

**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 управление библиотекой**

Студент гр.8303
Преподаватель

Рудько Д.Ю.
Беляев С.А.

Санкт-Петербург
2020

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, которая предоставляет список книг, их можно отфильтровать по признакам «в наличии», «возврат просрочен», есть возможность выдать книгу для чтения и вернуть книгу. Основные требования:

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.
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 её 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 или кто её взял (имя) и когда должен вернуть (дата).
6. Информация о читателе вводится с использованием всплывающего модального 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 AJAX запросов.
10. Логика приложения реализована на языке JavaScript.

Ход выполнения.

Создан web-сервер на основе express. Написан шаблон для первой и основной страниц, реализован ввод из json файлов. Разработан дизайн и стили всех окон и написаны методы для обработки необходимых кнопок.

Домашняя библиотека



Войти

Книжный шкаф:

Номер	Название	Автор	Год выпуска	Наличие	Дата возврата	Действие
0	20 000 лье под водой	Жюль Верн	1956-01-01	да	-	x
1	Путешествие к центру земли	Жюль Верн	1958-01-01	да	-	x
2	Дневной дозор	Сергей Лукьяненко	2000-01-01	да	-	x
	<input type="text"/>	<input type="text"/>	дд . мм . гггг			+

Фильтр:

Все книги

Книги в наличии

Просрочено

Написан код шаблона страницы с редактированием книги. Созданы методы для обработки всех кнопок.

Книга:	20 000 лье под водой	Взять книгу
Автор:	Жуль Верн	Вернуть книгу
Год выпуска:	1956-01-01	Редактировать
В наличии:	да	
Читатель	-	
Дата возврата:	-	

Назад

Книга:	20 000 лье под водой	Взять книгу
Автор:	Жуль Верн	Читатель ✕ ФИО читателя <input type="text"/> Дата возврата <input type="text" value="ДД . ММ . ГГГГ"/> <input type="button" value="Взять"/>
Год выпуска:	1956-01-01	
В наличии:	да	
Читатель	-	
Дата возврата:	-	
<input type="button" value="Назад"/>		

Выводы.

В ходе выполнения лабораторной работы было изучение взаимодействию клиентского приложения с серверной частью, освоены шаблонов web-страниц, и сформированы навыки разработки динамических HTML-страниц. Также освоены принципы построения приложений с насыщенным интерфейсом пользователя.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД

www

```
#!/usr/bin/env node

var app = require('../app');
var debug = require('debug')('lab2:server');
var http = require('http');

var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

var server = http.createServer(app);

server.listen(port);
server.on('error', onError);
server.on('listening', onListening);

function normalizePort(val) {
  var port = parseInt(val, 10);

  if (isNaN(port)) {
    // named pipe
    return val;
  }

  if (port >= 0) {
    // port number
    return port;
  }

  return false;
}

function onError(error) {
  if (error.syscall !== 'listen') {
    throw error;
  }

  var bind = typeof port === 'string'
    ? 'Pipe ' + port
    : 'Port ' + port;

  // handle specific listen errors with friendly messages
  switch (error.code) {
    case 'EACCES':
      console.error(bind + ' requires elevated privileges');
      process.exit(1);
      break;
    case 'EADDRINUSE':
      console.error(bind + ' is already in use');
      process.exit(1);
      break;
    default:
      throw error;
  }
}

function onListening() {
  var addr = server.address();
  var bind = typeof addr === 'string'
    ? 'pipe ' + addr
    : 'port ' + addr.port;
  debug('Listening on ' + bind);
}
```

app.js

```
var createError = require('http-errors');
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');

var indexRouter = require('./routes/routes');

var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');

app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', indexRouter);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  next(createError(404));
});

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;
```

