Пенза 2020

Выполнили:

студенты группы 16ВВ1

Тимирящев А.В.

Мчедлишвили А.М

Принял:

к.т.н., доцент Дубравин А.В.

к.т.н., доцент Карамышева Н.С.

**ОТЧЕТ**

по лабораторной работе №7

по курсу «Web-программирование»

на тему «Frontend на ReactJS»

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

### Название

Frontend на ReactJS.

### Цель работы

Реализовать Frontend на ReactJS.

### Лабораторное задание

Необходимо для лабораторной работы №5 реализовать клиентское приложение с использованием технологи ReactJS.

Необходимо в интерфейсе предусмотреть элементы управления для всех параметров, которые можно передать на сервер через REST API. Нужно не забыть, что часть параметров может быть опциональной.

Так же нужно реализовать кнопку, которая должна активировать запрос к серверу и по результатам ответа изменить состояние элементов управления в соответствии с полученными значениями, а также заполнить поле с именем изображения и вместе с этим необходимо выполнить второй запрос к серверу, загрузить изображение по его имени и отобразить на странице.

|  |  |
| --- | --- |
| Вариант | Задание |
| 3 | Животные: гиена, газель, геккон. |

### Порядок выполнения работы

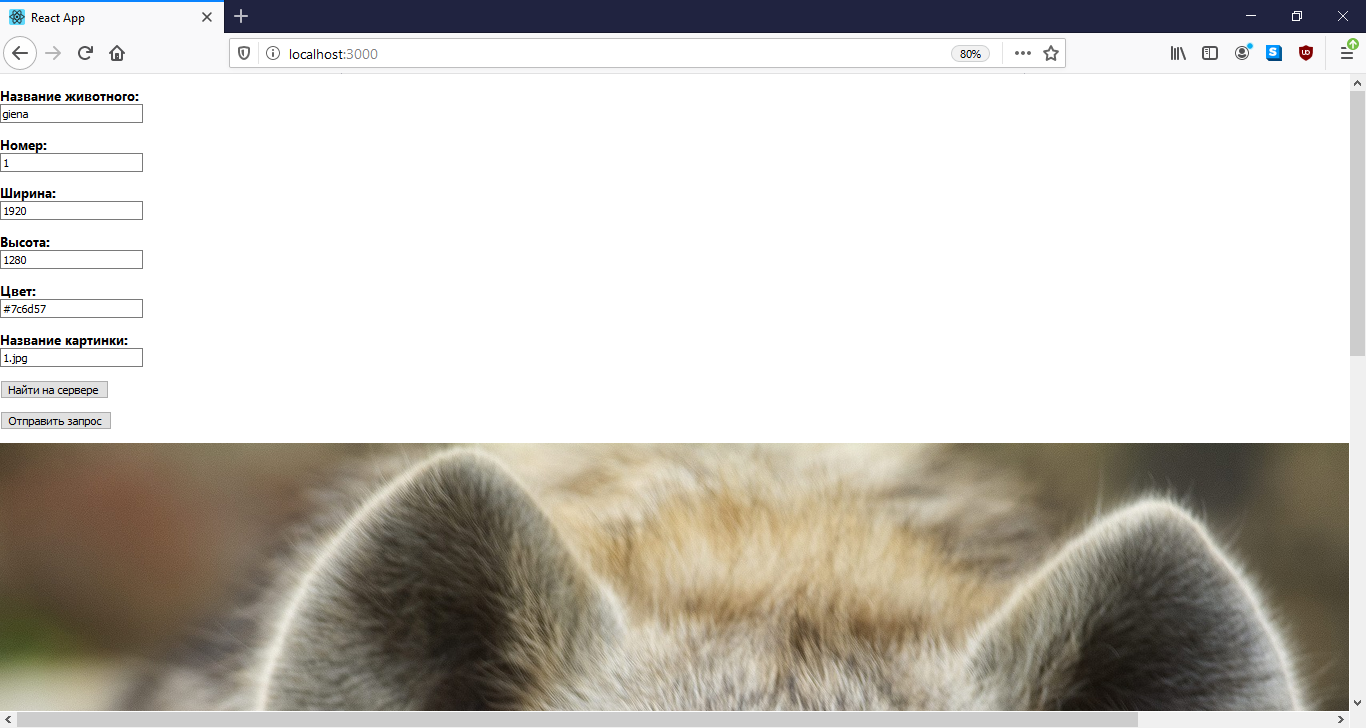


Рисунок 1 – вывод случайного изображения при вводе giena

