

# Book-Directory

## 1. Introduction

For my project I would like to implement a simple idea, so that I can learn the best way possible. I think of doing a book directory, with a small list of books and authors. Maybe the Website would also include some Bestseller lists and the best books of the week. It could be a good starting point for learning about NodeJS and MongoDB and express.

Some features would be:

- A home page
- Every user has a list of books he likes, that only he can see. He can add, delete and modify the books
- Depending on which book you click, it will show you its details.
- MongoDB database

```
var bookSchema = new Schema({
  title: {
    type: string,
    required: true
  },
  author: {
    type: String,
    required: true
  },
  image: {
    type: String,
    required: true
  },
  description: {
    type: string,
    required: true
  },
  publishingCompany: {
    type: string,
    required: false
  },
  year: {
    type: string,
    required: true
  },
  isbn: {
    type: Number,
    required: true,
    unique: true
  }
}, {
  timestamps: true
});
```

## 2. Design and Implementation

First I created two simple html files: "index.html" and "favoritelist.html". In this honor I won't concentrate on the Design as in the last two honors, because this is about the server-side development.

I went on by generating an Express application using the express-generator ("honorsServer"). I then implemented a REST API and created a "bookRouter" in my "routes" folder within the Express application and included it in the app.js.

I created a schema for the book.

Furthermore I set up the MongoDB database and connected to it. In the "bookRouter" file I then handle all my requests. As mentioned before the operations: GET, POST and DELETE are supported on "/books". This is important to add and delete books.

Additionally I added a db.json file, which contains sample books to add.

I also decided to add user authentication with passport. So now it supports signup, login and logout.

Down below we can see, how a user is successfully signed up. A logged in user can now add, delete and get the book list. As soon as he signs out, he is not authorized anymore to GET, POST or DELETE.

POST localhost:3000/users/signup

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```
1 {"username": "user", "password": "password"}
```

Body Cookies (1) Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true,
3   "status": "Registration Successful!"
4 }
```

The user can now also log in successfully

POST localhost:3000/users/login

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```
1 {"username": "user", "password": "password"}
```

Body Cookies (1) Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true,
3   "status": "You are successfully logged in!"
4 }
```

And of course logging out is also possible. Then the user gets redirected successfully to the main page

GETlocalhost:3000/users/logout

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

BodyCookiesHeaders (7)Test Results

PrettyRawPreviewVisualize

HTML

1

!DOCTYPE html

2

<html>

3

4

<head>

5

<title>Express</title>

6

<link rel="stylesheet" href="/stylesheets/style.css">

7

</head>

8

9

<body>

10

<h1>Express</h1>

11

<p>Welcome to Express</p>

12

</body>

13

14

</html>

As a logged in user it is also possible to add books, as mentioned:

POST localhost:3000/books

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
6 ..... "publishingCompany": "Atlas",
7 ..... "year": "1818",
8 ..... "isbn": "1"
9
```

Body Cookies (1) Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "_id": "609d3ddd84bfd06264eef854",
3   "title": "Frankenstein",
4   "author": "Mary Shelly",
5   "image": "images/frankenstein.png",
6   "description": "A book about a dead man",
7   "publishingCompany": "Atlas",
8   "year": "1818",
9   "isbn": 1,
10  "createdAt": "2021-05-13T14:55:25.728Z",
11  "updatedAt": "2021-05-13T14:55:25.728Z",
12  "__v": 0
13 }
```

Here a book was successfully added. It is also possible to add other books or delete them as the user wishes.

### 3. Conclusion

I already stated the features and the results I obtained. They can be seen above in the screenshots. I would have done differently the following things: it was very difficult to decide how I want to do the authentication. I basically knew that I would have to do some sort of authentication, because it is a book directory, where users can collect their books and there is no sense it it to be public. There are many ways to authenticate users, I think I chose one of the easier ones.

### 4. References

- <http://www.readprint.com/>
- <https://www.coursera.org/learn/server-side-nodejs/home/welcome>
- <https://www.goodreads.com/>