**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.**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 9304 |  | Рослова Л.С. |
| Преподаватель |  | Беляев С.А. |

Санкт-Петербург

2021

## 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.

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, все web-страницы должны быть сделаны с использованием pug.

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

5. Предусмотрена страница для карточки книги, в которой ее можно отредактировать (минимум: автор, название, дата выпуска) и дать 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 модульного 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 AJAXзапросов.

10. Логика приложения реализована на языке JavaScript.

## 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.

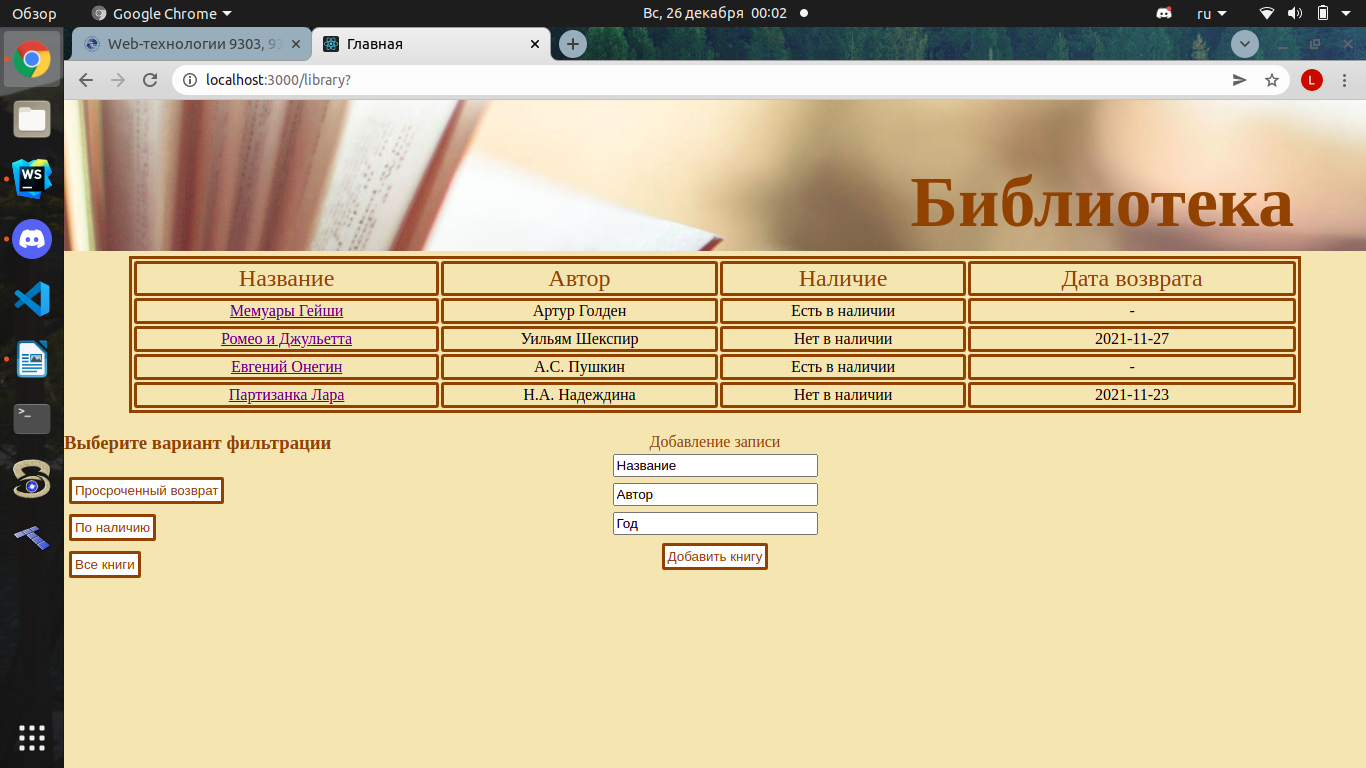
Express – это минималистичный и гибкий web-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и web-приложений.

Pug – модуль, позволяющий 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.

Сначала были написаны шаблоны 2-х веб-страниц на PUG: страница со списком книг и их добавлением, карточка книги. Результат представлен на рисунках 1-2.

