Пенза 2020

Выполнили:

студенты группы 16ВВ1

Кривошеев С.В.

Щетинин К.А.

Принял:

к.т.н., доцент Дубравин А.В.

к.т.н., доцент Карамышева Н.С.

**ОТЧЕТ**

по лабораторной работе №6

по курсу «Web-программирование»

на тему «Реализация REST API»

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

### Название

Реализация REST API.

### Цель работы

Реализовать RESTful api.

### Лабораторное задание

aНеобходимо 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.

Параметры должны передаваться на сервер в виде POST запроса, в теле которого должен находиться JSON.

Ответ от сервера так же должен возвращаться в формате JSON.

Необходимо расширить набор параметров изображения животного, добавив в него не обязательные значения цвета и размера ожидаемого изображения (количество пикселей по горизонтали и по вертикали).

В ответ на запрос сервер должен сформировать ответ, в котором перечислить переданные параметры, а недостающие заполнить случайными значениями (случайные значения должны выбираться из разумного диапазона), а также должен добавить поле с именем файла, соответствующего заданным параметрам.

|  |  |
| --- | --- |
| Вариант | Задание |
| 4 | Животные: гиена, газель, геккон. |

### Порядок выполнения работы

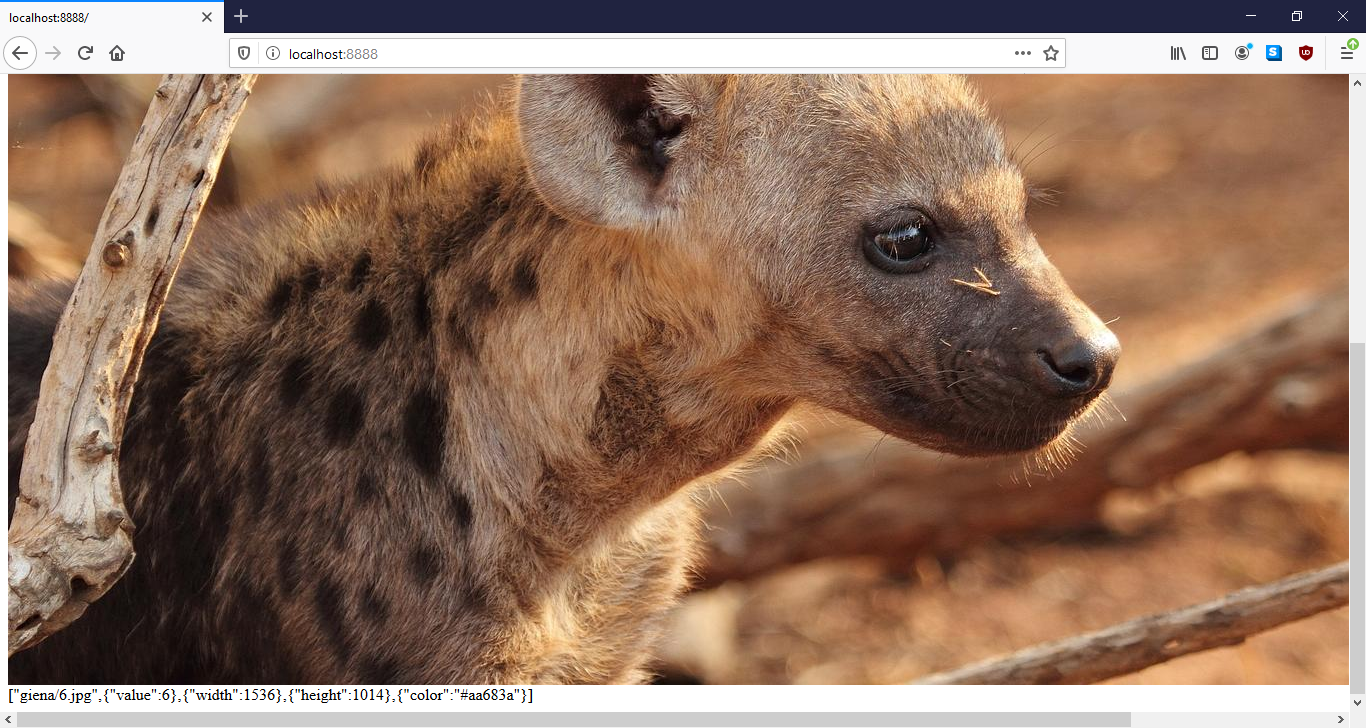


Рисунок 1 – вывод случайного изображения при вводе giena

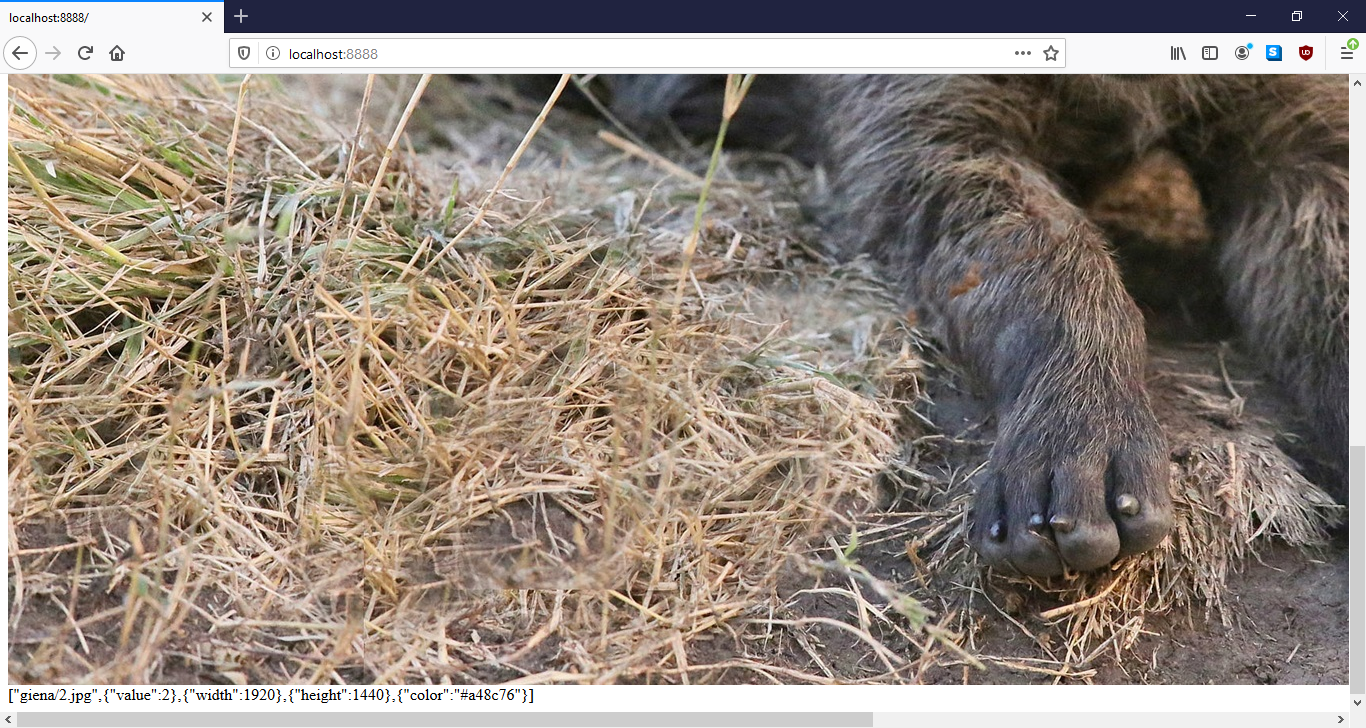


Рисунок 2 – вывод изображения giena с указанием номера

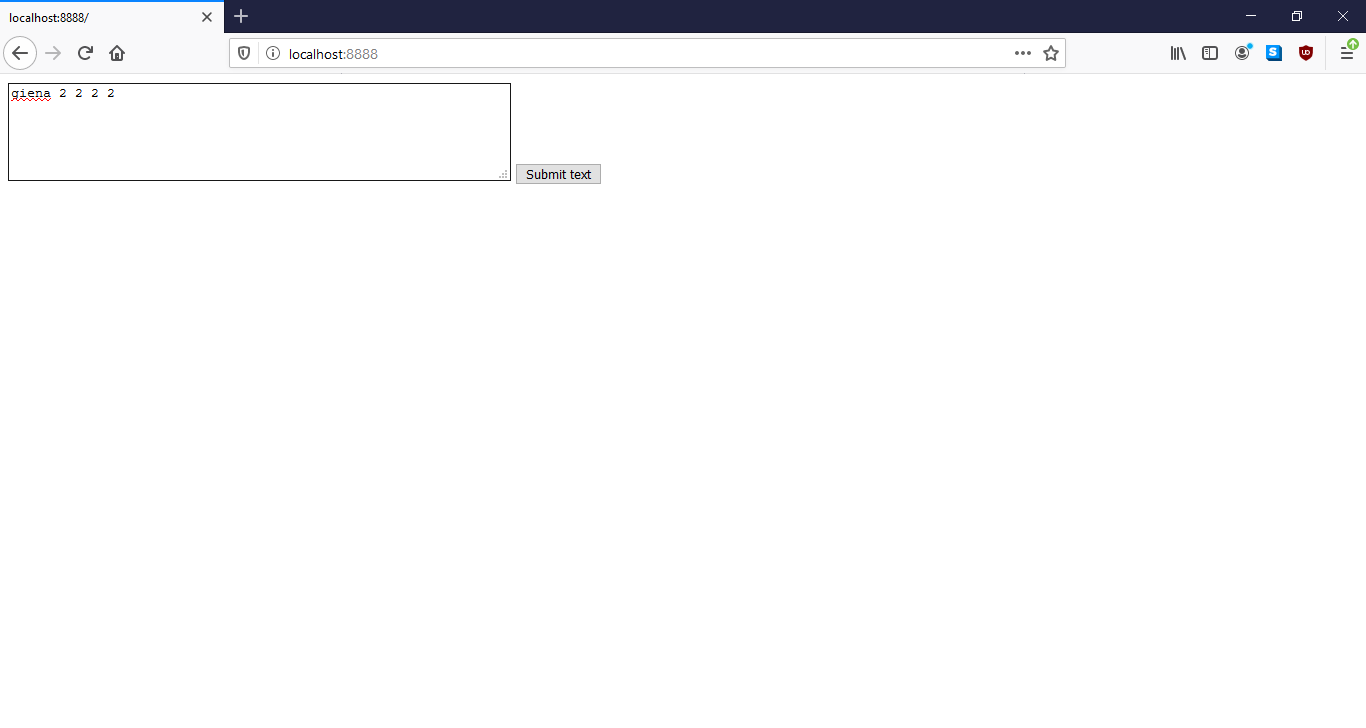


Рисунок 3 – ввод сообщения

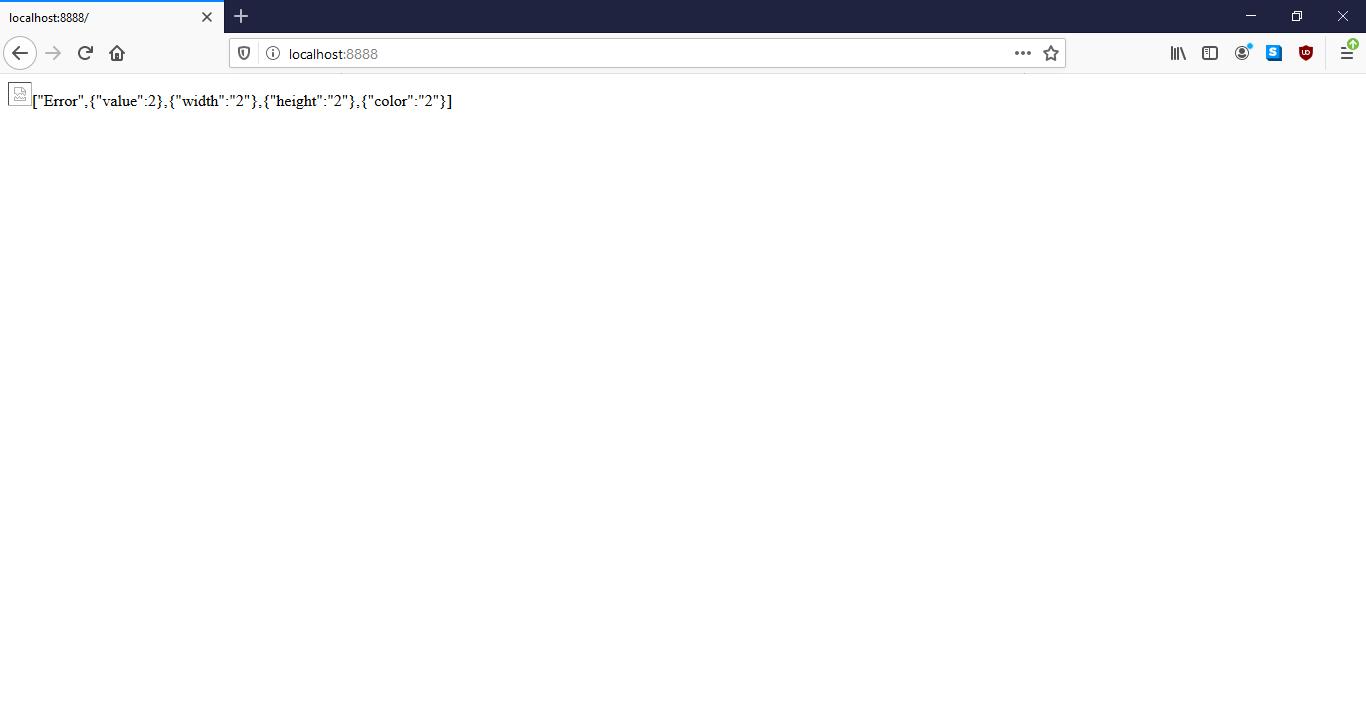


Рисунок 4 – вывод сообщения об ошибке

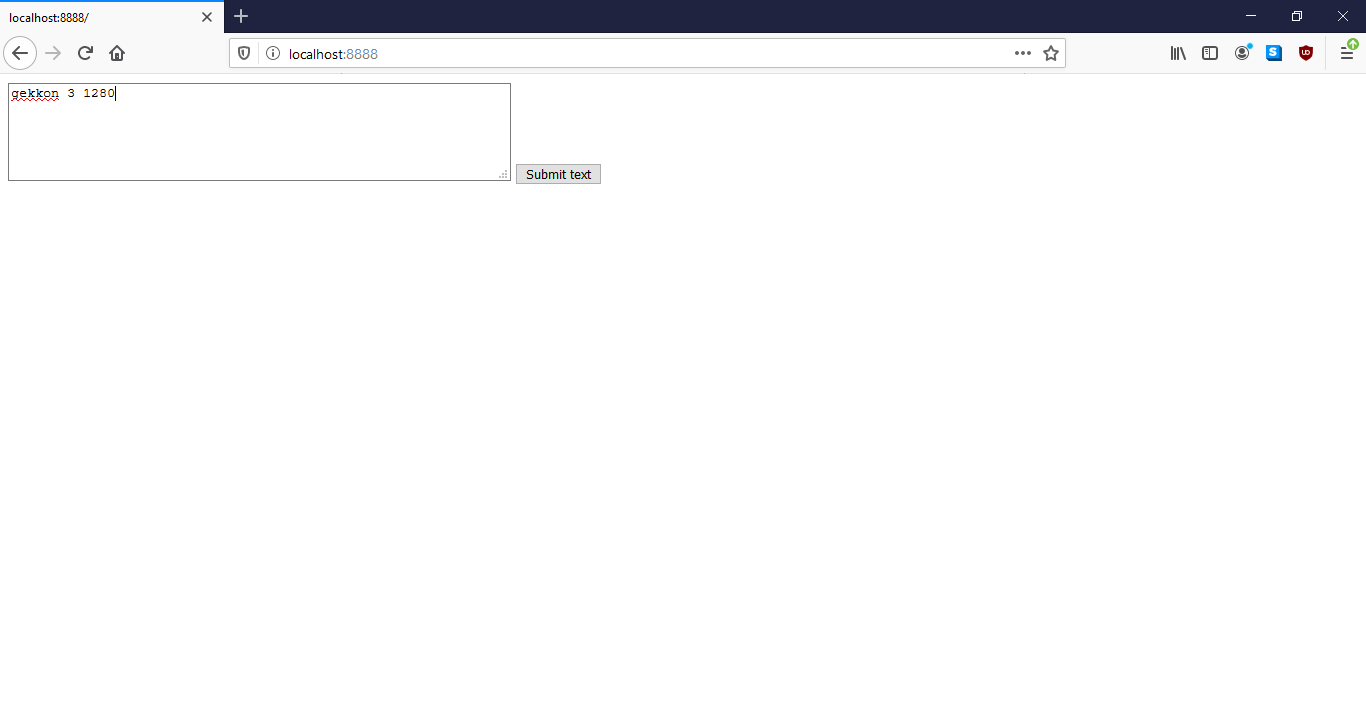


Рисунок 5 - вводсообщения

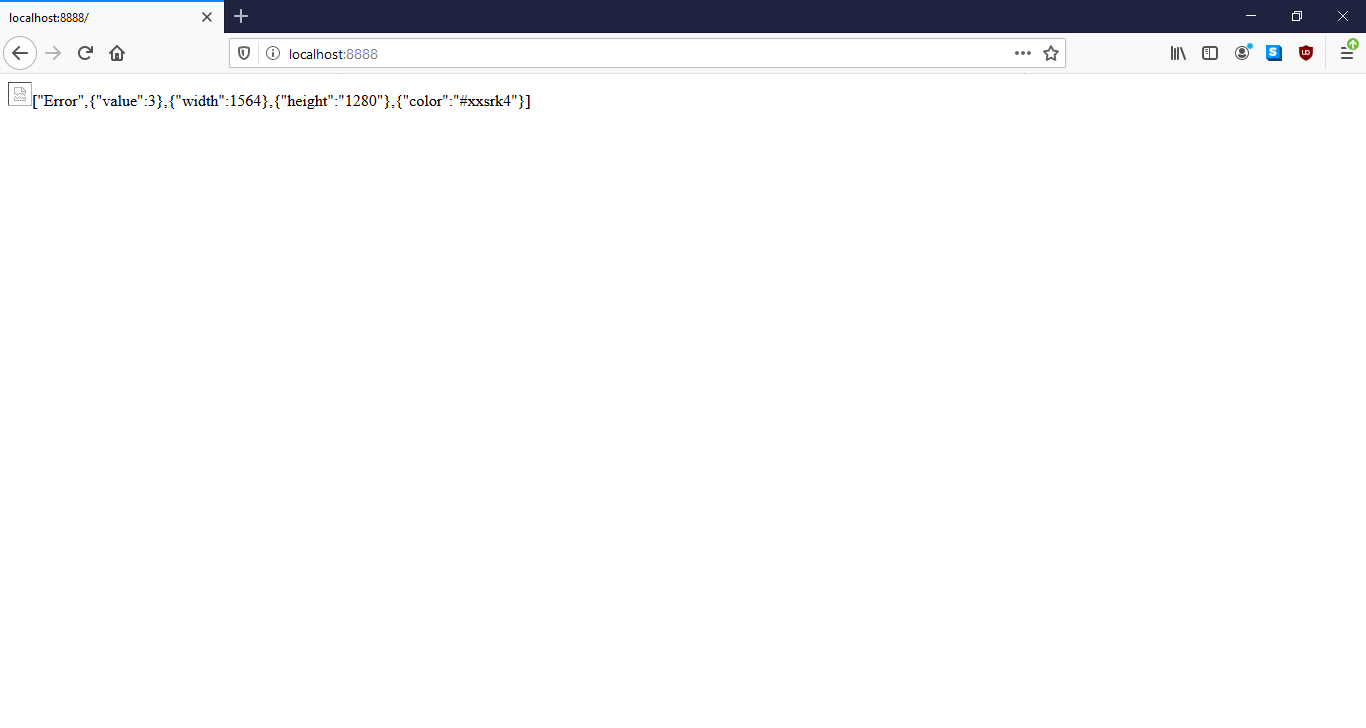


