**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

# Кафедра МОЭВМ

**ОТЧЕТ**

# по лабораторной работе №2

**по дисциплине «Web-технологии» Тема: REST-приложение управления библиотекой**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 1303 |  | Новак П.И. |
| Преподаватель |  | Беляев С.А. |

Санкт-Петербург 2023

# Цель работы.

изучение взаимодействия клиентского приложения с серверной частью, освоение шаблонов web-страниц, формирование навыков разработки динамических HTML-страниц, освоение принципов построения приложений с насыщенным интерфейсом пользователя.

# Задание.

* разработка интерфейса web-приложения;
* задание стилей для отображения web-приложения с учетом размера экрана (использование на компьютере, на мобильном телефоне);
* создание web-сервера на основе express;
* создание шаблонов web-страниц;
* настройка маршрутов;
* создание json-хранилища;
* обработка REST-запросов.

Необходимо создать web-приложение управления домашней библиотекой, которое предоставляет список книг, их можно отфильтровать по признакам «в наличии», «возврат просрочен», есть возможность выдать книгу для чтения и вернуть книгу. Основные требования следующие:

1. Начальное состояние библиотеки хранится в JSON-файле на сервере, текущее состояние – в переменной в памяти сервера.

2. В качестве сервера используется Node.JS с модулем express.

3. В качестве модуля управления шаблонами HTML-страниц используется pug либо ejs, все web-страницы должны быть сделаны с использованием pug либо ejs.

4. Предусмотрена страница для списка книг, в списке предусмотрена фильтрация по дате возврата и признаку «в наличии», предусмотрена возможность добавления и удаления книг. Удаление книг – с подтверждением.

5. Предусмотрена страница для карточки книги, в которой ее можно отредактировать (минимум: автор, название, дата выпуска) и выдать читателю или вернуть в библиотеку. В карточке книги должно быть очевидно: находится ли книга в библиотеке, кто ее взял (имя) и когда должен вернуть (дата).

6. Информация о читателе вводится с использованием всплывающего модального диалогового окна ().

7. Оформление страниц выполнено с использованием CSS (допустимо использование w3.css).

8. Взаимодействие между браузером и web-сервером осуществляется с использованием REST.

9. Фильтрация списка книг осуществляется с использованием AJAX-запросов.

10. Логика приложения реализована на языке JavaScript.

11. Для всех страниц web-приложения разработан макет интерфейса с использованием Figma (<https://www.figma.com/>).

12. При оформлении элементов управления используются иконки (например, Font Awesome, https://fontawesome.ru/all-icons/).

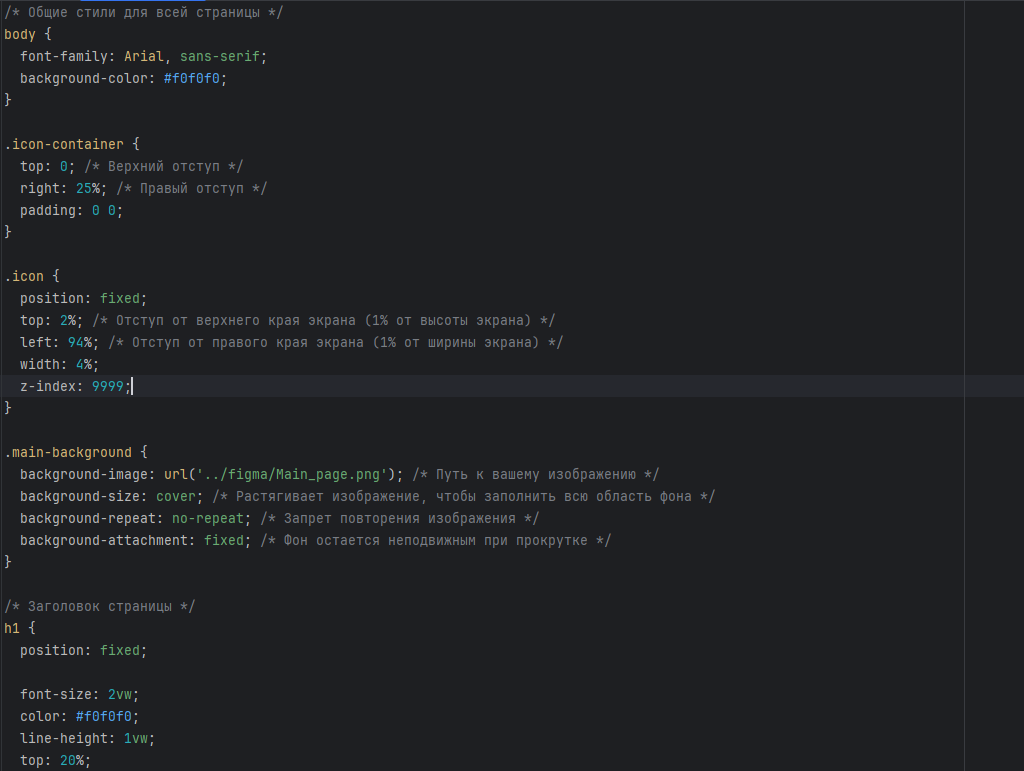
# Выполнение работы.

Изначально была настроена среда выполнения Node.JS с модулем express. Также установлен шаблон для работы с HTML pug.

