Designing a REST API Practice
[LatAm] Introduction to Java Development

Student:
Julián Andrés Gaitán Cañas

Teacher:
Juan Cardona

EPAM
February of 2023

Overview

For this exercise a **“Library catalog”** was chosen, the idea is to let the library have a digital record for every book owned, with the possibility to add, modify and remove them. Also give the library the capability to organize said books with bookshelves that will also be able to be added, modified and removed. Finally, to have people (users) borrow books from the library, a system to track these books will be implemented.

Functional Requirements

The system must:

- Allows the creation and usage of an user account (sign up, sign in)
- Retrieve a list of all the books in the library (paginated accordingly)
- Retrieve a list of books filter by: id, bookshelf, title, author, category, publication year, number of pages, language (paginated accordingly)
- Allow to reserve a book to be borrowed through his id
- Add a new book to the library
- Update details of an existing book
- Delete a book from the library
- Retrieve a list of all bookshelves in the library (paginated accordingly)
- Retrieve a list of bookshelves filter by: id, label, floor, section, available space
- Add a new bookshelf to the library
- Update details of an existing bookshelf
- Delete a bookshelf from the library

Non-Functional Requirements

The system should:

- Follow the constraints of the REST architectural style
- Be at level 3 in the Richardson Maturity Model
- Be a service that can be accessible through the web
- Be able to communicate through HTTP request/response pairs
- Implement authentication and authorization to allow/restring the usage of the API
- Respond and receive information through JSON y XML format
- Use appropied status codes to denote the intent of all responses (successful or not)
- Support pagination for all relevant queries
- Support caching for all relevant endpoints

Use Cases

Title	**Authentication**
Primary Actor	User
Success Scenario	1. The user inputs the username and password in the corresponding entrypoint (URI) 2. If the user don't have an existing account, the system offers to create one 3. The system generates a token (1 day before expiration) to be used for the user 4. The user is authenticated
Extensions	1a.The username and/or password are incorrect and the system can verified the user
Preconditions	

Title	**Obtain books**
Primary Actor	User, Admin
Success Scenario	1. The user access the corresponding URI with the correct HTTP method and parameters 2. The system responds back with the correct status code and content in JSON or XML format 3. The response body will show the corresponding page of the list of all the books in the library ordered by title (ascending) 4. The objects in the list will correspond to all the information in a book: id, bookshelf, title, author, category, publication year, number of pages, language
Extensions	3a.The response body will show no results if the page specified do not exist
Preconditions	Authentication

Title	**Obtain specific books**
Primary Actor	User, Admin
Success Scenario	1. The user access the corresponding URI with the correct HTTP method and parameters 2. The system responds back with the correct status code and content in JSON or XML format 3. The response body will show the corresponding page of the list of the books that match the specified parameters by the user in the library ordered by title (ascending) 4. The objects in the list will correspond to all the information in a book: id, bookshelf, title, author, category, publication year, number of pages, language
Extensions	3a. The response body will show no results if the parameters don't match any book in the library
Preconditions	Authentication

Title	**Borrow a book**
Primary Actor	Admin
Success Scenario	1. The admin access the corresponding URI with the correct HTTP method and parameter 2. The admin input the corresponding id of the book to be borrowed and de id of the user taking it with the appropriate HTTP method 3. The system responds back with a message indicating the outcome of the operation 4. The response body will show all the info of the book, user, date and time:
Extensions	2a. The book can't be marked as borrowed because is already occupied 2b. The user is not allowed to borrow books
Preconditions	Authentication

Title	**Return a book**
Primary Actor	Admin
Success Scenario	1. The admin access the corresponding URI with the correct HTTP method and parameter 2. The admin input the corresponding id of the book to be returned with the appropriate HTTP method 3. The system responds back with a message indicating the outcome of the operation 4. The response body will show all the info of the book, user, date and time:
Extensions	2a.The book can’t be marked as returned because is already available
Preconditions	Authentication

Title	**Add a new book**
Primary Actor	Admin
Success Scenario	1. The admin access the corresponding URI with the correct HTTP method 2. The admin sends all the information of the new book in the appropriate JSON or XML format and HTTP method 3. The system responds back with a message indicating the outcome of the operation 4. The response body will show all the info of the book: id, bookshelf, title, author, category, publication year, number of pages, language
Extensions	3a.The system will fail to add the new book if some required information is missing or if the bookshelf do not exist or if the bookshelf has no space available 3b.The operation will fail if the user is not authorized to add new books to the system
Preconditions	Authentication

