**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Web-технологии»**

Тема: Модуль администрирования картин **«АУКЦИОН КАРТИН».**

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

Санкт-Петербург

2021

## Цель работы.

Изучение возможностей применения компилятора Babel, библиотеки jQuery, препроцессора LESS, инструмента выполнения повторяющихся задач GULP, регистрация разработанных модулей, формирование навыков построения структурированных web-приложений, освоение особенностей стандартных библиотек.

## Задание.

Необходимо создать web-приложение, обеспечивающее администрирование аукциона картин: можно выбрать картины для участия в аукционе, определить начальные ставки, перечень участников и параметры аукциона. Основные требования следующие:

1. Перечень доступных картин с описаниями и ссылками на рисунки хранится в JSON-файле.
2. В качестве сервера используется Node.JS с модулем express.
3. Разработка ведется с использованием стандарта не ниже ECMAScript2015.
4. Стили описываются с использованием LESS, при этом используются ключевые методы LESS (переменные, вложенные блоки, миксины, операторы и т. п.).
5. Клиентская часть разрабатывается с использованием jQuery (работа с DOM, AJAX-запросы).
6. Предусмотрена HTML-страница для перечня картин и карточка отдельной картины (название, автор, описание, изображение, начальная цена, минимальный и максимальный шаги аукциона). Предусмотрена возможность 22 редактировать текстовые и числовые параметры, а также включить или исключить картину из участия в предстоящих торгах, загрузить рисунок картины.
7. Предусмотрена HTML-страница для списка потенциальных участников аукциона. Есть возможность добавлять или удалять участников, изменять запас денежных средств.
8. Предусмотрена HTML-страница для настроек аукциона (настройка даты и времени начала аукциона, настройка таймаута продажи картины, настройка интервала времени отсчета до окончания торга по картине, паузы на изучение информации по картине для начала торга по ней).
9. Взаимодействие браузера с сервером осуществляется по протоколу HTTPS.
10. Сборка клиентской части (преобразования less, pug, babel, минификация) осуществляется с использованием GULP.
11. Регистрация и удаление разработанных модулей в npm.
12. Сохранение сформированных настроек в JSON-файл.

## Основные теоретические положения.

LESS – это динамический язык стилей, обеспечивает следующие расширения CSS: переменные, вложенные блоки, миксины, операторы и функции. LESS может работать на стороне клиента или на стороне сервера под управлением Node.js.

jQuery – библиотека JavaScript, предназначенная для упрощения взаимодействии JavaScript и HTML. Библиотека jQuery помогает получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими, предоставляет простой API для работы с AJAX.

Babel – компилятор JavaScript, который позволяет разработчику использовать в своих проектах самые последние стандарты ECMAScript с поддержкой во всех браузерах.

Gulp – это менеджер задач для автоматического выполнения часто используемых задач, написанный на JavaScript. Программное обеспечение поддерживает командную строку для запуска задач, определенных в конфигурационном файле.

## Выполнение работы.

Сначала были написаны шаблоны 4-х веб-страниц на PUG: страницы со списком картин, страницы со списком участников, страницы с настройками, карточка картины. Результат представлен на рисунках 1-4.

