PJT명	도서 데이터를 제공하는 RESTful API 서버 구축	
단계	[Django 개발]	
진행일자	2025.11.14	
예상 구현 시간	필수기능	5H
	추가기능	2H
	심화기능	1H

1. 목표

- Django REST Framework를 활용하여 API 서버를 제작할 수 있다.
- HTTP request methods에 대하여 이해한다.
- HTTP response status codes에 대하여 이해한다.
- Many to one relationship(N:1)에 대하여 이해한다.
- Many to many relationship(N:M)에 대하여 이해한다.

2. 준비사항

1) 제공 파일

- fixture의 형태로 제공되는 초기 데이터 (JSON 파일)
  - books.json, categories.json, threads.json, comments.json

2) 개발언어 및 툴

- Python 3.11
- Postman
- Visual Studio Code

3) 필수 라이브러리 / 오픈소스

- Django 5.2
- Django REST Framework

3. 작업 순서

1) 팀원과 같이 요구사항을 확인하고, GitLab에 프로젝트를 생성한다.
   - 프로젝트 이름은 07-pjt로 지정한다.
   - 각 반 담당 강사님을 Maintainer로 설정한다.
2) 팀원과 합의하여 협업 방식을 결정하고, 요구사항을 구현한다.
3) 작성한 코드들을 정리하고, README를 작성한다.
   - .gitignore 파일을 활용하여 불필요한 파일 및 폴더는 제출하지 않는다.
4) README 작성이 완료되면 심화 학습을 진행한다.
5) 제출 기한에 맞춰 모든 산출물이 GitLab에 업로드 될 수 있도록 한다.

4. 요구사항

AI 기반 도서 분석과 창작 지원 통합 솔루션 서비스를 구축하려고 한다. 다양한 장르와 형태의 도서 데이터를 수집 및 관리하고, 이를 기반으로 AI 기반 도서 분석 기능을 설계 및 구현한다. 또한 작가들에게 아이디어 발상 지원, 플롯 구성 제안, 문장 교정 및 스타일 개선 등 창작 전반에 걸친 다양한 AI 기반 창작 지원 기능을 제공한다. 또한 도서에 대한 사용자 리뷰 및 감상평 공유 커뮤니티 기능을 제공하여, 사용자들이 활발하게 소통하고 정보를 교환할 수 있는 기능을 제공한다. 사용자는 자신이 읽은 도서를 평가하고, 다른 사용자의 리뷰를 참고하여 다음 도서를 선택하는데 도움을 받을 수 있다. 나아가, 관심 도서 목록을 맞춤형으로 구성하는 등 다양한 편의 기능을 제공한다. 팀원과 상의하여 아래 요구사항을 만족할 수 있도록 요구 사항 명세서를 작성 및 구현해보자.

도서 관련 데이터를 제공하는 RESTful API 서버를 만들고자 한다. 도서, 카테고리, 게시글, 댓글 데이터의 조회가 가능하며, 사용자가 게시글을 남기고 관리할 수 있는 API를 제공하는 서버를 구현해보자.

- 요구사항 예시(참고용)
  - 아래의 내용을 참고하여 추가적인 아이디어에 대해 요구사항을 추가 또는 수정하여 기능을 구현한다. 단, **필수 기능은 구현해야 하며, 수정할 수 없다**.

번호	분류	요구사항명	요구사항 상세	우선순위
기능적 요구사항				
F01	프로젝트	프로젝트 구성	도서 커뮤니티 서비스의 API 서버 구현을 위한 Django 프로젝트 및 앱 생성	필수
F02	books model	Category 클래스	카테고리 데이터를 데이터베이스에 저장할 수 있도록 Django Model 클래스 구현	필수
F03	books model	Book 클래스	도서 데이터를 데이터베이스에 저장할 수 있도록 Django Model 클래스 구현	필수
F04	books model	Thread 클래스	게시글 데이터를 데이터베이스에 저장할 수 있도록 Django Model 클래스 구현	필수
F05	books model	Comment 클래스	게시글 댓글 데이터를 데이터베이스에 저장할 수 있도록 Django Model 클래스 구현	필수
F06	books serializers	Serializer 클래스	사용자의 입력 데이터 검증 및 응답 데이터 형식을 위한 Serializer 클래스를 개별 요구사항에 맞춰서 구현	필수
F07	books view	category_list	전체 카테고리 데이터를 조회하는 view 함수 구현	필수
F08	books view	book_list	전체 도서 데이터를 조회하는 view 함수 구현	필수
F09	books view	book_detail	단일 도서 데이터를 조회하는 view 함수 구현	필수
F10	books view	thread_list	전체 게시글 데이터를 조회하는 view 함수 구현	필수
F11	books view	thread_detail	단일 게시글 데이터를 조회, 수정, 삭제하는 view 함수 구현	필수
F12	books view	create_thread	전달받은 게시글 데이터를 데이터베이스에 저장하는 view 함수 구현	필수
F13	books view	create_comment	전달받은 게시글 댓글 데이터를 데이터베이스에 저장하는 view 함수 구현	필수

