**ОТЧЕТ**

по лабораторной работе №6

по курсу «Разработка Web-приложений»

Выполнили:

студенты группы 16ВВ2

Аккуратнов И.М.

Кожаев В.Д.

Принял:

Дубравин А.В.

Пенза 2020

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

### Название

### 6.Реализация REST API

### Цель работы

Необходимо доработать лабораторную работу №5 реализовав в ней поддержку REST API и

реализовать два обработчика:

1. Получает запрос с названием животного и дополнительными параметрами и подбирает по

этому запросу подходящее изображение из числа имеющихся;

2. По имени и параметрам изображения возвращает картинку

Вариант Задание

1. Животные: антилопа, акула, ара.

### Листинг

**Файл** *start.html*

<!DOCTYPE *html*>  
<html *lang*="ru">  
 <head>  
 <meta *charset*="UTF-8">  
 <title>Лаборатоная 4</title>  
 <link *rel*="icon" *href*="http://localhost:8888/favicon.ico" *type*="image/x-icon">  
 </head>  
 <body>  
 <form *action*="/upload" *method*="post">  
 Введите название картинки с номером  
 <input *name*="name" *type*="text" *required pattern*="(shark|antelope|macaw)( [1-9])?$"  
 *title*="Можно ввести только shark, antelope, macaw с цифрой от 1 до 9"/>  
 <br/>  
 Введите ширину  
 <input *name*="width" *type*="text" *min* = "100" *max* = "1500"  
 *title*="Диапазон от 100 до 1500 пикселей"/>  
 Введите высоту  
 <input *name*="height" *type*="text" *min* = "100" *max* = "1500"  
 *title*="Диапазон от 100 до 1500 пикселей"/>  
 <br/>  
 Значения цвета  
 <input *name*="colour" *type*="text" *pattern*="(blue|red|green)?$"  
 *title*="Можно ввести только blue, red, green"/>  
 <br/>  
 <input *type*="submit" *value*="Submit" />  
 </form>  
 </body>  
</html>

**Файл** *api.js*

*const express* = require("express");  
*const* router = *express*.Router();  
*const* path = require('path');  
*const* fs = require("fs");  
*const* requestHandlers = require(path.resolve(\_\_dirname, 'requestHandlers.js'));  
  
router.get("/start", (req, res)=>{  
 *const* page = fs.*readFileSync*(path.resolve(\_\_dirname, 'Start.html'));  
 res.write(page);  
});  
  
router.post("/upload", (req, res)=>{  
 *console*.log(req.body);  
 *if* (req.body.name.split(' ').length === 2)  
 requestHandlers.firstHandler(req.body, res);  
 *else* requestHandlers.secondHandler(req.body, res);  
});  
  
  
module.exports = router;

**Файл** *index.js*

*const express* = require("express");  
*const* app = *express*();  
*var* bodyParser = require('body-parser');  
  
app.use(bodyParser.json());  
app.use(bodyParser.urlencoded({extend:*true*}));  
  
app.use("/", require("./api"));  
  
  
app.listen(8888, ()=>{  
 *console*.log("Server listening 8888 port");  
});

