빅데이터 시각화

정택부

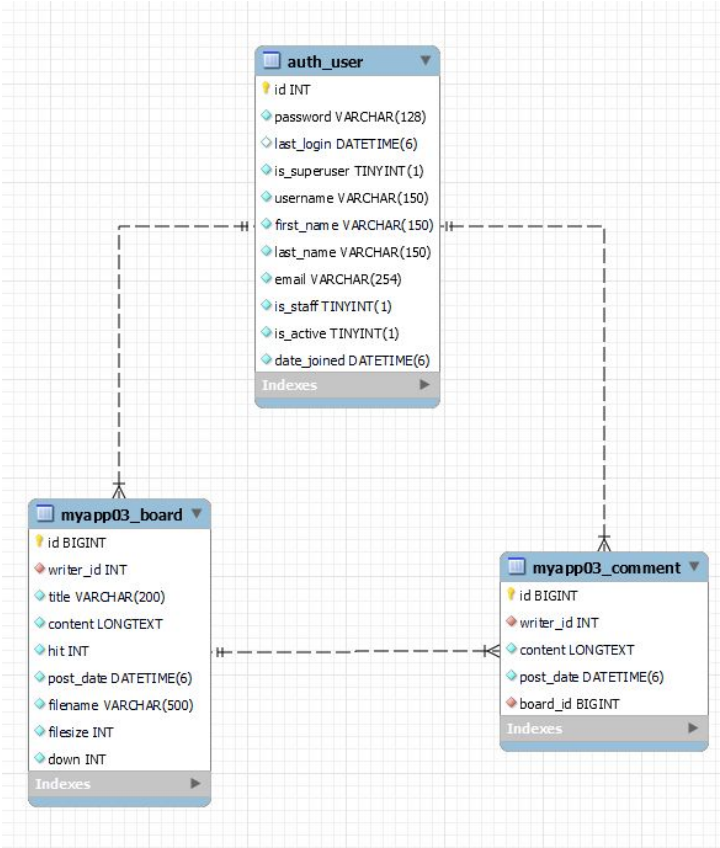
urls.py path설정

```python
from django.contrib.auth import views as auth_views
from myapp03 import views

urlpatterns = [
    path("admin/", admin.site.urls),
    path("", views.base),
    path("write_form/", views.write_form),
    path("insert/", views.insert),
    path("list/", views.list),
    path('download_count/', views.download_count),
    path('download/', views.download),
    path("list_page/", views.list_page),
    path("detail/<int:board_id>", views.detail),
    path("update_form/<int:board_id>", views.update_form),
    path("update/", views.update),
    path('delete/<int:board_id>', views.delete),
    path('comment_insert/', views.comment_insert),

    ####################로그인 +
    회원가입####################################

    path("signup/", views.signup),
    path("login/", auth_views.LoginView.as_view(template_name='common/login.
    html'), name='login'),
    path("logout/", auth_views.LogoutView.as_view(), name='logout'),

    ##########################
    path("melon/", views.melon),
    path("weather/", views.weather),
    path("map/", views.map),
    path("wordcloud/", views.wordcloud),
    path("webtoon/", views.webtoon),
]
```

게시판 구현

models.py

```python
class Board(models.Model):
    # writer = models.CharField(null=False, max_length=50)
    writer = models.ForeignKey(User, on_delete=models.CASCADE)
    title = models.CharField(null=False, max_length=200)
    content = models.TextField(null=False)
    hit = models.IntegerField(default=0)
    post_date = models.DateTimeField(default=datetime.now, blank=True)
    filename = models.CharField(null=False, max_length=500,
                                blank=True, default='')
    filesize = models.IntegerField(default=0)
    down = models.IntegerField(default=0)

    def hit_up(self):
        self.hit += 1

    def down_up(self):
        self.down += 1


class Comment(models.Model):
    # CASCADE : 외래키를 받아오는 테이블이 삭제될 때 외래키로 받는 테이블도 삭제
    board = models.ForeignKey(Board, on_delete=models.CASCADE)
    # writer = models.CharField(null=False, max_length=50)
    writer = models.ForeignKey(User, on_delete=models.CASCADE)
    content = models.TextField(null=False)
    post_date = models.DateTimeField(default=datetime.now, blank=True)
```



