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 **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

О Т Ч Е Т

по лабораторной работе № 4

Дисциплина: Архитектура ЭВМ

Студент ИУ7-52Б _____ В.А. Иванов
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ А.Ю. Попов
(Подпись, дата) (И.О. Фамилия)

Москва, 2020

Цель работы

Целью лабораторной работы освоение методов работы с взаимодействием серверов, скриптами и дочерними процессами.

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 **С**. 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).

Программная реализация

servA.js (сервер A)

```
"use strict";

const express = require("express");
const fs = require("fs");

const app = express();
const port = 5003;
app.listen(port);
console.log("Car server on port " + port);

const file_name = "A.txt";

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 get_file_arr(f_name) {
    let file_arr;
    if (fs.existsSync(f_name)) {
        const file_str = fs.readFileSync(f_name, "utf-8");
        if (file_str == "")
            file_arr = [];
        else
            file_arr = JSON.parse(file_str);
    } else {
        file_arr = [];
    }
    return file_arr;
}

function is_valid(car) {
    if (!car.name)  return false;
    if (!car.cost)  return false;
    return true;
}

function is_car_in(car_arr, car) {
    for (let i=0; i<car_arr.length; i++)
        if (car_arr[i].name == car.name)
            return true;
    return false;
}

function find_car(car_arr, name) {
```

```
    for (let i=0; i<car_arr.length; i++)
        if (car_arr[i].name == name)
            return car_arr[i];
    return null;
}


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) {
        let msg = "Car added";
        const obj = JSON.parse(body);

        let file_arr = get_file_arr(file_name);
        if (!is_car_in(file_arr, obj)) {
            file_arr.push(obj);
            fs.writeFileSync(file_name, JSON.stringify(file_arr));
        } else {
            msg = "Car allready in file";
        }

        response.end(msg);
    });
});

app.post("/select/record", function(request, response) {
    loadBody(request, function(body) {
        const car_name = JSON.parse(body);

        const file_arr = get_file_arr(file_name);
        let car = find_car(file_arr, car_name);
        if (car === null)
            response.end();
        else
            response.end(JSON.stringify(car));
    });
});

app.post("/select/all", function(request, response) {
    loadBody(request, function(body) {
        const file_arr = get_file_arr(file_name);
```

```
        response.end(JSON.stringify(file_arr));
    });
});
```

servB.js (сервер B)

```
"use strict";

const express = require("express");
const fs = require("fs");

const app = express();
const port = 5002;
app.listen(port);
console.log("Storage server on port " + port);

const file_name = "B.txt";

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 get_file_arr(f_name) {
    let file_arr;
    if (fs.existsSync(f_name)) {
        const file_str = fs.readFileSync(f_name, "utf-8");
        if (file_str == "")
            file_arr = [];
        else
            file_arr = JSON.parse(file_str);
    } else {
        file_arr = [];
    }
    return file_arr;
}

function is_valid(storage) {
    if (!storage.name)  return false;
    if (!storage.car_arr)  return false;
    return true;
}

function is_storage_in(storage_arr, storage) {
    for (let i=0; i<storage_arr.length; i++)
        if (storage_arr[i].name == storage.name)
            return true;
```

```
    return false;
}

function find_storage(storage_arr, name) {
    for (let i=0; i<storage_arr.length; i++)
        if (storage_arr[i].name == name)
            return storage_arr[i];
    return null;
}


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) {
        let msg = "Storage added";
        const obj = JSON.parse(body);

        let file_arr = get_file_arr(file_name);
        if (!is_storage_in(file_arr, obj)) {
            file_arr.push(obj);
            fs.writeFileSync(file_name, JSON.stringify(file_arr));
        } else {
            msg = "Storage allready in file";
        }

        response.end(msg);
    });
});


app.post("/select/record", function(request, response) {
    loadBody(request, function(body) {
        const storage_name = JSON.parse(body);

        const file_arr = get_file_arr(file_name);
        let storage = find_storage(file_arr, storage_name);
        if (storage === null)
            response.end();
        else
            response.end(JSON.stringify(storage));
    });
});
```