F14	books view	comment_detail	단일 게시글 댓글 데이터를 조회, 수정, 삭제하는 view 함수 구현	필수
F15	AI 활용	도서 검색	도서 제목과 작가를 대상으로 검색을 진행하는 기능 구현	도전
비기능적 요구사항				
NF01	프로젝트 관리	Git 활용	개발자 간 작업 충돌이 일어나지 않게끔 Git을 활용하여 프로젝트 관리	필수
NF02	설계	RESTful 원칙	RESTful 설계 원칙을 잘 따르도록 하여 사용성 증대	필수
NF03	보안	HTTP Method 허용	허용된 HTTP Method를 사용하는 요청만 허락하도록 구현	필수

1) 프로젝트 관리 (필수)

Git의 기능을 적극적으로 활용하여, 여러 개발자가 동시에 개발할 수 있는 환경을 조성한다.

- 요구사항 번호: NF01
- 팀원과 합의하여 branch 생성 원칙 정리하기
  - Gitflow, GitHub Flow 등을 참고하여 원칙을 정할 수 있음
  - branch 운영 중 문제상황이 발생할 경우 원인에 대한 분석을 꼭 진행할 것
- 팀원과 합의하여 commit 내역 기록 원칙 정리하기
  - 기능 개발, 버그 수정 등 그 목적이 명확히 드러날 수 있도록 commit 메시지를 작성할 것
  - 어느 시점에 commit을 남길지에 대하여 충분한 합의를 이룰 것
- 프로젝트 종료 시 초기에 세운 원칙이 잘 지켜졌는지 점검
  - README에 초기에 세운 원칙을 설명하여 정리할 것
  - 생성했던 branch 들에 대한 용도를 정리할 것
  - 작성된 commit들을 GitLab에서 확인하여 스크린샷으로 README에 포함시킬 것

2) 프로젝트 및 앱 (필수)

도서 데이터를 생성, 조회, 수정, 삭제할 수 있는 API를 제공하는 Django 프로젝트를 만든다. 명시된 요구사항 이외 서비스를 위해 필요하다고 생각되는 기능 등은 자유롭게 구현해도 무관하며, Bootstrap을 활용하여 자유롭게 스타일링 한다.

- 요구사항 번호: F01
- 프로젝트 이름: mypjt
- 앱 이름: books

A. books

도서 데이터를 관리하기 위한 앱이다. 요구사항데로 Model 클래스를 정의해보자.

- 요구사항 번호: F02, F03, F04, F05, F06, NF02, NF03
- Category 클래스
  - 카테고리 이름을 지정할 필드(name) 1개 지정
- Book 클래스
  - 도서 제목, 설명, isbn, 커버 이미지, 출판사, 출간일, 저자명, 회원 리뷰 평점(title, description, isbn, cover, publisher, pub_date, author, customer_review_rank)을 지정할 필드 8개 지정
  - Category 클래스와 N:1 관계를 맺고 있는, 도서의 카테고리를 나타내는 필드(category) 지정
- Thread 클래스
  - 게시글 제목, 본문, 독서일, 생성 시간, 수정 시간(title, content, reading_date, created_at, updated_at)을 저장할 필드 5개 지정
  - Book 클래스와 N:1 관계를 맺고 있는, 게시글이 작성된 대상 도서를 나타내는 필드(book) 지정
- Comment 클래스
  - 댓글 내용, 생성 시간, 수정 시간(content, created_at, updated_at)을 저장할 필드 3개 지정
  - Thread 클래스와 N:1 관계를 맺고 있는, 댓글이 작성된 게시글을 나타내는 필드(thread) 지정
- Model 클래스를 만든 후 제공된 fixture로 초기 데이터 불러오기
  - 각 Model 클래스의 관계에 주의할 것
- 기능 개발 시 유효성 검증을 위한 Serializer를 요구사항 구현에 필요한 형태로 작성하여 사용
- RESTful 설계 원칙에 알맞게 URL을 설계할 것

