

기획력과 리더십을 겸비한 개발자, 권순형입니다.

최종학력

서울시립대학교 전자전기컴퓨터공학부

자격증

SQLD

git

<https://github.com/SunHyongKwon>

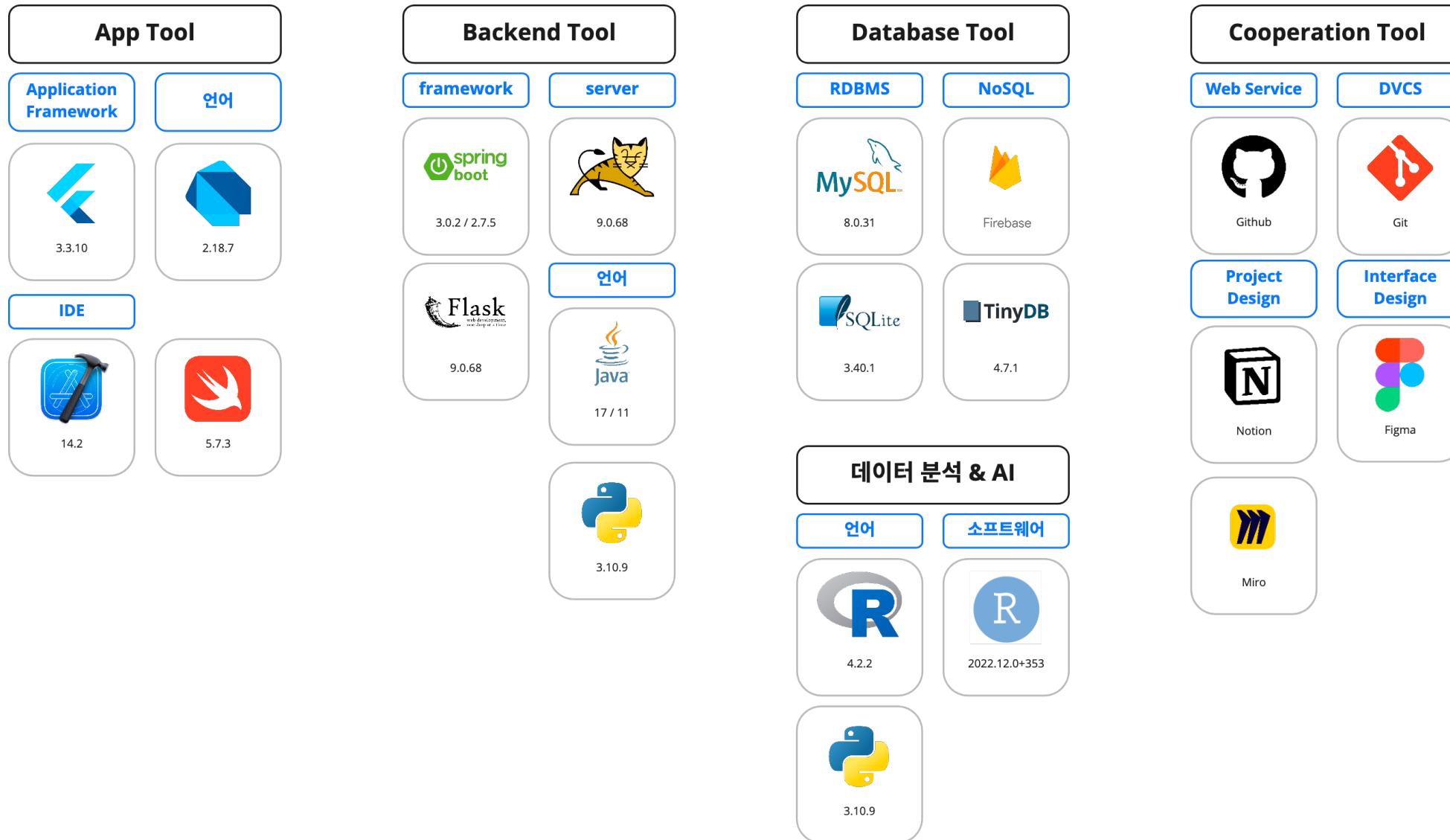
notion

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

목 차

1 사용한 프로그램	3	4 앱에 사용될 AI 모델링	19
2 App 프로젝트	4	4.1 서울시 전세 금액 예측 모델	20
2.1 솜솜 하우스 - 부동산 정보 및 예측 어플	5	4.2 서울시 상권 분기당 매출 금액 예측 모델	22
2.2 성연(星演) - 소상공인을 위한 어플	7	5 앱에 사용될 Backend 구현	24
2.3 가계부자 - 간편하게 사용가능한 가계부 어플	9	5.1 RestAPI 구현	25
3 Flutter	11	5.2 기능 구현	27
3.1 위젯 분리	12	6 기타 기술 스택	30
3.2 자체 위젯 만들기	13	6.1 데이터 모델링	31
3.3 SQLite	14	6.2 Scraping	32
3.4 restAPI 활용	15		
3.5 openAPI 활용	18		

사용한 프로그램



App 프로젝트

목 차 | 1 솜솜 하우스 - 부동산 정보 및 예측 어플

1.1 [프로젝트 개요](#) 5

1.2 [System Flow Diagram](#) 6

2 성연(星演) - 소상공인을 위한 어플

2.1 [프로젝트 개요](#) 7

2.2 [System Flow Diagram](#) 8

3 가계부자 - 간편하게 사용가능한 가계부 어플

3.1 [프로젝트 개요](#) 9

3.2 [System Flow Diagram](#) 10

솜솜 하우스

프로젝트 개요



주제

서울 전월세 정보 조회 및
전세 가격 예측 서비스 제공

주제 선정 이유

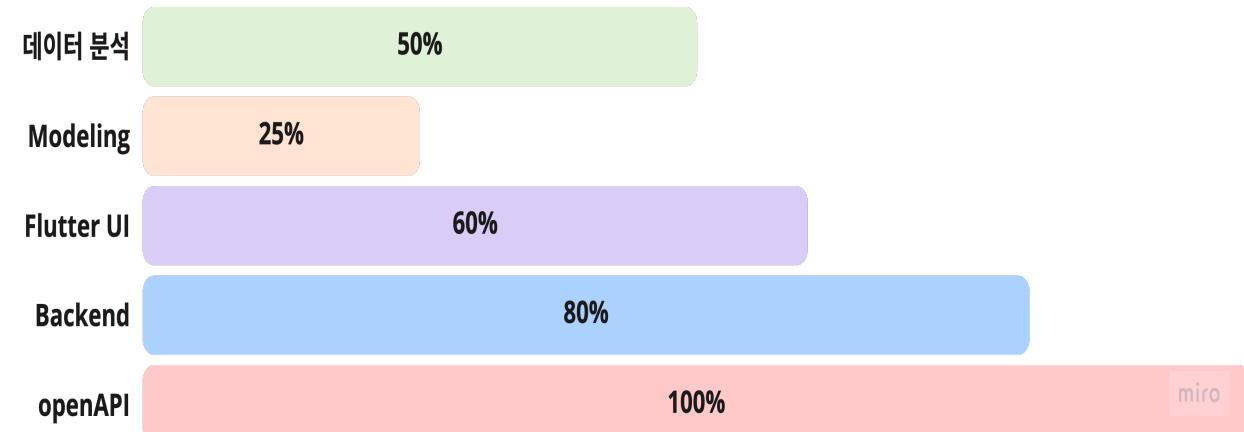
치솟는 부동산 가격으로 인한 주거안정에 대한 불안감과
부동산 문제가 인생 전반의 계획에 큰 부분을 차지하기에
부동산 전세 가격 예측과 전월세 정보를 제공함으로써
주거안정에 대한 불안감 해소와
인생 계획에 도움이 되고자 주제를 선정했습니다.

