



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»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4
По курсу: "Архитектура ЭВМ"

Студент _____ Сукочева Алис
Группа _____ ИУ7-53Б
Название предприятия _____ МГТУ им. Н. Э. Баумана, каф. ИУ7
Тема Взаимодействие серверов. Дочерние процессы. Аргументы командной строки

Студент:	_____	<u>Сукочева А.</u>
	подпись, дата	Фамилия, И.О.
Преподаватель:	_____	<u>Попов А. Ю.</u>
	подпись, дата	Фамилия, И. О.

TASK_7.

Цель работы:

- Изучать и реализовать взаимодействие между серверами.
- Изучать и реализовать дочерние процессы.
- Изучать и реализовать `process.argv`.
- Изучить `prolog`.

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 1 — Код программы. TASK_7. Сервер А

```
1      "use strict";
2
3      // импорт библиотеки
4      const express = require("express");
5      const fs = require("fs");
6
7
8      // запускаем сервер
9      const app = express();
10     const port = 5003;
11     app.listen(port);
12     console.log("Server on port " + port);
13
14     // заголовки для ответа
```

```

15 app.use(function (req, res, next) {
16     res.header("Cache-Control", "no-cache, no-store, must-revalidate");
17     res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With,
        Content-Type, Accept");
18     res.header("Access-Control-Allow-Origin", "*");
19     next();
20 });
21
22 // Загрузка тела.
23 function loadBody(request, callback) {
24     let body = [];
25     request.on('data', (chunk) => {
26         body.push(chunk);
27     }).on('end', () => {
28         body = Buffer.concat(body).toString();
29         callback(body);
30     });
31 }
32
33 // Приём запроса.
34 app.post("/insert/record", function (request, response) {
35     loadBody(request, function (body) {
36         // Получаем данные.
37         const obj = JSON.parse(body);
38         const type = obj.type;
39         const price = obj.price;
40
41         // Открываем файл и парсим.
42         const fileName = "data_car.json";
43         const objInfo = fs.readFileSync(fileName, "utf-8");
44         const infoJson = JSON.parse(objInfo);
45         let answer = "Model is exist!";
46
47         let flag = true;
48         // Ищем модель.
49         for (let i in infoJson) {
50             if (infoJson[i].type === type) {
51                 // console.log(infoJson[i]);
52                 flag = false;
53                 break;
54             }
55         }
56
57         // Добавляем в файл информацию,
58         // Если такой модели еще нет.
59         if (flag) {
60             infoJson.push({ type, price })

```

```

61         fs.writeFileSync(fileName, JSON.stringify(infoJson, null, 4));
62         answer = "Model add";
63     }
64
65     response.end(JSON.stringify({ answer: answer }));
66 });
67 });
68
69 // Приём запроса.
70 app.post("/select/record", function (request, response) {
71     loadBody(request, function (body) {
72         // Получаем данные.
73         const obj = JSON.parse(body);
74         const type = obj.type;
75
76         // Открываем файл и парсим.
77         const fileName = "data_car.json";
78         const objInfo = fs.readFileSync(fileName, "utf-8");
79         const infoJson = JSON.parse(objInfo);
80
81         // Ответ пользователю.
82         let answer = "Model does not!";
83
84         // Ищем модель.
85         for (let i in infoJson) {
86             if (infoJson[i].type === type) {
87                 // console.log(infoJson[i]);
88                 answer = infoJson[i];
89                 break;
90             }
91         }
92
93         response.end(JSON.stringify({ answer: JSON.stringify(answer) }));
94     });
95 });

```

Листинг 2 — Код программы. TASK_7. Сервер В

```

1     "use strict";
2
3 // импорт библиотеки
4 const express = require("express");
5 const fs = require("fs");
6
7
8 // запускаем сервер
9 const app = express();

