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»

ДИСЦИПЛИНА «Архитектура ЭВМ»

Лабораторная работа № 4

Тема Взаимодействие между серверами. Передача параметров скрипту. Дочерние процессы. Prolog.

Студентка Воякин А. Я.

Группа ИУ7-54Б

Оценка (баллы) _____

Преподаватель Попов А. Ю.

Москва.
2020 г.

Задание 1.1

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 server_1/index.js:

```
"use strict";

// Импорт библиотеки.
const express = require("express");
const fs = require("fs");

// Запуск сервера, порт 5001
const app = express();
const port = 5001;
app.listen(port);
console.log("Server on port " + port);

// Заголовки для ответа.
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();
});

// Загрузка тела для post запросов
function loadBody(request, callback) {
    let body = [];
    request.on('data', (chunk) => {
        body.push(chunk);
    }).on('end', () => {
        body = Buffer.concat(body).toString();
        callback(body);
    });
}

// Приём запроса /insert/record
app.post("/insert/record", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        let result = false;
        let data = [];
        if (fs.existsSync("cars.txt")) {
            data = fs.readFileSync("cars.txt", "utf8");
            data = JSON.parse(data);
        }
        if(data.filter(item => item.car_name === obj.car_name).length === 0)
        {
            data.push(obj);
            data = JSON.stringify(data);
            fs.writeFileSync("cars.txt", data);
            result = true;
        }
        response.end(JSON.stringify({inserted: result}));
    });
});

// Приём запроса /select/record
app.post("/select/record", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        let result;
        let exist = false;
        if (fs.existsSync("cars.txt")) {
            let data = fs.readFileSync("cars.txt", "utf8");
            data = JSON.parse(data);
            result = data.filter(item => item.car_name === obj.car_name);
            if (result.length > 0) {
                result = result[0].cost;
                exist = true;
            }
        }
          response.end(JSON.stringify({founded: exist, cost: result}));
     });
});
```

На сервере 1 реализовал проверку уникальности названий добавляемых машин.

Листинг server_2/index.js:

```
"use strict";

// Импорт библиотеки.
const express = require("express");
const fs = require("fs");

// Запуск сервера, порт 5002
const app = express();
const port = 5002;
app.listen(port);
console.log("Server on port " + port);

// Заголовки для ответа.
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();
});

// Загрузка тела для post запросов
function loadBody(request, callback) {
    let body = [];
    request.on('data', (chunk) => {
        body.push(chunk);
    }).on('end', () => {
        body = Buffer.concat(body).toString();
        callback(body);
    });
}

// Приём запроса /insert/record
app.post("/insert/record", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        let result = false;
        let data = [];
        if (fs.existsSync("storage.txt")) {
            data = fs.readFileSync("storage.txt", "utf8");
            data = JSON.parse(data);
        }
        if (data.filter(item => item.storage name === obj.storage name).length === 0) {
            data.push(obj);
            data = JSON.stringify(data);
            fs.writeFileSync("storage.txt", data);
            result = true;
        }
        response.end(JSON.stringify({inserted: result}));
    });
});

// Приём запроса /select/record
app.post("/select/record", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        let result;
        let exist = false;
        if (fs.existsSync("storage.txt")) {
            let data = fs.readFileSync("storage.txt", "utf8");
            data = JSON.parse(data);
            result = data.filter(item => item.storage_name === obj.storage_name);
            if (result.length > 0) {
                    result = result[0].cars;
                exist = true;
            }
        }
        response.end(JSON.stringify({founded: exist, cars: result}));
    });
});
```

На сервере 2 реализовал проверку уникальности названий добавляемых складов.