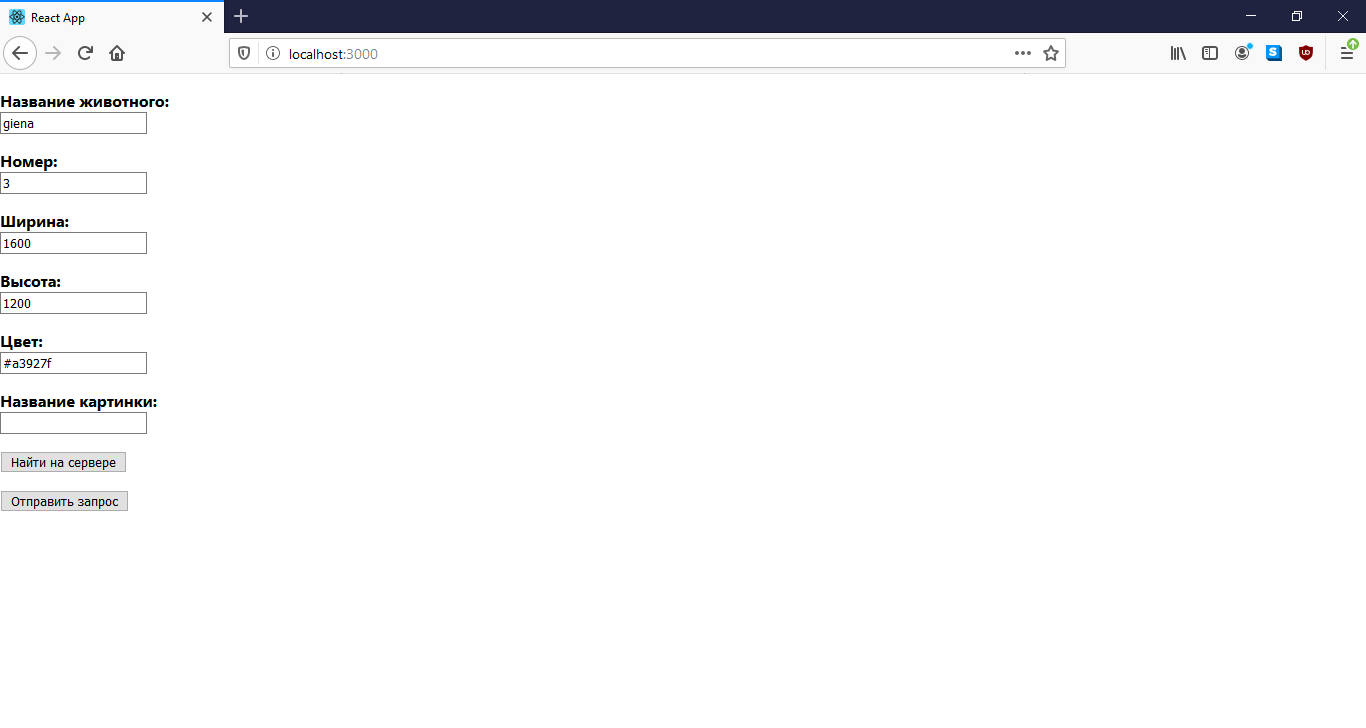


Рисунок 2 – поиск по БД данных giena с номером 3

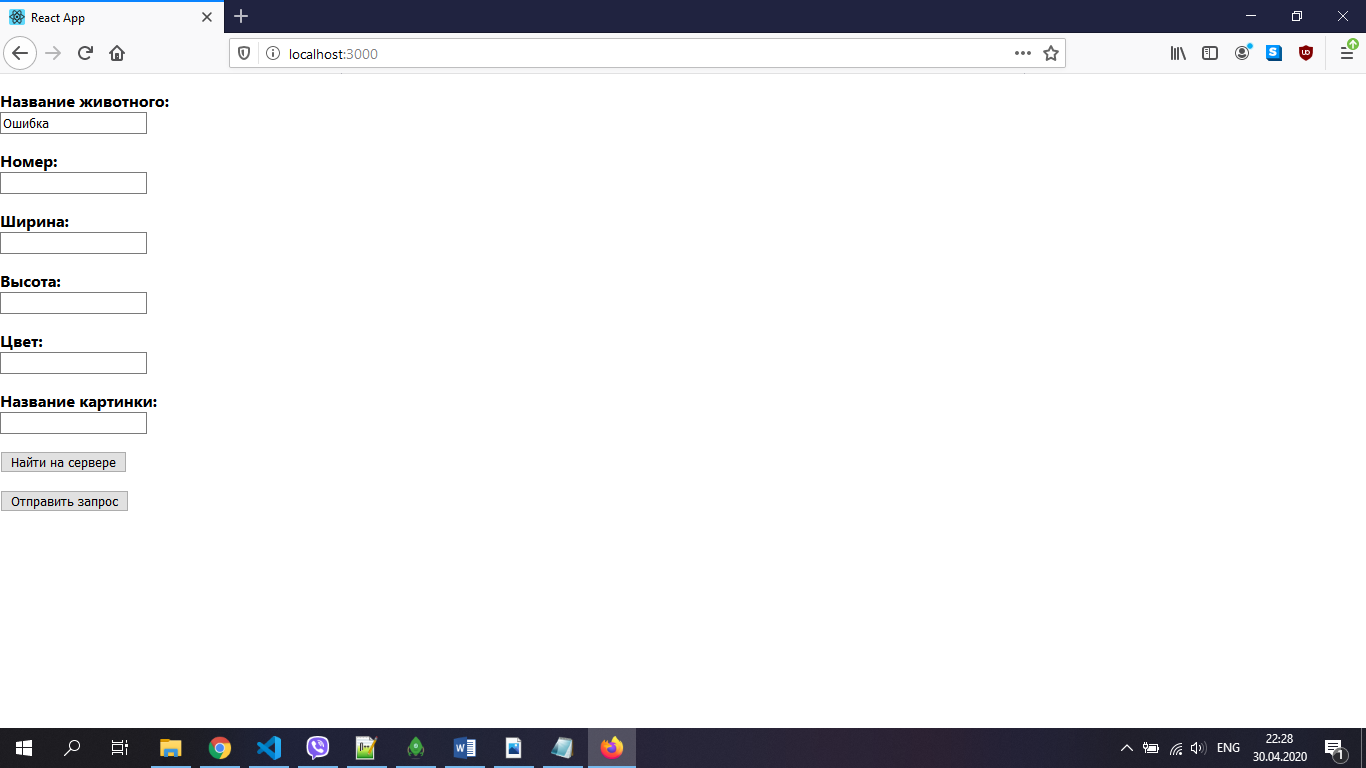


Рисунок 3 – ошибка при неправильном вводе названия

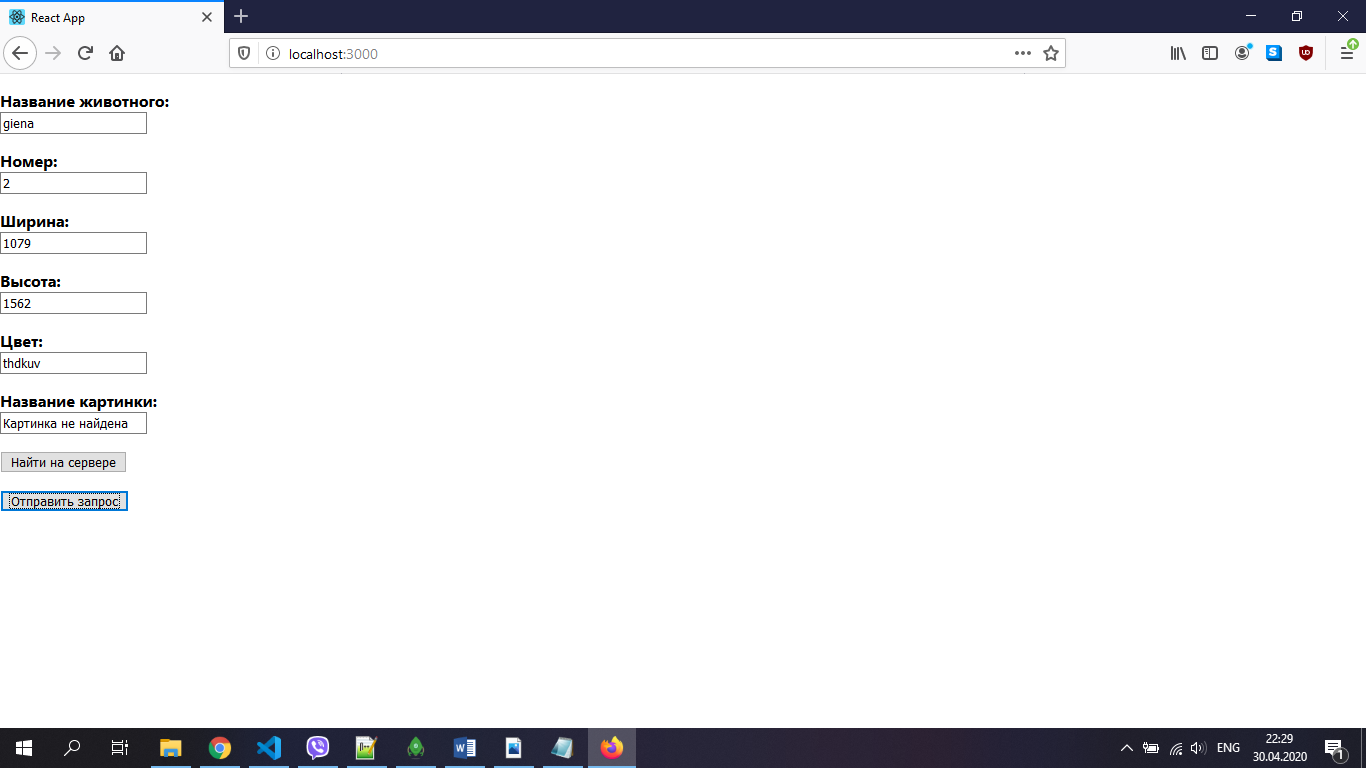


Рисунок 4 – отправка запроса без ввода данных кроме названия животного

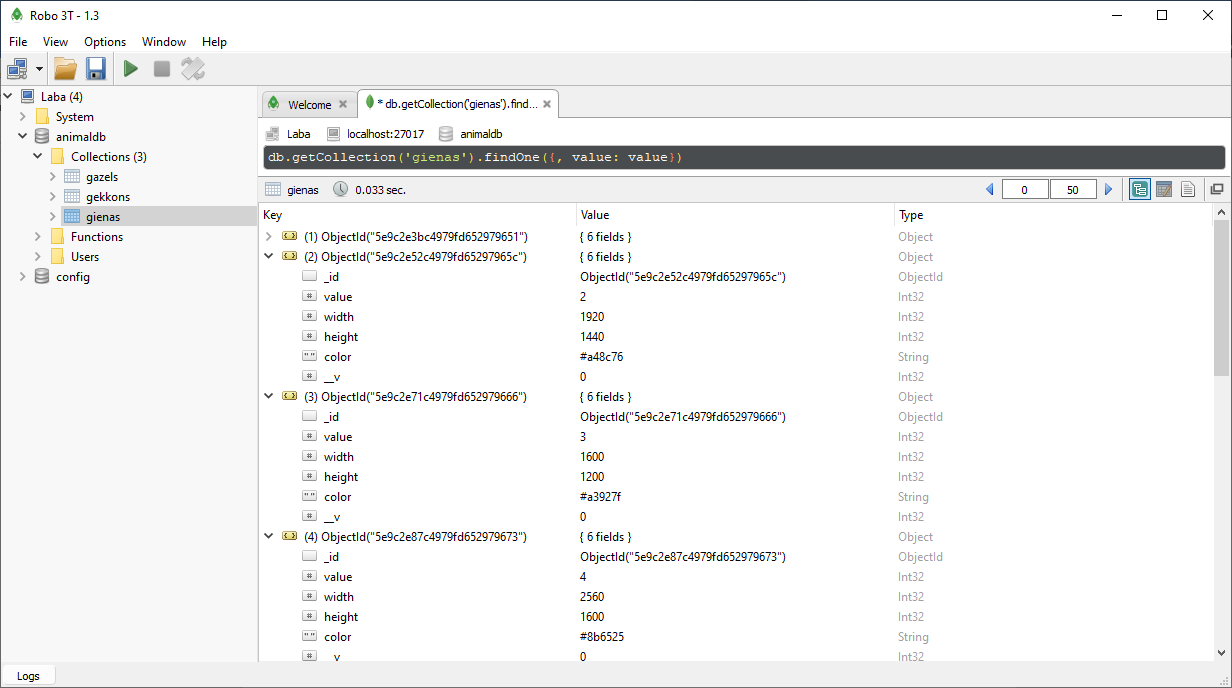


Рисунок 5 – проверка данных в БД MongoDB

### Листинг

#### App.js

import React, { useState } from 'react';

function App() {

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

const [name, setName] = useState('')

const [value, setValue] = useState('')

const [width, setWidth] = useState('')

const [height, setHeight] = useState('')

const [color, setColor] = useState('')

const [image, setImage] = useState('')

const [img, setImg] = useState('')

/// Функция получения всех параметров

function startSearch(event) {

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

if (name === masanim[i] ) {

if (name.trim()) {

var imgname = null;

var imgvalue = null;

imgname = name;

imgvalue = value;

let animanlsJSON = {

name: imgname,

value: imgvalue,

};

fetch('/findanimal', {

method: 'POST',

mode: 'no-cors',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(animanlsJSON)

