목차

1. 요구조건 및 제약 사항 분석에 대한 수정사항

1-1. 과제목표

이번 졸업과제의 목적은 사용자에게 맛집 정보 및 추천 서비스를 제공하는 것이다. 기존 타 플랫폼의 맛집 서비스들을 분석해봤을 때, 해당 플랫폼의 사용자들이 남기는 데이터에 의존하거나 수많은 광고가 존재하는 것을 확인할 수 있었다. 이를 위해 조금 더 정확한 정보를 사용자에게 제공하기 위해 Youtube를 이용하게 되었다.

그림 1 - 웹 사이트 방문자 수

그림 2 - 분당 생성되는 Data 크기

ㄱ 그림 1과 같이 현재 우리나라에서 가장 많은 누적 방문자 수를 보유하고 있는 플랫폼은 Youtube이다. 또 그림 2의 1분당 생성되는 예상 데이터를 보면 약 500시간 분량으로 많은 데이터가 Youtube에 갱신되고 있는 것을 확인할 수 있다. ㄴ

Youtube에는 사용자들이 다양한 맛집에 다녀온 이후 남기는 솔직한 리뷰가 영상으로 존재한다. Youtube 플랫폼에 존재하는 Raw Media Data에서 Meta Data를 추출 및 가공함으로 사용자들에게 높은 신뢰도를 줄 수 있는 맛집 검색 및 추천 서비스를 제공하도록 한다.

1-2. 요구사항

요구사항은 1. 웹 서비스 디자인, 2. DB, 3. 데이터 수집 및 처리 3가지로 분류할 수 있다.

먼저 웹 서비스 디자인는 사용자의 편리성과 정확성을 최우선으로 한다. 이를 위해 지도, Youtube, 일반음식점-공공 Data 등의 API를 활용하여 사용자에게 필요한 정보를 제공한다.

다음으로 DB는 서버에 DB를 생성하여 사용하도록 한다. 영상 Data의 크기를 고려했을 때 raw data에 대한 전처리가 필수로 요구된다. 따라서 식당 정보 및 Youtube 영상에 대한 전처리 결과를 미리 DB 서버에 저장하여 최적화 된 서비스를 제공할 수 있게 한다.

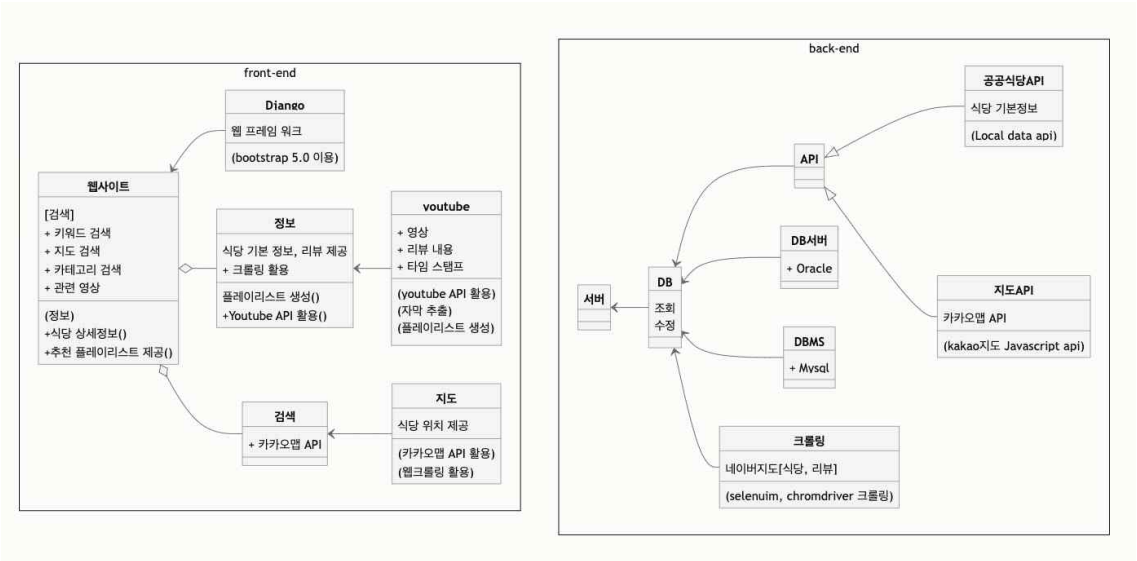
마지막으로 데이터 수집 및 처리는 앞에서 언급한 바와 같이 전처리를 진행하도록 한다. 단순히 Youtube에 영상을 검색했을 때 나오는 무작위의 영상을 사용자에게 제공하는 것이 아니라 검색된 영상에 일정한 기준을 적용하여 새로운 재생목록을 만들어 사용자에게 제공한다. 또한 자막, 영상 내 이미지 정보를 활용하여 사용자에게 신뢰할 수 있는 정보를 제공한다.

