<2024 국민대학교 겨울방학 PBL 부트캠프>

Team 3 - 최종 프로젝트 발표

: AD Manager Platform Server

Team 3

20213062 이찬우

20171634 성예린

20223104 양준영

20233110 최승환

목차

1. 개발 준비

- 구축해야 할 시스템: **AD Manager Platform Server**

- 역할 분배

Back-end

이찬우 – 서버 관리 및 views.py API 작성, Readme.md 및 API 문서 작성 담당
양준영 – 모델 및 MySQL DB 관리, 최종 ERD 및 다이어그램 작성 담당

Front-end

성예린 – main.html 구축 담당
최승환 – adManager.html 구축 담당

2. 시스템 구조도

- **Server**
  - Django
  - DRF (Django REST Framework)
- **Templates**
  - HTML
  - CSS
  - JavaScript
- **DB**
  - MySQL
- **Tools**
  - git / GitHub, MySQL Workbench

2. 시스템 구조도

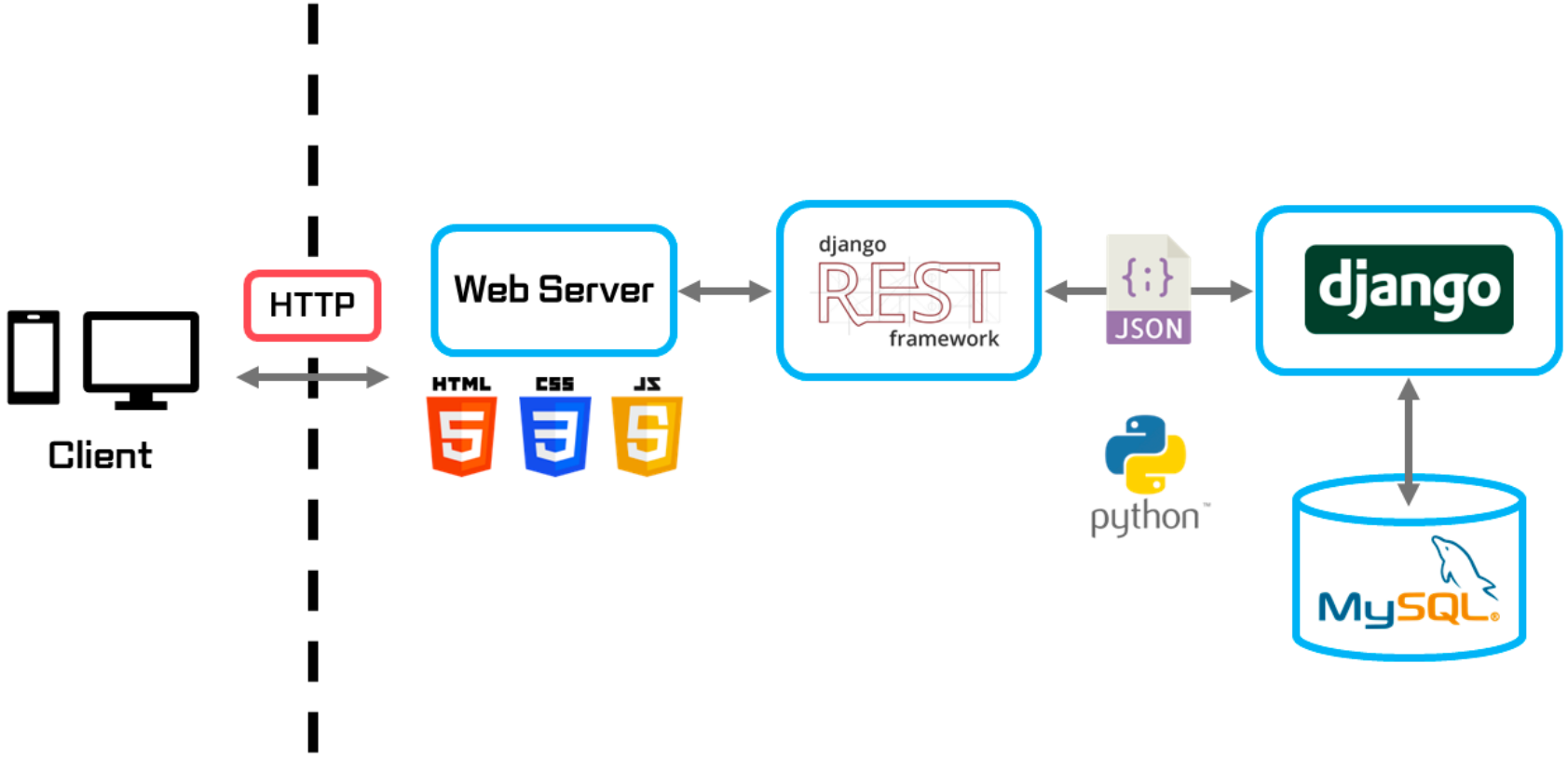
Django 프레임워크에서 의도한 대로, **MTV** (**Model** – **Template** – **View**) 패턴을 적용함

- **Model**
  데이터베이스와 상호작용하는 부분
  MVC 패턴의 Model에 해당

- **Template**
  사용자에게 보여지는 부분
  MVC 패턴의 View

- **View**
  Model과 Template 사이의 중개자 역할, API
  MVC 패턴의 Controller

2. 시스템 구조도


Client
HTTP
Web Server
HTML
CSS
JS
django
REST
framework
{;}
JSON
python
django
MySQL

