Test & Documentation

TDD

- TDD
  - ✓ Test Driven Development, 테스트 주도 개발이라는 뜻의 TDD는 개발에 있어 그 진행을 담당하는 주체를 테스트에 두고 있음
  - ✓ 테스트라는 것에는 종류가 여러 가지 있는데 TDD에서의 테스트는 Unit Test(단위 테스트)로 함수 단위로 해당 함수가 잘 작동하는지 확인하는 테스트
  - ✓ 프로세스
    - 구현하려는 기능에 대한 테스트 코드를 작성
    - 테스트를 실행시키고 기능이 없으니 실패
    - 테스트를 통과할 수 있는 최소한의 기능을 구현
    - 테스트를 실행시키고 통과시키면 코드를 정리
    - 모든 기능을 구현할 때까지 이를 반복
  - ✓ 주요한 개념은 기능을 구현하기에 앞서 해당 기능에 대한 테스트 코드를 먼저 작성한다는 점이며 기능을 구현하는 것은 오직 테스트를 통과하는 것에만 집중해 진행
  - ✓ 테스트를 통과할 만큼의 기능 코드가 완성되면 해당 코드를 적당히 깔끔하게 잘 정리하여 해당 기능에 대한 구현을 마무리
  - ✓ 적어도 기능 별로 충분히 검증된 코드를 작성하는 것은 안정적인 서비스를 만드는 데 필요한 첫 단계

TDD

- 기본 설정
  - ✓ 가상 환경 생성
  - ✓ 필요한 패키지 설치
    - pip install django
    - pip install djangorestframework
    - pip install mysqlclient
  - ✓ 프로젝트 와 앱 생성
    - 프로젝트 생성: django-admin startproject api .
    - 앱 생성: python manage.py startapp book

TDD

- 기본 설정
  - ✓ settings.py 수정

```
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = "django-insecure-
    =5!6(l=%dj(=f5uf2l+w7f4&h5)%d_k9_x$2+i1gto0%@1&hyj"

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []
```

TDD

- 기본 설정
  - ✓ settings.py 수정

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

MIDDLEWARE = [
    "django.middleware.security.SecurityMiddleware",
    "django.contrib.sessions.middleware.SessionMiddleware",
    "django.middleware.common.CommonMiddleware",
    "django.middleware.csrf.CsrfViewMiddleware",
    "django.contrib.auth.middleware.AuthenticationMiddleware",
    "django.contrib.messages.middleware.MessageMiddleware",
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
]
```

TDD

- 기본 설정
  - ✓ settings.py 수정

```
ROOT_URLCONF = "apiserver.urls"

TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "DIRS": [],
        "APP_DIRS": True,
        "OPTIONS": {
            "context_processors": [
                "django.template.context_processors.debug",
                "django.template.context_processors.request",
                "django.contrib.auth.context_processors.auth",
                "django.contrib.messages.context_processors.messages",
            ],
        },
    },
]

WSGI_APPLICATION = "apiserver.wsgi.application"
```

TDD

- 기본 설정
  - ✓ settings.py 수정

```
# Database
# https://docs.djangoproject.com/en/4.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'adam',
        'USER': 'root',
        'PASSWORD': 'wnddkd', # mariaDB 설치 시 입력한 root 비밀번호 입력
        'HOST': '127.0.0.1',
        'PORT': ''
    }
}
```

TDD

- 기본 설정
  - ✓ settings.py 수정

```
# Password validation
# https://docs.djangoproject.com/en/4.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        "NAME": "django.contrib.auth.password_validation.UserAttributeSimilarityValidator",
    },
    {"NAME": "django.contrib.auth.password_validation.MinimumLengthValidator",},
    {"NAME": "django.contrib.auth.password_validation.CommonPasswordValidator",},
    {"NAME": "django.contrib.auth.password_validation.NumericPasswordValidator",},
]
```

TDD

- 기본 설정
  - ✓ settings.py 수정

```
# Internationalization
# https://docs.djangoproject.com/en/4.1/topics/i18n/

LANGUAGE_CODE = "en-us"
TIME_ZONE = "Asia/Seoul"
USE_I18N = True
USE_TZ = True
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.1/howto/static-files/

STATIC_URL = "static/"

# Default primary key field type
# https://docs.djangoproject.com/en/4.1/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = "django.db.models.BigAutoField"
```

TDD

- 모델 테스트
  - ✓ book 애플리케이션에 tests.py 파일에 클래스 작성

