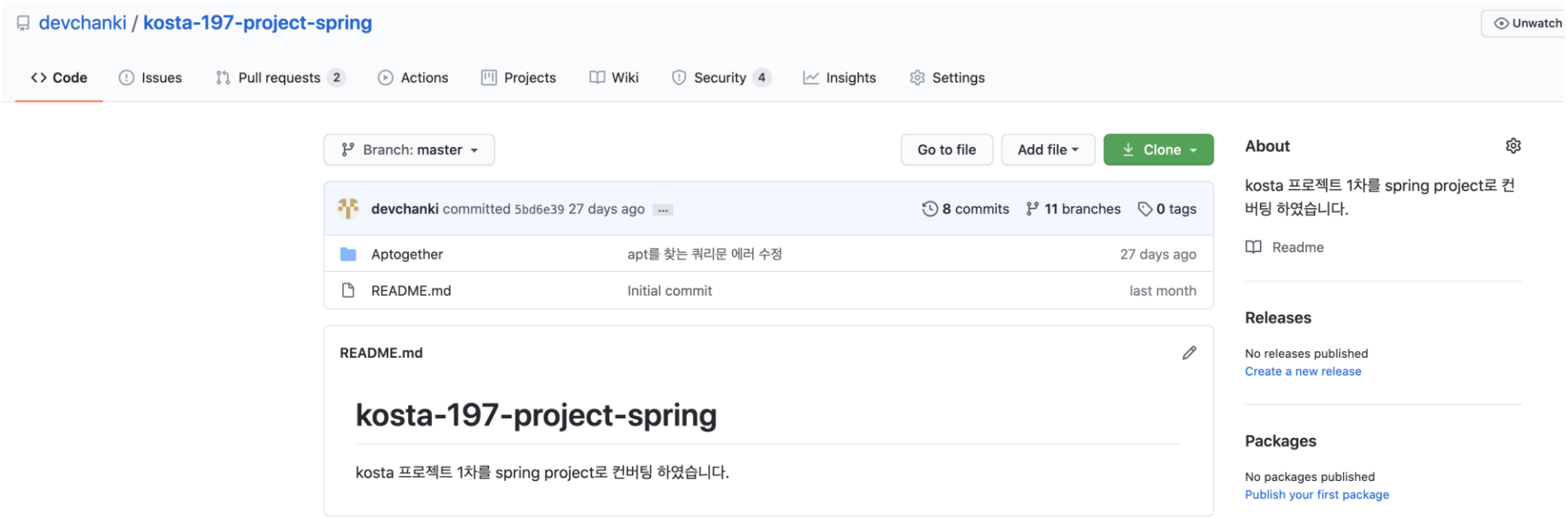
spring : 자바 플랫폼을 위한 오픈소스 애플리케이션 프레임 워크



스프링 MVC를 통해 웹사이트 개발을 하게 되었습니다.
스프링 MVC는 Model, View, Controller 를 분리한 디자인 패턴이 적용되어 있고,
비지니스 처리 로직과 뷰가 분리가 이루어져 있어
각각 분업이 편리해 개발에 용이하고 유지보수 시에 편리한 장점을 가지고 있습니다.

사용 기술

GitHub : 형상관리를 위한 도구, 분산형 관리 시스템으로 여러명이 동시에 작업이 가능



Git을 이용하여 소스코드의 형상관리를 하였고, Github을 통해서 소스코드를 공유를 통한 협업을 진행하였습니다.
Branch를 이용한 각각의 작업과, Branch들의 Merge를 통해서 여러가지 협업에 대한 기술들을 익히게 되었습니다.

