МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Веб-технологии»

Тема: REST приложение управления библиотекой

|  |  |  |
| --- | --- | --- |
| Студент гр. 0304 |  | Асташёнок М.С. |
| Преподаватель |  | Беляев С.А. |

Санкт-Петербург

2022

Цель и задачи.

Целью работы является 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. Для достижения поставленной цели требуется решить следующие задачи:

– разработка интерфейса web-приложения;

– задание стилей для отображения web-приложения с учётом 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;

Общая формулировка задачи.

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 <dialog>.

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/>).

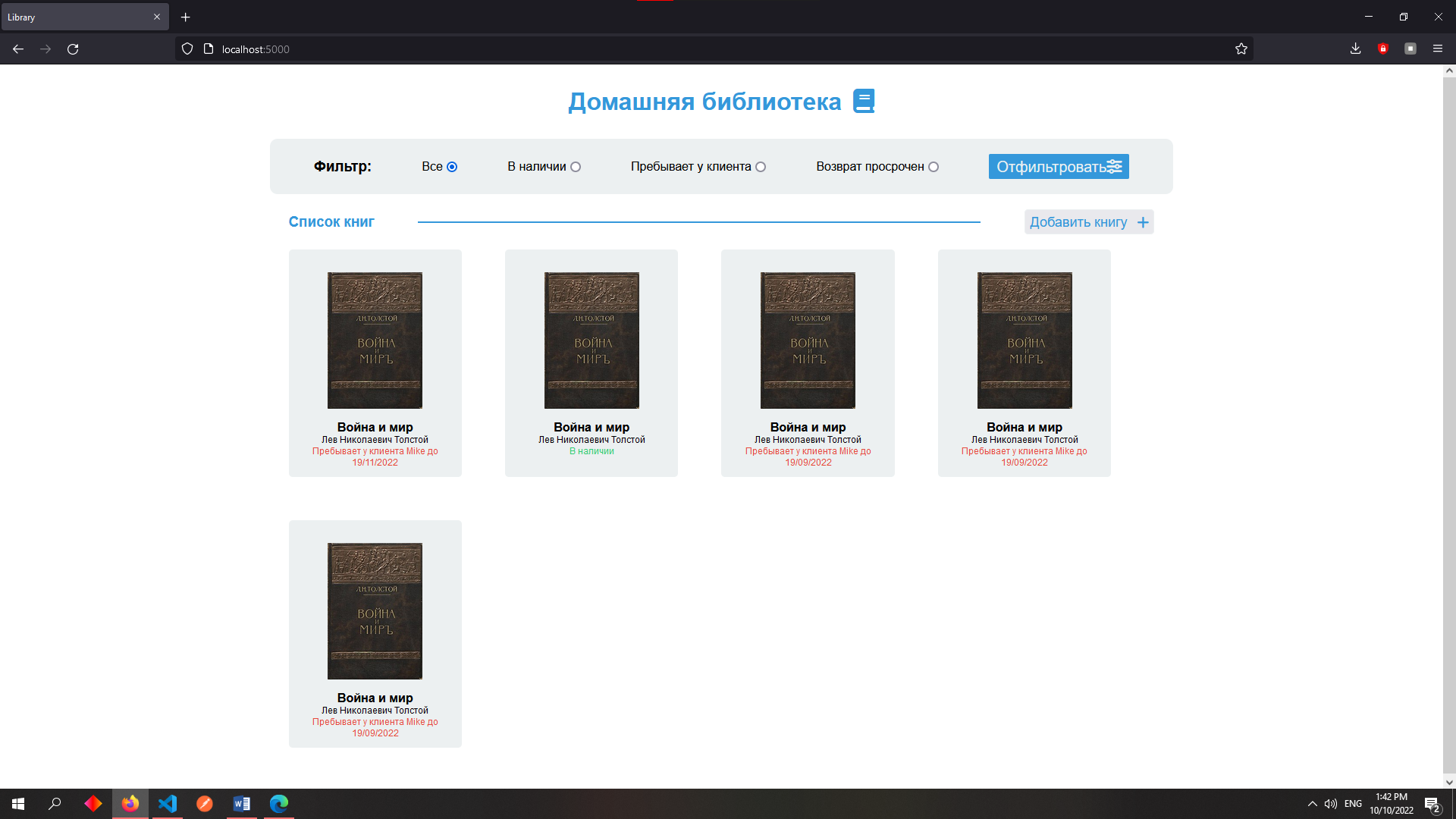
Преимуществом будет создание и использование аутентификации на основе passport.js (http://www.passportjs.org/).

Преимуществом будет реализация загрузки и отображения обложек книг.