3. 시연 및 유저 인터페이스

main.html: 광고들을 받아와 보여주는 기능 페이지

메뉴 및 네비게이션 바
- Home, Git, Notion, Trello, Admin 각각을 누를 시 메인 화면, 깃허브, 노션, 트렐로, 관리자 페이지로 이동
- DB 에 사용자명과 비밀번호를 등록하고 관리자 페이지에 들어갈 수 있음

```
<header class="menu">
  <nav>
    <ul>
      <li><a href="/main/">Home</a></li>
      <li><a href="https://github.com/chanwoo7/PBL-Team3-Project.git">Git</a></li>
      <li><a href="https://www.notion.so/codestates/3-310f4ebadc5b42cd89e2db1e144317db?pvs=4">Notion</a></li>
      <li><a href="https://trello.com/b/HxSMGz9c/kmu-ad-system-pbl-3">Trello</a></li>
      <li><a href="/admin/">Admin</a></li>
    </ul>
  </nav>
</header>
```

3. 시연 및 유저 인터페이스

main.html: 광고들을 받아와 보여주는 기능 페이지

광고 배너
- 이미지 위와 양 옆, 게시판 아래에 들어가는 광고

콘텐츠 섹션
- 현재는 사진 하나가 있지만 동영상이나 텍스트 등 다른 것으로 대체될 수 있음

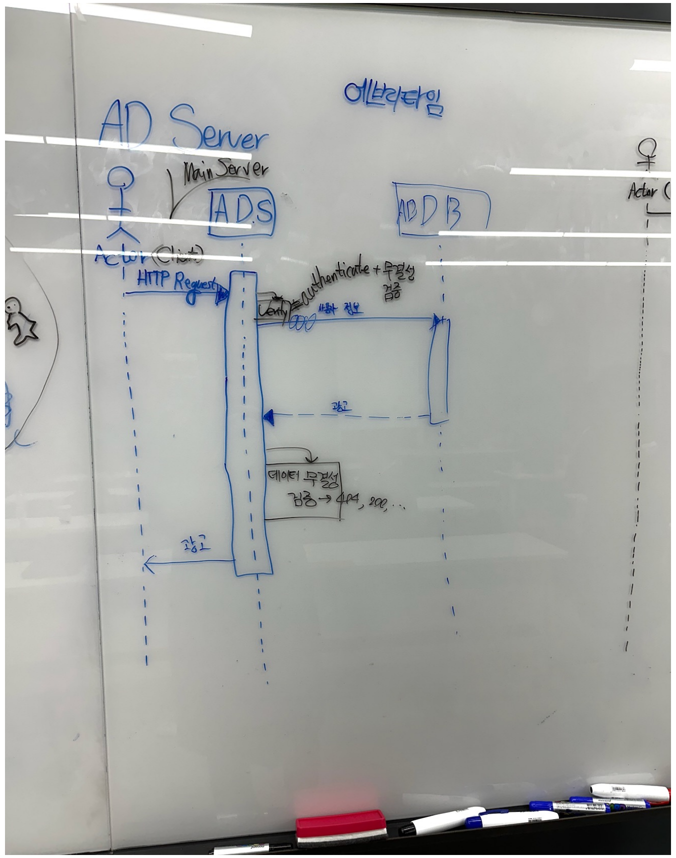
```
<aside class="small-ad-banner">
    <!-- 작은 광고 배너 내용 -->
    <p>광고</p>
</aside>
<main class="container">
    <aside class="ad-box">
        <!-- 왼쪽 광고 콘텐츠 -->
        <p>광고</p>
    </aside>
    <section class="image-box">
        <img src="/static/image/comDev.avif" alt="Computer Development">
    </section>
    <aside class="ad-box">
        <!-- 오른쪽 광고 콘텐츠 -->
        <p>광고</p>
    </aside>
</main>
```

