



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»

Отчёт

по лабораторной работе № 4

Название: Взаимодействие между серверами. Пролог.

Дисциплина: Архитектура ЭВМ

Студент

ИУ7-55Б

(Группа)

Д.О. Склифасовский

(Подпись, дата)

(И.О. Фамилия)

Преподаватель

А.Ю. Попов

(Подпись, дата)

(И.О. Фамилия)

Москва, 2020

Содержание

| | |
|-----------------------|-----------|
| Введение | 3 |
| 1 Задание 7.1 | 4 |
| 1.1 Условие | 4 |
| 1.2 Решение | 4 |
| 1.3 Тесты | 13 |
| 2 Задание 7.1 | 16 |
| 2.1 Условие | 16 |
| 2.2 Решение | 16 |
| 2.3 Тесты | 17 |
| 3 Задание 8.1 | 18 |
| 3.1 Условие | 18 |
| 3.2 Решение | 18 |
| 3.3 Тесты | 18 |
| 4 Задание 8.2 | 19 |
| 4.1 Условие | 19 |
| 4.2 Решение | 19 |
| 4.3 Тесты | 19 |
| Вывод | 20 |

Введение

Цель работы: Научиться работать с взаимодействиями между серверами. Познакомиться с прологом.

1 Задание 7.1

1.1 Условие

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).

1.2 Решение

Листинг 1 – Сервер А

```
1  "use strict";
2
3  const express = require("express");
4  const fs = require("fs");
5
6  const app = express();
7  const port = 1010;
8  app.listen(port);
9  console.log("Server on port " + port);
10
11 app.use(function(req, res, next) {
12     res.header("Cache-Control", "no-cache, no-store, must-revalidate");
13     res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With,
        Content-Type, Accept");
14     res.header("Access-Control-Allow-Origin", "*");
15     next();
16 });
17
18 function loadBody(request, callback) {
19     let body = [];
20     request.on('data', (chunk) => {
21         body.push(chunk);
22     }).on('end', () => {
23         body = Buffer.concat(body).toString();
24         callback(body);
25     });
26 }
27
28 function insertRecord(carName, carCost) {
29     let info = JSON.parse(fs.readFileSync("cars.txt", "utf-8"));
30     let check = true;
31     for (let i = 0; i < info.length; i++) {
32         const obj = info[i];
33         if (obj["name"] == carName) {
34             check = false;
35             break;
36         }
37     }
38     if (check) {
```

```

39     info.push({"name" : carName, "cost" : carCost});
40     fs.writeFileSync("cars.txt", JSON.stringify(info));
41 }
42 return check;
43 }
44
45 app.post("/insert/record", function(request, response) {
46     loadBody(request, function(body) {
47         const obj = JSON.parse(body);
48         const carName = obj.carName;
49         const carCost = obj.carCost;
50         let res = insertRecord(carName, carCost);
51         if (res) {
52             response.end(JSON.stringify({
53                 answer: "Written down (OK)."
54             }));
55         } else {
56             response.end(JSON.stringify({
57                 answer: "Already written down (ERROR)."
58             }));
59         }
60     });
61 });
62
63 function selectRecord(carName) {
64     let info = JSON.parse(fs.readFileSync("cars.txt", "utf-8"));
65     for (let i = 0; i < info.length; i++) {
66         const obj = info[i];
67         if (obj["name"] == carName) {
68             return obj;
69         }
70     }
71
72     return null;
73 }
74
75 app.post("/select/record", function(request, response) {
76     loadBody(request, function(body) {
77         const obj = JSON.parse(body);
78         const carName = obj.carName;

```

```

79     console.log(carName);
80     let res = selectRecord(carName);
81     response.end(JSON.stringify({
82         answer: res
83     }));
84 });
85 };

```

Листинг 2 – Сервер В

```

1  "use strict";
2
3  const express = require("express");
4  const fs = require("fs");
5
6  const app = express();
7  const port = 1011;
8  app.listen(port);
9  console.log("Server on port " + port);
10
11 app.use(function(req, res, next) {
12     res.header("Cache-Control", "no-cache, no-store, must-revalidate");
13     res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With,
14         Content-Type, Accept");
15     res.header("Access-Control-Allow-Origin", "*");
16     next();
17 });
18
19 function loadBody(request, callback) {
20     let body = [];
21     request.on('data', (chunk) => {
22         body.push(chunk);
23     }).on('end', () => {
24         body = Buffer.concat(body).toString();
25         callback(body);
26     });
27 }
28
29 function insertRecord(warehouse, cars) {
30     let info = JSON.parse(fs.readFileSync("carWarehouse.txt", "utf-8"));
31     let check = true;

