



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»

Лабораторная работа № 4

Взаимодействие серверов и знакомство с языком Prolog

Студент Жигалкин Д.Р

Группа ИУ7-55Б

Оценка (баллы) _____

Москва.
2020 г.

Цель: Ознакомиться с созданием, запуском и взаимодействия нескольких серверов.

Task_7

```
1  "use strict";
2
3  // импорт библиотеки
4  const express = require("express");
5  const fs = require("fs");
6
7  // запускаем сервер
8  const app = express();
9  const port = 5002;
10 app.listen(port);
11 console.log("Server on port " + port);
12
13 // заголовки для ответа
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, Content-Type, Accept");
17   res.header("Access-Control-Allow-Origin", "*");
18   next();
19 });
20
21 // загрузка тела
22 function loadBody(request, callback) {
23   let body = [];
24   request.on('data', (chunk) => {
25     body.push(chunk);
26   }).on('end', () => {
27     body = Buffer.concat(body).toString();
28     callback(body);
29   });
30 }
31
32 // приём запроса
33 app.post("/insert/record", function(request, response) {
34   loadBody(request, function(body) {
35     const obj = JSON.parse(body);
36     const carName = obj.carname;
37     const carPrice = obj.carprice;
```

```

37     const carPrice = obj.carprice;
38
39     let carJson = JSON.stringify({
40         carname: carName, carprice: carPrice
41     });
42
43     fs.appendFileSync("carInfo.txt", "\n" + carJson);
44
45     response.json(JSON.stringify({
46         answer: "Car added ok"
47     }));
48 });
49 });
50
51 // npwñw ñanpoca
52 app.post("/select/record", function(request, response) {
53     loadBody(request, function(body) {
54         const obj = JSON.parse(body);
55         const carName = obj.carname;
56
57         let fileContent = fs.readFileSync("carInfo.txt", "utf8");
58
59         let beginIndex = fileContent.indexOf(carName);
60         let carprice = "Not found";
61
62         if (beginIndex != -1)
63         {
64
65             let i = beginIndex;
66
67             while (fileContent[i] != "{")
68             {
69                 i++;
70             }
71
72             let j = beginIndex;

```

```

73         let j = beginIndex;
74
75         while (fileContent[j] != "{")
76         {
77             j--;
78         }
79
80         let startIndex = j;
81         let endIndex = i;
82
83         let result = fileContent.slice(startIndex, endIndex + 1);
84
85         const answerObject = JSON.parse(result);
86         carprice = answerObject.carprice;
87     }
88
89     response.json(JSON.stringify({
90         answer: carprice
91     }));
92 });
93 });

```

```

1  "use strict";
2
3  // импорт библиотеки
4  const express = require("express");
5  const fs = require("fs");
6
7  // запускаем сервер
8  const app = express();
9  const port = 5000;
10 app.listen(port);
11 console.log("Server on port " + port);
12
13 // заголовки для ответа
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, Content-Type, Accept");
17   res.header("Access-Control-Allow-Origin", "*");
18   next();
19 });
20
21 // загрузка тела
22 function loadBody(request, callback) {
23   let body = [];
24   request.on('data', (chunk) => {
25     body.push(chunk);
26   }).on('end', () => {
27     body = Buffer.concat(body).toString();
28     callback(body);
29   });
30 }
31
32 // приём запроса
33 app.post("/insert/record", function(request, response) {
34   loadBody(request, function(body) {
35     const obj = JSON.parse(body);
36     const storageName = obj.storageName;
37     const carsName = obj.carsName;
38

```

```

38
39     console.log(storageName);
40     let carJson = JSON.stringify({
41       storageName: storageName, carsName: carsName
42     });
43
44     fs.appendFileSync("storageInfo.txt", "\n" + carJson);
45
46     response.json(JSON.stringify({
47       answer: "Storage added ok"
48     }));
49   });
50 });
51
52 // приём запроса
53 app.post("/select/record", function(request, response) {
54   loadBody(request, function(body) {
55     const obj = JSON.parse(body);
56     const storageName = obj.storageName;
57
58     let fileContent = fs.readFileSync("storageInfo.txt", "utf8");
59
60     let beginIndex = fileContent.indexOf(storageName);
61     let i = beginIndex;
62
63     while (fileContent[i] != ")")
64     {
65       i++;
66     }
67
68     let j = beginIndex;
69
70     while (fileContent[j] != "{")
71     {
72       j--;
73     }