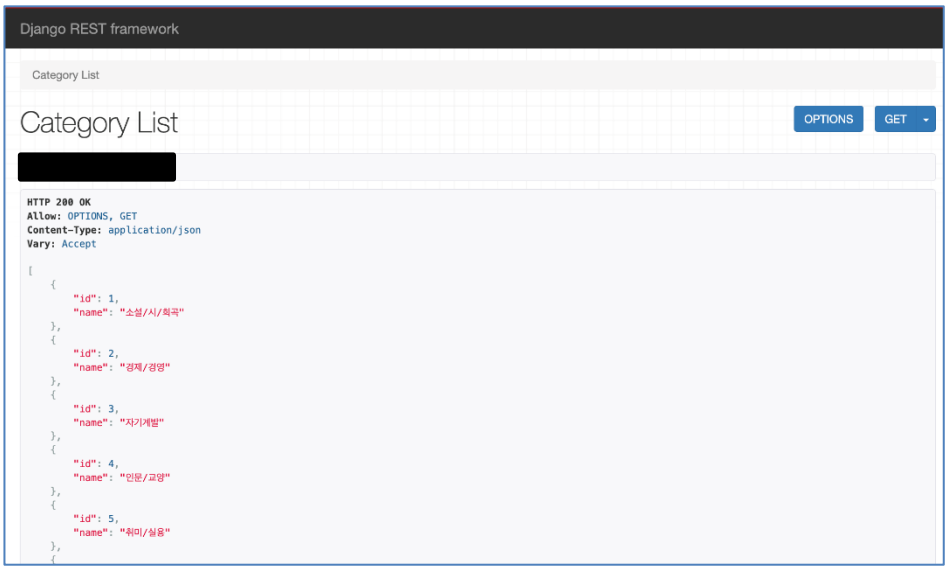
3) books 앱 view 함수

도서 서비스와 연관된 다양한 데이터를 생성, 조회, 수정, 삭제하는 기능을 제공하는 books 앱을 구현하자.

A. category_list

전체 도서 카테고리 목록을 제공해주는 category_list view 함수를 구현한다.

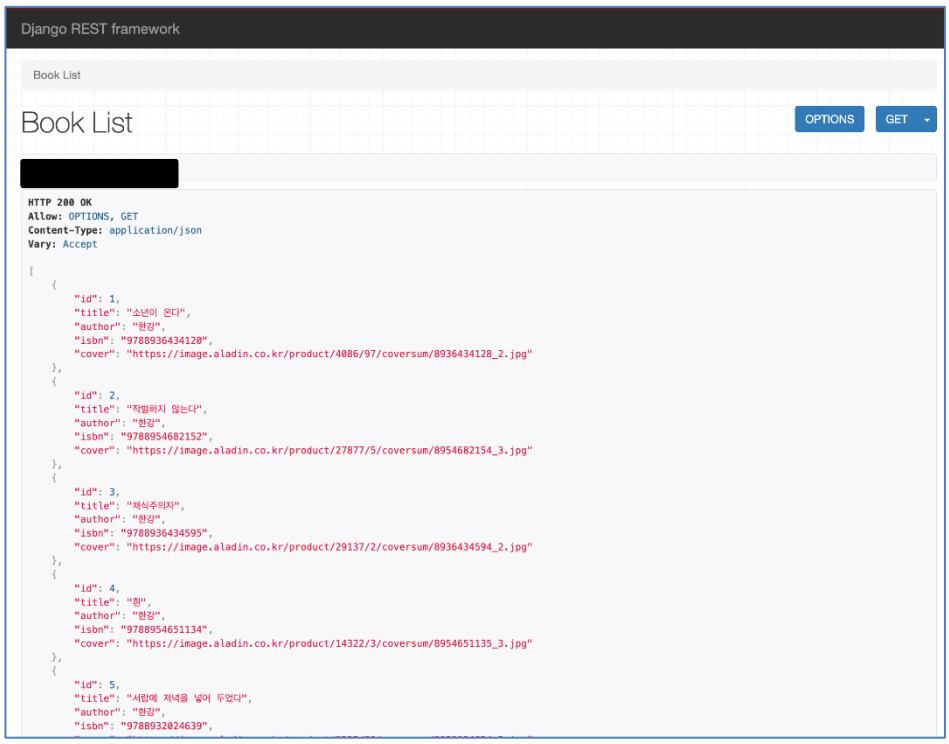
- 요구사항 번호: F07
- 각 카테고리의 id, 이름 데이터가 조회되어야 함
- URL은 RESTful 설계 원칙을 고려하여 작성
- GET method에 대해서만 동작해야 함
  - 브라우저에서 요청했을 경우 HTML을 Django REST Framework가 제공하는 반환
  - Postman같은 API 클라이언트에서 요청했을 경우 JSON을 반환
- 브라우저 접근 출력 예시



B. book_list

전체 도서 목록을 제공해주는 book_list view 함수를 구현한다.