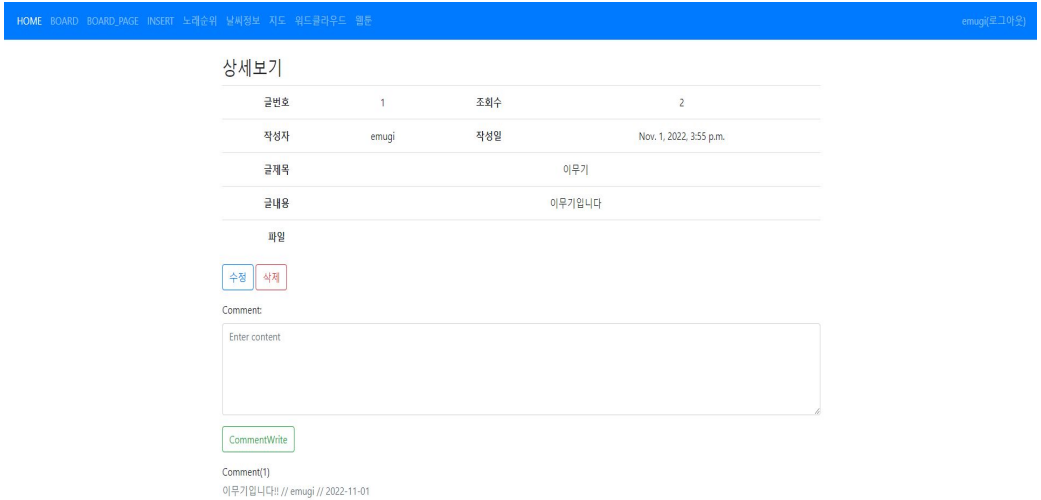
게시판 구현(댓글)

views.py

```python
# comment_insert
@ csrf_exempt
@login_required(login_url='/login/')
def comment_insert(request):
    id = request.POST['id']
    board = get_object_or_404(Board, pk=id)
    dto = Comment(writer=request.user,
                  content=request.POST['content'], board=board)
    dto.save()
    return redirect('/detail/'+id)
```

views.py

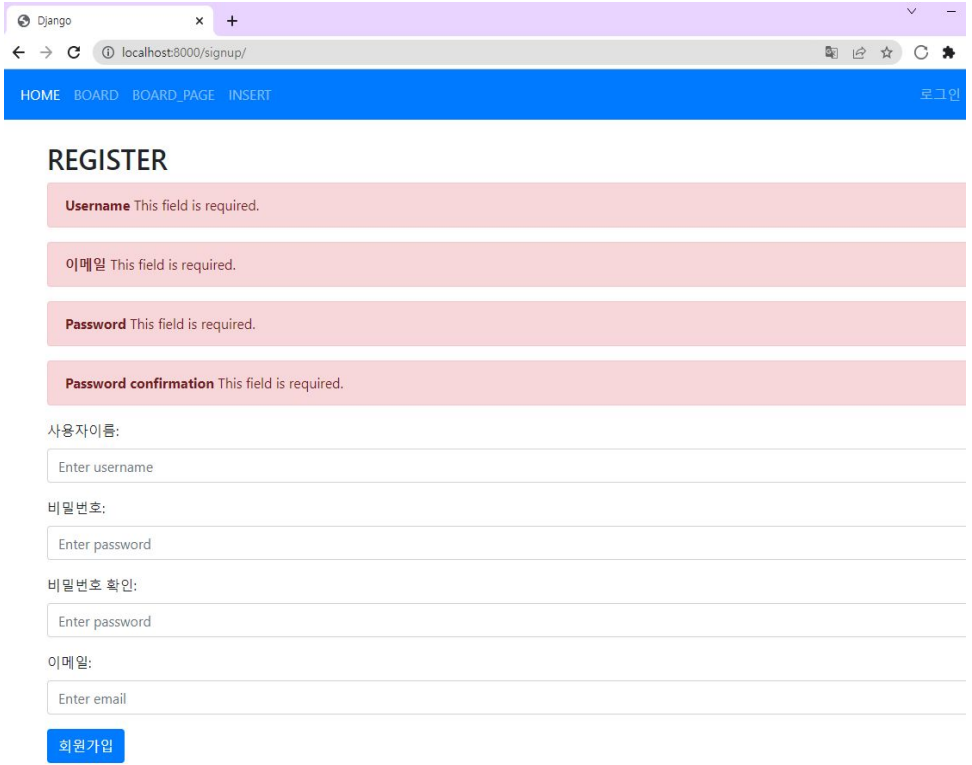
```python
# detail
def detail(request, board_id):
    dto = Board.objects.get(id=board_id)
    dto.hit_up()
    dto.save()
    return render(request, 'board/detail.html', {'dto': dto})
```