Листинг server_3/index.js:

```
"use strict";

// Импортируем библиотеки.
const express = require("express");
const request = require("request");

// Запуск сервера.
const app = express();
const port = 5000;
app.listen(port);
console.log(`Server on port ${port}`);

// Отправка статических файлов.
const way =   dirname + "/static";
app.use(express.static(way));

// Заголовки в ответ клиенту.
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);
    });
}

// Функция для отправки POST запросов на другие сервера.
function sendPost(url, body, callback) {
    // задаём заголовки
    const headers = {};
    headers["Cache-Control"] = "no-cache, no-store, must-revalidate";
    headers["Connection"] = "close";
    // отправляем запрос
    request.post({
        url: url,
        body: body,
        headers: headers,
    }, function (error, response, body) {
        if(error) {
            callback(null);
        } else {
            callback(body);
        }
    });
}

app.post("/new_car", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        sendPost("http://localhost:5001/insert/record", JSON.stringify({
            car_name: obj.car_name,
            cost: obj.cost
        }), function(ans_str) {
            const ans_obj = JSON.parse(ans_str);
            const inserted = ans_obj.inserted;
            if (inserted) {
                response.end(JSON.stringify({answer: "ЗАПИСЬ ДОБАВЛЕНА"}));
            }
            response.end(JSON.stringify({answer: "МАШИНА С ТАКИМ НАЗВАНИЕМ УЖЕ ДОБАВЛЕНА"}));
        });
    });
});
```

```
app.post("/car_cost", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        sendPost("http://localhost:5001/select/record", JSON.stringify({
            car_name: obj.car_name
        }), function(ans_str) {
            const ans obj = JSON.parse(ans str);
            const founded = ans obj.founded;
            if (founded) {
                response.end(JSON.stringify({answer: `СТОИМОСТЬ: ${ans_obj.cost}`}));
            }
            response.end(JSON.stringify({answer: "МАШИНА НЕ НАЙДЕНА"}));
        });
    });
});

app.post("/new_storage", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        sendPost("http://localhost:5002/insert/record", JSON.stringify({
            storage_name: obj.storage_name,
            cars: obj.cars
        }), function(ans str) {
            const ans obj = JSON.parse(ans str);
            const inserted = ans_obj.inserted;
            if (inserted) {
                response.end(JSON.stringify({answer: "ЗАПИСЬ ДОБАВЛЕНА"}));
            }
            response.end(JSON.stringify({answer: "СКЛАД С ТАКИМ НАЗВАНИЕМ УЖЕ ДОБАВЛЕН"}));
        });
    });
});

app.post("/storage_cars", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        sendPost("http://localhost:5002/select/record", JSON.stringify({
            storage_name: obj.storage_name,
        }), function(ans str) {
            const ans obj = JSON.parse(ans str);
            const founded = ans_obj.founded;
            if (founded) {
                response.end(JSON.stringify({answer: `МАШИНЫ НА СКЛАДЕ: ${ans_obj.cars}`}));
            }
            response.end(JSON.stringify({answer: "СКЛАД НЕ НАЙДЕН"}));
        });
    });
});
```

Листинг server_3/static/new_car.html:

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>CREATION</title>
    <link rel="stylesheet" href="/style.css">
</head>
<body>
<a href="http://localhost:5000/new_car.html" style="color:white">СОЗДАНИЕ НОВОГО ТИПА МАШИНЫ</a>
<br><br>
<a href="http://localhost:5000/car cost.html" style="color:white">ПОЛУЧЕНИЕ ИНФОРМАЦИИ О
СТОИМОСТИ МАШИНЫ ПО ЕЁ НАЗВАНИЮ</a><br><br>
<a href="http://localhost:5000/new_storage.html" style="color:white">СОЗДАНИЕ НОВОГО СКЛАДА С
НАХОДЯЩИМИСЯ В НЁМ МАШИНАМИ</a><br><br>
<a href="http://localhost:5000/storage_cars.html" style="color:white">ПОЛУЧЕНИЕ ИНФОРМАЦИИ О
МАШИНАХ НА СКЛАДЕ ПО НАЗВАНИЮ СКЛАДА</a>

