

습득력과 이해력이 빠른 개발자, 권순형입니다.

최종학력

서울시립대학교 전자전기컴퓨터공학부

자격증

SQLD

git

<https://github.com/SunHyongKwon>

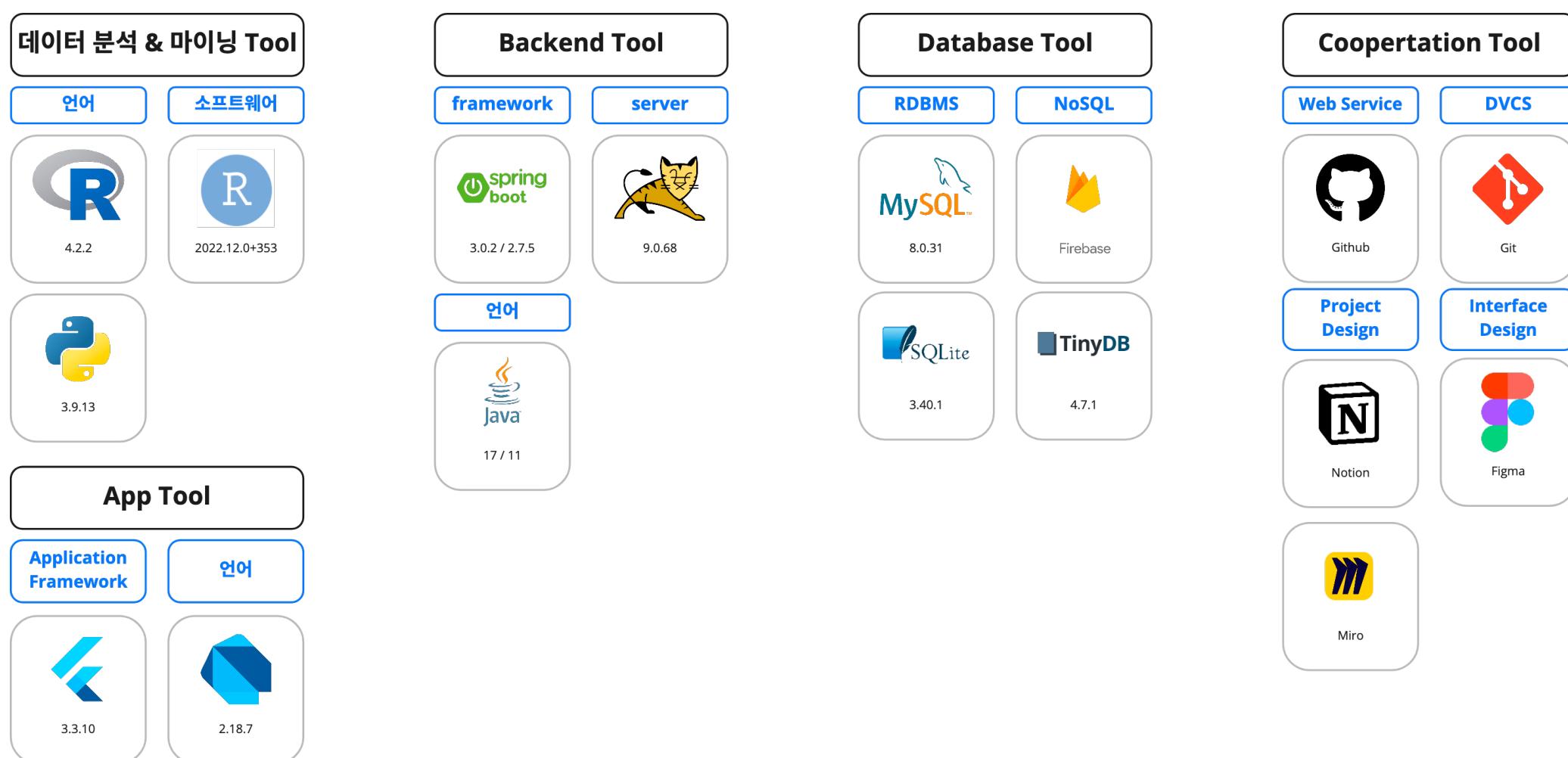
notion

<https://neuroo.notion.site/>

목 차

1	사용한 프로그램	3	6.2	구현 기능	24
2	데이터 분석 및 모델링	4	7	Open API	
	2.1 R: 서울 전세 보증금 데이터 분석 및 머신러닝 모델	4	7.1	Google Map API	16
3	Scraping	11	7.2	서울 열린 데이터 광장 open API	17
	3.1 주요 기능	12	7.3	Kakao Login API	25
	3.2 사용한 module	11	8	Data Modeling	
4	Flutter	14	8.1	ERD	29
	4.1 System Flow Diagram	15	8.2	DBMS	
	4.2 구현한 기능	16	8.2.1	MySQL	28
	4.3 사용한 Package	14	8.2.2	SQLite	19
5	Web	20	8.3	No SQL	
	5.1 MVC pattern	21			
	5.2 구현 기능	23			
	5.3 사용한 dependencies	20			
6	Backend	20			
	6.1 RestAPI	22			

사용한 프로그램



R : 서울 전세 보증금 예측 데이터 분석 및 머신러닝 모델링

목 차 |

1 사용한 데이터	4
2 서울시 데이터 EDA	5
3 서울시 구별 추가 데이터 EDA	6
4 동 별 예측으로 변경한 이유	8
5 예측범위 동으로 축소 후 EDA	9
6 Modeling(광장동)	10

사용한 데이터

- 서울시 부동산 전월세가 정보
(출처 : 서울 열린데이터 광장)
- 서울시 공원 통계
(출처 : 서울 열린데이터 광장)
- 서울시 공동주택 아파트 정보
(출처 : 서울 열린데이터 광장)
- 서울시 소음진동민원 현황 통계
(출처 : 서울 열린데이터 광장)
- 서울시 (안심이) CCTV 설치 현황
(출처 : 서울 열린데이터 광장)
- 광역버스정류장 인근 토지건물 특성정보
(출처 : 국가 교통데이터 오픈 마켓)
- 서울교통공사_자치구별지하철역정보
(출처 : 공공데이터포털)

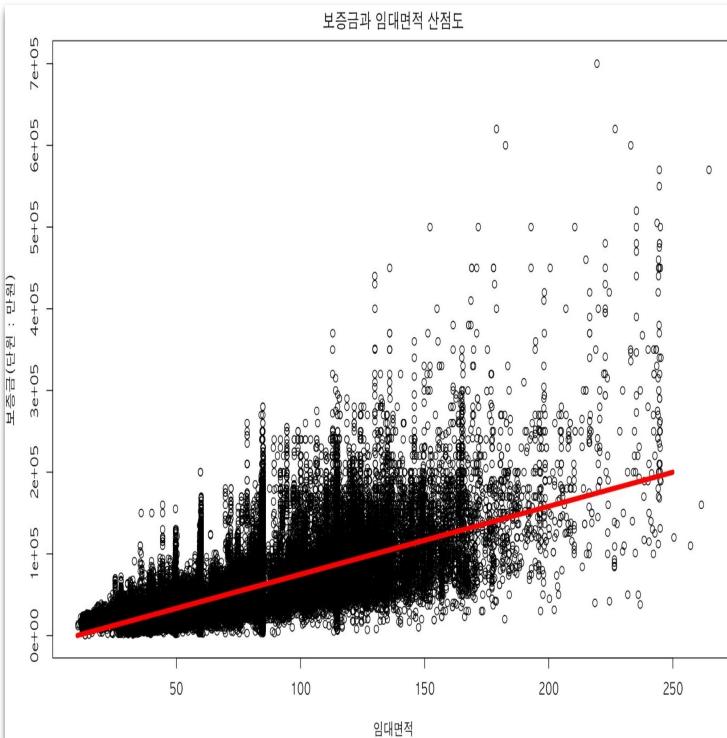
서울시 데이터 EDA

Feature Column과 Target column의 상관계수

	보증금	총	임대면적	계약계절
보증금	1.000	-	-	-
총	0.021	1.000	-	-
임대면적	0.646	0.0692	1.000	-
계약계절	0.176	0.201	-0.070	1.000

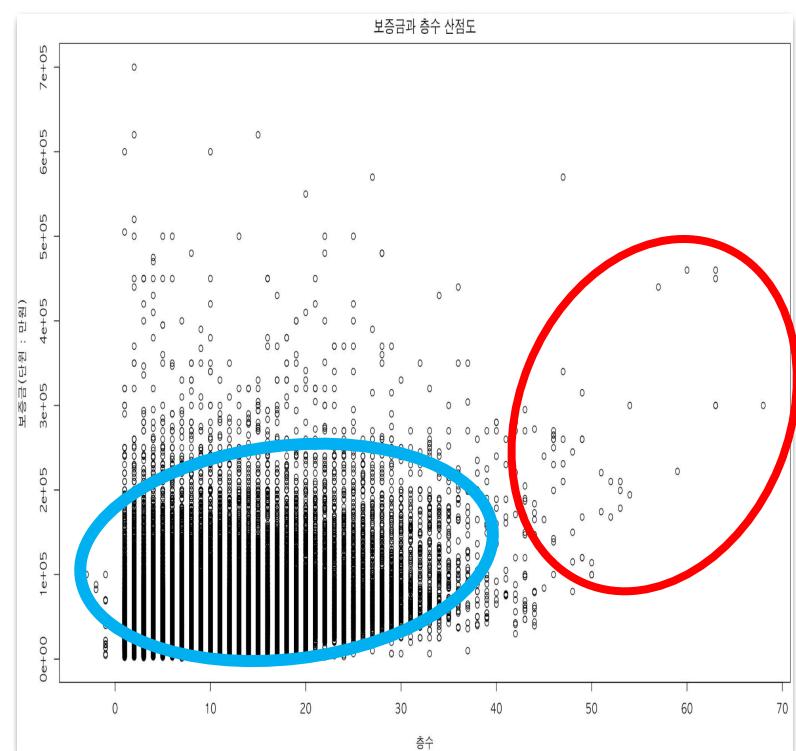
- Feature Column 과 Target Column 의 상관계수
 - **임대면적 > 총 > 건축년도**
- 순으로 상관계수가 높게 나왔다.

임대면적과 보증금 산점도



임대면적과 보증금의 상관계수가 높은 것에서 유추할 수 있듯이, **선형적인 관계**를 띠는 것을 확인 할 수 있었고, **feature column**으로 적당하다고 판단하였다.

