**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В. И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №2**

**по дисциплине «WEB-технологии»**

Тема: **REST-ПРИЛОЖЕНИЕ УПРАВЛЕНИЯ**

**БИБЛИОТЕКОЙ**

|  |  |  |
| --- | --- | --- |
| Студент гр. 1381 |  | Возмитель В. Е. |
| Преподаватель |  | Беляев С. А. |

Санкт-Петербург

2023

**Цель.**

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.

**Основные теоретические сведения.**

*CSS* (*Cascading Style Sheets* – каскадные таблицы стилей) – язык описания внешнего вида документа, написанного с использованием языка разметки, используется как средство оформления внешнего вида *HTML*-страниц.

*Express* – это минималистичный и гибкий *web*-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и *web*-приложений.

Pug и EJS – модули, позволяющие использовать шаблоны для *HTML*-страниц.

*REST* *(Representational State Transfer* – передача состояния представления) – стиль взаимодействия компонентов распределенного приложения. В рамках лабораторной работы – браузера и сервера *web*-приложения. Для взаимодействия используются стандартные методы:

– GET – получение записи (записей);

– POST – добавление записи;

– PUT – обновление или добавление записи;

– DELETE – удаление записи.

**Задание.**

Необходимо создать *web*-приложение управления домашней библиоте-

кой, 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 от-

редактировать (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/*](https://fontawesome.ru/all-icons/)*).*

**Выполнение работы.**

1. Создание web-сервера на основе express.

Был создан сервер с поддержкой статический папок на сервере для хранения ресурсов и использования шаблонов web-страниц, который слушает 3000 порт, адрес – <http://localhost:3000>.

Выше описанное реализуется с помощью команд server.use и server.set.

1. Создание шаблонов web-страниц и задание для них стилей.

Шаблоны web-страниц могут создаваться с использованием шаблонов pug или шаблонов ejs.

Был создан ряд шаблонов для отображения разных web-страниц:

*Main.pug, input.pug, header.pug, book.pug, adding.pug.*

Далее для каждого шаблона задавался стиль с помощью каскадных таблиц стилей (css).

1. Создание json хранилища.

Был создан json файл, который содержит всю информацию о книгах:

название книги, автор, год первого издания, есть ли книга в наличии, и если нет – то кто ее взял.

С помощью команды *require("./books.json").books* было загружено и использовано содержимое *json*-файла *books.json.* Содержимое json файла автоматически преобразуется в полноценный json-объект, к которому можно обращаться как к обычному объекту.

1. Настройка маршрутов.

Настройка маршрутов обычно происходит в отдельных файлах.

Так в файле routes.js обрабатывались все возможные маршруты, к которым пользователь может обратиться с использованием их идентификатора.

Имена и значения заключены в кавычки. Значения такого объекта – всегда строки, при необходимости использования других типов их придется преобразовывать из строк.

Обращение происходит с использованием метода GET. Добавление записи – метод POST.

Параметры в GET-запросе передаются в URL, а в POST-запросе – в теле запроса.

Обновление или добавление записи – метод PUT. Принципы обработки PUT-запроса и получения параметров идентичны POST-запросу. Отличия в логике: если удается найти объект, то он обновляется, иначе – добавляется.

Удаление записи – метод DELETE.

1. Подключение иконок Font Awesome и разработка макетов интерфейсов с использованием Figma.

В заключение работы были загружены и добавлены в проект некоторые иконки Font Awesome.

Также разработаны макеты интерфейсов для каждой страницы с использованием Figma. Скриншоты и ссылка находятся в приложении А.

**Вывод.**

Было изучено взаимодействие клиентского приложение с серверной частью, освоены шаблоны web-страниц, сформированы навыки разработки динамических HTML-страниц. Также получены навыки в принципах построения приложения с насыщенным интерфейсом пользователя.