<br><br><br><br><br>

<h1>СОЗДАНИЕ НОВОГО ТИПА МАШИНЫ</h1>

<p>
    <span>НАЗВАНИЕ МАШИНЫ</span><br>
```

```
    <input id="car_name" type="text" spellcheck="false" autocomplete="off">
</p>

<p>
    <span>СТОИМОСТЬ</span><br>
    <input id="cost" type="text" spellcheck="false" autocomplete="off">
</p>

<br>

<div id="new-car-send-btn" class="btn-class">ОТПРАВИТЬ</div>

<br>
<br>

<h1 id="result-label"></h1>

<script src="/new_car_code.js"></script>
</body>
</html>
```

Листинг server_3/static/new_car_code.js:

```
"use strict";

window.onload = function() {
    const car_name_in = document.getElementById("car_name");
    const cost_in = document.getElementById("cost");

    const btn = document.getElementById("new-car-send-btn");

    const label = document.getElementById("result-label");

    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);
        }
    }

    btn.onclick = function() {
        const car_name = car_name_in.value;
        const cost = cost in.value;

        ajaxPost("/new_car", JSON.stringify({
            car_name, cost
        }), function(answerString) {
            const objectAnswer = JSON.parse(answerString);
            label.innerHTML = objectAnswer.answer;
        });
    };

    car_name_in.onkeydown = car_name_in.onkeypress = car_name_in.onkeyup = function () {
        label.innerHTML = "";
    }

    cost_in.onkeydown = cost_in.onkeypress = cost_in.onkeyup = function () {
        label.innerHTML = "";
    }

};
```

Листинг server_3/static/car_cost.html:

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
```

```
    <title>CAR PRICE</title>
    <link rel="stylesheet" href="/style.css">
</head>
<body>
<a href="http://localhost:5000/new_car.html" style="color:white">СОЗДАНИЕ НОВОГО ТИПА МАШИНЫ</a>
<br><br>
<a href="http://localhost:5000/car_cost.html" style="color:white">ПОЛУЧЕНИЕ ИНФОРМАЦИИ О
СТОИМОСТИ МАШИНЫ ПО ЕЁ НАЗВАНИЮ</a><br><br>
<a href="http://localhost:5000/new storage.html" style="color:white">СОЗДАНИЕ НОВОГО СКЛАДА С
НАХОДЯЩИМИСЯ В НЁМ МАШИНАМИ</a><br><br>
<a href="http://localhost:5000/storage_cars.html" style="color:white">ПОЛУЧЕНИЕ ИНФОРМАЦИИ О
МАШИНАХ НА СКЛАДЕ ПО НАЗВАНИЮ СКЛАДА</a>

<br><br><br><br><br>

<h1>ПОЛУЧЕНИЕ ИНФОРМАЦИИ О СТОИМОСТИ МАШИНЫ ПО ЕЁ НАЗВАНИЮ</h1>

<br>
<p>
    <label>
        <span>НАЗВАНИЕ МАШИНЫ</span><br>
        <input id="car_name" type="text" spellcheck="false" autocomplete="off">
    </label>
</p>

<br>

<div id="car-cost-send-btn" class="btn-class">ОТПРАВИТЬ</div>

<br><br>

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

<script src="/car_cost_code.js"></script>
</body>
</html>
```

Листинг server_3/static/car_cost_code.js:

```
"use strict";

window.onload = function() {
    const car_name_in = document.getElementById("car_name");

    const btn = document.getElementById("car-cost-send-btn");

    const label = document.getElementById("result-label");

    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);
        }
    }

    btn.onclick = function() {
        const car_name = car_name_in.value;
        ajaxPost("/car_cost", JSON.stringify({
            car_name
        }), function(answerString) {
            const objectAnswer = JSON.parse(answerString);
            label.innerHTML = objectAnswer.answer;
        });
    };

    car_name_in.onkeydown = car_name_in.onkeypress = car_name_in.onkeyup = function () {
        label.innerHTML = "";
    }
};
```