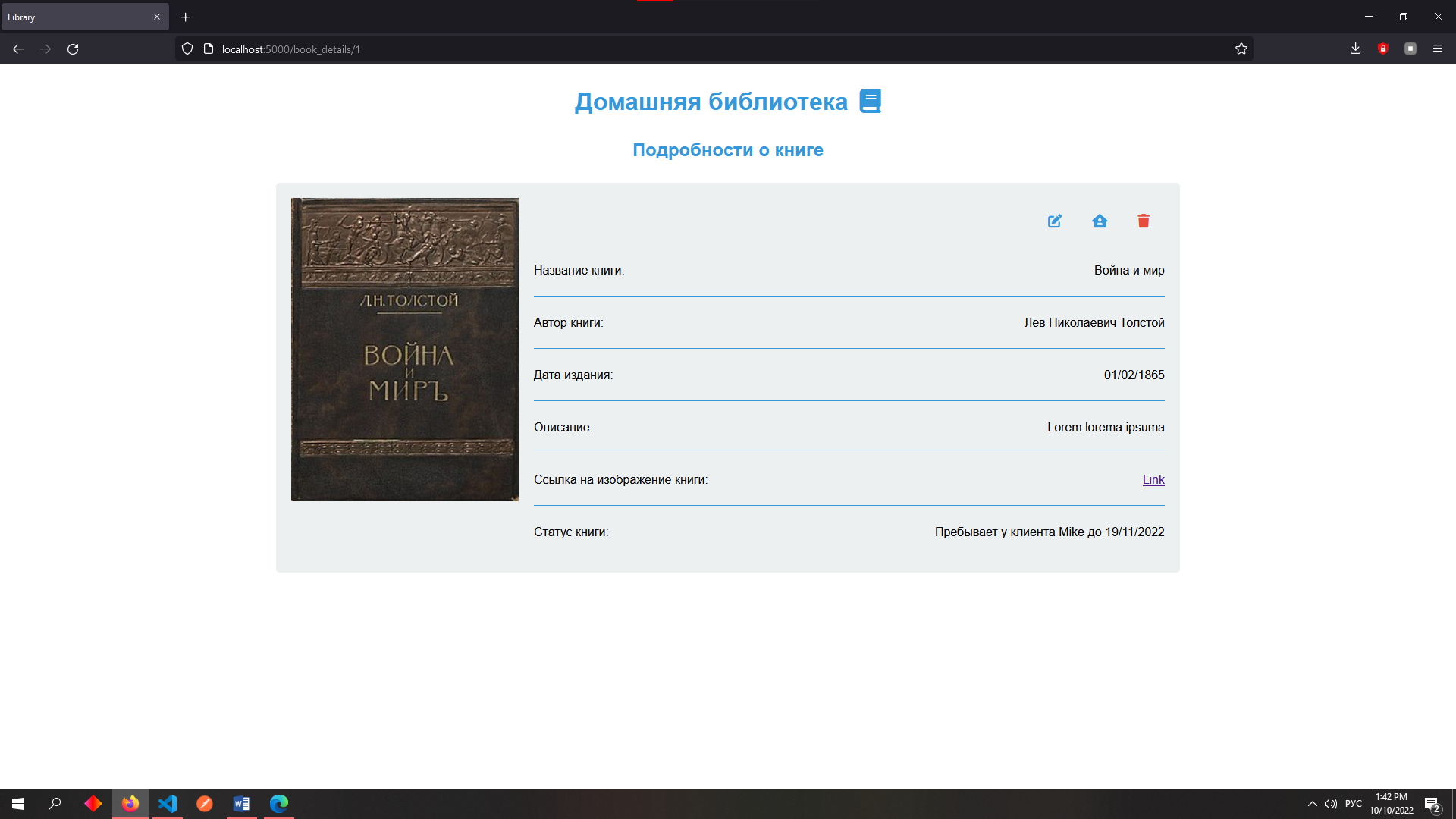
Выполнение задания.

Дизайн в Figma.