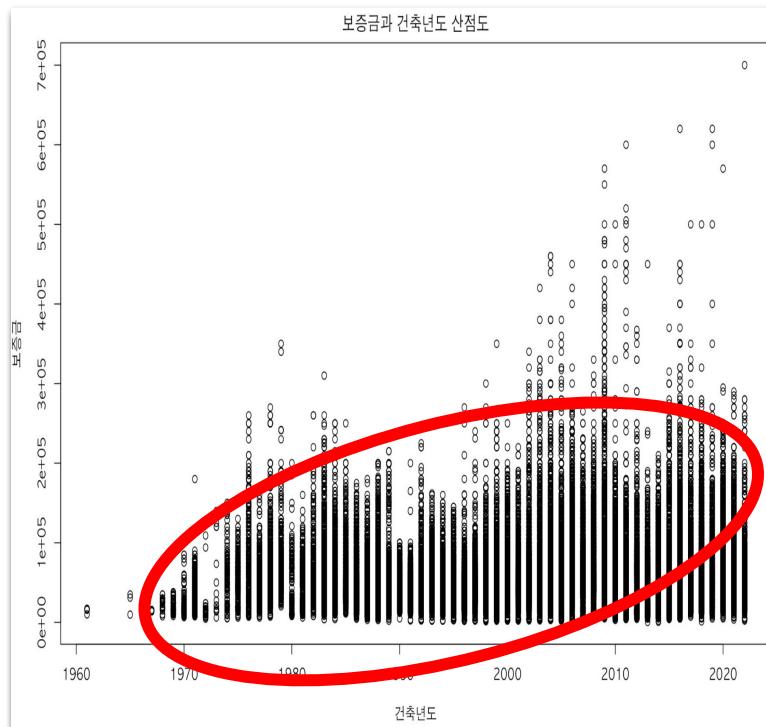
총수와 보증금 산점도



40층을 기준으로 낮은 총수와 높은 총수를 보았을 때 높은 총수의 분포가 보증금이 높을 확률을 보였다. 그렇기에 **feature column**으로 적합하다고 판단하였다.

서울시 데이터 EDA

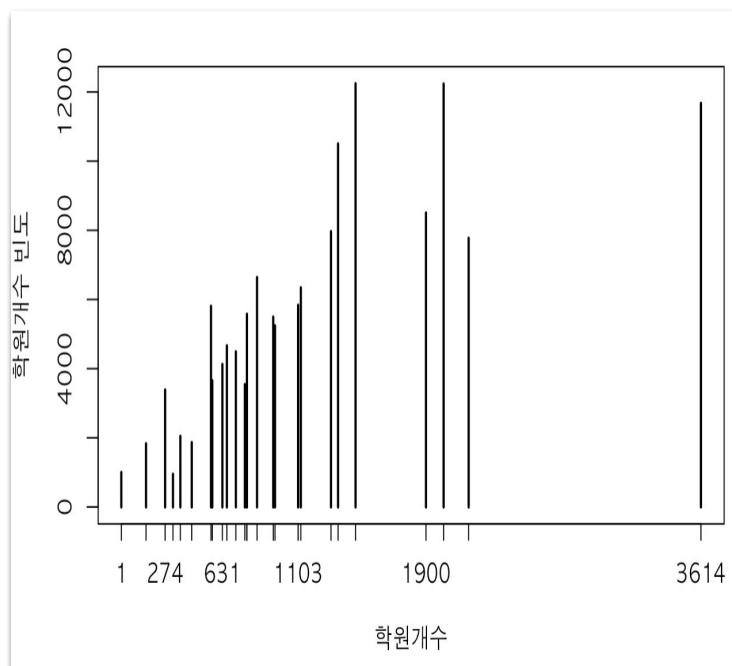
건축 년도와 보증금 산점도



건축년도에 따라서 보증금의 분포가
상이하기 때문에 **feature column** 으로
적합하다고 판단하였다.

서울시 구별 추가 데이터 EDA

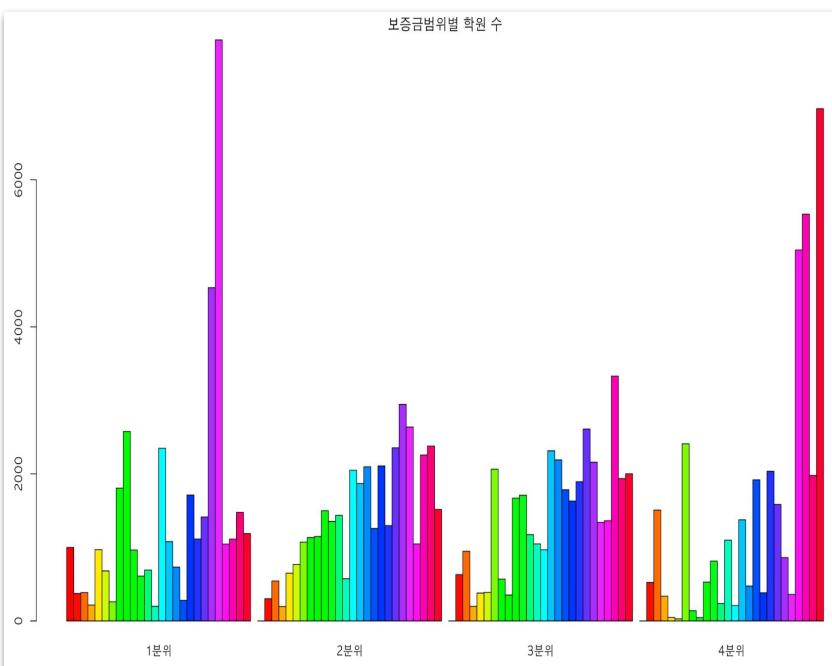
학원 수 히스토그램과 상관계수



상관계수 : 0.31462568

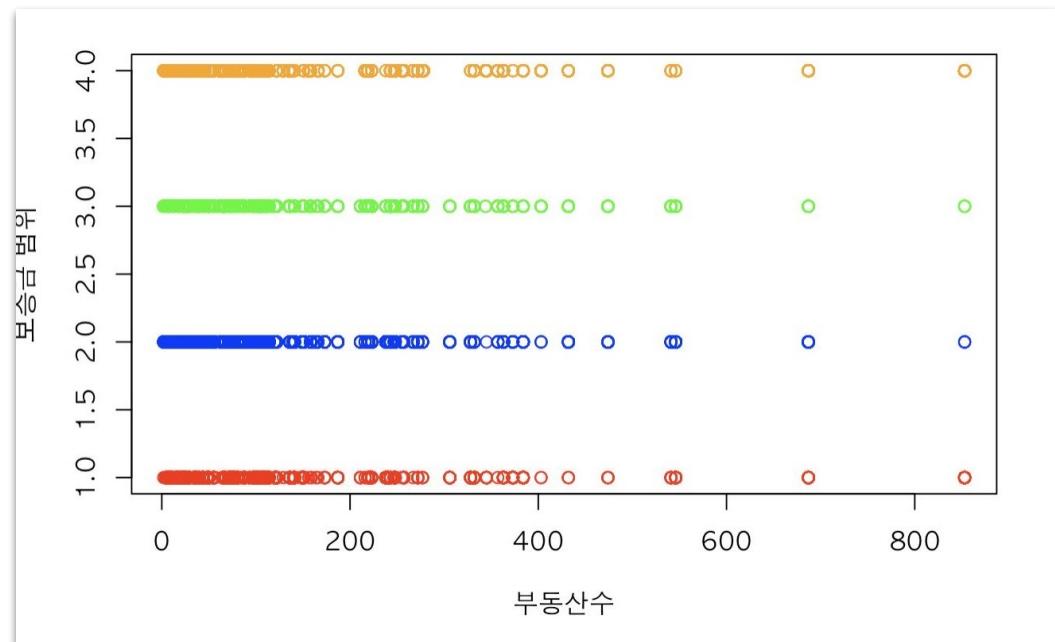
학원수 개수별 빈도수가 다르고
보증금 범위별로 학원 수의 분포와 상관계수 역시 다르고,
어느정도 약한 관계를 가지기 때문에
feature column으로 적합하다고 판단했다.

보증금 범위 별 학원 수 막대 그래프



서울시 구별 추가 데이터 EDA

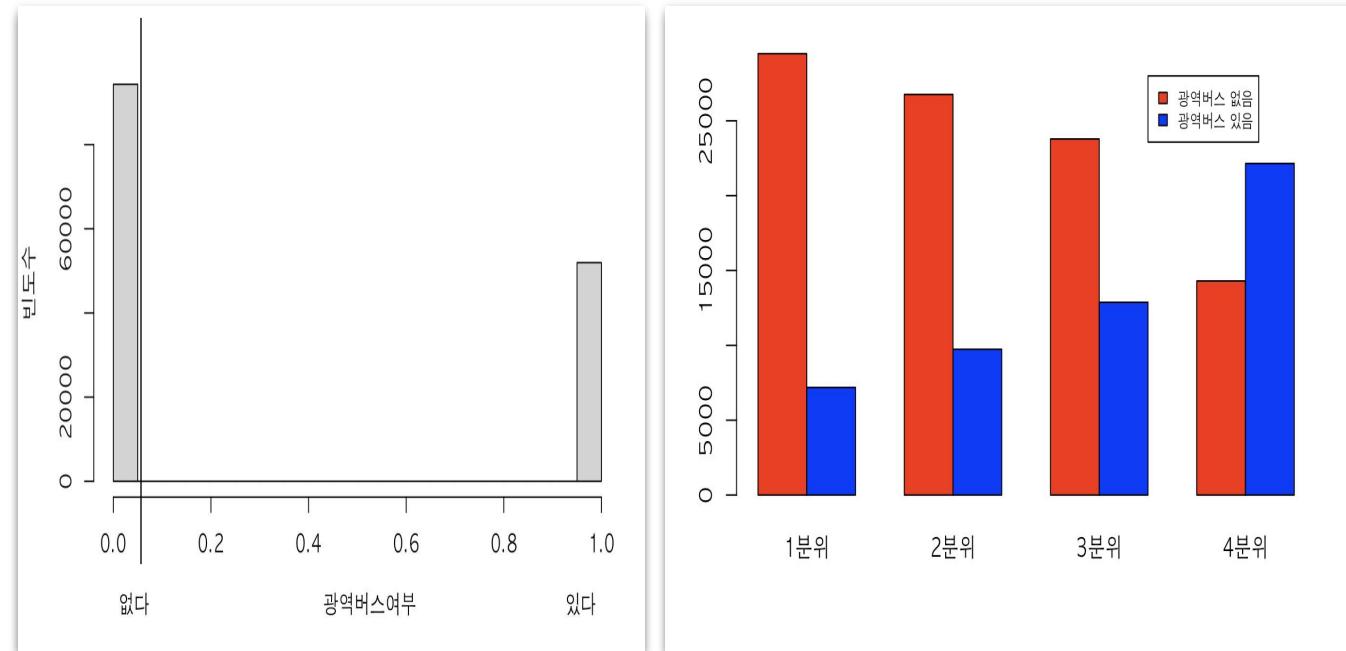
부동산 수와 보증금 범위 산점도



상관계수 : 0.1219081

부동산 수에 대한 보증금의 범위의 분포가 개수별로 비슷하고,
둘의 상관관계가 낮으므로
Feature column으로 적합하지 않다고 판단했다.

버스 노선 여부 히스토그램과 보증금 범위별 버스노선여부 막대 그래프



상관계수 : 0.3196375

히스토그램 상으로 각각의 빈도수가 다른 것을 파악했다.
또한 보증금 범위별로 광역버스 유무가 역상관관계를 가지고, 범위 별 분포가
다르기에 **Feature column**으로 적합하다.