Листинг server_3/static/new_storage.html:

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>NEW STORAGE</title>
    <link rel="stylesheet" href="/style.css">
</head>
<body>
<a href="http://localhost:5000/new_car.html" style="color:white">СОЗДАНИЕ НОВОГО ТИПА МАШИНЫ</a>
<br><br>
<a href="http://localhost:5000/car cost.html" style="color:white">ПОЛУЧЕНИЕ ИНФОРМАЦИИ О
СТОИМОСТИ МАШИНЫ ПО ЕЁ НАЗВАНИЮ</a><br><br>
<a href="http://localhost:5000/new_storage.html" style="color:white">СОЗДАНИЕ НОВОГО СКЛАДА С
НАХОДЯЩИМИСЯ В НЁМ МАШИНАМИ</a><br><br>
<a href="http://localhost:5000/storage cars.html" style="color:white">ПОЛУЧЕНИЕ ИНФОРМАЦИИ О
МАШИНАХ НА СКЛАДЕ ПО НАЗВАНИЮ СКЛАДА</a>

<br><br><br><br><br>

<h1>СОЗДАНИЕ НОВОГО СКЛАДА С НАХОДЯЩИМИСЯ В НЁМ МАШИНАМИ</h1>

<p>
    <span>НАЗВАНИЕ СКЛАДА</span><br>
    <input id="storage_name" type="text" spellcheck="false" autocomplete="off">
</p>

<p>
    <span>НАЗВАНИЯ МАШИН</span><br>
    <input id="cars" type="text" spellcheck="false" autocomplete="off">
</p>

<br>

<div id="new-storage-send-btn" class="btn-class">ОТПРАВИТЬ</div>

<br>
<br>

<h1 id="result-label"></h1>

<script src="/new_storage_code.js"></script>
</body>
</html>
```

Листинг server_3/static/new_storage_code.js:

```
"use strict";

window.onload = function() {
    const storage_name_in = document.getElementById("storage_name");
    const cars_in = document.getElementById("cars");

    const btn = document.getElementById("new-storage-send-btn");

    const label = document.getElementById("result-label");

    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);
        }
    }

    btn.onclick = function() {
        const storage name = storage name in.value;
        let cars = cars in.value;
        cars = cars.split(' ');

        ajaxPost("/new_storage", JSON.stringify({
```

```
            storage_name, cars
        }), function(answerString) {
            const objectAnswer = JSON.parse(answerString);
            label.innerHTML = objectAnswer.answer;
        });
    };

    storage name in.onkeydown = storage name in.onkeypress = storage name in.onkeyup = function
() {
        label.innerHTML = "";
    }

    cars in.onkeydown = cars in.onkeypress = cars in.onkeyup = function () {
        label.innerHTML = "";
    }

};
```

Листинг server_3/static/storage_cars.html:

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>CARS INFO</title>
    <link rel="stylesheet" href="/style.css">
</head>
<body>
<a href="http://localhost:5000/new_car.html" style="color:white">СОЗДАНИЕ НОВОГО ТИПА МАШИНЫ</a>
<br><br>
<a href="http://localhost:5000/car cost.html" style="color:white">ПОЛУЧЕНИЕ ИНФОРМАЦИИ О
СТОИМОСТИ МАШИНЫ ПО ЕЁ НАЗВАНИЮ</a><br><br>
<a href="http://localhost:5000/new_storage.html" style="color:white">СОЗДАНИЕ НОВОГО СКЛАДА С
НАХОДЯЩИМИСЯ В НЁМ МАШИНАМИ</a><br><br>
<a href="http://localhost:5000/storage cars.html" style="color:white">ПОЛУЧЕНИЕ ИНФОРМАЦИИ О
МАШИНАХ НА СКЛАДЕ ПО НАЗВАНИЮ СКЛАДА</a>

<br><br><br><br><br>

<h1>ПОЛУЧЕНИЕ ИНФОРМАЦИИ О МАШИНАХ НА СКЛАДЕ ПО НАЗВАНИЮ СКЛАДА</h1>

<br>
<p>
    <label>
        <span>НАЗВАНИЕ СКЛАДА</span><br>
        <input id="storage_name" type="text" spellcheck="false" autocomplete="off">
    </label>
</p>

<br>

<div id="cars-info-send-btn" class="btn-class">ОТПРАВИТЬ</div>

<br><br>

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

<script src="/storage_cars_code.js"></script>
</body>
</html>
```

