Library Catalogue

Example for a Library Catalogue REST API.

The API would count with the following features: 1. Authentication:The API requires users to authenticate before they can reserve a book from the library. Users can register and log in to the system and admins need it so they can administer the system information. 2. Books: Users can get information about all books and specific books from the system. Admins can add, update, or delete books. 3. Authors: Users can get information about all authors and specific authors from the system. Admins can add, update, and delete authors. 4. Categories: Users can get information about all categories and specific categories. Admins can add, update, and delete categories. 5. Reservations: Users can get a list of all book reservations made by them. Users can reserve a specific book as long as there are enough copies available.

Requirements

Functional Requirements

Books Collection

- Retrieve a list of all books.
- Retrieve a specific book by its id.
- Add new books.
- Update books in the catalog.
- Delete books from the catalog.

Authors Collection

- Retrieve a list of all authors.
- Retrieve a specific author by their id.
- Add a new author.
- Update an existing author.
- Delete an author.

Categories Collection

- Retrieve a list of all categories.
- Retrieve a specific category by its id.
- Add a new category.
- Update an existing category.
- Delete a category.

Users Collection

- Retrieve a list of all users.

- Retrieve a specific user by their id.
- Add a new user.
- Update an existing user.
- Delete a user.

Reservations Collection

- Retrieve a list of all reservations.
- Retrieve a specific reservation by its id.
- Add a new reservation.

Non-Functional Requirements

- The system must be secure and require authentication for any changes to the catalog or reservations.
- The system should be designed with error handling in mind, providing clear error messages.
- The system should be designed to handle a large number of requests.
- The system should be documented clearly and comprehensively.
- The system should be designed with scalability in mind.

Entities

- **Books**: Represents the books in the library. It has attributes like **Title**, **Category**, **Author**, **Publication Date**, **language**, and **number of copies** available at the library.
- **Authors**: Represents the authors of the books in the library. It includes attributes like **name**, **primary language**, and **biography**.
- **Categories**: Represents the categories or genres of the books in the library. It includes attributes like **name** and **description**.
- **Users**: Represents the users of the library system. It includes attributes such as **username**, **email**, **password**, and **role**.
- **Reservation**: Represents the book reservations made by the users.

Operations:

Method	Endpoint	Description	Possible Status Codes
POST	/register	For user registration	**201** (User Created), **400** (Bad Request)
POST	/login	For users to log in to the system. A JWT will be returned and be alive for 1 year	**200** (OK), **400**(Bad Request)

Method	Endpoint	Description	Possible Status Codes
GET	/books	Returns a list of all books and their respective information	**200** (OK), **400** (Bad Request) **500** (Server Error)
POST	/books	Used to add new books to the catalogue. This is accessible only by users with the **admin** role	**201** (Created), **400** (Bad Request), **401** (Unauthorized, only admins may create), **403** (Forbidden), **500** (Internal Server Error)
GET	/books/{bookId}	Returns the information of one book by the given id	**200** (OK), **400** (Bad Request), **404**(Not Found), **500** (Internal Server Error)
PUT	/books/{bookId}	Updates the information of one book by the given id. This is accessible only by users with the **admin** role	**200**(OK), **400** (Bad Request), **401** (Unauthorized, only admins may do this), **403** (Forbidden), **404**(Not Found), **500**(Internal Server Error)
DELETE	/books/{bookId}	Deletes one book from the system by the given id. This is accessible only by users with the **admin** role	**204** (No Content - Successful Delete), **400** (Bad Request), **401** (Unauthorized, only admins may do this), **403** (Forbidden), **404** (Not Found), **500** (Internal Server Error)
GET	/authors	Returns a list of all authors in the system	**200** (OK), **400** (Bad Request), **500** (Server Error)
POST	/authors	Creates a new author. This is accessible only by users with the **admin** role	**201** (Created), **400** (Bad Request), **401** (Unauthorized, only admins may create), **403** (Forbidden), **500** (Internal Server Error)
GET	/authors/{authorId}	Returns information about one author by the given id	**200** (OK), **400** (Bad Request), **404**(Not Found), **500** (Internal Server Error)

Method	Endpoint	Description	Possible Status Codes
PUT	/authors/{authorId}	Updates the information about an existing author by the given id, This is accessible only by users with the **admin** role	**200**(OK), **400** (Bad Request), **401** (Unauthorized, only admins may do this), **403** (Forbidden), **404**(Not Found), **500**(Internal Server Error)
DELETE	/authors/{authorId}	Deletes an author by the given id. This is accessible only by users with the **admin** role	**204** (No Content - Successful Delete), **400** (Bad Request), **401** (Unauthorized, only admins may do this), **403** (Forbidden), **404** (Not Found), **500** (Internal Server Error)
GET	/categories	Returns a list of all categories	**200** (OK), **400** (Bad Request), **500** (Server Error)
POST	/categories	Creates a new category. This is accessible only by users with the **admin** role	**201** (Created), **400** (Bad Request), **401** (Unauthorized, only admins may create), **403** (Forbidden), **500** (Internal Server Error)
GET	/categories/{categoryId}	Returns information about a specific category by the given id	**200** (OK), **400** (Bad Request), **404**(Not Found), **500** (Internal Server Error)
PUT	/categories/{categoryId}	Updates information about a category by the given id. This is accessible only by users with the **admin** role	**200**(OK), **400** (Bad Request), **401** (Unauthorized, only admins may do this), **403** (Forbidden), **404**(Not Found), **500**(Internal Server Error)

Method	Endpoint	Description	Possible Status Codes
DELETE	/categories/{categoryId}	Deletes a specific category by the given id. This is accessible only by users with the **admin** role	**204** (No Content - Successful Delete), **400** (Bad Request), **401** (Unauthorized, only admins may do this), **403** (Forbidden), **404** (Not Found), **500** (Internal Server Error)
GET	/users/{userId}/reservations	Returns a list of all book reservations made by a specific user by the given id. This is accessible only by users with the **admin** role AND the used with that very same id	**200** (OK), **400** (Bad Request), **404**(Not Found), **500** (Server Error), **401** (Unauthorized, only admins may look for this and the specific user)
POST	/users/{userId}/reservations/{bookId}	Reserves a specific book identified by its id	**201** (Created), **400** (Bad Request), **401** (Unauthorized, only admins and the specified user may create reservations for himself), **403** (Forbidden), **500** (Internal Server Error)

Caching:

- **GET /books:** Since the books may not be updated too often, it may be beneficial to have it cached with a max-age of 1 hour
- **GET /authors:** Since the authors may rarely change we could leave a cache with a max-age of 1 day
- **GET /authors/{authorId}:** The single author information may change even less often than the full authors list, a possible cache max-age could be 1 day as well
- **GET /categories:** The list of categories may rarely change if at all, the max-age for these can be of 1 week
- **GET /categories/categoryId:** The category information is expected to remain static and only changed if absolutely necessary or a mistake was made. The max-age for these can be of 1 month
- **POST /login:** Opted to return a JWT with an expiration time of one year

Pagination

- **GET /books, GET /categories, GET /authors:**
  - **limit:** Parameter that can be used to limit the amount of search results
  - **offset:** We can use offset alongside limit to be more specific about our results
  - **page:** Parameter to specify the starting page for the results
  - **order:** To sort our results, in this case alphabetically can work for all books, categories, and authors results
- **GET /books:**
  - **order_by:** Specifically for books, we can use this parameter to order by author, language, title, or category
- **GET /authors:**
  - **order_by:** Specifically for authors, we can use this parameter to order by number of books, country of origin, or primary language