Отчет по разделу #7

Цель работы

Ознакомиться с взаимодейтсвием серверов в Node.js.

Задание 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).

Код программы

Язык: **Javascript**

Сервер А

src/app.js

```
const express = require("express");
const body_parser = require("body-parser");

const router = require("./router");
const logger = require("./controllers").logger;

const port = process.env.PORT | 4001;

const app = express();

app.use(body_parser.json());
app.use(logger);

app.use(router);

app.listen(port);
console.log(`Сервер запущен. Порт: ${port}`);
```

src/controllers.js

```
"use strict";

const utils = require("./utils");

module.exports.insert_record = (req, res) => {
    try {
        const { name, price } = req.body;
        if (!name || !price)
            throw Error("");

        utils.put({ name, price });
        res.status(200).send(JSON.stringify({ result: "Ok" }));
    } catch(_) {
        res.status(400).send(JSON.stringify({ result: "No object in body" }));
    }
}

module.exports.select_record = (req, res) => {
    const car = utils.get(req.query.name);
    if (car)
        res.status(200).send(JSON.stringify({ result: "Ok", car }))
    else
        res.status(400).send(JSON.stringify({ result: `No car with name ${req.query.name}` }));
}


module.exports.bad_request_controller = (req, res) => {
    res.status(400).send(JSON.stringify({ result: `Unknown request: ${req.method} ${req.url}` }));
}

module.exports.logger = (req, _, next) => {
    console.log(`${req.url} ${req.method}`);

    if (req.method === "GET") {
        console.log(req.query);
    } else if (req.method === "POST") {
        console.log(req.body);
    }

    next();
}
```

src/router.js

```
const Router = require("express").Router;

const controllers = require("./controllers");

const router = Router();

router.post("/insert/record", controllers.insert_record);
router.get("/select/record", controllers.select_record);
router.use(controllers.bad_request_controller);

module.exports = router;
```

src/utils.js

```
"use strict";

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

const DB_FILE_FOLDER = path.join(__dirname, "..", "meta");
const DB_FILE = path.join(DB_FILE_FOLDER, "db.txt");


if (!fs.existsSync(DB_FILE_FOLDER)) {
    fs.mkdirSync(DB_FILE_FOLDER);
}

if (!fs.existsSync(DB_FILE)) {
    fs.writeFileSync(DB_FILE, JSON.stringify([]));
}

const db = JSON.parse(fs.readFileSync(DB_FILE));

console.log(db);

module.exports.get = (name) => {
    return db.find(car => car.name === name);
}

module.exports.put = (car) => {
    db.push(car);
    fs.writeFileSync(DB_FILE, JSON.stringify(db));
}
```

Сервер Б

src/app.js

```
const express = require("express");
const body_parser = require("body-parser");

const router = require("./router");
const logger = require("./controllers").logger;

const port = process.env.PORT | 4002;

const app = express();

app.use(body_parser.json());
app.use(logger);

app.use(router);

app.listen(port);
console.log(`Сервер запущен. Порт: ${port}`);
```

src/controllers.js

```
"use strict";

const utils = require("./utils");

module.exports.insert_record = (req, res) => {
    try {
        const { name, cars } = req.body;
        if (!name || !cars)
            throw Error("");
        utils.put({ name, cars });

        res.status(200).send(JSON.stringify({ result: "Ok" }));
    } catch(_) {
        res.status(400).send(JSON.stringify({ result: "No object in body" }));
    }
}

module.exports.select_record = (req, res) => {
    const stock = utils.get(req.query.name);
    if (stock)
        res.status(200).send(JSON.stringify({ result: "Ok", stock }))
    else
        res.status(400).send(JSON.stringify({ result: `No stock with name ${req.query.name}` }));
}


module.exports.bad_request_controller = (req, res) => {
    res.status(400).send(JSON.stringify({ result: `Unknown request: ${req.method} ${req.url}` }));
}

module.exports.logger = (req, _, next) => {
    console.log(`${req.url} ${req.method}`);

    if (req.method === "GET") {
        console.log(req.query);
    } else if (req.method === "POST") {
        console.log(req.body);
    }

    next();
}
```

src/router.js

```
const Router = require("express").Router;

const controllers = require("./controllers");

const router = Router();

router.post("/insert/record", controllers.insert_record);
router.get("/select/record", controllers.select_record);
router.use(controllers.bad_request_controller);

module.exports = router;
```

src/utils.js

```
"use strict";

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

const DB_FILE_FOLDER = path.join(__dirname, "..", "meta");
const DB_FILE = path.join(DB_FILE_FOLDER, "db.txt");

if (!fs.existsSync(DB_FILE_FOLDER))
    fs.mkdirSync(DB_FILE_FOLDER);

if (!fs.existsSync(DB_FILE))
    fs.writeFileSync(DB_FILE, JSON.stringify([]));

const db = JSON.parse(fs.readFileSync(DB_FILE));

module.exports.get = (name) => {
    return db.find(stock => stock.name === name);
}

module.exports.put = (stock) => {
    db.push(stock);
    fs.writeFileSync(DB_FILE, JSON.stringify(db));
}
```