사용 기술

: jQuery는 HTML 속 클라이언트 사이드 스크립트 언어를 단순화하도록 설계된 브라우저 호환성이 있는 자바스크립트 라이브러리

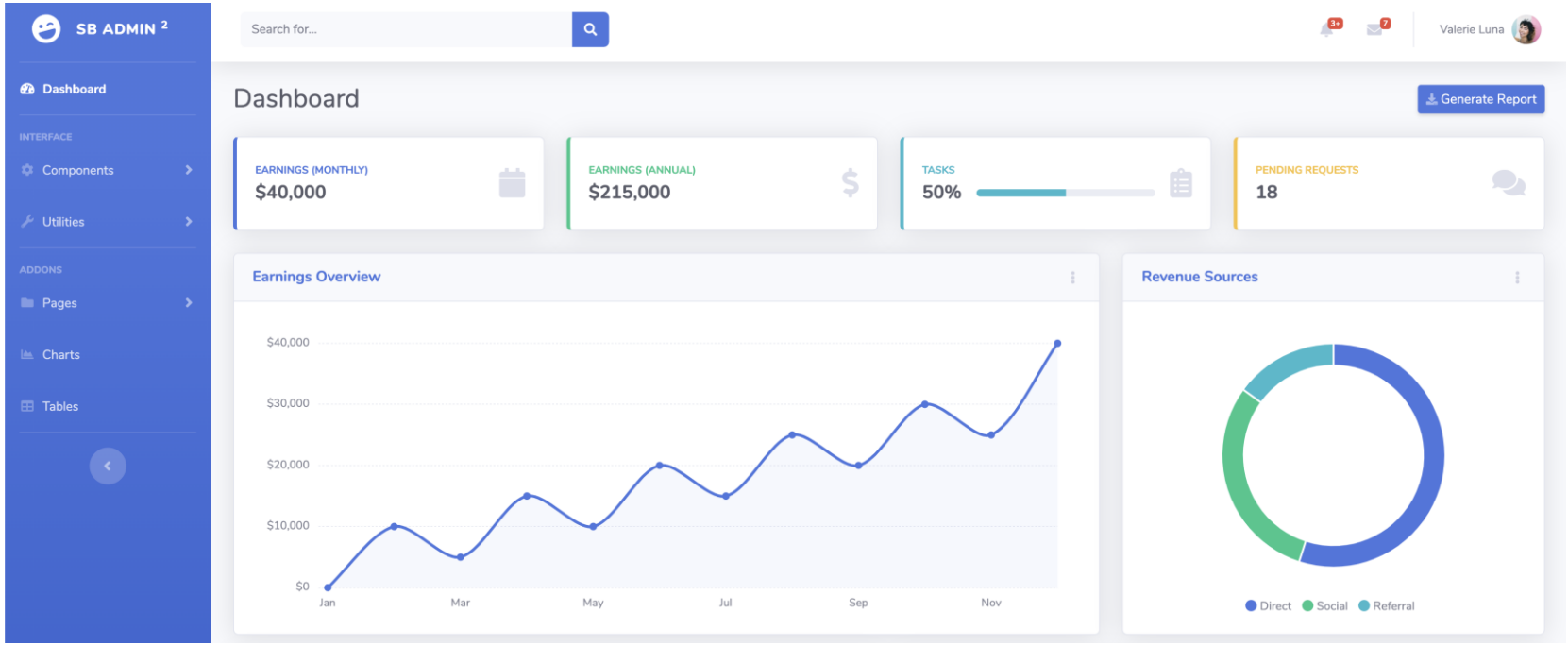
```
function doAction() {
    var tmpJson = {
        "kaptCode" : kaptCode,
        "aptName" : aptName,
        "location" : aptLocation,
        "x" : x,
        "y" : y
    };
    tmpJson = JSON.stringify(tmpJson);
    $.ajax({
        url : "/apt/aptInsert",
        type : "POST",
        dataType : "json",
        contentType: "application/json",
        data : tmpJson,
        success : function(data) {
            console.log(data);
            if (data.status == "true") {
                alert("입력에 성공하셨습니다.");
            } else if(data.status = "exist"){
                alert("이미 등록 된 아파트입니다.");
            } else{
                alert("잠시후 다시 시도해주세요.");
            }
        },
        error : function(request, status, error) {
            console.log(request, status, error);
            alert("요청에 실패하였습니다. 조금 있다 다시 요청해주세요.")
        }
    })
}
```

jQuery를 이용하여 돔을 쉽고 편하게 조작하고,
AJAX 등 비동기 통신을 할 때에도 유용하게 사용하였습니다.