```

```

10 const port = 5002;
11 app.listen(port);
12 console.log("Server on port " + port);
13
14 // заголовки для ответа
15 app.use(function (req, res, next) {
16     res.header("Cache-Control", "no-cache, no-store, must-revalidate");
17     res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With,
        Content-Type, Accept");
18     res.header("Access-Control-Allow-Origin", "*");
19     next();
20 });
21
22 // Загрузка тела.
23 function loadBody(request, callback) {
24     let body = [];
25     request.on('data', (chunk) => {
26         body.push(chunk);
27     }).on('end', () => {
28         body = Buffer.concat(body).toString();
29         callback(body);
30     });
31 }
32
33 // Приём запроса.
34 app.post("/insert/record", function (request, response) {
35     loadBody(request, function (body) {
36         // Получаем данные.
37         const obj = JSON.parse(body);
38         const stock_name = obj.stock_name;
39         const car_array_str = obj.car_array;
40
41         // Открываем файл и парсим.
42         const fileName = "data_stock.json";
43         const objInfo = fs.readFileSync(fileName, "utf-8");
44         const infoJson = JSON.parse(objInfo);
45         let answer = "Stock is exist!";
46
47         let flag = true;
48         // Ищем склад.
49         for (let i in infoJson) {
50             if (infoJson[i].stock_name === stock_name) {
51                 // console.log(infoJson[i]);
52                 flag = false;
53                 break;
54             }
55         }

```

```

56
57     // Добавляем в файл информацию,
58     // Если такой модели еще нет.
59     if (flag) {
60         let car_array = car_array_str.split(" ");
61         infoJson.push({ stock_name, car_array })
62         fs.writeFileSync(fileName, JSON.stringify(infoJson, null, 4));
63         answer = "Model add";
64     }
65
66     response.end(JSON.stringify({ answer: answer }));
67 });
68 });
69
70 // Приём запроса.
71 app.post("/select/record", function (request, response) {
72     loadBody(request, function (body) {
73         // Получаем данные.
74         const obj = JSON.parse(body);
75         const name_stock_find = obj.name_stock_find;
76
77         // Открываем файл и парсим.
78         const fileName = "data_stock.json";
79         const objInfo = fs.readFileSync(fileName, "utf-8");
80         const infoJson = JSON.parse(objInfo);
81
82         // Ответ пользователю.
83         let answer = "Model does not!";
84
85         // Ищем модель.
86         for (let i in infoJson) {
87             if (infoJson[i].stock_name === name_stock_find) {
88                 // console.log(infoJson[i]);
89                 answer = infoJson[i];
90                 break;
91             }
92         }
93
94         response.end(JSON.stringify({ answer: JSON.stringify(answer) }));
95     });
96 });

```

Листинг 3 — Код программы. TASK_7.Сервер C

```

1     "use strict";
2
3     // импорт библиотек

```

```

4  const express = require("express");
5  const request = require("request");
6  const fs = require("fs");
7
8  const ENCODING = "utf-8"
9
10 // запускаем сервер
11 const app = express();
12 const port = 5000;
13 app.listen(port);
14 console.log('Server on port ${port}');
15
16 // Отправка статических файлов.
17 const way = __dirname + "/static";
18 app.use(express.static(way));
19
20 // заголовки в ответ клиенту
21 app.use(function (req, res, next) {
22     res.header("Cache-Control", "no-cache, no-store, must-revalidate");
23     res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With,
        Content-Type, Accept");
24     res.header("Access-Control-Allow-Origin", "*");
25     next();
26 });
27
28 // функция для отправки POST запроса на другой сервер
29 function sendPost(url, body, callback) {
30     // задаём заголовки
31     const headers = {};
32     headers["Cache-Control"] = "no-cache, no-store, must-revalidate";
33     headers["Connection"] = "close";
34     // отправляем запрос
35     request.post({
36         url: url,
37         body: body,
38         headers: headers,
39     }, function (error, response, body) {
40         if (error) {
41             callback(null);
42         } else {
43             callback(body);
44         }
45     });
46 }
47
48 app.get("/", (_request, response) => {

