Catalogue a library – books, authors, categories, etc

Functional and non-functional requirements

1. Functional Requirements

Book: Create, read, update, delete books

Author: Manage author information and their works

Category: Organize books by categories/genres

User: Handle library users and authentication

Borrowing: Track book loans and returns

Search: Find books by various criteria

Inventory : Track book availability

2. Non-Functional Requirements

- short response time

- scale

- JWT-based authentication

- Role-based access control (Admin, Librarian, User)

- Stateless design

- Availability

- Graceful error handling

- Health check endpoints

Model description

*Data Model*

- A book can have multiple authors (M:M)

- A book belongs to one or more categories

- Users can borrow multiple books simultaneously

- Books have availability status (available, borrowed, reserved, maintenance)

- Borrowing period is 14 days with possible extension

Entities

Book

{

"id": "uuid",

"isbn": "string (unique)",

"title": "string",

"subtitle": "string-30",

"description": "string-300",

"publishedDate": "date",

"pageCount": "integer",

"language": "string",

"publisher": "string",

"edition": "string-30",

"totalCopies": "integer",

"availableCopies": "integer",

"createdAt": "datetime",

"updatedAt": "datetime",

"authors": [ **"Author"** ],

"categories": [ **"Category"** ],

"coverImage": "string (URL)",

"status": "enum [active, archived, maintenance]"

}

Author

{

"id": "uuid",

"firstName": "string",

"lastName": "string",

"biography": "string-30",

"birthDate": "date",

"nationality": "string-30",

"createdAt": "datetime",

"updatedAt": "datetime",

"books": [ **"Book"** ],

"photo": "string (URL)?"

}

Category

{

"id": "uuid",

"name": "string (unique)",

"description": "string-30",

"parentCategoryId": "uuid?",

"createdAt": "datetime",

"updatedAt": "datetime",

"books": [ **"Book"** ],

"subcategories": [ **"Category"** ]

}

User

{

"id": "uuid",

"email": "string (unique)",

"firstName": "string",

"lastName": "string",

"phoneNumber": "string-30",

"address": "Address?",

"role": "enum [admin, librarian, user]",

"membershipDate": "date",

"status": "enum [active, suspended, expired]",

"createdAt": "datetime",

"updatedAt": "datetime"

}

Loan

{

"id": "uuid",

"userId": "uuid",

"bookId": "uuid",

"loanDate": "datetime",

"dueDate": "datetime",

"returnDate": "datetime?",

"status": "enum [active, returned, overdue, extended]",

"renewalCount": "integer",

"fineAmount": "decimal",

"notes": "string-30",

"createdAt": "datetime",

"updatedAt": "datetime"

}

Address

{

"street": "string",

"city": "string",

"state": "string",

"zipCode": "string",

"country": "string"

}

4. API Endpoints (Richardson Maturity Model - Level 3)

Base URL: https://api.library.com/v1

*Authentication Endpoints*

**POST** /auth/login

Description: User authentication

*Request:*

{

"email": "user@example.com",

"password": "password123"

}

*Response (200):*

{

"accessToken": "jwt\_token",

"refreshToken": "refresh\_token",

"user": { "id": "uuid", "email": "string", "role": "string" },

"expiresIn": 3600

}

**POST** /auth/refresh

Description: Refresh access token

Headers: Authorization: Bearer refresh\_token

*Response (200):*

{

"accessToken": "new\_jwt\_token",

"expiresIn": 3600

}

**POST** /auth/logout

Description: User logout (invalidate tokens)

Headers: Authorization: Bearer access\_token

*Response (204): No content*

Book Management

**GET** /books

Description: Retrieve books with filtering, sorting, and pagination

Query Parameters:

- page: Page number (default: 1)

- limit: Items per page (default: 20, max: 100)

- search: Search in title, author name, ISBN

- category: Filter by category ID

- author: Filter by author ID

- available: Filter by availability (true/false)

- language: Filter by language

- publishedAfter: Filter by publication date

- publishedBefore: Filter by publication date

- sortBy: Sort field (title, publishedDate, createdAt)

- sortOrder: Sort direction (asc, desc)

*Response (200):*

{

"data": [

{

"id": "uuid",

"isbn": "978-0123456789",

"title": "Example Book",

"authors": [{"id": "uuid", "firstName": "John", "lastName": "Doe"}],

"categories": [{"id": "uuid", "name": "Fiction"}],

"availableCopies": 3,

"totalCopies": 5,

"publishedDate": "2023-01-15"