CSS, 애니메이션, AJAX, 이벤트, 돔 조작 등의 기능들을
사용해 보았습니다.

사용 기술

Bootstrap : 레이아웃, 버튼, 입력창 등의 디자인을 CSS와 Javacript 로 만들어 놓은 웹 프론트엔드 프레임워크

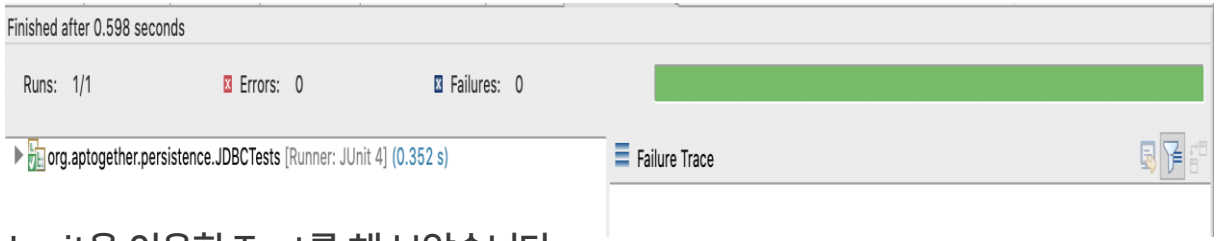


Sb Admin2 라는 부트스트랩 기반의 템플릿을 사용하였습니다.
개발 시간의 단축과 디자이너 없이 깔끔하고 멋진 뷰를 얻을 수 있었습니다.

사용 기술

JUnit : JUnit은 자바 프로그래밍 언어용 유닛 테스트 프레임워크

```java
package org.aptogether.persistence;

import static org.junit.Assert.fail;

@Log4j
public class JDBCTests {
    static{
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Test
    public void testConnection(){

        try (Connection con =
                DriverManager.getConnection(
                        "jdbc:oracle:thin:@localhost:1521:XE",
                        "aptogether",
                        "1234")){

                log.info(con);
        } catch (Exception e) {
            fail(e.getMessage());
        }
    }
}
```



Junit을 이용한 Test를 해 보았습니다.

뷰가 필요한 부분엔 Mock객체도 사용해 뷰 개발이 완성되지 않았을 때에 테스트도 해 보았고, 의도한 대로 코드가 잘 작동하는지 테스트를 위하여 사용해 보았습니다.

사용 기술

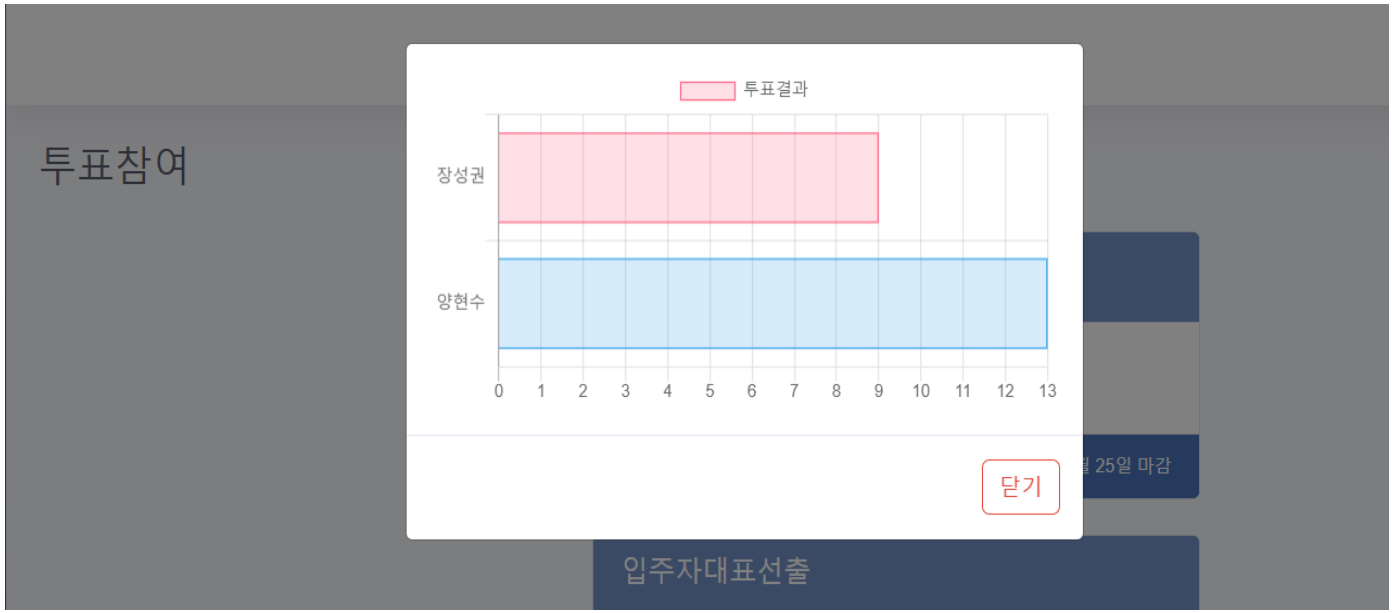
kakaomap : 카카오 맵을 이용한 지도 출력 및 여러 위치 정보 획득