Рисунок 6 - вывод сообщения об ошибки со случайно заполненными данными

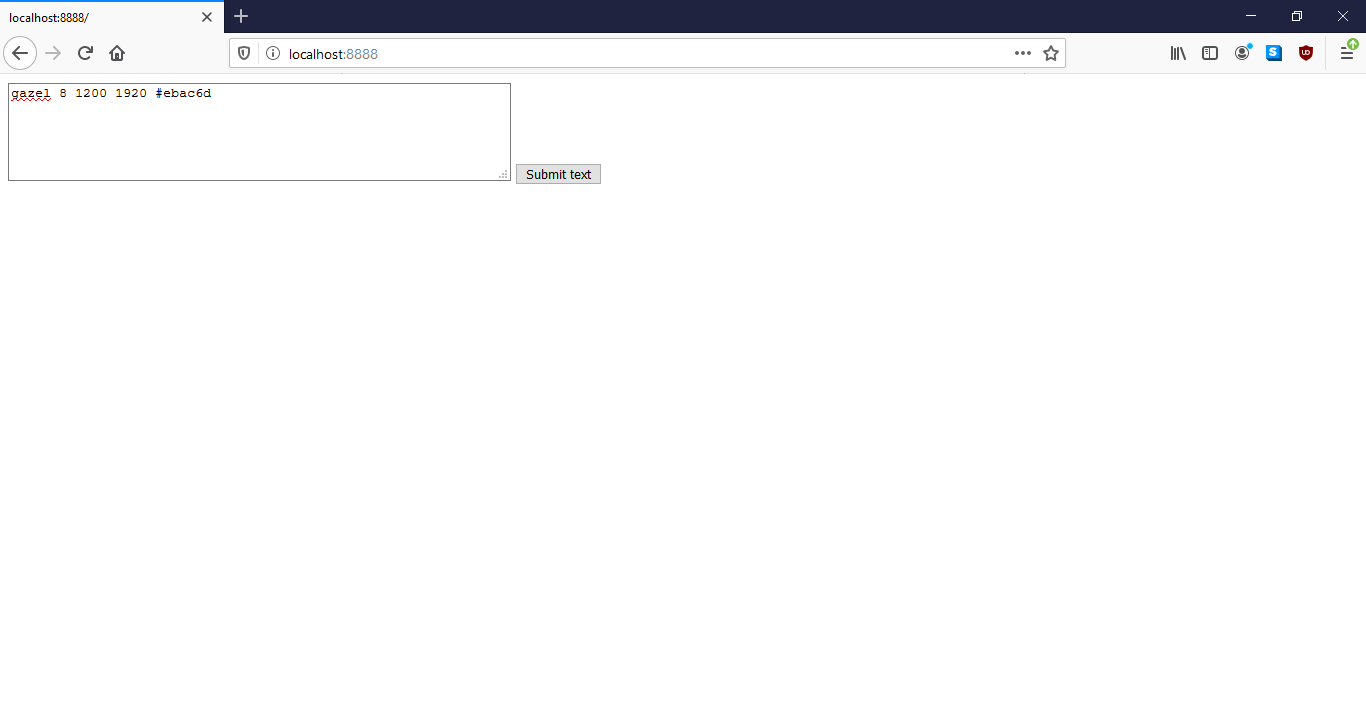


Рисунок 7 – ввод всех данных изображения

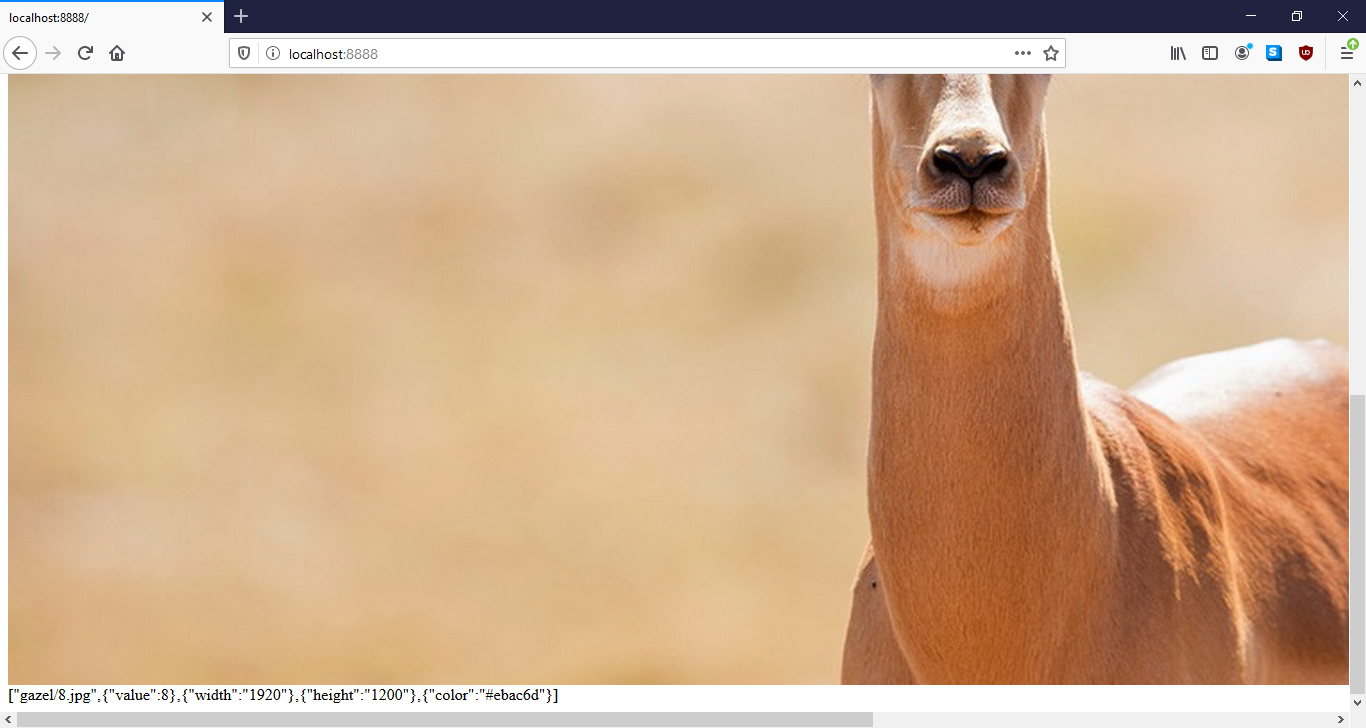


Рисунок 8 – вывод изображения по введенным данным

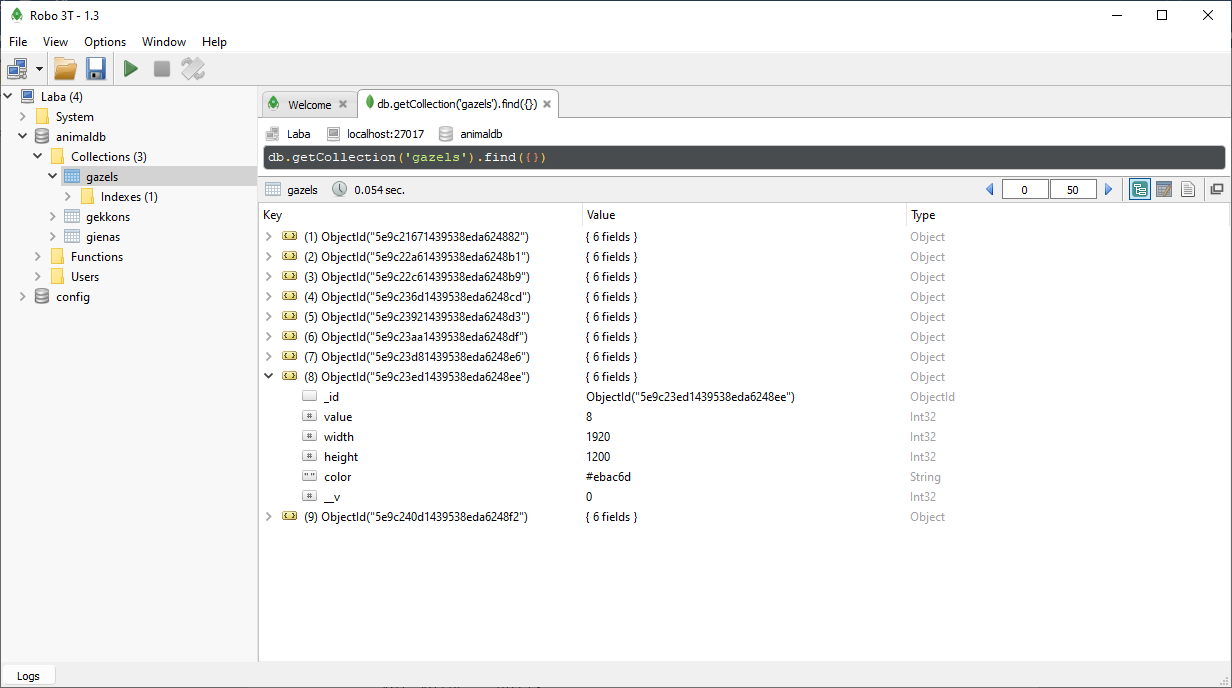


Рисунок 9 – проверка данных в БД MongoDB

### Листинг

#### six.html

<html>

<head>

<meta http-equiv="Content-Type" content="text/html;

charset=UTF-8" />

<script>

let animanls = ["giena", "gazel", "gekkon"];

var value = null;

var height = null;

var width = null;

