06 Additional API - v2

## Description

Besides the functionalities in the context of the given front-end (previous stories), additional functionalities need to be provided.

Below you find the API of these functionalities (URL, request type, input, output, acceptance criteria).

When a Book object is returned in the output, it has always following fields: “id” (long), “title” (String), “numberInStock” (int), “price” (double), “inColor” (Boolean), “priceInDollar” (double)

**Find book with given id**

|  |  |
| --- | --- |
| Request | GET |
| URL | <http://localhost:8080/api/book/search?id=4> |
| Input | Id (int) |
| Output: happy case | A Book object (JSON) |
| Output: unhappy case |  |
| Acceptance Criteria | The book with the given id is returned  I If the book does not exist: returns nothing |

**Delete book with given id**

|  |  |
| --- | --- |
| Request | DELETE |
| URL | <http://localhost:8080/api/book/remove?id=1> |
| Input | id (int) |
| Output: happy case | A Book object (JSON) |
| Output: unhappy case | If no book with given id exists: response “400 BAD REQUEST” with error (JSON) with field “id” and message “book with id {id} does not exist.”  "id": "Book with id 10 does not exist." |
| Acceptance Criteria | The book with the given id is removed from the library.  Returns the removed book if it exists |

**Find books with price and number in stock greater than**

|  |  |
| --- | --- |
| Request | GET |
| URL | [http://localhost:8080/api/book/search/price/{price}/numberInStock/{numberInStock}](http://localhost:8080/api/book/search/price/%7bprice%7d/numberInStock/%7bnumberInStock%7d) |
| Input | Price (double)  numberInStock(int) |
| Output: happy case | A JSON array of books |
| Output: unhappy case |  |
| Acceptance Criteria | Returns all books with given price and numberInStock greater than the given number.  If no books are found: returns empty list |

**Find books not in color with price less than**

|  |  |
| --- | --- |
| Request | GET |
| URL | [http://localhost:8080/api/book/search/notInColor/priceLessThan/{price}](http://localhost:8080/api/book/search/notInColor/priceLessThan/%7bprice%7d) |
| Input | Price (double) |
| Output: happy case | A JSON array of books |
| Output: unhappy case |  |
| Acceptance Criteria | Returns all books that are not in color and have price less than the given price. |

**Order books**

|  |  |
| --- | --- |
| Request | GET |
| URL | <http://localhost:8080/api/book/all/order?order=desc&column=price> |
| Input | Order: desc or asc  Column: price, numberInStock, title |
| Output: happy case | A JSON array of books |
| Output: unhappy case | If the given column does not exist: “400 BAD REQUEST” with error (JSON) with field “column” and message “Invalid column order”.  "column": "Invalid column order" |
| Acceptance Criteria | Returns all books ordered by given column in given order (descending or increasing)  The URL in the example returns all books ordered by descending price  If given order is not “desc” or “asc”, books are ordered descending. |
| Technical implementation | You can implement this request with nested if-statements or with a switch-statement (<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/switch.html>).  Or alternatively, the guide <https://www.baeldung.com/spring-data-sorting#2-sorting-with-a-sort-parameter> allows you to implement the request with one line of code. |