})

.then(response => {

if(!response.ok) {

throw new Error(response.statusText)

}

return response;

})

.then(response => response.json())

.then(result => {

console.log(result)

if (result.error !== 0)

{

setValue(result.value)

setWidth(result.width)

setHeight(result.height)

setColor(result.color)

setImage('')

}

else

setImage('Какая то ошибка ^\_^')

});

}

return;

}

}

setName('Ошибка');

setValue('Ошибка');

event.preventDefault()

}

/// Функция получения картинки

function sendReq(event) {

event.preventDefault()

if (name.trim()) {

var imgname = null;

var imgvalue = null;

var imgwidth = null;

var imgheight = null;

var imgcolor = null;

imgname = name;

imgvalue = value;

imgwidth = width;

imgheight = height;

imgcolor = color;

let animanlsJSON = {

name: imgname,

value: imgvalue,

width: imgwidth,

height: imgheight,

color: imgcolor

};

fetch('/animal', {

method: 'POST',

mode: 'no-cors',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(animanlsJSON)

})

.then(response => {

if(!response.ok) {

throw new Error(response.statusText)

}

return response;

})

.then(response => response.json())

.then(result => {

console.log(result)

if (typeof result.error === 'undefined')

{

setImage(result.value+'.jpg')

setImg('../public/'+name+'/'+ result.value + '.jpg')

}

else{

setValue(result.value)

setWidth(result.width)

setHeight(result.height)

setColor(result.color)

setImage('Картинка не найдена')

}});

}

}

return (

<div className="wrapper">

<label>

<b>Название животного:</b> <br />

<input

value={name}

onChange={event => setName(event.target.value)}

/><p/>

<b>Номер:</b> <br />

<input

value={value}

onChange={event => setValue(event.target.value)}

/><p/>

<b>Ширина:</b><br />

<input

value={width}

onChange={event => setWidth(event.target.value)}

></input><p/>

<b>Высота:</b><br />

<input

value={height}

onChange={event => setHeight(event.target.value)}

></input><p/>

<b>Цвет:</b><br />

<input

value={color}

onChange={event => setColor(event.target.value)}

></input><p/>

<b>Название картинки:</b><br />

<input

value={image}

onChange={event => setImage(event.target.value)}

></input><p/>

<input

type="submit"

value="Найти на сервере"

onClick={startSearch}

/><p/>

<input

type="submit"

value="Отправить запрос"

onClick={sendReq}

/><p/>

<img

src={img}

onChange={event => setImg(event.target.value)}

/>

</label>

</div>

);

}

export default App;

#### 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");

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

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

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

const router = express.Router();

let jsonbody = null;

let imgerror = null;

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

console.log(jsonbody);

res.send(jsonbody);

});

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

var result = JSON.parse(req.body);

var name = result.name;

if (result.value == '')

{

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

}

else{

var value = result.value;

}

var modelan = null;

if (name == "gekko"){

var modelan = Gekkon;

}

else if (name == "gazel"){

var modelan = Gazel;

}

else if (name == "giena"){

var modelan = Giena;

}

modelan.findOne({value: value})

.then(animal => {

{

res.send({name, value: animal.value, width: animal.width, height: animal.height, color: animal.color});

}

})

});

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

var result = JSON.parse(req.body);

var name = result.name;

var value = result.value;

var height = result.height;

var width = result.width;

var color = result.color;

console.log(height);

console.log(width);

console.log(color);

var modelan = null;

if (name == "gekko"){

var modelan = Gekkon;

}

else if (name == "gazel"){

var modelan = Gazel;

}

else if (name == "giena"){

var modelan = Giena;

}

if (width == "")

{

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

}

if (height == "")

{

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

}

if (color == "")

{

var coloring = "abcdefghijklmnopqrstuvwxyz0123456789";

var color = "";

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

color += coloring.charAt(Math.floor(Math.random() \* coloring.length));

}

console.log(height);

console.log(width);

console.log(color);

modelan.findOne({value: value})

.then(animal => {

modelan.findOne({width: width, height: height})

.then(animal => {

if (animal != null)

{

modelan.findOne({color: color})

.then(animal => {

if (animal != null)

{

jsonbody = animal;

res.redirect("./animals");

}

else

{

imgerror = {error: "Color error", name: name, value: value, width: width, height: height, color: color};

res.send(imgerror);

}

});

}

else

{

imgerror = {error: "Size error", name: name, value: value, width: width, height: height,color: color};

res.send(imgerror);

}

});

}

);

});

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 был реализован графический интерфейс приложения «Frontend» с помощью ReactJS.