도곡삼성래미안 근처 편의시설 입니다

카카오 맵 API를 이용해 아파트의 위치 정보의 위도 경도를 알 수 있었고,
지도를 통해서 아파트의 위치를 시각적으로 보여줄 뿐만 아니라 여러가지 부가적인 정보도 알 수 있었습니다.

사용 기술

Chart.js : 시각적으로 아름다운 차트를 그리기 위한 오픈소스 차트 라이브러리



숫자와 수치로 되어 있는 데이터들을 시각적으로 차트를 이용해서 보여주어 표현해 보았습니다.
AJAX와 JSON 데이터를 변형하여 차트를 그렸고, 차트를 보여주는 데에 유용한 라이브러리라고 생각합니다.

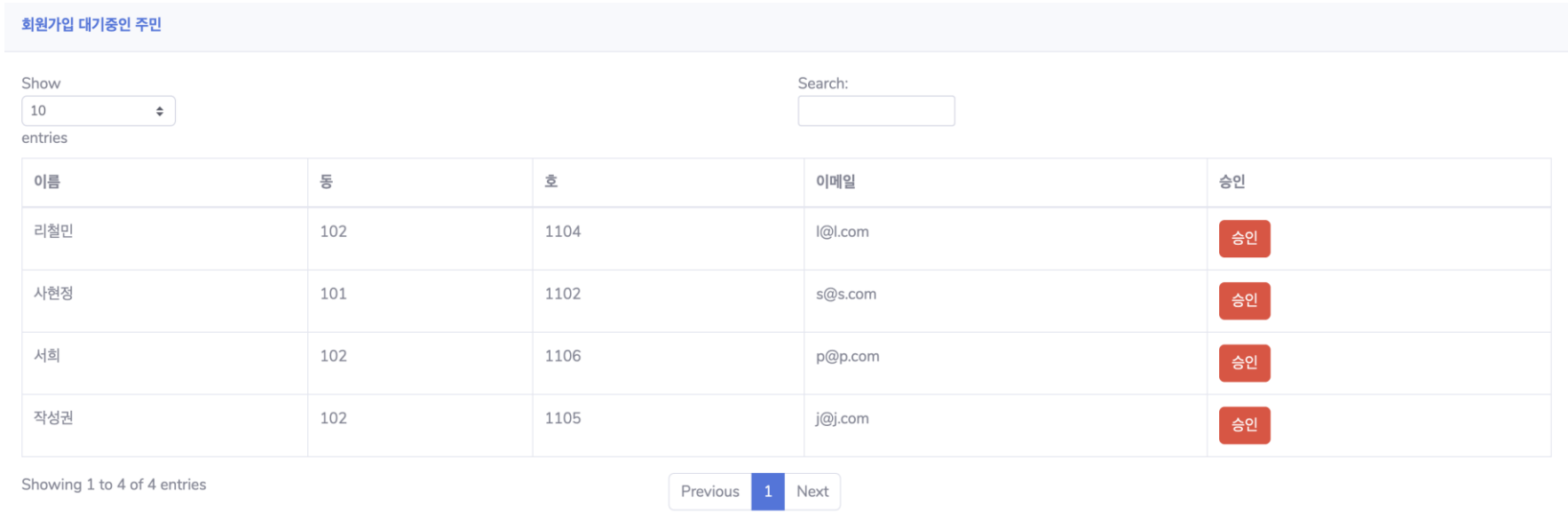
사용 기술

SIGNATURE PAD : 화면에서 마우스를 이용하여 그림을 그릴 수 있는 자바스크립트 라이브러리

전자 서명에 들어간 기능입니다. 서명을 프론트엔드에서 받게 되었고, 그 결과값은 BASE64 형태로 반환됩니다.
그 값을 서버에 보내서 서버에 이미지(png) 파일로 저장하여 보여주게 됩니다.

사용 기술

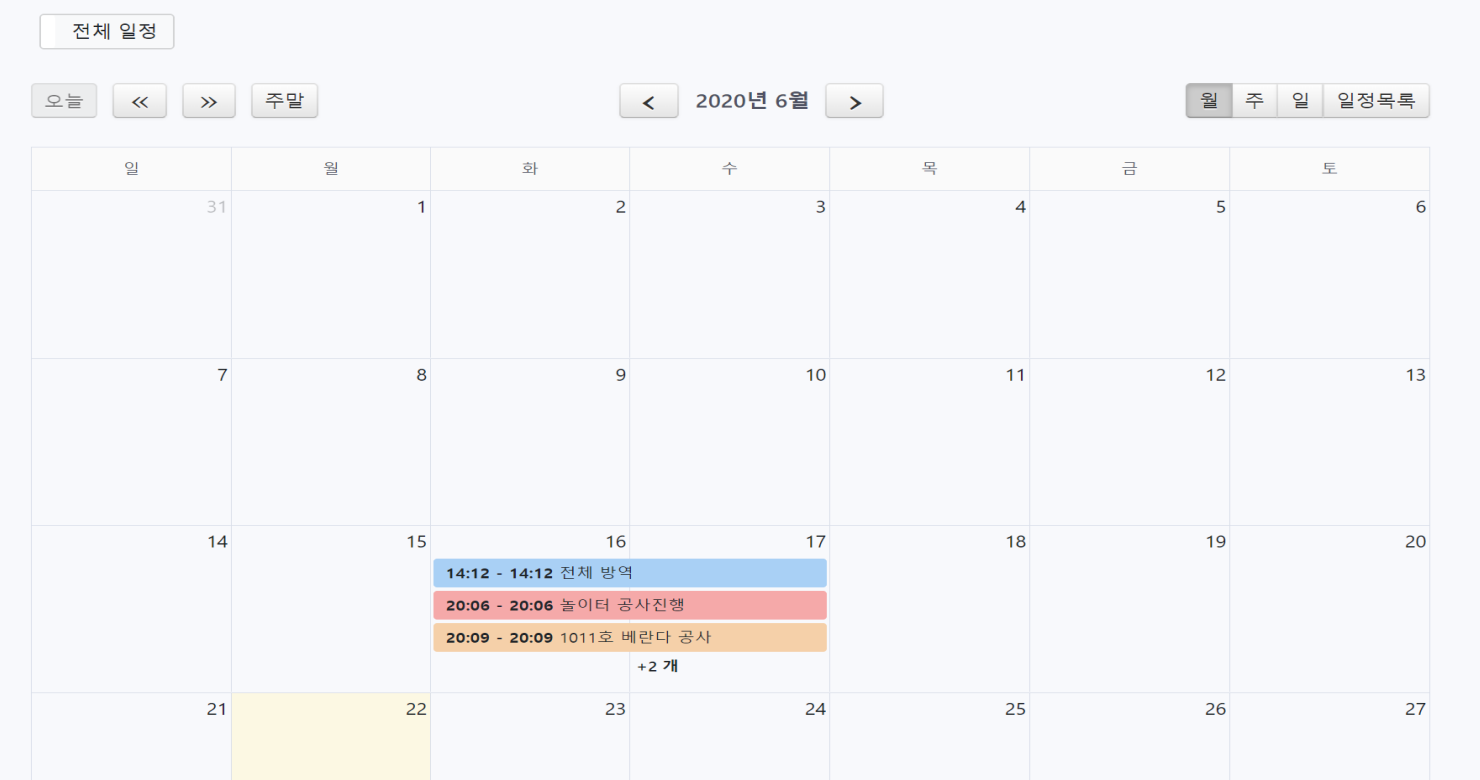
: HTML로 작성된 표를 다양한 기능을 가진 표로 만들어주는 라이브러리