```

```

31     for (let i = 0; i < info.length; i++) {
32         const obj = info[i];
33         if (obj["warehouse"] == warehouse) {
34             check = false;
35             break;
36         }
37     }
38     if (check) {
39         const obj = {"warehouse" : warehouse, "cars" : cars}
40         info.push(obj);
41         fs.writeFileSync("carWarehouse.txt", JSON.stringify(info));
42     }
43     return check;
44 }
45
46 app.post("/insert/record", function(request, response) {
47     loadBody(request, function(body) {
48         const obj = JSON.parse(body);
49         const warehouse = obj.wareHouseName;
50         const cars = obj.arrOfCars;
51         let res = insertRecord(warehouse, cars);
52         if (res) {
53             response.end(JSON.stringify({
54                 answer: "Written down (OK)."
55             }));
56         } else {
57             response.end(JSON.stringify({
58                 answer: "Already written down (ERROR)."
59             }));
60         }
61     });
62 });
63
64 function selectRecord(warehouse) {
65     let info = JSON.parse(fs.readFileSync("carWarehouse.txt", "utf-8"));
66     for (let i = 0; i < info.length; i++) {
67         const obj = info[i];
68         if (obj["warehouse"] == warehouse) {
69             return obj;
70         }

```



```

71     }
72     return null;
73 }
74
75 app.post("/select/record", function(request, response) {
76     loadBody(request, function(body) {
77         const obj = JSON.parse(body);
78         const warehouse = obj.wareHouseName;
79         let res = selectRecord(warehouse);
80         response.end(JSON.stringify({
81             answer: res
82         }));
83     });
84 });

```

Листинг 3 – Сервер С

```

1     "use strict";
2
3     const express = require("express");
4     const request = require("request");
5
6     const app = express();
7     const port = 1100;
8     app.listen(port);
9     console.log('Server on port ${port}');
10
11     const way = __dirname + "/static";
12     app.use(express.static(way));
13
14     app.use(function(req, res, next) {
15         res.header("Cache-Control", "no-cache, no-store, must-revalidate");
16         res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With,
17             Content-Type, Accept");
18         res.header("Access-Control-Allow-Origin", "*");
19         next();
20     });
21
22     function sendPost(url, body, callback) {
23         const headers = {};
24         headers["Cache-Control"] = "no-cache, no-store, must-revalidate";

```

```

24     headers["Connection"] = "close";
25     request.post({
26         url: url,
27         body: body,
28         headers: headers,
29     }, function (error, response, body) {
30         if(error) {
31             callback(null);
32         } else {
33             callback(body);
34         }
35     });
36 }
37
38 app.get("/insertCar", function(request, response) {
39     const carName = request.query.car;
40     const carCost = request.query.cost;
41     sendPost("http://localhost:1010/insert/record", JSON.stringify({
42         carName: carName,
43         carCost: carCost
44     }), function(answerString) {
45         const answerObject = JSON.parse(answerString);
46         const answer = answerObject.answer;
47         response.end(JSON.stringify({
48             result: answer
49         }));
50     });
51 });
52
53 app.get("/selectCar", function(request, response) {
54     const carName = request.query.car;
55     sendPost("http://localhost:1010/select/record", JSON.stringify({
56         carName: carName,
57     }), function(answerString) {
58         const answerObject = JSON.parse(answerString);
59         const answer = answerObject.answer;
60         if (answer == null) {
61             response.end(JSON.stringify({
62                 result: "Didn't find"
63             }));

```

```

64     } else {
65         response.end(JSON.stringify({
66             result: answer
67         }));
68     }
69 });
70
71
72 function loadBody(request, callback) {
73     let body = [];
74     request.on('data', (chunk) => {
75         body.push(chunk);
76     }).on('end', () => {
77         body = Buffer.concat(body).toString();
78         callback(body);
79     });
80 }
81
82 async function checkCars(arr) {
83     let check = true;
84     for (let i = 0; i < arr.length; i++) {
85         sendPost("http://localhost:1010/select/record", JSON.stringify({
86             carName: arr[i]
87         }), function(answerString) {
88             const answerObject = JSON.parse(answerString);
89             const answer = answerObject.answer;
90             if (answer == null) {
91                 check = false;
92             }
93         });
94         await new Promise((resolve, reject) => setTimeout(resolve, 50));
95         if (!check) {
96             return check;
97         }
98     }
99     return check;
100 }
101
102 app.post("/insertWarehouse", function(request, response) {
103     loadBody(request, async function(body) {

