Library

Requirements

Functional Requirements

Books

The API must provide an endpoint URL for getting the list of all books in the database
The API must provide an endpoint URL to retrieve a particular book by ID
Users must be able to add new books by specifying the book description in the body of the request
Users must be able to delete books by specifying the book ID in the endpoint URL
Users must be able to update books by specifying the book ID in the endpoint URL and the new book description in the body of the request

Authors

The API must provide an endpoint URL for getting the list of all authors in the database.
The API must provide an endpoint URL to retrieve a particular author by ID.
Users must be able to add new authors by specifying the author's description in the body of the request.
Users must be able to delete authors by specifying the author's ID in the endpoint URL.
Users must be able to update authors by specifying the author's ID in the endpoint URL and the new author's description in the body of the request.

Genres

The API must provide an endpoint URL for getting the list of all genres in the database.
The API must provide an endpoint URL to retrieve a particular genre by ID.
Users must be able to add new genres by specifying the genre description in the body of the request.
Users must be able to delete genres by specifying the genre ID in the endpoint URL.
Users must be able to update genres by specifying the genre ID in the endpoint URL and the new genre description in the body of the request.

Non-Functional Requirements

The API should respond within a reasonable time frame
The API should be intuitive and comprehensive
The API should ensure the integrity of data, making sure updates, additions, deletions are accurate

Entities Description:

- Books
- Authors
- Genres

Functions supported by the API:

- GET
- POST
- UPDATE
- DELETE

Specific endpoints of the API for the entities:

Books

library/books - Returns a list of all the books - GET

- 200 OK: Successful retrieval of the list of all books.
- 404 Not Found: No books are available in the library (empty catalog).

library/books/{id} - Returns an specific object by id if there exists one - GET

- 200 OK: Successful retrieval of the book with the specified ID.
- 404 Not Found: No book found with the provided ID.

library/books/addBook - Adds a new book to the database - POST

- 201 Created: The new book has been successfully added to the database.
- 400 Bad Request: Invalid or missing parameters in the request body.
- 409 Conflict: A book with the same ID already exists (if using ID as a unique identifier).

library/books/updateBook/{id} - Updates the information of the specified book - PUT

- 200 OK: The information of the specified book has been successfully updated.
- 400 Bad Request: Invalid or missing parameters in the request body.
- 404 Not Found: No book found with the provided ID.

library/books/deleteBook/{id} - Deletes the specified book from the library - DELETE

- 204 No Content: The specified book has been successfully deleted.
- 404 Not Found: No book found with the provided ID.

Authors

library/authors - Returns a list of all the authors - GET

- 200 OK: Successful retrieval of the list of all authors.
- 404 Not Found: No authors are available in the library (empty author list).

library/authors/{id} - Returns an specific object by id if there exists one - GET

- 200 OK: Successful retrieval of the author with the specified ID.
- 404 Not Found: No author found with the provided ID.

library/authors/addAuthors - Adds a new author to the database - POST

- 201 Created: The new author has been successfully added to the database.
- 400 Bad Request: Invalid or missing parameters in the request body.
- 409 Conflict: An author with the same ID already exists (if using ID as a unique identifier).

library/authors/updateAuthor/{id} - Updates the information of the specified author - PUT

- 200 OK: The information of the specified author has been successfully updated.
- 400 Bad Request: Invalid or missing parameters in the request body.
- 404 Not Found: No author found with the provided ID.

library/authors/deleteAuthor/{id} - Deletes the specified author from the library - DELETE

- 204 No Content: The specified author has been successfully deleted.
- 404 Not Found: No author found with the provided ID.

Genre

library/genres - Returns a list of all the genres - GET

- 200 OK: Successful retrieval of the list of all genres.
- 404 Not Found: No genres are available in the library (empty genre list).

library/genres/{id} - Returns a specific object by id if there exists one - GET

- 200 OK: Successful retrieval of the genre with the specified ID.
- 404 Not Found: No genre found with the provided ID.

library/genres/addGenre - Adds a new genre to the database - POST

- 201 Created: The new genre has been successfully added to the database.
- 400 Bad Request: Invalid or missing parameters in the request body.
- 409 Conflict: A genre with the same ID already exists (if using ID as a unique identifier).

library/genres/updateGenre/{id} - Updates the information of the specified genre - PUT

- 200 OK: The information of the specified genre has been successfully updated.
- 400 Bad Request: Invalid or missing parameters in the request body.
- 404 Not Found: No genre found with the provided ID.

library/genres/deleteGenre/{id} - Deletes the specified genre from the library - DELETE

- 204 No Content: The specified genre has been successfully deleted.
- 404 Not Found: No genre found with the provided ID.

Collections, filters and pagination

Collections of the entities are available under their library/{entity} endpoint

For further specification, we can use the filtering by specifying query strings in the endpoint URL

Filtering in books:

library/books?author=George%well

Filtering in authors:

library/authors?birthdate=”1920-%%-%%”

Filtering in genre:

library/genres?name=”Rom%”

Pagination:

We can further specify the pagination and limit on the endpoint URL by passing in the page and limit query strings:

library/books/page=1&limit=20

library/authors/page=5&limit=10

library/genres/page=3&limit=15

Errors and authentication method

When a user is registered into the application, a JWT is generated; then the client must provide the generated JWT in every request to verify their identity and proceed with the request response.
If no JWT is provided or it has expired an error message along with the 401 status code will be returned.

The token must be provided inside the headers following the next way:
Headers: {
  "Authorization": "{JWT-TOKEN}”
}

Caching
The next methods will allow caching for a certain duration

- library/books - GET
- library/authors - GET
- library/genres - GET