게시판 구현(회원가입)

signup.html

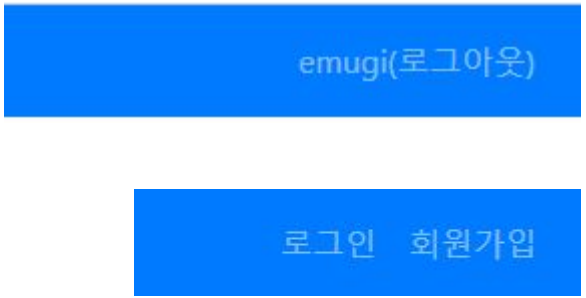
```
{% extends 'base.html' %}
{% block content %}
<div class="container">
    <h2>REGISTER</h2>
    <form method="post">
        {% csrf_token %}
        <!-- include : 포함시킬때 -->
        {% include "form_errors.html" %}
        <div class="form-group">
            <label for="username">사용자이름:</label>
            <input type="text" class="form-control" id="username" placeholder="Enter username"
            name="username"
                value="{{forms.username.value|default_if_none:''}}">
        </div>
        <div class="form-group">
            <label for="password1">비밀번호:</label>
            <input type="password" class="form-control" id="password1" placeholder="Enter
            password" name="password1"
                value="{{forms.password1.value|default_if_none:''}}">
        </div>
        <div class="form-group">
            <label for="password2">비밀번호 확인:</label>
            <input type="password" class="form-control" id="password2" placeholder="Enter
            password" name="password2"
                value="{{forms.password2.value|default_if_none:''}}">
        </div>
        <div class="form-group">
            <label for="email">이메일:</label>
            <input type="text" class="form-control" id="email" placeholder="Enter email"
            name="email"
                value="{{forms.email.value|default_if_none:''}}">
        </div>

        <button type="submit" class="btn btn-primary">회원가입</button>
    </form>
</div>
{% endblock %}
```



게시판 구현(로그인 , 로그아웃)

base.html

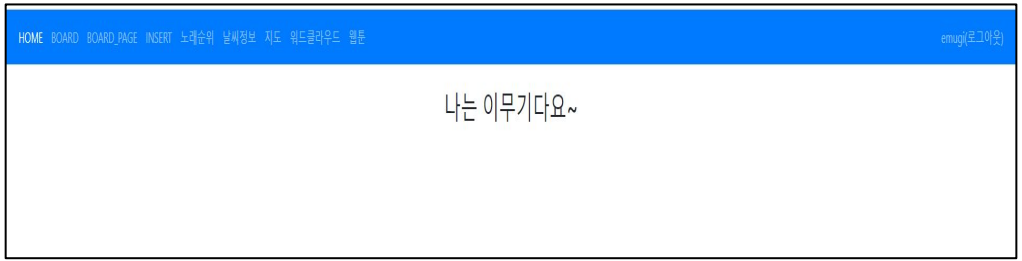
```
<ul class="navbar-nav">
    {% if user.is_authenticated%}
    <!-- 인증객체가 있다면 로그아웃 -->
    <li class="nav-item">
        <a class="nav-link" href="/logout">{{user.username}}(로그아웃)</a>
    </li>
    <!-- 인증객체가 없다면 로그인 + 회원가입 -->
    {% else %}
    <li class="nav-item">
        <a class="nav-link" href="/login">로그인</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="/signup">회원가입</a>
    </li>
    {% endif %}
</ul>
```



게시판 구현(로그인, 로그아웃 클릭 시 경로)

settings.py

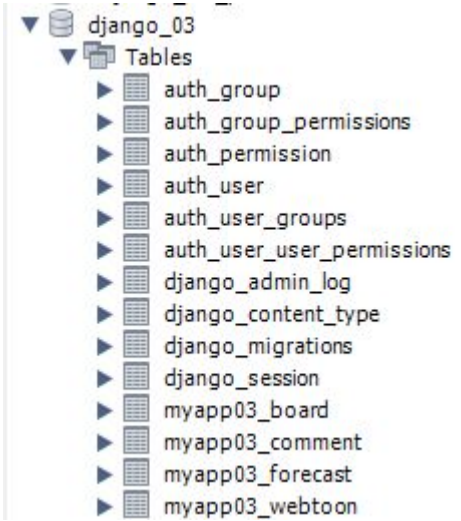
```
# 로그인 성공 후 이동하는 URL
LOGIN_REDIRECT_URL = '/'
LOGOUT_REDIRECT_URL = '/login'
```



날씨정보 시각화

models.py

```python
class Comment(models.Model):
    # CASCADE : 외래키를 받아오는 테이블이 삭제될 때 외래키로 받는 테이블도 삭제
    board = models.ForeignKey(Board, on_delete=models.CASCADE)
    # writer = models.CharField(null=False, max_length=50)
    writer = models.ForeignKey(User, on_delete=models.CASCADE)
    content = models.TextField(null=False)
    post_date = models.DateTimeField(default=datetime.now, blank=True)
```



