**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. 0304 |  | Жиглов Д.С. |
| Преподаватель |  | Беляев С.А. |

Санкт-Петербург

2022

## Цель работы.

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

## Задание.

– 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;

## 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.

## Выполнение работы.

1. Для решения поставленной задачи был разработан интерфейс страниц с использованием web-приложения Figma.



Рисунок 1: Список книг

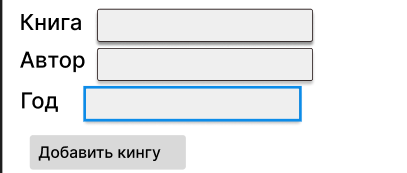


Рисунок 2: Добавление книги

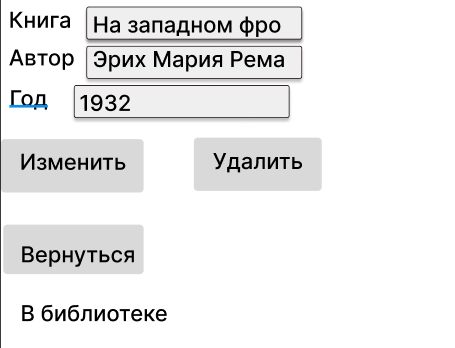


Рисунок 3: Редактирование или удаление книги

2. Были разработаны страницы библиотеки, используя фреймворк Pug. Были созданы страницы:

index.pug — приветственная страница и по совместительству добавление книг .

edit\_page.pug — страница редактирования книг..

lend\_book.pug – страница выдачи книги читателю.

Для установки стилей использовался .css файл. Для установки иконок использовалась библиотека font awesome 4.

3. На основе модуля Express была разработана логика сервера.

Файл app.js содержит основные настройки сервера на порту 3001. В файле app.js подключаются основные модули для работы сервера и скриптовые файлы сервера, в данной лабораторной таких файлов три — это script\_homepage.js, script\_edit.js,script\_lend\_book.js. В нём происходит обработка приходящих REST запросов, обработка информации, сохранение данных.

4. Для хранения данных был создан файл books.json, который хранит в себе map вида {«id»: «data»}. В качестве данных используются:

* Name - название книги
* author — имя автора
* year — год издания
* owner — текущий владелец книги, если книга свободна — None.
* date — дата возврата книги.
* Id – собственно айдишник книги

4. Для работы с серверными данными формата json был создан файл app.js, которые подключается к скриптовым файлам и реализует все функции работы с библиотекой: обновление данных книги, добавление новой книги, удаление и сортировка списка книг.

Id является уникальным идентификатором и определяется на базе id последнего элемента в библиотеке плюс единица.

**Выводы.**

В ходе лабораторной работы был разработан сервер на основе модуля Express, с использованием html-шаблонов на базе pug. Между браузером и сервером реализована передача AJAX запросов с данными, которые извлекаются с помощью модуля multiparty. Хранение данных о книгах реализовано на основе .json файла.

# Приложение А Исходный код программы

Файл: script\_homepage.js

let button\_available = document.querySelector(".btn-available")

let button\_all = document.querySelector(".btn-all")

let button\_given = document.querySelector(".btn-given")

button\_all.addEventListener("click",show\_all\_books)

button\_given.addEventListener("click",show\_given\_books)

button\_available.addEventListener("click",show\_available\_books)

let rows = document.querySelector("tbody");

async function getBooks(){

let response = await fetch("/books",{

method: "GET",

headers: {

"Content-Type": "application/json; charset=utf-8"

}

})

let books = await response.json()

rows.innerHTML = ""

for (let book of books ){

//vision\_row(book)

rows.append(vision\_row(book))

//document.querySelector(".books\_all").append(vision\_row((book)))

}

}

async function show\_available\_books(){

let response = await fetch("/books/available", {

method: "GET",

headers: {

"Content-Type": "application/json; charset=utf-8"

}

})

//let innerHTML = ""

let books = await response.json()

rows.innerHTML = ""

for (let book of books ){

if(book.owner === ""){

rows.append(vision\_row(book))

}

}

//document.querySelector(".books\_all").innerHTML = innerHTML

}

async function show\_all\_books(){

let response = await fetch("/books", {

method: "GET",

headers: {

"Content-Type": "application/json; charset=utf-8"

}

})

let books = await response.json()

//document.querySelector(".books\_all").innerHTML = ""

rows.innerHTML = ""

for (let book of books ){

//vision\_row(book)

//document.querySelector(".books\_all").append(vision\_row((book)))

rows.append(vision\_row(book))

}

}

async function show\_given\_books(){

let response = await fetch("/books", {

method: "GET",

headers: {

"Content-Type": "application/json; charset=utf-8"

}

})

let books = await response.json()

//document.querySelector(".books\_all").innerHTML = ""

rows.innerHTML = ""

for (let book of books){

if(book.owner !== ""){

// let date\_info = book.date

// let id\_info = book.id

// let book\_info = {id:id\_info,

// date: date\_info};

//

rows.append(vision\_row(book))

}

}

}

let form = document.querySelector("form");

form.addEventListener("submit",function (event){

event.preventDefault()

let name = this.elements["name"].value;

let author = this.elements["author"].value;

let year = this.elements["year"].value

//console.log(name)

createUser(name,author,year)

