**ОТЧЕТ**

по лабораторной работе №5

по курсу «Web-программирование»

на тему «Реализация ответа обработчиков и

POST-запросы»

Выполнили:

Студенты группы 16ВВ1

Соловьев А.А.

Чиркин К.Д.

Принял:

к.т.н., доцент Дубравин А.В.

Пенза 2020

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

### Лабораторное задание.

1. Выполнить редактирование кода http-сервера в соответствии с заданием:
2. На стороне клиента реализовать проверку: если в текстовом поле указано одно из трех животных в следующем формате: «antilopa» или «antilopa 7» (число от 0 до 9), выполнить формирование и отправку на сервер POST-запроса, в противном случае вывести сообщение о некорректно указанных данных;
3. На стороне сервера реализовать три обработчика, соответствующих каждому из указанных в варианте животных;
4. В обработчике, в случае, если в URL запросе отсутствует параметр с указанием номера запрашиваемого изображения животного, выполнить генерацию случайного числа в диапазоне от 0 до 9 и выполнить отправку клиенту соответствующего изображения. Если в URL запросе присутствует параметр с указанием номера запрашиваемого изображения животного – выполнить отправку клиенту соответствующего изображения;

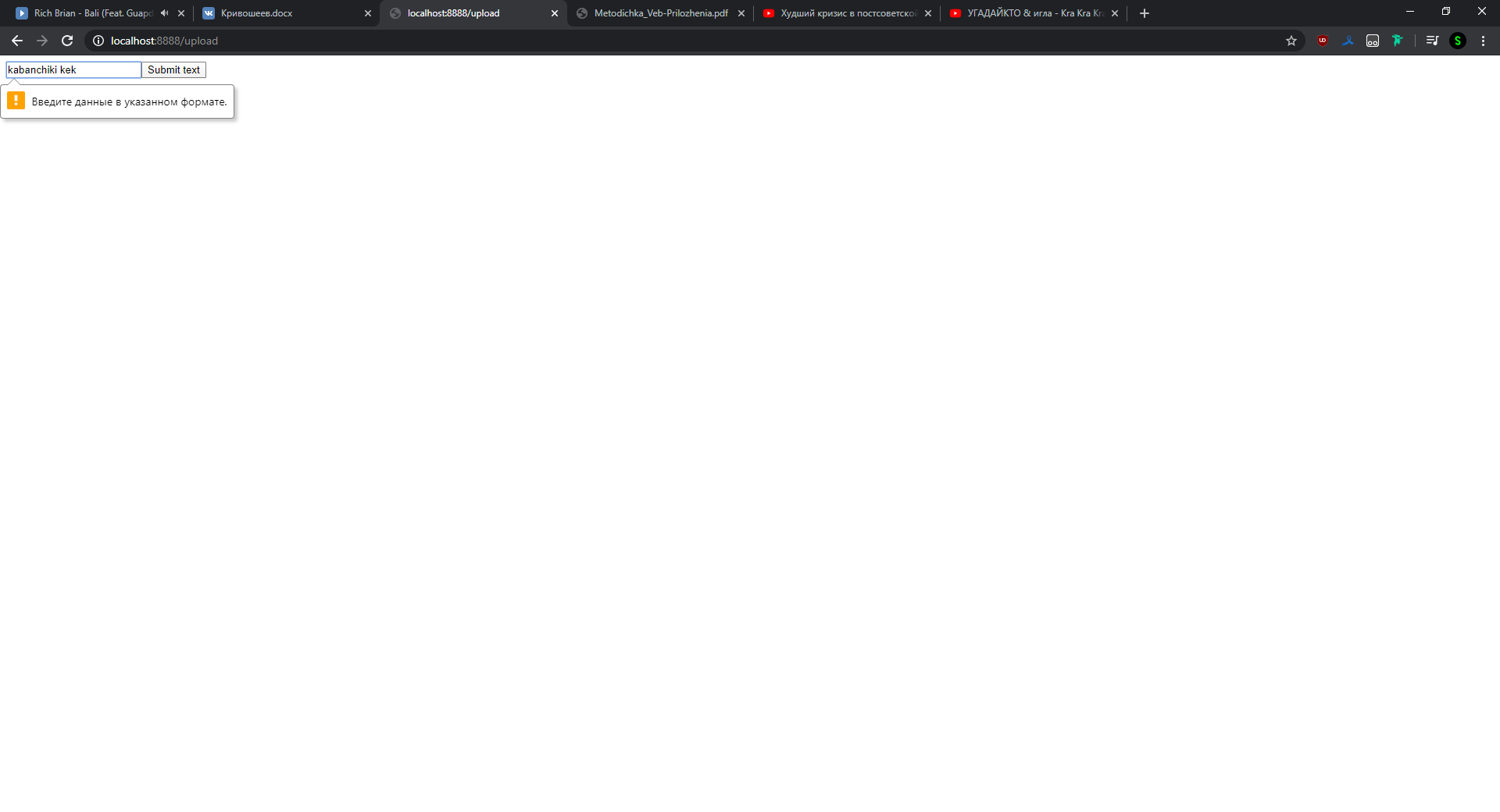
### Ход работы.

