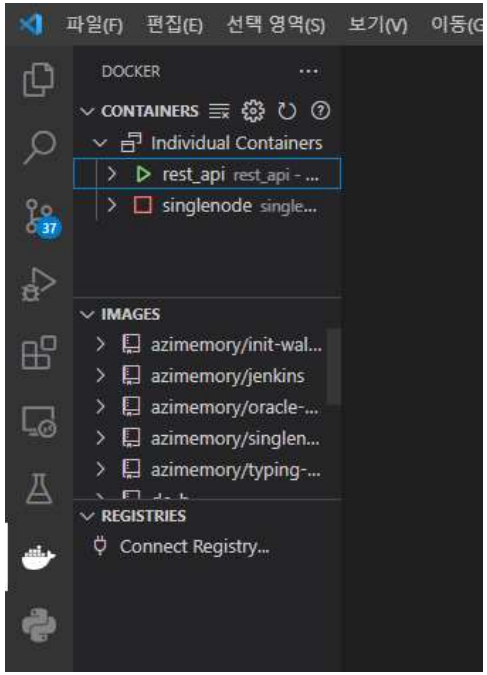
VsCode – Docker Container 연동

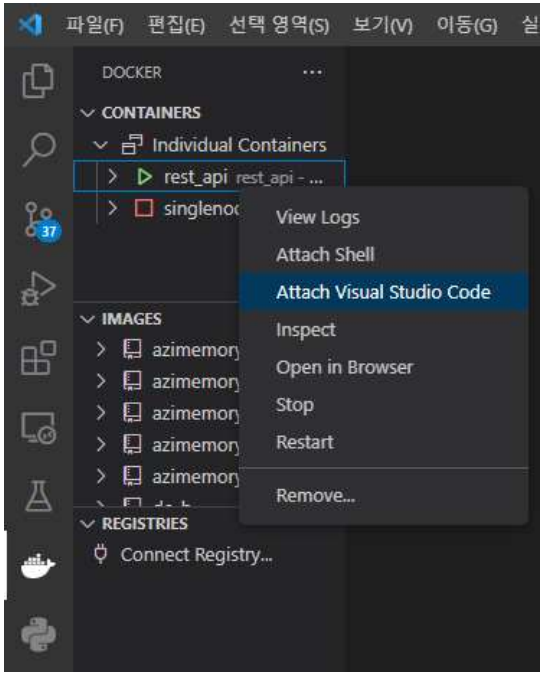
1. VSCODE 설치

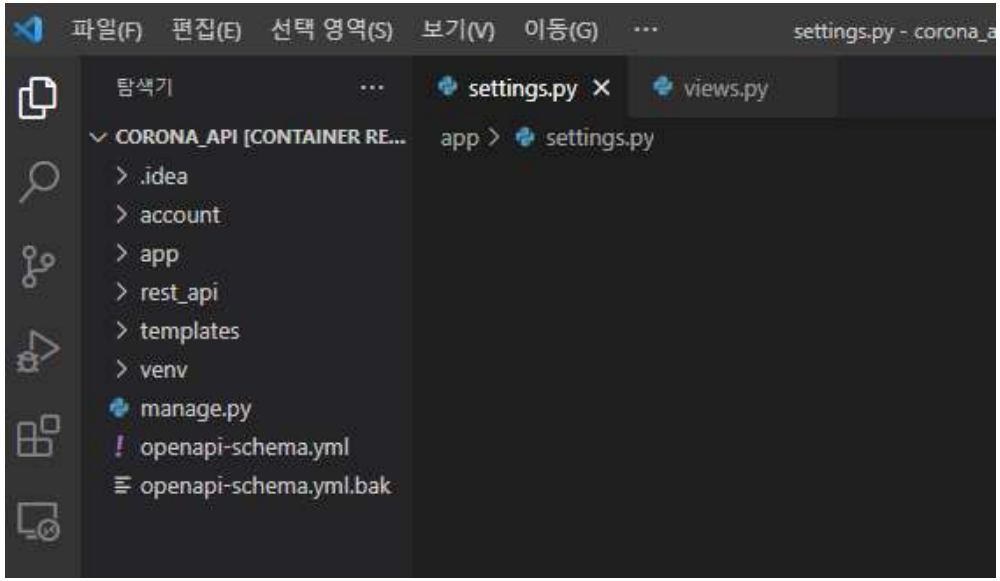
2. Python, Pylint Extention 설치

3. Docker, Remote-Container Extention 설치

4. 좌측에서 Docker Tab을 선택한 이후 VSCODE와 를 통해 접근할 컨테이너 우클릭 > Attach Visual Studio Code 선택