var color = null;

async function validate() {

if (dannye.value.split(" ")[2]!=null)

{

height =dannye.value.split(" ")[2]

}

if (dannye.value.split(" ")[3]!=null)

{

width =dannye.value.split(" ")[3]

}

if (dannye.value.split(" ")[4]!=null)

{

color =dannye.value.split(" ")[4]

}

for (let i = 0; i < animanls.length; i++) {

if (dannye.value == animanls[i]) {

var msgJSON = jsonpost(dannye.value, value, height, width, color);

const response = await fetch("/" + dannye.value, {

method: "POST",

mode: 'no-cors',

body: JSON.stringify(msgJSON),

headers: {

'Content-Type': 'application/json'

}

})

const json = await response.json();

console.log(json);

var pjson = JSON.stringify(json);

console.log(pjson);

document.write('<img id=myImg src= "">');

document.getElementById("myImg").src = "public/"+json[0];

document.write(pjson);

return;

}

for (let j = 0; j < 9; j++) {

if (dannye.value.split(" ")[0] == animanls[i] & dannye.value.split(" ")[1] == j) {

var animal = dannye.value.split(" ")[0];

var msgJSON = jsonpost(animal, j, height, width, color);

const response = await fetch("/" + animal, {

method: "POST",

mode: 'no-cors',

body: JSON.stringify(msgJSON),

headers: {

'Content-Type': 'application/json'

}

})

const json = await response.json();

console.log(json);

var pjson = JSON.stringify(json);

console.log(pjson);

document.write('<img id=myImg src= "">');

document.getElementById("myImg").src = "public/"+json[0];

document.write(pjson);

return;

}

}

}

alert("Error Data");

}

function jsonpost(name, value, height, width, color) {

let animanlsJSON = {

name: name,

value: value,

height: height,

width: width,

color: color

};

return animanlsJSON;