동별 예측으로 변경한 이유

예측률이 떨어진 이유 분석

- Feature column 추가 시 최대 예측률

기본	구별 총학원수 데이터 추가	구별 동별 데이터 추가
60.9 %	72 %	69.3 %

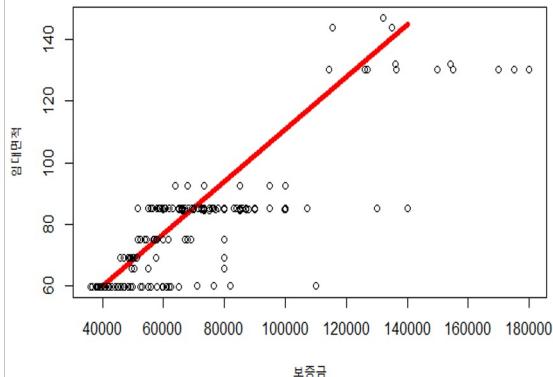
- 구별 데이터를 넣었을 경우, 예측률이 올라가므로 구별 데이터가 **보증금의 특징을 나타내는 것으로 확인된다.**
- 하지만, 더이상의 구별 동별 데이터 추가는 예측률을 떨어뜨리기에 무의미하다.
- 그러므로, 보증금 예측하기 위해 구별 데이터를 넣음으로써 보증금의 특징을 **구별, 동별로 묶어 버리기 때문**이다.

최종 결론

- 앞서 말한 이유로 구별이나 동별 데이터를 feature column으로 사용하기에 적합하지 않다고 판단
- 예측률을 올리기 위해서는 구별, 동별이 아닌 각 Row에 해당하는 추가적인 데이터가 있거나 서울시 전체가 아닌 특징을 나타내는 그룹끼리 묶어서 분석을 해야 된다.
- 임대면적, 층, 건축년도와 같이 각 **Row마다 해당되는 데이터를 구하기는 어렵다고 판단**
- 구별, 동별 데이터를 넣었을 때, 보증금 예측률이 올라갔고, 이는 보증금이 구별 동별로 특징이 있다는 것을 의미한다고 파악
- 그러므로, 앞서 동별 구별로 넣은 **feature column은 제거하기**로 했다.

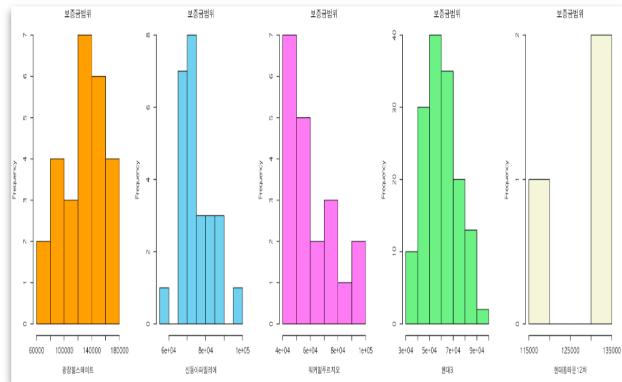
예측범위 동으로 축소 후 EDA

보증금과 아파트 임대 면적



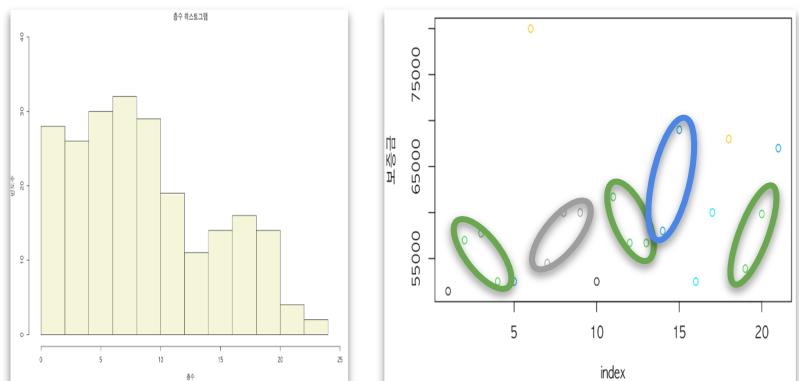
아파트의 임대 면적과 보증금의 상관계수는 0.8530484 이므로 보증금과 선형적인 관계를 뛴다고 판단하여 **feature column**으로 적합하다.

아파트별 보증금 히스토그램과 범위



- 아파트별 보증금 범위를 보았을 때 차이가 있음을 알 수 있다.
- 아파트별 보증금 히스토그램으로 봤을 때도 빈도수가 높은 보증금이 아파트별로 상이하다.
- feature column**으로 적합하다고 판단

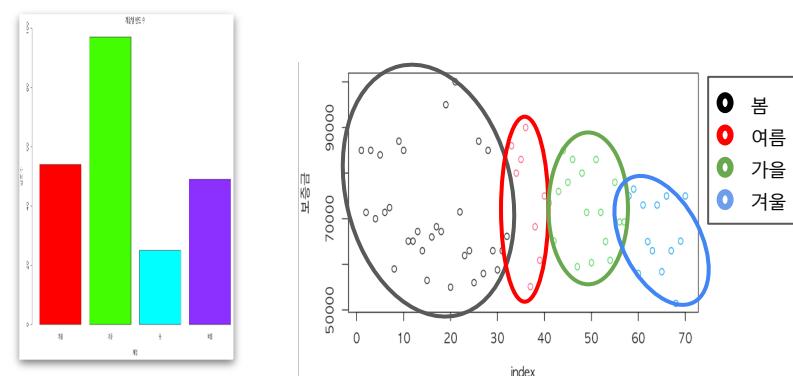
층수별 히스토그램 및 보증금 산점도를 층수별로 분류 시



아파트 층은 **feature column**으로 사용할 수 있다.

- 층수별 데이터의 빈도수가 다름을 확인할 수 있다.
- 층수별로 보증금 범위가 다름을 확인할 수 있다.

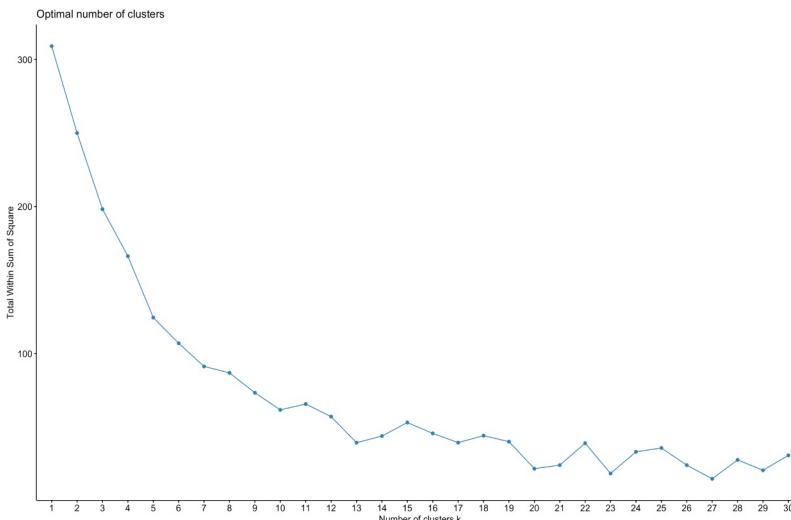
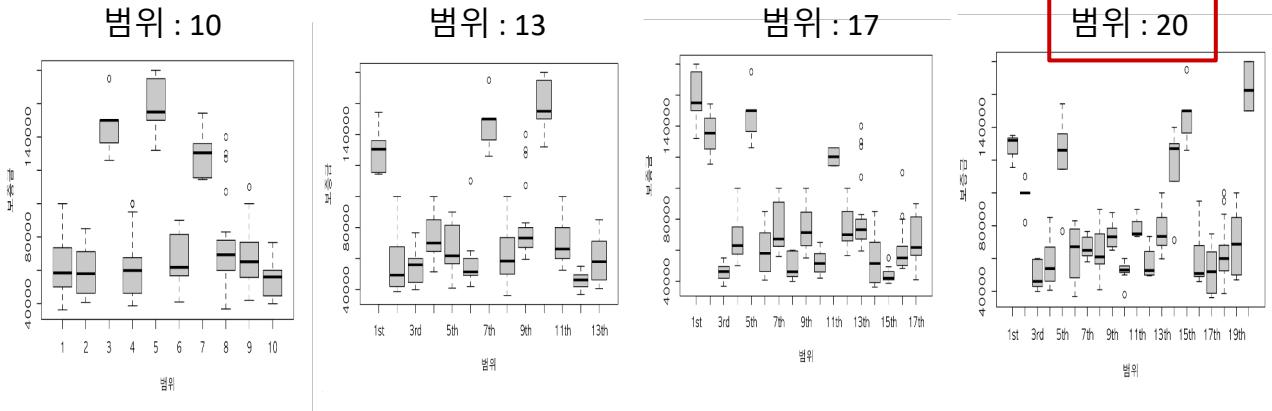
계절별 히스토그램 및 보증금 산점도를 계절별로 분류 시



계절은 **feature column**으로 사용할 수 있다.

- 계절별 데이터의 빈도수가 다름을 확인할 수 있다.
- 계절별로 보증금 범위가 다름을 확인할 수 있다.

Modeling(광장동) - 군집화(clustering)



범위로 20이 적합한 이유

- 정규분포된 데이터가 많다
- 최소값, 최대값의 범위가 겹치지 않는다
- 고르게 분포되어 있다

Modeling(광장동) - 예측치

제거 전에는 확률이 높게 나왔지만, 범위가 넓기 때문에 이상치를 제거 후의 정규화 후 인공신경망을 선택하였다.

이상치 제거 전	정규화 전(%)	정규화 후(%)
인공신경망	41.7	98.2
랜덤포레스트	99.3	98.6
SVM	95.8	95.5

이상치 제거 후	정규화 전(%)	정규화 후(%)
인공신경망	47.6	98.9
랜덤포레스트	97.8	98.5
SVM	93	93.4

위와 동일한 과정으로 예측치 구하기를 다른 동도 진행 하였다.

Scraping

목 차		1	핵심 기능	12
		1.1	CSS selector scroll	12
		1.2	multi-processing	13
2			사용한 module		
		2.1	BeautifulSoup	12
		2.2	Selenium	12-13

핵심 기능 - css selector scroll

올리브영 전국 매장 이름과 주소 scraping

1

```
scroll_bar = event.find_element(By.CSS_SELECTOR, 'div[class="sroll_store scrollbar mCustomScrollbar _mCS_2"]')
```

scroll 하고 reload를 하기 위해서 PAGE_DOWN action 실행될 div 설정

2

```
while True:
    before_height=event.execute_script("return document.querySelector('div[class='mCSB_container']").scrollHeight")
    event.execute_script("document.querySelector('div[class='mCSB_container'].style.top = '-' + str(before_height) + 'px'")
    actions = webdriver.ActionChains(event).send_keys_to_element(scroll_bar).send_keys(Keys.PAGE_DOWN)
    actions.perform()
    time.sleep(2)

    now_height=event.execute_script("return document.querySelector('div[class='mCSB_container'].scrollHeight")
    if now_height == before_height:
        break
```

css selector의 top을 div의 최대 높이로 설정하여 scroll 하고, reload를 위해 PAGE_DOWN 액션을 준다.

scroll 되기 전 높이와 된 후의 높이를 비교하여 마지막 까지 scroll이 됐는지 확인 할 수 있다.