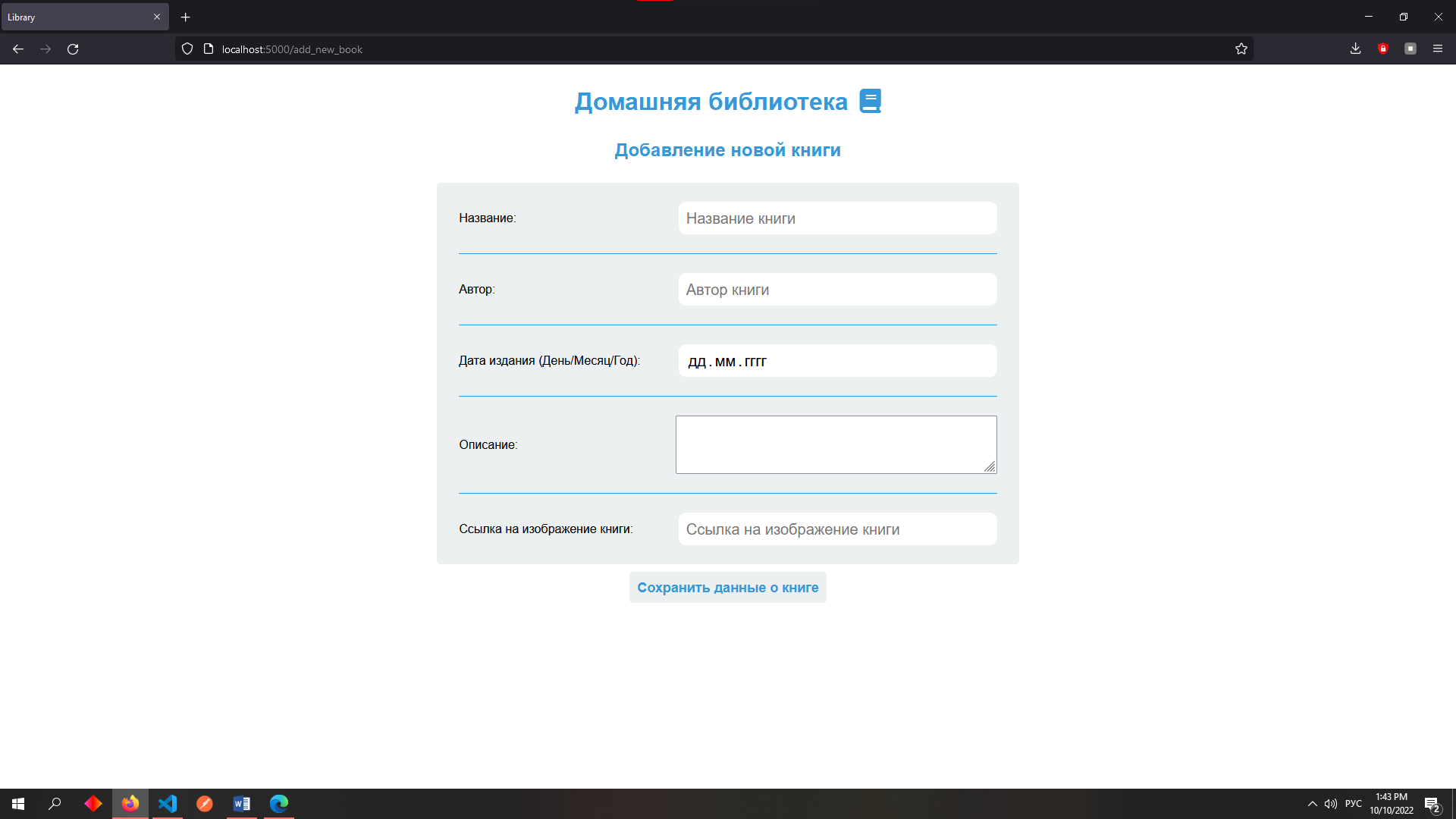
Первым делом был создан дизайн будущего сайта в программе Figma. По итогу получились следующие виды сверстанного сайта:



Домашняя страница.

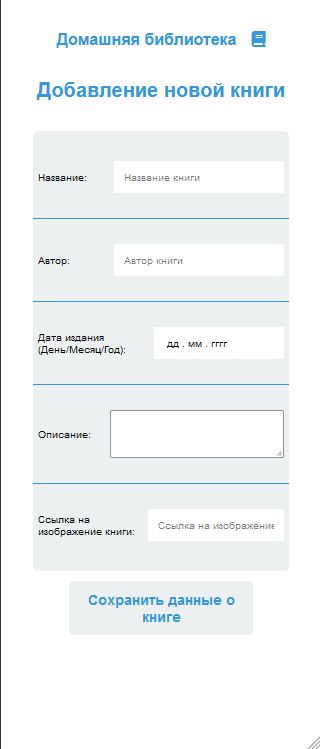
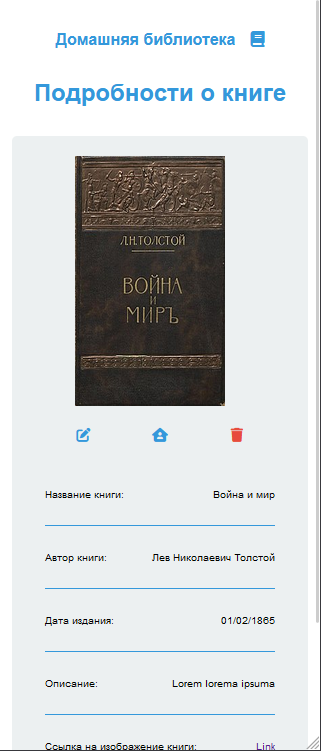
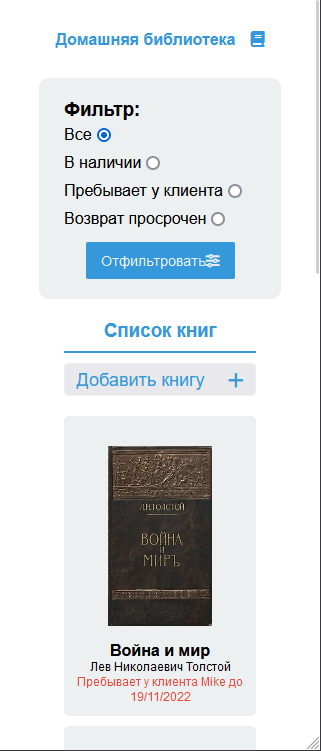