**Файл** *router.js*

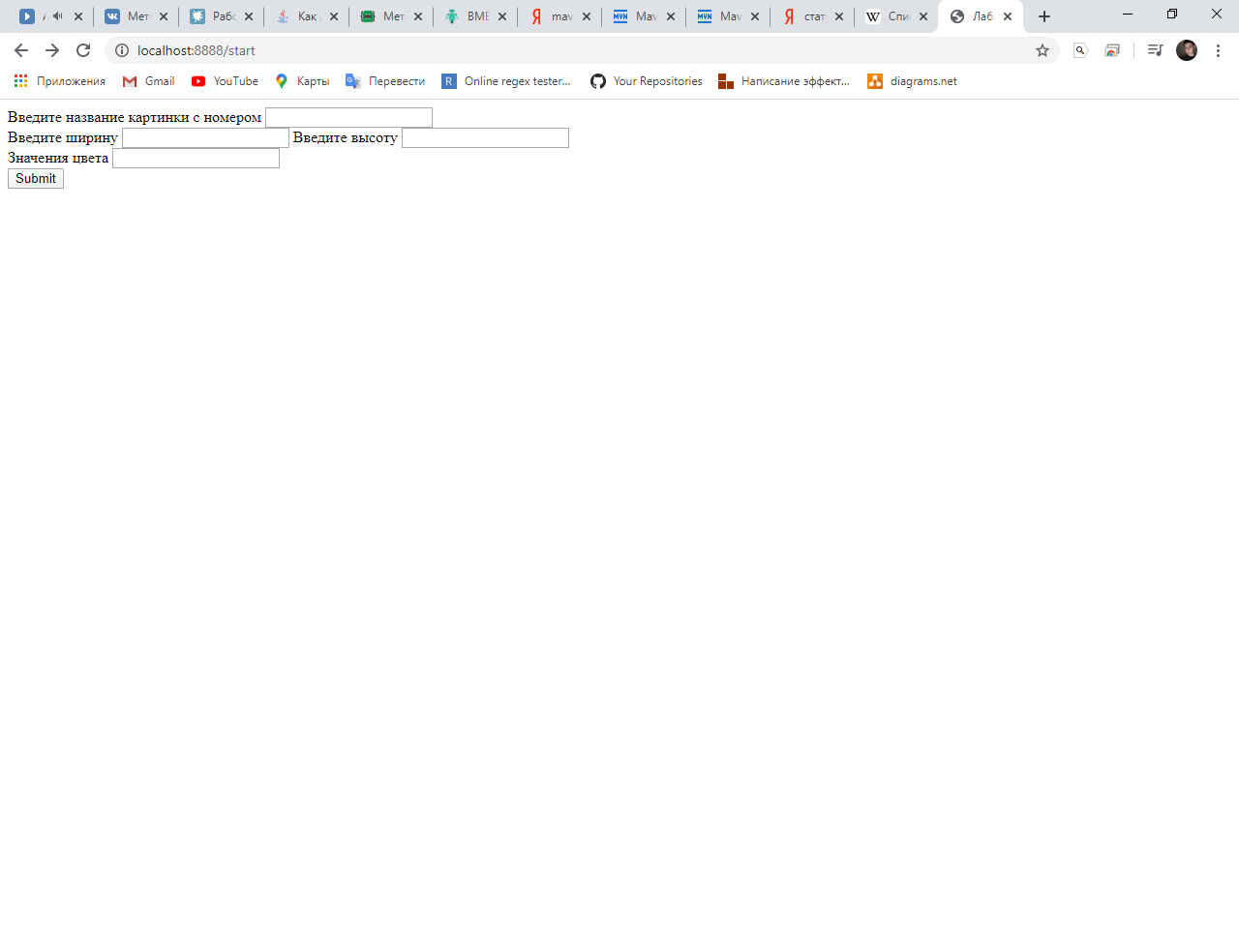
*const* querystring = require("querystring");  
  
*function* route(handle, SiteUrl, response, postData) {  
 *console*.log("About to route a request for " + SiteUrl.pathname);  
 *if* (*typeof* handle[SiteUrl.pathname] === 'function') {  
 handle[SiteUrl.pathname](response, postData, querystring.*parse*(SiteUrl.query).key);  
 } *else* {  
 *console*.log("No request handler found for " + SiteUrl.pathname);  
 response.writeHead(404, {"Content-Type": "text/plain"});  
 response.write("404 Not found");  
 response.end();  
 }  
}  
exports.route = route;