# Описание функций и структур данных алгоритма.

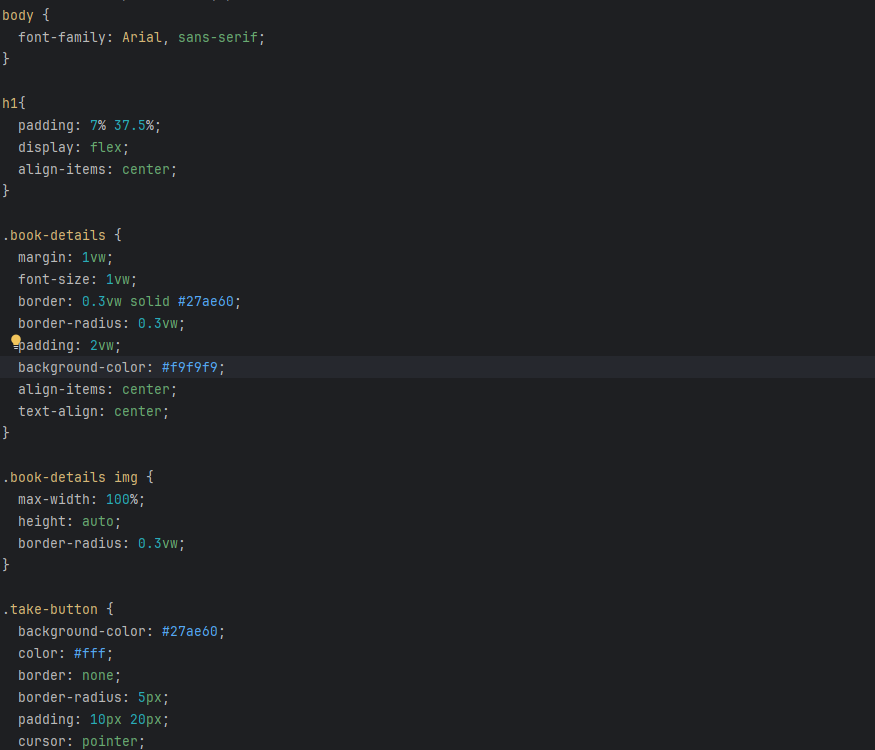
Файл style.css

В данном файле происходит определение стилей оформления для шаблона openPage.pug для главной страницы сайта. Так, например, для body устанавливается гибкость элементов(display), их выравнивание по центру(align-items).



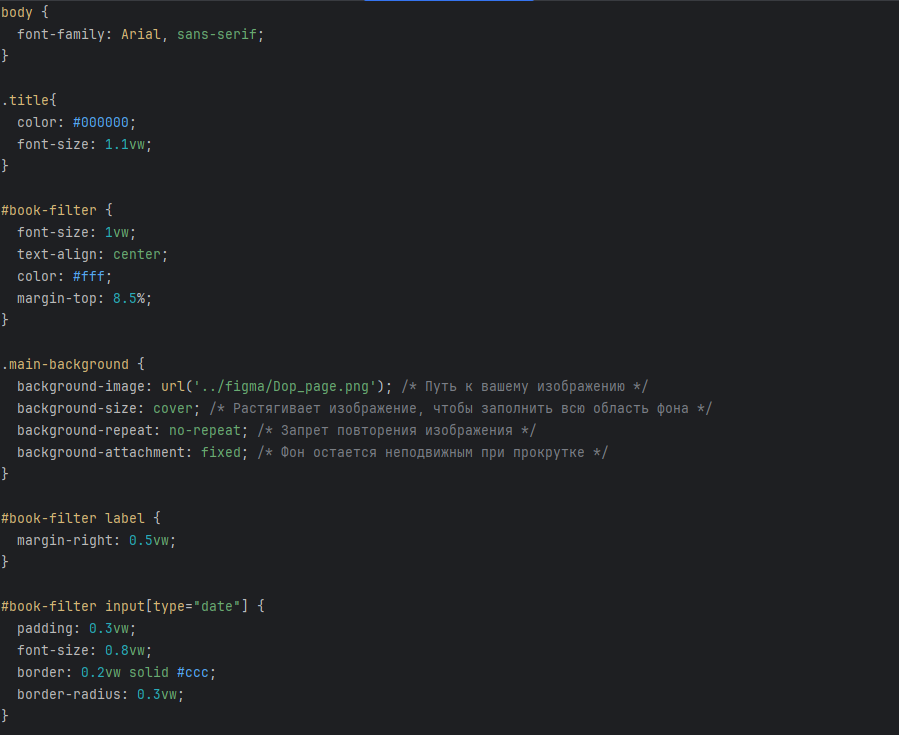
Файл style\_book.css

В данном файле происходит определение стилей оформления для шаблона book.pug для страницы с карточкой определенной книги.



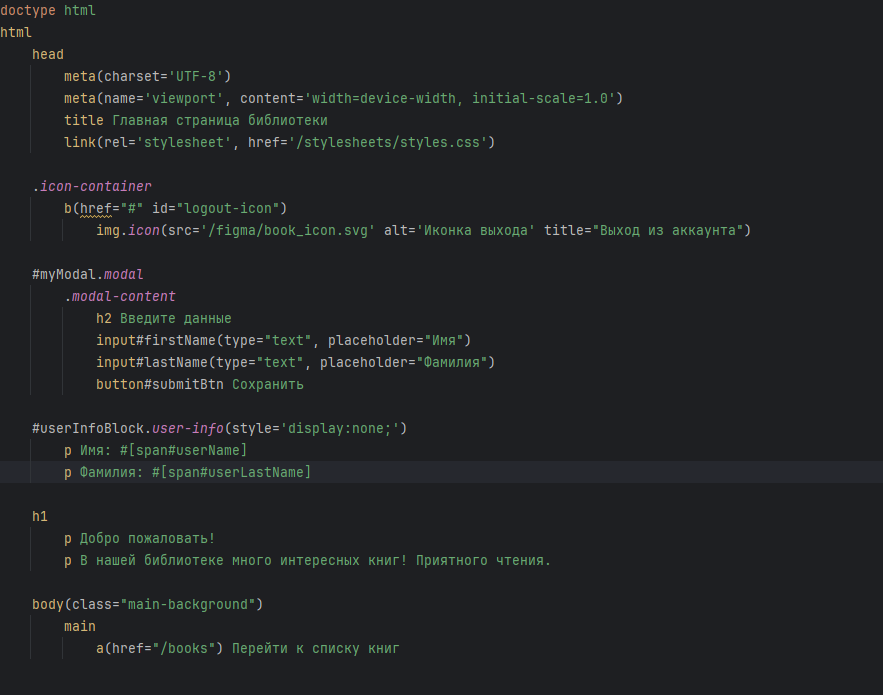
Файл style\_books.css

В данном файле происходит определение стилей оформления для шаблона books.pug для страницы со списком книг в библиотеке.



