**ОТЧЕТ**

по лабораторной работе №4

по курсу «Web-программирование»

на тему «Маршрутизация запросов»

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. Реализовать роутер.
2. Реализовать маршрутизацию внутри http-сервера.
3. Выполнить контрольное задание:

Выполнить редактирование кода, реализующего сервер, следующим образом: добавить обработку для трех различных URL запросов к серверу. В обработчиках запросов выводить в консоль сообщение о том, какой обработчик был вызван.

### Ход работы.

#### Реализация роутера.

В зависимости от URL, запрошенного браузером у http-сервера, сервер должен реагировать по-разному. Задание соответствия между разными HTTP-запросами и разными частями кода называется «маршрутизация» («routing», роутинг). Нужно иметь возможность передавать запрошенный URL и возможные добавочные GET- и POST-параметры разрабатываемому роутеру и, с учётом этого, роутер должен определять, какой код выполнять.

Создаем router.js следующего содержания:

function route(pathname) {

console.log("About to route a request for " + pathname);

}

exports.route = route;

Редактируем index.js и server.js для вызова функции route. Если сейчас выполнить запуск приложения и запросить какой-либо URL, то мы увидим в консоли, что HTTP-сервер использует разработанный роутер и передает ему запрошенный pathname.

#### Реализация обработки входящих http-запросов.

«Роутинг» подразумевает, что сервер должен обрабатывать запросы на разные URL по-разному. Это подразумевает реализацию «бизнес-логики» для запросов к /start в одной функции, а для запросов к /upload в другой. Роутер – не место, чтобы делать что-то с запросами, потому что будет затруднено масштабирование, а разрабатываемое приложение будет становиться сложнее. Выносим функции обработки запросов в отдельный requestHandlers.js:

function start() {

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

}

function upload() {

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

}

exports.start = start;

exports.upload = upload;

Передаем список из requestHandlers как объект, для этого редактируем index.js:

var server = require("./server");

var router = require("./router");

var requestHandlers = require("./requestHandlers");

var handle = {}

handle["/"] = requestHandlers.start;

handle["/start"] = requestHandlers.start;

handle["/upload"] = requestHandlers.upload;

server.start(router.route, handle);

Добавляем параметр handle в функцию start() в server.js и передаем объект handle в callback-функцию route() в качестве первого параметра.

Редактируем метод route в router.js:

function route(handle, pathname) {

console.log("About to route a request for " + pathname);

if (typeof handle[pathname] === 'function') {

handle[pathname]();

} else {

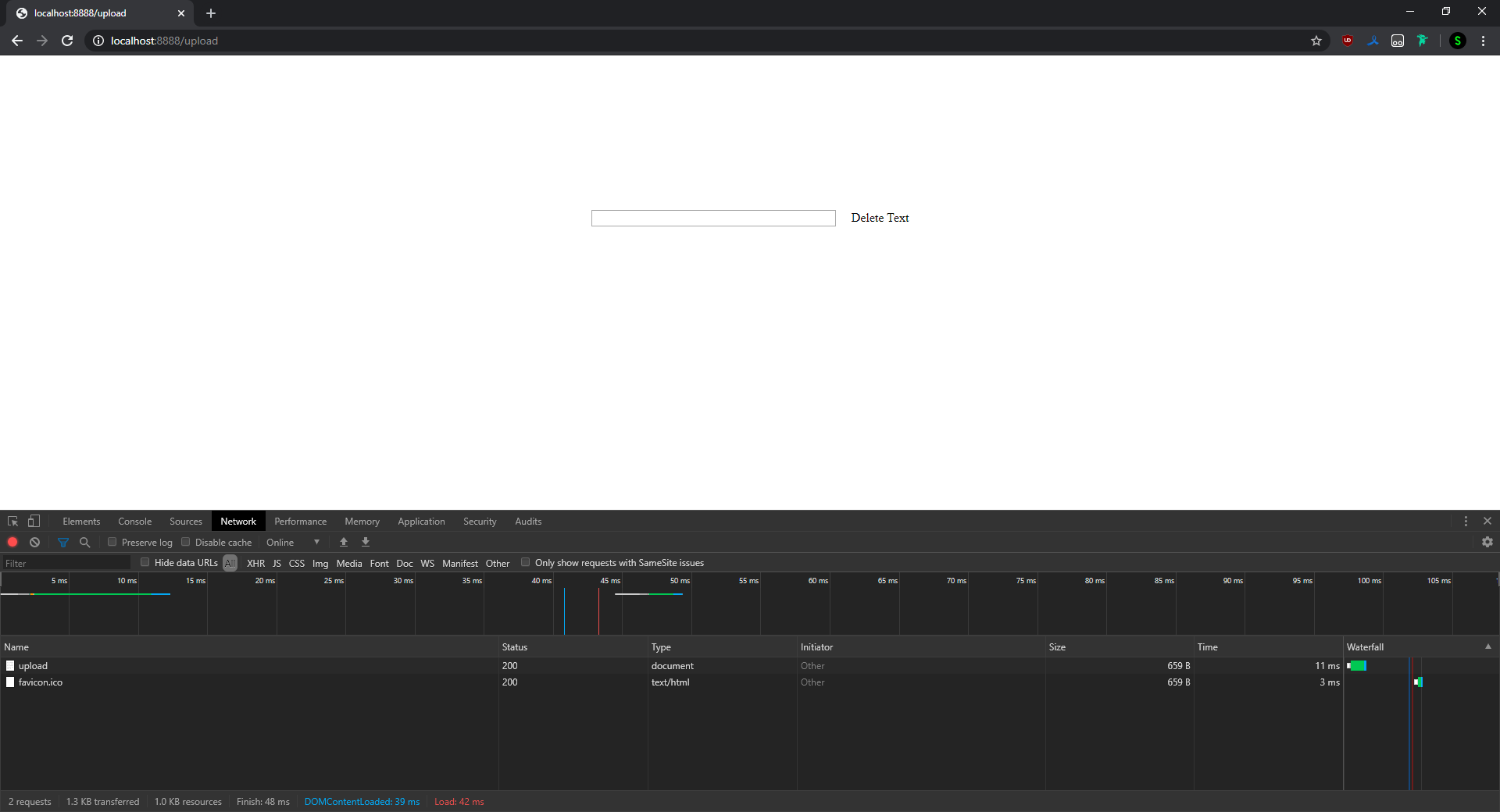
console.log("No request handler found for " + pathname);