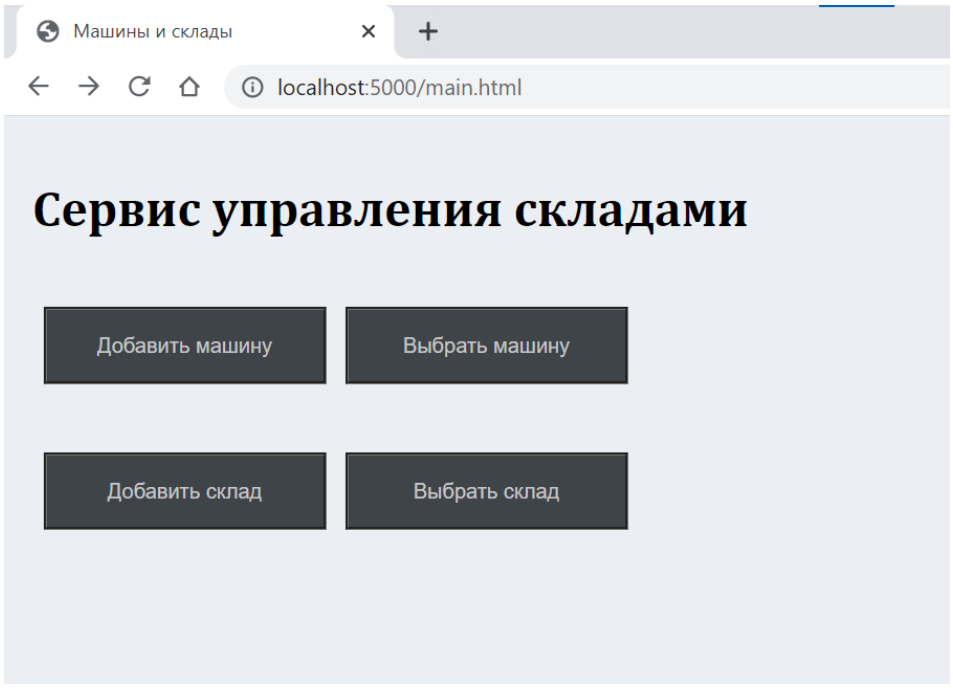
servC.js (сервер C)

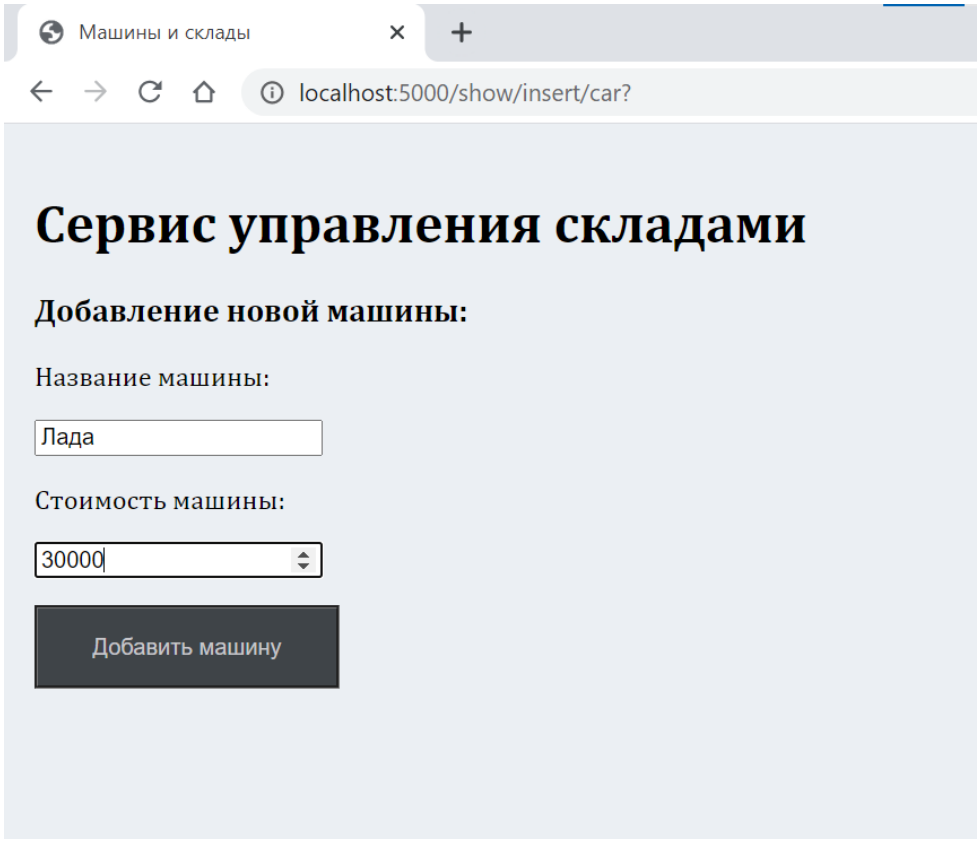
```
"use strict";

const express = require("express");
const request = require("request");

const app = express();
const port = 5000;
app.listen(port);
console.log(`Web-interface server on port ${port}`);

const way = __dirname + "/static";
app.use(express.static(way));
app.set("view engine", "hbs");

const car_addr = "http://localhost:5003";
const storage_addr = "http://localhost:5002";

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 sendPost(url, body, callback) {
    const headers = {};
    headers["Cache-Control"] = "no-cache, no-store, must-revalidate";
    headers["Connection"] = "close";

    request.post({
        url: url,
        body: body,
        headers: headers,
    }, function (error, response, body) {
        if(error) {
            callback(null);
        } else {
            callback(body);
        }
    });
}


app.get("/insert/car", function(request, response) {
    const name_ = request.query.name;
```