Листинг server_3/static/storage_cars_code.js:

```
"use strict";

window.onload = function() {
    const storage name in = document.getElementById("storage name");

    const btn = document.getElementById("cars-info-send-btn");

    const label = document.getElementById("result-label");

    function ajaxPost(urlString, bodyString, callback) {
        let r = new XMLHttpRequest();
        r.open("POST", urlString, true);
```

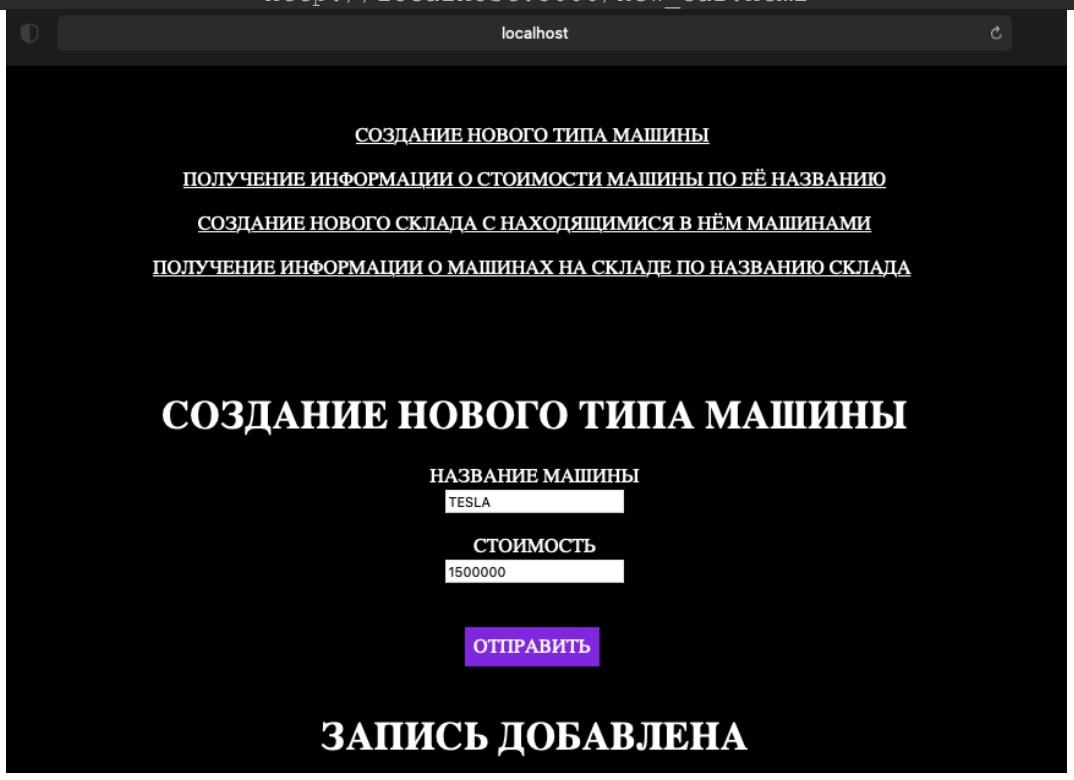
```
        r.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
        r.send(bodyString);
        r.onload = function() {
            callback(r.response);
        }
    }

    btn.onclick = function() {
        const storage name = storage name in.value;
        ajaxPost("/storage_cars", JSON.stringify({
            storage_name
        }), function(answerString) {
            const objectAnswer = JSON.parse(answerString);
            label.innerHTML = objectAnswer.answer;
        });
    };

    storage_name_in.onkeydown = storage_name_in.onkeypress = storage_name_in.onkeyup = function
() {
        label.innerHTML = "";
    }
};
```