```
from django.test import TestCase
from .models import Book

class ModelTest(TestCase):
    def setUp(self):
        self.book_title = "My Book"
        self.book_author = "ADAM"
        self.book = Book(title=self.book_title, author=self.book_author)

    def test_model_can_create_a_bucketlist(self):
        old_count = Book.objects.count()
        self.book.save()
        new_count = Book.objects.count()
        self.assertNotEqual(old_count, new_count)
```

TDD

- 모델 테스트
  - ✓ 터미널에서 수행 - python manage.py test

```
(venv) (base) adam@Adamsoftui-MacBookPro api % python manage.py test
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
E
======================================================================
ERROR: test_model_can_create_a_bucketlist (api.tests.ModelTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/adam/PycharmProjects/tdd/api/api/tests.py", line 7, in setUp
    self.book = Book(title=self.book_title, author=self.book_author)
NameError: name 'Book' is not defined

----------------------------------------------------------------------
Ran 1 test in 0.002s

FAILED (errors=1)
Destroying test database for alias 'default'...
```

TDD

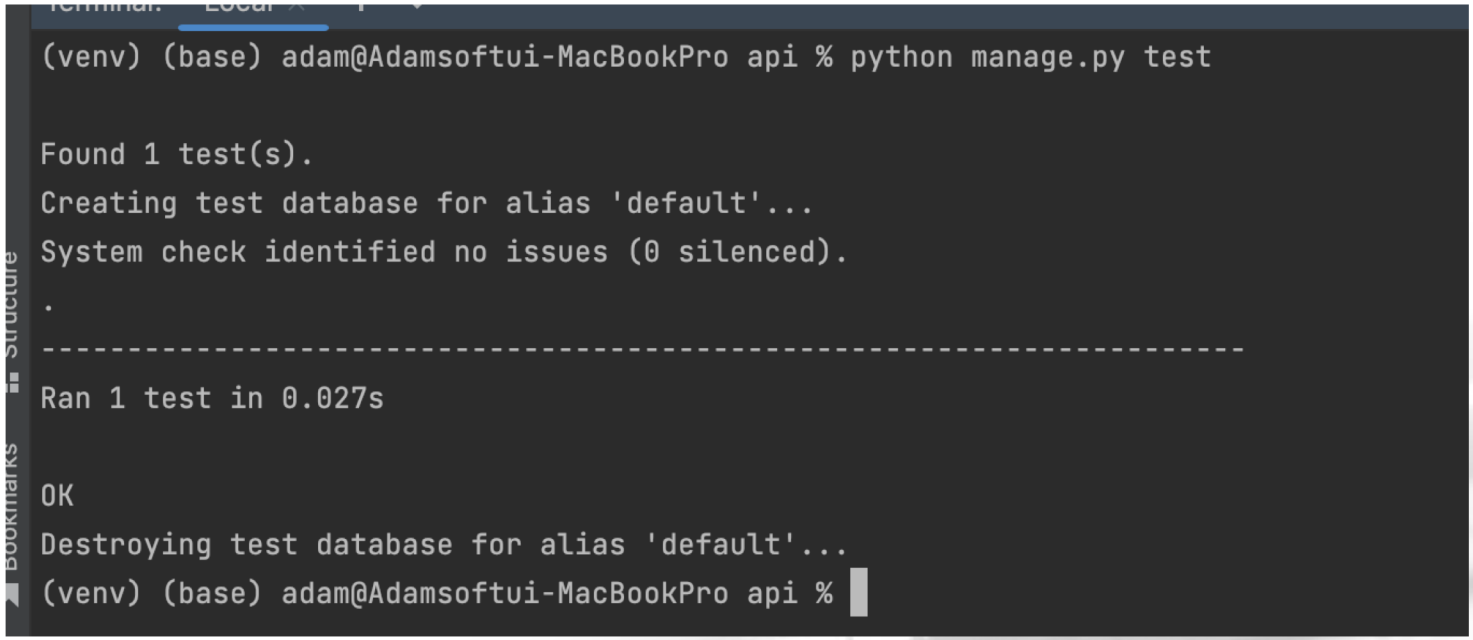
- 모델 테스트
  - ✓ book 애플리케이션의 models.py 파일에 Book 모델 생성

```
from django.db import models

class Book(models.Model):
    title = models.CharField(max_length=128)
    author = models.CharField(max_length=128)
```

TDD

- 모델 테스트
  - ✓ python manage.py makemigrations
  - ✓ python manage.py migrate
  - ✓ python manage.py test



TDD

- 뷰 테스트
  - ✓ tests.py 파일에 View 테스트를 위한 클래스 추가