router.js

```
var express = require('express');
var router = express.Router();
var book_json = require('../public/json/lib');

/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Домашняя библиотека' });
});

router.get('/library', function(req, res, next) {
  res.render('library', {title: 'library', books: book_json});
});

router.post('/new', function (req, res) {
  for(let value of book_json){
    if(value.name === req.body.name){
      res.redirect('/library');
      return
    }
  }

  let reserv = book_json.pop();
  book_json.push(reserv);
  let newId = reserv.id+1;

  book_json.push({
    "id": newId,
    "name": req.body.name,
    "author": req.body.author,
```

```

        "date": req.body.date,
        "in_library": "да",
        "person": "-",
        "date_return": "-"
    });
    res.redirect('/library');

});

router.post('/del/:number', function (req, res) {
    let id = req.params.number * 1;
    book_json = book_json.filter(it => {
        return it.id !== id;
    });
    res.redirect('/library');
});

router.get('/book/:number', function (req, res) {
    let id = req.params.number * 1;
    const value = book_json.filter(it => {
        return it.id === id;
    });
    res.render('book', {title: 'library', ID: `${value[0].id}`, name: `${value[0].name}`,
        author: `${value[0].author}`, date: `${value[0].date}`, in_library: `${value[0].in_library}`,
        person: `${value[0].person}`, date_return: `${value[0].date_return}`});
});

router.post('/book/read/:number', function (req, res){
    let id = req.params.number*1;
    for (let value of book_json) {
        if (value.id === id) {
            value.person = req.body.name;
            value.date_return = req.body.date_return;
            value.in_library = "нет";
            break;
        }
    }
    //res.redirect('/library');
    res.redirect('/book/' + id);
});

router.post('/book/return/:number', function (req, res){
    let id = req.params.number *1;
    for (let value of book_json) {
        if (value.id === id) {
            value.person = "-";
            value.date_return = "-";
            value.in_library = "да";
            break;
        }
    }
    //res.redirect('/library');
    res.redirect('/book/' + id);
});

router.post('/book/change/:number', function (req, res){
    let id = req.params.number *1;
    for (let value of book_json) {
        if (value.id === id) {
            if (req.body.name)
                value.name = req.body.name;
            if (req.body.author)
                value.author = req.body.author;
            if (req.body.date)
                value.date = req.body.date;
            break;
        }
    }
    res.redirect('/book/' + id);
    //res.redirect('/library');
});

```



```

})

router.post('/book/home', function (req, res){
  res.redirect('/library');
});

router.get('/filter/:functionName', function (req, res){
  let id = req.params.functionName;
  if(id === 'inLib'){
    let idArray = [];
    book_json.forEach((v, i) => {
      if (v.in_library === "нет")
        idArray.push(v.id)
    });
    res.end(JSON.stringify(idArray));
    return;
  }

  if(id === 'allBooks'){
    let allArray = [];
    for (let value of book_json)
      allArray.push(value.id)
    res.end(JSON.stringify(allArray));
    return;
  }

  if (id === 'dateReturn') {
    let dateArray = [];
    var curDate = new Date();
    book_json.forEach((v, i) => {
      let vDate = new Date(v.date_return + 'T23:59:59.999Z')
      if (vDate > curDate || v.in_library === "да") {
        dateArray.push(v.id);
      }
    });
    res.end(JSON.stringify(dateArray));
    return;
  }
})

module.exports = router;

```

ajax.js

```

function sendRequest(r_method, r_path, r_args, r_handler){
  var xhr = new XMLHttpRequest();

  if(!xhr){
    alert("ERROR");
    return;
  }

  xhr.onreadystatechange = function (){
    if(xhr.readyState == 4){
      r_handler(xhr);
    }
  };

  xhr.open(r_method, r_path, true);
  xhr.send(null);
}

function inLib(button){
  var Handler = function (xhr){
    let result = JSON.parse(xhr.responseText);
    for(let id of result) {
      //document.getElementById(id).style.visibility = "hidden";
      document.getElementById(id).style.display='none';
    }
  }
}

```

```

        sendRequest("GET", `/filter/${button.id}`, "", Handler);
    }

function allBooks(button){
    var Handler = function (xhr){
        let result = JSON.parse(xhr.responseText);
        for(let id of result) {
            //document.getElementById(id).style.visibility = "visible";
            document.getElementById(id).closest('tr').style.display='table-row';
        }
    }
    sendRequest("GET", `/filter/${button.id}`, "", Handler);
}

function dateReturn(button){
    var Handler = function (xhr){
        let result = JSON.parse(xhr.responseText);
        for(let id of result) {
            //document.getElementById(id).style.visibility = "hidden";
            document.getElementById(id).style.display='none';
        }
    }
    sendRequest("GET", `/filter/${button.id}`, "", Handler);
}

```

style.css

```

body {
    padding: 50px;
    font: 14px "Lucida Grande", Helvetica, Arial, sans-serif;
    background-color: darkgray;
}

a {
    color: #00B7FF;
}

#IN{
    padding: 14px 40px;
    display: block;
    margin-left: auto;
    margin-right: auto;
    margin-top: 50px;
    font-size: 24px;
    color: lightgray;
    background-color: gray;
    border-radius: 8px;
    width: 250px;
}

#imgIn{
    display: block;
    margin-left: auto;
    margin-right: auto;
    margin-top: 50px;
}

#titleIn{
    color: darkslateblue;
    text-align: center;
    font-size: 40px;
}

#titleLib{
    color: darkslateblue;
    font-size: 40px;
    padding: 0;
    margin-top: -25px;
}

#filterDiv{
    position: absolute;
    top: 95px;
}

