



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

ИУ7-526

(Группа)

\_\_\_\_\_  
(Подпись, дата)

Кузин А.А.

(И.О. Фамилия)

Преподаватель

\_\_\_\_\_  
(Подпись, дата)

Попов А.Ю.

(И.О. Фамилия)

Москва, 2020

# Цель

Целью данной лабораторной работы является знакомство с взаимодействием между серверами, передачей параметров скриптам и вызовом дочерних процессов.

## Часть 1

### Задание 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).

## Листинг программы:

**Код серверной части сервера С:**

```
"use strict"
"use strict";

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

const app = express();
const port = 5000;
app.listen(port);

app.use(express.static(__dirname + "/src"));
console.log(`Server listens port ${port}`);

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);
    }
  });
}

function loadBody(request, callback) {
  let body = [];
  request.on('data', (chunk) => {
    body.push(chunk);
  }).on('end', () => {
```

```

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

app.post("/save/car", function (request, response) {
    loadBody(request, function (body) {
        const obj = JSON.parse(body);
        const type = obj.type;
        const cost = obj.cost;
        sendPost("http://localhost:5002/insert/record", JSON.stringify({
            type: type,
            cost: cost
        })), function(answerString){
            response.end(answerString);
        })
    });
});

app.post("/get/car", function (request, response) {
    loadBody(request, function (body) {
        const obj = JSON.parse(body);
        const type = obj.type;
        sendPost("http://localhost:5002/select/record", JSON.stringify({
            type: type
        })), function (answerString) {
            response.end(answerString);
        })
    });
});

app.post("/save/whouse", function (request, response) {
    loadBody(request, function (body) {
        const obj = JSON.parse(body);
        const name = obj.name;
        const cars = obj.cars;
        sendPost("http://localhost:5001/insert/record", JSON.stringify({
            name: name,
            cars: cars
        })), function (answerString) {
            response.end(answerString);
        })
    });
});

app.post("/get/whouse", function (request, response) {
    loadBody(request, function (body) {
        const obj = JSON.parse(body);
        const name = obj.name;
        console.log(name);
        sendPost("http://localhost:5001/select/record", JSON.stringify({
            name: name

```

```

    )), function (answerString) {
        response.end(answerString);
    })
    });
});

```

#### **Код клиентской части:**

```
"use strict";
```

```

function ajaxPost(urlString, bodyString, callback) {
    let r = new XMLHttpRequest();
    r.open("POST", urlString, true);
    r.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
    r.send(bodyString);
    r.onload = function () {
        callback(r.response);
    }
}

```

```

function addCar(){
    const typeField = document.getElementById("car-type");
    const costField = document.getElementById("car-cost");
    let resField = document.getElementById("result-label");

    ajaxPost("/save/car", JSON.stringify({
        type: typeField.value,
        cost: costField.value
    })), function(answerString){
        resField.innerHTML = JSON.parse(answerString).answer;
    });
}

```

```

function getCar() {
    const typeField = document.getElementById("car-type2");
    let resField = document.getElementById("result-label2");

    ajaxPost("/get/car", JSON.stringify({
        type: typeField.value,
    })), function (answerString) {
        const answerObject = JSON.parse(answerString);
        const found = answerObject.isFound;
        const cost = answerObject.cost;
        if (found)
            resField.innerHTML = 'Cost = ' + cost;
        else
            resField.innerHTML = 'Not found';
    });
}

```

```

function addWarehouse() {
    const nameField = document.getElementById("whouse-name");
    const carsField = document.getElementById("whouse-cars");

```

```

let resField = document.getElementById("result-label3");

const carsString = carsField.value;
const cars = carsString.split(",");

ajaxPost("/save/whouse", JSON.stringify({
  name: nameField.value,
  cars: cars
}), function (answerString) {
  resField.innerHTML = JSON.parse(answerString).answer;
});
}

function getWarehouse() {
  const nameField = document.getElementById("whouse-name2");
  let resField = document.getElementById("result-label4");

  ajaxPost("/get/whouse", JSON.stringify({
    name: nameField.value,
  }), function (answerString) {
    const answerObject = JSON.parse(answerString);
    const found = answerObject.isFound;
    const cars = answerObject.cars;
    if (found)
      resField.innerHTML = 'Cars: ' + cars;
    else
      resField.innerHTML = 'Not found';
  });
}

```

### **Код серверной части сервера А:**

```

"use strict";

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

const app = express();
const port = 5002;
app.listen(port);
console.log(`Server on port ${port}`);

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 obj = JSON.parse(body);
        const record = { type: obj.type, cost: obj.cost };
        const recordString = JSON.stringify(record) + "\n";

        fs.appendFileSync("src/cars.txt", recordString);

        response.end(JSON.stringify({
            answer: "Saved"
        }));
    });
});

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

        let strings = fs.readFileSync("src/cars.txt", "utf-8");
        strings = strings.split("\n");

        let found = false;
        let cost = 0;
        for (let i = 0; i < strings.length && strings[i] !== ""; i++){
            const tobj = JSON.parse(strings[i]);
            if (tobj.type === obj.type){
                cost = tobj.cost;
                found = true;
            }
        }

        response.end(JSON.stringify({
            isFound: found,
            cost: cost
        }));
    });
});

```

### **Код серверной части сервера В:**

```
"use strict";
```

```

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

```

```

const app = express();
const port = 5001;
app.listen(port);
console.log(`Server on port ${port}`);

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 obj = JSON.parse(body);
    const record = { name: obj.name, cars: obj.cars };
    const recordString = JSON.stringify(record) + "\n";

    fs.appendFileSync("src/whouses.txt", recordString);

    response.end(JSON.stringify({
      answer: "Saved"
    }));
  });
});

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

    let strings = fs.readFileSync("src/whouses.txt", "utf-8");
    strings = strings.split("\n");

    let found = false;
    let cars = [];
    for (let i = 0; i < strings.length && strings[i] !== ""; i++) {
      const tobj = JSON.parse(strings[i]);
      if (tobj.name === obj.name) {
        cars = tobj.cars;
        found = true;
      }
    }
  });
});