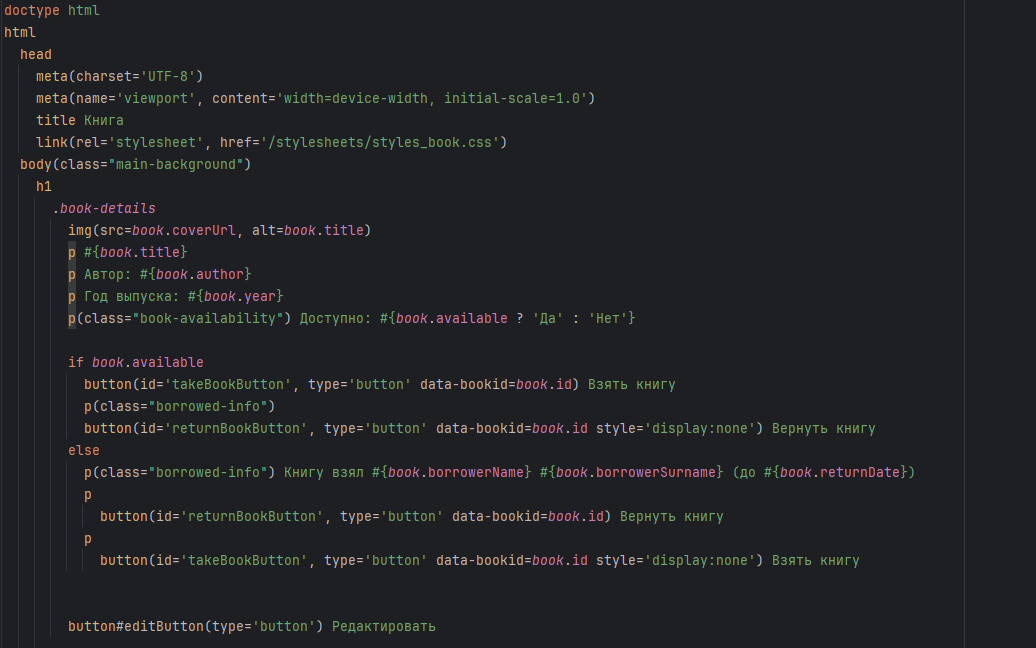
Файл openPage.pug

В данном файле написан шаблон для начальной HTML страницы сайта.



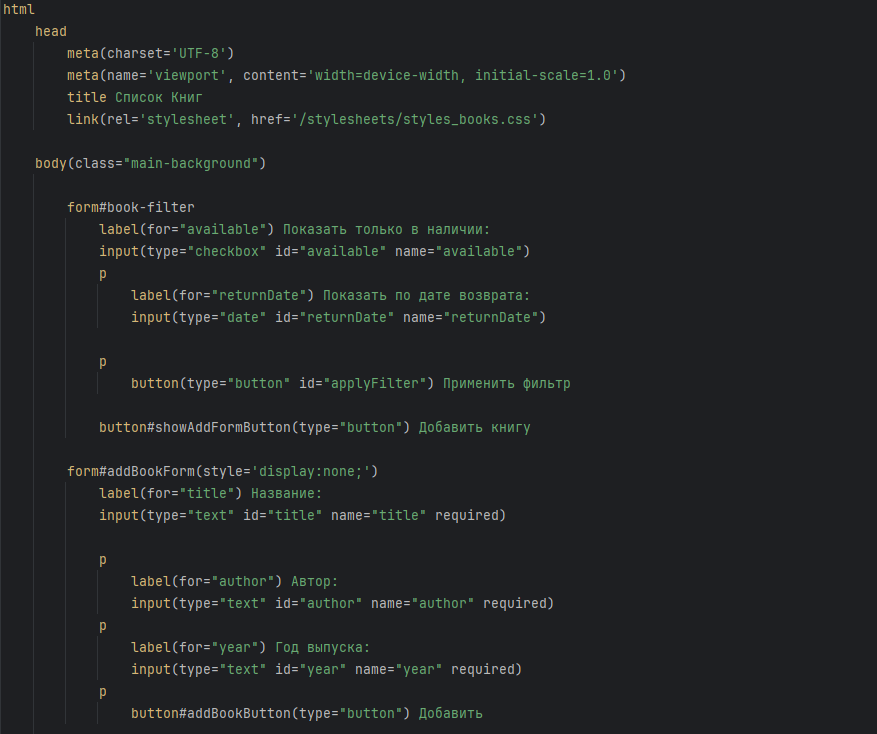
Файл book.pug

В данном файле написан шаблон для начальной HTML страницы сайта.