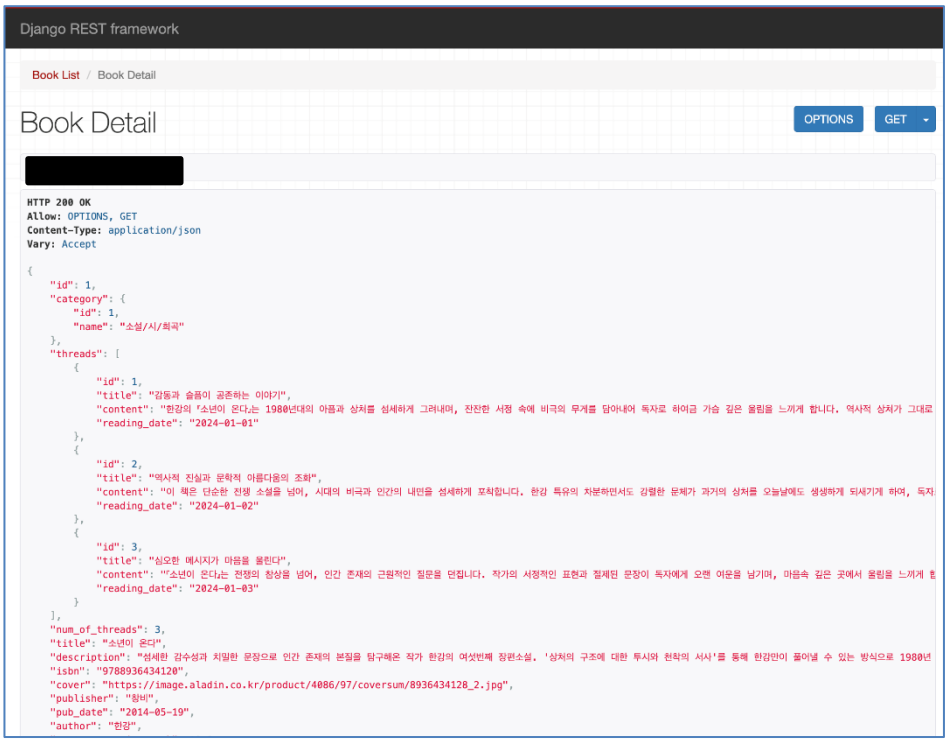
- 요구사항 번호: F08
- 각 도서의 id, 제목, 작가, isbn, 커버 이미지 데이터가 조회되어야 함.
- URL은 RESTful 설계 원칙을 고려하여 작성
- GET method에 대해서만 동작해야 함
  - 브라우저에서 요청했을 경우 HTML을 Django REST Framework가 제공하는 반환
  - Postman같은 API 클라이언트에서 요청했을 경우 JSON을 반환
- 브라우저 접근 출력 예시



C. book_detail

단일 도서 데이터를 제공해주는 book_detail view 함수를 구현한다.

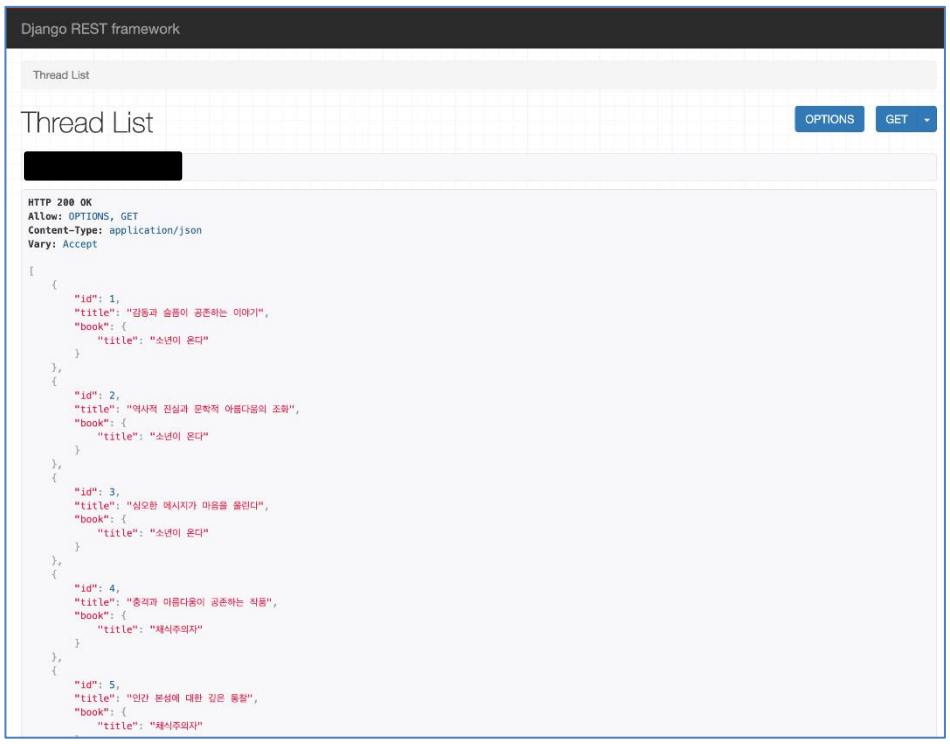
- 요구사항 번호: F09
- 대상 도서의 데이터와 함께, 도서의 카테고리, 게시글 목록, 총 게시글 개수 정보가 함께 조회되어야 함
  - 조회되는 게시글의 경우 id, 제목, 내용, 독서일 데이터가 조회되어야 함
- URL은 RESTful 설계 원칙을 고려하여 작성
- GET method에 대해서만 동작해야 함
  - 브라우저에서 요청했을 경우 HTML을 Django REST Framework가 제공하는 반환
  - Postman같은 API 클라이언트에서 요청했을 경우 JSON을 반환
- 브라우저 접근 출력 예시



D. thread_list

전체 게시글 목록을 제공해주는 thread_list view 함수를 구현한다.