}

],

"pagination": {

"page": 1,

"limit": 20,

"total": 150,

"totalPages": 8,

"hasNext": true,

"hasPrev": false

},

"links": {

"self": "/books?page=1&limit=20",

"next": "/books?page=2&limit=20",

"first": "/books?page=1&limit=20",

"last": "/books?page=8&limit=20"

}

}

**GET** /books/{id}

Description: Retrieve specific book

*Response (200): Complete book object*

Cache: 5 minutes

**POST** /books

Description: Create new book (Admin/Librarian only)

*Request:*

{

"isbn": "978-0123456789",

"title": "New Book",

"authors": ["author\_id\_1", "author\_id\_2"],

"categories": ["category\_id\_1"],

"publishedDate": "2023-01-15",

"totalCopies": 5,

"publisher": "Example Publisher"

}

*Response (201): Created book object*

**PUT** /books/{id}

Description: Update book (Admin/Librarian only)

*Response (200): Updated book object*

**DELETE** /books/{id}

Description: Delete book (Admin only)

*Response (204): No content*

Author Management

**GET** /authors

Description: List authors with pagination

Query Parameters: Similar to books (page, limit, search, sortBy, sortOrder)

*Response(200): Paginated author list*

Cache: 10 minutes

**GET** /authors/{id}

Description: Get specific author

*Response (200): Complete author object including books*

Cache: 5 minutes

/authors/{id}/books

Description: Get books by specific author

Query Parameters: Pagination and filtering parameters

*Response (200): Paginated book list*

Cache: 5 minutes

**POST** /authors

Description: Create author (Admin/Librarian only)

*Response (201): Created author object*

**PUT** /authors/{id}

Description: Update author (Admin/Librarian only)

*Response(200): Updated author object*

**DELETE** /authors/{id}

Description: Delete author (Admin only)

*Response (204): No content*

Category Management

**GET** /categories

Description: List categories (hierarchical structure)

*Response(200):*

{

"data": [

{

"id": "uuid",

"name": "Fiction",

"subcategories": [

{"id": "uuid", "name": "Science Fiction"},

{"id": "uuid", "name": "Fantasy"}

],

"bookCount": 150

}

]

}

Cache: 15 minutes

**GET** /categories/{id}/books

Description: Get books in category

*Response (200): Paginated book list*

Cache: 5 minutes

**POST** /categories

Description: Create category (Admin/Librarian only)

**PUT** /categories/{id}

Description: Update category (Admin/Librarian only)

**DELETE** /categories/{id}

Description: Delete category (Admin only)

User Management

**GET** /users

Description: List users (Admin/Librarian only)

Query Parameters: Pagination, search, role filter

*Respons (200): Paginated user list*

**GET** /users/{id}

Description: Get user details

*Response (200): User object (filtered by permissions)*

**GET** /users/me

Description: Get current user profile

*Response(200): Current user object*

**PUT** /users/{id}

Description: Update user

*Response (200): Updated user object*

**PUT** /users/me

Description: Update current user profile

*Response(200): Updated user object*

Loan Management

**GET** /loans

Description: List loans

Query Parameter:

- userId: Filter by user (admins can see all)

- status: Filter by loan status

- overdue: Show only overdue loans

- dueDate: Filter by due date range

*Response(200): Paginated loan list*

**GET** /loans/{id}

Description: Get specific loan

*Response (200): Loan object*

**POST** /loans

Description: Create new loan (borrow book)

Request:

{

"userId": "uuid",

"bookId": "uuid",

"dueDate": "2023-12-15"

}

*Respons(201): Created loan object*

**PUT** /loans/{id}/return

Description: Return book

Response(200): Updated loan object

**PUT** /loans/{id}/extend

Description: Extend loan period

*Response (200): Updated loan object*

Search Endpoints

**GET** /search

Descriptio: Global search across books, authors, categories

Query Parameters:

- q: Search query

- type: Search type (books, authors, categories, all)