Title	**Update an existing book**
Primary Actor	Admin
Success Scenario	1. The admin access the corresponding URI with the correct HTTP method and parameters 2. The admin sends all the information of the book to be updated in the appropriate JSON or XML format and HTTP method 3. The system responds back with a message indicating the outcome of the operation 4. The response body will show all the info of the book: id, bookshelf, title, author, category, publication year, number of pages, language
Extensions	3a.The system will fail to update an existing book if the bookshelf do not exist or if the bookshelf has no space available 3b.The operation will fail if the user is not authorized to update any existing book in the system
Preconditions	Authentication

Title	**Delete a book**
Primary Actor	Admin
Success Scenario	1. The admin access the corresponding URI with the correct HTTP method and parameters 2. The admin input the corresponding id of the book to be deleted with the appropriate HTTP method 3. The system responds back with a message indicating the outcome of the operation 4. The response body will show all the info of the deleted book: id, bookshelf, title, author, category, publication year, number of pages, language
Extensions	3a.The system will fail to delete an existing book if the id don't exist 3b.The operation will fail if the user is not authorized to delete any existing book in the system
Preconditions	Authentication

Title	**Obtain bookshelves**
Primary Actor	User, Admin
Success Scenario	1. The user access the corresponding URI with the correct HTTP method and parameters 2. The system responds back with the correct status code and content in JSON or XML format 3. The response body will show the corresponding page of the list of all the bookshelves in the library ordered by label (ascending) 4. The objects in the list will correspond to all the information in a bookshelf: id, label, floor, section, available space
Extensions	3a. The response body will show no results if the page specified do not exist
Preconditions	Authentication

Title	**Obtain specific bookshelves**
Primary Actor	User, Admin
Success Scenario	1. The user access the corresponding URI with the correct HTTP method and parameters 2. The system responds back with the correct status code and content in JSON or XML format 3. The response body will show the corresponding page of the list of the bookshelves that match the specified parameters by the user in the library ordered by label (ascending) 4. The objects in the list will correspond to all the information in a bookshelf: id, label, floor, section, available space
Extensions	3a. The response body will show no results if the parameters don’t match any bookshelf in the library
Preconditions	Authentication

Title	**Add a new bookshelf**
Primary Actor	Admin
Success Scenario	1. The admin access the corresponding URI with the correct HTTP method 2. The admin sends all the information of the new bookshelf in the appropriate JSON or XML format and HTTP method 3. The system responds back with a message indicating the outcome of the operation 4. The response body will show all the info of the bookshelf: id, label, floor, section, available space
Extensions	3a.The system will fail to add the new bookshelf if some required information is missing 3b.The operation will fail if the user is not authorized to add new bookshelves to the system
Preconditions	Authentication

Title	**Update an existing bookshelf**
Primary Actor	Admin
Success Scenario	1. The admin access the corresponding URI with the correct HTTP method and parameters 2. The admin sends all the information of the bookshelf to be updated in the appropriate JSON or XML format and HTTP method 3. The system responds back with a message indicating the outcome of the operation 4. The response body will show all the info of the bookshelf: id, label, floor, section, available space
Extensions	3a.The operation will fail if the user is not authorized to update any existing bookshelf in the system
Preconditions	Authentication

Title	**Delete a bookshelf**
Primary Actor	Admin
Success Scenario	1. The admin access the corresponding URI with the correct HTTP method and parameters 2. The admin input the corresponding id of the bookshelf to be deleted with the appropriate HTTP method 3. The system responds back with a message indicating the outcome of the operation 4. The response body will show all the info of the deleted bookshelf: id, label, floor, section, available space
Extensions	3a.The system will fail to delete an existing bookshelf if the id don't exist 3b.The operation will fail if the user is not authorized to delete any existing bookshelf in the system
Preconditions	Authentication

Entities

User/Admin

FirstName	LastName
Username	Password
BooksInPossession	AuthorizationLevel

Book

ID	Bookshelf
Title	Author
Category	PublicationYear
NumberOfPages	Language
IsAvailable	

Bookshelf

ID	Label
Floor	Section
AvailableSpace	

REST API

1. Authentication successful:

- End Point : `https://www.mylibrarycatalog.com:8080/login`
- URI Design: `https://www.mylibrarycatalog.com:8080/login`