Datatables 라이브러리를 사용하면 다음과 같이 검색, 페이징 처리등을 간단하게 처리할 수 있습니다.
필터, 검색, 표시 건수 제한 등 여러가지 기능들이 있고, 사용해 보았습니다.

사용 기술

FullCalendar : 시간 데이터들을 달력을 통해 그리고, 분류해주는 라이브러리



달력을 통해서 날짜에 이벤트들을 시각적으로 보여주고, 달력을 클릭하여 이벤트를 저장할 수 있게 만들었습니다.
FullCalendar을 이용하면 월, 주, 일 단위로 쉽게 보여줄 수 있고,
여러가지 색으로 시각적으로도 구분할 수 있게 만들어져 있습니다.

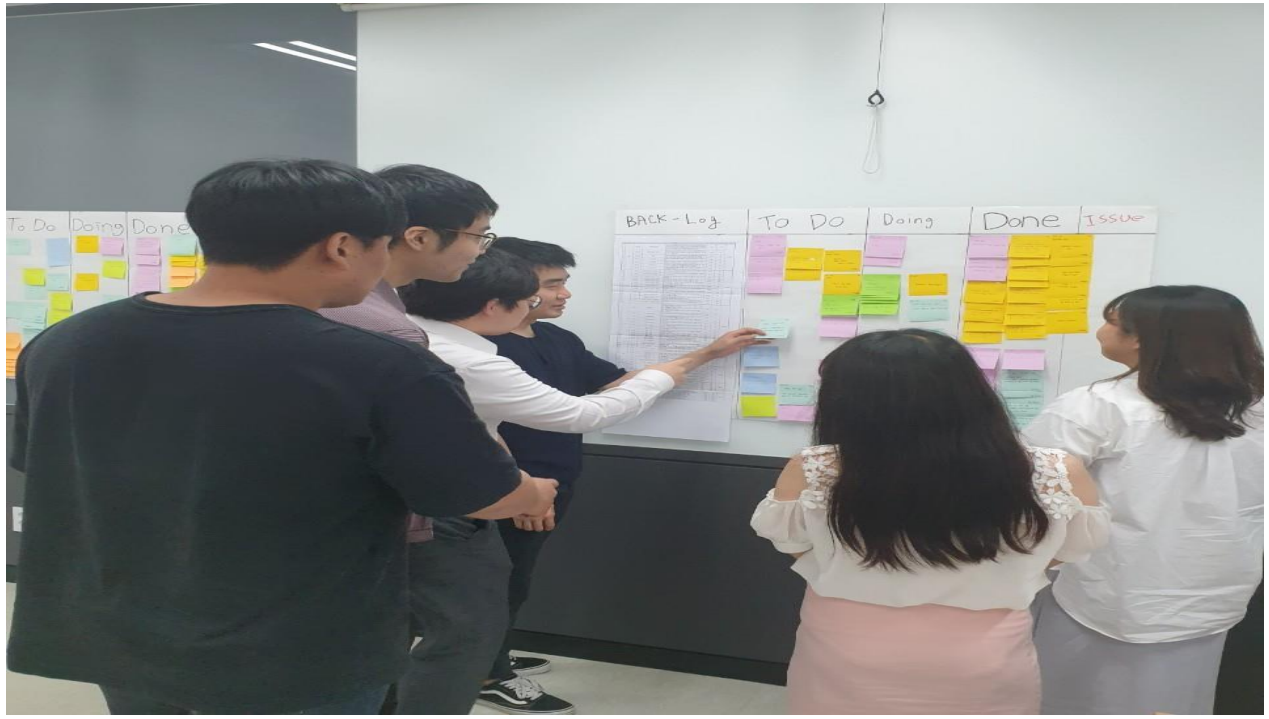
APTOGETHER

아파트 통합관리 웹 솔루션

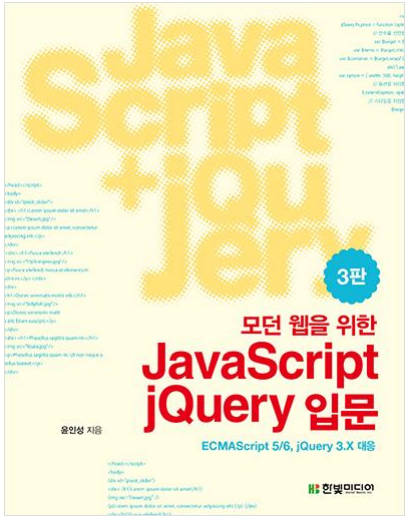
06. 비고

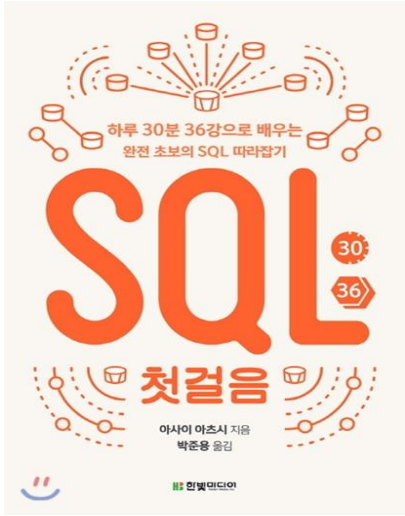
Agile Scrum

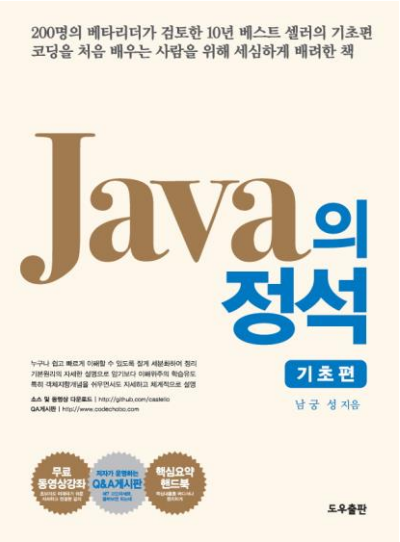
: 애자일 소프트웨어 공학 중 하나로,
프로젝트 관리를 위한 상호 점진적 개발 방법론이며