```

```

49     const fileContent = fs.readFileSync("static/" + "index.html",
50         ENCODING);
51     response.end(fileContent);
52 });
53 // принимаем GET запрос и отправляем POST запрос на другой сервер
54 app.get("/set_info_car/", (request, response) => {
55     const type = request.query.field_type_car;
56     const price = request.query.field_price_car;
57
58     sendPost("http://localhost:5003/insert/record", JSON.stringify(
59         { type, price }
60     ), function (answerString) {
61         const answerObject = JSON.parse(answerString);
62         const answer = answerObject.answer;
63         response.end("Answer: " + answer);
64     });
65 });
66
67 app.get("/get_info_car/", (request, response) => {
68     const type = request.query.field_type_car_find;
69
70     sendPost("http://localhost:5003/select/record", JSON.stringify(
71         { type }
72     ), function (answerString) {
73         const answerObject = JSON.parse(answerString);
74         const answer = answerObject.answer;
75         response.end("Answer: " + answer);
76     });
77 });
78
79 // принимаем GET запрос и отправляем POST запрос на другой сервер
80 app.get("/set_info_stock/", (request, response) => {
81     const stock_name = request.query.field_stock_name;
82     const car_array = request.query.field_car_array;
83
84     sendPost("http://localhost:5002/insert/record", JSON.stringify(
85         { stock_name, car_array }
86     ), function (answerString) {
87         const answerObject = JSON.parse(answerString);
88         const answer = answerObject.answer;
89         response.end("Answer: " + answer);
90     });
91 });
92
93 app.get("/get_info_stock/", (request, response) => {
94     const name_stock_find = request.query.field_name_stock_find;

```



```

95
96     sendPost("http://localhost:5002/select/record", JSON.stringify(
97         { name_stock_find }
98     ), function (answerString) {
99         const answerObject = JSON.parse(answerString);
100         const answer = answerObject.answer;
101         response.end("Answer: " + answer);
102     });
103 });

```

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 4 — Код программы. TASK_7. Задание 2. Главный процесс

```

1     "use strict";
2
3     // импортируем библиотеку
4     const execSync = require('child_process').execSync;
5
6     const MY_ARG = 2
7     const OPTIONS = { encoding: 'utf8' };
8
9     // Функция, для считывания аргументов
10    // Переданных в командной строке.
11    function readArgv(array) {
12        let i = MY_ARG;
13
14        while (process.argv[i])
15            array.push(parseInt(process.argv[i++]));
16
17        return array;
18    }
19
20    // Функция, вызывающая дочерний процесс
21    // Для каждого элемента из массива array.
22    // Дочерний процесс в свою очередь
23    // Считает факториал числа.
24    function arrayFactorial(array) {
25        let cmd;

```

```

26
27     for (let i in array) {
28         cmd = 'node factorial ${array[i]}'
29         console.log(execSync(cmd, OPTIONS))
30     }
31 }
32
33
34 function main() {
35     let array = [];
36     readArgv(array);
37     arrayFactorial(array);
38 }
39
40 main();

```

Листинг 5 — Код программы. TASK_2. Задание 2. Дочерний процесс.

```

1  "use strict";
2
3  // Функция, которая вычисляет факториал
4  // Числа, переданного аргументом командной строки.
5  function factorial() {
6      let num = parseInt(process.argv[2]);
7      let result = 1;
8
9      for (let i = 1; i < num; i++)
10         result *= i;
11
12     console.log(result);
13 }
14
15 factorial();