3

```
html = event.page_source
soup = BeautifulSoup(html, 'html.parser')
```

```
shops=soup.select('#wordStoreList a')
addresses=soup.select('#wordStoreList p')
```

BeautifulSoup을 이용하여 마지막 까지 scroll이 된 page를 parsing해서 내가 원하는 정보인 가게 이름과 가게 주소를 선택하여 변수에 저장한다.

핵심 기능 - multi processing

가장 맛있는 족발 가게명 , 주소, 전화번호, 이미지 저장하여 csv 파일로 저장

1

```

if __name__ == "__main__":
    # recursionlimit 늘려주기
    sys.setrecursionlimit(10000)

    p = Pool(os.cpu_count - 2)

    # 가게 몇 개인지 판단하기 위해서
    url='https://gajok.kr/sub/store.php'

    res = req.urlopen(url)

    soup = BeautifulSoup(res,"html.parser")

    shops = soup.select('dt')

    shops = [shop.text for shop in shops]
    shops.remove('')

    # multi processing
    ret1 = p.apply_async(get_data,(1,100,len(shops)+1))
    ret2 = p.apply_async(get_data,(100,200,len(shops)+1))
    ret3 = p.apply_async(get_data,(200,250,len(shops)+1))
    ret4 = p.apply_async(get_data,(350,len(shops)+1,len(shops)+1))
    ret5 = p.apply_async(get_data,(250,300,len(shops)+1))
    ret6 = p.apply_async(get_data,(300,350,len(shops)+1))

    names = ret1.get()[0]+ret2.get()[0]+ret3.get()[0]+ret4.get()[0]+ret5.get()[0]+ret6.get()[0]
    addresses = ret1.get()[1]+ret2.get()[1]+ret3.get()[1]+ret4.get()[1]+ret5.get()[1]+ret6.get()[1]
    phones = ret1.get()[2]+ret2.get()[2]+ret3.get()[2]+ret4.get()[2]+ret5.get()[2]+ret6.get()[2]
    images = ret1.get()[3]+ret2.get()[3]+ret3.get()[3]+ret4.get()[3]+ret5.get()[3]+ret6.get()[3]

df=pd.DataFrame(data=zip(names,addresses,phones,images), columns=['Name','Address','Phone','Image'])

df.to_csv('./Data/gajok2.csv',encoding='utf-8',index=False)

p.close()
p.join()

```

multi process로 실행되기 위해
cpu의 코어 개수 파악

2

```

def get_data(start,end,len):
    chrom_options = webdriver.ChromeOptions()
    driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()),options=chrom_options)
    driver.get("https://gajok.kr/sub/store.php")

    names = []
    addresses = []
    phones = []
    images = []

    for i in range(start,end):
        try:
            xpath = f'//@id="result_store"/div[1]/ol/li[{i}]/a'
            driver.find_element(By.XPATH,xpath).click()

            time.sleep(1)

            xpath = f'//@id="map"/div[1]/div/div[6]/div[{len}]/div/div[2]/a'
            driver.find_element(By.XPATH,xpath).click()

            time.sleep(1)

            # BeautifulSoup 사용하기
            # selenium의 BeautifulSoup 이용해서 텍스트 정보 가져오기
            html = driver.page_source
            soup = BeautifulSoup(html, "html.parser")

            # 가게 이름
            names.append(soup.select_one('h4').text)
            print("가게이름 : ",soup.select_one('h4').text)

            # 주소
            address = soup.select_one('div.info li')
            addresses.append(address.text.replace('내선번호',''))
            print("가게주소 : ",address.text.replace('내선번호',''))

            # 번호
            phone=soup.select_one('div.info a')
            phones.append(phone.text)
            print("가게번호 : ",phone.text)

            # 이미지 저장하기
            image = driver.find_element(By.XPATH,'//@id="detail"/div/div/div[1]/div')
            url="https://gajok.kr"+image.get_attribute('style').split(':')[0][2:-2]

            # 공백 %20 바꾸기
            url=url.replace(' ','%20')

            # 한글 처리
            url_kore.compile('[-\u00ac]+|[^\u00ac-\u00ac]+').findall(url)

            images.append(url)
            print("이미지url : ",url)

            saveName = f'./image/{i}.jpg'
            req.urlretrieve(url, saveName)

            xpath = '//@id="detail"/div/div/a'
            driver.find_element(By.XPATH,xpath).click()
            time.sleep(1)

        except:
            print("error")

    return names, addresses , phones , images

```

가져와야 될 데이터의 개수 파악
하여 이를 해당 되는 core 개수에
맞게 개수를 분할

작업 완료 후 하나의 data-frame으로 만들어 csv로 저장

image는 따로 저장하고 이의
url을 문자열로 변수로 저장해
두었다.

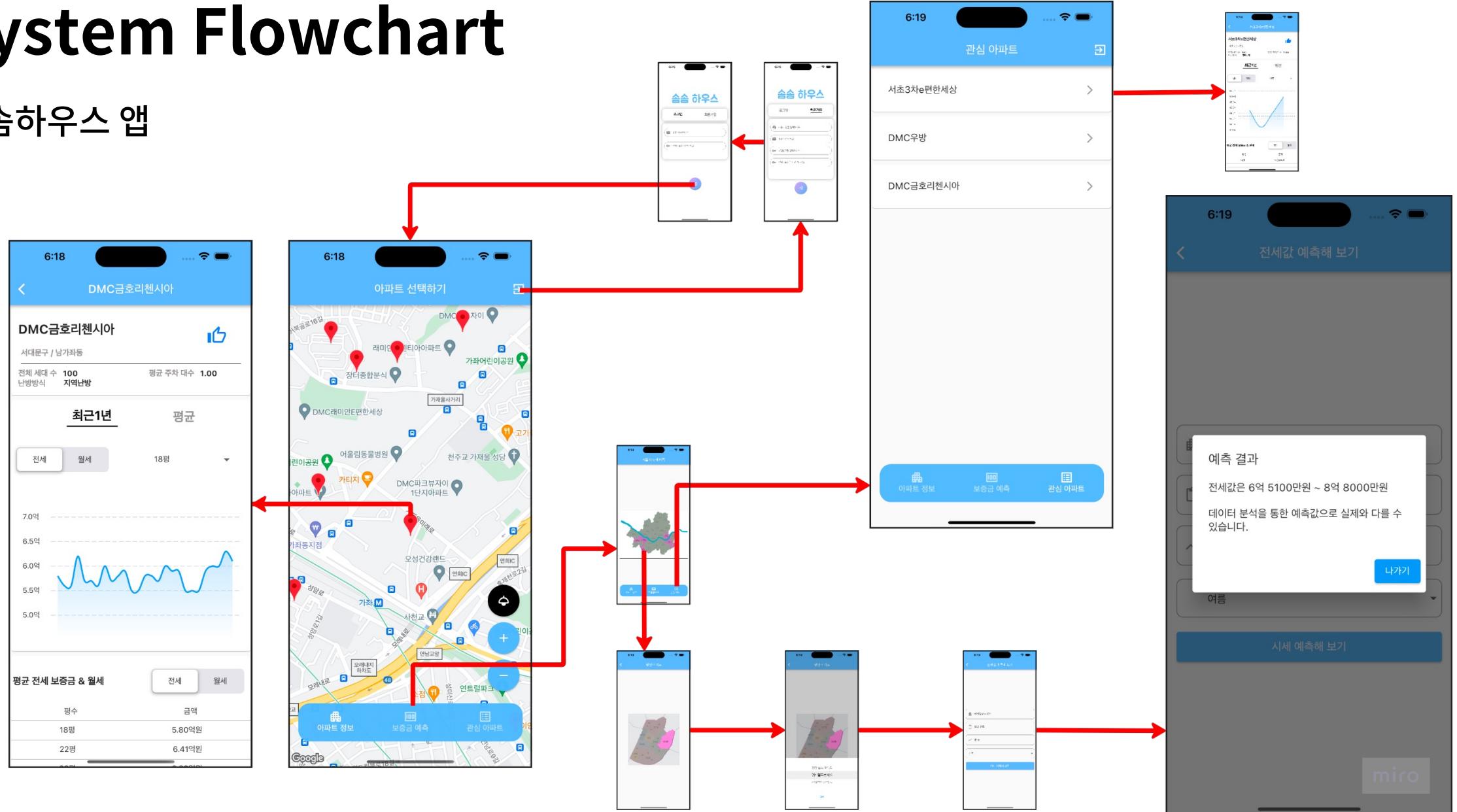
multi process로 실행될 함수

Flutter

목 차	1	System Flow Diagram	12
	2	핵심 기능	13
	2.1	Google Map API	13
	2.2	JSON data 이용한 차트 기능	14
	2.3	머신 러닝 모델 이용한 전세금 예측 기능	15
	2.4	SQLite 이용한 관심 목록 기능	16
3	Flutter Package			
	3.1	fl_chart	17
	3.2	flutter_spinkit	17
	3.3	dropdown_button2	17
	3.4	custom_sliding_segmented_control	...	17
	3.5	path	18
	3.6	http	18
	3.7	sqflite	19

