**Предмет: Разработка на клиент-сървър (fullstack) приложения с Node.js + Express.js + React.js**

***Летен семестър, 2019/2020 год.***

# BookWorld

# Документация

# Курсова работа

*Автор:*

*Йоана Григорова ФН:62021*

юли, 2019

София

# 1. Задание 1

| Project name | BookWorld |
|---|---|

| 1. Short project description (Business needs and system features) |
|---|
| With so many books to read it is hard to choose your next one and even harder to know if whatever you choose will be good to read. BookWorld offers you to compose your reading list, see others' reviews and express your opinion on books you have already read. The system will be developed as a *Single Page Application (SPA)* using ***React.js*** as front-end, and ***Node.js + express*** as backend technologies. Each view will have a distinct URL, and the routing between pages will be done client side using ***React Router***. The backend will be implemented as a ***REST/JSON API*** using JSON data serialization. The main user roles (actors in UML) are:<br>• *Anonymous User* – can only view the registration and login page<br>• *User* (extends *Registered User*)– can see the catalog of books and choose to add them to "Favourites", "Want to read" and "Read", can leave a comment and a rating of every book.<br>• *Administrator* (extends *Registered User*) – can add new books to the catalog and can manage the comment sections and remove comment if needed |

| 2. Main Use Cases / Scenarios | | |
|---|---|---|
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| **2.1. Register** | *Anonymous User* can register in the system by providing a valid e-mail address, first and last name, and choosing password. | *Anonymous User* |
| **2.2. Browse the catalogs** | *Registered User* can view the catalog<br><br>*Administrator* can view and edit the catalog | *Registered User, Administrator* |
| **2.3. Add books to personal library** | *Users* can choose to add books to their "Favourites", "Want to read" or "Read" libraries | *Registered User* |
| **2.4. Add a comment to a book** | *Users* can share their likes and dislikes about a book in the comment section | *Registered User* |
| **2.5. Search for books** | *Users* can use a search bar to look for a book by using the whole name or part of the name of a book | *Registered User* |
| **2.6. Filter the catalog by category** | *Users* can choose to see books by category | *Registered User* |
| **2.7. Manage catalog** | *Administrators* can add and edit books. | *Administrator* |

| | | |
|---|---|---|
| **2.8. Manage comments** | *Administrators* can choose to delete a comment if it breaks the rules of the site. | *Administrator* |
| **2.9 Rate book** | *Users* can rate books from 1 to 5 stars | *Registered User* |
| **2.10 Recommend Book** | *Users* can recommend books to other users | *Registered User* |

| 3. Main Views (SPA Frontend) | | |
|---|---|---|
| **View name** | **Brief Descriptions** | **URI** |
| **3.1. Home** | Presents a small selection of books | / |
| **3.2. Catalog** | Presents all books and allows users to add a book to their library without changing the page and to select a category of books to be shown | */catalog* |
| **3.3. Book** | Presents a book with all available details and all the comments | */book/{id}* |
| **3.4. Favourites** | Presents all books that the current user have added to their “Favourites” library | */favourites* |
| **3.5. Wish list** | Presents all books that the current user have added to their “Want to read” library | */wishlist* |
| **3.6. Read** | Presents all books that the current user have added to their “Read” library | */read* |
| **3.7. Registration** | Presents a view allowing the *Anonymous Users* to register in the system | */registration* |
| **3.8. Login** | Presents a view allowing the users to login. | */login* |

| 4. API Resources (Node.js Backend) | | |
|---|---|---|
| **View name** | **Brief Descriptions** | **URI** |
| **4.1. Users** | GET *User Data* for all users, and POST new *User Data*. Available only for *Administrators*. | */api/users/* |
| **4.2. User** | GET, PUT, DELETE *User Data* for *User* with specified *userId* | */api/users/{userId}* |
| **4.3. Login** | POST *User Credentials* (e-mail address and password) | */api/login* |
| **4.4. Logout** | POST a logout request for ending the active session | */api/signOut* |
| **4.5. Books** | GET, POST *Books* | */api/books/* |
| **4.6. Book** | GET, PUT *Book* with specified bookId | */api/books/{bookId}* |
| **4.7. Comment** | GET, PUT, POST, DELETE *Comment* for a specific book | */api/comment* |

| | | |
|---|---|---|
| **4.8. Favourites** | GET, POST, DELETE books from favourites library | */api/favourites* |
| **4.9. Want to read** | GET, POST, DELETE books from “Wish list” library | */api/wishList* |
| **4.10. Read** | GET, POST, DELETE books from “Read” library | */api/read* |
| **4.11. Notifications** | POST,GET,PUT notifications | */api/notifications* |

# 2. Конфигурация

a. **База данни**
   За база данни се използва MongoDB и се хоства в https://mlab.com/. Не са необходими допълнителни инсталации, за да се използва, само инсталация на модулите в папката back-end.

b. **Сървър**
   Сървърът е създаден с nodeJS и е направен като API, като основната логика е изнесена в интерфейса.

c. **Потребителският интерфейс**
   Потребителският интерфейс е направен с помощта на React/Redux.

