The system: A library

Functional Requirements:

The system must:

- REST API for interacting with the library system.
- Provide an opportunity to add books specifying name, publish date, authors category.
- Provide an opportunity to add authors specifying name, birthdate.
- Provide an opportunity to add books categories specifying name, description.
- Provide an opportunity to change books specifying name, publish date, authors category.
- Provide an opportunity to change authors specifying name, birthdate.
- Provide an opportunity to change books categories specifying name, description.
- Provide an opportunity to list books with pagination (20 entities per page) with the possible filter by category or get the specific one.
- Provide an opportunity to list authors with pagination (20 entities per page) or get the specific one.
- Provide an opportunity to list books with pagination (20 entities per page) categories or get the specific one.
- Provide an opportunity to delete books.
- Provide an opportunity to delete authors.
- Provide an opportunity to delete books categories.
- Provide a method to get an Auth token.

Non-Functional Requirements:

Usability Requirements:

The system should:

- Be RESTful with The Richardson Maturity applied on HATEOAS level.
- Use HTTP2 as a communication protocol.
- Be presented as an REST API with all methods URIs starting with “/api”.
- Provide any functionality only to authorized users.

Reliability Requirements:

The system should:

- Be available 24/7.
- Have a failure rate of no more than 1 in 100 attempts.

Performance Requirements:

The system should:

- Process all queries in not more than 50 ms.

- Caching should be provided where possible.

Supportability Requirements:

The system should:

- Be presented "as is" with no variations or customization options.

Entities:

Books, authors, categories.

Each book may have several authors (must have at least one), several categories

Operations:

For every entity there should be CRUD operations: GET, POST, PUT, PATCH, DELETE in terms of API

---

1. Books

Create (Add a Book)

- **Endpoint:** POST api/books
- **Response**: Success: **201 Created** with the newly created book resource.

List All Books:

- **Endpoint**: GET /api/books?page=<pageNum>&category=<categoryName>
- **Response**: Success: **200 Ok.** Returns a list of all books with pagination support (20 books per page) filtered by categoryName.

Get Specific Book:

- **Endpoint**: GET /api/books/{bookId}
- **Response**: Success: **200 Ok.** Returns details of a specific book, including associated authors and category.

Update Specific Book:

- **Endpoint**: PUT /api/books/{bookId}
- **Response**: Success: **200 Ok.** Updated book resource.

Update Specific Book (Partial):

- **Endpoint**: PATCH /api/books/{bookId}
- **Response**: Success: **200 Ok.** Updated book resource

Delete (Remove a Book)

- Endpoint: DELETE /api/books/{bookId}
- Response: Success: **204 No Content**.

.2. Authors

Create (Add a Author)

- **Endpoint:** POST api/authors
- **Response**: Success: **201 Created** with the newly created author resource.

List All Authors:

- **Endpoint**: GET /api/authors?page=<pageNum>
- **Response**: Success: **200 Ok.** Returns a list of all authors with pagination support (20 authors per page) filtered by categoryName.

Get Specific Author:

- **Endpoint**: GET /api/authors/{authorId}
- **Response**: Success: **200 Ok.** Returns details of a specific author.

Update Specific Author:

- **Endpoint**: PUT /api/authors/{authorId}
- **Response**: Success: **200 Ok.** Updated author resource.

Update Specific Author(Partial):

- **Endpoint**: PATCH /api/authors/{authorId}
- **Response**: Success: **200 Ok.** Updated author resource

Delete (Remove a Author)

- Endpoint: DELETE /api/authors/{authorId}
- Response: Success: **204 No Content**.

.3. Categories

Create (Add a Category)

- **Endpoint:** POST api/categories
- **Response**: Success: **201 Created** with the newly created category resource.

List All Categories:

- **Endpoint**: GET /api/categories?page=<pageNum>
- **Response**: Success: **200 Ok.** Returns a list of all categories with pagination support (20 categories per page) filtered by categoryName.

Get Specific Category:

- **Endpoint**: GET /api/categories/{categoryId}
- **Response**: Success: **200 Ok.** Returns details of a specific category.

Update Specific Category:

- **Endpoint**: PUT /api/categories/{categoryId}
- **Response**: Success: **200 Ok.** Updated category resource.

Update Specific Category(Partial):

- **Endpoint**: PATCH /api/categories/{categoryId}
- **Response**: Success: **200 Ok.** Updated category resource

Delete (Remove a Category)

- Endpoint: DELETE /api/categories/{categoryId}
- Response: Success: **204 No Content**.

In case if unauthorized user tries to perform any action he gets **401 Unauthorized**

.**User Login**:

- Endpoint: POST /api/auth/login

He should pass username and password. In response he gets Success: **200 Ok** and a **token** to make requests with it.

Caching:

GET /api/books	GET	5 minutes	Use Redis for in-memory caching. Include ETag and Cache-Control headers.

GET /api/books/{bookId}	GET	5 minutes	Use Redis for caching the specific book. Include ETag and Cache-Control headers.

GET /api/authors	GET	10 minutes	Use Redis for caching all authors. Include ETag and Cache-Control headers.
GET /api/authors/{authorId}	GET	10 minutes	Use Redis for caching specific author data. Include ETag and Cache-Control headers.
GET /api/categories	GET	10 minutes	Use Redis for caching all categories. Include ETag and Cache-Control headers.
GET /api/categories/{categoryId}	GET	10 minutes	Use Redis for caching specific category data. Include ETag and Cache-Control headers.