```
from rest_framework.test import APIClient
from rest_framework import status

class ViewTest(TestCase):
    def setUp(self):
        self.client = APIClient()
        self.book_data = {'title': 'My Book 2', 'author': 'TaeBbong'}
        self.response = self.client.post('/api/books/',
                                         self.book_data,
                                         format="json")

    def test_api_can_create_a_book(self):
        print(self.response.content)
        self.assertEqual(self.response.status_code, status.HTTP_201_CREATED)
```

TDD

- 뷰 테스트
  - ✓ 테스트 – python manage.py test

```
(venv) (base) adam@Adamsoftui-MacBookPro api % python manage.py test

Found 2 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.b'\n<!doctype html>\n<html lang="en">\n<head>\n  <title>Not Found</title>\n</head>\n<body>\n  <h1>Not Found</h1><p>The requested resource was not found on this server
.</p>\n</body>\n</html>\n'
F
======================================================================
FAIL: test_api_can_create_a_book (book.tests.ViewTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/adam/PycharmProjects/tdd/api/book/tests.py", line 29, in test_api_can_create_a_book
    self.assertEqual(self.response.status_code, status.HTTP_201_CREATED)
AssertionError: 404 != 201

----------------------------------------------------------------------
Ran 2 tests in 0.060s

FAILED (failures=1)
Destroying test database for alias 'default'...
(venv) (base) adam@Adamsoftui-MacBookPro api %
```

TDD

- 뷰 테스트
  - ✓ serializers.py 파일을 생성하고 작성

```
from rest_framework import serializers
from .models import Book

class BookSerializer(serializers.ModelSerializer):
    class Meta:
        model = Book
        fields = ('id', 'title', 'author')
```

TDD

- 뷰 테스트
  - ✓ views.py 파일에 작성

```
from rest_framework import generics
from .serializers import BookSerializer
from .models import Book

class CreateView(generics.CreateAPIView):
    queryset = Book.objects.all()
    serializer_class = BookSerializer

    def perform_create(self, serializer):
        serializer.save()
```

TDD

- 뷰 테스트
  - ✓ book 애플리케이션에 urls.py 파일을 생성하고 작성

```
from django.urls import path
from .views import CreateView

urlpatterns = [
    path('books/', CreateView.as_view()),
]
```

TDD

- 뷰 테스트
  - ✓ 프로젝트의 urls.py 파일에 작성

```
from django.urls import path, include
from django.contrib import admin

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('api.urls'))
]
```

TDD

- 뷰 테스트
  - ✓ 테스트

```
(venv) (base) adam@Adamsoftui-MacBookPro api % python manage.py test

Found 2 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.b'{"id":2,"title":"My Book 2","author":"TaeBbong"}'
.
----------------------------------------------------------------------
Ran 2 tests in 0.066s

OK
Destroying test database for alias 'default'...
```

문서화

- 문서화의 필요성
  - ✓ 모든 개발의 목적은 누군가가 편하게 쓰게 하기 위함에 있음
  - ✓ 개발자 간 소통을 하기 위해서 결과물에 대한 설명은 필수적이며 백엔드 개발자들에게는 API라는 결과물에 대한 API 문서가 필요
  - ✓ 문서를 만드는 경험을 해본 적 있는 분들이라면 이 작업이 아주아주 귀찮다는 것을 알 것
  - ✓ API 문서화를 자동으로 도와주는 Swagger라는 도구가 있고 이것이 발전된 형태가 drf_yasg
  - ✓ API의 각 endpoint별로 주소, 요청 메서드, 입력해야 하는 데이터와 각 데이터 별 타입, 그리고 응답 양식까지 한 번에 정리해둔 문서를 손쉽게 생성할 수 있음

문서화

- drf_yasg 패키지 적용
  - ✓ 패키지 적용: pip install drf-yasg
  - ✓ settings.py 파일에 앱 등록

```
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "rest_framework",
    "book",
    "drf_yasg"
]
```

문서화

- drf_yasg 패키지 적용
  - ✓ 프로젝트의 url.py 수정