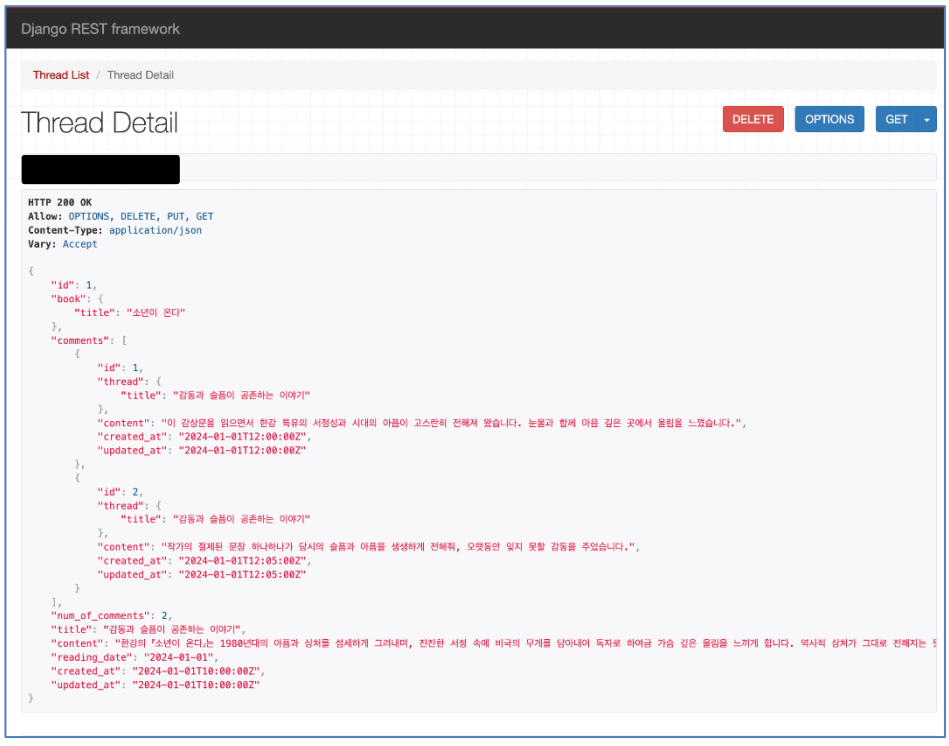
- 요구사항 번호: F10
- 각 게시글의 id, 제목과 함께, 대상 도서의 제목이 함께 조회되어야 함
- URL은 RESTful 설계 원칙을 고려하여 작성
- GET method에 대해서만 동작해야 함
  - 브라우저에서 요청했을 경우 HTML을 Django REST Framework가 제공하는 반환
  - Postman같은 API 클라이언트에서 요청했을 경우 JSON을 반환
- 브라우저 접근 출력 예시



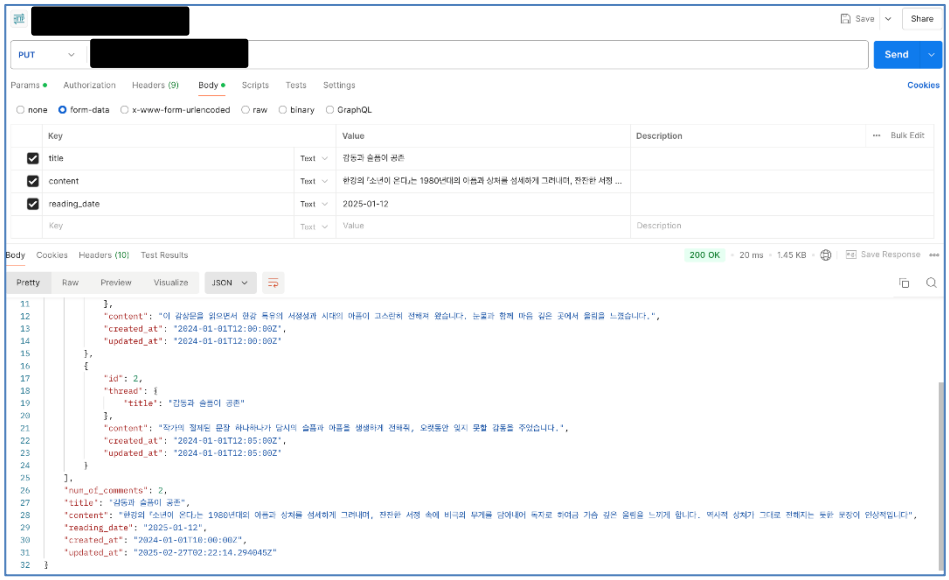
E. thread_detail

단일 게시글 정보를 request method에 따라 조회, 수정, 삭제해주는 thread_detail view 함수를 구현한다.