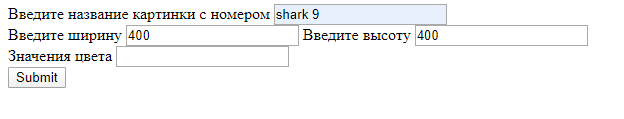
**Файл** *requestHandlers.js*

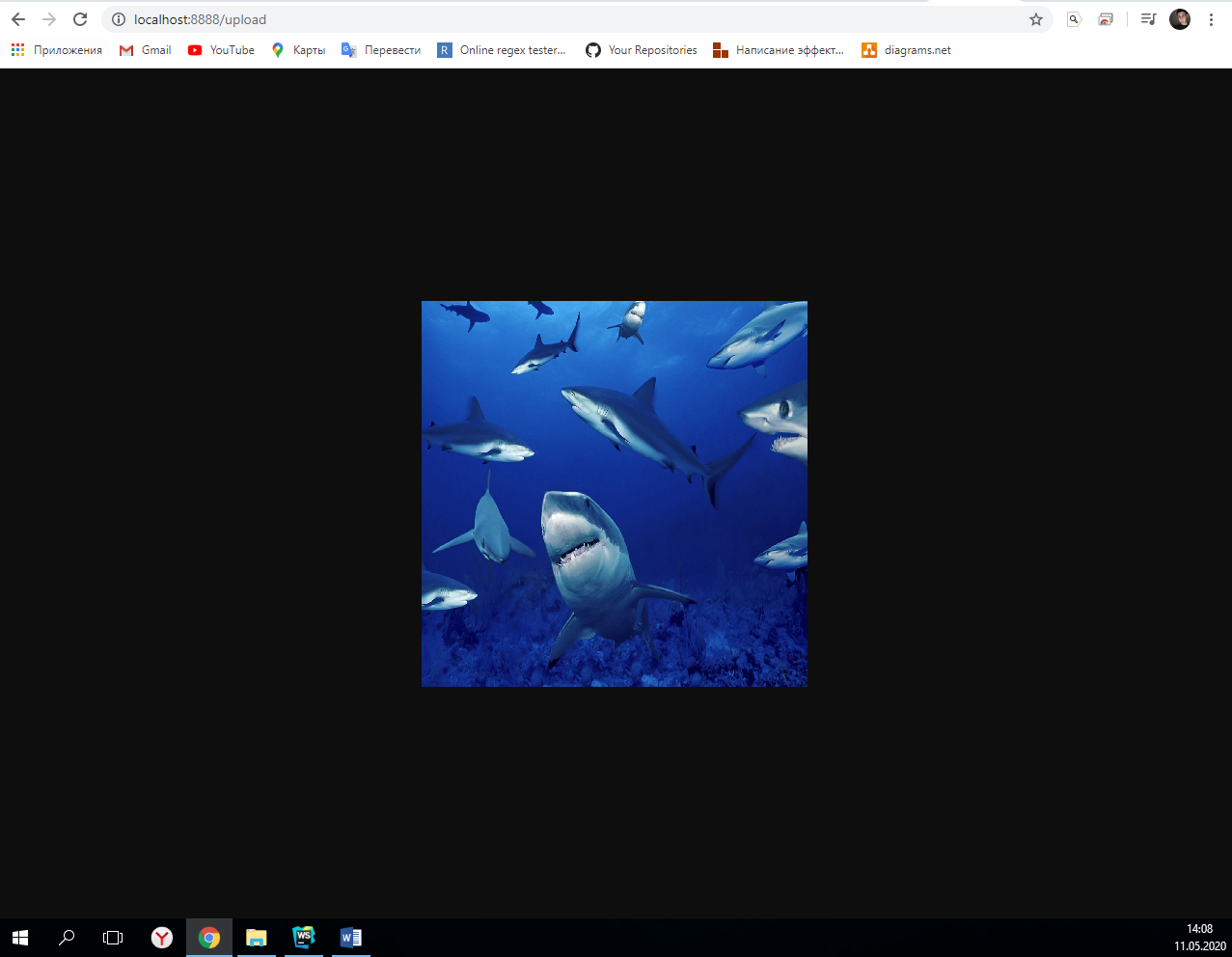
*const* fs = require("fs");  
*const* { Image } = require('image-js');  
*const* path = require('path');  
  
*async function* firstHandler(body, res) {  
 *let* name = body.name.split(' ');  
 *let* img = *await* Image.*load*(path.resolve(path.dirname(\_\_dirname), `IMG/${name[0]}s/${name[0]}-${name[1]}.jpg`));  
 *if* (!(body.weight === "") && !(body.height === "")){  
 img = img.resize(  
 {width: body.width,  
 height : body.height}  
 );  
 }  
 *if* (!(body.width === "") && (body.height === "")){  
 img = img.resize({width: body.width});  
 }  
 *if* ((body.width === "") && !(body.height === "")){  
 img = img.resize({height: body.height});  
 }  
  
 *await* img.save('toSend.png');  
 res.sendfile('toSend.png');  
}  
  
*async function* secondHandler(body, res) {  
 *let* name = body.name;  
 *let* find = *false*;  
 *let* i = 1;  
 *let* img;  
 *while* (find === *false* && i<10){  
 img = *await* Image.*load*(path.resolve(path.dirname(\_\_dirname), `IMG/${name}s/${name}-${i}.jpg`));  
 *if* (findColor(img) === body.colour) find = *true*;  
 i++;  
 }  
 *if* (find === *false*) {  
 res.sendStatus(404);  
 *return null*;  
 }  
  