Листинг server_3/static/style.css:

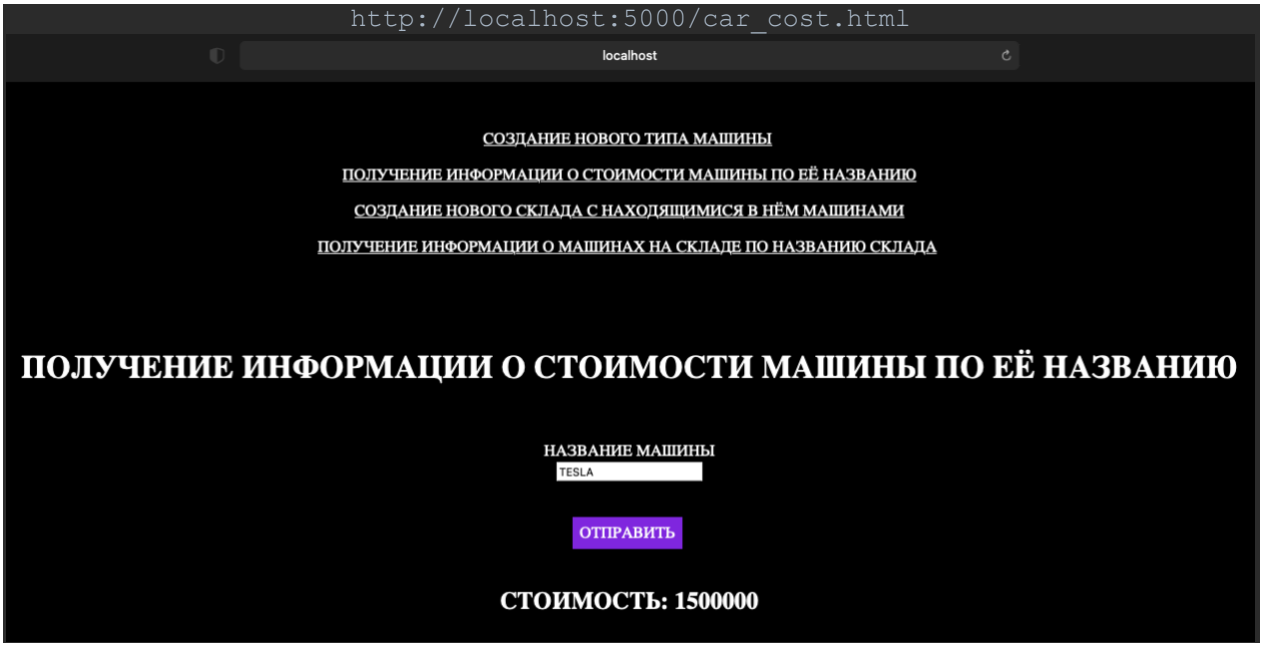
```
body {
    padding: 40px;
    background-color: #000;
    color: #fff;
    text-align: center;
}

.btn-class {
    padding: 7px;
    background: blueviolet;
    color: white;
    cursor: pointer;
    display: inline-block;
}
```

Демонстрация добавления нового типа машины:

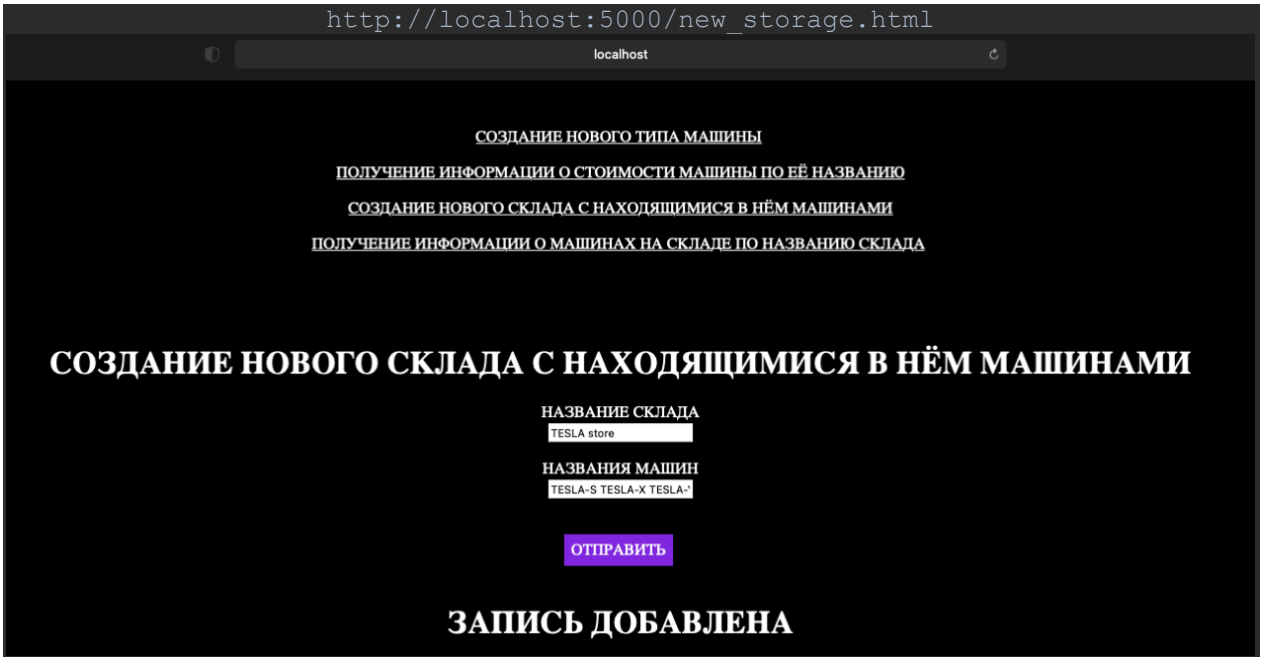
http://localhost:5000/new_car.html



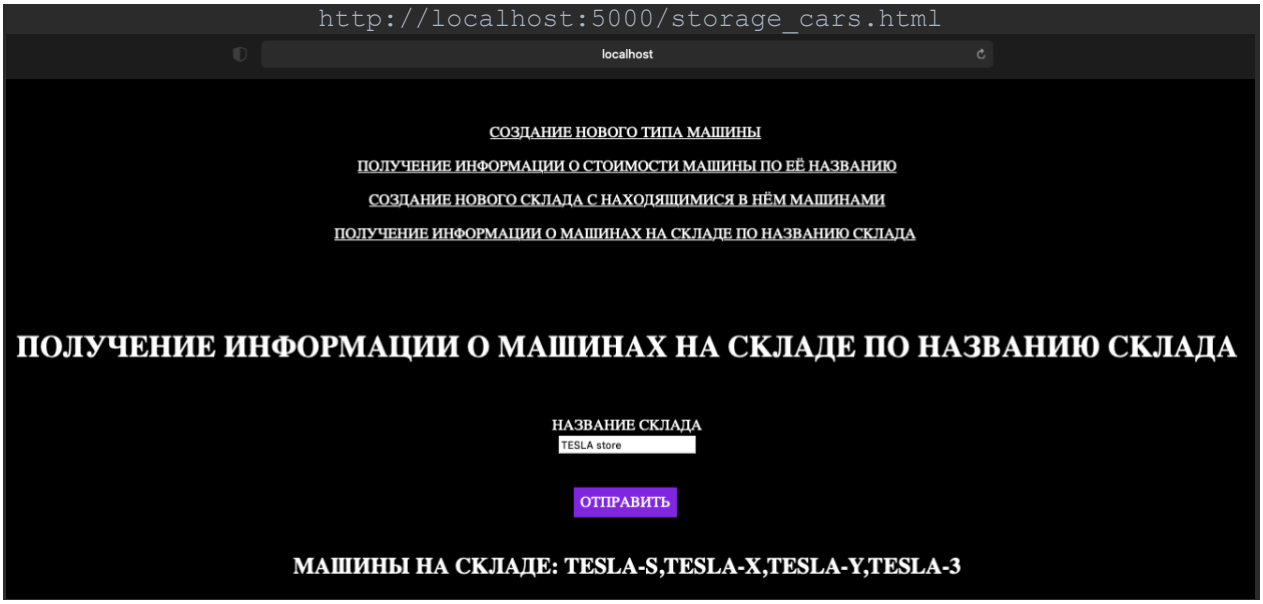
Получение информации о стоимости машины по её типу:



Создание нового склада с находящимися в нём машинами:



Получение информации о машинах на складе по названию склада:



Задание 1.2

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 fact_num.js:

```
"use strict";

function factorial(num) {
    if (num === 1 || num === 0) return 1;
    return num * factorial(num - 1);
}

if (process.argv.length > 2) {
    const num = parseInt(process.argv[2]);
    if (num >= 0) console.log(factorial(num));
    else console.log(undefined);
}
```

Листинг index_num.js:

```
"use strict";

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

// Получение параметров скрипта.
const val = process.argv[2];
if (!val) return;

function cmd(str) {
    const options = {encoding: 'utf8'};
    const command = str.toString();
    const answer = execSync(command, options);
    return answer.toString();
}

// Получение факториала числа
let result = cmd(`node fact_num.js ${val}`);
console.log(result);
```

Демонстрация работы программы:

```
~/Documents/NodeJS/task_7/2  master ±  node index_num.js 3
6
```

Листинг fact_arr.js:

```
"use strict";

function factorial(num) {
    if (num === 1 || num === 0) return 1;
    return num * factorial(num - 1);
}

if (process.argv.length > 2) {
    let arr = JSON.parse(process.argv[2]);
    for (let i = 0; i < arr.length; i++) {
        if (arr[i] < 0) {
            arr[i] = undefined;
            continue;
        }
        arr[i] = factorial(parseInt(arr[i]));
    }
    console.log(arr);
}
```

Листинг index.arr.js:

```
"use strict";

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

let arr = []

// Получение параметров скрипта.
for (let i = 2; i < process.argv.length; i++) {
    arr.push(parseInt(process.argv[i]));
}

function cmd(str) {
    const options = {encoding: 'utf8'};
    const command = str.toString();
    const answer = execSync(command, options);
    return answer.toString();
}

let result = (cmd(`node fact_arr.js ${JSON.stringify(arr)}`));
console.log(result);
```

Демонстрация работы программы:

```
~/Documents/NodeJS/task_7/2  master ±  node index_arr.js 3 4 5 6
[ 6, 24, 120, 720 ]
```

Задание 2

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

Листинг программы:

```
writeNumber(X) :- write(X).

fib(F1, F2, L, R) :-
    F1 =< R,
    (
        (F1 >= L, writeNumber(F1), nl, fail);
        (F3 is F1 + F2, fib(F2, F3, L, R))
    ).

start(L, R) :- L =< R, fib(0, 1, L, R).
```

Демонстрация работы программы:

```
?- start(1, 30).
1
1
2
3
5
8
13
21
```

Вывод: в ходе данной лабораторной работы познакомился с взаимодействием между серверами и реализовал сервер для отправки запросов на другие сервера. Научился передавать параметры программам через аргументы командной строки. Научился работать с дочерними процессами. Познакомился с языком Prolog и написал на нём простые программы.