vscode 초기설정

1. settings.json에 아래 내용 추가하여 root 디랙토리 지정

f1 -> settings.json -> 작업영역 settings.json 열기

```
"python.linting.pylintArgs": [
        "--init-hook",
        "import sys; sys.path.append('/home/big/study/CORONA_EXTRACTOR')"
        "--rcfile",
        "pylint.config",
        "--disable",
        "missing-docstring",
]
```

프로젝트 디랙토리에 pylint.config 파일 생성, 아래 내용 작성 후 저장

```
[MESSAGES CONTROL]
#C0111 Missing docstring
#C0103 Invalid constant name
#C0301 Line too long
#C0303 trailing whitespace
disable=C0111,C0103,C0303,C0301
```

2. ctrl - shift - p 누른 다음 사용할 파이썬 인터프리터를 선택하여 pylance 경고 해결

3. ~/.bashrc에 export PATH=$/home/big/.local/bin:$PATH

Oracle ATP – Django 연동

ref: https://blogs.oracle.com/opal/post/connecting-to-oracle-cloud-autonomous-database-with-django#connect

1. Oracle Instant Client 설치
https://www.oracle.com/database/technologies/instant-client/downloads.html

2. 다운받은 Oracle Instant Client 파일을 보안이 적용되는 아래의 경로에 압축해제
cd /usr/lib
sudo wget https://download.oracle.com/otn_software/linux/instantclient/1916000/instantclient-basic-linux.x64-19.16.0.0.0dbru.zip
sudo unzip instantclient-basic-linux.x64-19.16.0.0.0dbru.zip

3. Oracle wallet 폴더에서 cwallet.sso, sqlnet.ora, and tnsnames.ora 파일을 복사하여 아래 경로에 붙여 넣기
sudo cp /home/big/study/db/Wallet_DECORONA1/* /usr/lib/instantclient_19_16/network/admin/

4. sqlnet.ora 파일의 WALLET_LOCATION > METHOD_DATA > DIRECTORY 수정

cd /usr/lib/instantclient_19_16/network/admin
sudo vim sqlnet.ora

WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA = (DIRECTORY="/usr/lib/instantclient_19_16/network/admin")))
SSL_SERVER_DN_MATCH=yes

5. cx_Oracle 설치
ref : https://cx-oracle.readthedocs.io/en/latest/user_guide/installation.html#installing-cx-oracle-on-linux

# 리눅스에서 cx_Oracle를 사용하기 위해 libaio 설치
sudo apt install libaio1
sudo sh -c "echo /usr/lib/instantclient_19_16> /etc/ld.so.conf.d/oracle-instantclient.conf"
sudo ldconfig

파이썬에서 Oracle에 접근하도록 해주는 라이브러리
pip install cx_Oracle

6. 환경 변수 등록

export LD_LIBRARY_PATH = /usr/lib/instantclient_19_16:$ LD_LIBRARY_PATH
export TNS_ADMIN = /usr/lib/instantclient_19_16/network/admin
export PATH = /usr/lib/instantclient_19_16:$PATH

7. django 프로젝트 생성

pip install django
django-admin startproject app .
django-admin startapp rest_api

8. settings.py 작성

```
DATABASES = {
    'default':  {
            'ENGINE':'django.db.backends.oracle',
            'NAME':'decorona1_high', # tnsnames.ora 파일에 등록된 NAME을 등록
            'USER':'dm_admin',
            'PASSWORD':'123qwe!@#QWE', #Please provide the db password here
    }
}
```