## Приложения А. *Figma* макеты и ссылка.

Ссылка на Figma проект: <https://www.figma.com/file/AOKUh8jxCTL4zmpkSsU3Kw/Library?type=design&node-id=6%3A357&mode=design&t=TqTRzZJCNCXRkJkN-1>

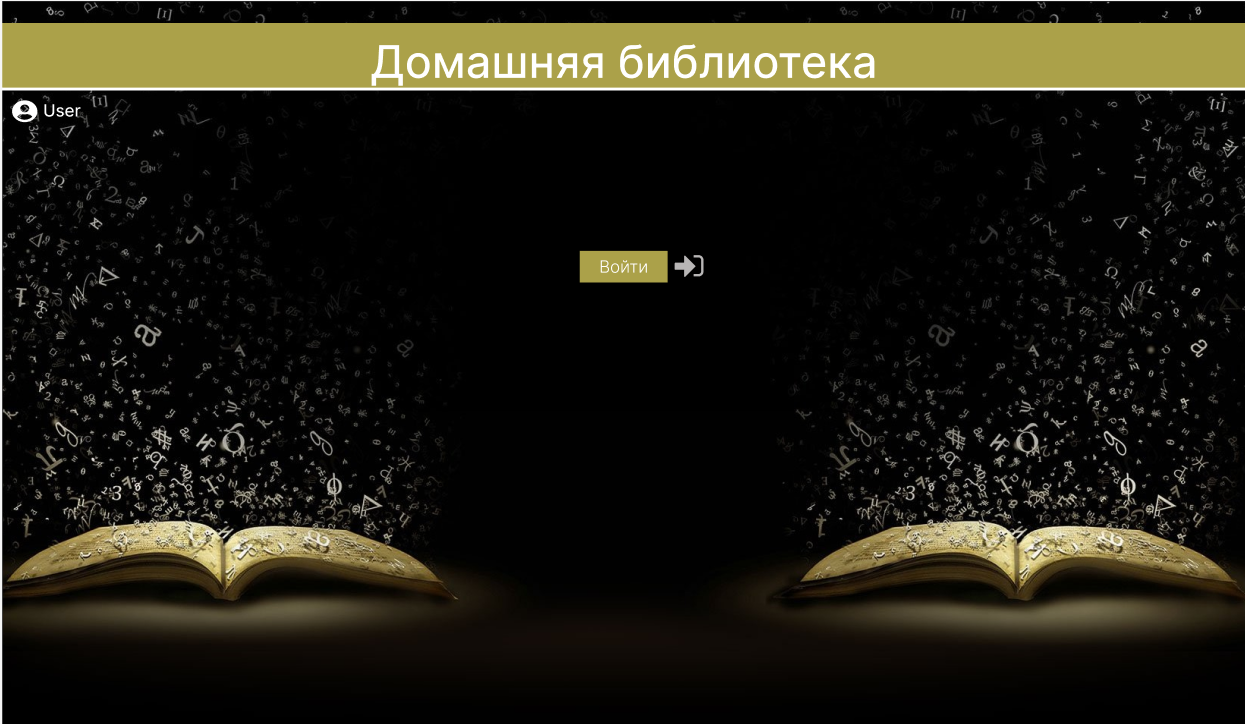


Рис №1 – Макет начальной страницы.