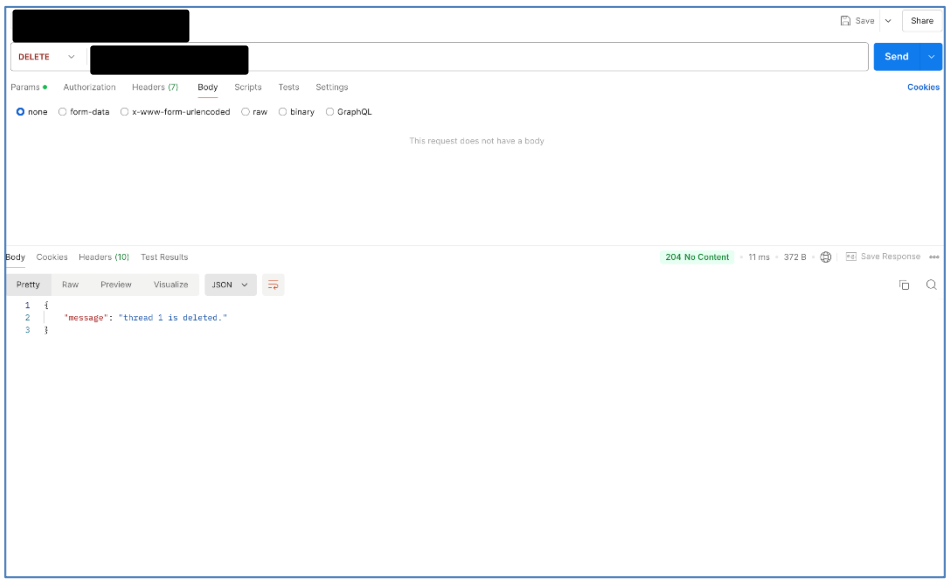
- 요구사항 번호: F11
- URL은 RESTful 설계 원칙을 고려하여 작성
- GET method로 요청했을 경우
  - 해당하는 게시글 데이터와 함께, 대상 도서, 댓글 목록, 댓글의 개수 정보가 조회되어야 함
  - 브라우저에서 요청했을 경우 Django REST Framework가 제공하는 HTML을 반환
  - Postman같은 API 클라이언트에서 요청했을 경우 JSON을 반환
- PUT method로 요청했을 경우
  - 유효한 데이터인 경우 대상 게시글 수정
- DELETE method로 요청했을 경우
  - 대상 게시글 삭제
- 브라우저 접근 출력 예시



- Postman 요청 예시
  - PUT method



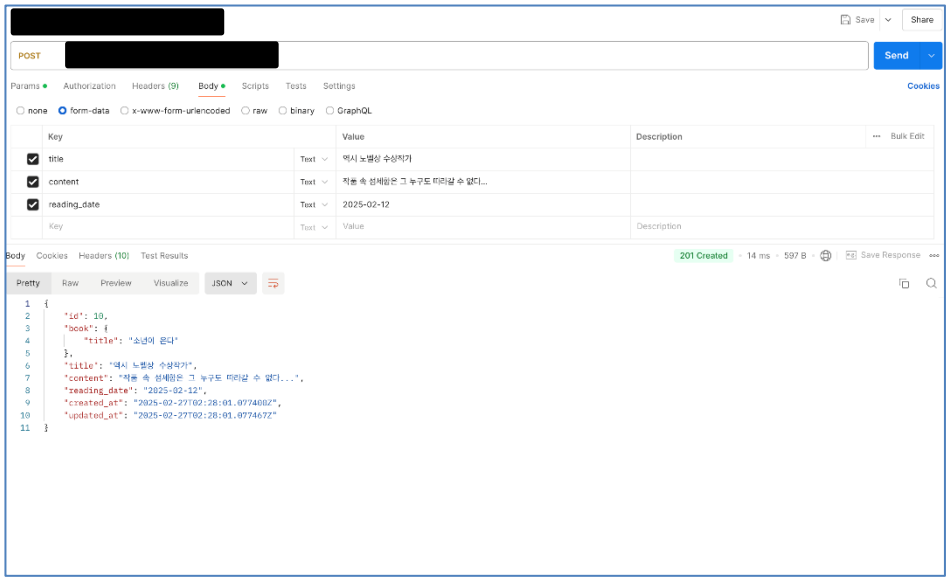
  - DELETE method



F. create_thread

게시글 데이터를 전달받아서 저장하는 create_thread view 함수를 구현한다.

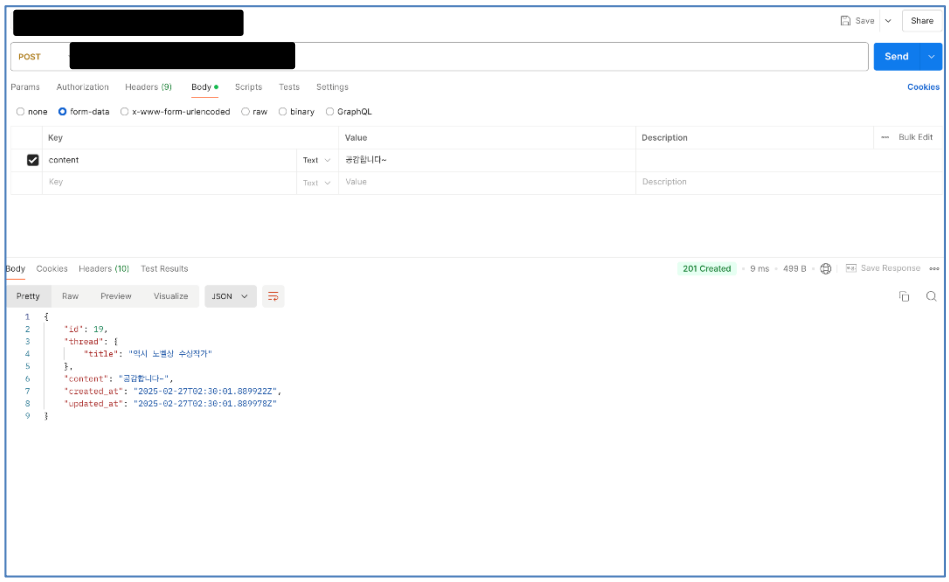
- 요구사항 번호: F12
- URL은 RESTful 설계 원칙을 고려하여 작성
- POST method에 대해서만 정상 도작해야 함
  - 제목, 내용, 독서일 데이터를 전달받아 새로운 리뷰 생성
  - 유효한 데이터인 경우에만 정상 작동
- 그 외의 method의 경우 405 응답을 반환해야 함
  - 브라우저에서 접근할 경우 데이터를 입력할 수 있는 UI를 표시할 수 있음
- Postman 요청 예시