1-3. 수정사항

기존에는 모든 데이터를 Youtube 영상에서만 추출하려고 했다. 그러나 Youtube 영상 데이터 특성상 맛집에 대한 소비자의 리뷰를 중점적으로 제공하고 있으며 해당 맛집 자체에 대한 데이터는 적었다. 따라서 크롤링을 활용해 네이버, 카카오, 구글 플랫폼에 있는 해당 맛집에 대한 데이터를 수집하여 더 다양하고 신뢰할만한 정보를 제공하도록 한다.

2. 설계 상세화 및 변경 내역

2-1. 설계도



2-2. 상세설계

크게 Front-End와 Back-End를 기준으로 구분했다.

먼저 Front-End에서 Python을 기반으로 서비스를 구현했으며 프레임워크로는 Django를 활용했다. 웹 서비스에서는 맛집 검색 및 해당 식당에 대한 정보를 제공한다. 이를 위한 몇가지 기능이 존재한다.

1. 웹 서비스 디자인

웹 프레임워크인 Bootstrap 5.0을 활용해 웹 서비스를 구현했다.

2. 검색

키워드 검색, 지도 검색, 카테고리 검색이 있다. 키워드 및 카테고리는 사전 분류 작업을 통해 해당되는 키워드 및 카테고리에 맞는 맛집이 검색되도록 한다. 지도 검색은 Kakao Maps API를 활용하여 현재 및 특정 장소를 기반으로 한 인근 맛집이 검색되도록 한다.

3. 정보

식당에 대한 상세 정보를 제공한다. 크롤링을 통해 네이버, 카카오, 구글 등 다양한 플랫폼에 존재하는 맛집에 대한 세부 정보들을 수집 및 가공하여 사용자에게 제공한다. 또한 Kakao Maps API에서 제공하는 데이터 역시 활용한다.

4. Youtube

사용자에게 맛집을 추천하는 기준은 Youtube 영상이다. 또한 제공되는 상세 정보로 해당 맛집에 대한 Youtube 리뷰 영상이 있다. Youtube API를 활용하여 맛집 추천 및 리뷰 플레이리스트를 생성한다.

다음으로 Back-End에 대한 내용이다. 영상 Data를 활용하기 때문에 데이터를 미리 수집 및 전처리 과정을 통해 서버 DB에 저장 및 활용하는 방식으로 구성했다.

1. DB

DB는 Oracle에서 제공하는 가상 서버를 활용해 구축했다. DBMS로는 Mysql을 활용하여 DB 관리를 진행한다.

2. API for Data

식당에 관한 데이터를 얻기 위해 공공 Data API와 Kakao API를 활용했다. 식당 목록 및 카테고리는 공공 데이터를 활용하고, 해당 식당에 대한 추가적인 정보는 Kakao API를 활용했다.

3. 크롤링

API로 얻을 수 없는 추가적인 데이터를 수집하기 위해 크롤링을 진행했다. 네이버에서 수집한 데이터를 전처리하여 DB에 저장 및 사용한다.

3. 갱신된 과제 추진 계획

<table>
<tr><th colspan="5">8월</th></tr>
<tr><th>1주</th><th>2주</th><th>3주</th><th>4주</th><th>5주</th></tr>
<tr><td colspan="2">DB 서버구축</td><td colspan="3"></td></tr>
<tr><td colspan="3">중간보고서 작성</td><td colspan="2"></td></tr>
<tr><td colspan="2"></td><td colspan="2">웹 서비스 디자인 개선</td><td></td></tr>
<tr><td colspan="3"></td><td colspan="2">영상 재생목록 생성</td></tr>
<tr><td colspan="3"></td><td colspan="2">사이트와 자막 기능 결합</td></tr>
<tr><td colspan="3"></td><td colspan="2">식당 추천 알고리즘 구현</td></tr>
<tr><td colspan="3"></td><td colspan="2">서비스에 기능 결합</td></tr>
<tr><td colspan="5">raw data 수집 및 전처리</td></tr>
</table>