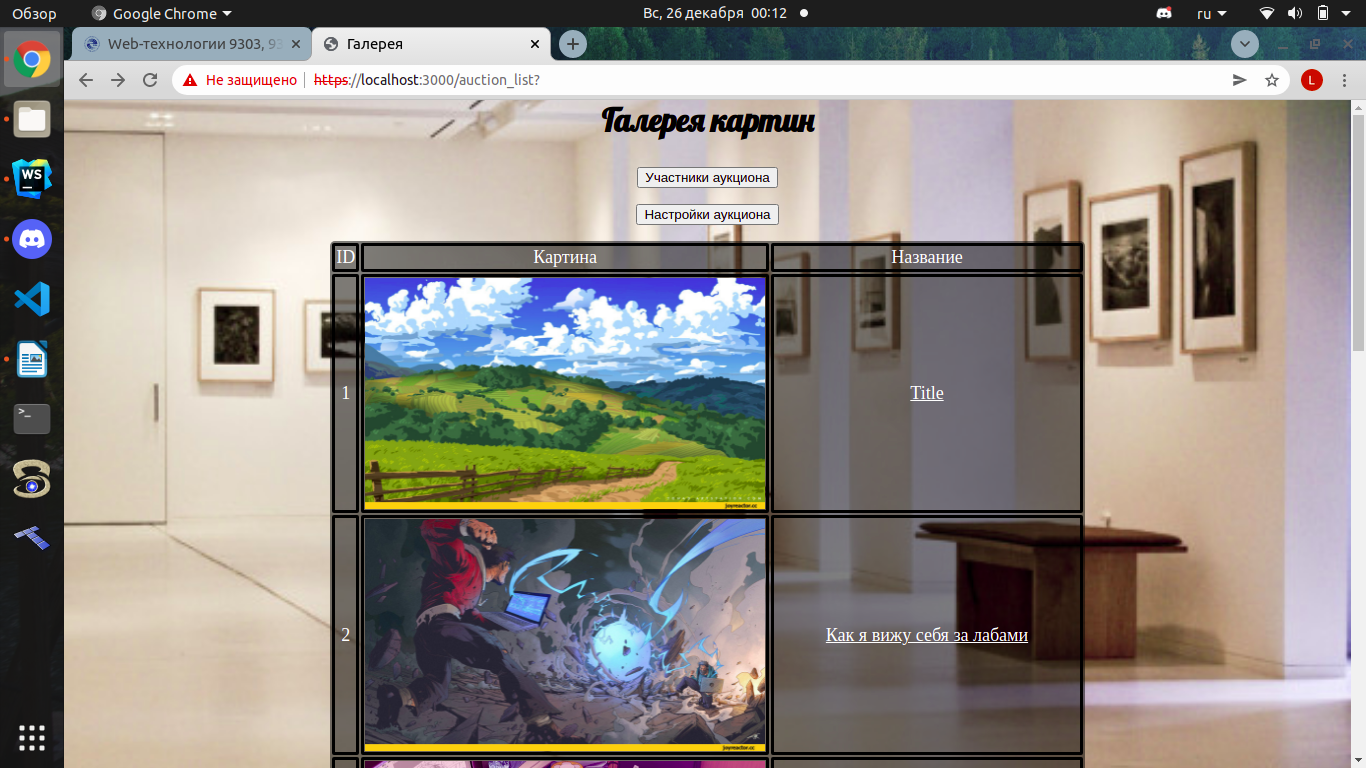


Рисунок 1 – Список книг

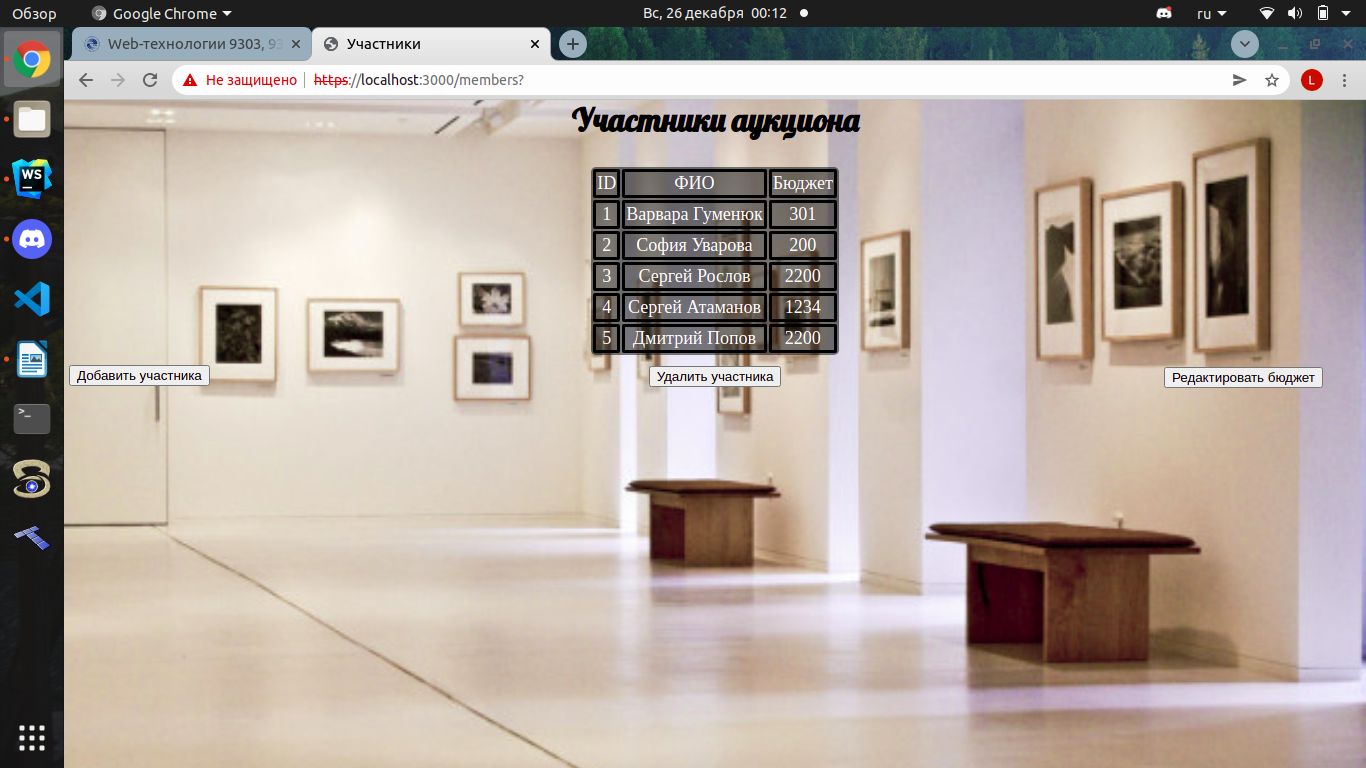


Рисунок 2 – Список участников

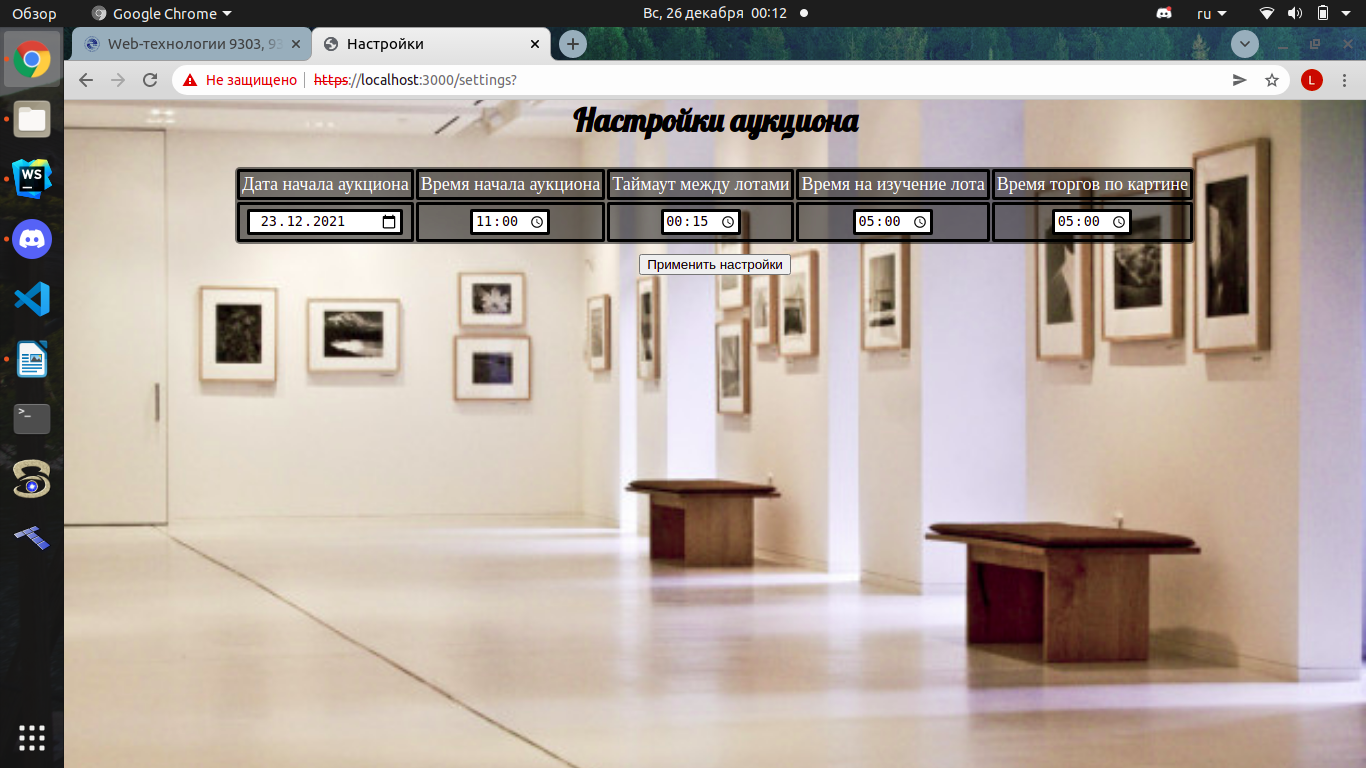


