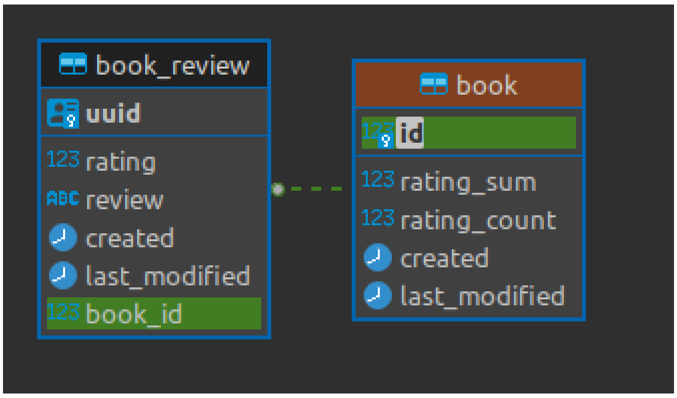
Thought Process

To meet these challenge requirements, I created a Django application with the specified endpoints.

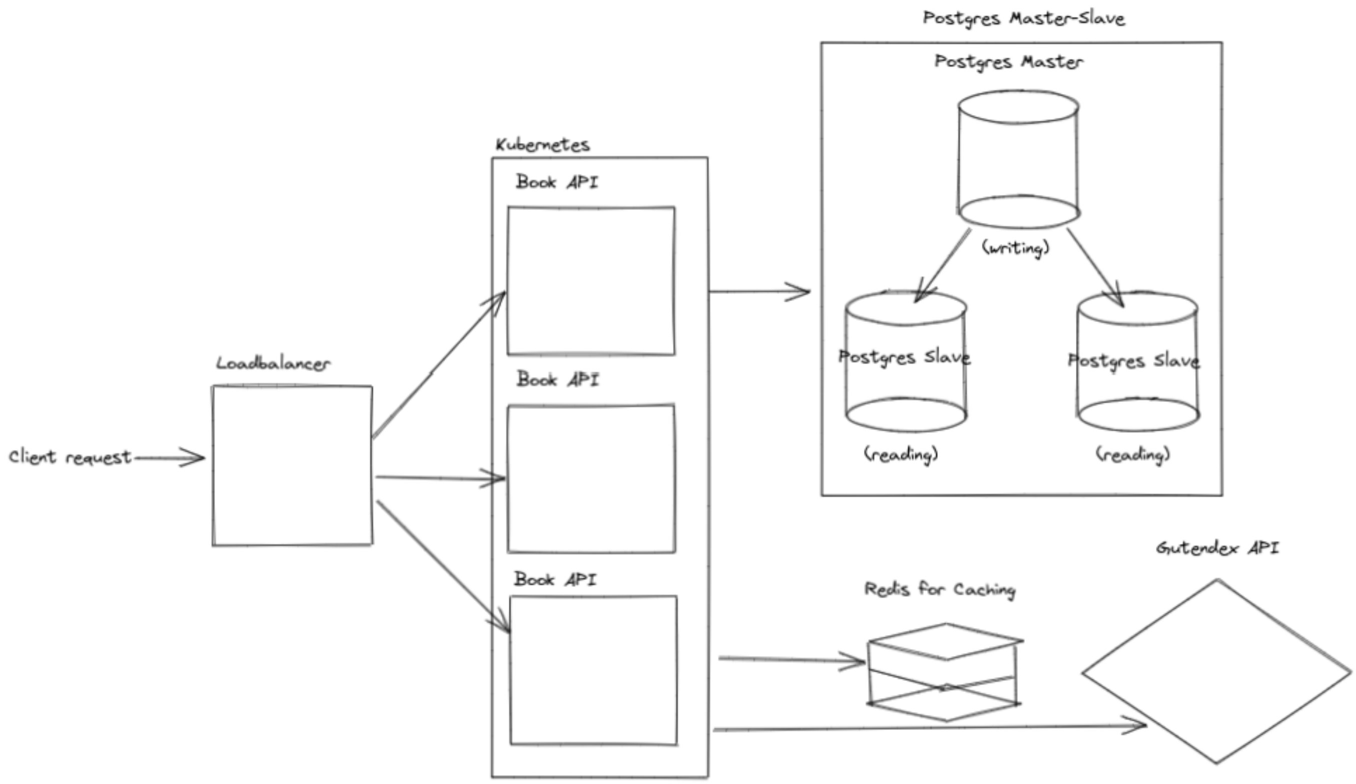
- Part 1: Searching for books
  - This part was very simple. I just requested the data from the gutendex API and “filtered” it with a serializer to return just the desired data.

- Part 2: Reviewing a book
  - For this part, I created two models, one to store the review data (book_id, rating, review) and the other one to store rating_sum and rating_count.
  - The book table is updated every time that a book review is posted, in order to keep rating_sum and rating_count values up to date, which are important to meet part 3 requirements.



- Part 3: Getting details of a specific book
  - To avoid having to go through the entire book_review table summing each rating value and dividing by the total number of rows in order to get the average rating value, the book table already provides rating_sum and rating_count values, so the average rating is obtained by dividing rating_sum by rating_count.
  - Review values are obtained by selecting in book_review table, passing book_id as a filter
  - The other details are requested from gutendex API

High-level System Diagram


Client request
Loadbalancer
Kubernetes
Book API
Book API
Book API
Postgres Master-Slave
Postgres Master
(writing)
Postgres Slave
(reading)
Postgres Slave
(reading)
Redis for Caching
Gutendex API