Library Catalogue

Description of Entities

Book

Attributes:

id (UUID) - Unique identifier.

title (String) - The book title.

authorId (UUID) - Reference to the author.

categoryId (UUID) - Reference to the category.

publishedYear (Integer) - Year of publication.

availableCopies (Integer) - Number of copies available for lending.

Author

Attributes:

id (UUID) - Unique identifier.

name (String) - Full name.

birthYear (Integer) - Birth year.

Category

Attributes:

id (UUID) - Unique identifier.

name (String) - Name of the category.

description (String) - Description of the category.

Operations Supported

Books

Create a book: Add a new book to the library.

Read books: Retrieve all books or filter by title, author, or category.

Retrieve a specific book: Fetch details of a specific book by ID.

Update a book: Modify the details of a book.

Delete a book: Remove a book from the library.

Authors

Create an author: Add a new author to the library system.

Read authors: Retrieve all authors or filter by name.

Retrieve a specific author: Fetch details of a specific author by ID.

Update an author: Modify author details.

Delete an author: Remove an author from the library system.

Categories

Create a category: Add a new category to the library system.

Read categories: Retrieve all categories.

Retrieve a specific category: Fetch details of a specific category by ID.

Update a category: Modify category details.

Delete a category: Remove a category.

REST API Design

Base URL:

https://api.librarycatalogue.com/v1

Endpoints

Books

POST /books

GET /books

Filters: title, authorId, categoryId, publishedYear.

Pagination: ?page=1&size=10.

GET /books/{id}

PUT /books/{id}

DELETE /books/{id}

Authors

POST /authors

GET /authors

Filters: name.

Pagination: ?page=1&size=10.

GET /authors/{id}

PUT /authors/{id}

DELETE /authors/{id}

Categories

POST /categories

GET /categories

Pagination: ?page=1&size=10.

GET /categories/{id}

PUT /categories/{id}

DELETE /categories/{id}

Functional and Non-functional Requirements

Functional Requirements:

Support CRUD operations for all entities.

Enable filters and pagination to manage large datasets.

Return meaningful status codes for all operations.

Provide appropriate authentication and error handling.

Non-functional Requirements:

API responses should be JSON.

The system must adhere to Level 3 of the Richardson Maturity Model, including HATEOAS.

The system must be scalable and cacheable for frequent read operations.

Response times must be below 300ms for 95% of requests.

Status Codes

200 OK: Success for GET, PUT, DELETE requests.

201 Created: Success for POST requests.

400 Bad Request: Validation errors.

401 Unauthorized: Authentication issues.

404 Not Found: Resource not found.

500 Internal Server Error: Server-side errors.

Authentication

Method: JWT (JSON Web Tokens)

Access Token: Short-lived, includes user permissions and roles.

Errors:

401 Unauthorized for invalid or missing tokens.

Pagination

Default: ?page=1&size=10.

Metadata in response:

```
{
  "data": [...],
  "page": 1,
  "size": 10,
  "totalPages": 5,
  "totalItems": 50
}
```

Caching

GET /books, GET /authors, GET /categories: Cacheable for 10 minutes.

Use ETag headers for validation.