```

```

104     const obj = JSON.parse(body);
105     const warehouse = obj["warehouse"];
106     const arr = obj["arr"];
107     let res = checkCars(arr);
108     await new Promise((resolve, reject) => setTimeout(resolve, arr.
        length * 50 + 100));
109     res = await Promise.resolve(res);
110     if (res) {
111         sendPost("http://localhost:1011/insert/record", JSON.stringify({
112             wareHouseName: warehouse,
113             arrOfCars : arr
114         }), function(answerString) {
115             const answerObject = JSON.parse(answerString);
116             const answer = answerObject.answer;
117             response.end(JSON.stringify({
118                 result: answer
119             }));
120         });
121     } else {
122         response.end(JSON.stringify({
123             result: "You didn't add all cars."
124         }));
125     }
126 });
127
128
129 async function getCars(arr) {
130     let resArr = [];
131     for (let i = 0; i < arr.length; i++) {
132         let curObj;
133         sendPost("http://localhost:1010/select/record", JSON.stringify({
134             carName: arr[i]
135         }), function(answerString) {
136             const answerObject = JSON.parse(answerString);
137             const answer = answerObject.answer;
138             curObj = [answer["name"], answer["cost"]];
139         });
140         await new Promise((resolve, reject) => setTimeout(resolve, 50));
141         resArr.push(curObj);
142     }

```

```

143     return resArr;
144 }
145
146 app.post("/selectWarehouse", function(request, response) {
147     loadBody(request, function(body) {
148         const obj = JSON.parse(body);
149         const warehouse = obj["warehouse"];
150         sendPost("http://localhost:1011/select/record", JSON.stringify({
151             wareHouseName: warehouse
152         })), async function(answerString) {
153             const answerObject = JSON.parse(answerString);
154             const answer = answerObject.answer;
155             if (answer == null) {
156                 response.end(JSON.stringify({
157                     result: "Didn't find anything"
158                 }));
159             } else {
160                 let res = getCars(answer.cars);
161                 let resStr = "Result: " + "warehouse - " + answer.warehouse;
162                 await new Promise((resolve, reject) => setTimeout(resolve,
163                     answer.cars.length * 50 + 50));
164                 let res2 = await Promise.resolve(res);
165                 for (let i = 0; i < res2.length; i++) {
166                     resStr += ", " + String(i + 1) + ") car - " + (res2[i])[0] + "
167                                     , cost - " + (res2[i])[1];
168                 }
169                 response.end(JSON.stringify({
170                     result: resStr
171                 }));
172             }
173         });

```

1.3 Тесты

Запросы на сервер А

Создание нового типа машин

Введите название машины

Введите цену машины

Результат:

Получение информации о стоимости машины по её типу

Введите название машины

Результат:

Создание нового склада с находящимися в нём машинами

Введите название склада

Введите массив машин (формат: машина, машина, машина:

Результат:

Получение информации о машинах на складе по названию склада

Введите название склада

Результат:

Рисунок 1 – Изначальная страница

Запросы на сервер А

Создание нового типа машин

Введите название машины

Введите цену машины

Результат: Written down (OK).

Получение информации о стоимости машины по её типу

Введите название машины

Результат: name - Lada Granta, cost - 483900

Создание нового склада с находящимися в нём машинами

Введите название склада

Введите массив машин (формат: машина, машина, машина:

Результат: Written down (OK).

Получение информации о машинах на складе по названию склада

Введите название склада

Результат: склад - Lada Togliatti, 1) машина - Lada Granta, цена - 483900

Рисунок 2 – Результат

2 Задание 7.1

2.1 Условие

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`.