매일 정해진 시간/장소에서 스프린트를 반복합니다.



참고 도서







여러가지 프로젝트와 관련된 도서들을 참고해 가면서 기초 지식들을 복습하고, 이해하도록 노력하였습니다.
도서 뿐만 아니라 부트스트랩, 제이쿼리 등의 라이브러리 공식 문서도 읽어 가면서 프로젝트를 완성하였습니다.

팀원 소개

김기찬

- 조장
- 스프링 시큐리티 적용
- 아파트 근처 편의시설 조회 기능
- Git Master
- 전반적인 프론트 관리

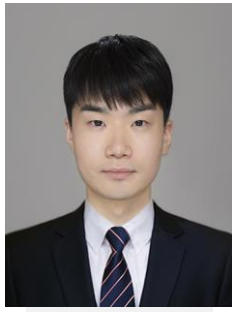
로그인 암호화, 유저 정보 관리 담당

이철민

- 투표 생성
- 전자 서명 기능
- 차트를 이용한 투표 결과 내역 조회
- Database 설계
- Sub Page CSS 담당

장성권

- 민원 기능
- Sub Page CSS 담당



양현수

- 관리비부과기능
- 관리비 차트 조회
- Database 설계
- Sub Page CSS 담당

박서희

- 중고장터 기능
- Sub Page CSS 담당

사현정

- 일정 관리
- 공지사항 관리
- Database 설계
- Sub Page CSS 담당