Рисунок 3 – Настройки

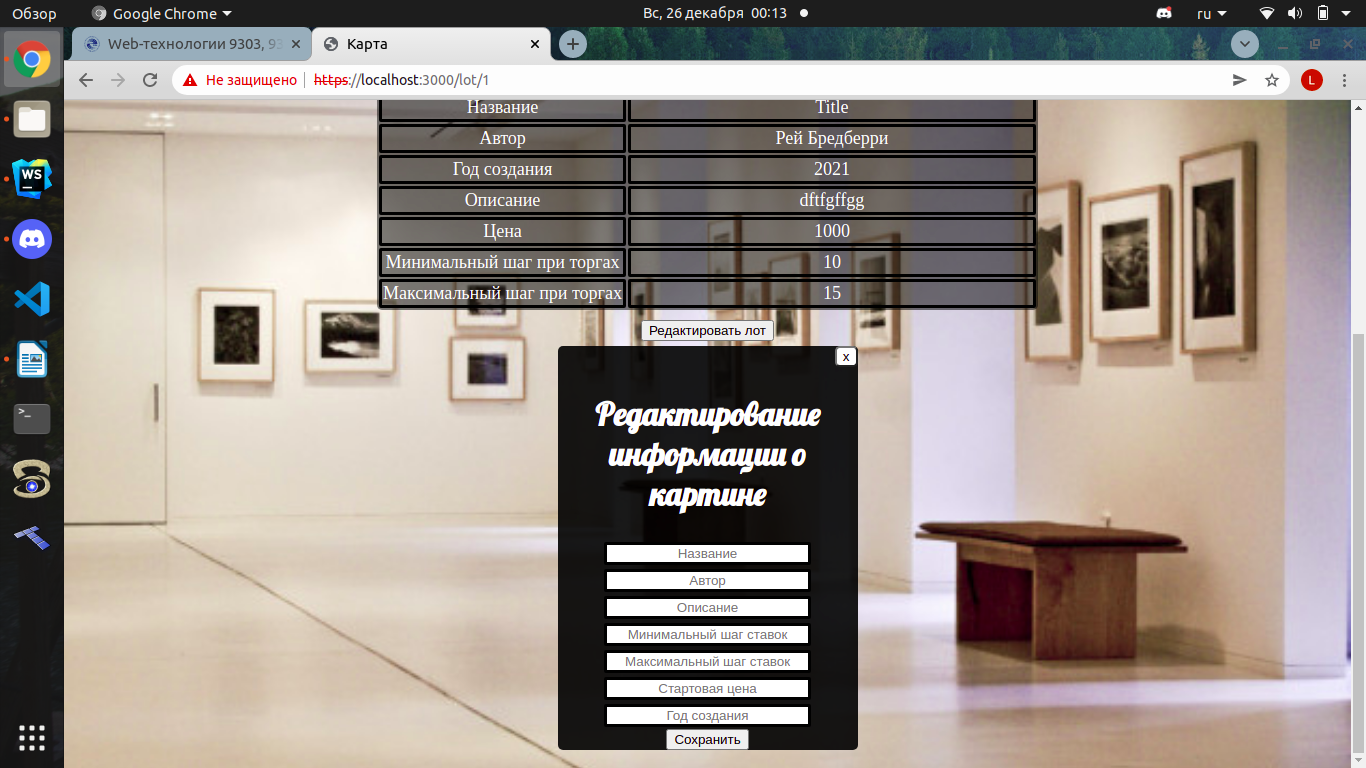


Рисунок 4 – Форма редактирования книги

Далее был написан app.js, который представляет из себя серверную часть приложения. Для каждой страницы были разработаны собственные стили на языке LESS, которые преобразовывались в CSS при помощи сборщика GULP.