Benchmark App

- | 직방
- | 호갱노노

- | 다방
- | 아파트 실거래가

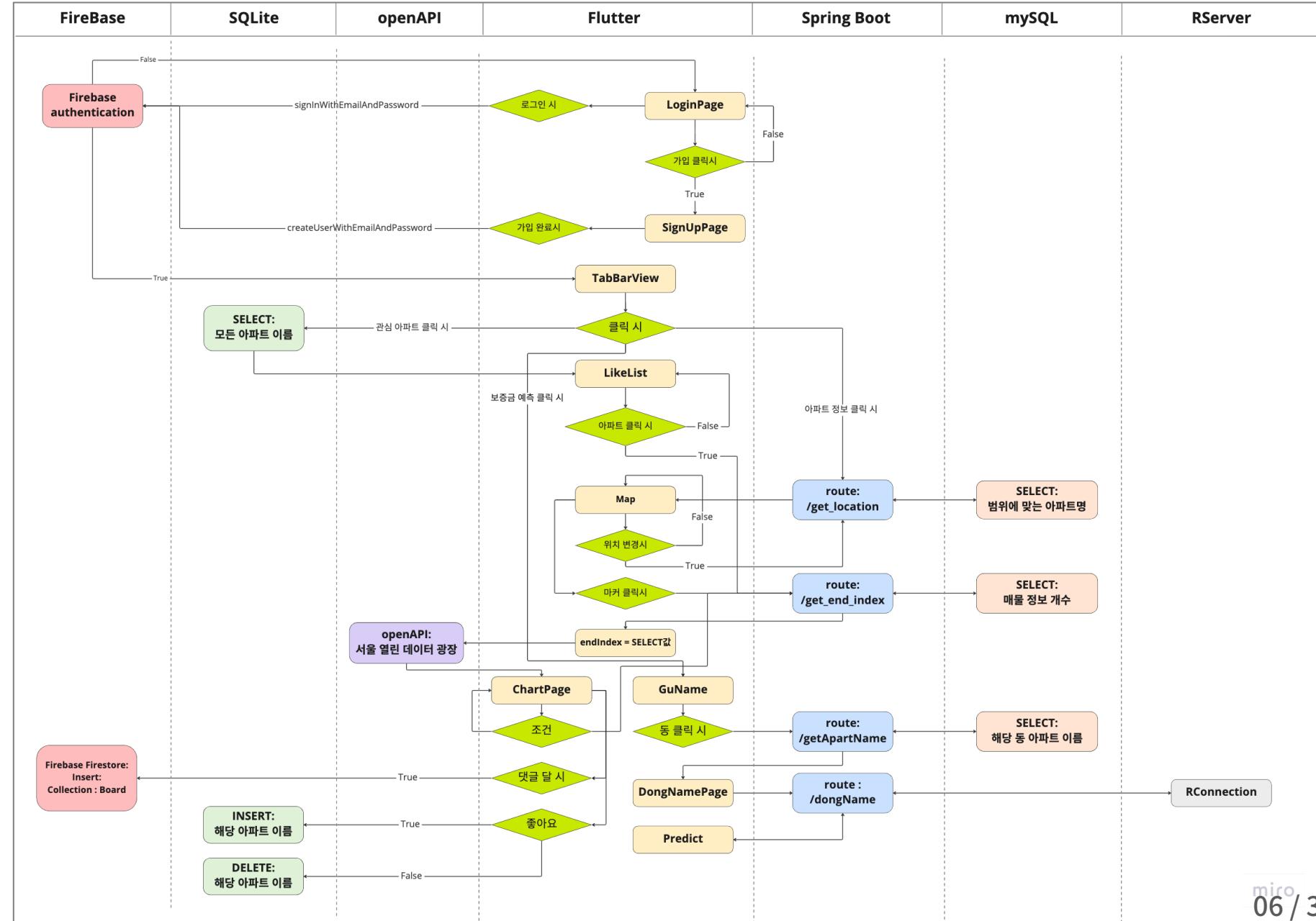
담당 역할



솜솜 하우스

System Flow Diagram

- Flutter 이용한 앱 UI 디자인
 - SQLite를 이용하여 Device 내부 DB 구현
 - Spring Boot를 이용하여 mySQL DB 연동과 RServe 연결하는 Backend 구성
 - Firebase 이용하여 Login ,RealTime DB 구축
 - openAPI 이용하여 필요한 데이터 가져오기



성연 (星演)

프로젝트 개요



주제

예비 자영업자를 위한
상권 분석과 매출 예측 서비스 제공

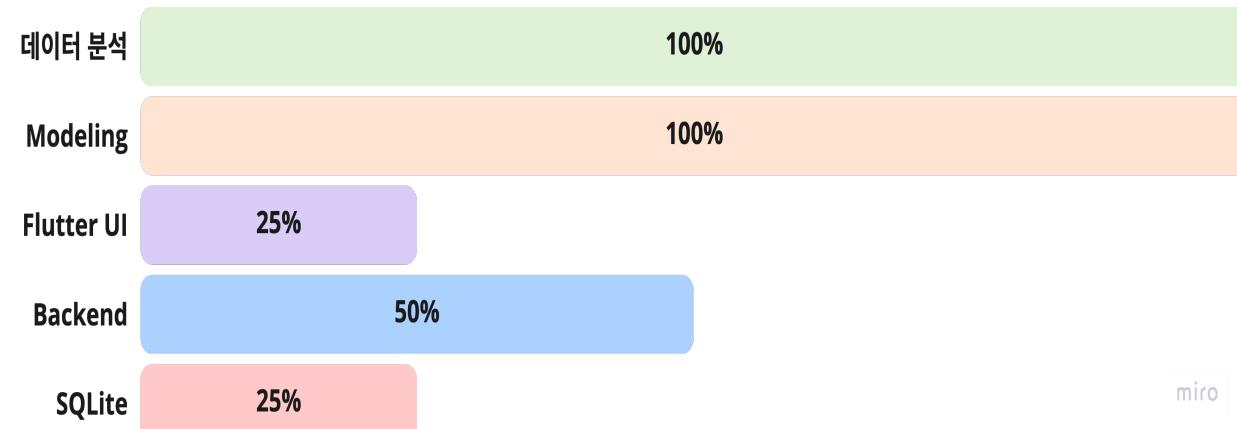
주제 선정 이유

전체 취업자 중 자영업자의 비율이 25% 정도입니다.
위와 같은 이유로 많은 사용자를 타겟으로 할 수 있고,
자영업의 경우, 상권에 따른 매출의 차이가 크다고 알고 있습니다.
이를 분석하여 정보를 제공함으로써 예비 자영업자한테
의미 있는 지표로 사용할 수 있을 것 같습니다.

프로젝트 화면 구성

- | 상권 분석 지도
- | 식재료 원가 정보
- | 매출 예측 및 시뮬레이션
- | 수입 지출 관리 가계부

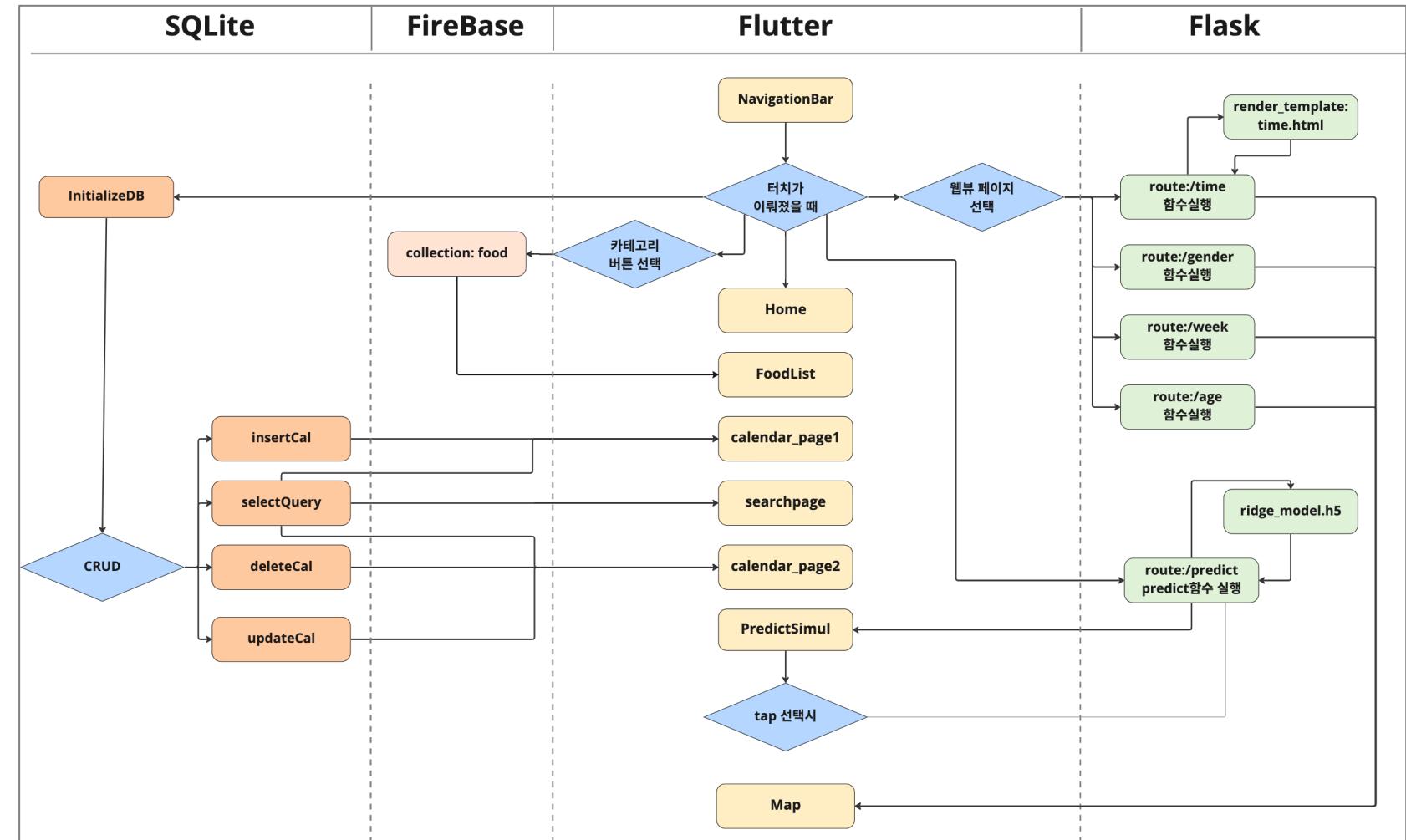
담당 역할



성연 (星演)

System Flow Diagram

- Flutter를 이용하여 앱 UI 디자인
- SQLite를 이용하여
Device 내부 DB 구현
- Flask를 이용해 AI 모델과 연동하여
매출액 예측 금액을 Flutter UI에 표시
- Firebase를 이용하여 crawling한
데이터를 Flutter 화면에 표현