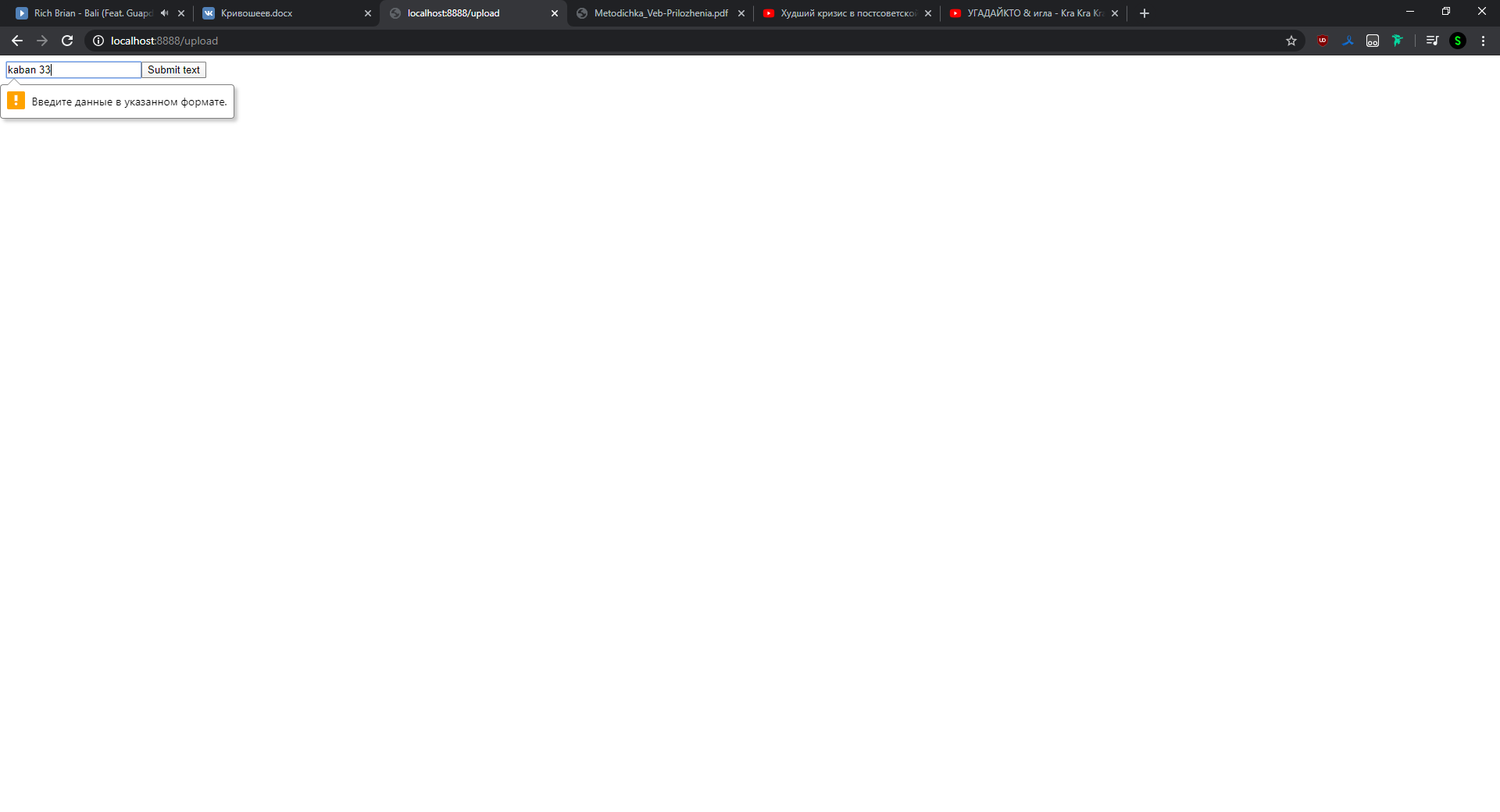
#### Реализация проверки на стороне клиента.