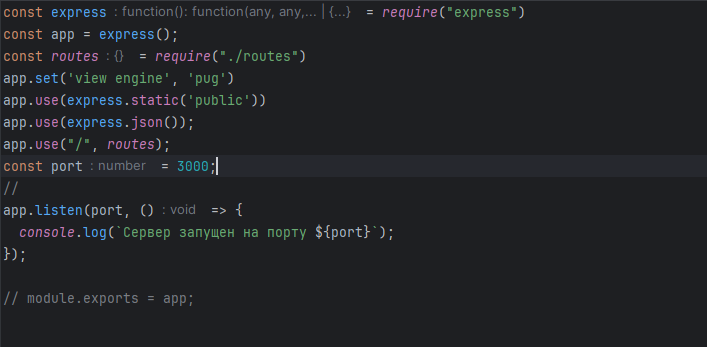
Файл books.pug

В данном файле написан шаблон для начальной HTML страницы сайта.



Файл app.js

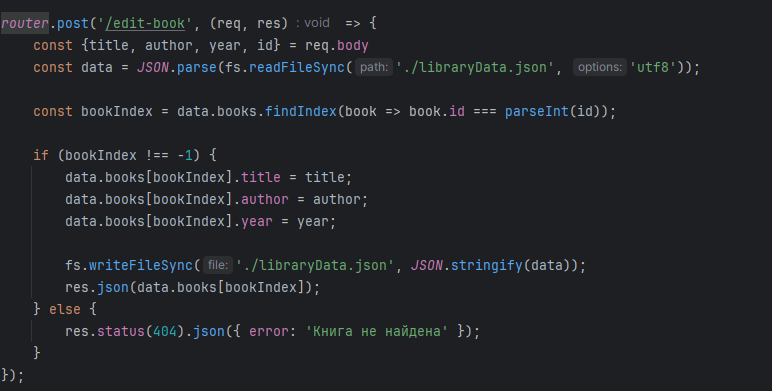
В данном классе собирается программа: устанавливается использование express, настраивается использование routes.js - файла с REST маршрутами. Также открывается порт для прослушивания.



Файл routes.js

В данном файле прописаны маршруты для перехода с одной страницы HTML на другую.

1. В router.post('/edit-book') в хранилище загружаются изменения, внесённые при редактировании книги и отправляется ответ для обновления страницы сайта.



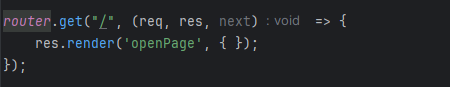
1. В router.post('/add-book') в хранилище добавляется новая книга и отправляется ответ для обновления страницы сайта.



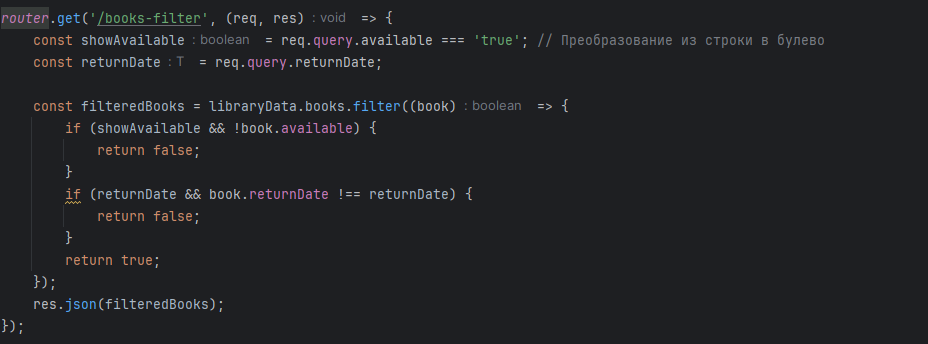
1. В router.delete("/delete-book/:bookId") из хранилища удаляется определенная книга и отправляетя ответ для её удаления со страницы сайта.



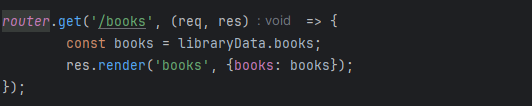
1. В router.get("/") запрашивается главная страница сайта.



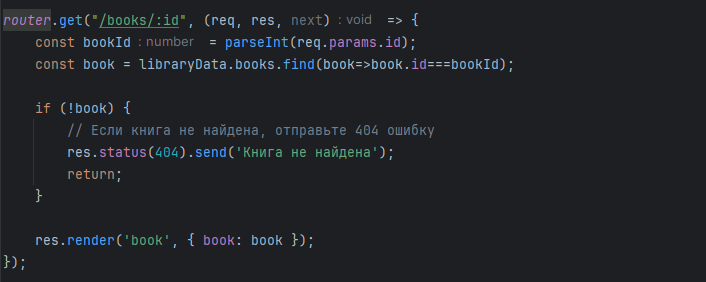
1. В router.get("/books/:id") запрашивается страница с карточкой определённой книги.



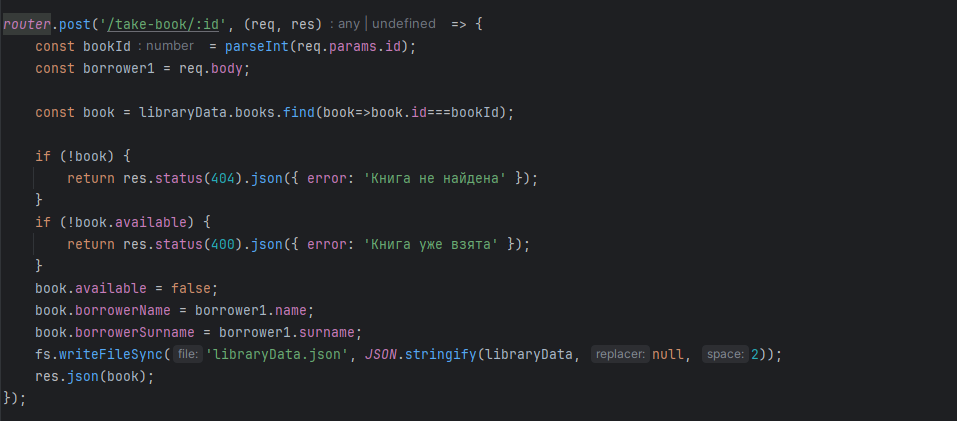
1. В router.get('/books-filter') происходит фильтрация книг по признаку и отправляется хранилище с отфильтрованными книгами .