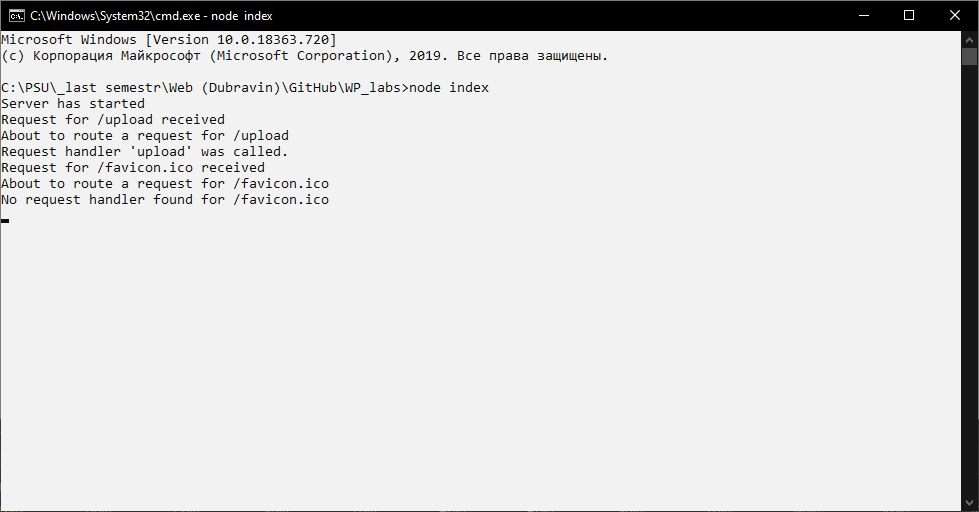
}

}

exports.route = route;

Проверяем работоспособность:





**Рисунок 1. Результат работы.**

#### Выполняем контрольное задание.

Редактируем содержимое requestHandlers.js:

function start() {

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

}

function upload() {

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

}

function first() {

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

}

function second() {

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

}

function third() {

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

}

exports.start = start;

exports.upload = upload;

exports.first = first;

exports.second = second;

exports.third = third;

Редактируем index.js:

var server = require("./server");

var router = require("./router");

var requestHandlers = require("./requestHandlers");

var handle = {}

handle["/"] = requestHandlers.start;

handle["/start"] = requestHandlers.start;

handle["/upload"] = requestHandlers.upload;

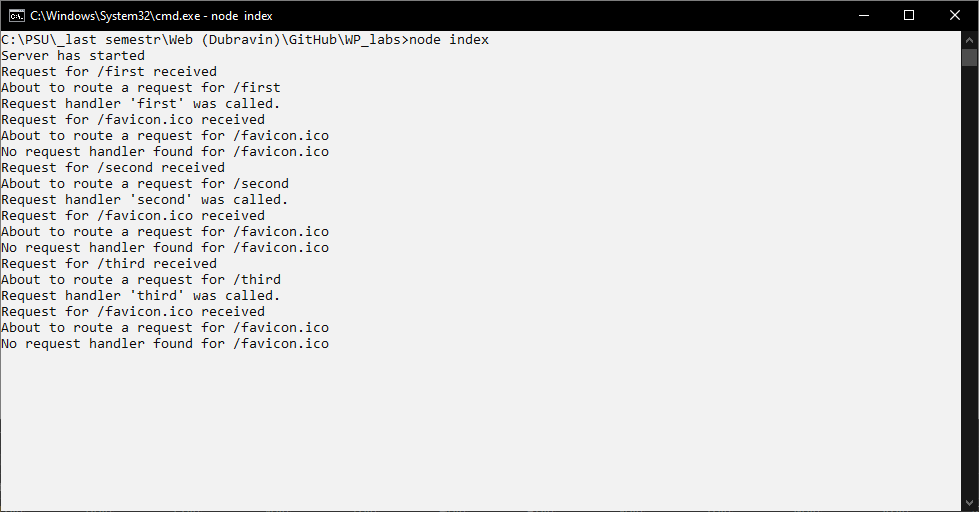
handle["/first"] = requestHandlers.first;

handle["/second"] = requestHandlers.second;

handle["/third"] = requestHandlers.third;

server.start(router.route, handle);

Смотрим результат:



**Рисунок 2. Результат выполнения контрольного задания.**

### Выводы

В ходе данной лабораторной работы получили практические навыки маршрутизации внутри http-сервера.