가계부자 - app store 배포 완료

프로젝트 개요



주제

회원가입없이 간편하게
사용할 수 있는 가계부

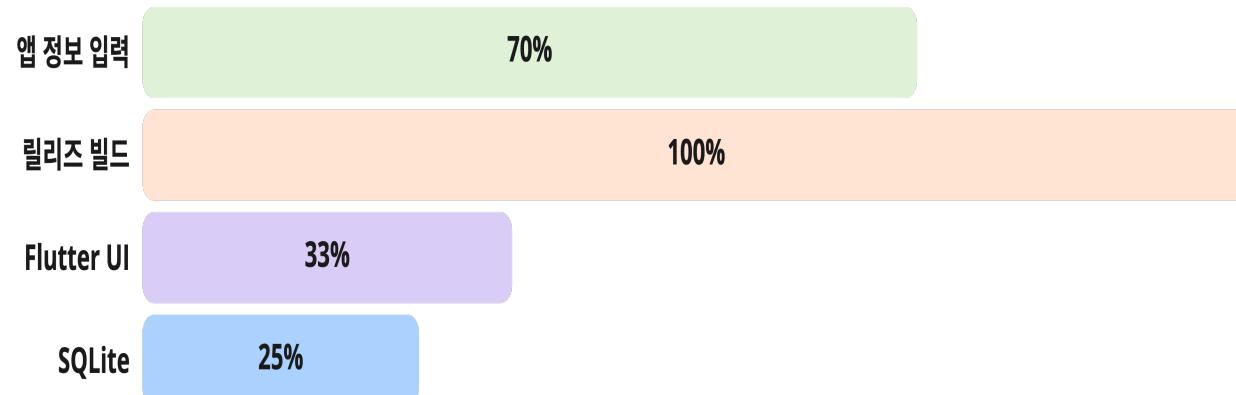
주제 선정 이유

성연 프로젝트에서 사용한 가계부가 SQLite만을 사용하여서
이를 개인이 사용할 수 있도록 카테고리 항목 추가와
일별, 카테고리별, 제목 별 검색 기능을 추가하여
app store에 배포하는 프로젝트를 추가적으로 진행하였습니다.

프로젝트 화면 구성

- | 캘린더 및 기록 추가 화면
- | 카테고리, 제목 별 검색 화면
- | 기간 검색 및 수정 삭제 화면

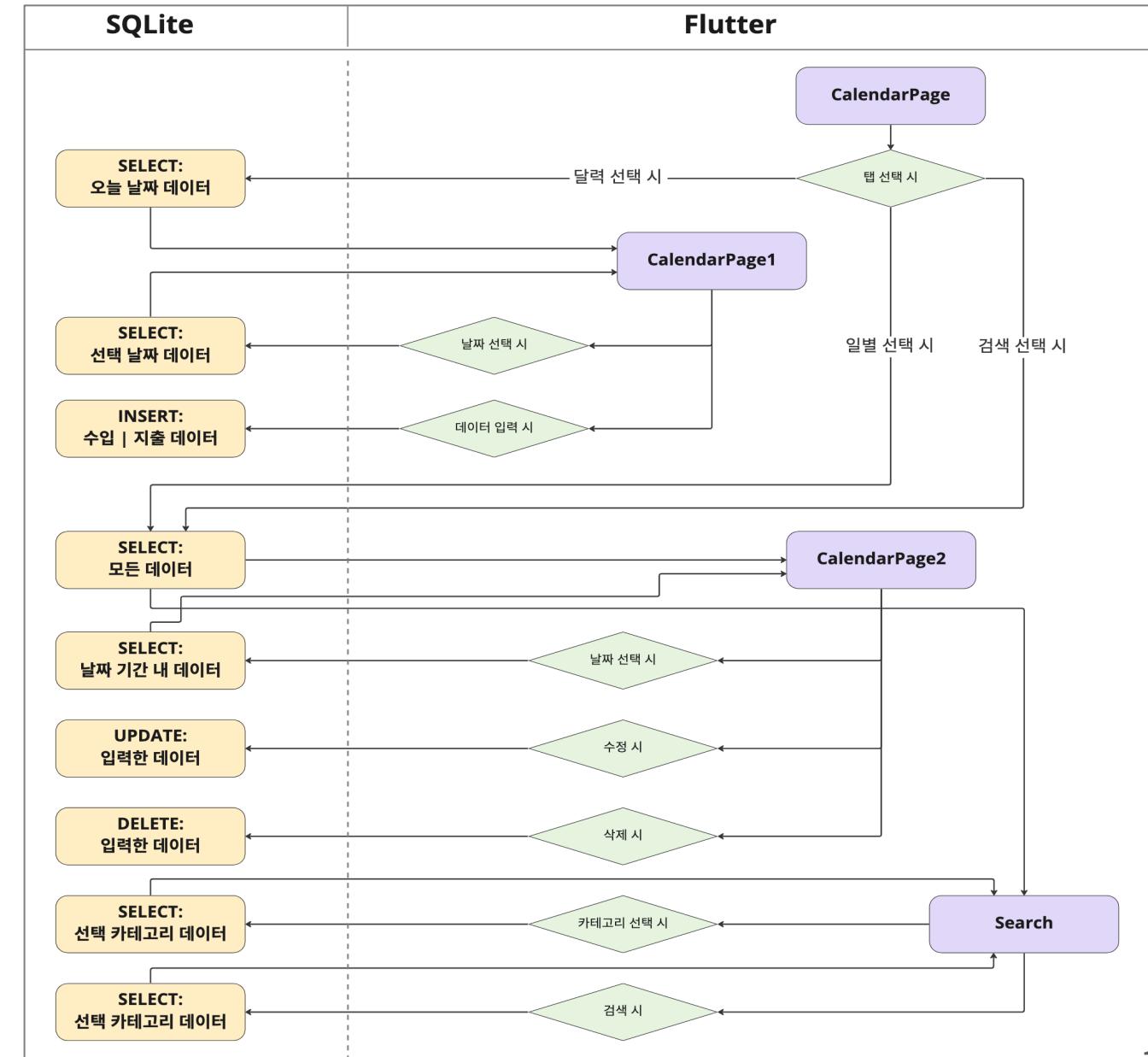
담당 역할



가계부자 - app store 배포 완료

System Flow Diagram

- Flutter를 이용하여 앱 UI 디자인
- SQLite를 이용하여 Device 내부 DB 구현



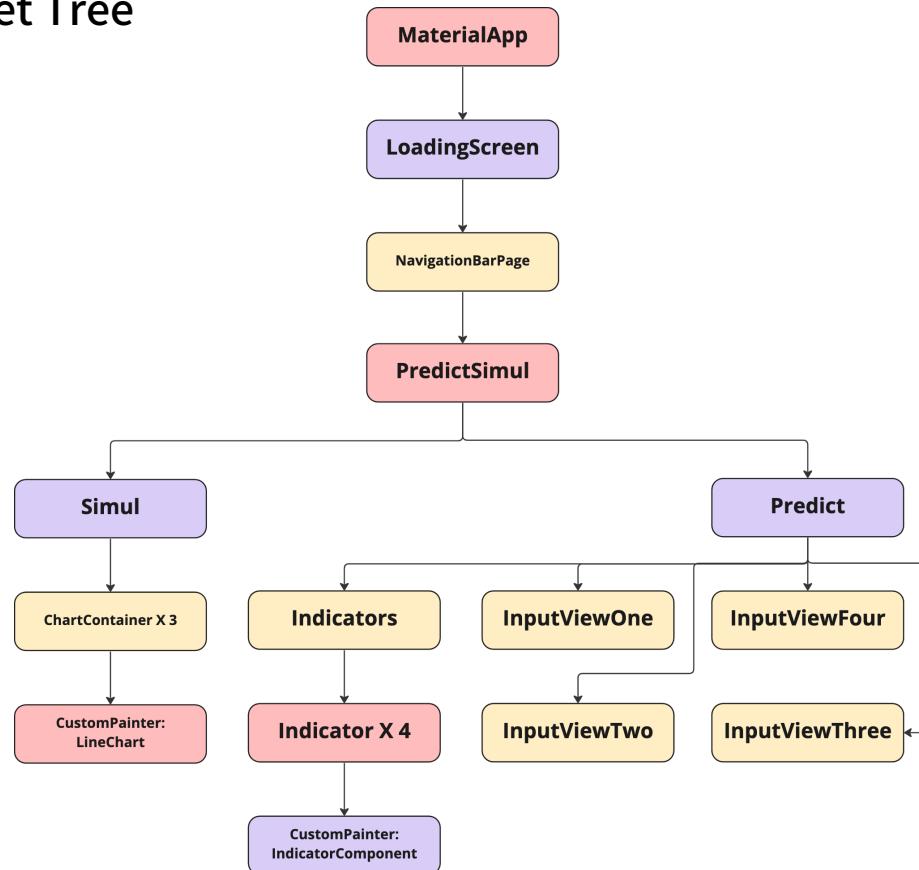
Flutter

목 차 	1	<u>위젯 분리</u>	12
	2	<u>자체 위젯 만들기</u>	13
	3	<u>SQLite</u>	14
	4	<u>restAPI 구현 및 활용</u>	
	4.1	<u>SpringBoot + Rserver</u>	15
	4.2	<u>SpringBoot + mySQL</u>	16
	4.3	<u>Flask + AI</u>	17
	5	<u>openAPI 활용</u>	18

위젯 분리

StreamController를 이용하여 다른 클래스 간에 데이터를 주고 받을 수 있는 기능을 구현

Widget Tree



- 위젯 간의 분리를 통해 유지 보수 시 어느 부분에서 오류가 났는지 쉽게 파악할 수 있다.
- 한 화면을 여러 위젯으로 분리를 함으로써 한 화면 사이에서 데이터 교환이 필요할 때 클래스 간의 데이터가 필요한 경우가 발생
- 그 때에는 StreamController를 이용하여 controller의 listen 함수와 controller.stream의 add 함수를 통해 변화가 생기는 위젯에는 listen 함수를 변화를 주는 위젯에는 add 함수를 줘서 다른 클래스여도 데이터를 주고 받을 수 있게 하였다.