- page, limit: Pagination

*Response(200): Mixed search results*

Cache: 2 minutes

5. HTTP Status Codes

*Success Codes*

- 200 OK: Successful GET, PUT requests

- 201 Created: Successful POST requests

- 204 No Content: Successful DELETE requests

*Client Error Codes*

- 400 Bad Request: Invalid request format or parameters

- 401 Unauthorized: Missing or invalid authentication

- 403 Forbidden: Insufficient permissions

- 404 Not Found: Resource not found

- 409 Conflict: Resource conflict (e.g., ISBN already exists)

- 422 Unprocessable Entity: Validation errors

- 429 Too Many Requests: Rate limit exceededServer Error Codes

- 500 Internal Server Error: Unexpected server error

- 502 Bad Gateway: Upstream service error

- 503 Service Unavailable: Temporary service unavailability

6. Error Response Format

{

"error": {

"code": "VALIDATION\_ERROR",

"message": "Validation failed",

"details": [

{

"field": "isbn",

"message": "ISBN must be unique",

"code": "DUPLICATE\_VALUE"

}

],

"timestamp": "2023-12-01T10:30:00Z",

"path": "/books",

"requestId": "req-123456"

}

}

*Error Codes*

- VALIDATION\_ERROR: Input validation failed

- RESOURCE\_NOT\_FOUND: Requested resource not found

- UNAUTHORIZED\_ACCESS: Authentication required

- INSUFFICIENT\_PERMISSIONS: Access denied

- RESOURCE\_CONFLICT: Resource conflict

- RATE\_LIMIT\_EXCEEDED: Too many requests

- INTERNAL\_ERROR: Server error

7. Authentication & Authorization

*JWT Structure*

{

"header": {

"alg": "HS256",

"typ": "JWT"

},

"payload": {

"sub": "user\_id",

"email": "user@example.com",

"role": "user",

"iat": 1638360000,

"exp": 1638363600,

"permissions": ["read:books", "create:loans"]

}

}

*Permission System*

- Admins: Full CRUD access to all resources

- Librarians: CRUD books, authors, categories; manage loans

- Users: Read access; manage own profile and loans

*Security Headers*

- Authorization: Bearer {jwt\_token}

- All requests must use HTTPS

- CORS configured for allowed origins

- Rate limiting: 100 requests/minute per user

8. Caching Strategy

*Cached Endpoints*

- GET /books: 5 minutes (frequently updated with availability)

- GET /books/{id}: 5 minutes

- GET /authors: 10 minutes (less frequently updated)

- GET /authors/{id}: 5 minutes

- GET /categories: 15 minutes (rarely updated)

- GET /search: 2 minutes (balance between freshness and performance)

*Non-Cached Endpoints*

- User-specific data (/users/me, /loans): Real-time accuracy required

- POST/PUT/DELETE operations: Mutations shouldn't be cached

- Authentication endpoints: Security sensitive

- Admin operations: Require real-time data

Cache Headers

Cache-Control: public, max-age=300

ETag: "abc123"

Last-Modified: Wed, 01 Dec 2023 10:00:00 GMT

9. Pagination

*Standard Pagination*

{

"pagination": {

"page": 1,

"limit": 20,

"total": 150,

"totalPages": 8,

"hasNext": true,

"hasPrev": false

},

"links": {

"self": "/books?page=1&limit=20",

"next": "/books?page=2&limit=20",

"prev": null,

"first": "/books?page=1&limit=20",

"last": "/books?page=8&limit=20"

}

}

*Cursor-based Pagination (for real-time data)*

For endpoints with frequently changing data:

{

"pagination": {

"cursor": "eyJpZCI6InV1aWQiLCJ0aW1lc3RhbXAiOiIyMDIzLTEyLTAxVDEwOjMwOjAwWiJ9",

"hasNext": true,

"limit": 20

},

"links": {

"next": "/loans?cursor=abc123&limit=20"

}

}

10. API Versioning

- Version in URL: /v1/

- Backward compatibility maintained for at least 2 versions

- Deprecation notices in response headers:

Deprecated: true

Sunset: Wed, 01 Jun 2024 10:00:00 GMT

Link: </v2/books>; rel="successor-version"

11. Health Check & Monitoring

**GET** /health

Description: Service health status

*Response (200):*

{

"status": "healthy",

"timestamp": "2023-12-01T10:30:00Z",

"version": "1.2.3",

"dependencies": {

"database": "healthy",

"cache": "healthy",

"external\_api": "degraded"

}

}