Выполняем проверку введенного пользователем текста в форму путем регулярных выражений:

<input pattern="[a-zA-Z]{1,8}\\s?\\d?" name="text">

Получаем следующий результат:





**Рисунок 1. Результат проверки значения.**

#### Реализация обработки запроса на сервере.

В requestHandlers.js добавляем еще три обработчика, соответствующих каждому из введенных животных. Реализуем в методе upload() проверку запроса и вызов необходимого обработчика. В каждом методе проверяем наличие индекса требуемого изображения, и в случае его отсутствия генерируем случайное число в интервале от 0 до 9 включительно. Передаем клиенту необходимое изображение животного.

Полученные методы в requestHandlers.js:

function upload(response, postData) {

    console.log("Request handler 'upload' was called.");

    var req = querystring.parse(postData).text.trim().split(' ');

    switch(req[0]) {

        case 'kaban':

            kaban(response, req)

            break;

        case 'kakadu':

            kakadu(response, req);

            break;

        case 'kalmar':

            kalmar(response, req);

            break;

        default:

            start(response, postData)

    }

}

function kaban(response, Data) {

    if (typeof Data[1] === 'undefined') {

        Data[1] = getRandomInt(0,9);

    }

    var i = Data[1];

    var image = fs.readFileSync('images/kaban/kaban\_'+i+'.jpg');

    response.writeHead(200, { "Content-Type": "image/jpg" });

    response.write(image);

    response.end();

}

function kakadu(response, Data) {

    if (typeof Data[1] === 'undefined') {

        Data[1] = getRandomInt(0,9);

    }

    var i = Data[1];

    var image = fs.readFileSync('images/kakadu/kakadu\_'+i+'.jpg');

    response.writeHead(200, { "Content-Type": "image/jpg" });

    response.write(image);

    response.end();

}

function kalmar(response, Data) {

    if (typeof Data[1] === 'undefined') {

        Data[1] = getRandomInt(0,9);

    }

    var i = Data[1];