1. В router.get('/books') запрашивается главная страница сайта.



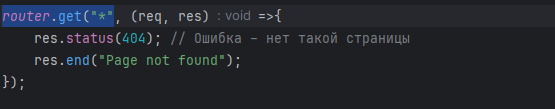
1. В router.post('/take-book/:id') определённая книга обозначается как та, которой нет в наличии, и отправляется ответ для изменения страницы сайта.



1. В router.post('/return-book/:id') определённая книга обозначается как та, которая есть в наличии, и отправляется ответ для изменения страницы сайта.



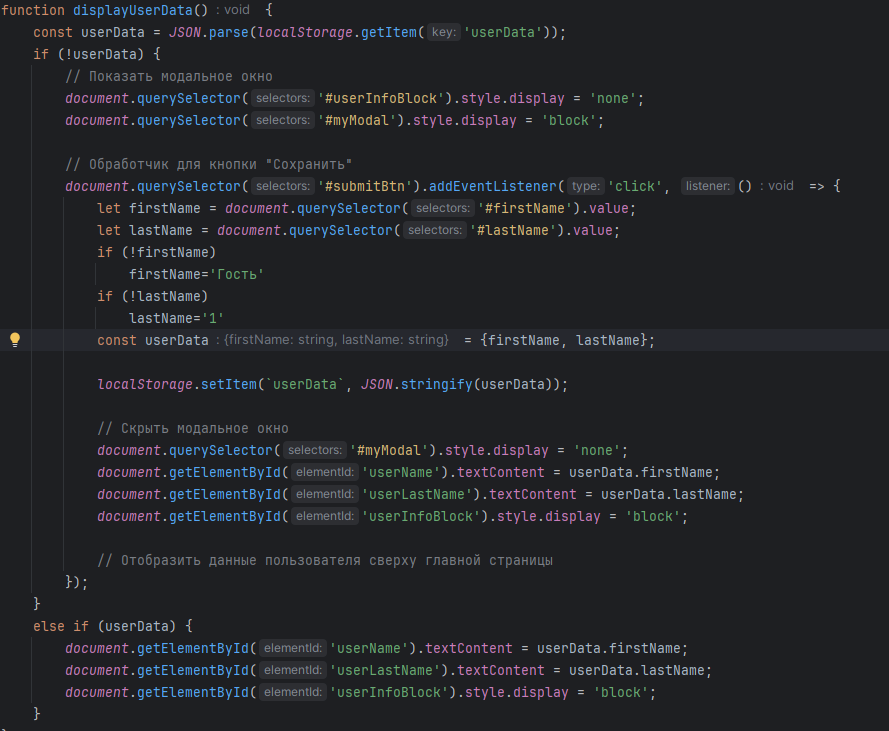
1. В router.get("\*") вылазиеот оповещение в случае, если определённая страница не найдена.



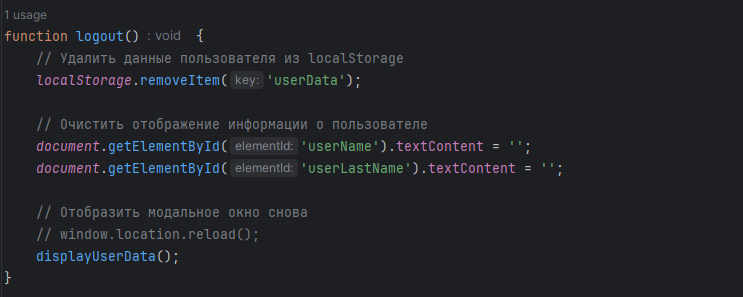
Файл script.js

В данном файле происходит javascript контроль для шаблона openPage.pug.

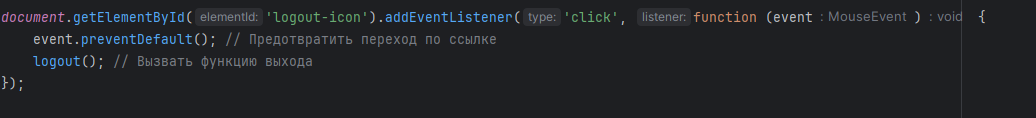
1. В функции displayUserData() происходит отображение пользовательских данных: имя и фамилия после входа на сайт.



1. В функции logout() происходит выход пользователя из системы.