System Flowchart

솜솜하우스 앱



핵심 기능 - Google Map API

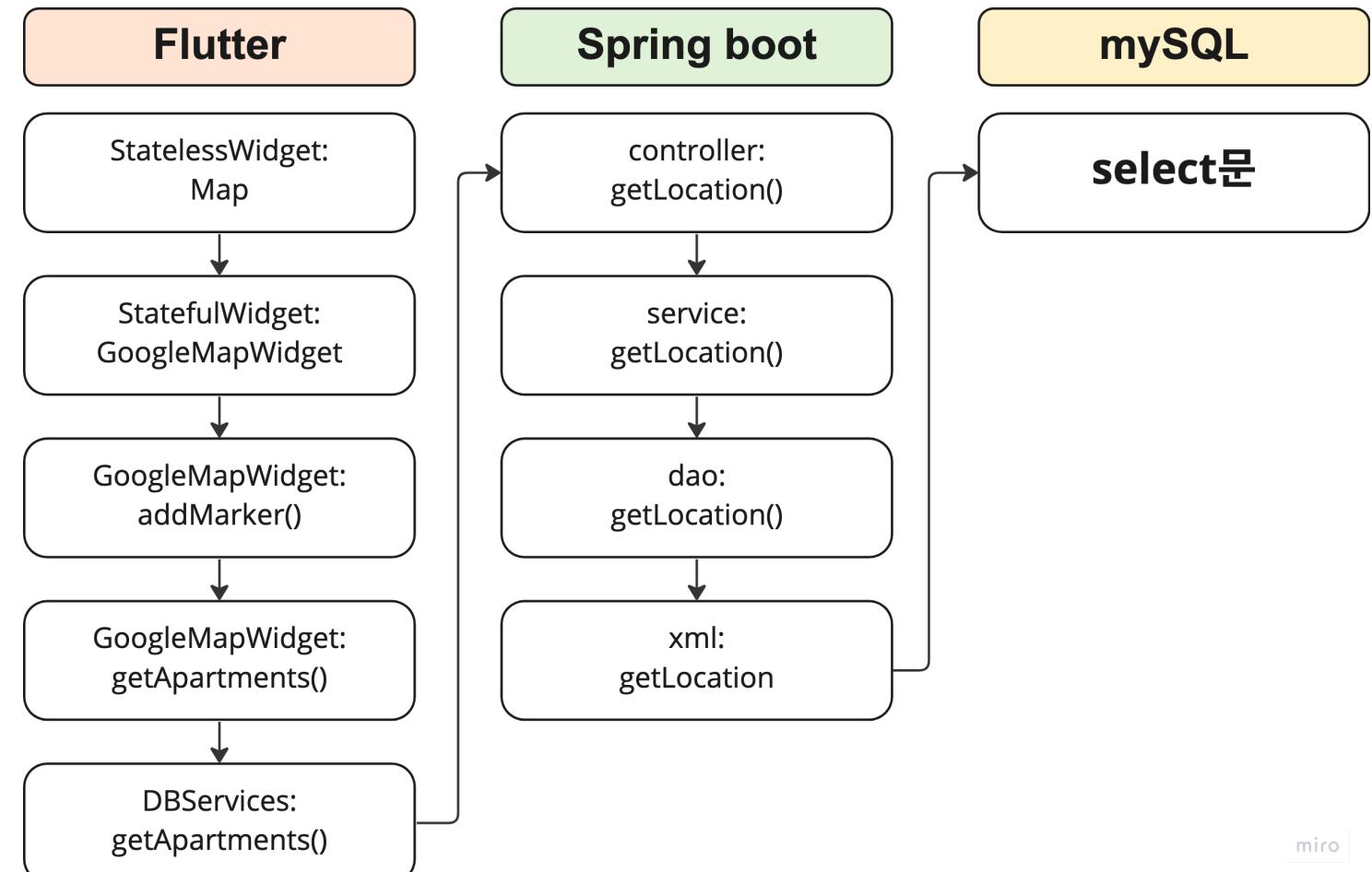
MySQL DB의 경도, 위도 이용한 Marker 기능 및 stack widget 활용하여 child 위젯 분리

- ZoomLevel에 따른 거리를 위도와 경도로 변환하여 중심 위치로부터의 일정 반경의 아파트 위치를 가져오도록 구현 했다.

- Stack 위젯을 활용해 GoogleMap 화면을 분리하여 추가적인 화면 구성 시 원활할 수 있도록 디자인 했다.

- Query문

```
select name, lat, lng from apartment_info
where
lat between (${lat} - 0.0091* ${km})
and (${lat} + 0.0091* ${km})
and
lng between (${lng} - 0.0113* ${km})
and (${lng} + 0.0113* ${km})
```



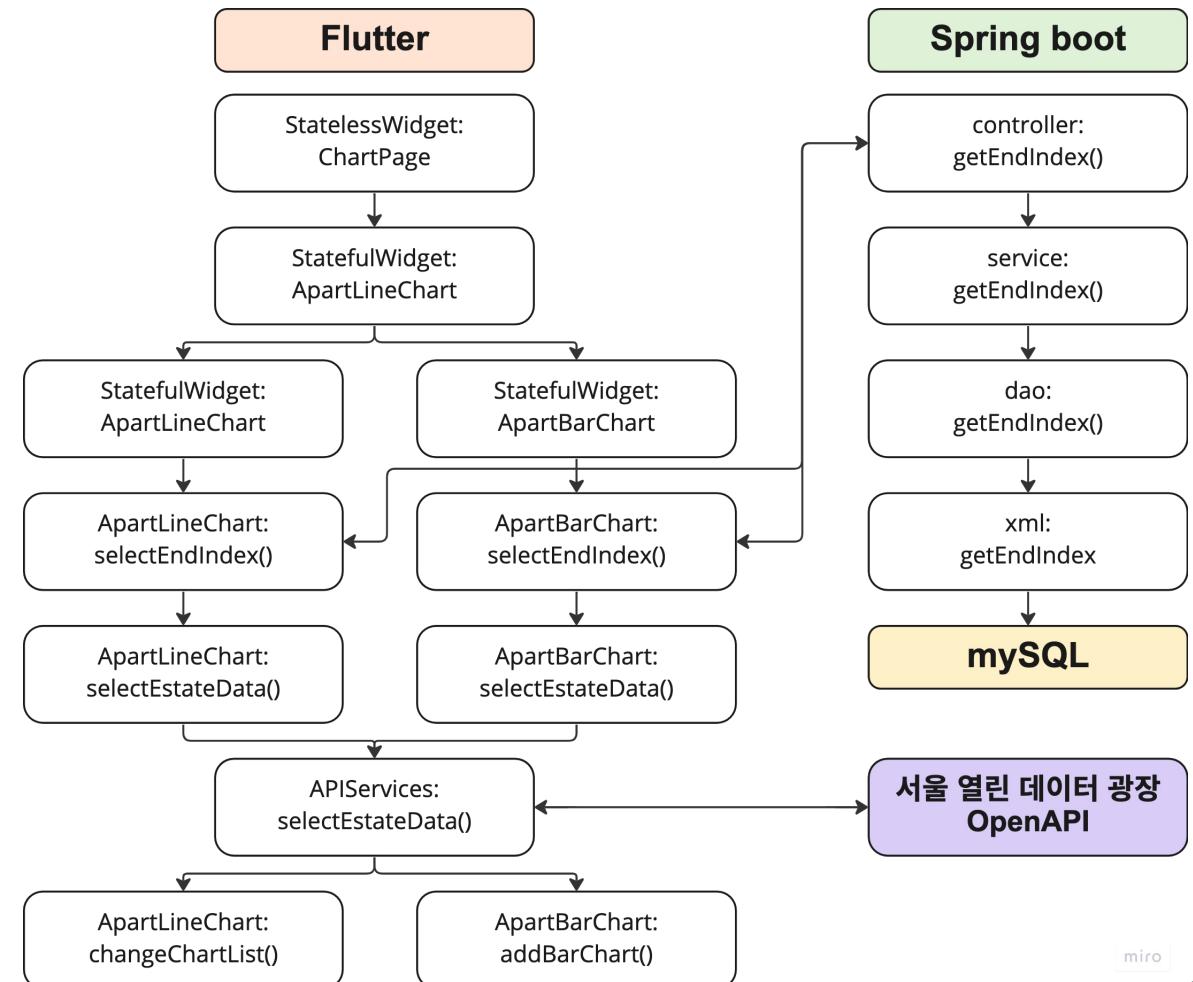
핵심 기능 - JSON data 이용한 차트

서울 열린 데이터 광장 openAPI로 JSON 구조의 데이터를 받아서 fl-chart 패키지 이용하여 차트로 표현

- openAPI로 받아온 JSON 구조의 데이터를 아래와 같은 형식(Map<String,Map<String,List>>)으로 변환시킨다.

```

for (var map in rowList) {
  if (map['RENT_GBN'] == '전세') {
    int area = (map['RENT_AREA'] / 3.3058).round();
    geonAreaMap['$area평'] == null
      ? geonAreaMap['$area평'] = [map['RENT_GTN']]
      : geonAreaMap['$area평']?.add(map['RENT_GTN']);
  } else if (map['RENT_GBN'] == '월세') {
    int area = (map['RENT_AREA'] / 3.3058).round();
    wallAreaMap['$area평'] == null
      ? wallAreaMap['$area평'] = [map['RENT_FEE']]
      : wallAreaMap['$area평']?.add(map['RENT_FEE']);
  }
}
  
```



- 위의 자료구조를 changeChartList() 와 addBarChart()를 통해 평수별 선그래프와 평수별 평균 막대 그래프가 요구하는 데이터로 변형시켜 차트로 표현한다.

핵심 기능 - 머신러닝 예측 모델 이용한 전세금 예측 기능

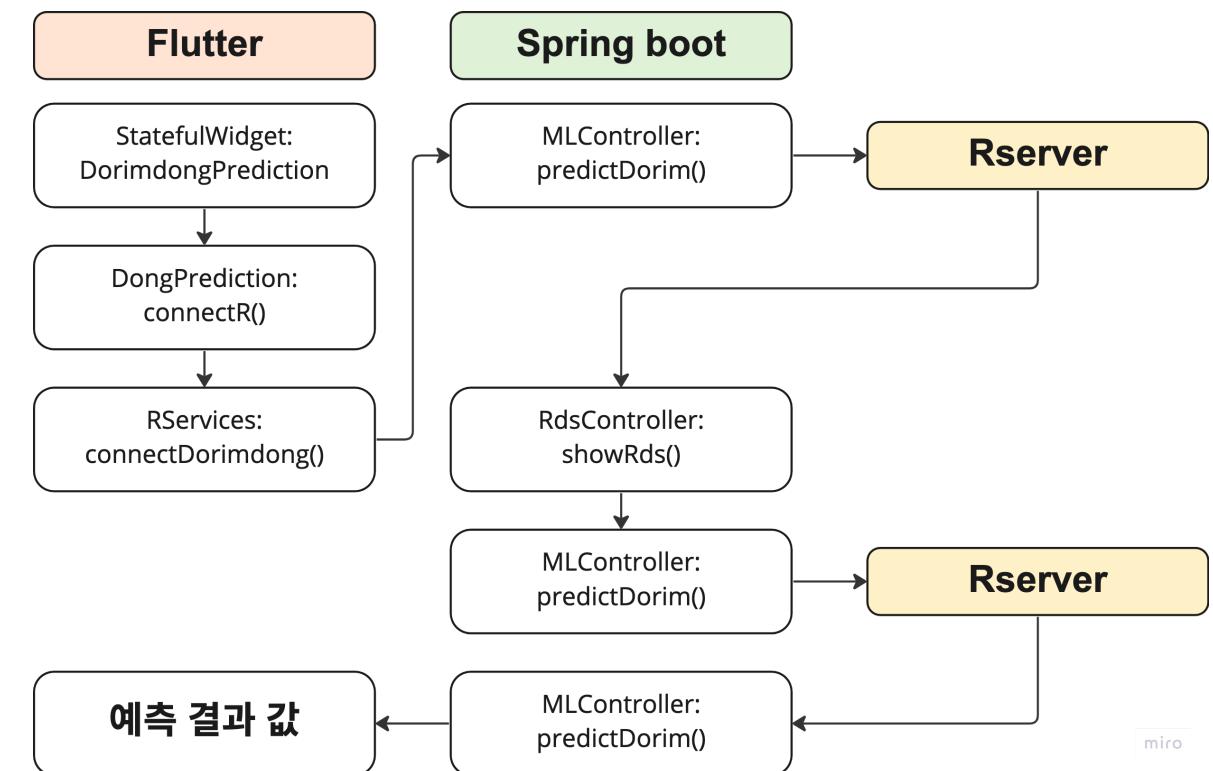
Web Server의 rds파일을 가져와서 Spring Boot에서 Rserver와 연결하여 예측결과를 불러와서 띄우는 기능

- Spring boot에서 Rserver와 연결을 한다.
- Web Server에 있는 rds 파일 Rserver와 연결시킨 후 모델링을 한 rds

파일을 다운로드 받은 후 Flutter의 입력값을 입력하여 예측 값을
Flutter로 보낸다.