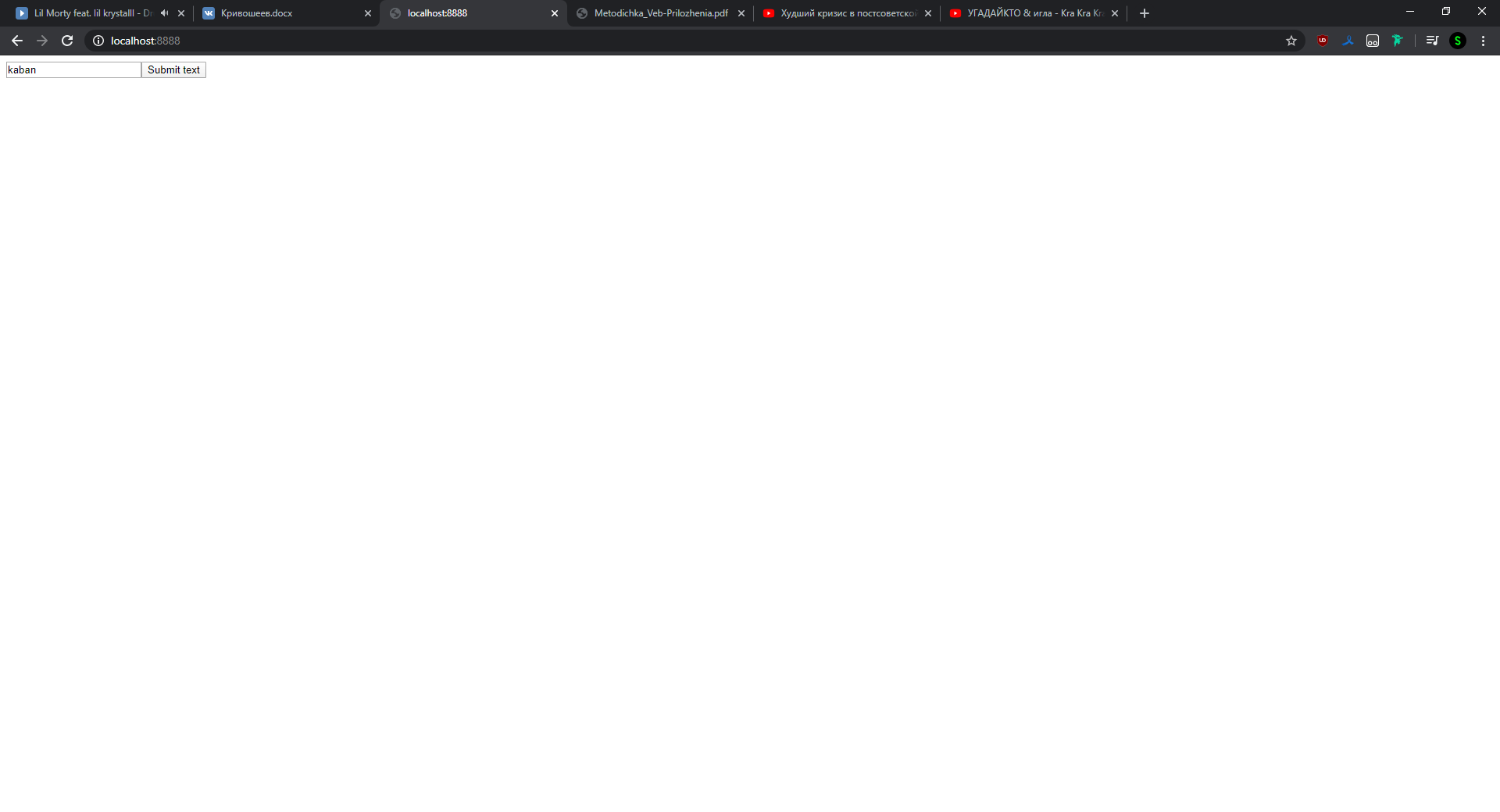
    var image = fs.readFileSync('images/kalmar/kalmar\_'+i+'.jpg');

