Test plan

## Test objectives for this iteration

The objective for this iteration is to verify and validate that the implemented functionality is working as expected as well as trying to identify as many faults as possible in the system. The idea is to build up confidence that the project so far is heading at the right direction. But since this is still development testing, the main objective is to find as many defects as possible.

## Test objects

To make sure that the correct content is rendered and presented as it should we need to test the client. Since not all features are implemented yet this iteration will focus on use cases 1 – 3. List books, view book information and delete book.

The delete book use case is dependent on a predicate function (**getAllBooksBut**) to filter the list of books. This needs to be tested to verify that it will return the right result.

The next step is to implement the functionality to add and edit books. The book data needs to be validated before being added to the system. One function (**validateTitle**) has already been implemented and needs to be tested. Another (**validatePublishDateFormat**) has not been implemented yet but to enable a test first approach there is a need to create a test for this function as well.

The API need to be tested to make sure that it fulfils its purpose and to give a clear picture of what’s been implemented.

## Testing techniques

The design of all the tests will have equivalence partition in mind to reduce the number of tests but still having as large coverage as possible.

Testing the implemented features and the client will be done using manual tests since this not only has to do with functionality but also has to do with navigating on the website and to make sure that everything is presented as it should.

Automatic unit tests will be used to test the functions to reduce the time spent on testing as well as allow for automatic regression testing. This will reduce the risk to make changes that will result in faults in the system. The tests of the implemented functions will have a decision coverage to verify the function.

The API will be tested using automatic API tests for the same reasons as the unit tests.

Using reviews on the code to find defects and to make sure that the code is following the code standard.

## Test matrix for manual test cases

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Req. 1. UC 1.1 | Req. 1. UC 1.2 | Req. 1. UC 2.1 | Req. 1. UC 3.1 | Req. 1. UC 4.1 | Req. 1. UC 4.2 | Req. 1. UC 5.1 | Req. 1. UC 5.2 | Req. 1. UC 6.1 | Req. 1. UC 6.2 |
| reqs. Tested | 1 | 1 | 2 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1.1.1 | x |  |  |  |  |  |  |  |  |  |
| 1.2.1 |  | x |  |  |  |  |  |  |  |  |
| 2.1.1 |  |  | x |  |  |  |  |  |  |  |
| 2.1.2 |  |  | x |  |  |  |  |  |  |  |
| 3.1.1 |  |  |  | x |  |  |  |  |  |  |
| 3.1.2 |  |  |  | x |  |  |  |  |  |  |
| 3.1.3 |  |  |  | x |  |  |  |  |  |  |

## Manual test cases

1.1.1. List existing books

To be able to show the available books in the system all of the books that are in books.xml should be presented in a list. The title and author of each books should be shown in the list.

#### Pre-conditions

The content in **books.xml** must be replaced with the content in **test/data/111books.xml**.   
The server must be running.

#### Steps

1. Go to <http://localhost:9090/>.
2. Click on the **Books** button in the menu.

**Expected output**

|  |  |  |
| --- | --- | --- |
|  | Pass | Fail |
| 4 books are listed | **x** |  |
| Each book has a **Title** and **Author** | **x** |  |
| All book information corresponds to the items in **books.xml** | **x** |  |
| There is a **New book** button | **x** |  |

**Comments**

The information for all 4 books matches the one in the XML and there is a button for adding new books.

1.2.1. List books without any saved books

This is a test to make sure that an error doesn’t occur and that the right information is shown in case the system has no books.

**Pre-conditions**

The content in **books.xml** must be replaced with the content in **test/data/121books.xml**.  
The server must be running.

**Steps**

1. Go to <http://localhost:9090/>.
2. Click on **Books** in the menu.

**Expected output**

|  |  |  |
| --- | --- | --- |
|  | Pass | Fail |
| No books are listed | **x** |  |
| There is a **New book** button | **x** |  |
| There is a text saying **No books available.** |  | **x** |

**Comments**

As expected there were no books listed. But instead of having the information message,the list headline with **Title** and **Author** is visible. It seems as if the client is trying to render the empty book list instead of presenting the message.

### **2.1.1. View book by clicking on the title**

The user should be able to view the book by clicking on the title. This will present all the information about the book as well as the possibility to alter the book.

#### Pre-conditions

The test [**1.1.1.** **List** **existing books**](#T) must have been performed.