자체 위젯 만들기

CustomPainter를 이용한 차트 그리기 및 진행 상태 바 만들기

차트 그리기



- flutter CustomPainter를 이용한 차트 그리기
 - 크기가 있는 값을 리스트로 만든다.
 - 크기는 MinMaxScaler 공식을 이용하여 0 – 1 사이로 변환시킨다.
 - 리스트의 index와 값을 이용하여 Offset 변수를 만든다.
 - 이를 Path class의 moveTo, lineTo 를 이용하여 선을 만든다.
 - canvas.drawPath를 이용하여 차트로 그린다.

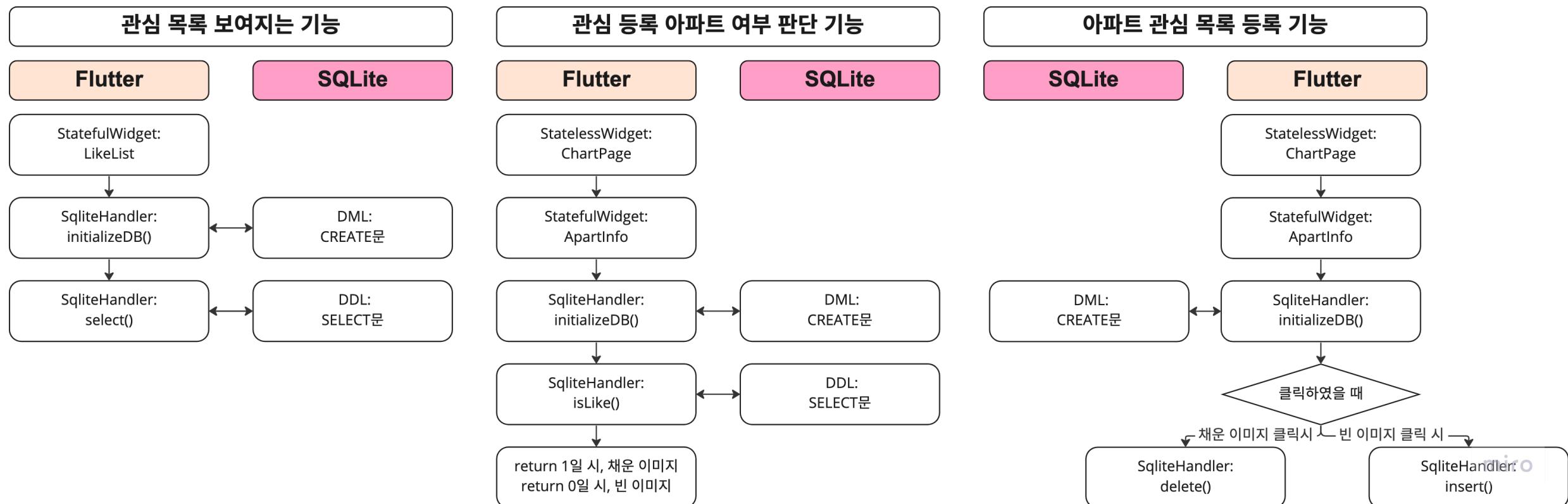
진행 상태 바 만들기



- CustomPainter를 이용하여 상태 바를 만들고 stack으로 4개의 화면을 구성하여 Visibility의 visible 값을 action이 일어날 때마다 변하도록 list를 만든다.
- 이 값에 맞춰서 상태바의 color 값을 변하도록 리스트를 구성했다.

SQLite - 관심목록

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

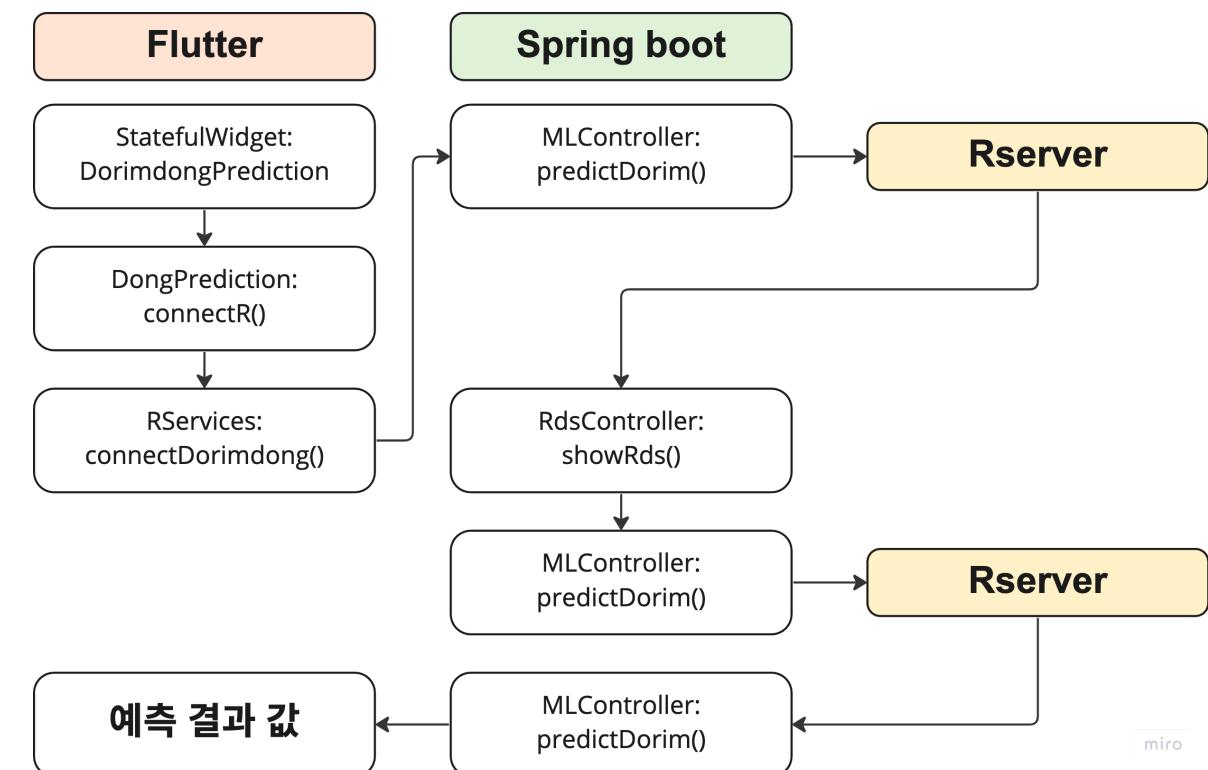


RestAPI 활용

Web Server의 rds파일을 가져와서 Spring Boot에서 Rserver와 연결하여 예측결과를 불러오는 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