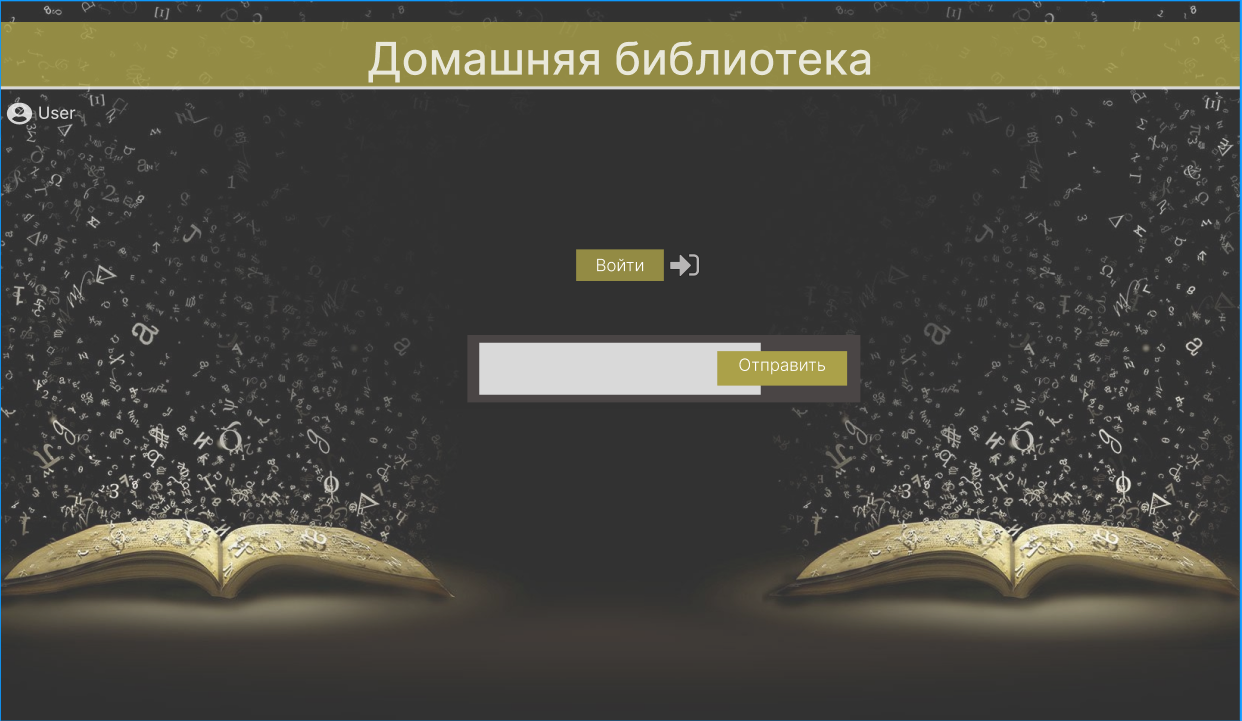


Рис №2 – 2 макет начальной страницы.



Рис №3 – макет основной страницы (библиотека).