Request

```
POST /login

Content-Type: application/json; charset=UTF-8
...
```

{
   "username": "my_user_name",
   "password": "pass123"
}
	<login>
	  <username>my_user_name</username>
	  <password>pass123</password>
	</login>

Response

```
HTTP/2.0 200 OK

Content-Type: application/json; charset=UTF-8
Cache-Control: max-age=3600, must-revalidate
Expires: Mon, XX xxx 20XX XX:XX:XX GMT
...
```

{
   "JWTtoken": "aBcDeFg123456"
}
	<login>
	  <JWTtoken>aBcDeFg123456</JWTtoken>
	</login>

2. Authentication failure:

- End Point : `https://www.mylibrarycatalog.com:8080/login`
- URI Design: `https://www.mylibrarycatalog.com:8080/login`

Request

```
POST /login

Content-Type: application/json; charset=UTF-8
...
```

```
{
   "username": "invalid_user",
   "password": "invalid_pass"
}
```

```
<login>
  <username>invalid_user</username>
  <password>invalid_pass</password>
</login>
```

Response

```
HTTP/2.0 401 Unauthorized

Content-Type: application/json; charset=UTF-8
Cache-Control: max-age=0 (Error responses are not cached)
...
```

```
{
   "error": "wrong user and/or
             password"
}
```

```
<login>
   <error>wrong user and/or
          password</error>
</login>
```

3. Registration successful:

- End Point : `https://www.mylibrarycatalog.com:8080/register`
- URI Design: `https://www.mylibrarycatalog.com:8080/register`

Request

```
POST /register

Content-Type: application/json; charset=UTF-8
...
```

{
    "username": "my_user_name",
    "password": "pass123"
}
	<register>
	  <username>my_user_name</username>
	  <password>pass123</password>
	</register>

Response

```
HTTP/2.0 200 OK

Content-Type: application/json; charset=UTF-8
Cache-Control: max-age=3600, must-revalidate
Expires: Mon, XX xxx 20XX XX:XX:XX GMT
...
```

{
   “ID”: “456”
   "JWTtoken": "aBcDeFg123456"
}
	<register>
	  <ID>123</ID>
	  <JWTtoken>aBcDeFg123456</JWTtoken>
	</register>

4. Registration failure:

- End Point : `https://www.mylibrarycatalog.com:8080/register`
- URI Design: `https://www.mylibrarycatalog.com:8080/register`

Request

```
POST /register

Content-Type: application/json; charset=UTF-8
...
```

```
{
    "password": "pass123"
}
```

```
<register>
  <password>pass123</password>
</register>
```

Response

```
HTTP/2.0 400 Bad Request

Content-Type: application/json; charset=UTF-8
Cache-Control: max-age=0 (Error responses are not cached)
...
```

```
{
   "error": "missing user and/or
             password"
}
```

```
<register>
   <error>wrong user and/or
          password</error>
</register>
```

5. List books:

- End Point : `https://www.mylibrarycatalog.com:8080/books`
- URI Design: `https://www.mylibrarycatalog.com:8080/books`
  `https://www.mylibrarycatalog.com:8080/books?page={number}`

Request

```
GET /books
GET /books?page=3

Authorization: myUserName aBcDeFg123456
...
```

Response

```
HTTP/2.0 200 OK

Content-Type: application/json; charset=UTF-8
Cache-Control: max-age=3600, must-revalidate
Expires: Mon, XX xxx 20XX XX:XX:XX GMT
...
```

```
{
  "pagination-count": "10",
  "pagination-page": "3",
  "pagination-limit": "6",
  "books": [
    {
     "ID": "123",
     "bookshelf_ID": "789",
     "title": "My Book",
     "author": "John Doe",
     "category": "Book Category",
     "publication-year": "19XX",
     "number-of-pages": "100",
     "language": "English",
     "is-available": "true",
     "cover-front":
       "http://myhost/coverF.png",
     "cover-back":
       "http://myhost/coverB.png",
     "borrow-book":
       "/books/123/borrow",
     "return-book":
       "/books/123/return"
    },
    ...
  ]
}
```

```
<books>
  <pagination-count>10</pagination-count>
  <pagination-page>3</pagination-page>
  <pagination-limit>6</pagination-limit>
  <book>
    <ID>123</ID>
    <bookshelf_ID>789</bookshelf_ID>
    <title>My Book</title>
    <author>John Doe</author>
    <category>Book Category</category>
    <publication-year>19XX</publication-year>
    <number-of-pages>100</number-of-pages>
    <language>English</language>
    <is-available>true</is-available>
    <cover-front>http://myhost/coverF.png
      </cover-front>
    <cover-back>http://myhost/coverB.png
      </cover-back>
    <borrow-book>/books/123/borrow
      </borrow-book>
    <return-book>/books/123/return
      </return-book>
  </book>
  ...
</books>
```