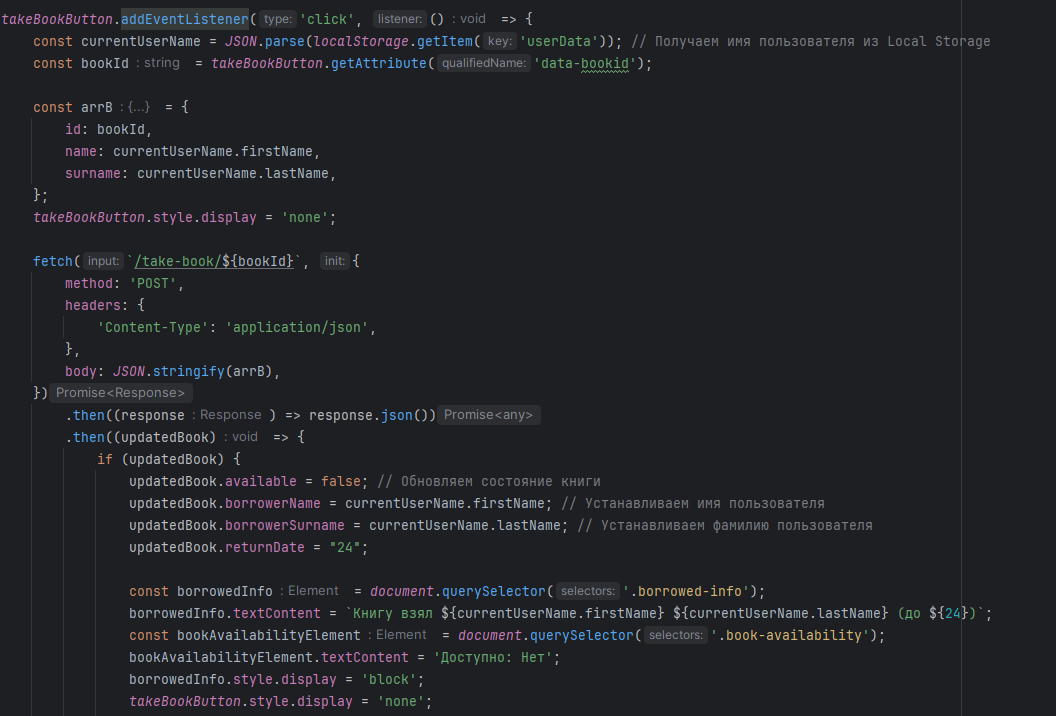
1. В функции document.getElementById('logout-icon') происходит привязка иконки к функции выхода из системы.



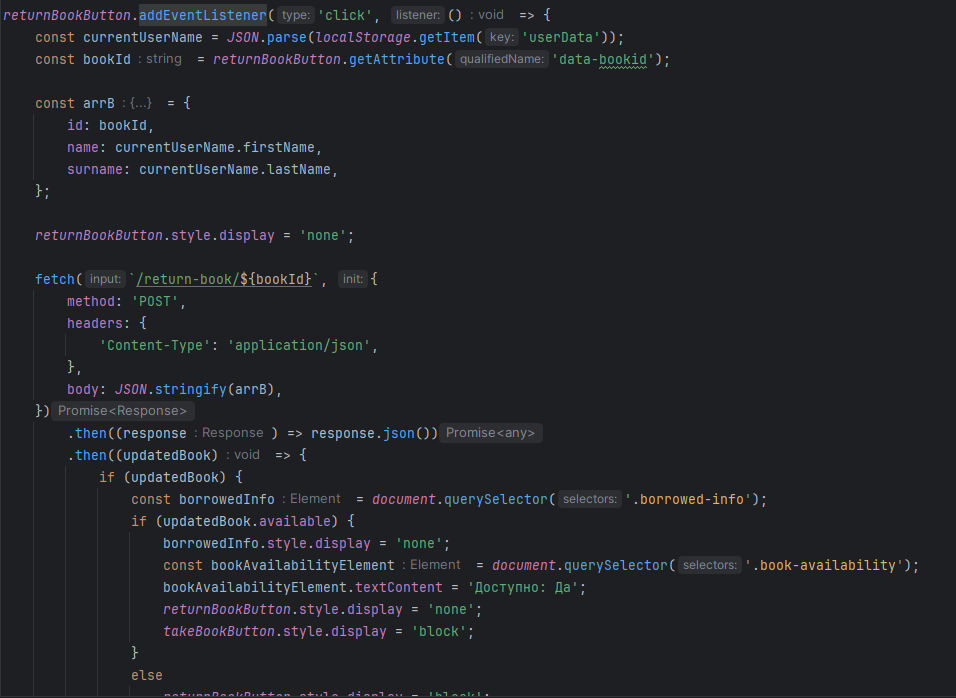
Файл script\_book.js

В данном файле происходит javascript контроль для шаблона book.pug.

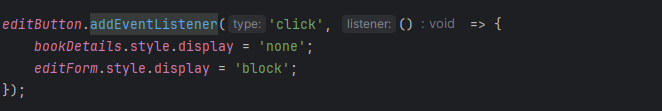
1. В функции takeBookButton.addEventListener происходит прослушивание клика на кнопку “Взять книгу”. При клике книга становится не доступна, а имя пользователя, кто её взял изменяется на имя текущего пользователя.



2. В функции returnBookButton.addEventListener происходит прослушивание клика на кнопку “Вернуть книгу”. При клике книга становится доступна, а имя пользователя, кто её взял, очищается



3. В функции editButton.addEventListener происходит прослушивание клика на кнопку “Редактировать”. При клике появляется возможность изменить название, год публикации, имя автора.