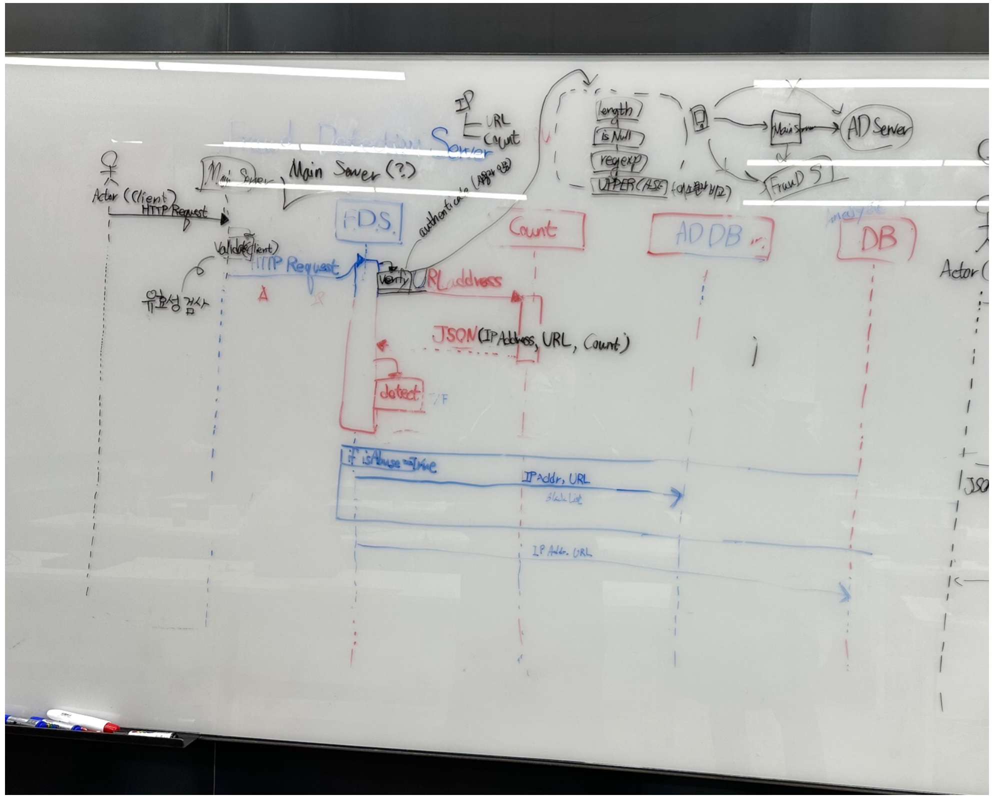
3. 유저 인터페이스

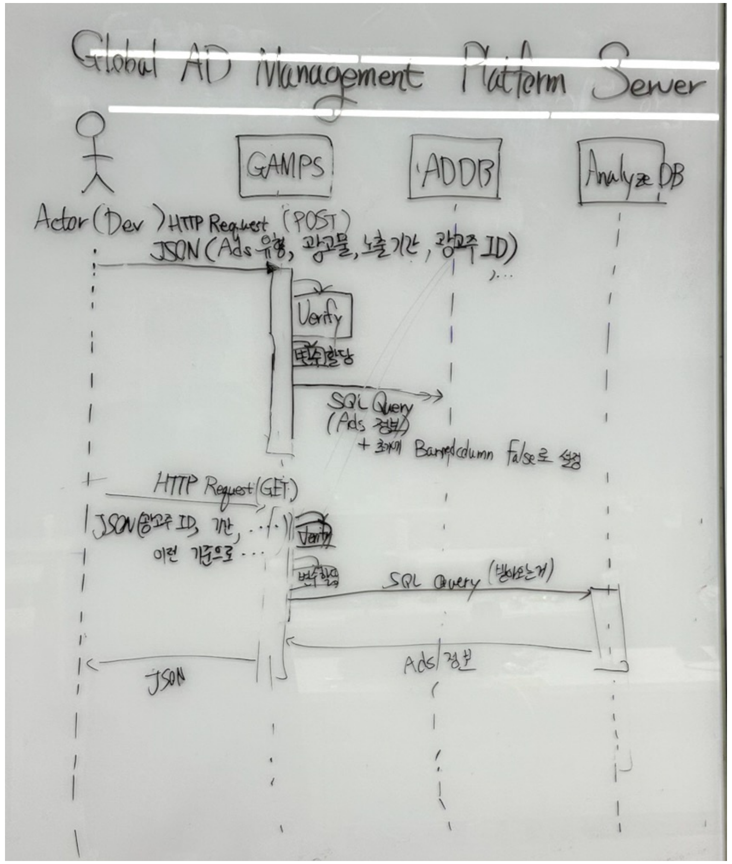
adManager.html: AD Manager Platform

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset = "UTF-8">
    <meta name = "viewport" content = "width=device-width, initial-scale=1.0">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet">
    <link href="https://getbootstrap.com/docs/5.3/assets/css/docs.css" rel="stylesheet">