#### Steps

1. Click on the title of the first element in the list **A Brief History of Time**.

**Expected output**

|  |  |  |
| --- | --- | --- |
|  | Pass | Fail |
| **Title** is **A Brief History of Time** | **x** |  |
| **Author** is **Stephen Hawking** | **x** |  |
| **Genre** is **Science** | **x** |  |
| **Price** is **199** | **x** |  |
| **Publish date** is **1988-09-01** | **x** |  |
| **Description** is **Hawking attempts to explain** […] **the beginning of time.** | **x** |  |
| There is a **Back** button | **x** |  |
| There is a **Save** button | **x** |  |
| There is a **Delete** button | **x** |  |

**Comments**

All information matches and everything is working as expected.

### **2.1.2 View book by clicking on the author**

The user should be able to view the book by clicking on the Author. This will present all the information about the book as well as the possibility to alter the book.

#### Pre-conditions

The test [**1.1.1.** **List** **existing books**](#T) must have been performed.

#### Steps

#### Click on the author of the second element in the list Astrid Lindgren.

**Expected output**

|  |  |  |
| --- | --- | --- |
|  | Pass | Fail |
| **Title** is **Ronia the Robbers Daughter** | **x** |  |
| **Author** is **Astrid Lindgren** | **x** |  |
| **Genre** is **Fantasy** | **x** |  |
| **Price** is **79** | **x** |  |
| **Publish date** is **1981-04-23** | **x** |  |
| **Description** is **Ronia is a girl growing** […] **the two bands.** | **x** |  |
| There is a **Back** button | **x** |  |
| There is a **Save** button | **x** |  |
| There is a **Delete** button | **x** |  |

**Comments**

Everything passed and the rendered information is the same as the expected result.

### **3.1.1. Delete book**

The user has to be able to remove books from the system. This test will make sure that it’s possible.

#### Pre-conditions

The test [**2.1.1.** **View book by clicking on the title**](#_2.1.1._View_book) must have been performed.

#### Steps

1. Click on the **Delete** button

**Expected output**

|  |  |  |
| --- | --- | --- |
|  | Pass | Fail |
| Got redirected to **http://localhost:9090/#/books** | **x** |  |
| There are 3 books listed | **x** |  |
| **A Brief History of Time** is not in the list | **x** |  |
| There is a **New book** button | **x** |  |

**Comments**

Everything went as expected. The Brief History of Time is not in the list and the remaining books are listed after the page redirected.

### **3.1.2. Delete a second book**

The user has to be able to remove books from the system. This test will make sure that it’s possible.

#### Pre-conditions

The test [**3.1.1. Delete book**](#_3.1.1._Delete_book) must have been performed.

#### Steps

1. Click on the author of the first element in the list **Astrid Lindgren**.
2. Click on the **Delete** button.

**Expected output**

|  |  |  |
| --- | --- | --- |
|  | Pass | Fail |
| Got redirected to **http://localhost:9090/#/books** | **x** |  |
| There are 2 books listed | **x** |  |
| **A Brief History of Time** is not in the list | **x** |  |
| **Ronia the Robbers Daughter** is not in the list | **x** |  |
| There is a **New book** button | **x** |  |

**Comments**

Both of the books are removed and the remaining books are unaltered.

### **3.1.3. Delete the last book**

In case the user removes the last book in the system we have to make sure nothing unexpected happens.

#### Pre-conditions

The content in **books.xml** must be replaced with the content in **test/data/313books.xml**.  
The server must be running.

#### Steps

1. Click on the author of the first element in the list **Astrid Lindgren**.
2. Click on the **Delete** button.

**Expected output**

|  |  |  |
| --- | --- | --- |
|  | Pass | Fail |
| Got redirected to **http://localhost:9090/#/books** | **x** |  |
| No books are listed | **x** |  |
| There is a text saying **No books available.** |  | **x** |
| There is a **New book** button | **x** |  |

**Comments**

I got redirected and no books were available but instead of having a text saying so the list headline saying “Title” and “Author”.