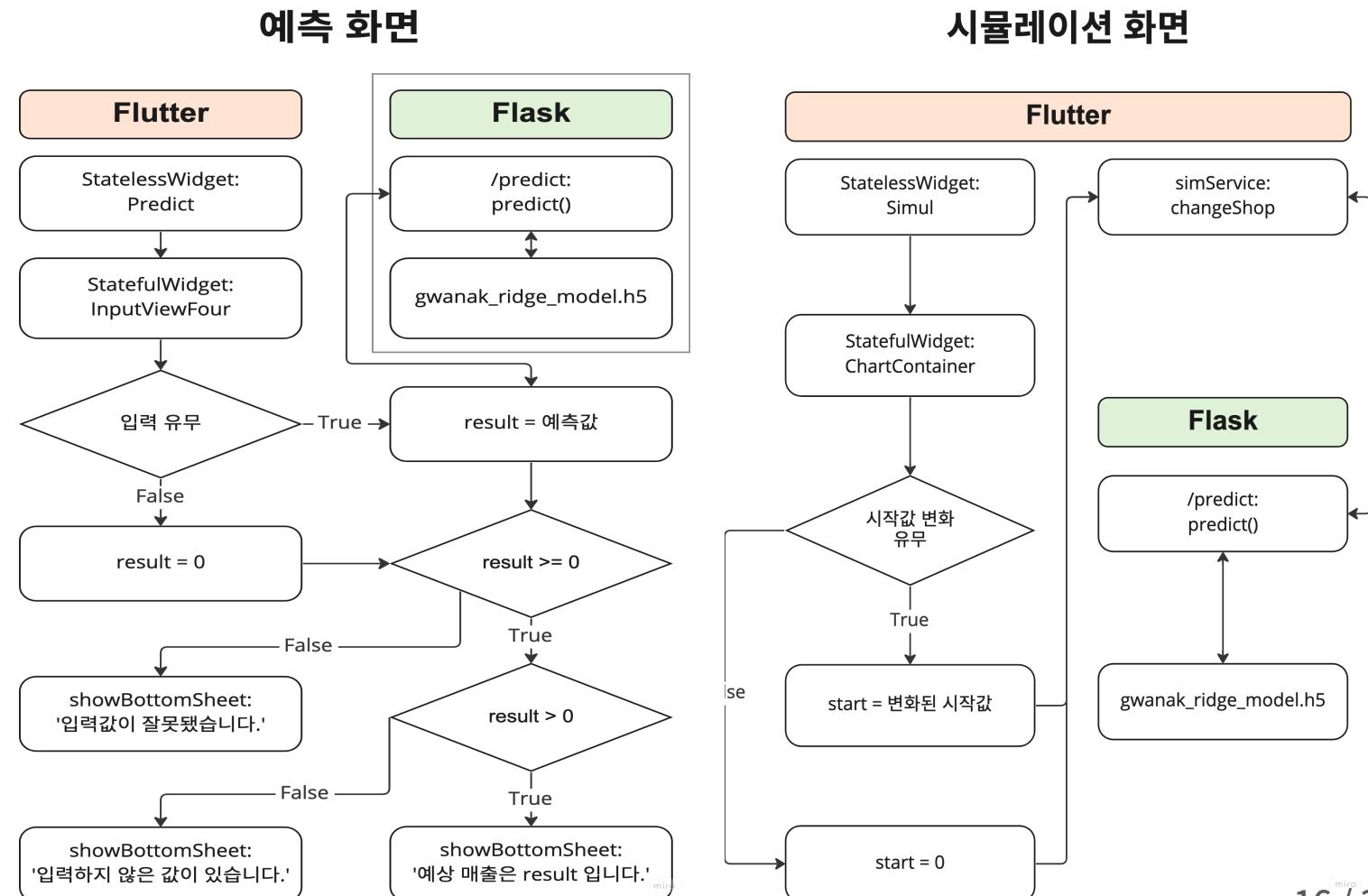
```
String num1NeckStr = num1Neck != 0 ? '$num1Neck억 ' : '';
String num1TrashStr = num1Trash != 0 ? '$num1Trash만원' : '';
String num2NeckStr = num2Neck != 0 ? '$num2Neck억 ' : '';
String num2TrashStr = num2Trash != 0 ? '$num2Trash만원' : '';
```



RestAPI 활용

parameter 값을 받아 Web Server의 h5 파일을 이용해서 예측결과를 flutter 화면으로 불러오는 기능

- flutter의 http 패키지를 이용하여 flask 서버 주소를 불러온다.
 - async와 await를 이용하여 비동기 처리를 한다.
- parameter 값을 feature 값에 맞게 바꿔준다.
 - one-hot encoding으로 추가된 컬럼에 맞게
 - feature engineering으로 추가된 컬럼에 맞게
- 머신러닝 모델을 저장해둔 h5 파일을 불러온다.
- 각색한 feature들을 순서에 맞춰서 머신러닝 모델에 넣어 예측값을 받는다.
- 예측값을 json 형태로 만들어서 flutter로 보낸다.