    <title>AD Manager Platform</title>
    {% load static %}

    <style>
        .container {
            text-align: center;
        }
        .outer-container {
            border: 2px solid #ccc;
            border-radius: 10px;
            padding: 20px;
            background-color: #f9f9f9;
            min-height: 565px;
            width: auto;
            display: flex;
            flex-direction: column;
            align-items: center;
        }
        .detail-modal {
            border: 2px solid rgb(255, 255, 255);
            border-radius: 10px;
            margin-bottom: 20px;
            padding: 15px;
            background-color: deepskyblue;
            color: white;
            width: 90%;
            text-align: center;
            overflow: hidden;
        }
```

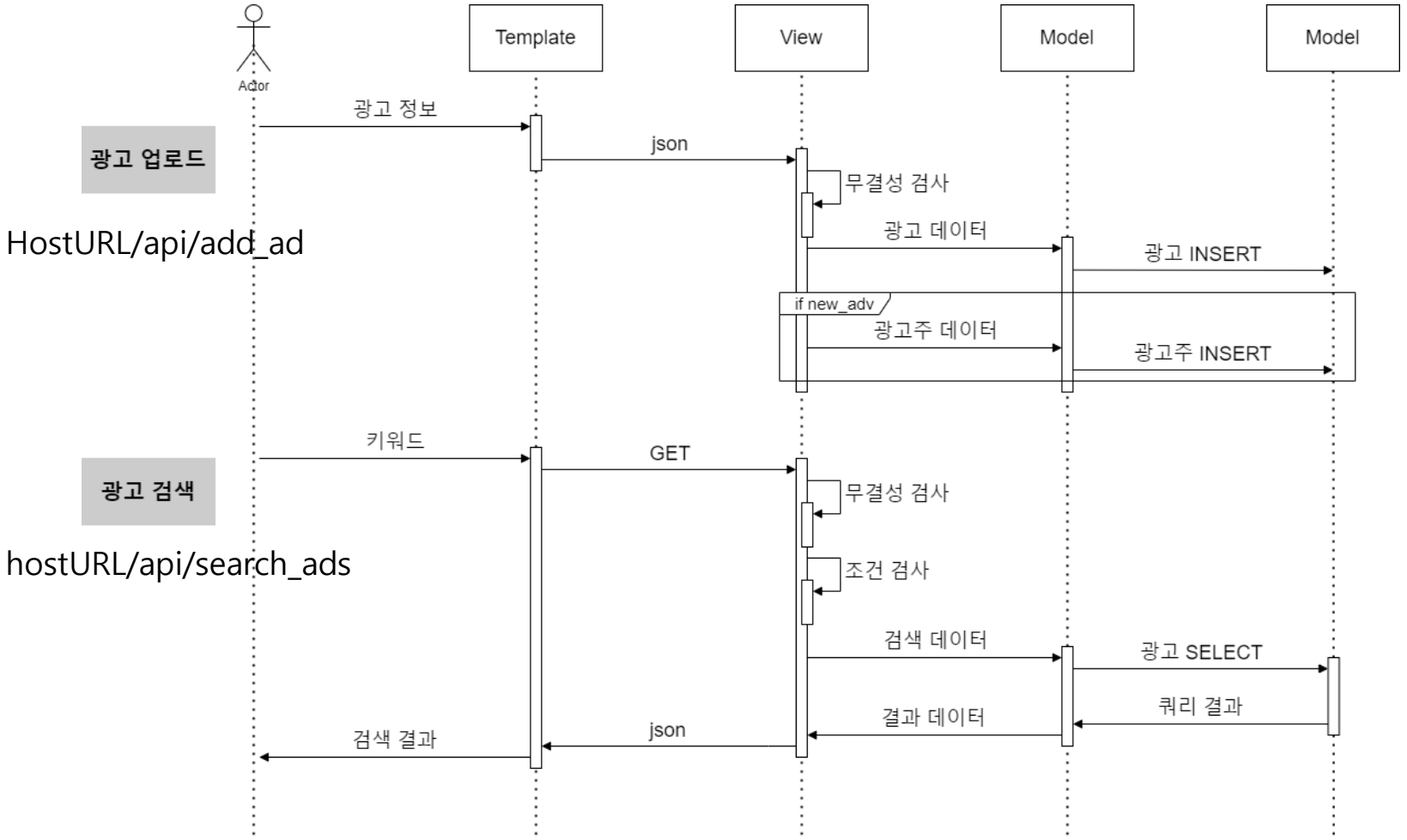
4. 시퀀스 다이어그램


AD Server
Main Server
ADS
AD DB
HTTP Request
Verify
광고

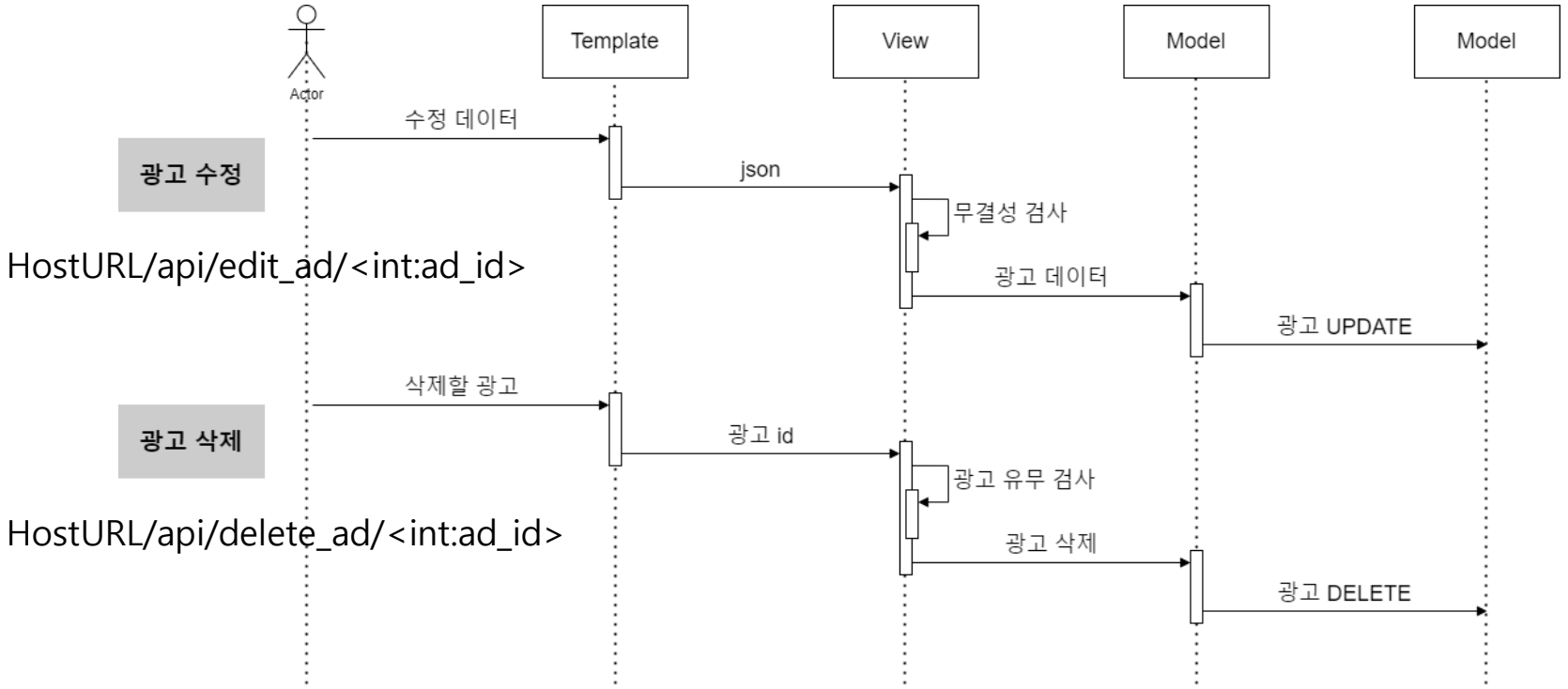

Main Server (?)
Actor (Client)
HTTP Request
FDS
Count
AD DB
DB
Verify
URL address
JSON (IP Address, URL, Count)
detect
유효성 검사
length
is Null
regexp
AD Server


Global AD Management Platform Server
GAMPS
ADDB
Analyze DB
Actor (Dev)
HTTP Request (POST)
JSON (Ads 유형, 광고물, 노출기간, 광고주 ID)
Verify
SQL Query
HTTP Request (GET)
SQL Query
JSON
Ads 정보

4. 시퀀스 다이어그램