## Unit tests

**Remove book resource**

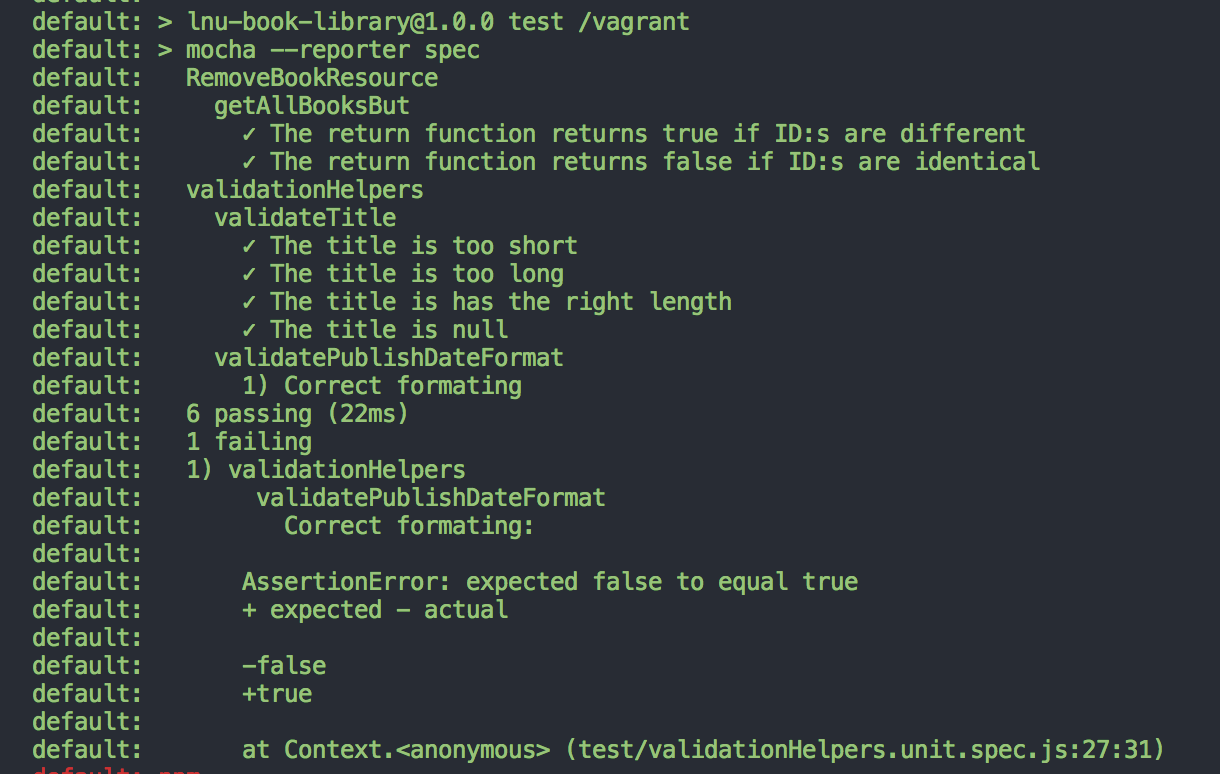
**getAllBooksBut** is a function in **RemoveBookResource**. It returns a predicate function to filter out the book with the provided ID. ­



**Validation helpers**

The public functions from **ValidationHelpers**. **validatePublishDateFormat** has not been implemented yet.