 *if* (!(body.weight === "") && !(body.height === "")){  
 img = img.resize(  
 {width: body.width,  
 height : body.height}  
 );  
 }  
 *if* (!(body.width === "") && (body.height === "")){  
 img = img.resize({width: body.width});  
 }  
 *if* ((body.width === "") && !(body.height === "")){  
 img = img.resize({height: body.height});  
 }  
  
 *await* img.save('toSend.png');  
 res.sendfile('toSend.png');  
}  
  
*function* findColor(img){  
 *let* pixelCount = 0, redPixel = 0, greenPixel = 0, bluePixel = 0;  
 *for* (*let* i = 0; i<img.size; i=i+4){  
 pixelCount++;  
 *let* pixel = img.getPixel(i);  
 redPixel += pixel[0];  
 greenPixel += pixel[1];  
 bluePixel += pixel[2];  
 }  
 *if* (redPixel > greenPixel && redPixel > bluePixel) *return* 'red';  
 *if* (greenPixel > redPixel && greenPixel > bluePixel) *return* 'green';  
 *if* (bluePixel > greenPixel && bluePixel > redPixel) *return* 'blue';  
}  
  
exports.firstHandler = firstHandler;  
exports.secondHandler = secondHandler;

*const* fs = require("fs");  
*const* { Image } = require('image-js');  
*const* path = require('path');  
  
*async function* firstHandler(body, res) {  
 *let* name = body.name.split(' ');  
 *let* img = *await* Image.*load*(path.resolve(path.dirname(\_\_dirname), `IMG/${name[0]}s/${name[0]}-${name[1]}.jpg`));  
 *if* (!(body.weight === "") && !(body.height === "")){  
 img = img.resize(  
 {width: body.width,  
 height : body.height}  
 );  
 }  
 *if* (!(body.width === "") && (body.height === "")){  
 img = img.resize({width: body.width});  
 }  
 *if* ((body.width === "") && !(body.height === "")){  
 img = img.resize({height: body.height});  
 }  
  
 *await* img.save('toSend.png');  
 res.sendfile('toSend.png');  
}  
  
*async function* secondHandler(body, res) {  
 *let* name = body.name;  
 *let* find = *false*;  
 *let* i = 1;  
 *let* img;  
 *while* (find === *false* && i<10){  
 img = *await* Image.*load*(path.resolve(path.dirname(\_\_dirname), `IMG/${name}s/${name}-${i}.jpg`));  
 *if* (findColor(img) === body.colour) find = *true*;  
 i++;  
 }  
 *if* (find === *false*) {  
 res.sendStatus(404);  
 *return null*;  
 }  
  
 *if* (!(body.weight === "") && !(body.height === "")){  
 img = img.resize(  
 {width: body.width,  
 height : body.height}  
 );  
 }  
 *if* (!(body.width === "") && (body.height === "")){  
 img = img.resize({width: body.width});  
 }  
 *if* ((body.width === "") && !(body.height === "")){  
 img = img.resize({height: body.height});  
 }  
  
 *await* img.save('toSend.png');  
 res.sendfile('toSend.png');  
}  
  
*function* findColor(img){  
 *let* pixelCount = 0, redPixel = 0, greenPixel = 0, bluePixel = 0;  
 *for* (*let* i = 0; i<img.size; i=i+4){  
 pixelCount++;  
 *let* pixel = img.getPixel(i);  
 redPixel += pixel[0];  
 greenPixel += pixel[1];  
 bluePixel += pixel[2];  
 }  
 *if* (redPixel > greenPixel && redPixel > bluePixel) *return* 'red';  
 *if* (greenPixel > redPixel && greenPixel > bluePixel) *return* 'green';  
 *if* (bluePixel > greenPixel && bluePixel > redPixel) *return* 'blue';  
}  
  
exports.firstHandler = firstHandler;  
exports.secondHandler = secondHandler;