날씨정보 시각화

views.py

```python
# weather
def weather(request):
    last_date = Forecast.objects.values('tmef').order_by('-tmef')[:1]
    print('last_date : ', len(last_date))
    weather = {}
    bigdataProcess.weather_crawing(last_date, weather)
    print("last_date query : ", str(last_date.query))
    for i in weather:
        for j in weather[i]:
            dto = Forecast(city=i, tmef=j[0], wf=j[1], tmn=j[2], tmx=j[3])
            dto.save()

    # 검색
    word = request.GET.get('word', '')

    result = Forecast.objects.filter(Q(city__contains=word))
    result_pd = Forecast.objects.filter(Q(city__contains=word)).values(
        'wf').annotate(dcount=Count('wf')).values("dcount", "wf")
    print("result_pd query : ", str(result_pd.query))

    df = pd.DataFrame(result_pd)
    image_dic = bigdataProcess.weather_make_chart(result, df.wf, df.dcount)

    return render(request, 'bigdata/weather_chart.html', {"img_data":
    image_dic, "result": result})
```

날씨정보 시각화

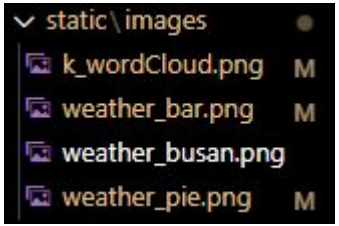
bigdataProcess.py

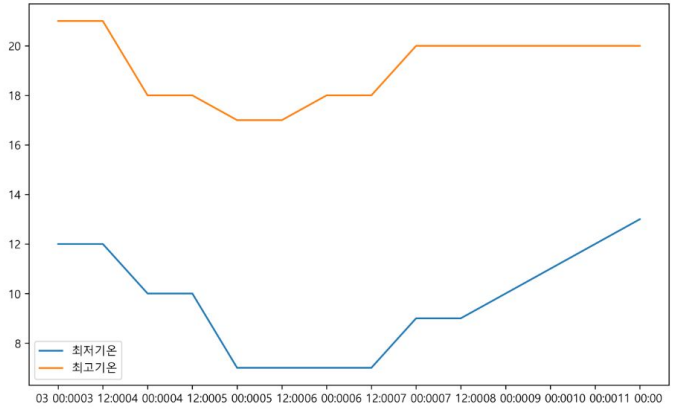
```python
# 날씨
def weather_crawing(last_date, weather):

    url = 'https://www.weather.go.kr/weather/forecast/mid-term-rss3.jsp?
    stnId=108'
    response = requests.get(url)
    html = response.text
    # xml형식의 사이트를 가져오므로 'lxml'를 사용
    soup = BeautifulSoup(html, 'lxml')

    for i in soup.find_all('location'):
        city = i.find('city').string
        weather[city] = []
        for j in i.find_all('data'):
            temp = []
            if(len(last_date) == 0) or (last_date[0]['tmef'] < j.find
            ('tmef').text):
                temp.append(j.find('tmef').text)
                temp.append(j.find('wf').text)
                temp.append(j.find('tmn').text)
                temp.append(j.find('tmx').text)
                # print('temp :', temp)
                weather[i.find('city').string].append(temp)
```