```

TASK_8

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

Листинг 6 — Числа Фибоначчи

```

1  writeNumber(X, A) :- X_NEW is X, ((X >= A) -> write(X_NEW), write(" "));
   write("").
2
3  fib(PREV, CURR, A, B) :-
4     writeNumber(PREV, A),
5     CURR =< B,
6     fib(CURR, PREV + CURR, A, B).
7

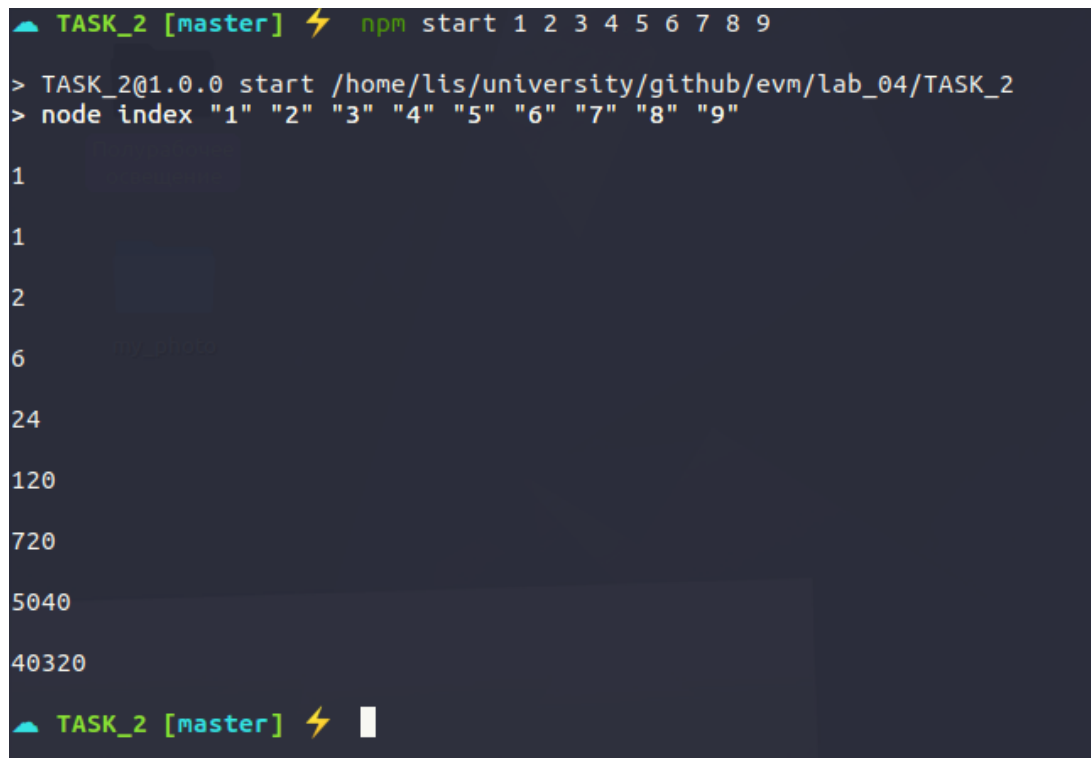
```

```
8 f :- read(A), read(B), fib(1, 1, A, B).
```

Вывод:

- Мы изучили и реализовали взаимодействие между серверами.
- Изучили и реализовали дочерние процессы.
- Изучили и реализовали process.argv.
- Изучили prolog.

Пример работы:



```
TASK_2 [master] npm start 1 2 3 4 5 6 7 8 9
> TASK_2@1.0.0 start /home/lis/university/github/evm/lab_04/TASK_2
> node index "1" "2" "3" "4" "5" "6" "7" "8" "9"
1
1
2
6
24
120
720
5040
40320
TASK_2 [master]
```