9월			
1주	2주	3주	4주
서비스 최적화 및 오류 수정			
DB 업데이트			
		최종 보고서 작성	

4. 구성원별 진척도

김민호	- 전반적인 서비스 운영 담당 - 웹 서비스 프론트 엔드 수행 (웹 서비스 디자인 & UI : Django & Bootstrap5 활용) - 웹 서비스 백 엔드 작업 수행 (웹 - DB - API 연결 작업)
하평안	- Data 수집 및 가공 담당 - 데이터 크롤링을 통한 Raw Data 수집 (Naver, Google, Kakao 플랫폼 활용) - API를 활용한 Raw Data 수집 (국가 공공 API를 활용한 지역별 식당 list 수집)
한정훈	- 서버 및 DB 관리 담당 - 데이터 베이스 및 서버 구축 (Oracle VM 인스턴스를 활용한 서버 구축) (Mysql을 활용한 DB 관리) - 기존의 서비스와 차별화 된 기능 개발

5. 보고 시점까지의 과제 수행 내용 및 중간 결과

웹 서비스의 디자인은 추후에 수정하도록 하고 Back-End 작업을 중점적으로 수행했다.

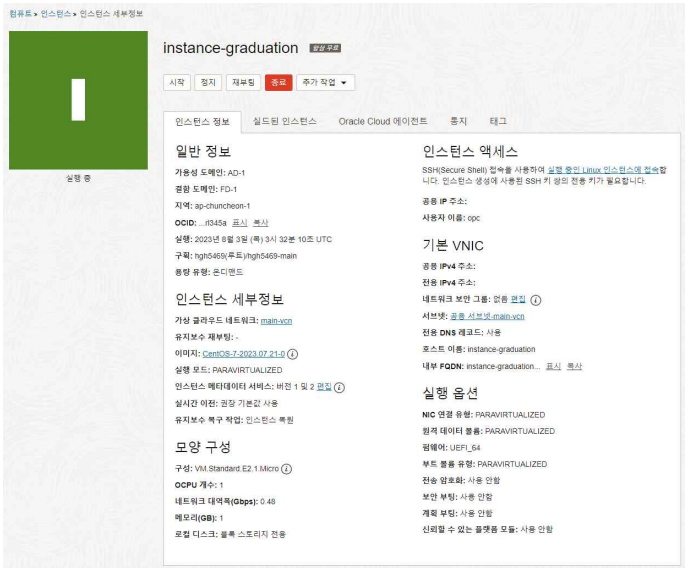


그림 4

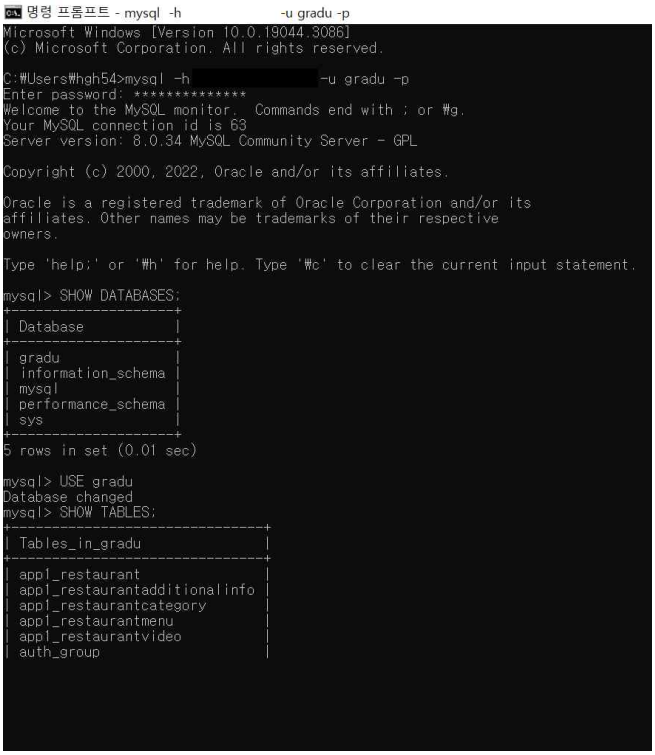


그림 5

먼저 DB 서버이다. 그림 4와 같이 오라클 VM 인스턴스를 활용해 서버를 만들고 해당 서버에서 Mysql을 설치하여 DB를 사용한다. 그림 5는 서버에 Mysql을 설치한 후 gradu라는

데이터베이스를 생성한 모습이다.