```
from django.contrib import admin
from django.urls import path, include, re_path
from django.conf import settings
from rest_framework import permissions
from drf_yasg.views import get_schema_view
from drf_yasg import openapi

schema_view = get_schema_view(
    openapi.Info(
        title="Swagger Study API",
        default_version="v1",
        description="Swagger Study를 위한 API 문서",
        terms_of_service="https://www.google.com/policies/terms/",
        contact=openapi.Contact(name="test", email="dhkep03@gmail.com"),
        license=openapi.License(name="Test License"),
    ),
    public=True,
    permission_classes=(permissions.AllowAny,),
)
```

문서화

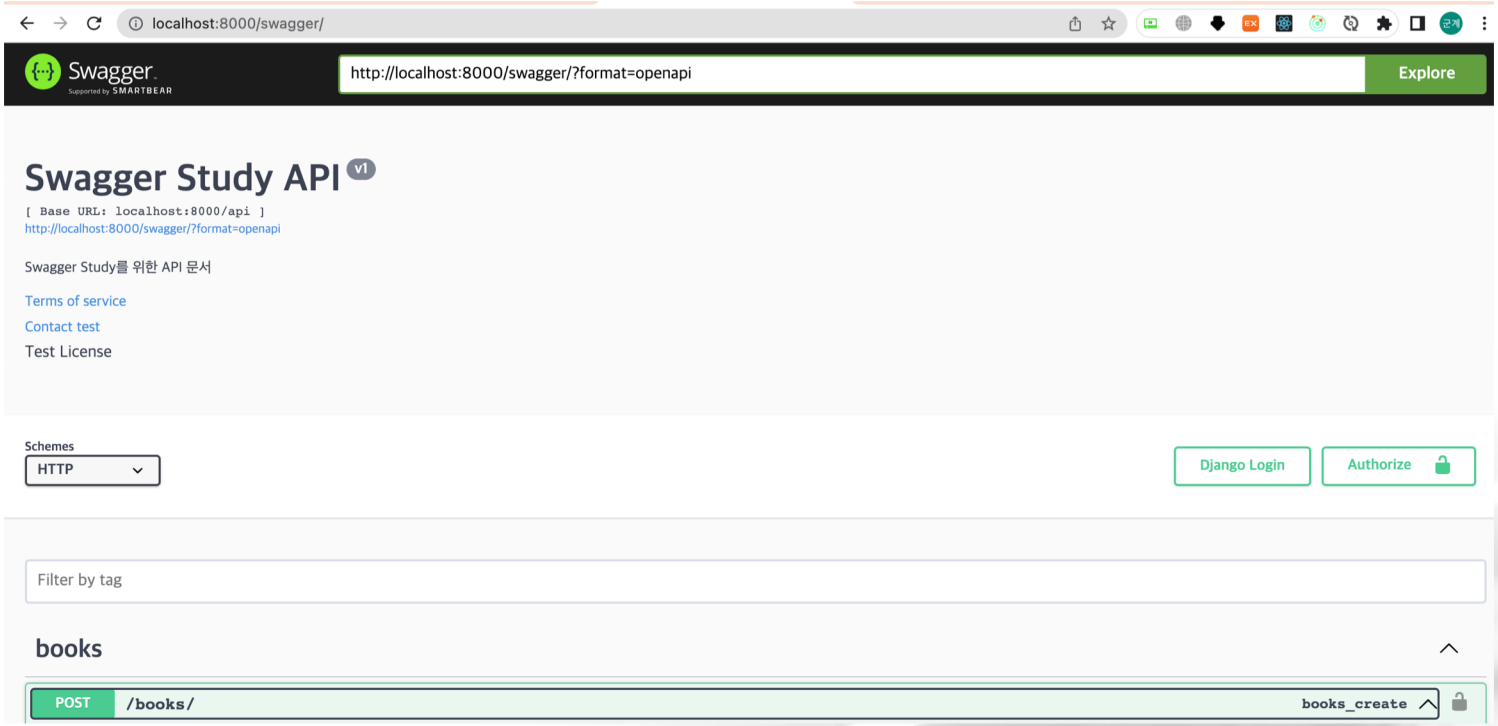
- drf_yasg 패키지 적용
  - ✓ 프로젝트의 url.py 수정

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('book.urls'))
]

if settings.DEBUG:
    urlpatterns += [
        re_path(r'^swagger(?P<format>₩.json|₩.yaml)$',
    schema_view.without_ui(cache_timeout=0), name="schema-json"),
        re_path(r'^swagger/$', schema_view.with_ui('swagger', cache_timeout=0),
    name='schema-swagger-ui'),
        re_path(r'^redoc/$', schema_view.with_ui('redoc', cache_timeout=0),
    name='schema-redoc'),     ]
```

문서화

- drf_yasg 패키지 적용
  - ✓ 프로젝트 실행 후 http://localhost:8000/swagger



문서화

- drf_yasg 패키지 적용
  - ✓ Model 의 각 필드에 도움말 추가: models.py

```
from django.db import models

class Book(models.Model):
    title = models.CharField(help_text="제목", max_length=128)
    author = models.CharField(help_text="저자", max_length=128)
```