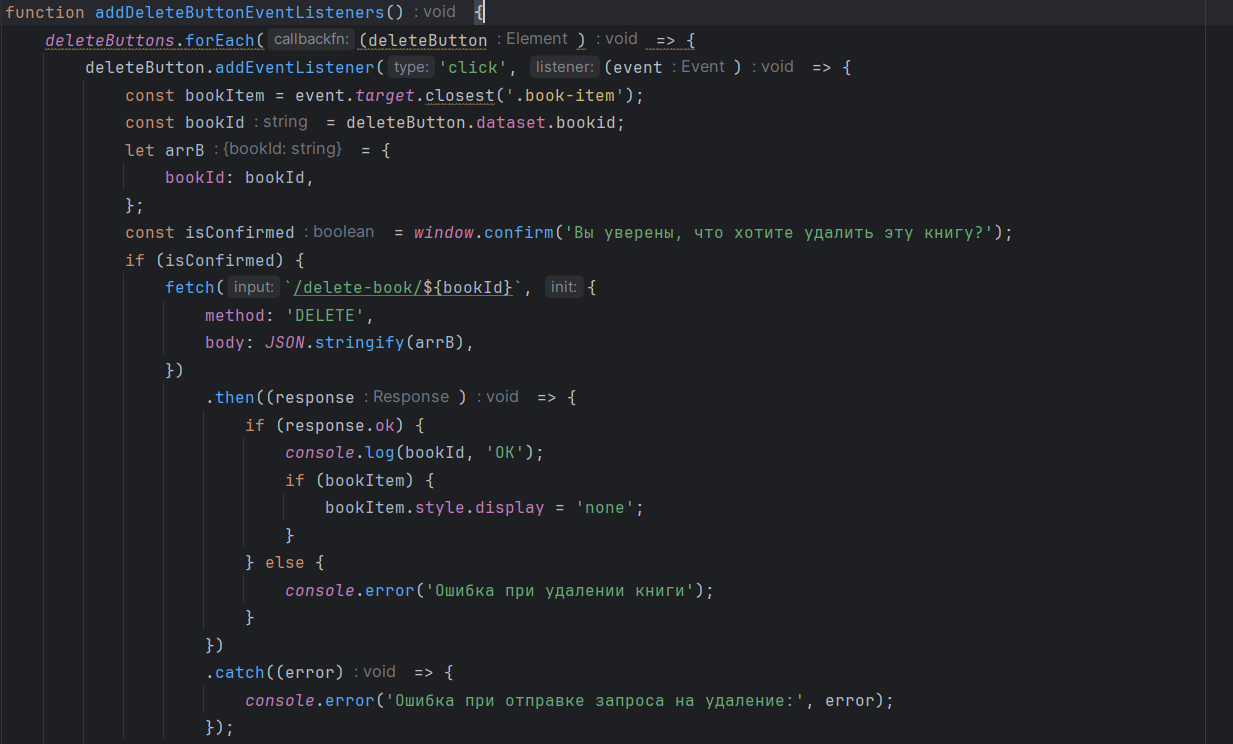
4. В функции saveEditButton.addEventListener происходит прослушивание клика на кнопку “Сохранить” при редактировании.



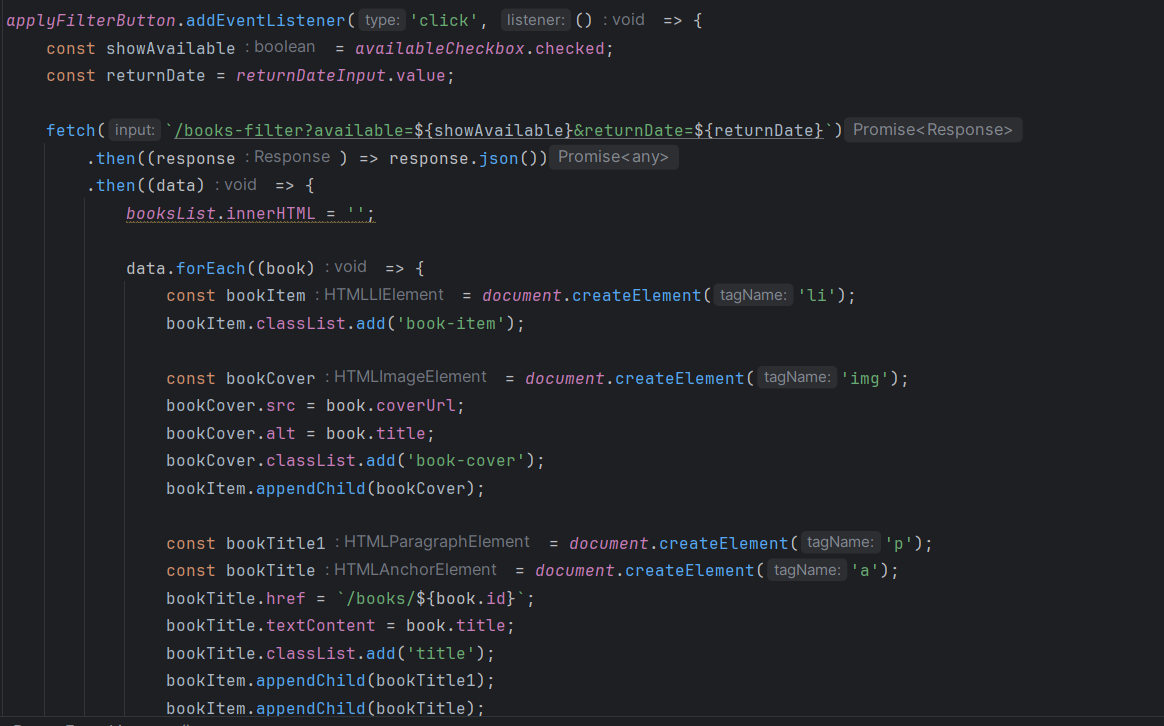
Файл script\_books.js

В данном файле происходит javascript контроль для шаблона books.pug.