G. create_comment

댓글 데이터를 전달받아서 저장하는 create_comment view 함수를 구현한다.

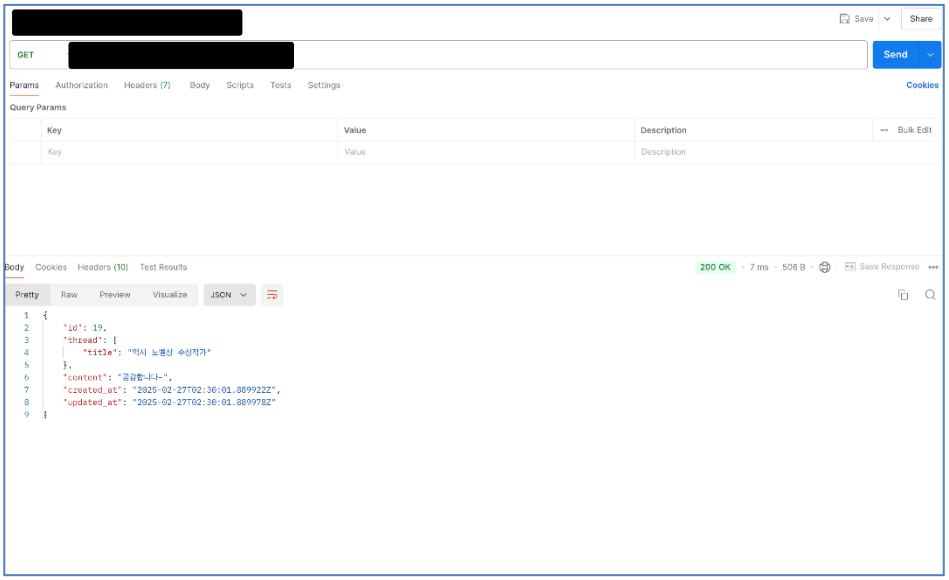
- 요구사항 번호: F13
- URL은 RESTful 설계 원칙을 고려하여 작성
- POST method에 대해서만 정상 동작해야 함
  - 내용 데이터를 전달받아 새로운 댓글 생성
  - 대상이 되는 게시글은 RESTful 설계 원칙을 고려하여 판별
  - 유효한 데이터인 경우에만 정상 작동
- 그 외의 method의 경우 405 응답을 반환해야 함
  - 브라우저에서 접근할 경우 데이터를 입력할 수 있는 UI를 표시할 수 있음
- Postman 요청 예시



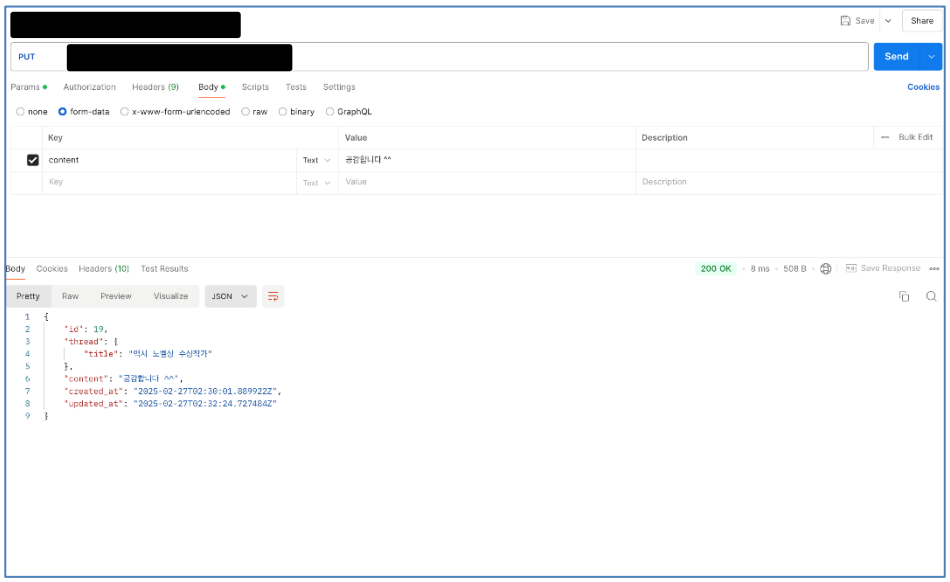
H. comment_detail

단일 댓글 정보를 request method에 따라 조회, 수정, 삭제해주는 comment_detail view 함수를 구현한다.

