|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ДИСЦИПЛИНА «Архитектура ЭВМ»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № 4**

|  |  |
| --- | --- |
| **Студент** \_\_Аникин И. А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Группа** \_\_ИУ7-51Б\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Преподаватель** \_\_Попов А. Ю.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |

Москва.

2020 г.

Оглавление

[Задание 7.1 3](#_Toc53870235)

[Задание 7.2 8](#_Toc53870236)

Цель работы: научиться организовывать общение между серверами и передачу данных через аргументы командной строки.

# Задание 7.1

Создать сервер **А**. На стороне сервера хранится файл с содержимым в формате **JSON**. При получении запроса на **/insert/record** идёт добавление записи в файл. При получении запроса на **/select/record** идёт получение записи из файла. Каждая запись хранит информацию о машине (*название* и *стоимость*).

Создать сервер **Б**. На стороне сервера хранится файл с содержимым в формате **JSON**. Каждая запись в файле хранит информацию о складе и массиве машин, находящихся на данном складе. То есть каждая запись хранит в себе название склада (*строку*) и массив названий машин (*массив строк*). При получении запроса на **/insert/record** идёт добавление записи в файл. При получении запроса на **/select/record** идёт получение записи из файла.

Создать сервер **C**. Сервер выдаёт пользователю страницы с формами для ввода информации. При этом сервер взаимодействует с серверами **А** и **Б**. Реализовать для пользователя функции:

* создание нового типа машины
* получение информации о стоимости машины по её типу
* создание нового склада с находящимися в нём машинами
* получение информации о машинах на складе по названию склада

Реализовать удобный для пользователя интерфейс взаимодействия с системой (использовать поля ввода и кнопки).

Листинг программы

Сервер А

"use strict";

const express = require("express");

const fs = require("fs");

const app = express();

const port = 5016;

app.listen(port);

console.log("My server on port " + port);

let arr = JSON.parse(fs.readFileSync("a.txt"));

app.get("/select/record", function(request, response) {

const name = request.query.name;

if (!name)

return response.end('No name in query');

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

if (name == arr[i]["name"])

return response.end(JSON.stringify({name: arr[i]["name"], cost: arr[i]["cost"]}));

return response.end("No car with such name!");

});

app.use(function(req, res, next) {

res.header("Cache-Control", "no-cache, no-store, must-revalidate");

res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

res.header("Access-Control-Allow-Origin", "\*");

next();

});

function loadBody(request, callback) {

let body = [];

request.on('data', (chunk) => {

body.push(chunk);

}).on('end', () => {

body = Buffer.concat(body).toString();

callback(body);

});

}

app.post("/insert/record", function(request, response) {

loadBody(request, function(body) {

const name = request.query.name;

const cost = request.query.cost;

if (!name || !cost)

return response.end("Invalid query params");

arr.push({name: name, cost: cost});

response.end("Car added successfully!");

});

});

Сервер Б

"use strict";

const express = require("express");

const fs = require("fs");

const app = express();

const port = 5017;

app.listen(port);

console.log("My server on port " + port);

let arr = JSON.parse(fs.readFileSync("b.txt"));

app.get("/select/record", function(request, response) {

const name = request.query.name;

if (!name)

return response.end('No name in query');

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

if (name == arr[i]["name"])

return response.end(JSON.stringify({name: arr[i]["name"], cars: arr[i]["cars"]}));

return response.end("No warehouse with such name!");

});

app.use(function(req, res, next) {

res.header("Cache-Control", "no-cache, no-store, must-revalidate");

res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

res.header("Access-Control-Allow-Origin", "\*");

next();

});

function loadBody(request, callback) {

let body = [];

request.on('data', (chunk) => {

body.push(chunk);

}).on('end', () => {

body = Buffer.concat(body).toString();

callback(body);

});

}

app.post("/insert/record", function(request, response) {

loadBody(request, function(body) {

const name = request.query.name;

console.log(request.query);

const cars = JSON.parse(request.query.cars);

if (!name || !cars)

return response.end("Invalid query params");

arr.push({name: name, cars: cars});

response.end("Warehouse added successfully!");

});

});

Сервер С

"use strict";

const express = require("express");

const fs = require("fs");

const request = require("request");

const app = express();

const port = 5015;

app.listen(port);

console.log("My server on port " + port);

app.get("/a", function(request, response) {

return response.end(fs.readFileSync("./a.html"));

});

app.get("/b", function(request, response) {

return response.end(fs.readFileSync("./b.html"));

});

app.get("/c", function(request, response) {

return response.end(fs.readFileSync("./c.html"));

});

app.get("/d", function(request, response) {

return response.end(fs.readFileSync("./d.html"));

});

function sendPost(url, body, callback) {

// задаём заголовки

const headers = {};

headers["Cache-Control"] = "no-cache, no-store, must-revalidate";

headers["Connection"] = "close";

// отправляем запрос

request.post({

url: url,

body: body,

headers: headers,

}, function (error, response, body) {

if(error) {

callback(null);

} else {

callback(body);

}

});

}

function sendGet(url, body, callback) {

// задаём заголовки

const headers = {};

headers["Cache-Control"] = "no-cache, no-store, must-revalidate";

headers["Connection"] = "close";

// отправляем запрос

request.get({

url: url,

body: body,

headers: headers,

}, function (error, response, body) {

if(error) {

callback(null);

} else {

callback(body);

}

});

}

app.get("/select\_car", function(request, response) {

const name = request.query.name;

sendGet(`http://localhost:5016/select/record?name=${name}`, "", function(ans) {

return response.end(ans);

});