Actor
Template
View
Model
Model
광고 업로드
HostURL/api/add_ad
광고 정보
json
무결성 검사
광고 데이터
광고 INSERT
if new_adv
광고주 데이터
광고주 INSERT
광고 검색
hostURL/api/search_ads
키워드
GET
무결성 검사
조건 검사
검색 데이터
광고 SELECT
쿼리 결과
결과 데이터
json
검색 결과

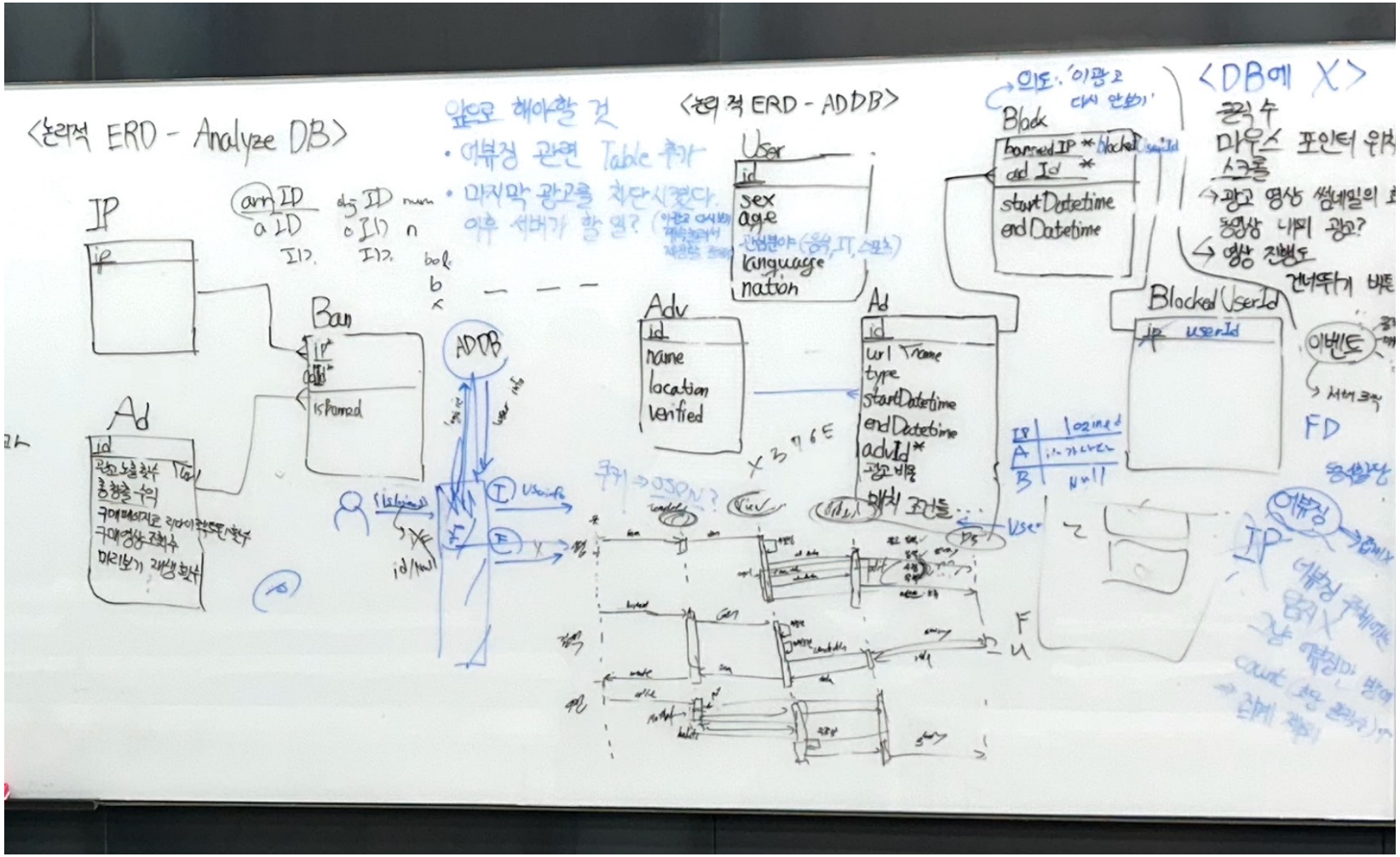
4. 시퀀스 다이어그램


Actor
Template
View
Model
Model
수정 데이터
광고 수정
json
무결성 검사
HostURL/api/edit_ad/<int:ad_id>
광고 데이터
광고 UPDATE
삭제할 광고
광고 삭제
광고 id
광고 유무 검사
HostURL/api/delete_ad/<int:ad_id>
광고 삭제
광고 DELETE

5. 데이터 구조화 및 ERD

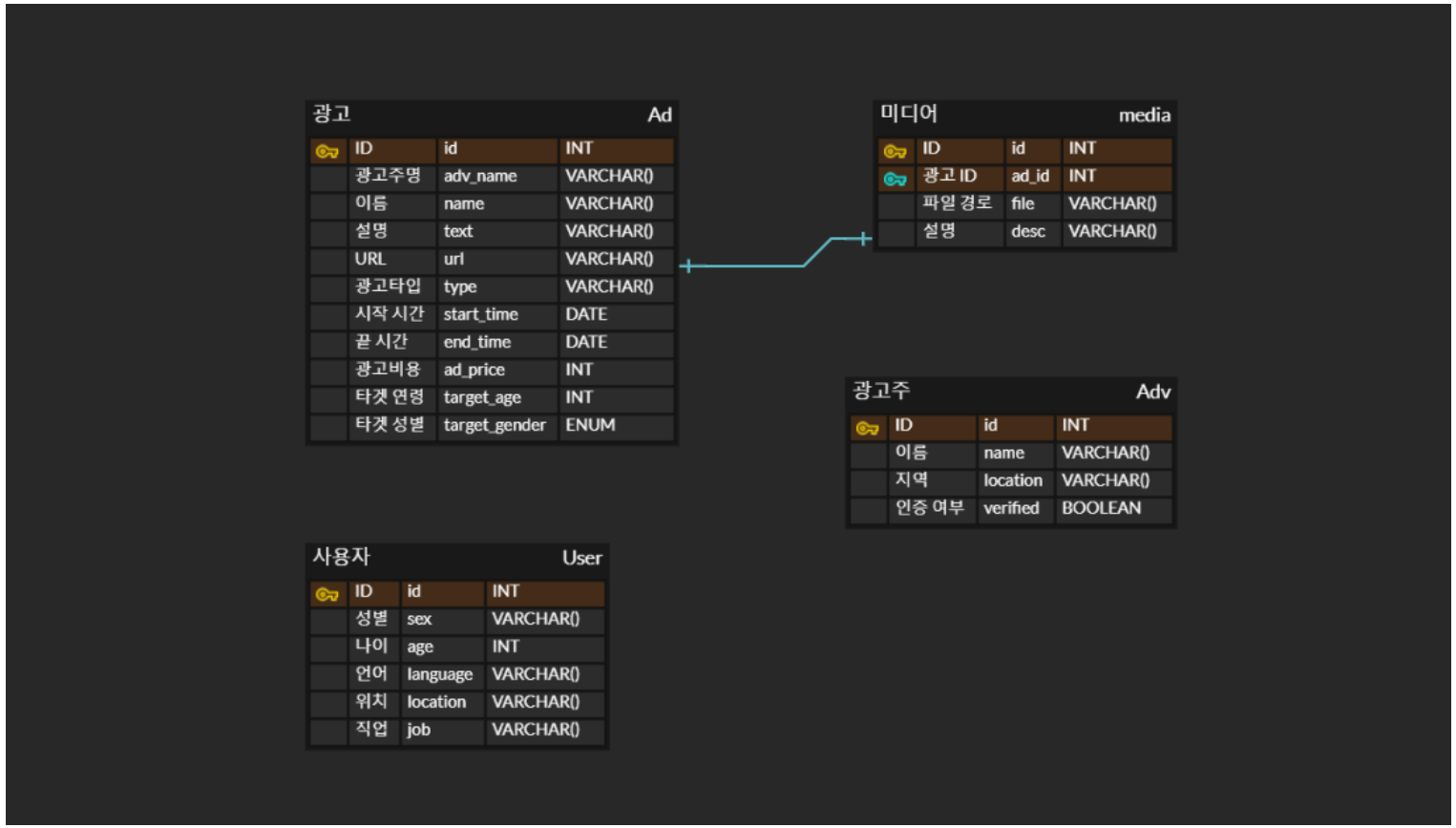
초기 Table

- Ad (광고)
- Adv (광고주)
- media (미디어)
- User (유저)
- 어뷰징 및 광고 차단 관련 테이블…



5. 데이터 구조화 및 ERD

최종 ERD



5. 데이터 구조화 및 ERD

models.py

