### Application requirements

Our goal is to build a simple solution exposing an API that allows an external entity (i.e., a browser client, a mobile app, manual api calls…) to execute a few actions against a stored movie database. The detailed responses and http codes of the api you’ll be implementing are defined below.

We want to expose information about the movies our app knows about to consumers of our apis, and allow them to query that data in multiple ways:

* + Api **A** should return the details of movies that pass certain filter criteria provided by the api consumers.
    - At least one filter criteria should be provided by the caller, else the api should return an error.
    - The criteria filters can be one or more of the following:
    - Title or partial title of the movie.
    - Year of release, Genre(s), etc.

Api **B** should return the details of the top 5 movies based on total user average ratings.

* + - In case of a rating draw, (e.g., 2 movies have 3.768 average rating) return them by ascending title alphabetical order.
  + Api **C** should return the details of the top 5 movies based on the highest ratings given by a specific user, provided by the api consumer.
    - In case of a rating draw (e.g., user scored 5 for 2 movies) return them by ascending title alphabetical order.

1. We want to allow api consumers to add a rating to a movie for a certain user (api **D**):
   * The rating must be an integer between 1 and 5;
   * If the user already had a rating for that movie, the old rating should be updated to the new value
2. When returning to an api consumer the average rating associated with a movie, you should always round the number to the closest 0.5:
   * An average rating of 2.91 should be displayed as 3.0
   * 3.249 should be displayed as 3.0
   * 3.25 should be displayed as 3.5
   * 3.6 should be displayed as 3.5
   * 3.75 should be displayed as 4.0.

### API Specification

Below you can find the summary of what each api you should implement do, and what you should respond given different scenarios. Routing, parameters and verbs for the api request are left at your discretion but should roughly follow the REST standards, with detailed description of when you had to clearly diverge from that.

|  |  |  |
| --- | --- | --- |
| **Id** | **Description** | **Response** |
| **A** | Query movie data based on provided filter criteria: title, year of release, genre(s) | * **404** (if no movie is found based on the criteria) * **400** (if invalid / no criteria is given) * **200** (OK)   It is required to return the below schema for each movie, running time should be in minutes and average rating should always be the average based on total existing ratings for the movie: [ {  id,  title,  yearOfRelease,  runningTime, averageRating  }, … ] |
| **B** | Query top 5 movies based on total user rating | Same as **A** |
| **C** | Query top 5 movies based on a certain user’s rating | Same as **A** |
| **D** | Add or update user rating for a movie | * **404** (if movie or user is not found) * **400** (if rating is an invalid value) * **200** (OK) |