6. List books (filter):

- End Point : `https://www.mylibrarycatalog.com:8080/books`
- URI Design: `https://www.mylibrarycatalog.com:8080/books?title={string}&author={string}&category={string}`
  `https://www.mylibrarycatalog.com:8080/books?title={string}&author={string}&category={string}&page={number}`

Request

```
GET /books?title="My Book"&author="John Doe"&category="Book Category"
GET /books?title="My Book"&author="John Doe"&category="Book Category"&page=2

Authorization: myUserName aBcDeFg123456
...
```

Response

```
HTTP/2.0 200 OK

Content-Type: application/json; charset=UTF-8
Cache-Control: max-age=3600, must-revalidate
Expires: Mon, XX xxx 20XX XX:XX:XX GMT
...
```

```
{
  "pagination-count": "3",
  "pagination-page": "2",
  "pagination-limit": "6",
  "books": [
    {
     "ID": "123",
     "bookshelf_ID": "789",
     "title": "My Book",
     "author": "John Doe",
     "category": "Book Category",
     "publication-year": "19XX",
     "number-of-pages": "100",
     "language": "English",
     "is-available": "true",
     "cover-front":
       "http://myhost/coverF.png",
     "cover-back":
       "http://myhost/coverB.png",
     "borrow-book":
       "/books/123/borrow",
     "return-book":
       "/books/123/return"
    },
    ...
  ]
}
```

```
<books>
  <pagination-count>3</pagination-count>
  <pagination-page>2</pagination-page>
  <pagination-limit>6</pagination-limit>
  <book>
    <ID>123</ID>
    <bookshelf_ID>789</bookshelf_ID>
    <title>My Book</title>
    <author>John Doe</author>
    <category>Book Category</category>
    <publication-year>19XX</publication-year>
    <number-of-pages>100</number-of-pages>
    <language>English</language>
    <is-available>true</is-available>
    <cover-front>http://myhost/coverF.png
      </cover-front>
    <cover-back>http://myhost/coverB.png
      </cover-back>
    <borrow-book>/books/123/borrow
      </borrow-book>
    <return-book>/books/123/return
      </return-book>
  </book>
  ...
</books>
```

7. List books (no results):

- End Point : `https://www.mylibrarycatalog.com:8080/books`
- URI Design: `https://www.mylibrarycatalog.com:8080/books?page={number}`

Request

```
GET /books?page=10000

Authorization: myUserName aBcDeFg123456
...
```

Response

```
HTTP/2.0 404 Not Found

Content-Type: application/json; charset=UTF-8
Cache-Control: max-age=0 (Error responses are not cached)
...
```

{
 	
}
	<root>
	
	</root>

8. Borrow a book:

- End Point : `https://www.mylibrarycatalog.com:8080/books/<id>/borrow`
- URI Design: `https://www.mylibrarycatalog.com:8080/books/{number}/borrow?userID={number}`

Request

```
PUT /books/123/borrow
PUT /books/123/borrow?userID=456

Authorization: myUserName aBcDeFg123456
...
```

Response

```
HTTP/2.0 200 OK
HTTP/2.0 403 Forbidden (if the user doesn't have the correct permissions)

Content-Type: application/json; charset=UTF-8
Cache-Control: max-age=3600, must-revalidate
...
```

```
{
  "date": XX-XX-XXXX,
  "userID": "456",
  "book": {
     "ID": "123",
     "bookshelf_ID": "789",
     "title": "My Book",
     "author": "John Doe",
     "category": "Book Category",
     "publication-year": "19XX",
     "number-of-pages": "100",
     "language": "English",
     "is-available": "false",
     "cover-front":
       "http://myhost/coverF.png",
     "cover-back":
       "http://myhost/coverB.png",
     "return-book":
       "/books/123/return"
  }
}
```

```
<borrow-book>
  <date>XX-XX-XXXX</date>
  <userID>456</userID>
  <book>
    <ID>123</ID>
    <bookshelf_ID>789</bookshelf_ID>
    <title>My Book</title>
    <author>John Doe</author>
    <category>Book Category</category>
    <publication-year>19XX</publication-year>
    <number-of-pages>100</number-of-pages>
    <language>English</language>
    <is-available>false</is-available>
    <cover-front>http://myhost/coverF.png
      </cover-front>
    <cover-back>http://myhost/coverB.png
      </cover-back>
    <return-book>/books/123/return
      </return-book>
  </book>
<borrow-book>
```