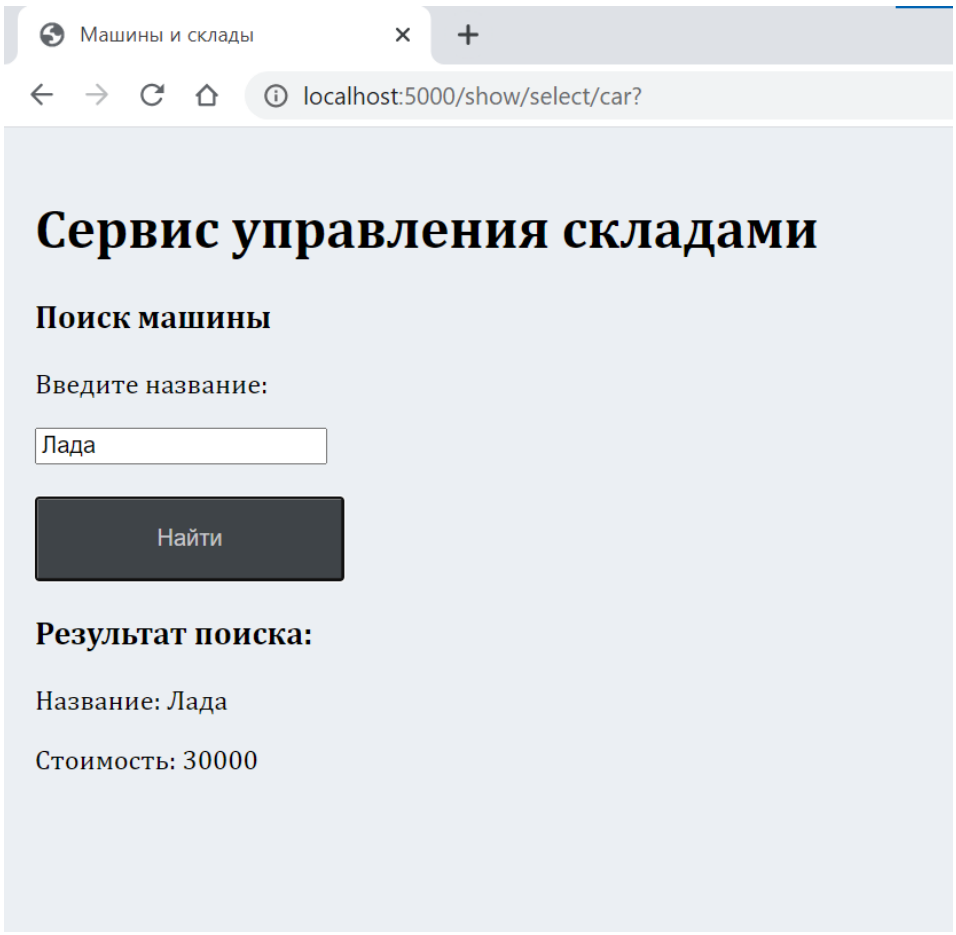
```
    const cost_ = request.query.cost;
    sendPost(car_addr + "/insert/record", JSON.stringify({
        name: name_,
        cost: cost_
    }), function(answerString) {
        response.end("Result: " + answerString);
    });
});

app.get("/insert/storage", function(request, response) {
    const name_ = request.query.name;
    const car_arr_ = request.query.car_arr;
    sendPost(storage_addr + "/insert/record", JSON.stringify({
        name: name_,
        car_arr: car_arr_
    }), function(answerString) {
        response.end("Result: " + answerString);
    });
});

app.get("/select/car", function(request, response) {
    const name_ = request.query.name;
    sendPost(car_addr + "/select/record", JSON.stringify(name_), function(answerS
tring) {
        response.end(answerString);
    });
});


app.get("/select/storage", function(request, response) {
    const name_ = request.query.name;
    sendPost(storage_addr + "/select/record", JSON.stringify(name_), function(ans
werString) {
        response.end(answerString);
    });
});


// Jump to other pages
app.get("/show/insert/car", function(request, response) {
    response.render("in_car.hbs", null);
});
app.get("/show/select/car", function(request, response) {
    response.render("out_car.hbs", null);
});

app.get("/show/insert/storage", function(request, response) {
    sendPost(car_addr + "/select/all", "", function(answerString) {
        let obj = { car_arr: JSON.parse(answerString)}
        response.render("in_storage.hbs", obj);
    });
```