Данные о книге.



Создание новой книги.

И мобильные версии сайта (под экраны шириной 320px):



Главная страница, страница с данными о книге и страница добавления новой книги соответственно.

Верстка сайта.

Далее были сверстаны все компоненты будущего сайта. HTML документы страниц находятся в папке /public/html. В ней содержится три файла:

Index.html – начальная страница сайта.

Details.html – страница с данными о книге.

Add\_book.html – страница создания новой книги.

К этим файлам были применены css-стили, располагающиеся в папке /public/styles. Название css-файлов для каждого html-файла соответственно index.css, details.css и add\_book.css.

К стандартным стилям были также добавлены стили для мобильных устройств с помощью следующей конструкции:

*@media screen and (max-width: 320px) { описание стилей }*

После проведения верстки были созданы шаблоны pug для каждой из страниц для последующего рендеринга с помощью сервера на express.

База данных.

В качестве базы данных в этой лабораторной работе выступает database.json. В ней хранится две переменных:

Books - массив для хранения объектов книг.

Latest\_id – число для хранения последнего заданного идентификатора книги.

Каждый из объектов книги содержит в себе следующие поля:

Label – название книги.

Author – автор книги.

Release – дата издания.

Description – описание книги.

Image – ссылка на обложку книги.

Status – статус книги: 1 если в наличии, 2 если у клиента.

Client – клиент, у которого хранится книга.

Due – дата возврата книги.

Id – идентификатор книги.

Написание серверной части.

Сервер написан на фреймворке express.js и состоит из двух файлов: server.js и routes.js.

Server.js инициализирует сервер, задает ему роутер, позволяет импортировать и использовать css стили и pug шаблоны.

Routes.js содержит в себе описания исполняемых запросов. Адреса и исполняемая функция для каждого запроса следующие:

GET / - рендер стартовой страницы.

GET /add\_new\_book – рендер страницы с добавлением книги.

GET /book\_details/:book\_id – рендер страницы описания книги по ее идентификатору book\_id.

GET /api/get\_booklist – загрузка списка всех книг.

GET /api/get\_book/:book\_id – загрузка определенной книги по ее идентификатору book\_id.

PUT /api/update\_book/:book\_id – обновление информации о книге по ее идентификатору book\_id и переданному в теле запроса объекту book.

POST /api/create\_new\_book – создание новой книги в библиотеке по переданному в теле запроса объекту book.

DELETE /api/delete\_book/:book\_id – удаление книги из библиотеки по ее идентификатору book\_id.

Клиентская часть.

JS-скрипты для описания клиентской части находятся в папке /public/scripts.

В файле client.js хранятся функции для запроса по каждому из перечисленных выше адресов. Данный файл подключается ко всем шаблонам pug, ибо на каждой странице сайта может исполняться http-запрос.

Остальные три файла: index.js, details.js и add\_book.js подключаются к соответствующим шаблонам pug и реализуют функционал для всех кнопок, форм и других элементов на каждой из страниц.

Запуск приложения.

Для того, чтобы запустить сервер, необходимо перейти в корневую директорию проекта и ввести следующую команду:

node server.js

Вывод.

В данной лабораторной работе я познакомился с технологией создания сервера на node.js – express.js. Также были изучены принципы работы с CSS, шаблонами PUG, RESTful приложениями и AJAX запросами, а также с редактором Figma для построения дизайна сайта. По итогу, применил все полученные знания на практике, написав приложение онлайн-библиотеки.