# Designing a REST API

By Diego Giraldo

**System:** Catalog (a library – books, authors, categories)

**Entities:**

**Book:** A book in the library.
**Author:** An author of a book.
**Category:** A category for the book.

**1. Functional and non-functional requirements**

Functional Requirements:

**The system must...**

- allow users CRUD operations (create, read, update, and delete) over the entities (books, authors, and categories).
- allow users to search for books using their name, author or category.
- allow users to view a list of all books, authors, and categories.
- allow users to filter a list of books, using authors, and categories.
- enforce ACID rules to ensure data integrity.
- authenticate users, and allow operations according to their access rights.

Non-functional requirements:

**The system should be…**

- available 24/7 , with minimal downtime.
- able to handle large number of users without losing performance
- protected against unauthorized access to the system.
- easy to use and understand.
- be responsive and able to handle requests quickly, even under heavy loads.
- be easy to maintain and update, with minimal downtime during maintenance and updates.
- extensible, allowing new features and functionality to be added easily.

**2. Model description**

**Book Model:**
ID: Unique identifier for the book (integer).
Title: Title of the book (string).
AuthorID: Foreign Key for the Author (integer).
CategoryID: Foreign Key for the Category (integer).
ISBN: International Standard Book Number of the book (string).
Description: Description of the book (string).

**Author Model:**
ID: Unique identifier for the author (integer).
Name: Name of the author (string).
Birth Date: Date of birth of the author (date).

**Category Model:**
ID: Unique identifier for the category (integer).
Name: Name of the category (string).
Description: Description of the category (string).

**API Description:**

### 6. Authentication

Protocol - HTTPS
End Point – `https://www.library.com/v1/login`
HTTP Method - POST
Headers - None
Request Body – JSON

```
{
  "username": "user",
  "password": "password"
}
```

Response Body – JSON

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_type": "bearer"
}
```

**3. Operations Description:**

Protocol: HTTP/HTTPS
Endpoint: www.library.com/api/v1/

URI Design:

- Books:
  - Retrieve all books: GET www.library.com/api/v1/books/
  - Retrieve a specific book: GET www.library.com/api/v1/books/{book_id}/
  - Create a new book: POST www.library.com/api/v1/books/

- Update a book: PUT www.library.com/api/v1/books/{book_id}/
- Delete a book: DELETE www.library.com/api/v1/books/{book_id}/
- Search for books: GET www.library.com/api/v1/books/?q={search_query}&page={page_number}
- Representation Examples(JSON):

**Create a new book:**


```
{
   "title": "Cien años de soledad",
   "author_id": 1,
   "category_id": 2,
   "isbn": "78450544487879",
   "description": "Una obra maestra colombiana."
}
```


**Get a list of Books:**


```
{
   "books": [
      {
         "id": 1,
         "title": "Cien años de soledad",
         "author": {
            "id": 1,
            "name": "Gabriel Garcia Marquez",
            "birth_date": "1957-08-31"
         },
         "category": {
            "id": 2,
            "name": "Drama",
            "description": "Historia ficticia con suspenso"
         },
         "isbn": "78450544487879",
         "description": "Una obra maestra colombiana"
      }
   ],
   "total_count": 1,
   "limit": 10,
   "offset": 0
}
```


**Get a book by ID:**


```
{
   "id": 1,
   "title": "Cien años de soledad",
```

```
  "author": {
    "id": 1,
    "name": "Gabriel Garcia Marquez",
    "birth_date": "1957-08-31"
  },
  "category": {
    "id": 2,
    "name": "Drama",
    "description": "Historia ficticia con suspenso"
  },
  "isbn": "78450544487879",
   "description": "Una obra maestra colombiana"
}
```


- Authors:
  - Retrieve all authors: GET www.library.com/api/v1/authors/
  - Retrieve a specific author: GET www.library.com/api/v1/authors/{author_id}/
  - Create a new author: POST www.library.com/api/v1/authors/
  - Update an author: PUT www.library.com/api/v1/authors/{author_id}/
  - Delete an author: DELETE www.library.com/api/v1/authors/{author_id}/
- Categories:
  - Retrieve all categories: GET www.library.com/api/v1/categories/
  - Retrieve a specific category: GET www.library.com/api/v1/categories/{category_id}/
  - Create a new category: POST www.library.com/api/v1/categories/
  - Update a category: PUT www.library.com/api/v1/categories/{category_id}/
  - Delete a category: DELETE www.library.com/api/v1/categories/{category_id}/

## 4. status codes

Content Type: application/json

HTTP Methods:

- Request Body: JSON
- HTTP Headers: Accept: application/json
- **HTTP status code:**
  - 200 OK - The request was successful
  - 201 Created - The request was successful and a new resource was created
  - 400 Bad Request - The request was invalid or could not be understood
  - 401 Unauthorized - The authentication credentials were missing or incorrect
  - 403 Forbidden - The client does not have permission to access the requested resource
  - 404 Not Found - The requested resource was not found

- 405 Method Not Allowed - The requested method is not allowed for the resource
- 500 Internal Server Error - An error occurred on the server

**7. Pagination Parameters:**

- page: The page number to retrieve. Default is 1.
- limit: The maximum number of resources to return per page. Default is 10.

In the response body, the data field contains an array of books, the meta field contains metadata about the pagination (current page, limit, and total number of resources), and the links field contains links to the first, last, next, and previous pages of results, if applicable.

Example Response (JSON):

```json
{
  "data": [
    {
      "id": 1,
      "title": "The Hitchhiker's Guide to the Galaxy",
      "author": "Douglas Adams",
      "category": "Science Fiction",
      "isbn": "9780345391803",
      "description": "Seconds before the Earth is demolished to make way for a galactic freeway, Arthur Dent is plucked off the planet by his friend Ford Prefect."
    },
    {
      "id": 2,
      "title": "To Kill a Mockingbird",
      "author": "Harper Lee",
      "category": "Fiction",
      "isbn": "9780446310789",
      "description": "The unforgettable novel of a childhood in a sleepy Southern town and the crisis of conscience that rocked it."
    }
  ],
```

```
  "meta": {
    "page": 1,
    "limit": 2,
    "total": 10
  },
  "links": {
    "self": "http://www.library.com/books?page=1&limit=2",
    "next": "http://www.library.com/books?page=2&limit=2",
    "prev": null,
    "first": "http://www.library.com/books?page=1&limit=2",
    "last": "http://www.library.com/books?page=5&limit=2"
  }
}
```

**8. Caching:**

- All GET requests can be cached for a period of 60 seconds, as the data is not expected to change frequently.
- POST, PUT, PATCH, and DELETE requests are not cached, as they modify the state of the server and can change frequently