다음으로 식당에 대한 데이터는 Django Model로 관리된다. API에서 필요한 데이터를 추출 및 저장하며, 이 데이터를 기반으로 추가적인 작업이 진행된다.

```python
class Restaurant(models.Model):
    '''
    Restaurant 에 대한 기본 정보\n
    반영: 이름 / 카테고리 / 중부원점 TM 좌표 / 소재지 전체 주소 / 도로명 전체 주소
    '''
    idx          = models.AutoField(primary_key=True)
    # 식당 이름
    name         = models.CharField(max_length=100)
    # 식당 분류 카테고리
    idx_category = models.ForeignKey(
        RestaurantCategory,
        on_delete=models.RESTRICT,
    )
    # 식당의 TM 중부원점좌표
    x_coordinate = models.DecimalField(max_digits=15, decimal_places=9)
    y_coordinate = models.DecimalField(max_digits=15, decimal_places=9)
    # 식당의 소재지 전체 주소
    full_address = models.CharField(max_length=100, null=True, blank=True)
    # 식당의 도로명 전체 주소
    road_address = models.CharField(max_length=100, null=True, blank=True)

    def __str__(self):
        return f'{self.name} - {self.idx_category.name}’


class RestaurantCategory(models.Model):
    '''Restaurant 을 구분하는 범주'''
    idx         = models.AutoField(primary_key=True)
    name        = models.CharField(max_length=100)

    def __str__(self) -> str:
        return self.name


class RestaurantMenu(models.Model):
    idx          = models.AutoField(primary_key=True)
    restaurant   = models.ForeignKey(
        Restaurant,
        on_delete=models.CASCADE,
        related_name='restaurant_menus',
    )
    name         = models.CharField(max_length=100)
    price        = models.IntegerField()

    def __str__(self):
        return f'점포명: {self.restaurant.name} [{self.name}({self.price}원)]'
```

```python
class RestaurantVideo(models.Model):
    idx         = models.AutoField(primary_key=True)
    restaurant   = models.ForeignKey(
        Restaurant,
        on_delete=models.CASCADE,
        related_name='restaurant_videos'
    )
    name        = models.CharField(max_length=100, null=True, blank=True)
    address      = models.CharField(max_length=255)

    def save(self, *args, **kwargs):
        replaced_address = self.address
        if 'watch?v=' in replaced_address:
            replaced_address = replaced_address.split('watch?v=')[1]
            if '&' in replaced_address:
                replaced_address = replaced_address.split("&")[0]
        elif 'youtu.be' in replaced_address:
            replaced_address = replaced_address.split('youtu.be/')[1]
            if '?' in replaced_address:
                replaced_address = replaced_address.split("?")[0]
        self.address = replaced_address.replace("\n", "")
        super(RestaurantVideo, self).save(*args, **kwargs)

    def __str__(self):
        return f'점포명: {self.restaurant.name} [제목: {self.name}(주소:
{self.address})]'


class RestaurantAdditionalInfo(models.Model):
    idx         = models.AutoField(primary_key=True)
    restaurant   = models.OneToOneField(
        Restaurant,
        on_delete=models.CASCADE,
        related_name='restaurant_info'
    )
```

생성된 모델에 관한 코드로 아래 그림 6과 같은 구조로 이루어져있다.

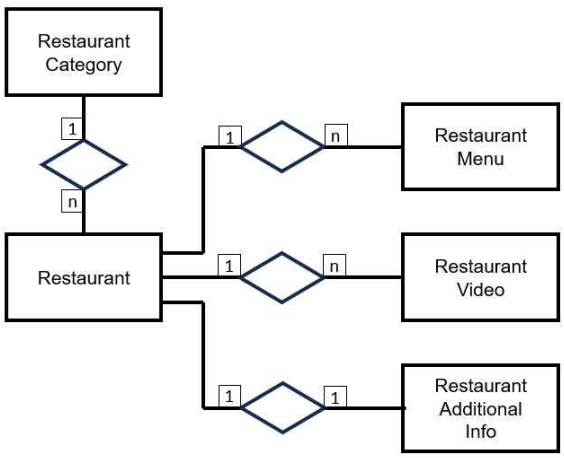


그림 6

마지막으로 웹 서비스의 동작 모습이다. 웹 서비스 디자인은 이후에 전면 수정될 예정이다.