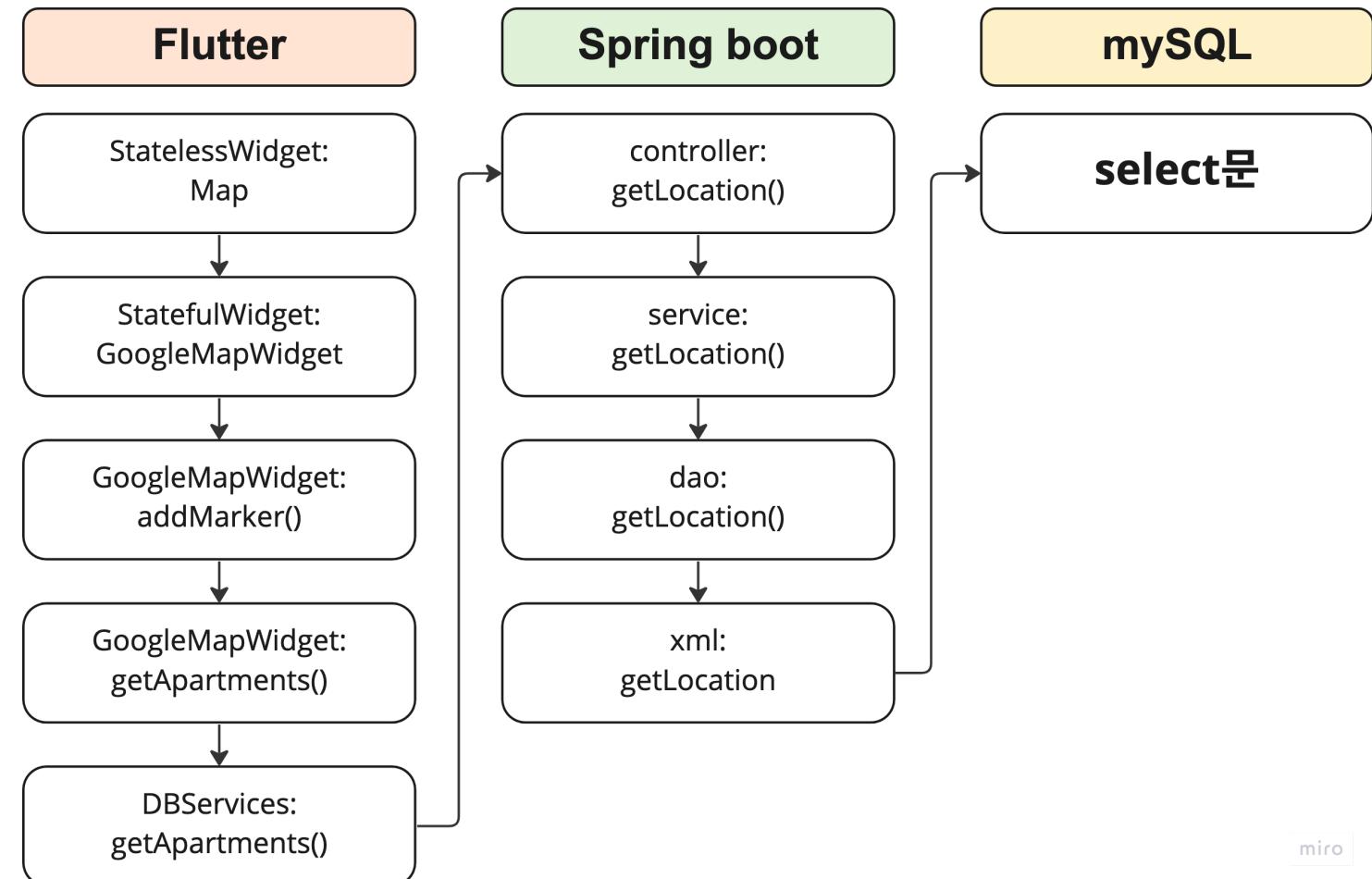


RestAPI 활용

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.
- 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}
```



openAPI 활용

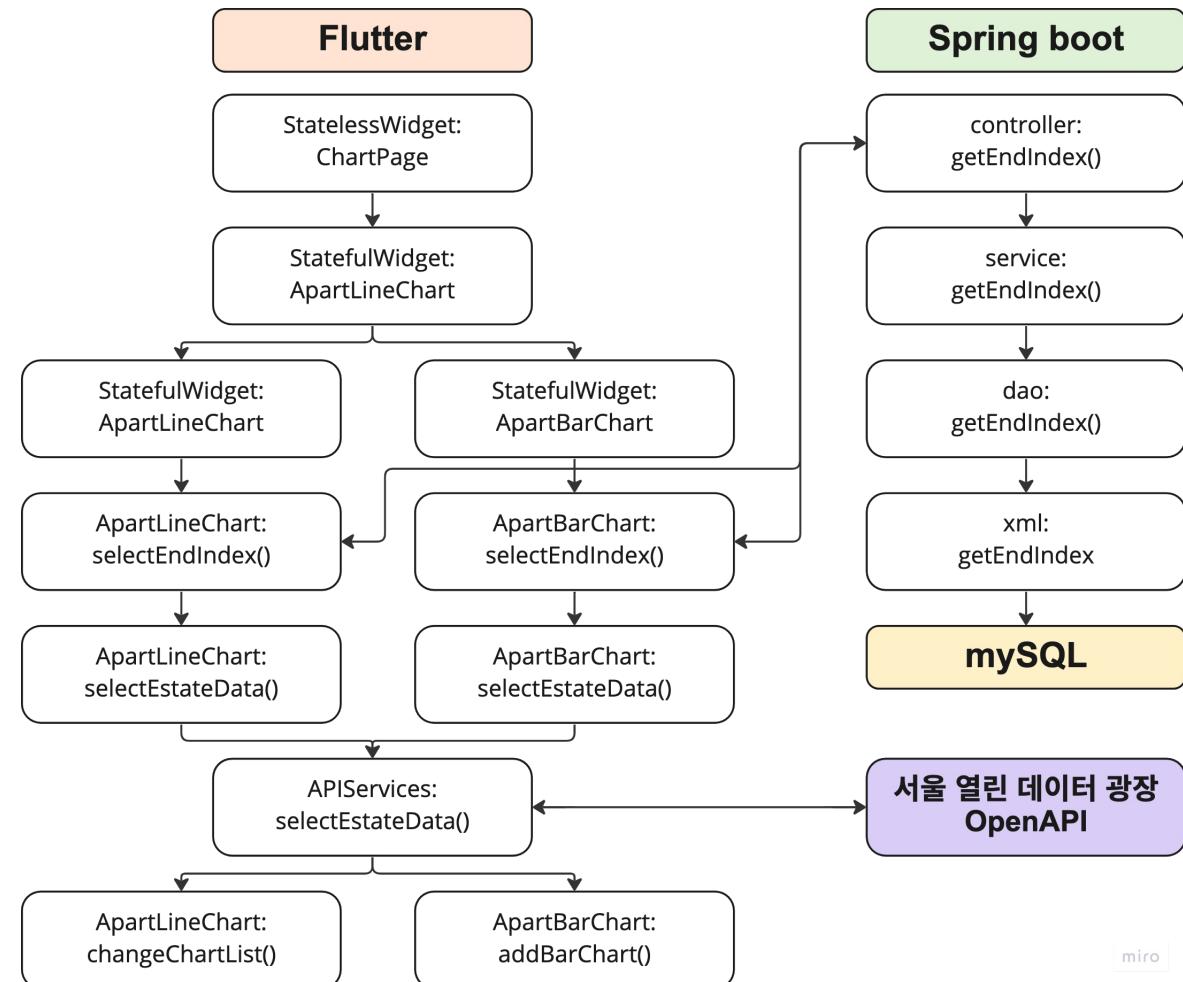
서울 열린 데이터 광장 openAPI로 JSON 구조의 데이터를 받아서 fl-chart 패키지 이용하여 차트로 표현

- 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.



AI 모델링

목 차 | 1 서울시 상권 분기당 매출 금액 예측 모델

1.1 [EDA](#) 20

1.2 [Modeling](#) 21

2 서울시 전세 금액 예측 모델

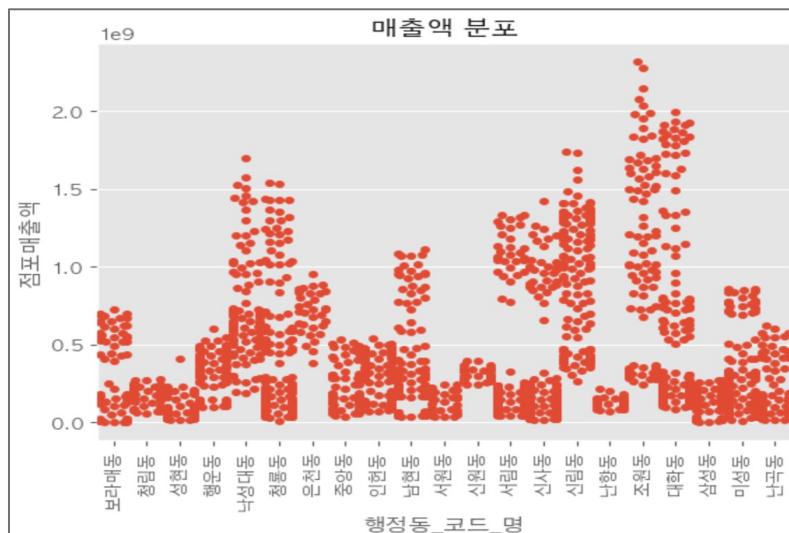
2.1 [EDA](#) 22

2.2 [Modeling](#) 23

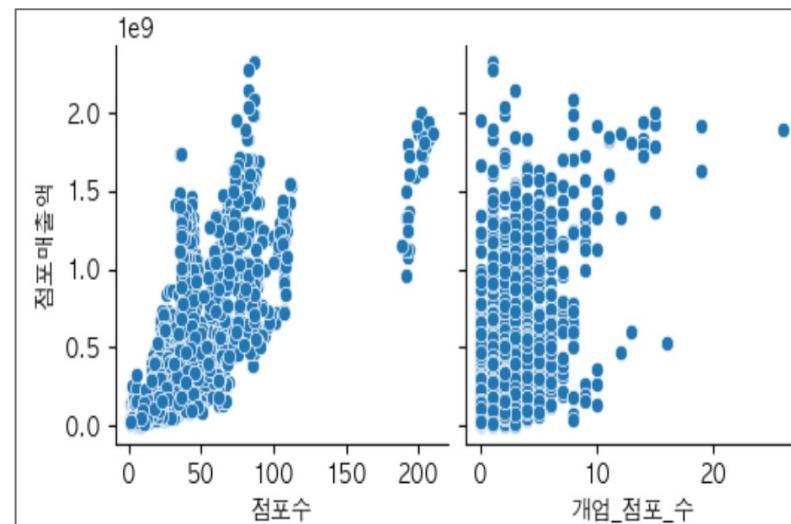
EDA

서울시 상권 데이터 중 머신 러닝 지도 학습의 회귀 모델의 feature로 사용 가능한 column 찾기

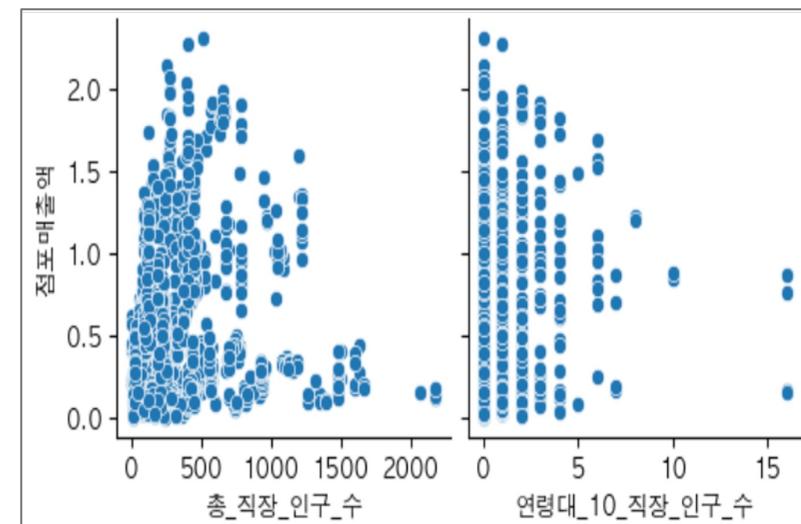
동 이름



점포 수 관련



직장 인구 수 관련



분석

동 별로 매출액의 분포가 다르다는 것을 확인할 수 있습니다.
점포 수, 개업 점포 수, 직장 인구 수, 10대 직장 인구 수가
매출액과 선형적인 관계를 가지는 것을 확인 할 수 있다.

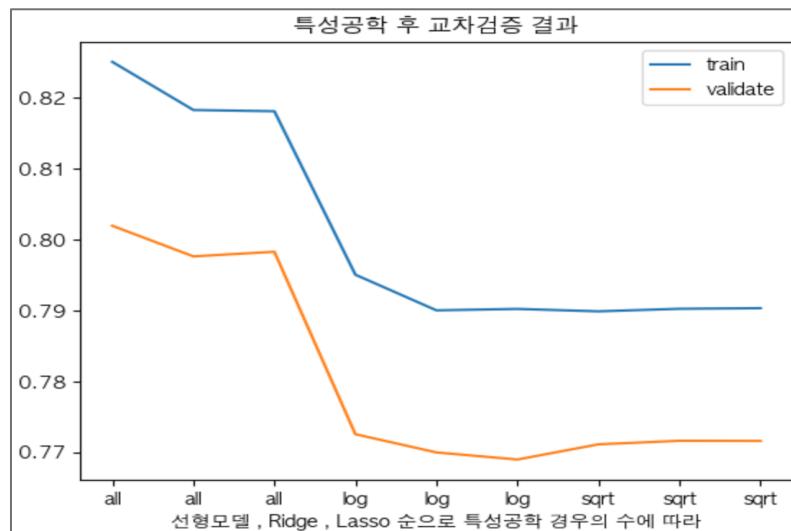
결론

선형적인 관계를 가지는 column별 매출액과의 분포가 다르고,
동 이름별로도 매출액의 분포가 다르기 때문에
위 5개의 column을 feature로 사용할 수 있다고 판단했습니다.

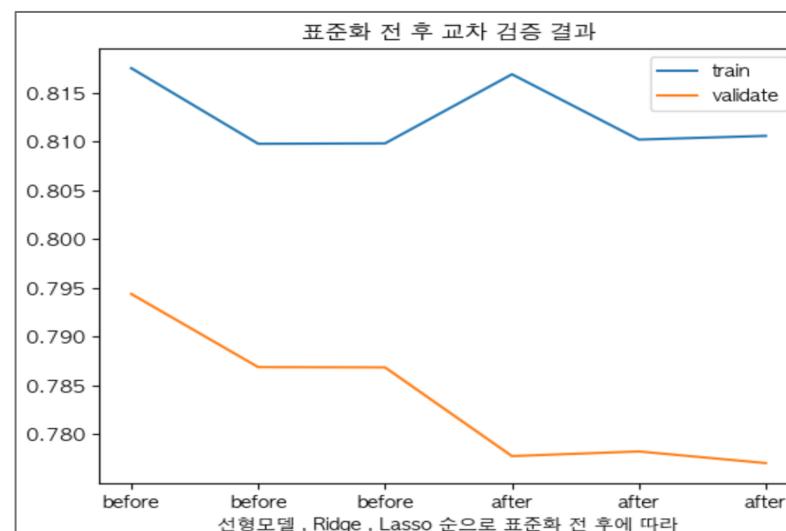
Modeling

표준화, 특성공학, 교차 검증을 통해 예측률을 높이고 grid search를 통해 일반화된 머신 러닝 모델 만들기

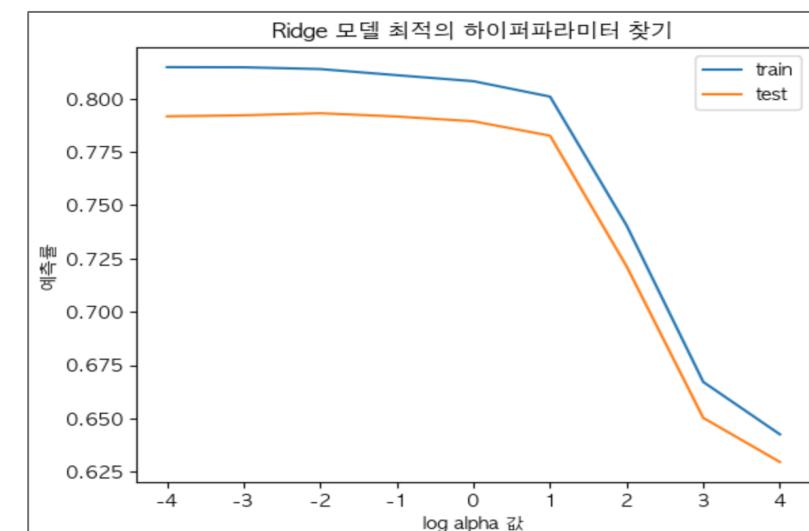
특성공학 후 교차 검증 결과



표준화 전 후 교차 검증 결과



최적의 hyper parameter값 찾기



분석

특성공학 한 것을 모두 사용했을 때 교차 검증결과가 가장 높습니다.
표준화 전후의 train 데이터의 예측률은 비슷하지만,