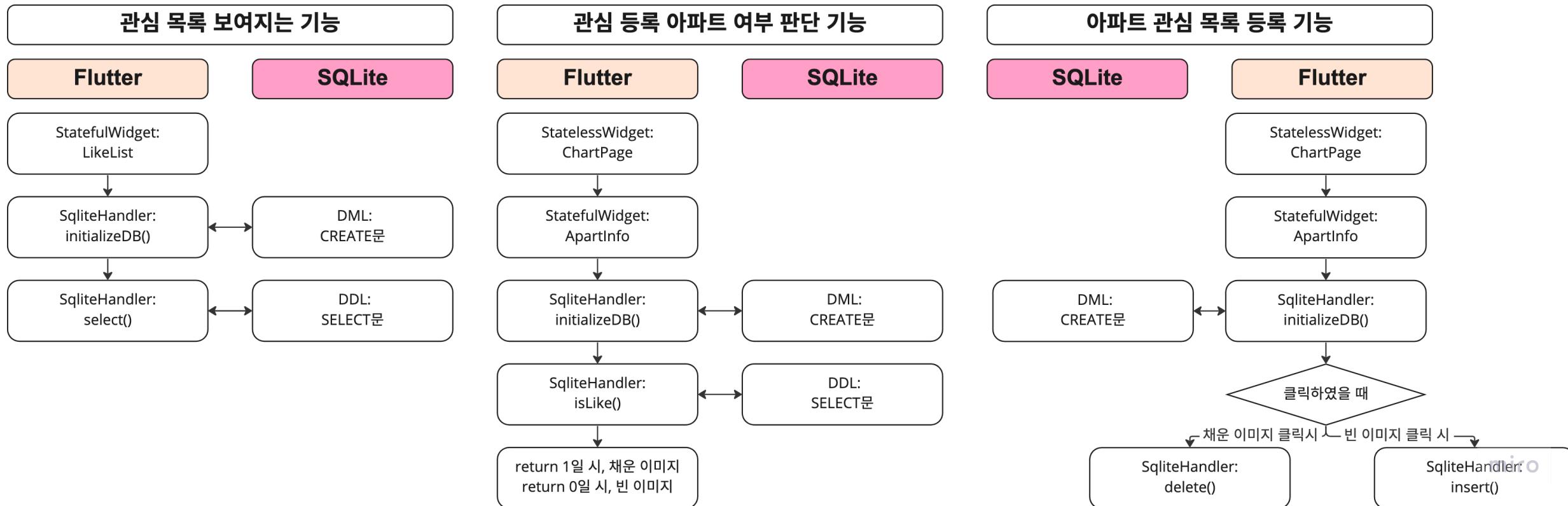
- 받아온 결과 값을 App 상에 '#억 ####만원' 형식으로 변환시키기 위한
코드

```
String num1NeckStr = num1Neck != 0 ? '$num1Neck억 ' : '';
String num1TrashStr = num1Trash != 0 ? '$num1Trash만원' : '';
String num2NeckStr = num2Neck != 0 ? '$num2Neck억 ' : '';
String num2TrashStr = num2Trash != 0 ? '$num2Trash만원' : '';
```



핵심 기능 - SQLite 이용한 관심 목록 기능

관심 목록 기능은 개인 사용자에게만 저장이 되기 때문에 SQLite를 이용하여 기능을 구현



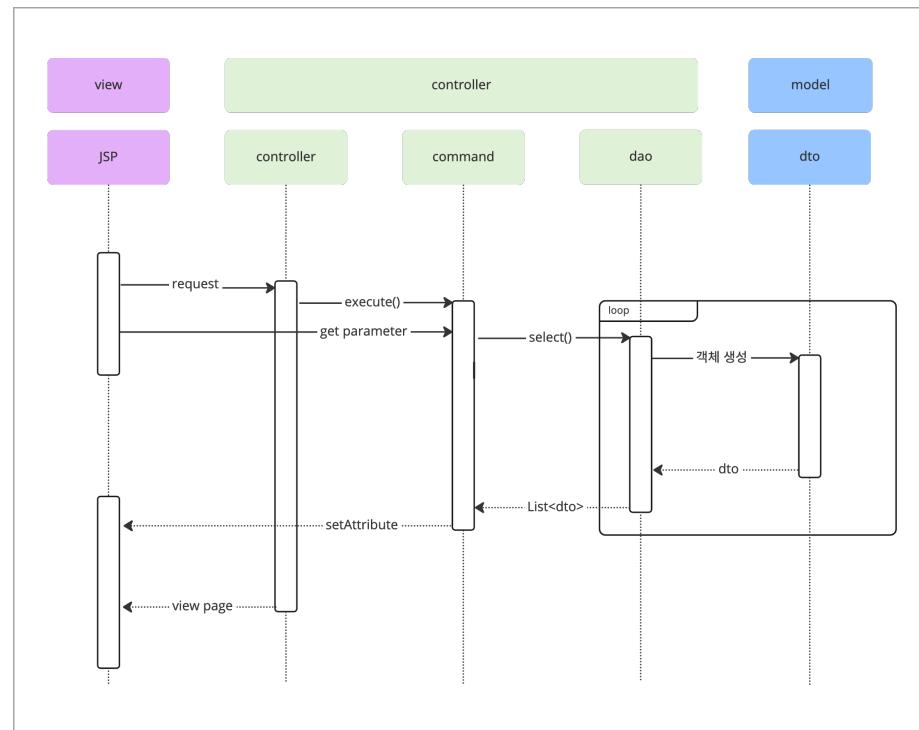
Web & Backend

목 차

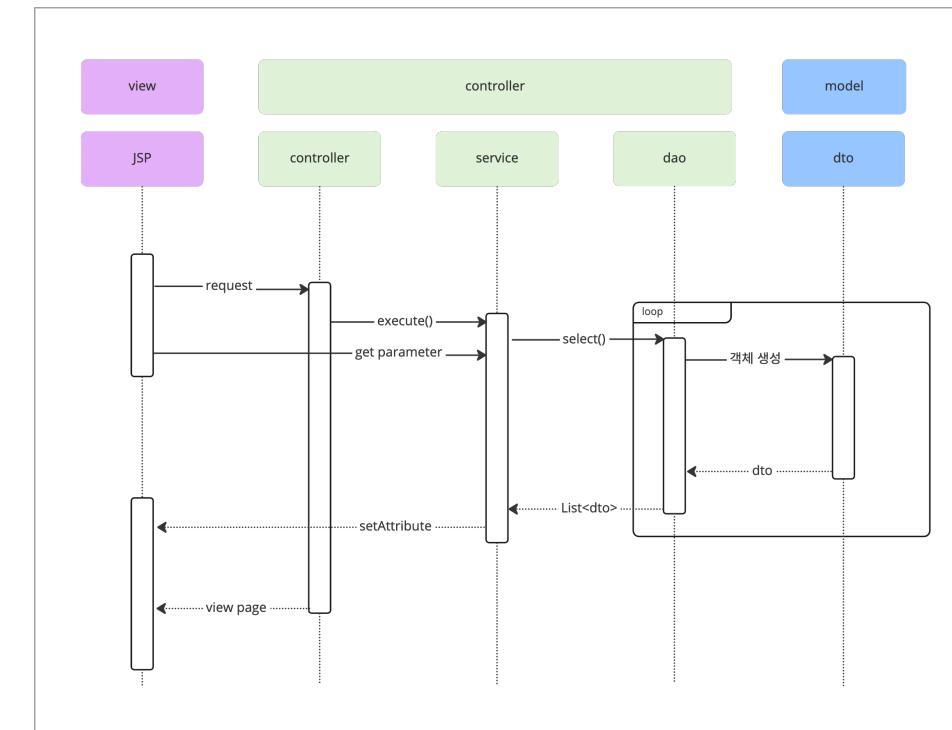
1	Web	21	3.5	Paging	28
1.1	MVC pattern	21			
2	Backend	22	4	사용한 Dependencies	
2.1	RestAPI 구현	22	4.1	json simple jar	22
3	핵심 기능	23	4.2	Rserve	22
3.1	multi 파일 업로드	23	4.3	Gson	25
3.2	파일 다운로드	24	4.4	email	26
3.3	Kakao Login API	25	4.5	MyBatis	28
3.4	회원가입	26	4.6	MySQL	28
3.4.1	이메일 인증	26	4.7	JSP	
3.4.2	비동기 처리	27	4.8	JSTL	
			4.9	JDBC	

MVC Pattern

Petmily JSP 프로젝트



Petmily Spring Boot 프로젝트



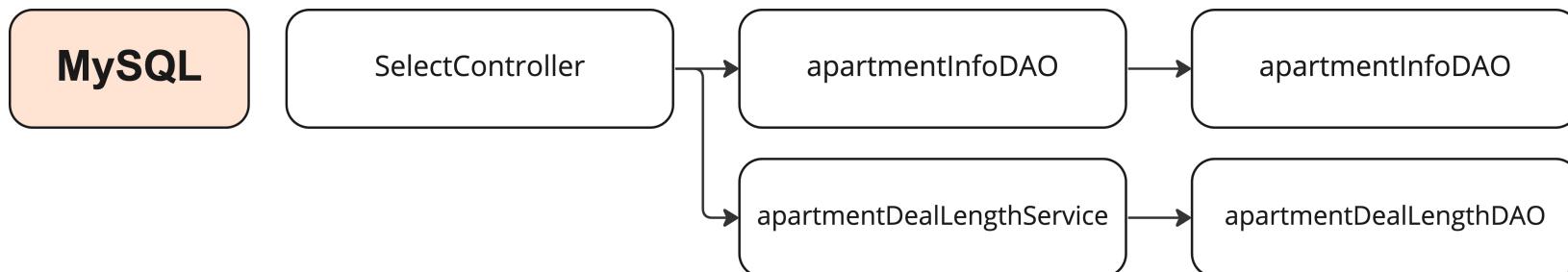
- 하나의 controller에 actionDo라는 함수를 줘서 switch case 문을 활용하여 여러개의 command를 용도에 맞게 mapping 한다.
- SQL과 연결되는 부분은 DAO에서 실행

- autowired annotation을 이용하여 singleton으로 객체를 관리
- application.properties 와 configuration class로 sql과 연결을 myBatis를 이용한다는 차이점
- JSP 프로젝트와는 달리 성격이 비슷한 request별로 controller를 생성하여 관리

RestAPI 구현

Spring boot의 Rest Controller를 이용하여 앱과 DB 및 RServer 연결하는 Backend 구현

MySQL 과 Backend 연결



- simple.jar 이용하여 DB 값을 JSON 형태로 parsing 하여 JSON 형태의 데이터를 반환해준다.

```

JSONObject jsonList = new JSONObject();
JSONArray itemList = new JSONArray();

for(ApartmentInfoDto myBatis : myBatisList) {
    JSONObject tempJson = new JSONObject();
    tempJson.put("name", myBatis.getName());
    tempJson.put("lat", myBatis.getLat());
    tempJson.put("lng", myBatis.getLng());
    itemList.add(tempJson);
}

jsonList.put("results", itemList);
  