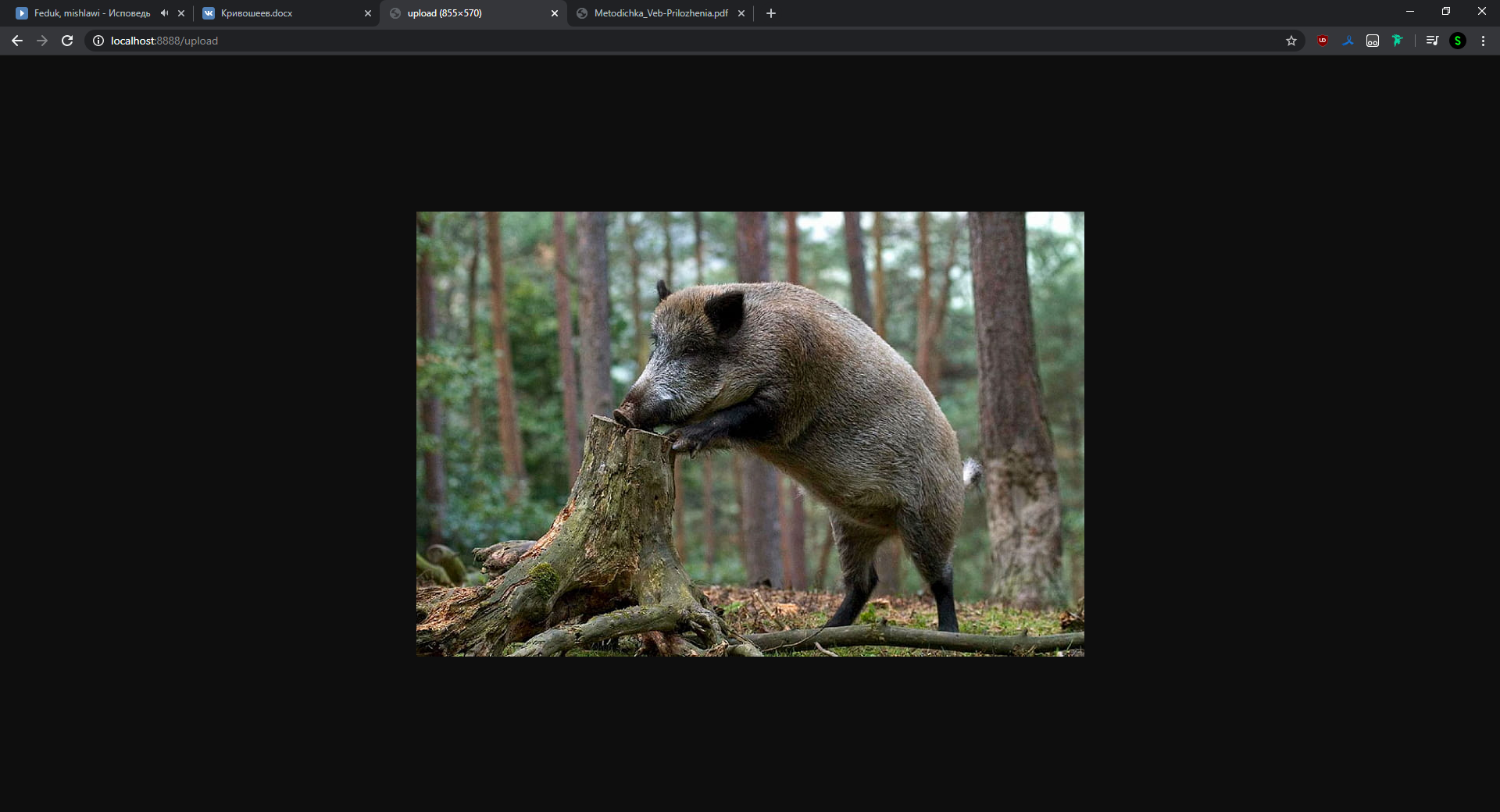
    response.writeHead(200, { "Content-Type": "image/jpg" });

    response.write(image);