표준화 전의 validate 예측률이 더 높습니다.

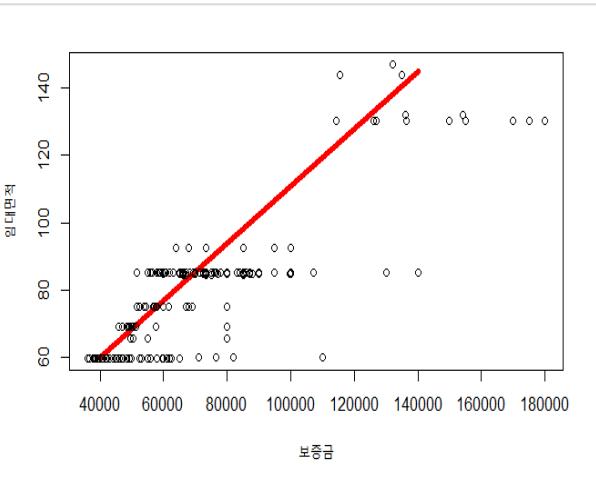
결론

특성공학 한 것을 모두 사용하고, 표준화 하지 않았을 때의 데이터를 가지고
Ridge 모델의 alpha 값을 10으로 했을 때가 예측률이 가장 높은 모델이고,
이를 매출 예상 머신 러닝 모델로 사용하기로 했습니다.

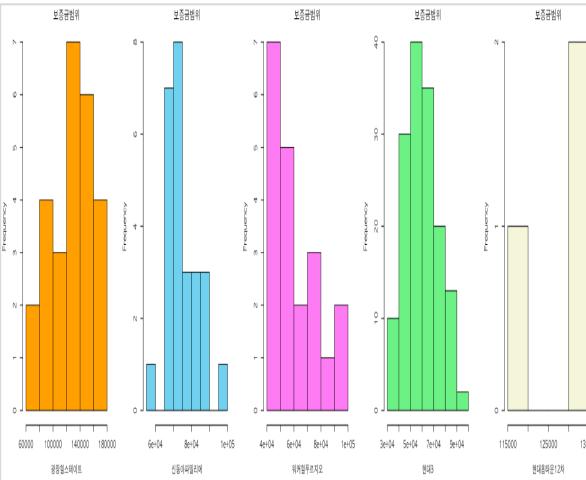
EDA

서울시 전세 데이터 중 머신 러닝 지도 학습의 분류 모델의 feature로 사용 가능한 column 찾기

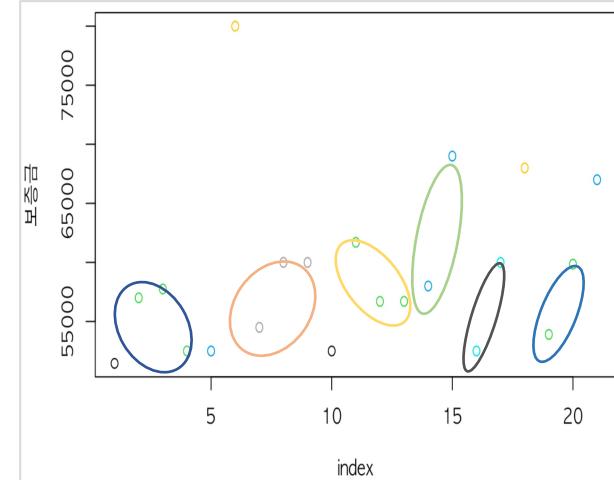
임대 면적



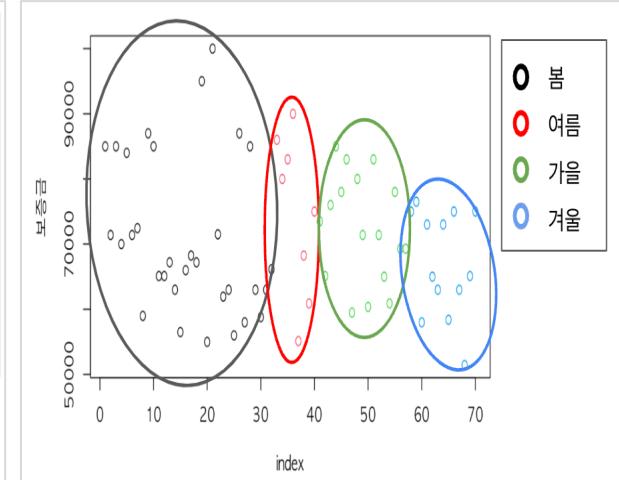
아파트 명



층 수



계약 계절



분석

보증금과 각 부제목과의 관계를 시각화 시킨 것입니다.

임대면적은 보증금과 선형적인 관계를 가진다는 것을 확인 할 수 있습니다.

나머지는 값들 별로 보증금의 분포가 다르다는 것을 확인 할 수 있습니다.

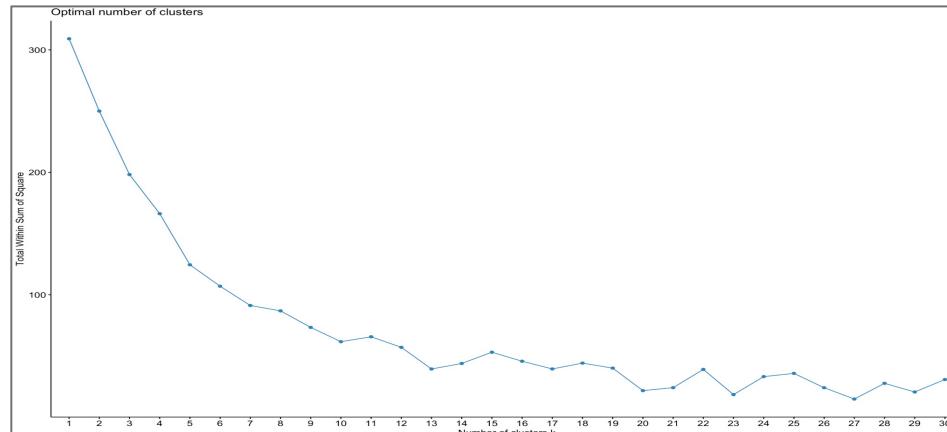
결론

위 4개의 column이 feature로서 적합하다고 판단했습니다.

Modeling

4개의 feature를 clustering 하여 label별 보증금 금액의 범위를 target으로 하여 모델링

Clustering



결론

elbow method를 통해 군집의 개수를 20으로 하기로 결정

Modeling

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

결론

feature를 정규화 한 후

인공신경망 모델을 사용할 경우의 예측률이 제일 높으므로
해당 모델을 머신 러닝 모델로 채택하였습니다.

Backend

목 차 | 1 RestAPI 구현

1.1 [Spring Boot](#) 25

1.2 [Flask](#) 26

2 기능 구현

2.1 [멀티 파일 업로드](#) 27

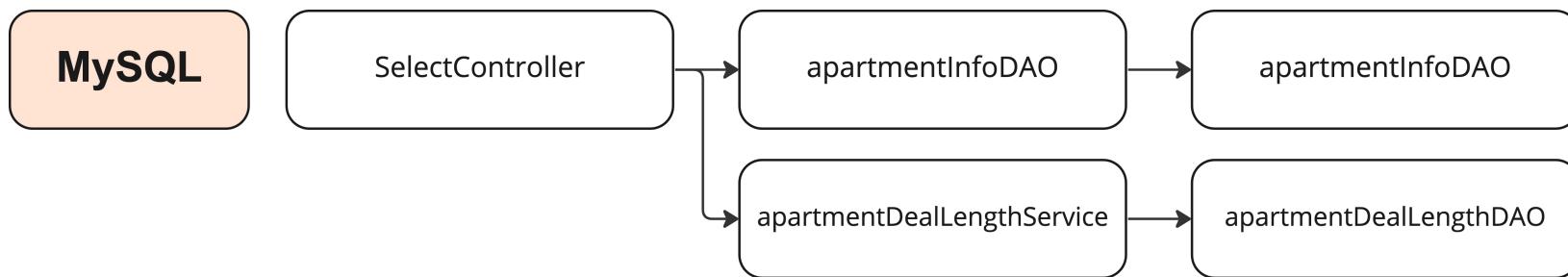
2.2 [파일 다운로드](#) 28

2.3 [이메일 전송](#) 29

RestAPI 구현 - SpringBoot 이용

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) +
    + )))))");

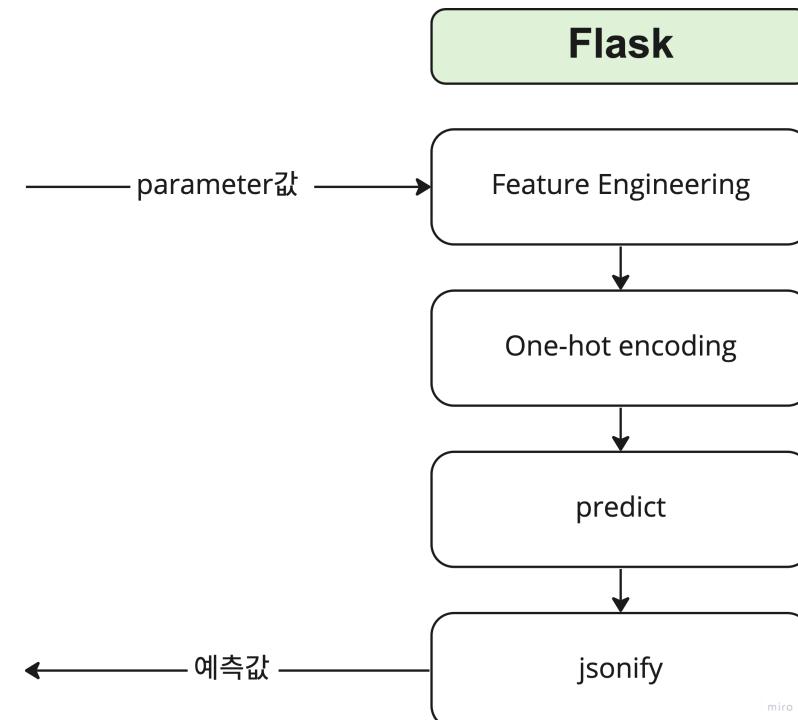
String result = conn.eval("result").asString();
  