```

simple.jar 이용하여 JSON 형태로 parsing 하는 code

R 과 Backend 연결



- Parameter로 입력한 값을 받는다.
- 입력한 값을 rds의 feature column 형식에 맞게 변환
- Rserver 연결 후 r code에 변환 값을 대입
- result 값을 simple.jar 이용하여 JSON 형태로 parsing 후 반환

```

RConnection conn = new RConnection();

conn voidEval("library(randomForest)");

conn voidEval("rf <- readRDS(url('http://localhost:8080/show_rds?name=ml_singeongdong.rds','rb'))");

conn voidEval("result <- as.character(predict(rf, (list(" +
    + "롯데캐슬=" + nameOneHot.get(0) +
    + ", 겨울=" + WeatherOneHot.get(3) +
    + "))))");

String result = conn eval("result").asString();
  
```

Rserver 연결 후, r code를 작성하여 result 값을 받는 코드

핵심 기능 - multi 파일 업로드

MultiPartHttpServletRequest를 이용한 Web Server에 다중 파일 업로드 기능

- user로부터 file을 web상에 받기 위해서 **enctype = “multipart/form-data”** 로 변경

```

multipartFiles = request.getFiles("file");
try {
    // 텍스트만 입력
    postingDAO.postingInsertText(ptitle, pcategory, pcontent, plocation_basic, plocation_detail, user_uid);
    // 이 유저의 마지막 포스팅 아이디
    lastPostingId = postingDAO.postingGetId(user_uid);

    int cnt = 1;
    if (!multipartFiles.isEmpty()) {
        for (MultipartFile file : multipartFiles) {
            String path = System.getProperty("user.dir") + "//src//main//webapp//posting";
            // 파일을 uid로 만들기 위한 기초단계
            // 확장자 가져오기
            String originalName = file.getOriginalFilename();
            if (cnt == 1) {
                originalName = lastPostingId + "_1_" + originalName;
                pimage1 = originalName;
            } else if (cnt == 2) {
                originalName = lastPostingId + "_2_" + originalName;
                pimage2 = originalName;
            } else if (cnt == 3) {
                originalName = lastPostingId + "_3_" + originalName;
                pimage3 = originalName;
            }
            // 파일 네임 짓기
            // 폴더에 "name" 으로 saveFile을 만들 빈 컨테이너를 생성해 준다.
            File saveFile = new File(path, originalName);
            // file을 saveFile이름과 path로 지어서 넣기
            file.transferTo(saveFile);
            cnt++;
        }
    }
}

```

- 파일 업로드 위한 설정
 - application.properties에 최대 파일 사이즈 지정
- 파일 업로드 부분을 제외한 나머지 text들을 DB에 **INSERT** 한다.
- 파일이 있을 시 파일명을 insert하기 위해 게시글의 PK를 **SELECT** 해온다.
- WebServer의 image 폴더에 저장하기 위한 **경로 지정**
- WebServer에 **중복된 이름**이 있을 수 있기에 이를 **방지**하기 위해 올린 이름 앞에 게시글 PK 값과 몇 번째 넣은 이미지인지 알기 위해 1부터 3의 숫자를 사용
- 이 값을 경로와 합쳐서 **WebServer에 저장**
- 저장 후 그 값을 DB에 해당 PK값을 찾아서 **INSERT** 한다.

핵심 기능 - WebServer 파일 다운로드

Rserve에서 WebServer의 rds 파일을 다운 받아 머신 러닝을 돌리기 위한 다운로드 기능

```
// [로직 : 서버 로컬 pc에 저장 된 rds 확인 >> 저장된 rds 파일이 존재하는 경우 >> 그 파일을 다운로드]
@RequestMapping("/show_rds")
public ResponseEntity<Object> showRds(@RequestParam Map<String, String> param){
    // rds가 저장된 폴더 경로 변수 선언
    String rdsRoot = System.getProperty("user.dir") + "/src/main/resources/webapp/rds/";

    // 서버 로컬 경로 + 파일 명 저장 실시
    rdsRoot = rdsRoot + String.valueOf(param.get("name"));

    try {
        Path filePath = Paths.get(rdsRoot);
        Resource resource = new InputStreamResource(Files.newInputStream(filePath)); // 파일 resource 얻기

        File file = new File(rdsRoot);

        HttpHeaders headers = new HttpHeaders();

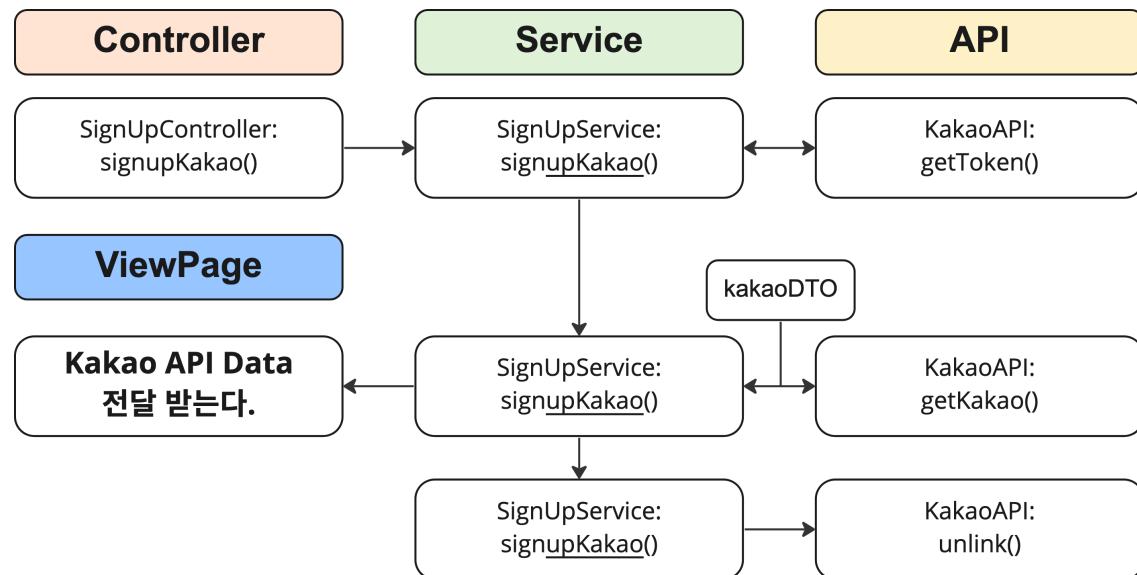
        // 다운로드 되거나 로컬에 저장되는 용도로 쓰이는지를 알려주는 헤더
        headers.setContentDisposition(ContentDisposition.builder("attachment").filename(file.getName()).build());

        return new ResponseEntity<Object>(resource, headers, HttpStatus.OK);
    } catch(Exception e) {
        return new ResponseEntity<Object>(null, HttpStatus.CONFLICT);
    }
}
```

- WebServer의 rds 파일 경로를 찾는다.
- InputStreamResource를 이용해 해당 파일을 생성
- HttpHeaders content disposition을 attachment로 줌으로써 body에 오는 값을 다운로드 받는다.

핵심 기능 - Kakao Login API

Kakao Login API를 이용하여 이름 , 프로필 이미지 , ID , 이메일 , 성별 , 생일 데이터 받아오기



- kakao로 회원가입 클릭 시 Kakao Login API를 이용하여 회원가입 시 필요한 데이터를 받아온다.
- getKakao()로 받아온 data를 redirectAttribute를 이용하여 view page로 전달한다.

Java RestAPI 요청 방식 및 JSON 형식 response 받는 방법

- Java에서 RestAPI 요청하는 코드

```

url = new URL(getPersonalInfoUrl);
conn = (HttpsURLConnection) url.openConnection();
// header는 setRequestProperty로 보내기
conn.setRequestProperty("Authorization", "Bearer " + access_token.trim());
conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
conn.setRequestProperty("charset", "utf-8");

conn.setRequestMethod("POST");
conn.setDoOutput(true);

writer = new OutputStreamWriter(conn.getOutputStream());
writer.flush();
  
```

- JSON 형식 response를 gson.jar 이용해서 받기

```

// json 형식으로 post 받은 결과값
String responseParam = buffer.toString();
// json 형식의 string을 json으로 parsing 하기 위해서
JsonParser parser = new JsonParser();
// 전체 element
JsonElement element = parser.parse(responseParam);
// element의 object 단위
JsonObject properties = element.getAsJsonObject().get("properties").getAsJsonObject();
JsonObject kakao_account = element.getAsJsonObject().get("kakao_account").getAsJsonObject();

// object에서의 원시 타입의 데이터
name = properties.get("nickname").getAsString();
profile = properties.get("profile_image").getAsString();
  
```

핵심 기능 - 이메일 인증

Google SMTP 이용하여 입력한 이메일로 랜덤한 9-10자리 숫자를 보내서 인증을 받게 하는 기능

이메일 발송 위한 초기 설정

pom.xml

```
<!-- email -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
</dependency>
```

application.properties

```
#email
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=#####
spring.mail.password=#####
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.auth=true
```

인증 번호 생성 및 이메일 발송 코드

이메일 발송 code

```
@Override
public int sendEmail(HttpServletRequest request) throws Exception {
    String email = request.getParameter("uemail");

    Random random = new Random();
    int certifyNum = random.hashCode();

    MimeMessage message = javaMailSender.createMimeMessage();
    MimeMessageHelper helper = new MimeMessageHelper(message, true, "UTF-8");

    String receiver = email; // 메일 받을 주소
    String title = "[펫밀리] 회원가입 이메일 인증";
    String content = "#### html 형식 "+certifyNum + "으로 된 이메일 형식 #####";
    helper.setSubject(title);
    helper.setTo(receiver);
    helper.setText(content, true);

    javaMailSender.send(message);

    return certifyNum;
}
```