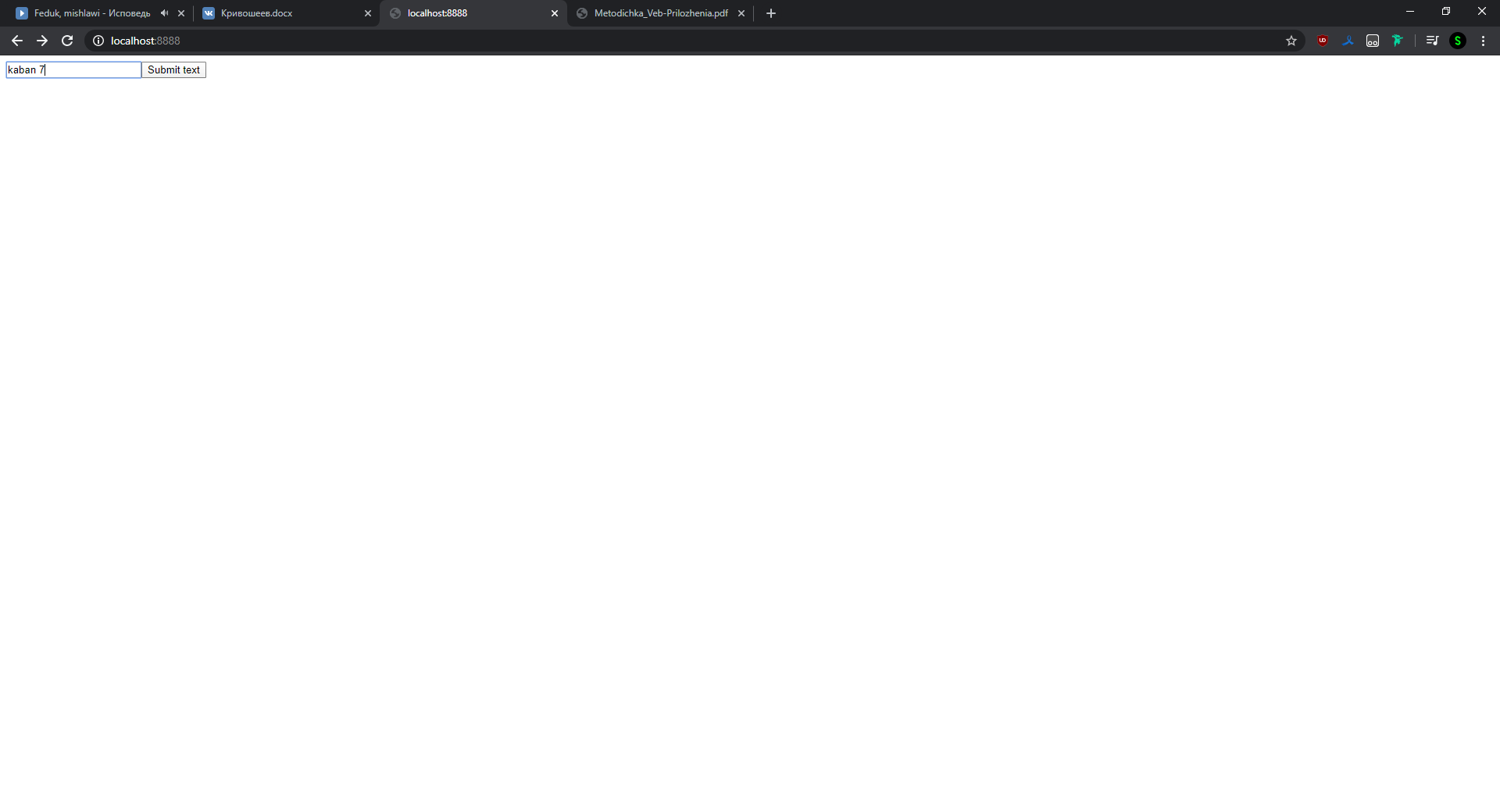
    response.end();