```



```

    }

    response.end(JSON.stringify({
      isFound: found,
      cars: cars
    }));
  });
});

```

### Код страницы:

```

<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <title>Моя страница</title>
  <link rel="stylesheet" href="/style.css">
</head>

<body>
  <div class="container">
    <div class="content">
      <h1>Добавить машину</h1>

      <p>Тип машины</p>
      <input id="car-type" type="text" spellcheck="false" autocomplete="off">

      <p>Стоимость</p>
      <input id="car-cost" type="number" spellcheck="false" autocomplete="off">

      <br>
      <br>

      <button onclick="addCar()">Сохранить машину</button>

      <br>
      <br>

      <h2 id="result-label"></h2>
    </div>

    <div class="content">
      <h1>Получить стоимость машины</h1>

      <p>Тип машины</p>
      <input id="car-type2" type="text" spellcheck="false" autocomplete="off">

      <br>
      <br>
    </div>
  </div>

```

```

    <button onclick="getCar()">Получить стоимость машины</button>

    <br>
    <br>

    <h2 id="result-label2"></h2>
</div>

<div class="content">
    <h1>Добавить склад</h1>

    <p>Название склада</p>
    <input id="whouse-name" type="text" spellcheck="false" autocomplete="off">

    <p>Список машин</p>
    <input id="whouse-cars" type="text" spellcheck="false" autocomplete="off",
placeholder="Названия машин">
    <p>Вводятся через запятую</p>

    <br>
    <br>

    <button onclick="addWarehouse()">Сохранить склад</button>

    <br>
    <br>

    <h2 id="result-label3"></h2>
</div>

<div class="content">
    <h1>Получить информацию о складе</h1>

    <p>Название склада</p>
    <input id="whouse-name2" type="text" spellcheck="false" autocomplete="off">
    <br>
    <br>

    <button onclick="getWarehouse()">Сохранить склад</button>

    <br>
    <br>

    <h2 id="result-label4"></h2>
</div>
</div>
<script src="/code.js"></script>
</body>

</html>

```

## Тесты

Страница:

## Добавить машину

Тип машины

Стоимость

Сохранить машину

## Получить стоимость машины

Тип машины

Получить стоимость машины

## Добавить склад

Название склада

Список машин

Названия машин

Вводятся через запятую

Сохранить склад

## Получить информацию о складе

Название склада

Получить информацию о складе

## Содержимое файла машин до операций:

Тип машины

  

Стоимость

Ввод:

Результат: **Saved**

Тип машины

  

Стоимость

Ввод:

Результат: **Saved**

Содержимое файла:

```
{"type":"Volvo","cost":"123"}
```

```
{"type":"Car","cost":"24"}
```

## Получение информации о машинах:

### Получить стоимость машины

Тип машины

  

**Not found**

### Получить стоимость машины

Тип машины

  

**Cost = 24**

## Добавление склада:

## Добавить склад

Название склада

Список машин

Вводятся через запятую

**Saved**

## Добавить склад

Название склада

Список машин

Вводятся через запятую

**Saved**

**Содержимое файла:**

```
{"name": "Izm", "cars": ["Car", "Volvo", "Onemore"]}
```

```
{"name": "Another", "cars": ["Merc", "bmw"]}
```

**Получение информации о складе:**

## Получить информацию о складе

Название склада

**Cars: Merc,bmw**

## Получить информацию о складе

Название склада

NotPresent

Получить информацию о складе

**Not found**

## Задание 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:

**Код для расчёта факториала:**

```
"use strict";
```

```
const number = process.argv[2];
```

```
let factorial = 1;
```

```
for (let i = 2; i <= number; i++)
```

```
    factorial *= i;
```

```
console.log(factorial);
```

**Код страницы:**

```
"use strict";
```

```
const execSync = require('child_process').execSync;
```

```
let i = 2;
```

```
while(true){
```

```
    const number = "" + process.argv[i];
```

```

if (!isNaN(parseInt(number)))
    console.log(execSync(`node factorial.js ${number}`, {encoding: 'utf8'}));
else
    break;
i++;
}

```

## Тесты

```

E:\study\5sem\arch\lab_04\task_2>node array_factorial.js 4 2 6 3
24

2

720

6

E:\study\5sem\arch\lab_04\task_2>node array_factorial.js 0
1

```

```

E:\study\5sem\arch\lab_04\task_2>node array_factorial.js 1 2 3
1

2

6

```

## Часть 2

### Задание 1

С клавиатуры считываются числа А и В. Необходимо вывести на экран все числа Фибоначчи, которые принадлежат отрезку от А до В.

### Листинг программы

```

ok.
input(A, B) :- read(A), read(B); ok.
fibonacci(X1, X2, A, B) :- X1 >= A, X1 <= B, write(X1), nl, T is (X1+X2), fibonacci(X2, T, A, B);
    X1 <= B, T is (X1+X2), fibonacci(X2, T, A, B);
ok.
f :- input(A, B), fibonacci(1, 1, A, B) ; ok.

```

### Тесты:

Вход: 1 10

```
1
1
2
3
5
8
Результат:
```

Вход: 10 50

```
13
21
34
Результат:
```

## Вывод

В результате выполнения данной лабораторной работы я получил навыки работы с форматом JSON, его обработкой, работы с файлами, а также навыки поднятия и использование серверов используя express.