Вся данные о картинах, участниках и настройках аукциона хранятся в соответствующих JSON-файлах.

После написания полноценного приложения, разработанный модуль был зарегистрирован в npm.

## Выводы.

Изучили возможности применения компилятора Babel, библиотеки jQuery, препроцессора LESS, инструмента выполнения повторяющихся задач GULP, зарегистрировали разработанный модуль, сформировали навыки построения структурированных web-приложений, освоили особенности стандартных библиотек.

# Приложение А Исходный код программы

Название файла: app.js

const express = require('express');

const app = express();

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

var fs = require('fs');

var https = require('https');

httpsOptions ={

key: fs.readFileSync('./key/server.key'),

cert: fs.readFileSync('./key/server.crt')

}

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('/JSON', express.static('JSON'));

app.use('/jquery\_scripts', express.static('jquery'));

app.use('/', route);

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

https.createServer(httpsOptions, app).listen(3000, ()=>{

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

});

Название файла: routes.js

var express = require('express');

var router = express.Router();

var galary = require('./JSON/auction.json');

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

const fs = require('fs');

var person = require('./JSON/members.json')

var settings = require('./JSON/settings.json')

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

res.render('index');

});

router.get('/auction\_list', (req, res, next) => {

res.render("auction\_list.pug", {img\_list: galary});

next();

});

router.get('/add\_lot', (req, res,next)=>{

let new\_lot = {

"id": `${galary.length+1}`,

"title": req.query.title,

"author": req.query.author,

"path": req.query.path,

"min\_step": req.query.min\_step,

"max\_step": req.query.max\_step,

"info": req.query.info,

"price": req.query.price,

"year": req.query.year

};

galary.push(new\_lot);

fs.writeFileSync("./JSON/auction.json", JSON.stringify(galary, null, 2));

res.end(JSON.stringify(new\_lot));

next();

})

router.get('/delete\_lot', (req, res,next)=>{

let id = req.query.id\_del;

for(item of galary){

if(item.id === id){

galary.splice(id-1, 1);

}

}

let n=1;

for(item of galary){

item.id = `${n}`;

n+=1;

}

fs.writeFileSync("./JSON/auction.json", JSON.stringify(galary, null, 2));

res.end(id);

//res.redirect('/auction\_list');

next();

})

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

id = req.params.num;

res.render("lot.pug", {lot: galary[id-1]});

next();

});

router.post('/edit\_lot/:num', (req, res, next) => {

let id = req.params.num;

for (item of galary){

if(item.id === id){

item.title = req.body.title;

item.author = req.body.author;

item.info = req.body.info;

item.min\_step = req.body.min\_step;

item.max\_step = req.body.max\_step;

item.price = req.body.price;

item.year = req.body.year;

console.log(req.body.year)

break;

}

}

fs.writeFileSync("./JSON/auction.json", JSON.stringify(galary, null, 2));

res.redirect(`/lot/`+id);

next();

});

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

res.render("members.pug", {members: person});

next();

});

router.get('/add\_person', (req, res,next)=>{

let new\_person= {

"id": `${person.length+1}`,

"name": req.query.name,

"money": req.query.money

};

person.push(new\_person);

fs.writeFileSync("./JSON/members.json", JSON.stringify(person, null, 2));

res.end(JSON.stringify(new\_person));

next();

})

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

let id = req.query.id\_pers;

for(item of person){

if(item.id === id){

person.splice(id-1, 1);

}

}

let n=1;

for(item of person){

item.id = `${n}`;

n+=1;

}

fs.writeFileSync("./JSON/members.json", JSON.stringify(person, null, 2));

res.end(id);

//res.redirect('/auction\_list');

})

router.post('/edit\_money', (req, res, next) => {

for (item of person){

if(item.id === req.body.id){

item.money = req.body.new\_money;

break;

}

}

fs.writeFileSync("./JSON/members.json", JSON.stringify(person, null, 2));

res.redirect('/members');

next();

});

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

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

next();

});

router.get('/set\_settings', (req, res, next) => {

let new\_settings={

"date\_s": req.query.date\_s,

"time\_s": req.query.time\_s,

"timeout": req.query.timeout,

"pause": req.query.pause,

"torg": req.query.torg

}

fs.writeFileSync("./JSON/settings.json", JSON.stringify(new\_settings, null, 2));

res.end(JSON.stringify(new\_settings));

next();

});

module.exports = router;