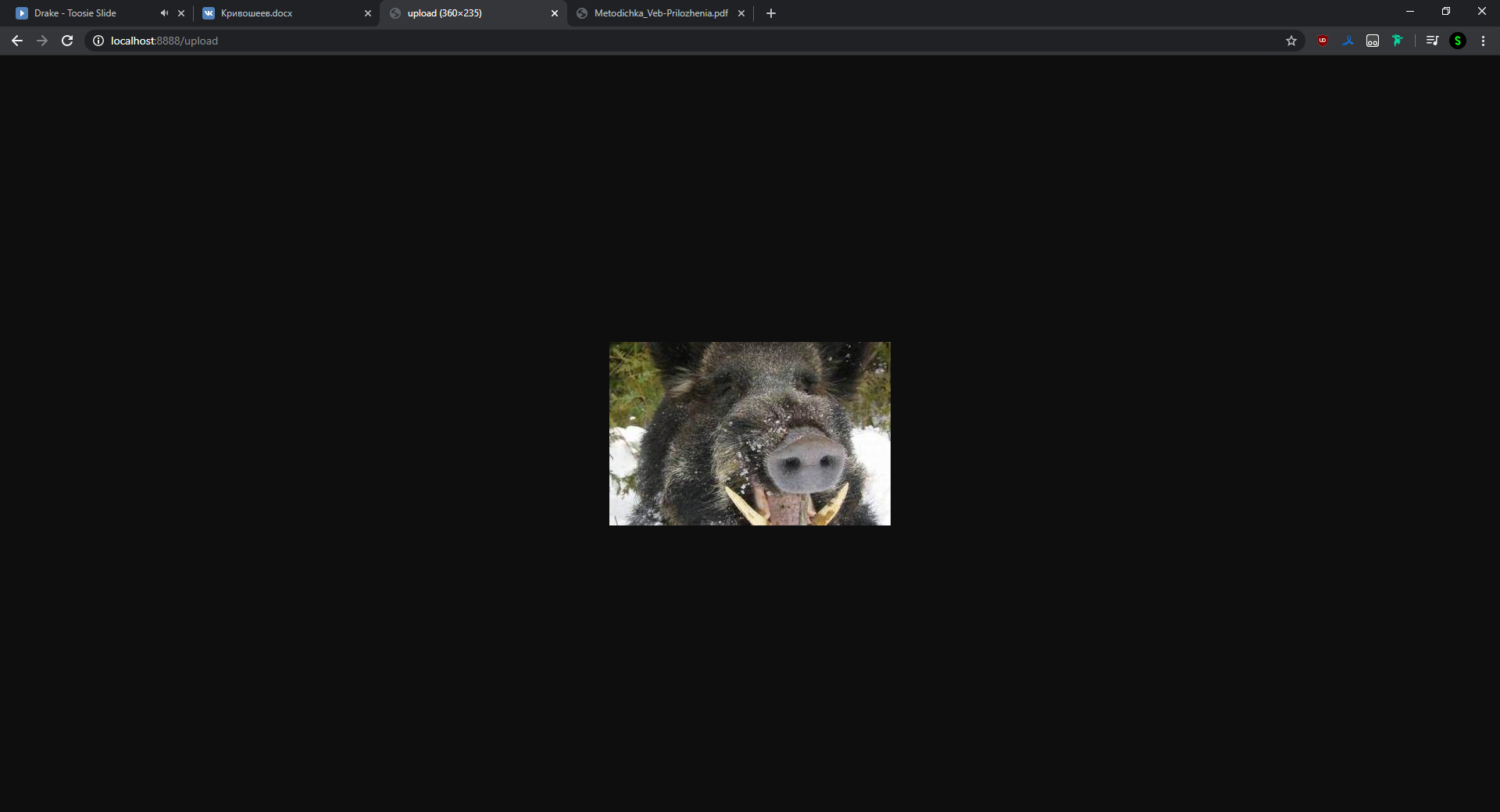
}





**Рисунок 2. Соответствие ответа запросу, 1 вариант.**





**Рисунок 2. Соответствие ответа запросу, 2 вариант.**

### Выводы

В ходе данной лабораторной работы получили практические навыки обработки post-запросов со стороны клиента. Для упрощения обработчики для каждого из животных можно объединить в один.