```python
# 광고
9 usages  jun
class Ad(models.Model):
    # 열거형 첫 번째는 db 저장 값. 쿼리 및 결과 출력에 사용
    # 열거형 두 번째는 폼에 보이는 값

    # TYPE_CHOICES = [
    #     ('image', '사진'),
    #     ('video', '동영상'),
    #     ('banner', '배너'),
    # ]
    TARGET_GENDER_CHOICES = [
        ('남', '남성'),
        ('여', '여성'),
        ('없음', '없음'),
    ]

    id = models.AutoField(primary_key=True)
    adv_name = models.CharField(max_length=10)
    name = models.CharField(max_length=50)
    text = models.CharField(max_length=100)
    url = models.CharField(max_length=500)
    type = models.CharField(max_length=10)
    start_time = models.DateField()
    end_time = models.DateField()
    ad_price = models.IntegerField()
    target_age = models.IntegerField(default=0)
    target_gender = models.CharField(max_length=2, choices=TARGET_GENDER_CHOICES, default='없음')
```

```python
# 미디어
6 usages  jun
class Media(models.Model):
    id = models.AutoField(primary_key=True)
    ad_id = models.ForeignKey(Ad, on_delete=models.CASCADE)
    file = models.FileField(upload_to='ad')
    desc = models.CharField(max_length=20)

    1 usage (1 dynamic)  jun
    def delete(self, *args, **kwargs):
        super(Media, self).delete(*args, **kwargs)
        os.remove(os.path.join(settings.MEDIA_ROOT, self.file.path))


# 광고주
5 usages  jun
class Adv(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=20)
    location = models.CharField(blank=True, max_length=50)
    verified = models.BooleanField(default=False)
```

6. API

API 정의 문서 작성

API name	간단한 설명
add_ad	광고를 생성한다.
edit_ad	광고를 수정한다.
delete_ad	광고를 삭제한다.
search_ads	광고를 (특정 조건에 따라) 검색한다.
get_all_ads	모든 광고를 불러온다.
get_ad_details	광고 상세 정보(JSON)을 불러온다.
upload_media	미디어를 업로드한다.
show_media	미디어를 재생한다.
get_media_details	미디어 상세 정보(JSON)을 불러온다.

6. API

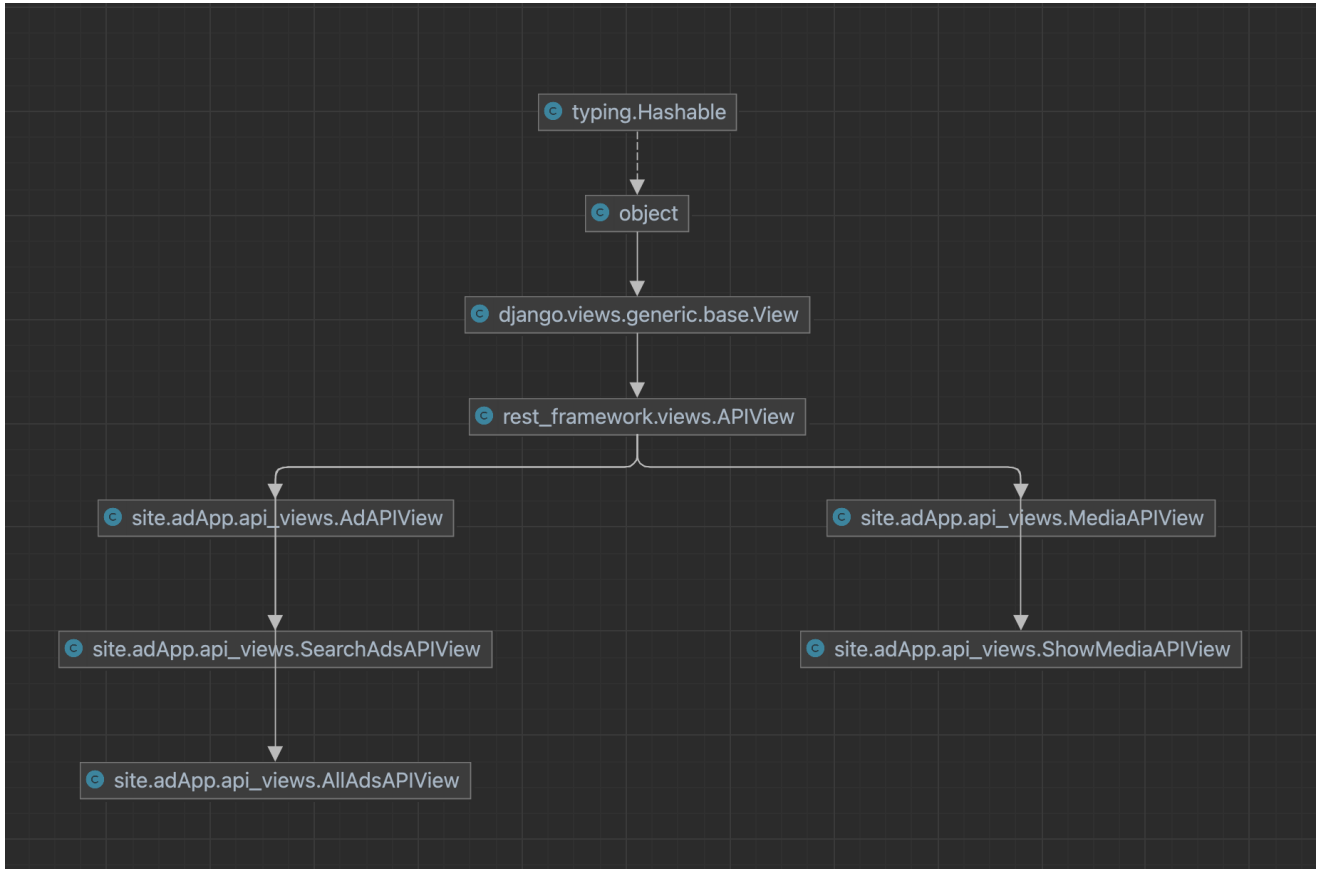
JSON 형식의 Request / Response를 위해 **Serializer** 작성