핵심 기능 - Ajax 이용한 비동기 처리

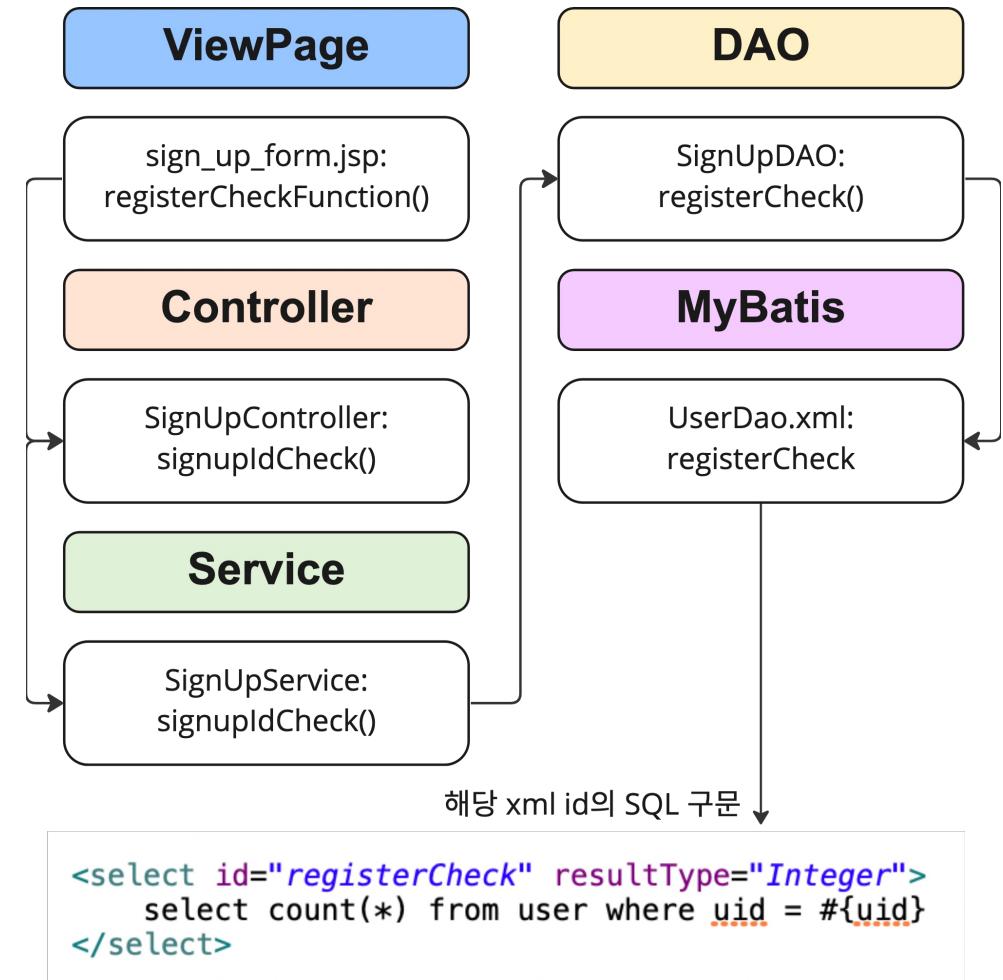
Ajax와 Spring boot의 @ResponseBody이용한 비동기 처리

ID 중복체크 , 이메일 인증 확인 , 동물 선택 시 해당 동물의 종의 종류 나타내는 기능에 사용
대표로 ID 중복체크 만을 다루도록 하겠다.

ID 중복체크

- Ajax 사용하여 DB에 존재하는지 아닌지 check하는 url을 request 한다.
- 반환값을 0과 1로 주어서 중복체크를 하여 해당 되는 값에 따라 웹페이지의 상태를 바꿔준다.

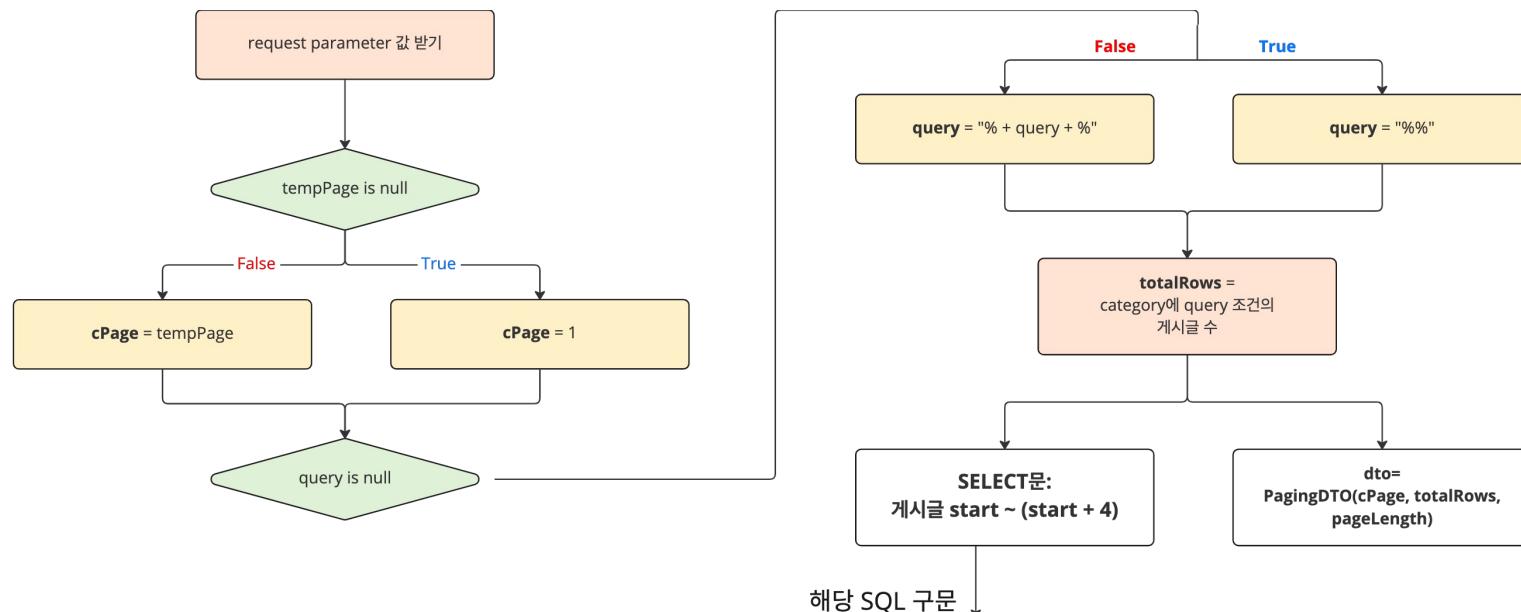
```
$.ajax({
    type : 'POST',
    url : 'sign_up_id_check',
    data : {uid , uid},
    success:function(result){
        if(result == 0){
            $("label[for='id_overlap_status']").text("사용가능한 아이디입니다.");
            $("label[for='id_overlap_status']").css("color","blue");
            $('#id_overlap_check').attr('value','available');
            $('#uid').attr("readonly",true);
        }else{
            $("label[for='id_overlap_status']").text("이미 존재하는 아이디입니다.");
            $("label[for='id_overlap_status']").css("color","red");
            $('#id_overlap_check').attr('value','unavailable');
        }
    }
});
```



핵심 기능 - Paging

게시판의 개수와 현재 page에 따라서 나타내야 할 게시글과 paging 숫자를 판단하는 기능

Paging Algorithm flow chart & query문



```

<select id="postingGetList" resultType="com.petmily.customer.dto.PostingDTO">
    select pid, ptitle, plocation_basic, pinitdate, user_uid, pcategory from posting
    where pccategory = #{pcategory} and pdeletedate is null and ${option}
        like #{query} order by pid desc limit #{start} ,#{rowLength}
</select>
    
```

PagingDTO()에 속한 변수

totalPage : 하단에 나타날 page의 수

totalRows / pageLength을 올림 한 수

currentBlock : 현재 페이지가 속한 블럭

예) 1~5 : 1 / 6~10 : 2 / 11~15 : 3

cPage / pageLength을 반올림 한 수

startPage : block의 시작 페이지

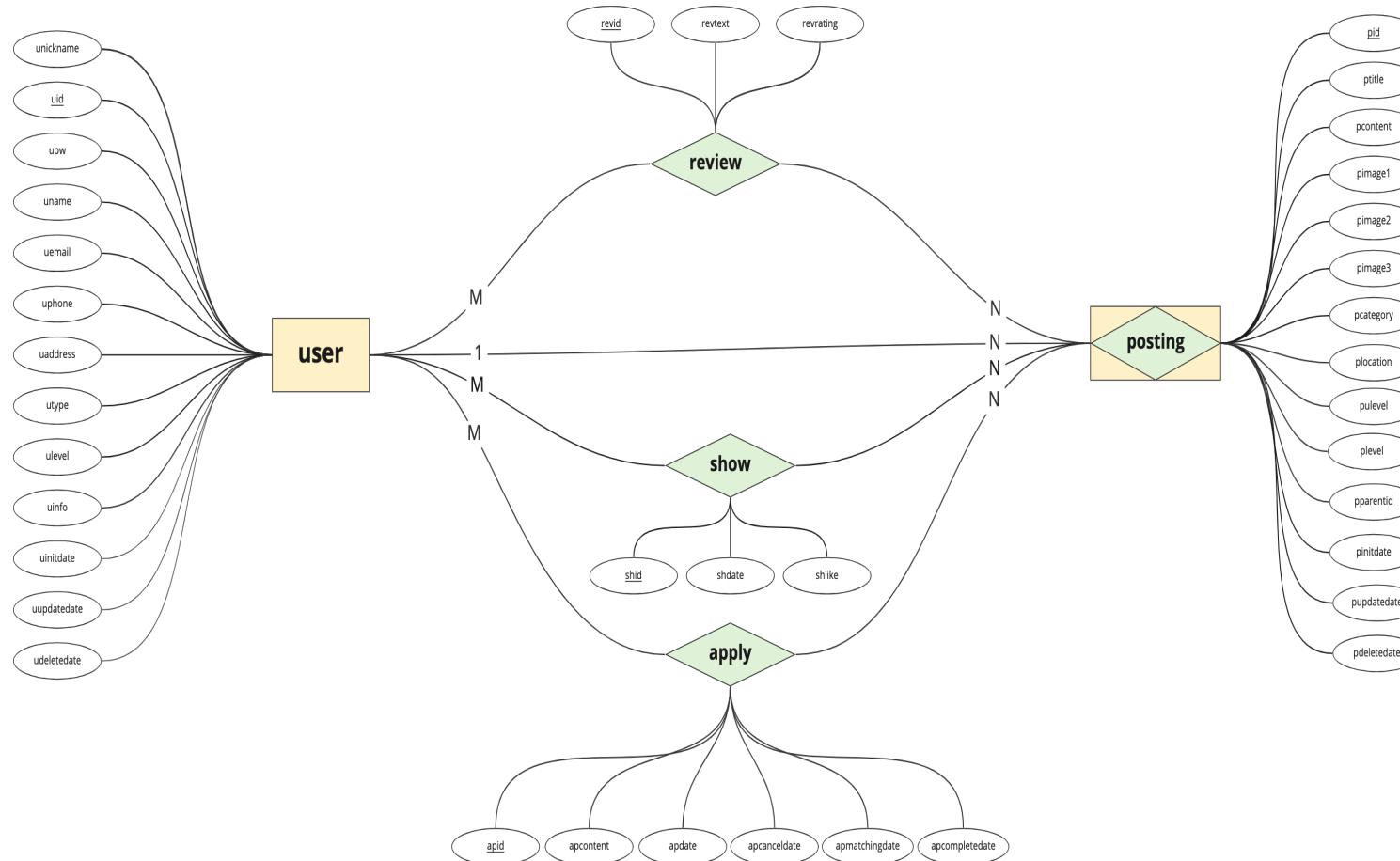
$(currentBlock - 1) * pageLength + 1$

endPage : block의 끝 페이지

$startPage + pageLength - 1$

ERD - Petmily Project

전체 ERD 중 Project의 핵심인 유저와 게시글 Entity 사이의 Relation Diagram



Entity

- user : 유저
- posting : 게시글

Relation

- **posting**
유저가 게시글을 작성한다는 관계를 가지기에 entity 이자 relation으로 표시
- **review / show / apply**
후기글과 평점 / 조회수와 좋아요 / 신청글은 유저와 게시글 사이의 관계

Attribute

- 게시글 삭제시 DELETE 문 사용하지 않고, UPDATE 문으로 delete date를 UPDATE한다.