```

Rserver 연결 후, r code를 작성하여 result 값을 받는 코드

RestAPI 구현 -Flask 이용

Flask를 이용하여 AI 모델에 데이터를 넣어서 예측 값을 리턴하는 Backend 구현

- Flutter로부터 받은 parameter 값을 AI 모델의 데이터로 넣기 위해서는 약간의 각색이 필요하다.
- scikit-learn을 이용하여 특성공학을 진행한다.
- 범주형 데이터의 경우에는 one-hot encoding을 진행한다.
- 각색한 데이터와 parameter로 받은 데이터를 합쳐서 AI 모델의 데이터로 넣어 예측값을 받는다.
- 이를 json구조로 만들어서 반환해준다.



기능 구현 - 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
            // 패스에 "name" 으로 saveFile을 만들 빈 컵데기를 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 이름이 있을 수 있기에 이를 방지하기 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.

기능 구현 - 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.

기능 구현 - 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;
}
```

기타 기술 스택

목 차 | 1 데이터 모델링

1.1 [ERD](#) 31

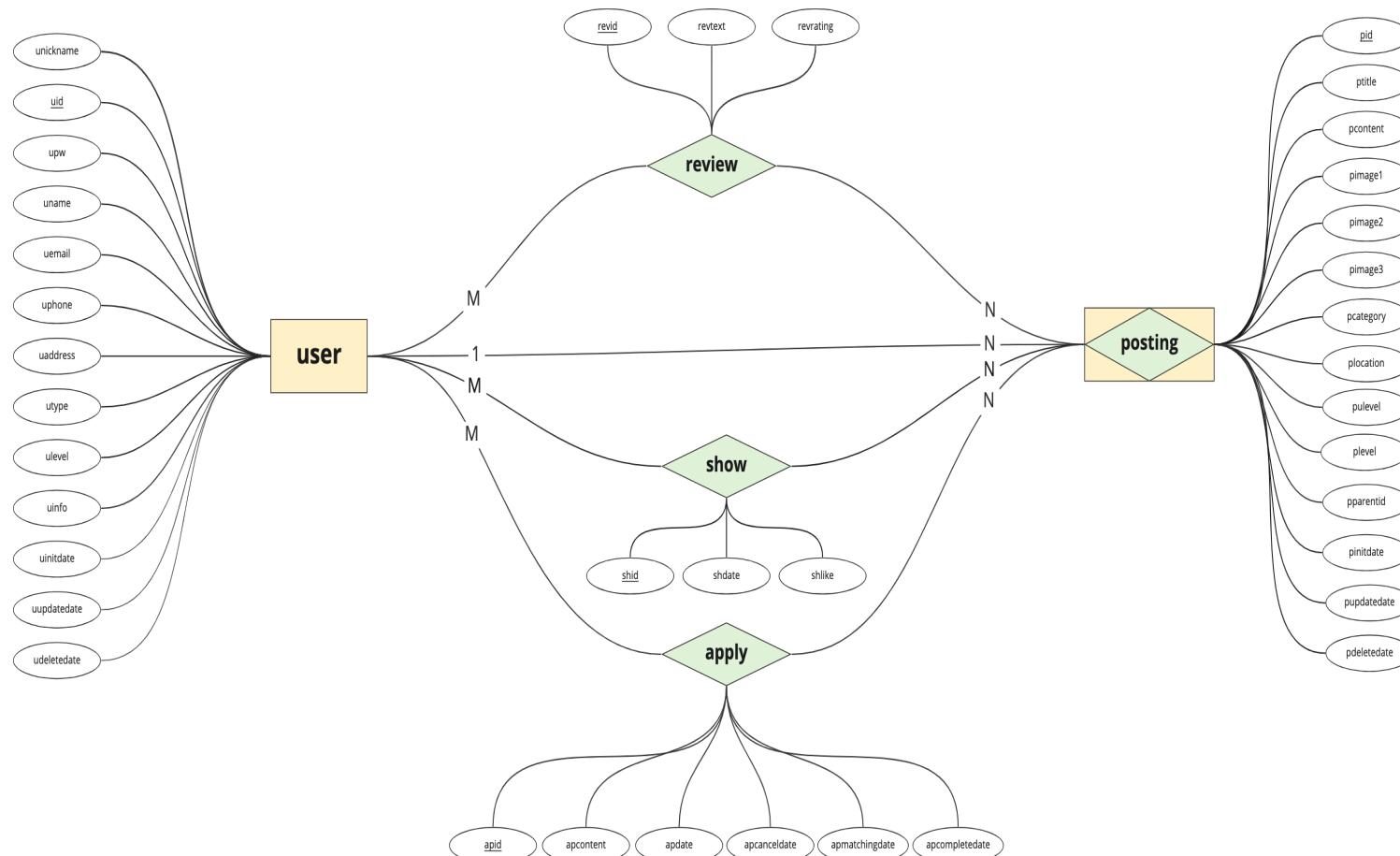
2 Scrapyng

2.1 [selenium을 이용한 Scrapyng](#) 32

2.2 [multi processing](#) 33

ERD

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.

Selenium & BeautifulSoup

올리브영 전국 매장 이름과 주소 scraping

1

```
scroll_bar = event.find_element(By.CSS_SELECTOR, 'div[class="sroll_store scrbar 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.

MultiProcessing Scrapping

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

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.

```

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")

    time.sleep(20)

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

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

            time.sleep(1)

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

            time.sleep(1)

            # selenium와 BeautifulSoup soup 이용해서 텍스트 정보 가져오기
            html = driver.page_source
            soup = BeautifulSoup(html, 'html.parser')

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

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

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

            # 디테일 정보하기
            image = driver.find_element(By.XPATH,'//a[@id="detail"]//div/div/div[1]/div')
            url = "https://gajok.kr"+image.get_attribute('style').split(':')[0][23:-2]

            # 콤마 제거하기
            url=url.replace(",","%2B")

            # 웹크리에이터
            url_ko=re.compile('([-=]+|-|[-=]+)*',findall=url)

            if not url_ko == None:
                for ko in url_ko:
                    enc_ko=parse.quote(ko)
                    url=url.replace(ko,enc_ko)

            images.append(url)
            print(f"이미지주소 : ",url)

            saveName = f'{i}_Image\{i}.jpg'
            resp.urlretrieve(url, saveName)

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

        except:
            print('error')

    return names,addresses , phones , images

```

multi process로 실행될 함수