```

```
61     let i = beginIndex;
62
63     while (fileContent[i] != "}")
64     {
65         i++;
66     }
67
68     let j = beginIndex;
69
70     while (fileContent[j] != "{")
71     {
72         j--;
73     }
74
75     let startIndex = j;
76     let endIndex = i;
77
78     let result = fileContent.slice(startIndex, endIndex + 1);
79     const answerObject = JSON.parse(result);
80     const carsName = answerObject.carsname;
81
82     response.json(JSON.stringify({
83         answer: carsName
84     }));
85 });
86 });
87
88
```

```

1  const express = require("express");
2  const request = require("request");
3
4  const app = express();
5  var port = 3000;
6  app.listen(port);
7  console.log(`Server on port ${port}`);
8
9  const way = __dirname + "/static";
10 app.use(express.static(way));
11
12 app.get("/", function(request, response){
13
14     response.sendFile(__dirname + "/static/server-C.html");
15 });
16
17 // функция для отправки POST запроса на другой сервер
18 function sendPost(url, body, callback) {
19     // задаём заголовки
20     const headers = {};
21     headers["Cache-Control"] = "no-cache, no-store, must-revalidate";
22     headers["Connection"] = "close";
23     // отправляем запрос
24     request.post({
25         url: url,
26         body: body,
27         headers: headers,
28     }, function (error, response, body) {
29         if(error) {
30             callback(null);
31         } else {
32             callback(body);
33         }
34     });
35 }
36
37 // принимаем GET запрос и отправляем POST запрос на другой сервер
38 app.get("/redirectA/insert/record", function(request, response) {

```

```

37 // принимаем GET запрос и отправляем POST запрос на другой сервер
38 app.get("/redirectA/insert/record", function(request, response) {
39     const carName = request.query.carName;
40     const carPrice = request.query.carPrice;
41
42     sendPost("http://localhost:5002/insert/record", JSON.stringify({
43         carname: carName,
44         carprice: carPrice
45     })), function(answerString) {
46         const answerObject = JSON.parse(answerString);
47         const answer = answerObject.answer;
48         response.json(answerString);
49     });
50 });
51
52 // принимаем GET запрос и отправляем POST запрос на другой сервер
53 app.get("/redirectA/select/record", function(request, response) {
54     const carName = request.query.carName;
55
56     sendPost("http://localhost:5002/select/record", JSON.stringify({
57         carname: carName
58     })), function(answerString) {
59         const answerObject = JSON.parse(answerString);
60         const answer = answerObject.answer;
61         response.json(answerString);
62     });
63 });
64
65 // принимаем GET запрос и отправляем POST запрос на другой сервер
66 app.get("/redirectB/insert/record", function(request, response) {
67     const storageName = request.query.storageName;
68     const carsName = request.query.carsName;
69
70     console.log(storageName, carsName);
71
72     sendPost("http://localhost:5000/insert/record", JSON.stringify({
73

```



```

72
73     sendPost("http://localhost:5000/insert/record", JSON.stringify({
74         storagename: storageName,
75         carsname: carsName
76     })), function(answerString) {
77         const answerObject = JSON.parse(answerString);
78         const answer = answerObject.answer;
79         response.json(answerString);
80     });
81 });
82
83
84 // принимаем GET запрос и отправляем POST запрос на другой сервер
85 app.get("/redirectB/select/record", function(request, response) {
86     const storageName = request.query.storageName;
87
88     console.log(storageName);
89
90     sendPost("http://localhost:5000/select/record", JSON.stringify({
91         storagename: storageName
92     })), function(answerString) {
93         const answerObject = JSON.parse(answerString);
94         const answer = answerObject.answer;
95         response.json(answerString);
96     });
97 });
98
99

```

Task_7


```

1  main :-
2      write("First Number: "), nl,
3      read(A), nl,
4      write("Second Number: "), nl,
5      read(B), nl,
6
7      calculate_border(A, BORDER0),
8      calculate_border(B, BORDER1),
9
10     calculate_answer_fibonacci(BORDER0, BORDER1).
11
12     fibonacci(0, 1) :- !.
13     fibonacci(1, 1) :- !.
14     fibonacci(N, F) :-
15         N1 is N-1,
16         N2 is N-2,
17         fibonacci(N1, F1),
18         fibonacci(N2, F2),
19         F is F1+F2.
20
21     calculate_border(X, N) :- tail(X, 0, N).
22
23     tail(X, N0, N) :- fibonacci(N0, F), (X >= F, N1 is N0 + 1, tail(X, N1, N); N = N0).
24
25     calculate_answer_fibonacci(N, N) :- !.
26     calculate_answer_fibonacci(N0, N1) :-
27         fibonacci(N0, F),
28         output_number(F),
29         N2 is N0 + 1,
30         calculate_answer_fibonacci(N2, N1).
31
32     output_number(X) :- X_NEW is X, write(X_NEW), write(" ").
33
34
35

```

```

1  main :-
2      write("First Number: "), nl,
3      read(A), nl,
4      write("Second Number: "), nl,
5      read(B), nl,
6      output_int_sqrt(A, B).
7
8     choice(Condition, Then) :- call(Condition) -> call(Then) ; true.
9
10    sqrt_check(X) :-
11        X1 is sqrt(X),
12        X2 is truncate(X1),
13        (X1 - X2) < 0.00001.
14
15    output_int_sqrt(A, A) :- !.
16    output_int_sqrt(A, B) :-
17        choice(sqrt_check(A), output_number(A)),
18        A1 is A + 1,
19        output_int_sqrt(A1, B).
20
21    output_number(X) :- X1 is X, write(X1), write(" ").
22

```

Вывод: Я ознакомился с созданием, запуском и взаимодействия нескольких серверов.