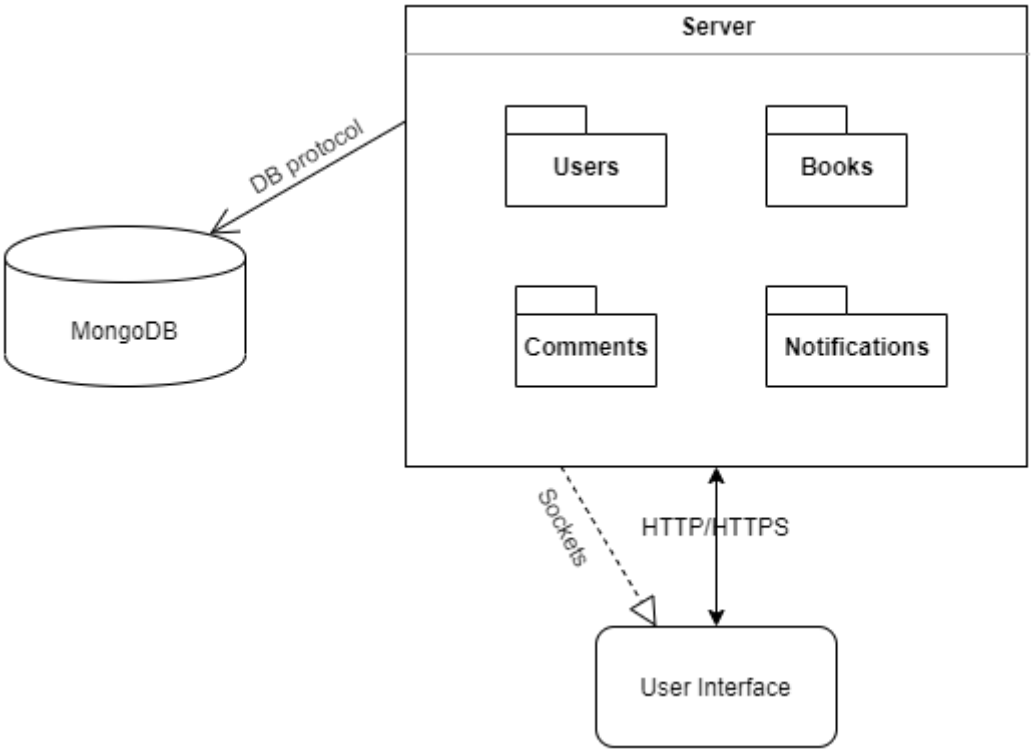
# 3. Архитектура

Сървърът е направен като API, който прави, обновява и взема записи от базата данни и ги сервира на интерфейса под формата на JSON файлове. Сървърът се свързва с интерфейса чрез HTTP/HTTPS протоколи, но също така използва и сокети за изпращане на съобщения (notifications) и обновяване на информацията, създадена от други потребители.

Сървърът е разделен на отделни модули: Users, Books, Comments, Notifications, с цел чистота на кода и пътищата. Всеки модул се грижи за обработката на информацията в своя колекция в базата данни.

Базата данни има 5 колекции: users, comments, books, notifications, sessions, и се хоства в https://mlab.com/. Колекцията sessions запазва сесиите на потребителите и ги изтрива, когато времето им изтече, след около 7 дни.

Потребителският интерфейс е направен с помощта на React/Redux. Компонентите са разделени в отделни папки. Във файла actions/index.js се намират всички заявки към сървъра, а reducers/index.js съдържа глобалния стейт на приложението.




# 4. Инсталация и билдване

Стъпки за инсталация и билдване на проекта:

- Клониране на проекта:
  git clone https://github.com/yoanagrigorova/BookWorld.git
- Пускане на сървъра
  - Влезте в папка back-end
  - Инсталирайте модулите: npm i
  - Стартиране на сървъра: node index.js
- Пускане на интерфейса
  - Влезте в папка front-end
  - Инсталирайте модулите: npm i
  - Стартиране на интерфейса: npm start

# 5. Реализация

Използваните технологии за реализацията са JavaScript, React, Redux, CSS, Materialize за потребителския интерфейс и NodeJS, Express за сървъра.
Основни решения:

- Анонимните потребители имат достъп само до страниците за вход и регистрация в системата.
- Регистрираните потребители имат достъп до каталог от книги, където могат да избират книги, които да добавят в своите колекции: Любими, Искам да прочета и Прочетени. Също така могат да добавят коментари към книгите и да дадат оценка на книгата.

- Регистрираните потребители могат да препоръчват книги на други потребители.
- Админа има достъп до каталога и може да добавя книги и да управлява коментарите.

# 6. Заключение и литература

Платформата има за цел да предостави изчистен интерфейс, служещ като лична виртуална библиотека, но също така предлага и комуникация с останалите читатели чрез коментари и препоръки за книги.

Литература:
[1] Redux API Example
[2] Materialize Library Documentation
[3] How to MAnage Sessions With NodeJS and Express
[4] Intro to Express.js: Simple REST API app with Monk and MongoDB
[5] React Redux Tutorial for Beginners
[6] Express routing