Рисунок 1 – Список книг

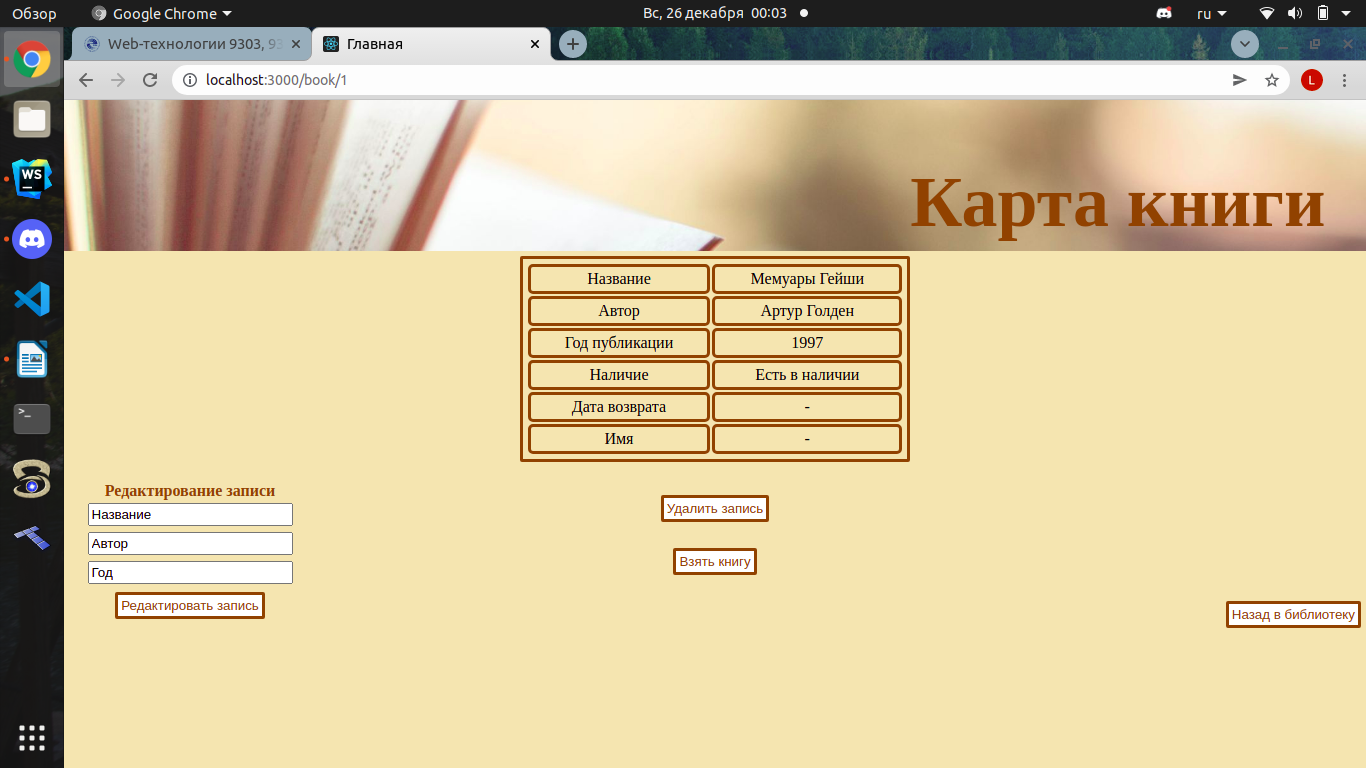


Рисунок 2 – Форма добавления книги

Далее был написан app.js, который представляет из себя серверную часть приложения.

Вся данные о книгах хранятся в соответствующих JSON-файлах.

При выдаче книги появляется модульное окно, что показано на рисунке 3.