```python
from rest_framework.serializers import ModelSerializer
from .models import *


 chanwoo7
class AdvSerializer(ModelSerializer):
    chanwoo7
    class Meta:
        model = Adv
        fields = '__all__'


7 usages  chanwoo7
class AdSerializer(ModelSerializer):
    chanwoo7
    class Meta:
        model = Ad
        fields = '__all__'


3 usages  chanwoo7
class MediaSerializer(ModelSerializer):
    chanwoo7
    class Meta:
        model = Media
        fields = '__all__'

```

6. API

Class-Based-View (CBV) 기반의 Views 작성
DRF에서 제공하는 **APIView** 클래스를 상속

- **AdAPIView**
  get_ad_details, add_ad, edit_ad, delete_ad
- **SearchAdsAPIView**
  search_ads
- **AllAdsAPIView**
  get_all_ads
- **MediaAPIView**
  get_media_details, upload_media
- **ShowMediaAPIView**
  show_media



6. API

```python
4 usages  chanwoo7 +1
class AdAPIView(APIView):
    # get_ad_details
    2 usages (2 dynamic)  chanwoo7
    def get(self, request, ad_id):
        ad = get_object_or_404(Ad, pk=ad_id)
        serializer = AdSerializer(ad)
        return Response(serializer.data)

    # add_ad
    chanwoo7 +1
    def post(self, request):
        serializer = AdSerializer(data=request.data)
        if serializer.is_valid():
            # new adv 추가
            adv_name = serializer.validated_data['adv_name']
            if not Adv.objects.filter(name=adv_name).exists():
                new_adv = Adv(name=adv_name)
                new_adv.save()
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

    # edit_ad
    chanwoo7
    def put(self, request, ad_id):
        ad = get_object_or_404(Ad, pk=ad_id)
        serializer = AdSerializer(ad, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

    # delete_ad
    1 usage (1 dynamic)  chanwoo7
    def delete(self, request, ad_id):
        ad = get_object_or_404(Ad, pk=ad_id)
        ad.delete()
        return Response({'message': 'Ad successfully deleted.'}, status=status.HTTP_204_NO_CONTENT)
```

```python
1 usage  chanwoo7
class SearchAdsAPIView(APIView):
    # search_ads
    2 usages (2 dynamic)  chanwoo7
    def get(self, request):
        search_type = request.GET.get('search_type', '')
        search_query = request.GET.get('search_query', '')

        # 유효한 search_types 정의
        valid_search_types = ['id', 'adv_name', 'name', 'type', 'text']

        if search_type not in valid_search_types:
            return Response({'error': 'Invalid search_type'}, status=status.HTTP_400_BAD_REQUEST)

        ads = Ad.objects.all()

        if search_query == '':
            serializer = AdSerializer(ads, many=True)
            return Response(serializer.data)

        # search_type에 따라 필터링
        if search_type == 'id':
            ads = ads.filter(id=search_query)
        elif search_type == 'adv_name':
            ads = ads.filter(adv_name__icontains=search_query)
        elif search_type == 'name':
            ads = ads.filter(name__icontains=search_query)
        elif search_type == 'type':
            ads = ads.filter(type__icontains=search_query)
        elif search_type == 'text':
            ads = ads.filter(text__icontains=search_query)

        serializer = AdSerializer(ads, many=True)
        return Response(serializer.data)
```

6. API

```python
1 usage  chanwoo7
class AllAdsAPIView(APIView):
    # get_all_ads
    2 usages (2 dynamic)  chanwoo7
    def get(self, request):
        ads = Ad.objects.all()
        serializer = AdSerializer(ads, many=True)
        return Response(serializer.data)


2 usages  chanwoo7
class MediaAPIView(APIView):
    # get_media_details
    2 usages (2 dynamic)  chanwoo7
    def get(self, request, media_id):
        media = get_object_or_404(Media, pk=media_id)
        serializer = MediaSerializer(media)
        return Response(serializer.data)

    # upload_media
    chanwoo7
    def post(self, request):
        serializer = MediaSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

```

```python
1 usage  chanwoo7
class ShowMediaAPIView(APIView):
    # show_media
    2 usages (2 dynamic)  chanwoo7
    def get(self, request, media_id):
        media = get_object_or_404(Media, pk=media_id)
        file_path = media.file.path
        return FileResponse(open(file_path, 'rb'))

```

Thank you