9. Return a book:

- End Point : `https://www.mylibrarycatalog.com:8080/books/<id>/return`
- URI Design: `https://www.mylibrarycatalog.com:8080/books/{number}/return`

Request

```
PUT /books/123/return

Authorization: myUserName aBcDeFg123456
...
```

Response

```
HTTP/2.0 200 OK
HTTP/2.0 403 Forbidden (if the user doesn't have the correct permissions)

Content-Type: application/json; charset=UTF-8
Cache-Control: max-age=3600, must-revalidate
...
```

```
{
  "date": XX-XX-XXXX,
  "userID": "456",
  "book": {
     "ID": "123",
     "bookshelf_ID": "789",
     "title": "My Book",
     "author": "John Doe",
     "category": "Book Category",
     "publication-year": "19XX",
     "number-of-pages": "100",
     "language": "English",
     "is-available": "true",
     "cover-front":
       "http://myhost/coverF.png",
     "cover-back":
       "http://myhost/coverB.png",
     "borrow-book":
       "/books/123/borrow?userID=456"
  }
}
```

```
<return-book>
  <date>XX-XX-XXXX</date>
  <userID>456</userID>
  <book>
    <ID>123</ID>
    <bookshelf_ID>789</bookshelf_ID>
    <title>My Book</title>
    <author>John Doe</author>
    <category>Book Category</category>
    <publication-year>19XX</publication-year>
    <number-of-pages>100</number-of-pages>
    <language>English</language>
    <is-available>true</is-available>
    <cover-front>http://myhost/coverF.png
      </cover-front>
    <cover-back>http://myhost/coverB.png
      </cover-back>
    <borrow-book>/books/123/borrow?userID=456
      </borrow-book>
  </book>
<return-book>
```

10. Add a book:

- End Point : `https://www.mylibrarycatalog.com:8080/books`
- URI Design: `https://www.mylibrarycatalog.com:8080/books`

Request

```
POST /books

Content-Type: application/json; charset=UTF-8
Authorization: myUserName aBcDeFg123456
...
```

```
{
  "book": {
     "ID": "123",
     "bookshelf_ID": "789",
     "title": "My Book",
     "author": "John Doe",
     "category": "Book Category",
     "publication-year": "19XX",
     "number-of-pages": "100",
     "language": "English",
     "cover-front":
       "http://myhost/coverF.png",
     "cover-back":
       "http://myhost/coverB.png"
  }
}
```

```
<new-book>
  <book>
    <ID>123</ID>
    <bookshelf_ID>789</bookshelf_ID>
    <title>My Book</title>
    <author>John Doe</author>
    <category>Book Category</category>
    <publication-year>19XX</publication-year>
    <number-of-pages>100</number-of-pages>
    <language>English</language>
    <cover-front>http://myhost/coverF.png
      </cover-front>
    <cover-back>http://myhost/coverB.png
      </cover-back>
  </book>
<new-book>
```

Response

```
HTTP/2.0 201 Created
HTTP/2.0 400 Bad Request (if required information is missing)
HTTP/2.0 403 Forbidden (if the user doesn't have the correct permissions)

Content-Type: application/json; charset=UTF-8
Cache-Control: max-age=3600, must-revalidate
...
```

```
{
  "date": XX-XX-XXXX,
  "book": {
     ... // same as above
  }
}
```

```
<new-book>
  <date>XX-XX-XXXX</date>
  <book>
    ... <!-- same as above -->
  </book>
<new-book>
```

11. Update a book:

- End Point : `https://www.mylibrarycatalog.com:8080/books/<id>`
- URI Design: `https://www.mylibrarycatalog.com:8080/books/{number}`