9. 기존 데이터베이스 테이블로 models.py 파일 생성

python manage.py inspectdb > rest_api/models.py

ValueError: source code string cannot contain null bytes 나는 경우
models.py 가 현재 UTF-8 이외의 인코딩이어서 생기는 문제
models.py 내용 복사 > models.py 삭제 > 빈 models.py 생성 > 붙여넣기

Django Rest Framework

Quick Start

Django Rest Framewrok Quick Start

ref : https://www.django-rest-framework.org/tutorial/quickstart/

1. djangorestframework 설치

pip install djangorestframework

2. settings.py 파일에 rest_framework, 새롭게 작성할 앱 등록

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'rest_api',
]
```

3. settings.py 파일에 REST_FRAMEWORK 설정 등록

```
REST_FRAMEWORK = {
    'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.PageNumberPagination',
    'PAGE_SIZE': 10,
}
```

4. serializers.py 파일 생성

사용하는 model 별 Serializer 클래스 생성

```python
from django.db import models

# Create your models here.
from rest_framework import serializers
from rest_api.models import *


class CoFacilitySerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = CoFacility
        fields = ['loc', 'fac_popu', 'qur_rate', 'std_day']
```

5. views.py에 ViewSet 작성 - ViewSet은 GenericView 의 후손클래스이다.

GenericView : https://www.django-rest-framework.org/api-guide/generic-views/

ViewSet : https://www.django-rest-framework.org/api-guide/viewsets/

```python
class CoPopuDensityViewSet(viewsets.ReadOnlyModelViewSet):
    queryset = CoPopuDensity.objects.all().order_by('-std_day')
    serializer_class = CoPopuDensitySerializer
    permission_classes = [permissions.IsAuthenticated]

    def list(self, request):
        page = self.paginate_queryset(self.queryset)
        serializers = self.get_serializer(page, many=True)
        return Response(serializers.data)
```

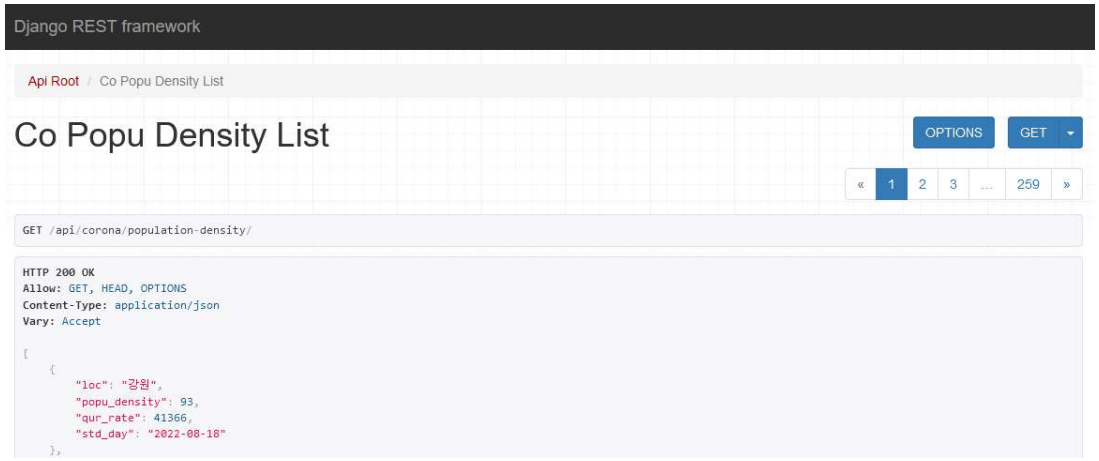
6. app 모듈의 urls.py에 router를 사용해 url 및 viewSet 등록