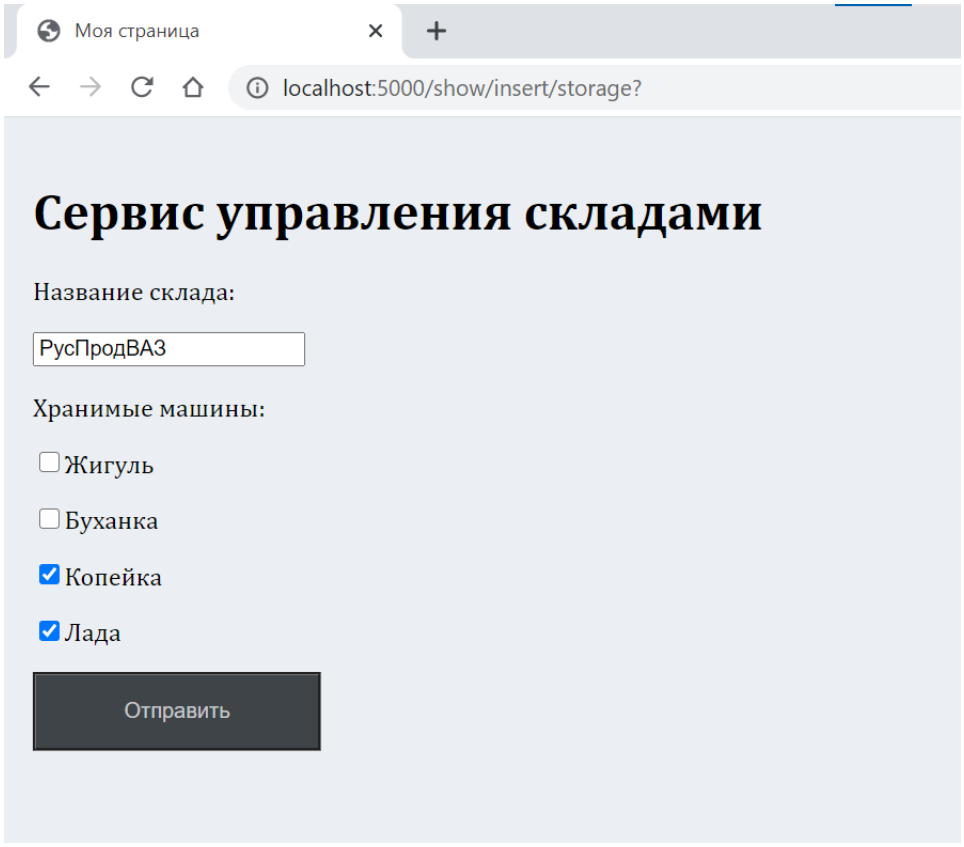
```
});
app.get("/show/select/storage", function(request, response) {
    response.render("out_storage.hbs", null);
});
```