Request

```
PUT /books/123

Content-Type: application/json; charset=UTF-8
Authorization: myUserName aBcDeFg123456
...
```

```
{
  "book": {
     "bookshelf_ID": "987",
     "cover-front":
       "http://myhost/newCoverF.png"
  }
}
```

```
<update-book>
  <book>
    <bookshelf_ID>987</bookshelf_ID>
    <cover-front>http://myhost/newCoverF.png
      </cover-front>
  </book>
<update-book>
```

Response

```
HTTP/2.0 200 OK
HTTP/2.0 403 Forbidden (if the user doesn't have the correct permissions)

Content-Type: application/json; charset=UTF-8
Cache-Control: max-age=3600, must-revalidate
...
```

```
{
  "book": {
     "ID": "123",
     "bookshelf_ID": "987",
     "title": "My Book",
     "author": "John Doe",
     "category": "Book Category",
     "publication-year": "19XX",
     "number-of-pages": "100",
     "language": "English",
     "cover-front":
       "http://myhost/newCoverF.png",
     "cover-back":
       "http://myhost/coverB.png"
  }
}
```

```
<update-book>
  <book>
    <ID>123</ID>
    <bookshelf_ID>987</bookshelf_ID>
    <title>My Book</title>
    <author>John Doe</author>
    <category>Book Category</category>
    <publication-year>19XX</publication-year>
    <number-of-pages>100</number-of-pages>
    <language>English</language>
    <cover-front>http://myhost/newCoverF.png
      </cover-front>
    <cover-back>http://myhost/coverB.png
      </cover-back>
  </book>
<update-book>
```

12. Delete a book:

- End Point : `https://www.mylibrarycatalog.com:8080/books/<id>`
- URI Design: `https://www.mylibrarycatalog.com:8080/books/{number}`

Request

```
DELETE /books/123

Content-Type: application/json; charset=UTF-8
Authorization: myUserName aBcDeFg123456
...
```

Response

```
HTTP/2.0 404 Not Found
HTTP/2.0 403 Forbidden (if the user doesn't have the correct permissions)

Content-Type: application/json; charset=UTF-8
Cache-Control: max-age=3600, must-revalidate
...
```

```
{
  "date": XX-XX-XXXX,
  "book": {
     "ID": "123"
  }
}
```

```
<delete-book>
  <date>XX-XX-XXXX</date>
  <book>
    <ID>123</ID>
  </book>
<delete-book>
```

13. List bookshelves:

- End Point : `https://www.mylibrarycatalog.com:8080/bookshelves`
- URI Design: `https://www.mylibrarycatalog.com:8080/bookshelves`
  `https://www.mylibrarycatalog.com:8080/bookshelves?page={number}`

Request

```
GET /bookshelves
GET /bookshelves?page=2

Authorization: myUserName aBcDeFg123456
...
```

Response

```
HTTP/2.0 200 OK

Content-Type: application/json; charset=UTF-8
Cache-Control: max-age=3600, must-revalidate
Expires: Mon, XX xxx 20XX XX:XX:XX GMT
...
```

```
{
  "pagination-count": "4",
  "pagination-page": "2",
  "pagination-limit": "8",
  "bookshelves": [
    {
     "ID": "789",
     "label": "My Label",
     "floor": "1",
     "section": "Section A",
     "available-space": "10"
    },
    ...
  ]
}
```

```
<bookshelves>
  <pagination-count>4</pagination-count>
  <pagination-page>2</pagination-page>
  <pagination-limit>8</pagination-limit>
  <bookshelf>
    <ID>789</ID>
    <label>My Label</label>
    <floor>1</floor>
    <section>Section A</section>
    <available-space>10</available-space>
  </bookshelf>
  ...
</bookshelves>
```

14. List bookshelves (filter):

- End Point : `https://www.mylibrarycatalog.com:8080/bookshelves`
- URI Design: `https://www.mylibrarycatalog.com:8080/bookshelves`
  `https://www.mylibrarycatalog.com:8080/bookshelves?label={string}&section={string}&ava_space={number}`
  `https://www.mylibrarycatalog.com:8080/bookshelves?label={string}&section={string}&ava_space={number}&page={number}`

Request

```
GET /bookshelves?label="My Label"&section="Section A"&ava_space=10
GET /bookshelves?label="My Label"&section="Section A"&ava_space=10&page=2

Authorization: myUserName aBcDeFg123456
...
```