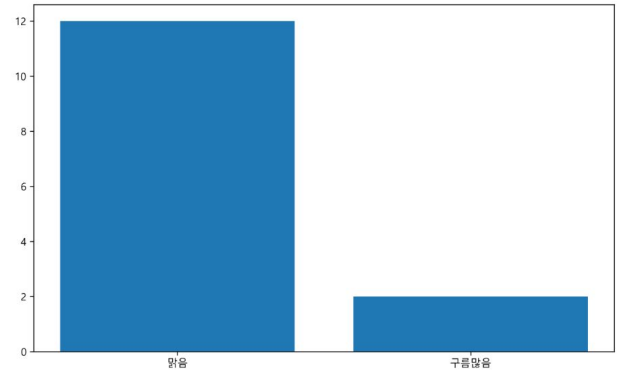
날씨정보 시각화

bigdataProcess.py

```python
def weather_make_chart(result, wfs, dcounts):

    # 한글
    font_path = 'c:/Windows/fonts/malgun.ttf'
    font_name = font_manager.FontProperties(fname=font_path).get_name()
    rc('font', family=font_name)

    # 최고기온
    high = []
    # 최저기온
    low = []
    # 날짜(tmef)
    xdata = []

    for row in result.values_list():
        # high.append(row[5])
        # low.append(row[4])
        # xdata.append(row[2].split('-')[2])
        print(row)
    # 이미지 청소한 후 이미지 넣기 위해
    plt.cla()
    plt.figure(figsize=(10, 6))
    plt.plot(xdata, low, label='최저기온')
    plt.plot(xdata, high, label='최고기온')
    plt.legend()
    # 이미지 저장
    plt.savefig(os.path.join(STATIC_DIR, 'images\\weather_busan.png'), dpi=300)

    plt.cla()
    plt.bar(wfs, dcounts)
    plt.savefig(os.path.join(STATIC_DIR, 'images\\weather_bar.png'), dpi=300)

    plt.cla()
    plt.pie(dcounts, labels=wfs, autopct='%.1f%%')
    plt.savefig(os.path.join(STATIC_DIR, 'images\\weather_pie.png'), dpi=300)

    image_dic = {
        'plot': 'weather_busan.png',
        'bar': 'weather_bar.png',
        'pie': 'weather_pie.png',
    }

    return image_dic
```



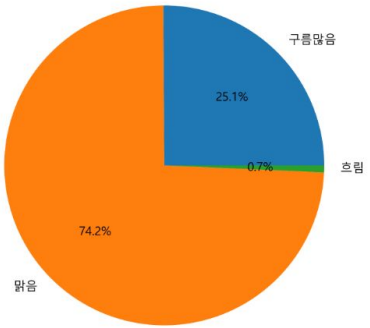




날씨정보 시각화

weather_chart.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
<div class="container my-3">
    <div class="d-flex justify-content-end mb-3">
        <form action="/weather" class="form-inline mr-3">
            <input type="text" class="form-control mr-sm-1" id="word" placeholder="Enter Search"
            name="word"
                value="{{word}}">
            <button type="submit" class="btn btn-primary">Submit</button>
        </form>
    </div>
    <div>
        <!-- {% get_static_prefix %} / 경로 / 파일명.확장자 -->
        <img src="{% get_static_prefix %}images/{{img_data.plot}}" width="100%" height="100%">
        <img src="{% get_static_prefix %}images/{{img_data.bar}}" width="100%" height="100%">
        <img src="{% get_static_prefix %}images/{{img_data.pie}}" width="100%" height="100%">
    </div>
    <table class="container table table-hover" style="text-align: center;">
        <thead>
            <tr>
                <th>번호</th>
                <th>도시</th>
                <th>날짜</th>
                <th>날씨상태</th>
                <th>최고온도</th>
                <th>최저온도</th>
            </tr>
        </thead>
        <tbody>
            {% for i in result %}
            <tr>
                <th>{{i.id}}</th>
                <th>{{i.city}}</th>
                <th>{{i.tmef}}</th>
                <th>{{i.wf}}</th>
                <th>{{i.tmx}}</th>
                <th>{{i.tmn}}</th>
            </tr>
            {% endfor %}
        </tbody>
    </table>
</div>
{% endblock %}
```



번호	도시	날짜	날씨상태	최고온도	최저온도
1	서울	2022-11-03 00:00	구름많음	14	7
2	서울	2022-11-03 12:00	구름많음	14	7
3	서울	2022-11-04 00:00	맑음	12	3
4	서울	2022-11-04 12:00	맑음	12	3

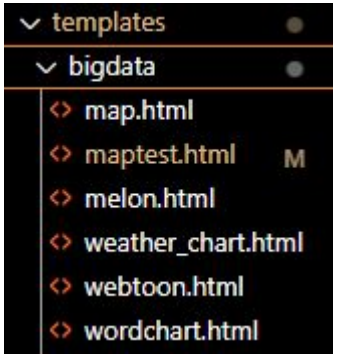
지도 시각화

views.py

```python
# map
def map(request):
    bigdataProcess.map()
    return render(request, 'bigdata/map.html')
```

settings.py