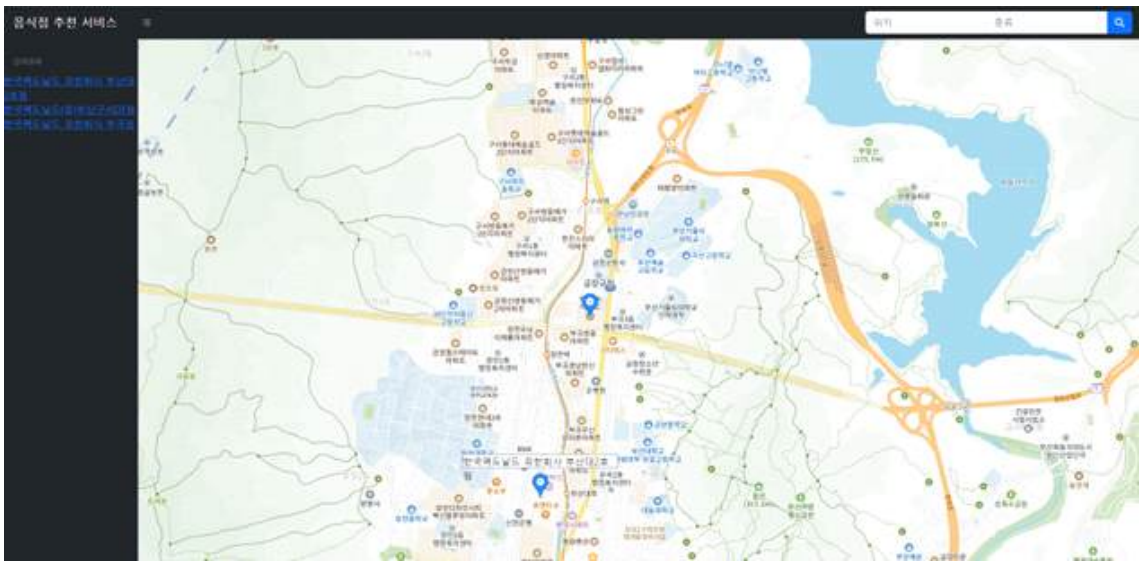


그림 7

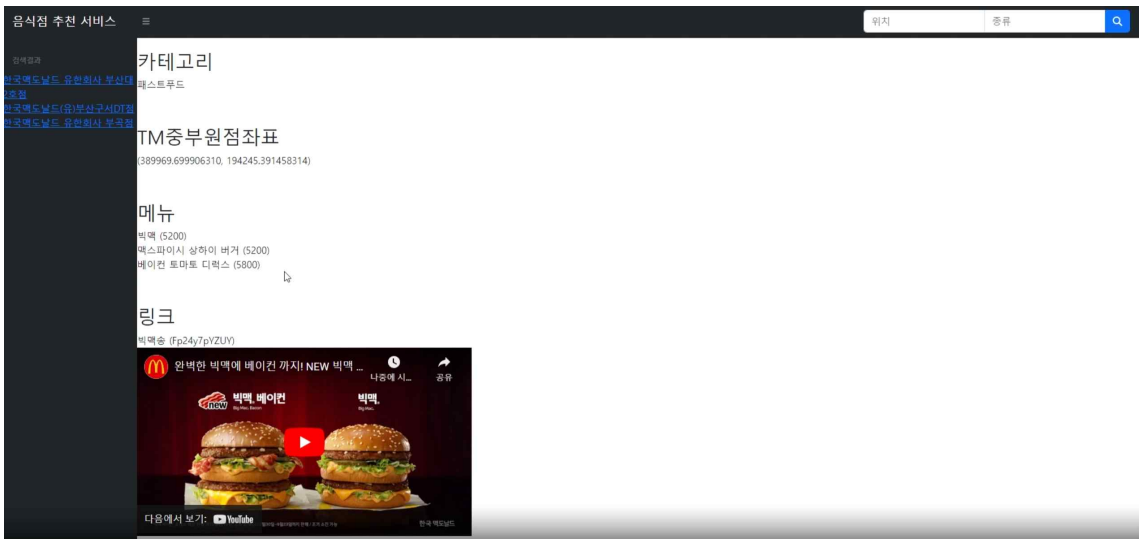


그림 8

먼저 그림 7과 같이 현재 위치를 중심으로 한 지도가 제공된다. 해당 지역의 맛집들을 좌측 사이드 바에 목록으로 제공한다. 또한 우측 검색바를 활용해 추가적인 검색이 가능한데, 이후 카테고리, 키워드 검색이 가능하도록 추가할 예정이다.

또한 현재 테스트를 위해 "부산시 금정구" 내 일부 매장만을 데이터로 사용하고 있다. 공공데이터 API에서 전국 일반음식점 목록을 제공하기에 테스트 이후 즉시 데이터를 추가할 수 있다.

그림 8은 검색한 식당에 대한 세부 정보를 제공하는 화면이다. 식당의 위치, 카테고리, 메뉴(이미지 추가 예정), Youtube 리뷰 영상을 제공한다. 세부 정보 화면 디자인 역시 개선될 예정이며 크롤링을 통해 얻은 데이터 역시 추가될 예정이다.