Library REST application.

This application represents library management system, which could by used by librarians to track all necessary information about library.

Requirements for system could be divided in several sections.

1. Book management requirements:
- get information about all books (using filters, pagination, order)
- get information about one book
- add information about book
- edit information about book
- delete information about book

2. Author management requirements:
- get information about all authors (using filters, pagination, order)
- get information about one author
- add information about author
- edit information about author
- delete information about author

3. Category management requirements:
- get information about all categories (using filters, pagination, order)
- get information about one category
- add information about category
- edit information about category
- delete information about category

4. User (librarian) management requirements:
- login to application
- get information about all users (using filters, pagination, order, only admin)
- get information about one user
- add user to system (only admin)
- edit user information (only admin)
- delete user from system (only admin)

5. Client (visitor) management requirements
- get information about all clients (using filters, pagination, order)
- get information about client
- add client to system
- edit client information
- delete client from system
- assign book to client
- accept return of book from client

Other requirements:
- system should be available 24/7
- all personal information should be protected with encryption
- answer to request should be less than 1 second.

Model.

Entities

Person – abstract base class for person
Integer id – unique id
String firstname
String lastname

User extends Person – class for user of system (librarian)
String login
String password (should be encoded)
Role role

Client extends Person
Integer age;
LocalDate dateOfRegistration
List <Books> books – current books taken

Role (enum) – roles of user
LIBRARIAN, ADMIN

Book – entity for books
String title
Author author
Integer yearOfPublication
Client currentlyTakeBy
Category category

Author extends Person – entity for authors of Books

Category– books category
String name;
Boolean allowedForTakingHome;
Boolean allowedForUnderage;

REST endpoints.

All of endpoints requires authentication with JWT.

Book

Method	URL	Description	Query params	Response	Response codes
GET	/book/all	Get information about all books. Cached.	Optional: page, order, authorLastName, authorFirstName, yearOfPublication, avalability	List <Book>	200 204 401
GET	/book/{id}	Get information about one book. Cached.	-	Book	200 204 401 404
POST	/book/add	Add information about one book. Method could return 409 in case of duplicate.	JSON body	Book (created)	201 400 401 409
PUT	/book/edit/{id}	Edit information about one book. Method could return 409 in case of duplicate.	JSON body	Book (updated)	200 400 401 404 409
DELETE	/book/delete/{id}	Delete information about one book. Method could return 409 in case of delete book, which is taken.	-	-	200 401 404 409

Author

Method	URL	Description	Query params	Response	Response codes
GET	/author/all	Get information about all authors. Cached.	Optional: page, order, authorLastName, authorFirstName	List <Author>	200 204 401
GET	/author/{id}	Get information about one author. Cached.	-	Author	200 204 401 404
POST	/author/add	Add information about one author. Method could return 409 in case of duplicate.	JSON body	Author (created)	201 400 401 409
PUT	/author /edit/{id}	Edit information about one author. Method could return 409 in case of duplicate.	JSON body	Author (updated)	200 400 401

					404 409
DELETE	/author/delete/{id}	Delete information about one author. Method could return 409 in case of delete author, that connected to book.	-	-	200 401 404 409

Category

Method	URL	Description	Query params	Response	Response codes
GET	/category/all	Get information about all categories. Cached.	Optional: page, order	List < Category >	200 204 401
GET	/category/{id}	Get information about one category. Cached.	-	Category	200 204 401 404
POST	/category/add	Add information about one category. Method could return 409 in case of duplicate.	JSON body	Category (created)	201 400 401 409
PUT	/category/edit/{id}	Edit information about one category. Method could return 409 in case of duplicate.	JSON body	Category (updated)	200 400 401 404 409
DELETE	/category/delete/{id}	Delete information about one category. Method could return 409 in case of delete category, that connected to book.	-	-	200 401 404 409

User

Method	URL	Description	Query params	Response	Response codes
POST	/user/login	Login to system	Login, password.	-	200
GET	/user/all	Get information about all users. Only admin. Cached.	Optional: page, order	List <User>	200 204 401 403
GET	/user/{id}	Get information about one user. Cached.	-	User	200 204 401 404
POST	/user/add	Add information about one user. Method could return 409 in case of duplicate. Only admin.	JSON body	User (created)	201 400 401 403 409
PUT	/user/edit/{id	Edit information about one	JSON body	User	200

	}	user. Method could return 409 in case of duplicate. Only admin.		(updated)	400 401 403 404 409
DELETE	/user/delete/{id}	Delete information about one user. Only admin.	-	-	200 401 403 404

Client

Method	**URL**	**Description**	**Query params**	**Response**	**Response codes**
GET	/client/all	Get information about all clients. Cached.	Optional: page, order	List <Client >	200 204 401
GET	/client/{id}	Get information about one client. Cached.	-	Client	200 204 401 404
POST	/client/add	Add information about one client. Method could return 409 in case of duplicate.	JSON body	Client (created)	201 400 401 409
PUT	/client/edit/{id}	Edit information about one client. Method could return 409 in case of duplicate.	JSON body	Client (updated)	200 400 401 404 409
DELETE	/client/delete/{id}	Delete information about one client. Method could return 409 in case of delete client, with taken book.	-	-	200 401 404 409
POST	/client/assign	Assign book to client. Method could return 404 in case of book or client not found. Method could return 409 in case of assign book which is already assigned.	clientId, bookId	Client (updated)	200 400 401 404 409
POST	/client/return	Return book from client. Method could return 404 in case of book or client not found. Method could return 400 in case of return book which is not assigned.	clientId, bookId	Client (updated)	200 400 401 404