```
router = routers.DefaultRouter()
router.register(r'api/corona/population-density', views.CoPopuDensityViewSet)

urlpatterns = [
    path('', include(router.urls)),
]
```

7. localhost:8000 접속

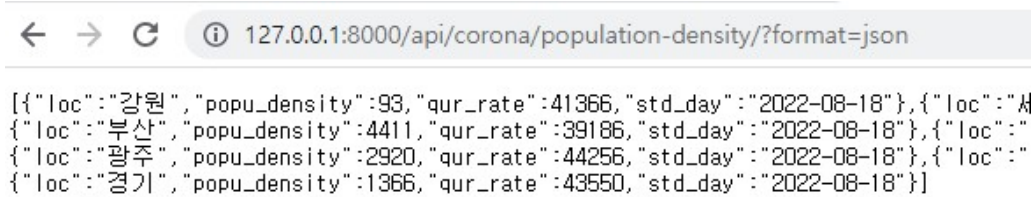
8. api 요청



9. json 양식으로 요청

요청 url에 queryString : format=json 추가

http://127.0.0.1:8000/api/corona/population-density/?format=json



10. api HTML 페이지 커스타마이징

ref : https://www.django-rest-framework.org/topics/browsable-api/

templates > rest_framework > api.html 생성
rest_framework/base.html을 상속하여 UI 수정
  * 외부 라이브러리 > rest_framework > templates > rest_framework > base.html 에서 확인

```
{% extends "rest_framework/base.html" %}
{% load static %}

{% block style %}
    <link rel="stylesheet" type="text/css" href="{% static "rest_framework/css/prettify.css" %}"/>
    <link rel="stylesheet" type="text/css" href="{% static "rest_framework/css/default.css" %}"/>

    <link rel="stylesheet" type="text/css" href="{% static "rest_framework/css/bootstrap.min.css" %}"/>
    <link rel="stylesheet" type="text/css" href="{% static "rest_framework/css/bootstrap-tweaks.css" %}"/>
{% endblock %}

<!--https://www.django-rest-framework.org/topics/browsable-api/-->
{% block body %}

<ol class="breadcrumb">
    <li><a href="#">DE</a></li>
```

Django Rest Framework

Token Authentication

ref : https://www.django-rest-framework.org/api-guide/authentication/#tokenauthentication

1. rest_framework.authtoken 추가

```
INSTALLED_APPS = [
                ...
                'rest_framework.authtoken'
]
```

2. python manage.py migrate 진행

3. settings.py 파일에 DEFAULT_AUTHENTICATION_CLASSES 옵션 추가

```
REST_FRAMEWORK = {
 ...
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.TokenAuthentication',
    ],

 ...
```

4. account app 생성

5. templates/account 에 login.html, signup.html, apikey.html 파일 생성

6. account app의 urls.py 작성

```
app_name = 'account'

urlpatterns = [
    path('login/',  auth_views.LoginView.as_view(template_name='account/login.html'), name='login'),
    path('logout/', auth_views.LogoutView.as_view(), name='logout'),
    path('signup/', views.signup, name='signup'),
    path('apikey/', views.get_apikey, name='apikey'),
]
```

7. account app의 views.py 작성

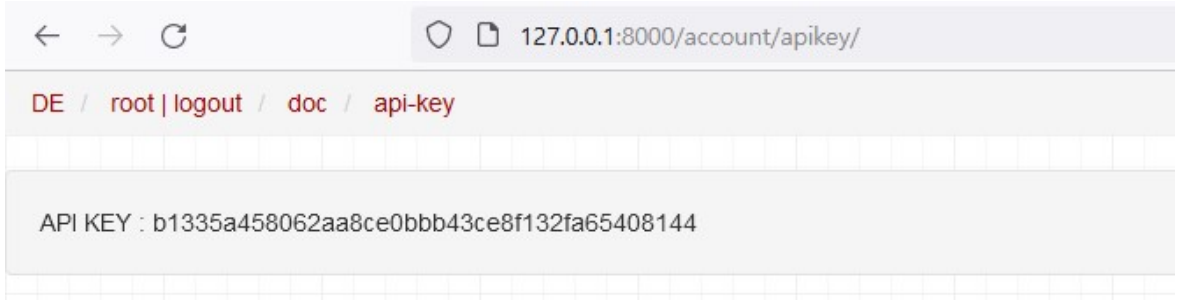
```
# Create your views here.
def signup(request):
    if request.method == 'POST':
        form = UserForm(request.POST)
        if form.is_valid():
            form.save()

            user_name = form.cleaned_data.get('username')
            raw_password = form.cleaned_data.get('password1')
            user = authenticate(username=user_name, password=raw_password)
            login(request, user)
            return redirect('/')
    else:
        form = UserForm()

    return render(request, 'account/signup.html', {'form': form})


