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. 3341 _____ Мальцев К.Л.

Преподаватель _____ Беляев С.А.

Санкт-Петербург

2025

Цель работы.

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.

Задание.

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 и текущее состояние библиотеки хранится в JSON-файле на сервере.

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 (либо TypeScript).

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/), в качестве примера можно использовать https://nodejsdev.ru/guides/webdraftt/authentication/.

Преимуществом будет реализация загрузки и отображения обложек книг.

Основные теоретические положения.

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.

Описание решения.

Проект реализован как клиент-серверное Web-приложение на JavaScript (ES6 modules) с применением Node.js, Express, pug и AJAX.

Для взаимодействия клиентской и серверной части был продуман API, соответствующий REST. Были разработаны следующие запросы:

– GET /books – возвращает основную страницу, отображающую все книги, в формате ‘text/html’;
– GET /books/:id – возвращает страницу книги с заданным id в формате ‘text/html’;
– GET /api/books – возвращает часть страницы со списком книг, удовлетворяющим условию (фильтр и поиск) в формате ‘text/html’ или ‘application/json’ в зависимости от параметров запроса;
– GET /api/books/:id – возвращает часть страницы карточки книги с заданным id в формате ‘text/html’ или ‘application/json’ в зависимости от параметров запроса;
– POST /api/books – реализует добавление новой книги. Обязательные параметры: название книги title и имя автора author. Необязательный параметр – year, год издания;

– PUT /api/books/:id – реализует обновление информации о книге по её id;

– DELETE /api/books/:id – реализует удаление книги по определенному id;

– POST /api/books/:id/borrow – реализует функцию выдачи книги читателю. В теле запроса обязательны параметры borrower – имя читателя и date – дата возврата книги;

– POST /api/books/:id/return – реализует функцию возврата книги в библиотеку.

Выводы.

В результате выполнения данной лабораторной работы были изучены принципы взаимодействия клиентского приложения с серверной частью, освоены шаблонов web-страниц, и принципы построения приложений с насыщенным интерфейсом пользователя.

Было написано web-приложение для управления домашней библиотекой. Было реализовано json-хранилище, спроектированы REST-запросы, спроектирован и реализован макет приложения при помощи шаблонов pug, стилей CSS с применением иконок. Была реализована адаптивность интерфейса, учитывающая размер экрана.