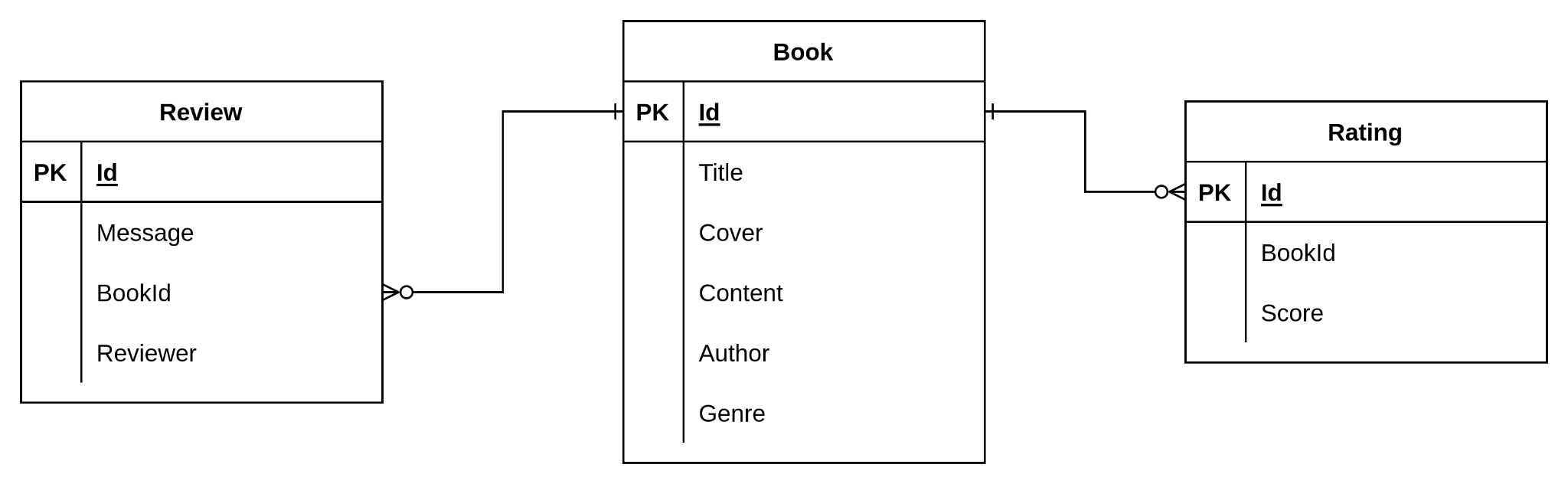
# Hometask #2 | ASP.NET Core | Library



Pic. 1 - Data Schema for Library

Your task is to create an [ASP.NET Core WebAPI](https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-6.0&tabs=visual-studio) application and design a model for the library. In this task you can use [layered](https://www.geeksforgeeks.org/what-is-net-3-tier-architecture/) architecture to separate different parts of the app. Keep it simple but cover all requirements. Here is a list of steps you need to go through:

1. Connect Entity Framework Core and use in-memory database. Create classes and relationships for all tables from pic. 1. Make sure to register them in EntityFramework.
2. Implement the following API:

@baseUrl = localhost:5000

**### 1. Get all books. Order by provided value (title or author)**

GET https://{{baseUrl}}/api/books?order=author

# Response

# [{

# "id": "number",

# "title": "string",

# "author": "string",

# "rating": "decimal", average rating

# "reviewsNumber": "decimal" count of reviews

# }]

**### 2. Get top 10 books with high rating and number of reviews greater than 10. You can filter books by specifying genre. Order by rating**

GET https://{{baseUrl}}/api/recommended?genre=horror

# Response

# [{

# "id": "number",

# "title": "string",

# "author": "string",

# "rating": "decimal", average rating

# "reviewsNumber": "decimal" count of reviews

# }]

**### 3. Get book details with the list of reviews**

GET https://{{baseUrl}}/api/books/{id}

# Response

# {

# "id": "number",

# "title": "string",

# "author": "string",

# "cover": "string",

# "content": "string",

# "rating": "decimal", average rating

# "reviews": [{

# "id": "number",

# "message": "string",

# "reviewer": "string",

# }]

# }}

**### 4. Delete a book using a secret key. Save the secret key in the config of your application. Compare this key with query param**

DELETE https://{{baseUrl}}/api/books/{id}?secret=qwerty

**### 5. Save a new book.**

POST https://{{baseUrl}}/api/books/save

{

"id": "number", // if id is not provided create a new book, otherwise - update an existing one

"title": "string",

"cover": "string", // save image as base64

"content": "string",

"genre": "string",

"author": "string"

}

# Response

# {

# "id": "number"

# }

**### 6. Save a review for the book.**

PUT https://{{baseUrl}}/api/books/{id}/review

{

"message": "string",

"reviewer": "string",

}

# Response

# {

# "id": "number"

# }

**### 7. Rate a book**

PUT https://{{baseUrl}}/api/books/{id}/rate

{

"score": "number", // score can be from 1 to 5

}

1. Validate all values that come into your app (query params, body, etc)
2. Return an appropriate [status code](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status) for every response.
3. Connect [middlewares](https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?view=aspnetcore-6.0) for error handling and logging. Log every request in a simple format (make sure to log HTTP method, headers, query params and body) in the console (you can also use other outputs).

## Notes:

* Do not create instances (DbContext, services, mappers) manually via “new()”. Use [Dependency Injection](https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-6.0) instead.
* You don’t need to create migrations. Instead you may add a data seeding. For better testing seed data for 10 books.
* Use [DTO](https://docs.microsoft.com/en-us/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5) for all requests and responses. Do not put entities in the response. You can use [Automapper](https://automapper.org/) to simplify the mapping process.
* In order to implement validation you can use the [Fluent Validation](https://docs.fluentvalidation.net/en/latest/) package. Feel free to come up with your own rules.
* Use async/await wherever possible (Controllers should return Task<TResult>)
* To test the endpoints you can use either [Postman](https://www.postman.com/) or this [extension](https://marketplace.visualstudio.com/items?itemName=humao.rest-client) for Visual Studio Code. The format of the requests above is supported by this extension.