2.2 Решение

Листинг 4 – Файл task1.js

```
1  "use strict";
2
3  let number = Number(process.argv[2]);
4  let cur = 1;
5  for (let i = 2; i <= number; i++) {
6      cur *= i;
7  }
8  console.log(cur);
```

Листинг 5 – Файл task2.js

```
1  "use strict";
2
3  const execSync = require('child_process').execSync;
4
5  function useCmd(s) {
6      const options = {encoding: 'utf8'};
7      const cmd = s.toString();
8      const answer = execSync(cmd, options);
9      return answer.toString();
10 }
11
12 let arr = []
```

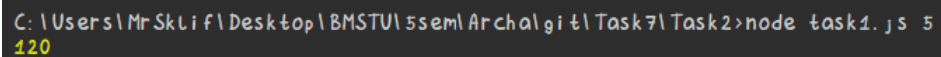


```

13   for (let i = 2; i < process.argv.length; i++) {
14       arr.push(Number(process.argv[i]));
15   }
16
17   for (let i = 0; i < arr.length; i++) {
18       const sumCommand = `node task1.js ${arr[i]}`;
19       let res = useCmd(sumCommand);
20       res = parseInt(res);
21       console.log(res);
22   }

```

2.3 Тесты

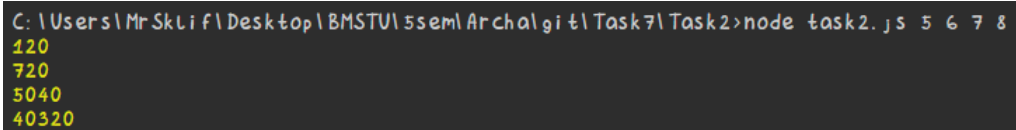


```

C:\Users\MrSkliF\Desktop\BMSTU\5sem\Archalg\Task7\Task2>node task1.js 5
120

```

Рисунок 3 – Первое задание



```

C:\Users\MrSkliF\Desktop\BMSTU\5sem\Archalg\Task7\Task2>node task2.js 5 6 7 8
120
720
5040
40320

```

Рисунок 4 – Второе задание

3 Задание 8.1

3.1 Условие

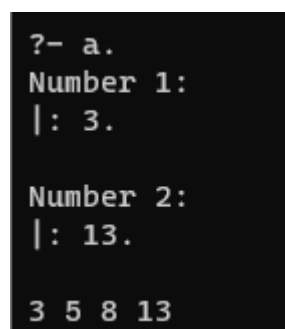
С клавиатуры считываются числа А и В. Необходимо вывести на экран все числа Фибоначчи, которые принадлежат отрезку от А до В.

3.2 Решение

Листинг 6 – Задание 1

```
1 out(A, B, Z, X, Y) :- checkOut(Z, A, B), Z_NEW = X + Y, X_NEW = Y, Y_NEW
    = Z_NEW, Z_NEW =< B, out(A,B,Z_NEW,X_NEW,Y_NEW).
2 checkOut(Z,A,B) :- Z_NEW is Z, (Z >= A, Z =< B -> write(Z_NEW), write("
    ");write("")).
3
4 a :- write("Number 1: "), nl,
5     read(A), nl,
6     write("Number 2: "), nl,
7     read(B), nl,
8     out(A, B, 1, 0, 1).
```

3.3 Тесты



```
?- a.
Number 1:
|: 3.

Number 2:
|: 13.

3 5 8 13
```

Рисунок 5 – Первое задание

4 Задание 8.2

4.1 Условие

С клавиатуры считываются числа A и B . Необходимо вывести на экран все числа, квадратный корень которых является целым числом. При этом, необходимо вывести только числа, которые принадлежат отрезку от A до B .

4.2 Решение

Листинг 7 – Задание 2

```
1 out(A, B, I) :- NEW_NUM = I * I, NEW_NUM =< B, checkOut(A, B, NEW_NUM),  
    NEW_I = I + 1, out(A, B, NEW_I).  
2 checkOut(A, B, NUM) :- NEW_NUM is NUM, (NUM >= A, NUM =< B -> write(  
    NEW_NUM), write(" "); write("")).  
3  
4 a :- write("Number 1: "), nl,  
5     read(A), nl,  
6     write("Number 2: "), nl,  
7     read(B), nl,  
8     out(A, B, 0).
```

4.3 Тесты

```
?- a.  
Number 1:  
|: 99.  
  
Number 2:  
|: 777.  
  
100 121 144 169 196 225 256 289 324 361 400 441 484 529 576 625 676 729
```

Рисунок 6 – Второе задание

Вывод

В ходе выполнения лабораторной работы я научился работать с взаимодействиями между серверами, также поработал с прологом.