});

app.get("/insert\_car", function(request, response) {

const name = request.query.name;

const cost = request.query.cost;

sendPost(`http://localhost:5016/insert/record?name=${name}&cost=${cost}`, "", function(ans) {

return response.end(ans);

});

});

app.get("/select\_warehouse", function(request, response) {

const name = request.query.name;

sendGet(`http://localhost:5017/select/record?name=${name}`, "", function(ans) {

return response.end(ans);

});

});

app.get("/insert\_warehouse", function(request, response) {

const name = request.query.name;

const cars = request.query.cars;

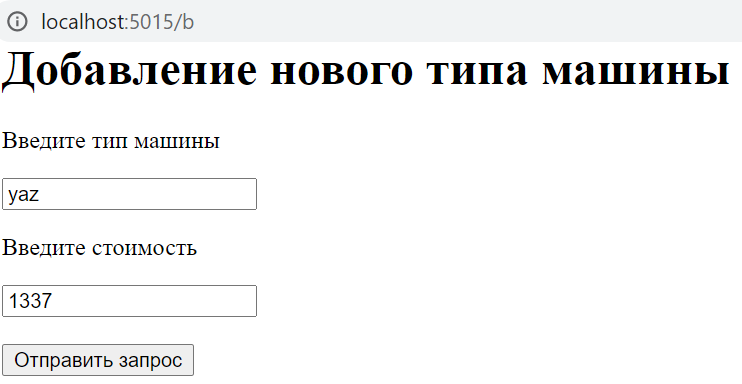
sendPost(`http://localhost:5017/insert/record?name=${name}&cars=${cars}`, "", function(ans) {

return response.end(ans);

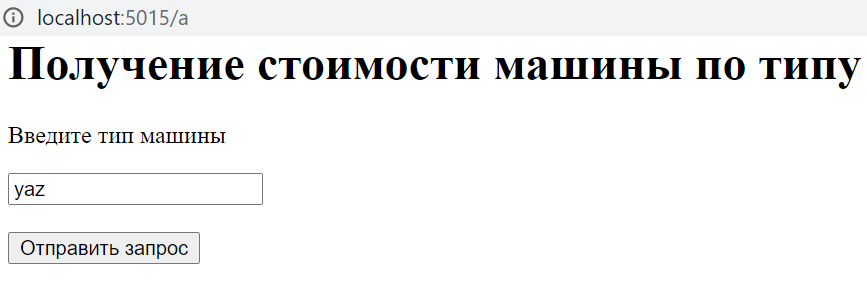
});

});

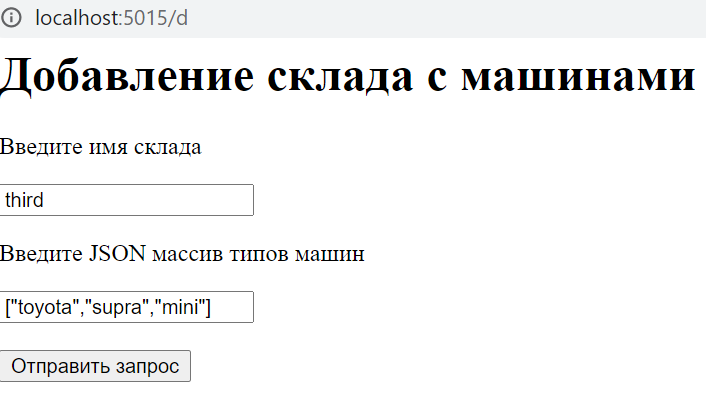
Примеры работы



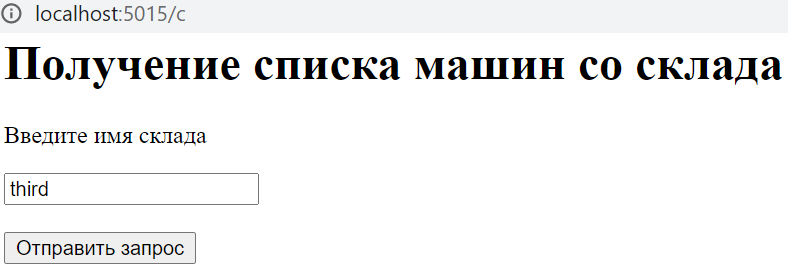














# Задание 7.2

Написать скрипт, который принимает на вход число и считает его факториал. Скрипт должен получать параметр через **process.argv**.

Написать скрипт, который принимает на вход массив чисел и выводит на экран факториал каждого числа из массива. Скрипт принимает параметры через **process.argv**.

При решении задачи вызывать скрипт вычисления факториала через **execSync**.

Листинг программы

Factorial.js

"use strict"

let n = process.argv[2];

let res = 1;

for (let i = 1; i <= n; i++)

res \*= i;

console.log(res);

task7\_2.js

"use strict"

const execSync = require('child\_process').execSync;

function useCmd(s) {

const options = {encoding: 'utf8'};

const cmd = s.toString();

const answer = execSync(cmd, options);

return answer.toString();

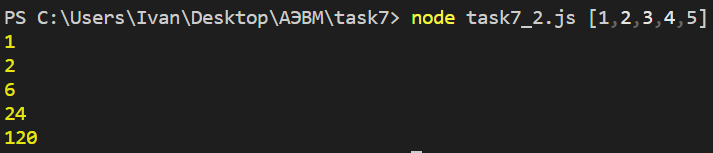
}

let arr = JSON.parse(process.argv[2]);

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

console.log(parseInt(useCmd(`node factorial.js ${arr[i]}`)));

Примеры работы



Вывод: в ходе данной лабораторной работы я научился организовывать общение между серверами и передавать параметры через аргументы командной строки.