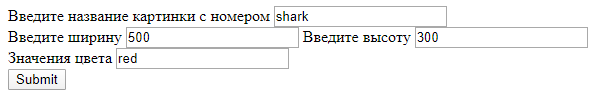
1. Окно поиска

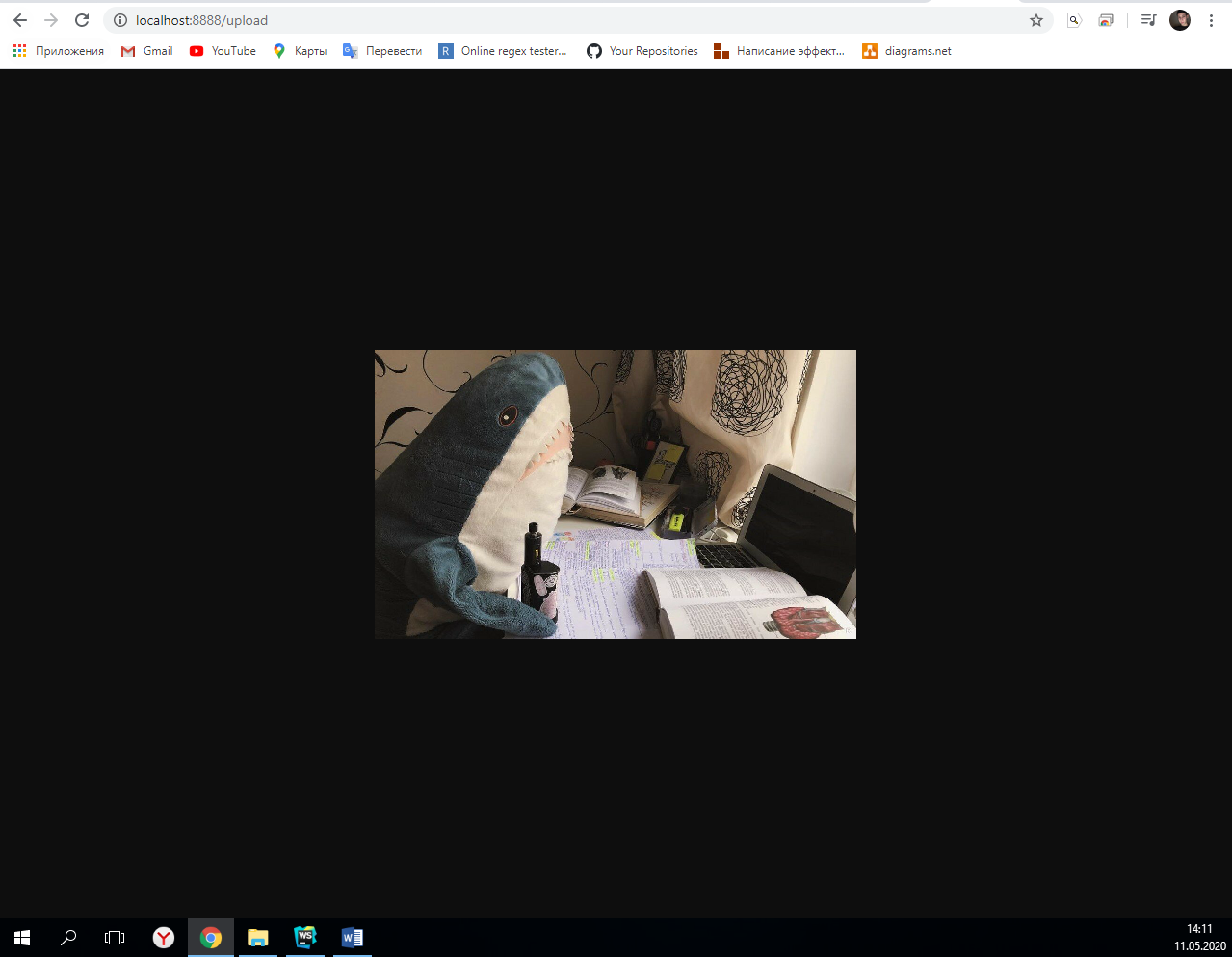


1. Первый вариант запроса. Ввести название животного и номер картинки в первую строку. Ввести ширину и высоту, если хотите не стандартный размер. 

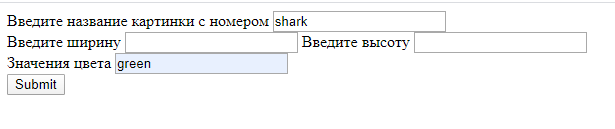


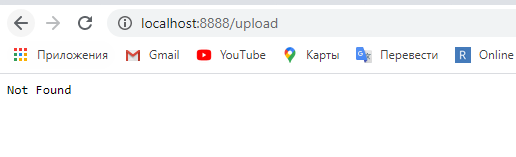
1. Второй вариант, ввести название животного, без цифры. Ввести ширину и высоту если хотите не стандартный размер. Ввести основной цвет изображение (red, green, blue).





1. Если ввести основной цвет, но изображения с таким цветом не будет, выведет ошибку от сервера.





### Выводы

### В ходе выполнения лабораторной работы реализовали REST API, работали с изображения, их размерами и основными цветами.