}

</script>

</head>

<body>

<textarea id="dannye" name="text" rows="5" cols="60"></textarea>

<input type="submit" value="Submit text" onclick="validate()">

</body>

</html>

#### index.JS

const express = require("express");

const mongoose = require("mongoose");

var bodyParser = require('body-parser')

var path = require('path')

var serveStatic = require('serve-static');

const app = express();

mongoose.connect("mongodb://localhost/animaldb")

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

app.use(bodyParser.text());

app.use("/",require("./router"));

app.listen(8888,()=>{

console.log("server is listening");

})

#### router.JS

const express = require("express");

var fs = require("fs");

var sizeOf = require('image-size');

const Gazel = require("./gazel");

const Gekkon = require("./gekkon");

const Giena = require("./giena");

var path = require('path')

var serveStatic = require('serve-static');

const router = express.Router();

var navivod;

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

console.log("GET /");

var body = fs.readFileSync('six.html');

res.writeHead(200, {

"Content-Type": "text/html"

});

res.write(body);

res.end();

});

router.post("/gazel", async (req, res) => {

var result = req.body;

console.log(result);

var acjson = JSON.parse(result);

if (acjson.value == null) {

acjson.value = Math.floor(Math.random() \* 9 + 1);

console.log(acjson.value);

}

const data = await Gazel.find({

value: acjson.value

}).exec();

console.log(data[0].color);

navivod = proverka("gazel", acjson, data[0]);

console.log("POST /gazel")

res.redirect("/gazel")

})

router.get("/gazel", async (req, res) => {

console.log(req.body);

res.json(navivod);

console.log("GET /gazel")

navivod = "";

})

router.post("/gekkon", async (req, res) => {

var result = req.body;

console.log(result);

var acjson = JSON.parse(result);

if (acjson.value == null) {

acjson.value = Math.floor(Math.random() \* 9 + 1);

console.log(acjson.value);

}

const data = await Gekkon.find({

value: acjson.value

}).exec();

console.log(data[0].color);

navivod = proverka("gekkon", acjson, data[0]);

res.redirect("/gekkon");

console.log("POST /gekkon");

})

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

console.log(req.body);

console.log(navivod);

res.json(navivod);

console.log("GET /gekkon");

navivod = "";

})

router.post("/giena", async (req, res) => {

var result = req.body;

console.log(result);

var acjson = JSON.parse(result);

if (acjson.value == null) {

acjson.value = Math.floor(Math.random() \* 9 + 1);

console.log(acjson.value);

}

const data = await Giena.find({

value: acjson.value

}).exec();

console.log(data[0].color);

navivod = proverka("giena", acjson, data[0]);

res.redirect("/giena");

console.log("POST /giena");

})

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

console.log(req.body);

console.log(navivod);

res.json(navivod);

console.log("GET /giena");

navivod = "";

})

function proverka(animal, acjson, data) {

if (acjson.width == null & acjson.height == null) {

acjson.width = data.width;

acjson.height = data.height;

if (acjson.color == null) {

acjson.color = data.color;

}

}

if (acjson.width == null) {

acjson.width = Math.floor(Math.random() \* 3000);

}

if (acjson.height == null) {

acjson.height = Math.floor(Math.random() \* 3000);

}

if (acjson.color == null) {

var possible = "abcdefghijklmnopqrstuvwxyz0123456789";

var text = "";

for (var i = 0; i < 6; i++)

text += possible.charAt(Math.floor(Math.random() \* possible.length));

acjson.color = "#" + text;

}

if (acjson.value == data.value & acjson.width == data.width &

acjson.height == data.height & acjson.color == data.color) {

console.log("Vse Norm");

} else {

console.log("error");

return ["Error" ,{

"value": acjson.value

}, {

"width": acjson.width

}, {

"height": acjson.height

}, {

"color": acjson.color

}];

}

return [animal + "/" + acjson.value + '.jpg', {

"value": acjson.value

}, {

"width": acjson.width

}, {

"height": acjson.height

}, {

"color": acjson.color

}]

}

module.exports = router;

#### gekkon.JS

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const gekkonSchema = new Schema({

value:Number,

width:Number,

height:Number,

color:String

});

const Gekkon = mongoose.model("gekkon",gekkonSchema);

module.exports = Gekkon;

#### gazel.JS

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const gazelSchema = new Schema({

value:Number,

width:Number,

height:Number,

color:String

});

const Gazel = mongoose.model("gazel",gazelSchema);

module.exports = Gazel;

#### giena.JS

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const gienaSchema = new Schema({

value:Number,

width:Number,

height:Number,

color:String

});

const Giena = mongoose.model("giena",gienaSchema);

module.exports = Giena;

### Выводы

В ходе выполнения лабораторной работы реализованы RESTful API с использованием базы данных mongoDB.