```python
TEMPLATE_DIR = os.path.join(BASE_DIR, 'myapp03/templates')
```



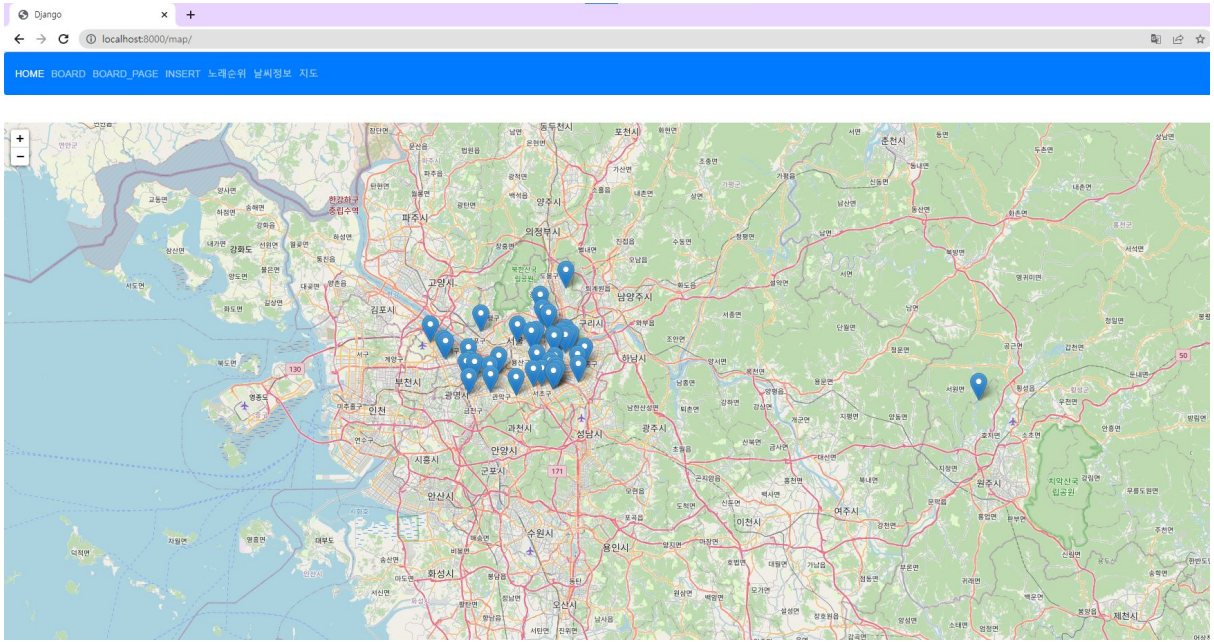
bigdataProcess.py

```python
# 지도
def map():
    ex = {'경도': [127.061026, 127.047883, 127.899220, 128.980455, 127.104071, 127.102490, 127.
    088387, 126.809957, 127.010861, 126.836078, 127.014217, 126.886859, 127.031702, 126.880898,
    127.028726, 126.897710, 126.910288, 127.043189, 127.071184, 127.076812, 127.045022, 126.
    982419, 126.840285, 127.115873, 126.885320, 127.078464, 127.057100, 127.020945, 129.068324,
    129.059574, 126.927655, 127.034302, 129.106330, 126.980242, 126.945099, 129.034599, 127.
    054649, 127.019556, 127.053198, 127.031005, 127.058560, 127.078519, 127.056141, 129.034605,
    126.888485, 129.070117, 127.057746, 126.929288, 127.054163, 129.060972],
          '위도': [37.493922, 37.505675, 37.471711, 35.159774, 37.500249, 37.515149, 37.549245,
          37.562013, 37.552153, 37.538927, 37.492388, 37.480390, 37.588485, 37.504067, 37.608392,
          37.503693, 37.579029, 37.580073, 37.552103, 37.545461, 37.580196, 37.562274, 37.535419,
          37.527477, 37.526139, 37.648247, 37.512939, 37.517574, 35.202902, 35.144776, 37.499229,
          35.150069, 35.141176, 37.479403, 37.512569, 35.123196, 37.546718, 37.553668, 37.488742,
          37.493653, 37.498462, 37.556602, 37.544180, 35.111532, 37.508058, 35.085777, 37.546103,
          37.483899, 37.489299, 35.143421],
          '구분': ['음식', '음식', '음식', '음식', '생활서비스', '음식', '음식', '음식', '음식',
          '음식', '음식', '음식', '음식', '음식', '음식', '음식', '음식', '음식', '소매', '음식', '음식',
          '음식', '음식', '소매', '음식', '소매', '음식', '음식', '음식', '음식', '음식', '음식',
          '음식', '음식', '음식', '음식', '소매', '음식', '음식', '의료', '음식', '음식', '음식',
          '소매', '음식', '음식', '음식', '음식', '음식', '음식', '음식']}
    ex_data = DataFrame(ex)
    # print(ex_data)

    # 위도 and 경도
    lat = ex_data['위도'].mean()
    long = ex_data['경도'].mean()

    m = folium.Map([lat, long], zoom_start=10)

    for i in ex_data.index:
        sub_lat = ex_data.loc[i, '위도']
        sub_long = ex_data.loc[i, '경도']

        title = ex_data.loc[i, '구분']

        # 지도에 데이터 찍어서 보여주기
        folium.Marker([sub_lat, sub_long], tooltip=title).add_to(m)
        # 웹페이지로 저장
        m.save(os.path.join(TEMPLATE_DIR, 'bigdata/maptest.html'))
```

지도 시각화

map.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}

{% include "bigdata/maptest.html" %}

{% endblock %}
```