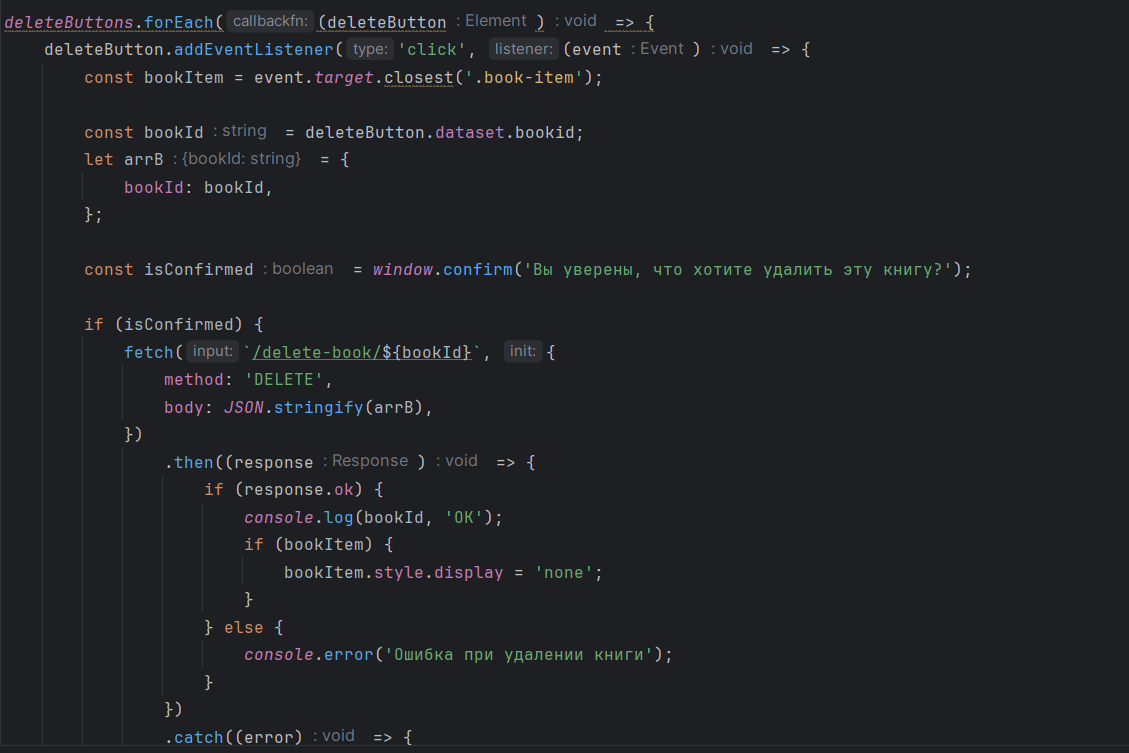
1. В функции addDeleteButtonEventListeners() происходит прослушивание клика на кнопку “Удалить книгу”. При клике вылезает оповестительное окошко. После повторного подтверждения книга удаляется.



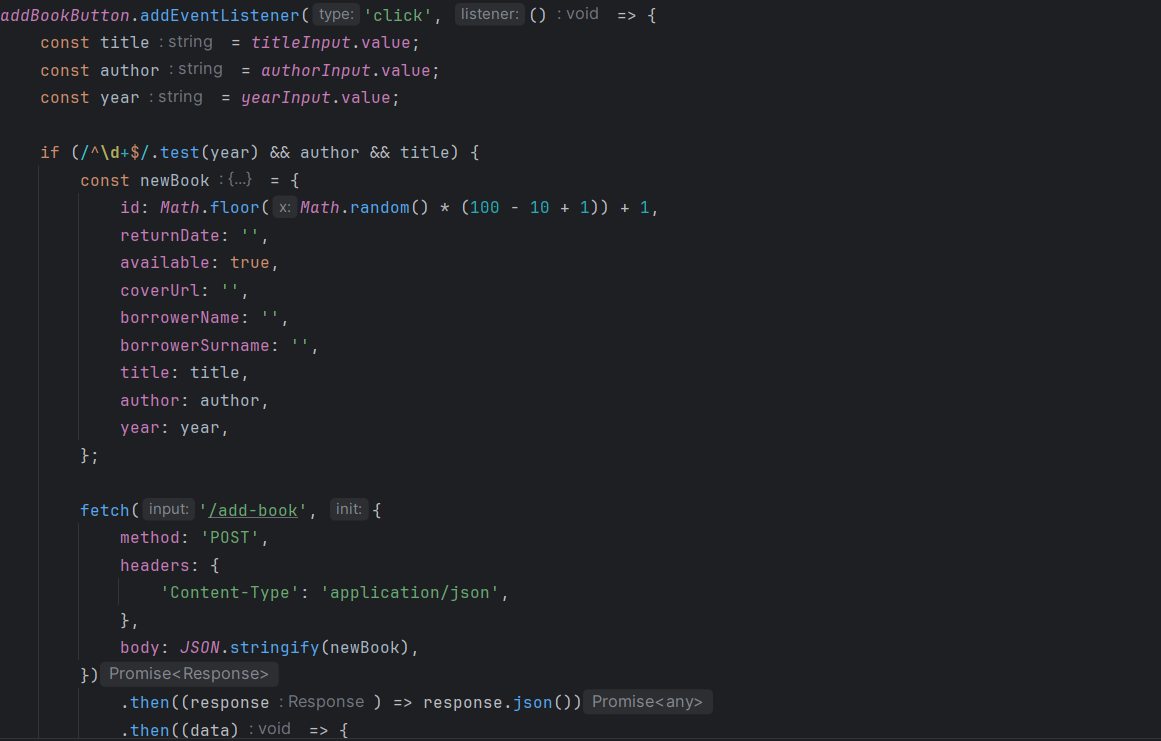
2. В функции applyFilterButton.addEventListener происходит прослушивание клика на кнопку “Фильтровать”. При клике книги отфильтровываются по определённому признаку.



3. В функции showAddFormButton.addEventListener происходит контроль над типом отображения формы для добавления книги.



4. В функции addBookButton.addEventListener происходит прослушивание клика на кнопку “Добавить книгу”. При клике происходит добавление книги.



# Выводы.

Был написан и отлажен сайт-библиотека с использованием REST-запросов, CSS стилей, web-фреймворка Express, Pug. Реализованы возможности редактирования книги, её удаления и добавления новой, а также авторизация пользователя.