@login_required(login_url='account:login')
def get_apikey(request):
    token = Token.objects.get_or_create(user=request.user)
    return render(request, 'account/apikey.html', {'apikey': token[0]})
```

8. apikey 생성 화면



Django Rest Framework

API DOCUMENT PAGE 생성

ref : https://www.django-rest-framework.org/api-guide/schemas/
https://www.django-rest-framework.org/topics/documenting-your-api/
https://drf-yasg.readthedocs.io/en/stable/

1. swagger.ui 를 사용하기 위해 drf-yasg 설치

pip install drf-yasg

settings.py 파일에 app 추가

```
INSTALLED_APPS = [
    ...
    'drf_yasg',
    ...
]
```

2. urls.py 파일 작성

```
schema_view = get_schema_view(
   openapi.Info(
      title="CORONA_API",
      default_version='v2',
      description="CORONA_API description",
   ),
   public=True,
)

urlpatterns = [
    path('', views.index),
    path('api/', include(router.urls)),
    path('accounts/', include('account.urls')),
    path('doc/', schema_view.with_ui('swagger', cache_timeout=0), name='doc'),
]
```

3. views.py 의 함수에 @swagger_auto_schema 어노테이션을 사용해 원하는 문서 내용을 지정

ref : https://drf-yasg.readthedocs.io/en/stable/custom_spec.html?highlight=swagger_auto_schema#the-swagger-auto-schema-decorator

```
@swagger_auto_schema(
    operation_summary="인구밀도와 10만명당 코로나 발생 환자 간의 상관관계",
    operation_description="""시작날짜와 끝날짜를 모두 생략하면 최근 1주일 데이터를 반환합니다. <br>
    시작날짜만 입력하면 시작날짜 이후의 데이터를 반환합니다.<br>
    끝날짜만 입력하면 끝날짜 이전 데이터를 반환합니다.<br>
    1 page는 10개의 데이터로 구성되어 있습니다. """,
    manual_parameters=[
        Parameter("start_date", IN_QUERY, type=TYPE_STRING,
                  description="시작 날짜, (format :yyyy-MM-dd), (required : False)"),
        Parameter("end_date", IN_QUERY, type=TYPE_STRING,
                  description="끝날짜, (format :yyyy-MM-dd), (required : False)", required=False),
    ],
)
```

4. swagger-ui.html 수정

ref : https://drf-yasg.readthedocs.io/en/stable/custom_ui.html

1. templates 아래에 drf-yasg 폴더 생성
2. 외부 라이브러리 > drf_yasg > templates > drf-yasg > swagger-ui.html 파일을 복사하여 위에서 생성한 폴더에 붙여넣기
3. 원하는 대로 수정

JSON형태로 응답하도록 설정

1. settings.py 파일에 REST_FRAMEWORK의 DEFAULT_AUTHENTICATION_CLASSES, DEFAULT_RENDERER_CLASSES, DEFAULT_PARSER_CLASSES 수정

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.TokenAuthentication',
    ],

    'DEFAULT_RENDERER_CLASSES': (
        'rest_framework.renderers.JSONRenderer',
    ),

    'DEFAULT_PARSER_CLASSES': (
        'rest_framework.parsers.JSONParser',
    )
}
```

2. views.py 파일의 응답객체를 Response에서 JsonResponse로 변경

```
def list(self, request):
    query_params = request.query_params
    queryset = get_queryset_by_date(CoWeekday, query_params)
    serializer = self.get_serializer(queryset, many=True)


    return JsonResponse(serializer.data, safe=False)
```