워드 클라우드 시각화

views.py

```
# wordcloud
def wordcloud(requset):
    w_path = "C:/Django_study03/myProject03/data/"
    # json 읽기
    data = json.loads(open(w_path+'4차 산업혁명.json',
    |   |   |   |   | 'r', encoding='utf-8').read())
    # print(data)
    bigdataProcess.make_wordCloud(data)
    return render(requset, 'bigdata/wordchart.html', {'img_data': 'k_wordCloud.png'})
```

워드 클라우드 시각화

bigdataProcess.py

```python
# wordCloud
def make_wordCloud(data):
    message = ''

    for item in data:
        if 'message' in item.keys():
            message = message + re.sub(r'[^\w]', ' ', item['message'])+''

    nlp = Okt()
    # 명사 추출
    message_N = nlp.nouns(message)
    # 명사의 갯수 추출
    count = Counter(message_N)

    word_count = {}

    for tag, counts in count.most_common(80):
        if(len(str(tag)) > 1):
            word_count[tag] = counts
            # print("%s : %d" % (tag, counts))

    font_path = 'c:/Windows/fonts/malgun.ttf'
    wc = WordCloud(font_path, background_color='ivory', width=800, height=600)

    cloud = wc.generate_from_frequencies(word_count)

    plt.figure(figsize=(8, 8))
    plt.imshow(cloud)
    plt.axis('off')
    cloud.to_file('./static/images/k_wordCloud.png')
```

워드 클라우드 시각화

wordchart.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
<div class="container my-3">
    <div>
        <!-- {% get_static_prefix %} / 경로 / 파일명.확장자 -->
        <img src="{% get_static_prefix %}images/{{img_data}}" width="100%" height="100%">
    </div>
</div>
{% endblock %}
```

자유주제 시각화

views.py

```
def webtoon(request):
    data = {}
    bigdataProcess.webtoon_crawing(data)
    # data
    for i in data:
        for j in data[i]:
            dto = Webtoon(webDay=i, title=j[0], writer=j[1], score=j[2])
            dto.save()

    # 검색
    word = request.GET.get('word', '')
    field = request.GET.get('field', 'title')

    if field == 'all':
        boardList = Webtoon.objects.filter(Q(title__contains=word) |
                                           Q(writer__contains=word)).order_by('-id')
    elif field == 'title':
        boardList = Webtoon.objects.filter(
            Q(title__contains=word)).order_by('-id')
    elif field == 'writer':
        boardList = Webtoon.objects.filter(
            Q(writer__contains=word)).order_by('-id')
    else:
        boardList = Webtoon.objects.all().order_by('-id')

    context = {'field': field,
               'word': word,
               'boardList': boardList}

    result = Webtoon.objects.filter(Q(title__contains=word))

    return render(request, 'bigdata/webtoon.html', {"result": result,  "context": context, "img_data":
    'webtoon_wordCloud.png'})
```

자유주제 시각화

bigdataProcess.py

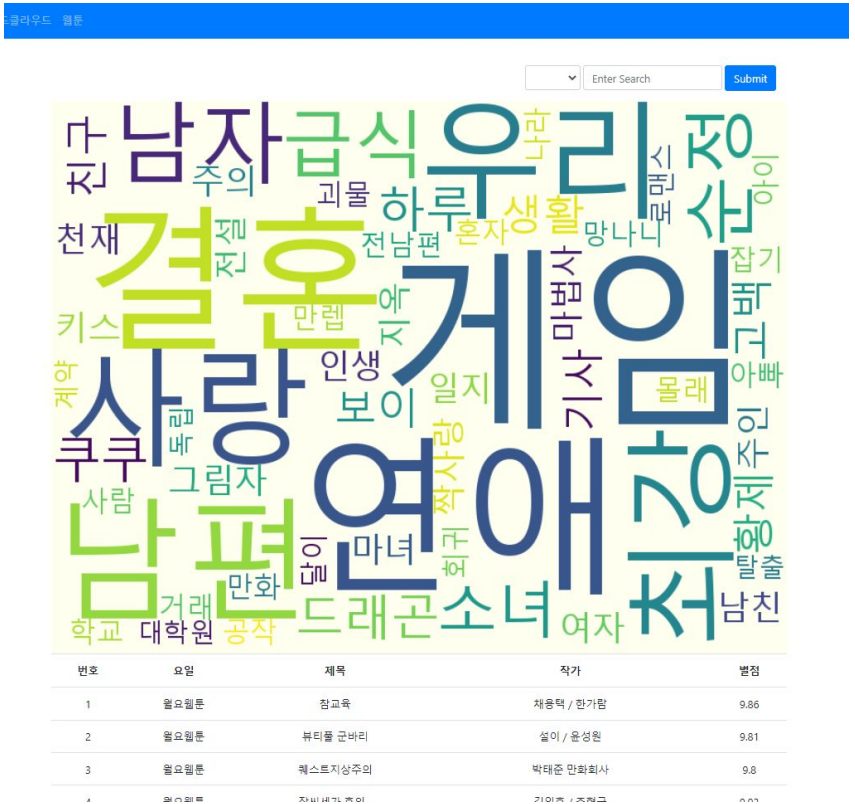
```python
# 웹툰
def webtoon_crawing(webtoon):
    dayName = ['mon', 'tue', 'wed', 'thu', 'fri', 'sat', 'sun']
    for i in range(0, 7):
        url = 'https://comic.naver.com/webtoon/weekdayList?week=' + dayName[i]

        # Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36
        header = {'User-Agent': 'Mozilla/5.0'}
        response = requests.get(url, headers=header)
        html = response.text

        soup = BeautifulSoup(html, 'html.parser')

        # day ul 태그
        day_ul = soup.find('ul', {'class': 'category_tab'})
        # print(day_ul)
        web_ul = soup.find('ul', {'class': 'img_list'})

        day_li = soup.find('ul', {'class': 'img_list'}).find_all('li')
        # print(day_li)
        web_li = web_ul.find_all('li')
        # print(len(web_li))

        day_li = day_ul.find_all('li')
        days = day_li[i+1].text
        webtoon[days] = []
        # print(days)

        for j in web_ul.find_all('li'):
            datas = []
            # if len(lis) > 0:
            webtoon_name = j.find('dl').find('a').text
            webtoon_writter = j.find(
                'dd', {'class': 'desc'}).find('a').text
            webtoon_scope = j.find(
                'div', {'class': 'rating_type'}).find('strong').text
            # print('제목 :', webtoon_name)
            # print('작가 :', webtoon_writter)
            # print('별점 :', webtoon_scope)
            if Webtoon.objects.filter(title=webtoon_name).exists() == False:
                datas.append(webtoon_name)
                datas.append(webtoon_writter)
                datas.append(webtoon_scope)
                webtoon[days].append(datas)
            # print('-'*100)
```

```python
message_list = Webtoon.objects.filter().values_list(
    'title', flat=True).order_by('title')
message = ''
for i in message_list:
    message += i
nlp = Okt()
# 명사 추출
message_N = nlp.nouns(message)
# 명사의 갯수 추출
count = Counter(message_N)

word_count = {}

for tag, counts in count.most_common(80):
    if(len(str(tag)) > 1):
        word_count[tag] = counts
        # print("%s : %d" % (tag, counts))

font_path = 'c:/Windows/fonts/malgun.ttf'
wc = WordCloud(font_path, background_color='ivory', width=800, height=600)

cloud = wc.generate_from_frequencies(word_count)

plt.figure(figsize=(8, 8))
plt.imshow(cloud)
plt.axis('off')
cloud.to_file('./static/images/webtoon_wordCloud.png')
```

자유주제 시각화

webtoon.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
<div class="container my-3">
    <div class="d-flex justify-content-end mb-3">
        <form action="/webtoon" class="form-inline mr-3">
            <select class="form-control mr-sm-1" name="field" id="field">
                <option value="all">전체</option>
                <option value="title">제목</option>
                <option value="writer">작가</option>
                <!-- 지정한 option이 안 뜰리는 설정 -->
                <script>
                    $("#field").val("{{field}}")
                </script>
            </select>
            <input type="text" class="form-control mr-sm-1" id="word" placeholder="Enter Search" name="word"
                value="{{word}}">
            <button type="submit" class="btn btn-primary">Submit</button>
        </form>
    </div>
    <div>
        <!-- {% get_static_prefix %} / 경로 / 파일명.확장자 -->
        <!-- <img src="{% get_static_prefix %}images/{{img_data.plot}}" width="100%" height="100%"> -->
    </div>
    <table class="container table table-hover" style="text-align: center;">
        <thead>
            <tr>
                <th>번호</th>
                <th>요일</th>
                <th>제목</th>
                <th>작가</th>
                <th>별점</th>
            </tr>
        </thead>
        <tbody>
            <!-- for문 시작(django) -->
            {% for i in result %}
            <tr>
                <td>{{i.id}}</td>
                <td>{{i.webDay}}</td>
                <td>{{i.title}}</td>
                <td>{{i.writer}}</td>
                <td>{{i.score}}</td>
            </tr>
            <!-- for문 끝(django) -->
            {% endfor %}
        </tbody>
    </table>
</div>
{% endblock %}
```



End