Рисунок 0.1 — Пример работы программы

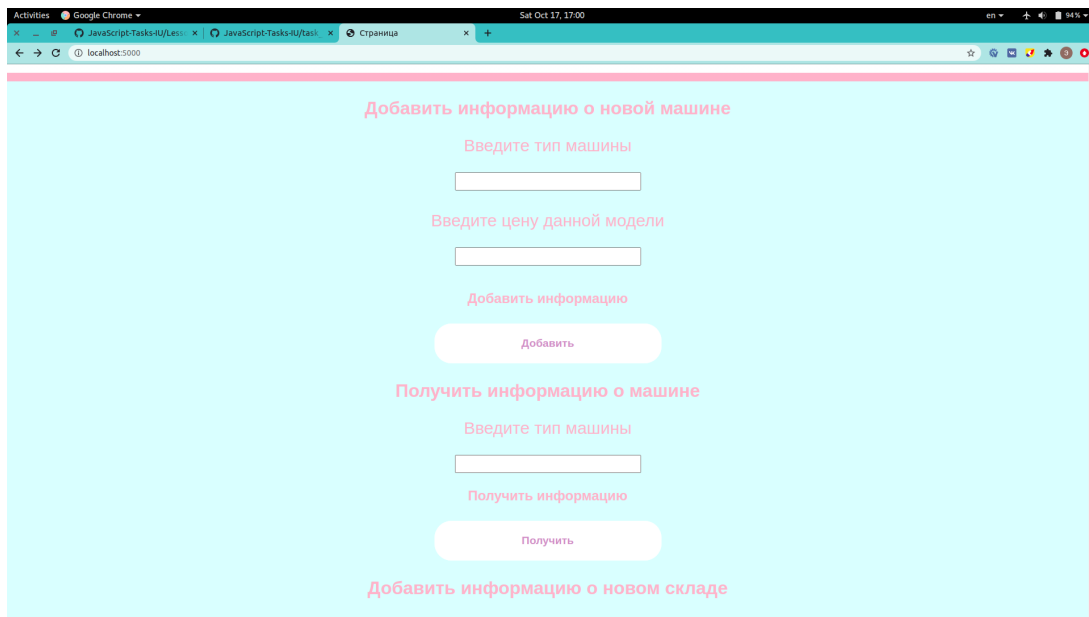


Рисунок 0.2 — Пример работы программы

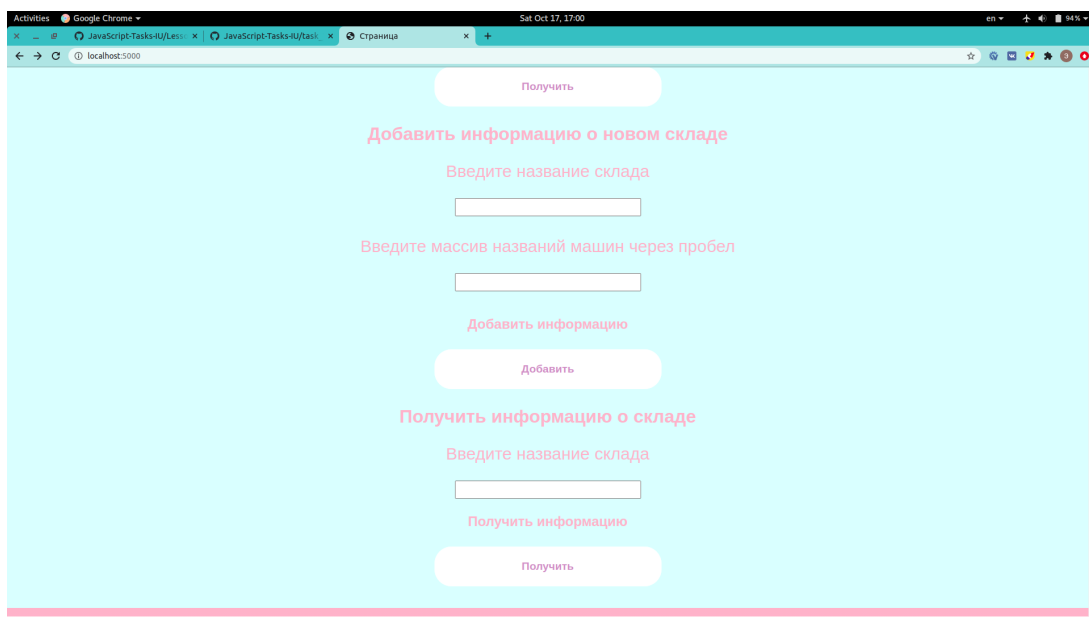


Рисунок 0.3 — Пример работы программы

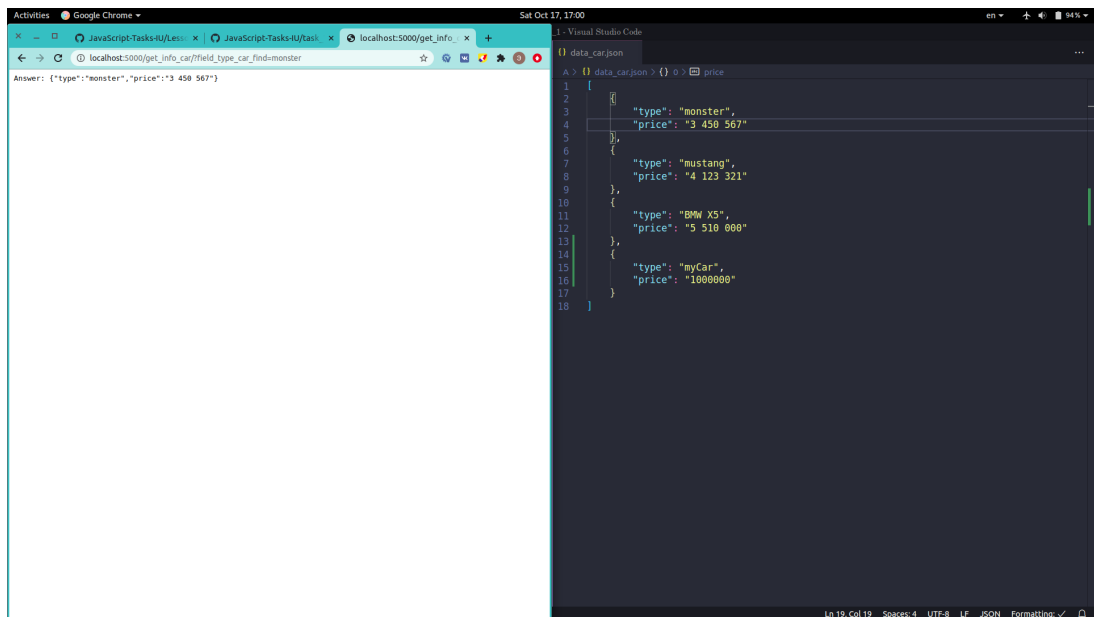