Response

```
HTTP/2.0 200 OK

Content-Type: application/json; charset=UTF-8
Cache-Control: max-age=3600, must-revalidate
Expires: Mon, XX xxx 20XX XX:XX:XX GMT
...
```

```
{
  "pagination-count": "2",
  "pagination-page": "2",
  "pagination-limit": "8",
  "bookshelves": [
    {
     "ID": "789",
     "label": "My Label",
     "floor": "1",
     "section": "Section A",
     "available-space": "10"
    },
    ...
  ]
}
```

```
<bookshelves>
  <pagination-count>2</pagination-count>
  <pagination-page>2</pagination-page>
  <pagination-limit>8</pagination-limit>
  <bookshelf>
    <ID>789</ID>
    <label>My Label</label>
    <floor>1</floor>
    <section>Section A</section>
    <available-space>10</available-space>
  </bookshelf>
  ...
</bookshelves>
```

15. Add a bookshelf:

- End Point : `https://www.mylibrarycatalog.com:8080/bookshelves`
- URI Design: `https://www.mylibrarycatalog.com:8080/bookshelves`

Request

```
POST /bookshelves

Content-Type: application/json; charset=UTF-8
Authorization: myUserName aBcDeFg123456
...
```

```
{
  "bookshelf": {
     "ID": "789",
     "label": "My Label",
     "floor": "1",
     "section": "Section A",
     "available-space": "10"
  }
}
```

```
<new-bookshelf>
  <bookshelf>
    <ID>789</ID>
    <label>My Label</label>
    <floor>1</floor>
    <section>Section A</section>
    <available-space>10</available-space>
  </bookshelf>
</new-bookshelf>
```

Response

```
HTTP/2.0 201 Created
HTTP/2.0 400 Bad Request (if required information is missing)
HTTP/2.0 403 Forbidden (if the user doesn't have the correct permissions)

Content-Type: application/json; charset=UTF-8
Cache-Control: max-age=3600, must-revalidate
...
```

```
{
  "date": XX-XX-XXXX,
  "bookshelf": {
     ... // same as above
  }
}
```

```
<new-bookshelf>
  <date>XX-XX-XXXX</date>
  <bookshelf>
    ... <!-- same as above -->
  </bookshelf>
<new-bookshelf>
```

16. Update a bookshelf:

- End Point : `https://www.mylibrarycatalog.com:8080/bookshelves/<id>`
- URI Design: `https://www.mylibrarycatalog.com:8080/bookshelves/{number}`

Request

```
PUT /bookshelves/789

Content-Type: application/json; charset=UTF-8
Authorization: myUserName aBcDeFg123456
...
```

```
{
  "bookshelf": {
     "label": "New Label"
  }
}
```

```
<update-bookshelf>
  <bookshelf>
    <label>New Label</label>
  </bookshelf>
</update-bookshelf>
```

Response

```
HTTP/2.0 200 OK
HTTP/2.0 403 Forbidden (if the user doesn't have the correct permissions)

Content-Type: application/json; charset=UTF-8
Cache-Control: max-age=3600, must-revalidate
...
```

```
{
  "bookshelf": {
     "ID": "789",
     "label": "New Label",
     "floor": "1",
     "section": "Section A",
     "available-space": "10"
  }
}
```

```
<new-bookshelf>
  <bookshelf>
    <ID>789</ID>
    <label>New Label</label>
    <floor>1</floor>
    <section>Section A</section>
    <available-space>10</available-space>
  </bookshelf>
</new-bookshelf>
```

17. Delete a bookshelf:

- End Point : `https://www.mylibrarycatalog.com:8080/bookshelves/<id>`
- URI Design: `https://www.mylibrarycatalog.com:8080/bookshelves/{number}`

Request

```
DELETE /bookshelves/123

Content-Type: application/json; charset=UTF-8
Authorization: myUserName aBcDeFg123456
...
```

Response

```
HTTP/2.0 404 Not Found
HTTP/2.0 403 Forbidden (if the user doesn't have the correct permissions)

Content-Type: application/json; charset=UTF-8
Cache-Control: max-age=3600, must-revalidate
...
```

```
{
  "date": XX-XX-XXXX,
  "book": {
     "ID": "789"
  }
}
```

```
<delete-bookshelf>
  <date>XX-XX-XXXX</date>
  <bookshelf>
    <ID>789</ID>
  </bookshelf>
<delete-bookshelf>
```