```

```

    right: 25px;
}

#tableLib{
    border: 4px double #333;
    border-collapse: separate;
    width: 87%;
    border-spacing: 0px;
}

#bookTable{
    border: 4px solid darkblue;
    border-collapse: collapse;
    text-align: left;
}

#getBook{
    float: left;
    margin-left: 320px;
    margin-top: -300px;
}

#returnBook{
    float: left;
    margin-left: 320px;
    margin-top: -270px;
}

#changeBook{
    float: left;
    margin-left: 320px;
    margin-top: -240px;
}

#home{
    margin-top: 19%;
}

.modal{
    display: none;
    background-color: chocolate;

    z-index: 1;
    overflow: auto;
}

```

book.pug

extends layout

```

block content
    script(src="/javascripts/ajax.js")
    div
        table(id="bookTable")
            tr
                th(style="padding: 15px") Книга:
                td(style="padding: 15px") #{name}
            tr
                th(style="padding: 15px") Автор:
                td(style="padding: 15px") #{author}
            tr
                th(style="padding: 15px") Год выпуска:
                td(style="padding: 15px") #{date}
            tr
                th(style="padding: 15px") В наличии:
                td(style="padding: 15px") #{in_library}
            tr
                th(style="padding: 15px") Читатель
                td(style="padding: 15px") #{person}
            tr

```

```

        th(style="padding: 15px") Дата возврата:
        td(style="padding: 15px") #{date_return}
        -var book_id = ID

div(id="getBook")
    p
    if in_library === "нет"
        button(disabled) Взять книгу
    else
        button(onclick="document.getElementById('take').style.display =
\"block\"") Взять книгу
        div(id= "take" class="modal" style="position: fixed")
            div(class="modal-content")
                div(class="modal-header")
                    span(class="button"
onclick="document.getElementById('take').style.display = \"none\"",
style="float: right") &#10006;
                    h2 Читатель
                div(class="modal-body")
                    form(action="read/" + book_id method="POST")
                        p ФИО читателя
                        input(required name="name")
                        p Дата возврата
                        input(required type="date" name="date_return")
                        p
                        button(type="submit") Взять
                div(class="modal-footer")

div(id="returnBook")
    p
    if in_library === "нет"
        form(action="return/" + book_id method="POST")
            button(type="submit") Вернуть книгу
    else
        button(type="submit" disabled) Вернуть книгу
div(id="changeBook")
    p
    button(onclick="document.getElementById('change').style.display =
\"block\"") Редактировать
    div(id= "change" class="modal")
        div(class="modal-content")
            div(class="modal-header")
                span(class="close"
onclick="document.getElementById('change').style.display = \"none\"",
style="float: right") &#10006;
            div(class="modal-body")
                form(action="change/" + book_id method="POST")
                    table
                        tr
                            th Название
                            th Автор
                            th Дата издания
                        tr
                            th
                                input(name="name" )
                            th
                                input(name="author" )
                            th
                                input(name="date" type="date" )
                        p
                        button(type="submit") Изменить
                div(class="modal-footer")
div(id="home")
    p
    form(action="home" method="POST")
        button(type="submit") Назад

```

index.pug

```
extends layout

block content
  h1(id="titleIn")= title
  img(src="../images/bookshelf.png", alt="bookshelf", id="imgIn")
  form(action="/library" method="GET")
    button(type="Submit", id="IN") Войти
```

error.pug

```
extends layout

block content
  h1= message
  h2= error.status
  pre #{error.stack}
```

layout.pug

```
doctype html
html
  head
    title= title
    link(rel='stylesheet', href='/stylesheets/style.css')
  body
    block content
```

library.pug

```
extends layout

block content
  script(src="../javascripts/ajax.js")
  h1(id="titleLib") Книжный шкаф:
  div
    table(id="tableLib", class="w3-table-all ")
      tr
        th Номер
        th Название
        th Автор
        th Год выпуска
        th Наличие
        th Дата возврата
        th Действие
      for item in books
        tr(id = item.id)
          th
            a(href = "book/" + item.id, style="color: black")
              #{item.id}
          th
            a(href = "book/" + item.id, style="color: black")
              #{item.name}
          th #{item.author}
          th #{item.date}
          th #{item.in_library}
          th #{item.date_return}
          form(action="del/" + item.id method = "POST")
            th
              button &times
            - last = item.id
          tr(id = last + 1)
            form(action="new" method="POST")
              th
              th
                input(required name="name")
              th
                input(required name="author")
              th
                input(required type="date" name="date")
              th
              th
              th
```

```
        button +
    p
    div(id="filterDiv")
        div
            p Фильтр:
            div
                p
                button(onclick="allBooks(this)" id="allBooks") Все книги
            div
                p
                button(onclick='inLib(this)' id="inLib" ) Книги в наличии
            div
                p
                button(onclick='dateReturn(this)' id="dateReturn" ) Просрочено
```