Рисунок 0.4 — Пример работы программы

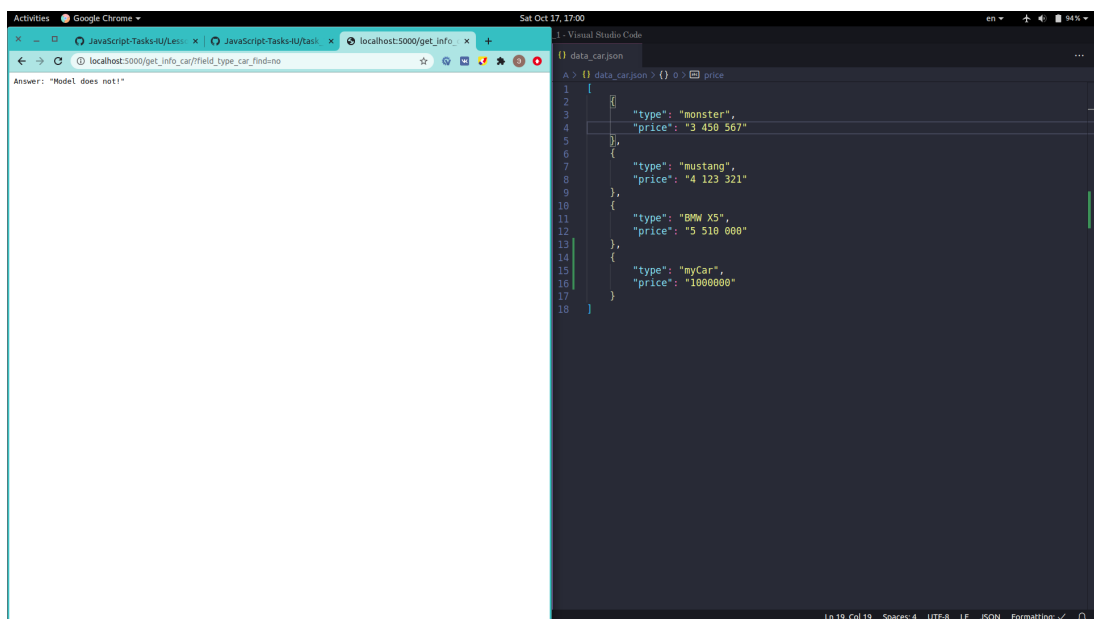


Рисунок 0.5 — Пример работы программы

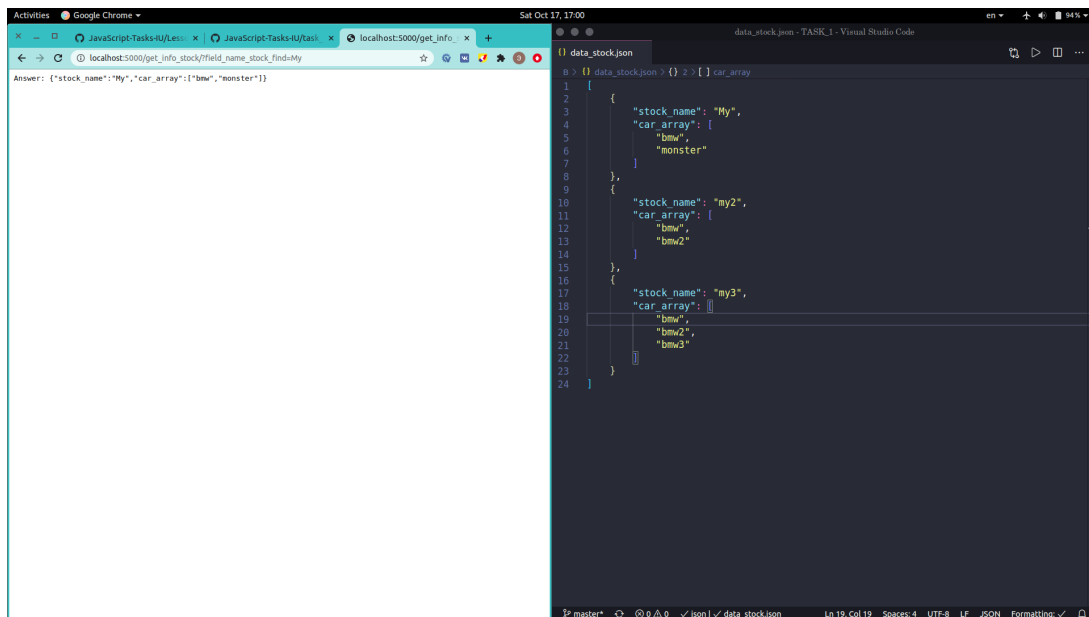


Рисунок 0.6 — Пример работы программы

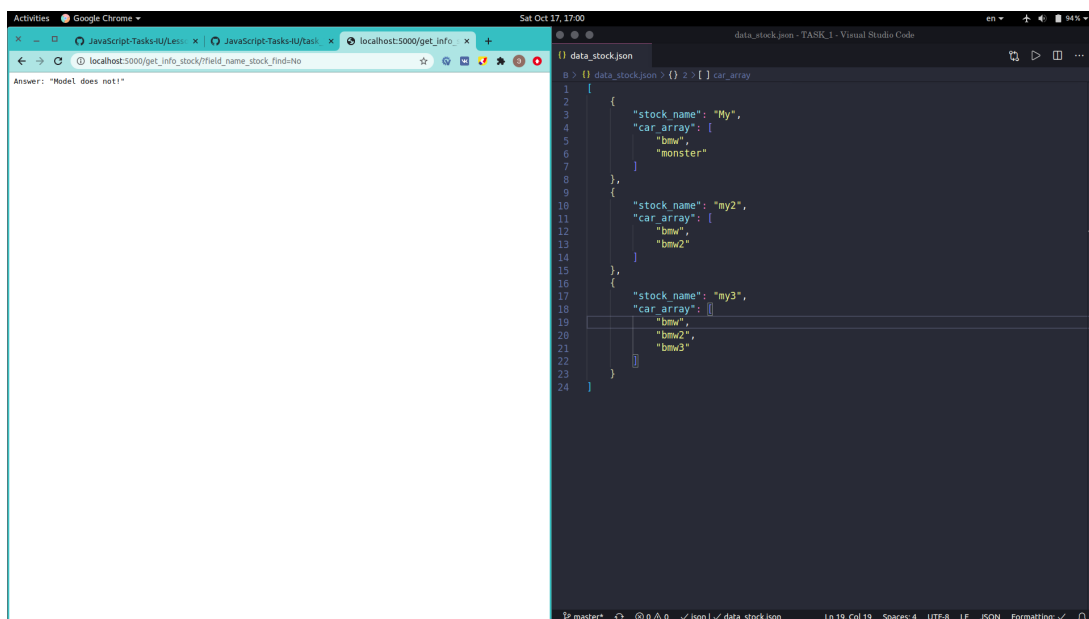


Рисунок 0.7 — Пример работы программы

```
prolog [master] ⚡ swipl fibonacci.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.1)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- f.
|: 8.
|: 100.
8 13 21 34 55 89
false.

?- 
```

Рисунок 0.8 — Пример работы программы