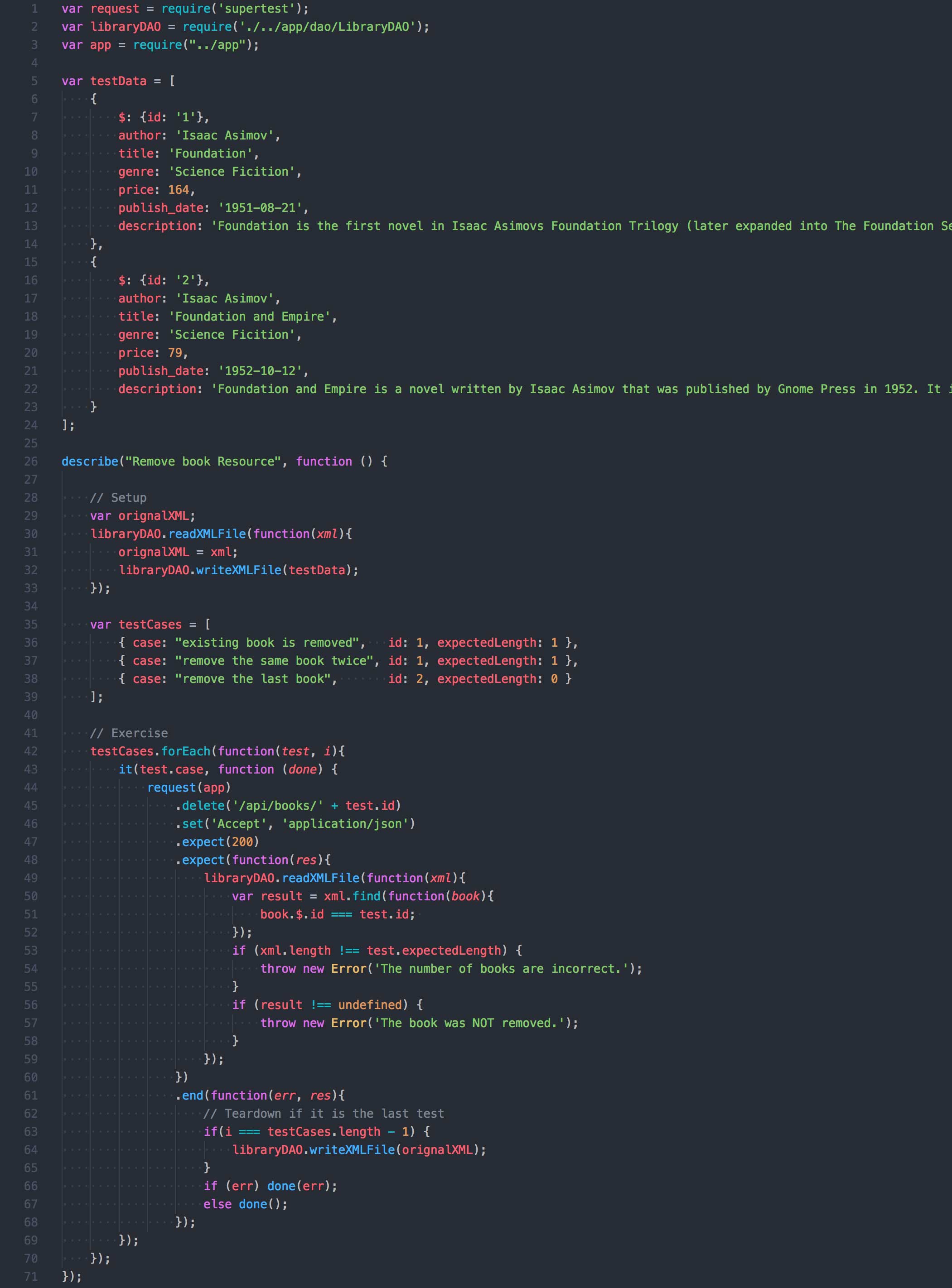
**Unit testing results**



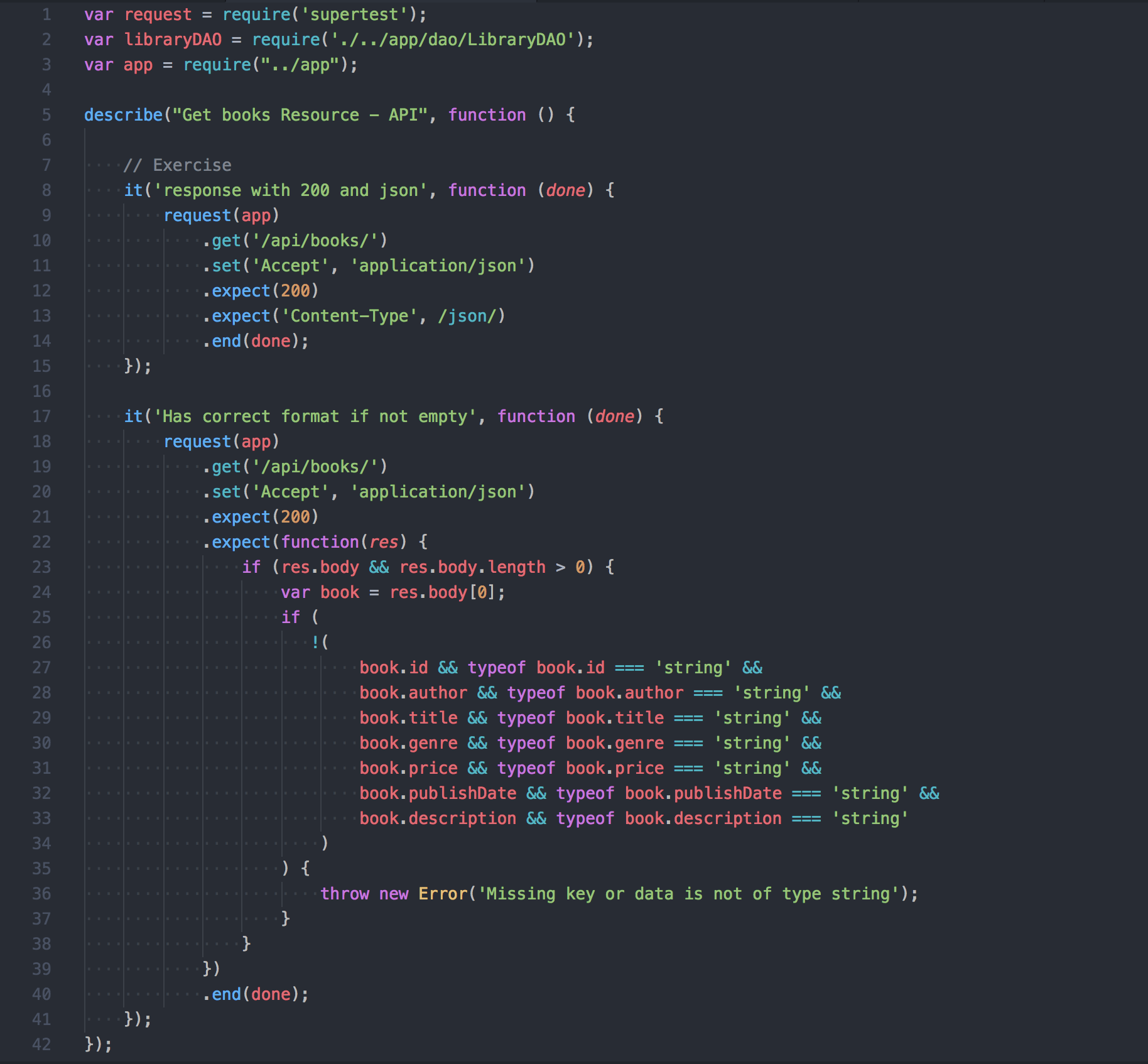
## API tests

**Remove book resource**

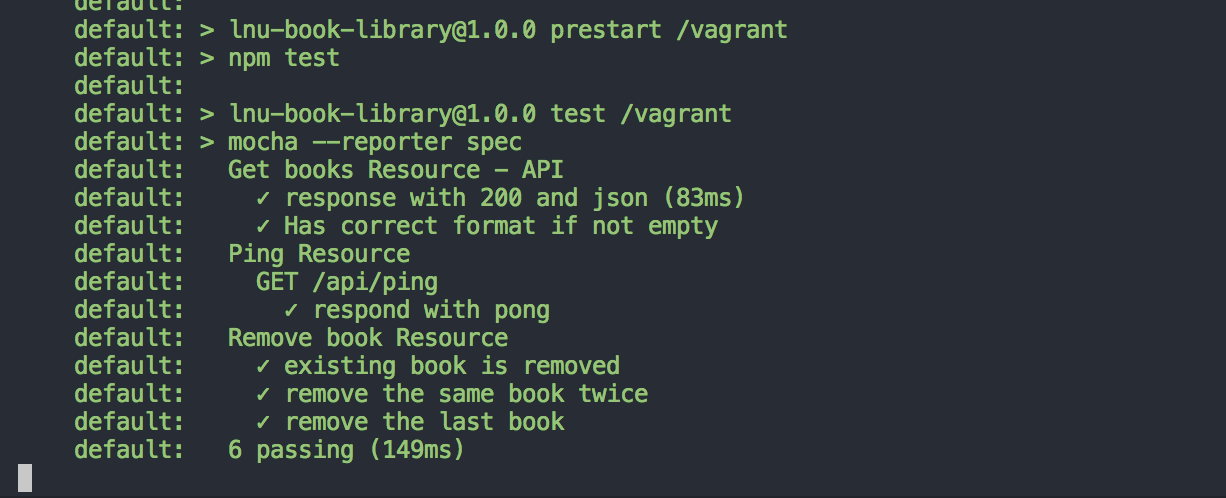
Testing to remove a book twice as well as removing the last book in the list.



**Get books resource**

Test if the response has status code 200 and is of type JSON as well as making sure that the book is formatted as it’s supposed and the data is strings. ****

**API testing results**



## Time log

|  |  |  |  |
| --- | --- | --- | --- |
| **Activity** | **Estimated time** | **Date** | **Effective time** |
| Write test plan | 180 min | 2018-02-26 | 176 min |
| Write manual test cases | 300 min | 2018-02-27 | 283 min |
| Create unit tests | 120 min | 2018-03-01 | 73 min |
| Create automated API tests | 300 min | 2018-03-04 | 247 min |
| Write reflections | 15 min | 2018-03-05 | 28 min |

## Reflections

My plan, in the beginning, was to create more API tests to get a better coverage. The best solution would have been to use the API to control the test but since I haven’t implemented the functionality to add books yet this wasn't really possible. Instead, I had to override the XML as I do in the Remove Book Resource test but when the tests are in different modules it becomes hard to control the synchronized order. A solution to this is to have all the API test in one module but that can easily become quite messy. Therefore, I decided to wait with the rest of the test until "add books" is implemented.