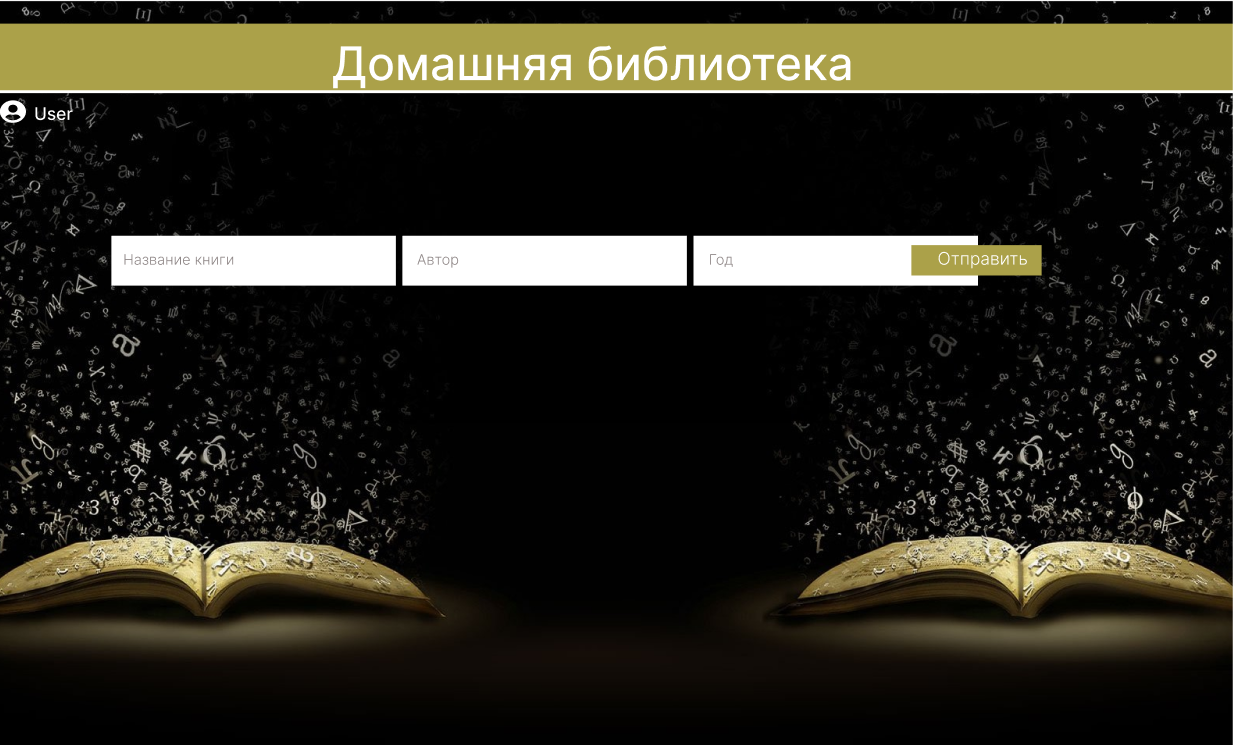


Рис №4 – макет страницы добавления книги.

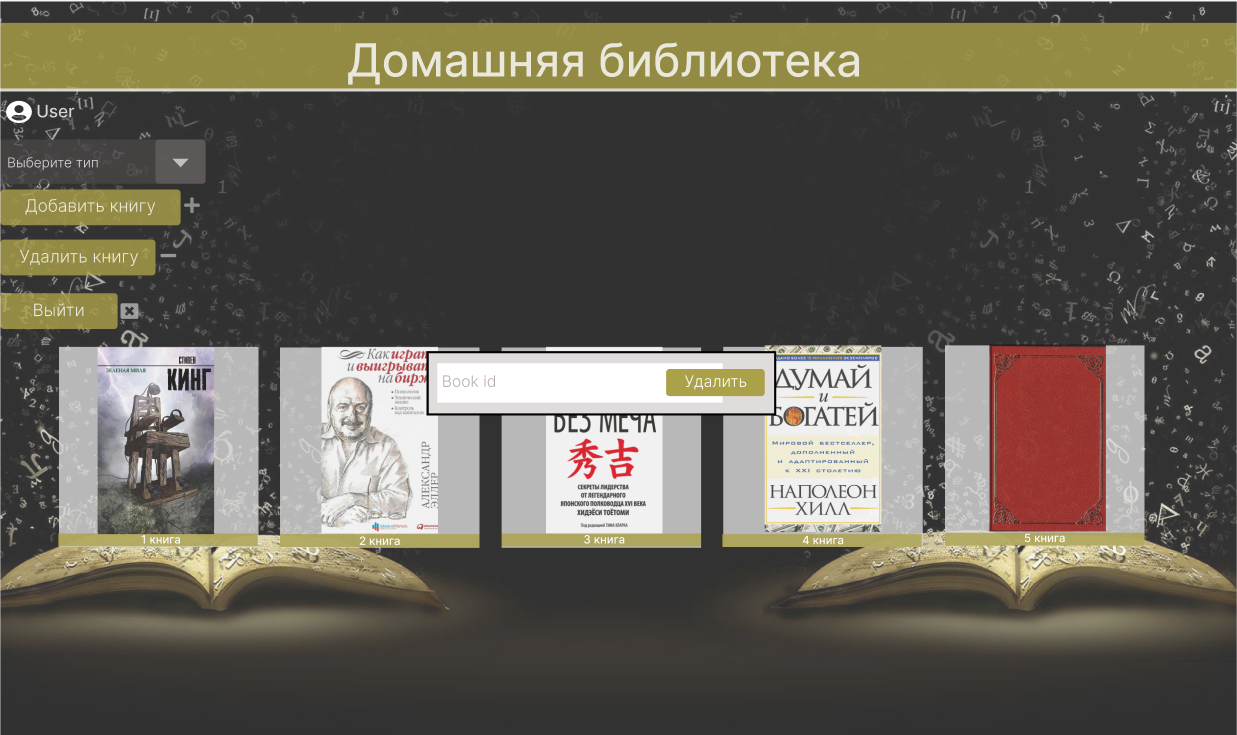


Рис №5 – макет страницы удаления книги.



Рис №6 – макет страницы для книги.