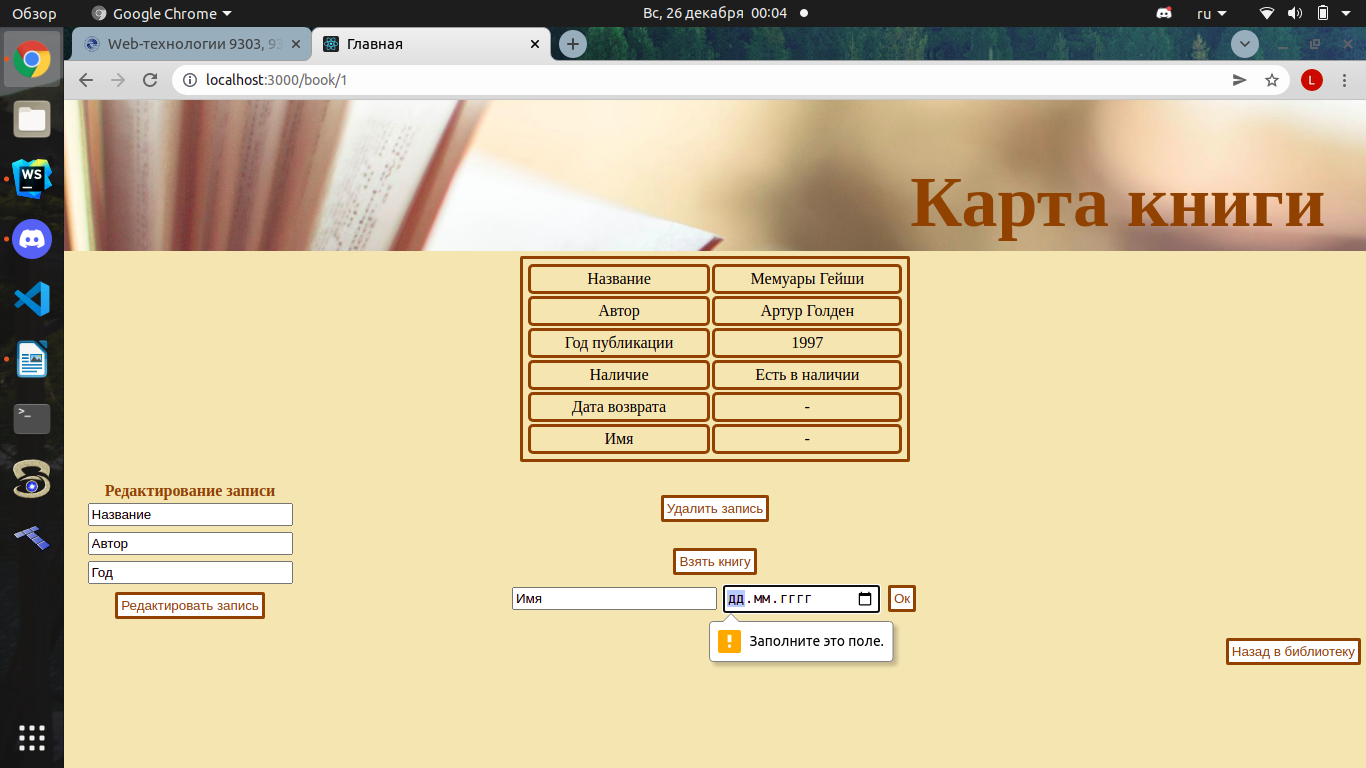


Рисунок 3 – Выдача книги

## Выводы.

Изучили взаимодействия клиентского приложения с серверной частью, освоили шаблоны web-страниц, сформировали навыки разработки динамических HTML-страниц, освоили принципы построения приложений с насыщенным интерфейсом пользователя.

# Приложение А Исходный код программы

Название файла: app.js

const express = require('express');

const app = express();

var bodyParser = require('body-parser');

app.use(bodyParser.urlencoded({extended: true}));

//позволяет генерировать абс.пути для передачи стат.файлов

const path = require('path');

//подключаем маршруты

let route = require('./route');

//Указываем механизм визуализации

app.set('view engine', 'pug');

app.set('views', path.join(\_\_dirname, 'views'));

app.use('/public', express.static('public'));

app.use('/', route);

app.use(express.static(\_\_dirname));

app.listen(3000, ()=>{

console.log("Server started at browser http://localhost:3000");

});

Название файла: routes.js

var express = require('express');

var router = express.Router();

var library = require('./book.json');

var bodyParser = require('body-parser');

router.get('/', (req, res)=>{

res.render('index');

});

router.get('/library', (req, res, next) => {

res.render("library.pug", {books: library});

next();

});

router.get('/filter/:num', (req, res, next) => {

const id = req.params.num;

if(id === "all"){

let array = [];

for(item of library){

array.push(item.id);

}

res.end(JSON.stringify(array));

return;

}

else if(id ==="returnDate"){

let array = [];

let todayDate = new Date();

for(item of library){

let itemDate = new Date(item.date\_return+"T23:59:59.999Z");

if(item.availability === "Есть в наличии" || itemDate > todayDate) {

array.push(item.id)

}

}

res.end(JSON.stringify(array));

return;

}

else if(id ==="avail"){

let array = [];

for(item of library){

if(item.availability === "Нет в наличии") {

array.push(item.id);

}

}

res.end(JSON.stringify(array));

return;

}

next();

});

router.get('/book/:num', (req, res, next) => {

const id = req.params.num;

for(let value of library) {

if (value.id === id) {

res.render('book.pug', {book: value})

}

}

next();

});

router.post('/lib\_edit', (req, res, next) => {

let last\_id;

if(library.length === 0){

last\_id = 1;

}

else{

last\_id = library.length+1;

}

library.splice(last\_id-1, 0, {

"id": `${last\_id}`,

"title": req.body.title,

"author": req.body.author,

"year": req.body.year,

"availability": "Есть в наличии",

"return\_date": "-",

"person": "-"

});

res.redirect('/library');

next();

});

router.post("/book/delete/:num", (req, res, next) => {

let id = req.params.num;

for(item of library){

if(item.id === id){

library.splice(id-1, 1);

}

}

let n=1;

for(item of library){

item.id = `${n}`;

n+=1;

}

res.redirect('/library');

next();

});

router.post("/book/edit/:num", (req, res, next) => {

let id = req.params.num;

for (item of library) {

if (item.id === id) {

item.title = req.body.title;

item.author = req.body.author;

item.year = req.body.year;

}

}

res.redirect(`/book/${id}`);

next();

});

router.post('/book/return/:num', (req, res, next) => {

const id = req.params.num;

for (item of library) {

if (item.id === id) {

item.name = "-";

item.date\_return = "-";

item.availability = "Есть в наличии";

}

}

res.redirect(`/book/${id}`);

next();

});

router.post('/book/get/:num', (req, res, next) => {

const id = req.params.num;

for (item of library) {

if (item.id === id) {

item.name = req.body.person ;

item.date\_return = req.body.return\_date;

item.availability = "Нет в наличии";

}

}

res.redirect(`/book/${id}`);

next();

});

module.exports = router;