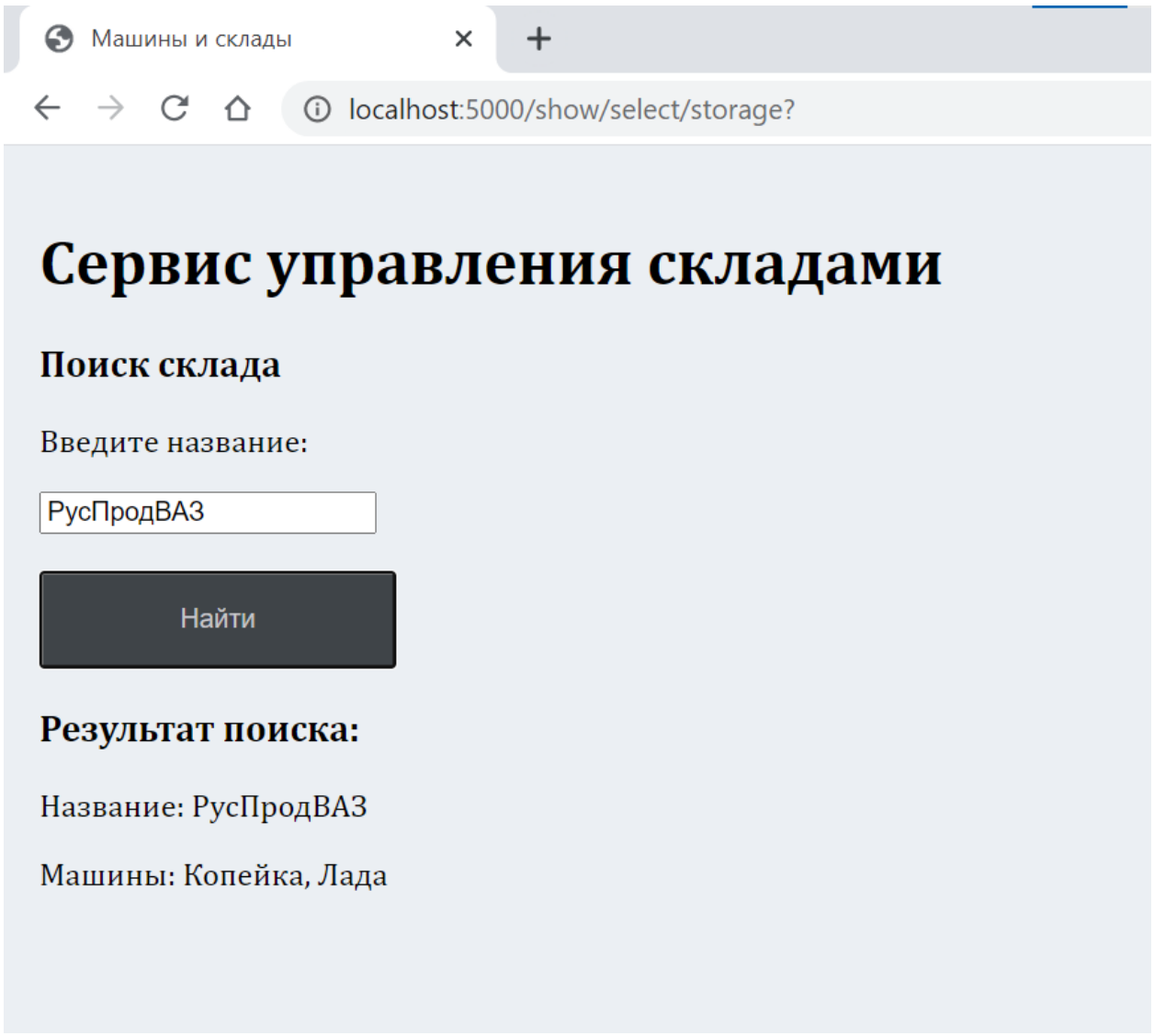
Тесты






Машины и склады
localhost:5000/show/select/car?
Сервис управления складами
Поиск машины
Введите название:
Лада
Найти
Результат поиска:
Название: Лада
Стоимость: 30000


Моя страница
localhost:5000/show/insert/storage?
Сервис управления складами
Название склада:
РусПродВАЗ
Хранимые машины:
Жигуль
Буханка
Копейка
Лада
Отправить


Машины и склады
localhost:5000/show/select/storage?
Сервис управления складами
Поиск склада
Введите название:
РусПродВАЗ
Найти
Результат поиска:
Название: РусПродВАЗ
Машины: Копейка, Лада

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**.

Программная реализация

factorial.js

```
"use strict"

function factroial(n) {
    let ans = 1;
    for (let i=2; i<=n; i++)
        ans *= i;
    return ans;
}

console.log(factroial(parseInt(process.argv[2])));
```

arr_factorial.js

```
"use strict;"

const execute = require("child_process").execSync;

function get_factorial(n) {
    const options = {encoding : 'utf8'}
    const command = `node factorial.js ${n}`;
    const ans = execute(command, options);
    return parseInt(ans);
}

console.log("Факториалы:");
for (let i=2; i<process.argv.length; i++)
    console.log(`${process.argv[i]}! = ${get_factorial(process.argv[i])}`);
```

obj

Тесты

```
PS D:\Work\JS-bmstu\lab4\task_7_2> node .\arr_factorial.js 1 2 3 4 5 15
Факториалы:
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
15! = 1307674368000
```

Вывод

В рамках лабораторной работы было выполнено ознакомление и практическое закрепление основ работы с взаимодействием серверов, скриптами и дочерними процессами.