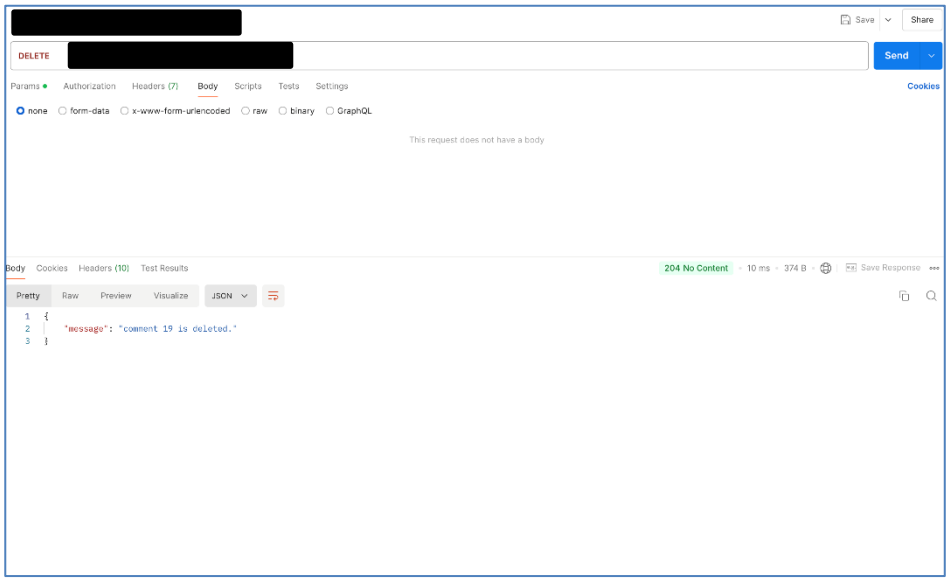
- 요구사항 번호: F14
- URL은 RESTful 설계 원칙을 고려하여 작성
- GET method로 요청했을 경우
  - 해당하는 댓글 데이터와 함께, 대상 게시글의 제목이 함께 조회되어야 함
- PUT method로 요청했을 경우
  - 유효한 데이터인 경우 대상 게시글 수정
- DELETE method로 요청했을 경우
  - 대상 게시글 삭제
- Postman 요청 예시
  - GET method



- PUT method



- DELETE method



4) 도전 과제

생성형 AI 도구를 활용하여 도전과제 요구사항을 해결해보자.

- 코드 생성, 아이디어 구상, 문제 해결 방법 탐색 등 다양한 방식으로 활용할 수 있다.
- 사용할 생성형 AI 서비스는 자유롭게 선택한다.
  - 제공되는 GPT API Key를 활용할 경우, 기능별로 정해진 모델을 활용한다.
- 최종 결과물은 AI 생성 내용을 바탕으로 직접 수정 및 개선하여 적용해본다.

보조 수단으로 활용하되, 능동적인 자세로 학습에 임할 것.

- 최종적인 이해와 적용은 자기 주도적 학습을 통해 이루어지며, 배운 내용을 스스로 기록하고 정리하며 학습 효과를 높일 것.

A. 도서 검색 기능 구현

찾고 싶은 도서 정보를 검색할 수 있는 기능을 구현해보자.

- 요구사항 번호: F15
- 도서 제목과 작가 필드를 대상으로 검색 진행
- 검색을 위한 추가 URL과 view 함수를 작성
  - URL의 각 요소의 목적을 이해하고 활용할 것
- 검색어가 없을 경우 적절한 메시지를 응답할 것
- Postman 응답 예시
  - '데미안'을 검색했을 때 응답

  - '한강'을 검색했을 때 응답

  - ''을 검색했을 때 응답

5. 참고자료

- Django 5.2

  https://www.djangoproject.com/start/overview/
- Django REST Framework

  https://www.django-rest-framework.org/

6. 결과

제출 기한은 진행일 18시까지이므로 제출 기한을 지킬 수 있도록 한다. 제출은 GitLab을 통해서 이뤄진다.

- **산출물과 제출**
  - 단계별로 구현 과정 중 학습한 내용, 어려웠던 부분, 새로 배운 것들 및 느낀 점을 상세히 기록한 README.md
  - 완성된 각 문제 별 소스코드 및 실행 화면 캡쳐본
  - 프로젝트 이름은 07-pjt로 지정 및 각자 제출
    - 한 명의 GitLab 계정에 Git 저장소 Push 및 작업
    - 나머지 인원은 제출 시 해당 저장소를 Fork 하여 제출
  - 각 반 담당 강사님을 Maintainer로 설정

- 끝 -