})

async function createUser(Name,author,year ) {

let response = await fetch("/books/available", {

method: "POST",

headers: {

"Content-Type": "application/json; charset=utf-8"

},

body: JSON.stringify({

book: Name,

author: author,

year: year

})

});

if(response.ok === true){

form.reset()

await getBooks()

console.log("hell")

}

}

function vision\_row(book){

const tr = document.createElement("tr");

//tr.className = "row"

const nameTd = document.createElement("td");

nameTd.append(book.book);

//nameTd.style.paddingRight = "50px";

//nameTd.style.width = "300px";

//nameTd.style.height = "50px"

tr.append(nameTd);

const authorTd = document.createElement("td");

authorTd.append(book.author)

//authorTd.style.width = "200px"

tr.append(authorTd)

const yearTd = document.createElement("td");

yearTd.append(book.year)

tr.append(yearTd)

let OwnerName = document.createElement("td");

if(book.owner !== ""){

let owner = book.owner

let date = book.date

OwnerName.append(owner+" " +date)

// OwnerName.style.paddingRight = "50px"

// OwnerName.style.width = "100px"

tr.append(OwnerName)

}

else if(book.owner === "") {

let in\_library = document.createElement("td");

in\_library.append("В библиотеке")

// in\_library.style.paddingRight = "200px"

tr.append(in\_library)

}

const links\_all = document.createElement("td");

const editLink = document.createElement("a")

let url = new URL('http://localhost:3001/book')

url.searchParams.set('id',book.id)

editLink.text = "Edit"

editLink.setAttribute("style", "cursor:pointer;padding:15px;")

editLink.href = url.toString()

// editLink.style.paddingRight = "50px";

links\_all.append(editLink)

// tr.append(editLink)

const GiveLink = document.createElement("a")

let url\_give = new URL('http://localhost:3001/book/give')

url\_give.searchParams.set('id',book.id)

GiveLink.text = "Выдать"

GiveLink.setAttribute("style", "cursor:pointer;padding:15px;")

GiveLink.href = url\_give.toString()

links\_all.append(GiveLink)

// tr.append(GiveLink)

tr.appendChild(links\_all)

return tr;

}

getBooks()

Файл: script\_edit.js

let id\_needed = 0

let form = document.querySelector("form");

async function Edit\_User(name,author,year) {

let response = await fetch("/book", {

method: "PUT",

headers: {

"Content-Type": "application/json; charset=utf-8"

},

body: JSON.stringify({

book: name,

author: author,

id: id\_needed,

year: year

})

})

if (response.ok === true){

console.log("good")

}

}

async function get\_id(){

let response = await fetch("/id", {

method: "GET",

headers: {

"Content-Type": "application/json; charset=utf-8"

}

})

id\_needed = await response.json()

}

async function fill\_info() {

let response = await fetch("/books", {

method: "GET",

headers: {

"Content-Type": "application/json; charset=utf-8"

}

})

await get\_id()

let books = await response.json()

for (let bookk of books) {

if (bookk.id === id\_needed) {

form.elements["name"].value = bookk.book;

form.elements["author"].value = bookk.author;

form.elements["year"].value = bookk.year;

if(bookk.owner !== ""){

const tr = document.createElement("tr")

const nameTd = document.createElement("td");

nameTd.append(bookk.owner)

tr.append(nameTd)

const dateTd = document.createElement("td");

dateTd.append(bookk.date)

tr.append(dateTd)

const yearTd = document.createElement("td");

yearTd.append(bookk.year)

tr.append(yearTd)

document.querySelector(".info\_book").append(tr)

}

else{

const tr = document.createElement("tr")

let text = document.createElement("td");

text = "Книга находится в библиотеке"

tr.append(text)

document.querySelector(".info\_book").append(tr)

}

break;

}

}

}

form.addEventListener("submit",function (event){

event.preventDefault()

let name = this.elements["name"].value;

let author = this.elements["author"].value;

let year = this.elements["year"].value;

Edit\_User(name,author,year)

})

let button\_delete = document.querySelector(".btn-delete")

button\_delete.addEventListener("click", book\_delete)

async function book\_delete(){

if(confirm("Точно хотите удалить?")){

let response = await fetch("/book", {

method: "DELETE",

headers: {

"Content-Type": "application/json; charset=utf-8"

},

body: JSON.stringify({

id: id\_needed

})

})

window.location.href = 'http://localhost:3001';

if (response.ok === true){

console.log("good")

}

}

}

fill\_info()

Файл: script\_lend\_book.js

let id\_needed = 0;

let form = document.querySelector("form");

async function get\_id(){

let response = await fetch("/id", {

method: "GET",

headers: {

"Content-Type": "application/json; charset=utf-8"

}

})

id\_needed = await response.json()

console.log(id\_needed)

}

async function Add\_owner(name,date){

let response = await fetch("/book/give", {

method: "PUT",

headers: {

"Content-Type": "application/json; charset=utf-8"

},

body: JSON.stringify({

name: name,

date: date,

id: id\_needed

})

})

if (response.ok === true){

console.log("good")

}

}

form.addEventListener("submit",function (event){

event.preventDefault()

let name = this.elements["name"].value;

let date = this.elements["date"].value;

Add\_owner(name,date)

})

get\_id()