Сервер С

src/app.js

```
const express = require("express");
const body_parser = require("body-parser");
const path = require("path");

const router = require("./router");
const logger = require("./controllers").logger;

const port = process.env.PORT | 3000;
const VIEWS_FOLDER = path.join(__dirname, "..", "views");
const PUBLIC_FOLDER = path.join(__dirname, "..", "public");

const app = express();

app.set("view engine", "ejs");
app.set("views", VIEWS_FOLDER)

app.use(body_parser.urlencoded({ extended: false }));
app.use(express.static(PUBLIC_FOLDER))
app.use(logger);

app.use(router);

app.listen(port);
console.log(`Сервер запущен. Порт: ${port}`);
```

src/controllers.js

```
"use strict";

const axios = require("axios");

const CARS_SERVER = "http://localhost:4001/";
const INSERT_QUERY = "insert/record";
const SELECT_QUERY = "select/record";
const STOCKS_SERVER = "http://localhost:4002/";
```

```
const HOME_PAGE = "index.ejs";
const CREATE_CAR_PAGE = "create_car.ejs";
const CREATE_STOCK_PAGE = "create_stock.ejs";
const SEARCH_CAR_PAGE = "search_car.ejs";
const SEARCH_STOCK_PAGE = "search_stock.ejs";
const CAR_PAGE = "car.ejs";
const STOCK_PAGE = "stock.ejs";

module.exports.render_create_car = (_, res) => {
    res.render(CREATE_CAR_PAGE, { meta: {} });
}

module.exports.create_car = (req, res) => {
    axios.post(CARS_SERVER + INSERT_QUERY, {
        name: req.body.name,
        price: req.body.price
    }).then(result => {
        if (result.status !== 200)
            throw Error(result.data.result);

        res.status(200).render(CREATE_CAR_PAGE, { meta: { log: "Машина была зарегистрированна" } });
    }).catch(err => {
        res.status(400).render(CREATE_CAR_PAGE, { meta: { error: `Невозможно зарегистрировать машину: ${err}!` } });
    });
}

module.exports.search_car = (_, res) => {
    res.render(SEARCH_CAR_PAGE);
}

module.exports.show_car = (req, res) => {
    axios.get(CARS_SERVER + SELECT_QUERY, {
        params: {
            name: req.query.name
        }
    }).then(result => {
        res.status(200).render(CAR_PAGE, { car: result.data.car, error: null });
    }).catch(() => {
        res.status(400).render(CAR_PAGE, { car: {}, error: "Машина не найдена" });
    })
}

module.exports.render_create_stock = (_, res) => {
    res.render(CREATE_STOCK_PAGE, { meta: {} });
}

module.exports.create_stock = (req, res) => {
    axios.post(STOCKS_SERVER + INSERT_QUERY, {
        name: req.body.name,
        cars: req.body.cars.split(' ').filter(str => str !== '')
    }).then(result => {
        console.log(`RESULT: ${result}`)
        res.status(200).render(CREATE_STOCK_PAGE, { meta: { log: "Склад был зарегистрирован" } });
    }).catch(err => {
        res.status(400).render(CREATE_STOCK_PAGE, { meta: { error: `Невозможно зарегистрировать склад: ${err}` } });
    });
}

module.exports.search_stock = (_, res) => {
    res.render(SEARCH_STOCK_PAGE);
}

module.exports.show_stock = (req, res) => {
    axios.get(STOCKS_SERVER + SELECT_QUERY, {
        params: {
            name: req.query.name
```

```
        }
    }).then(result => {
        console.log(result.data.stock);
        res.status(200).render(STOCK_PAGE, { stock: result.data.stock, error: null });
    }).catch(() => {
        res.status(400).render(STOCK_PAGE, { stock: {}, error: "Склад не найден" });
    })
}

module.exports.default_controller = (_, res) => {
    res.render(HOME_PAGE);
}

module.exports.logger = (req, _, next) => {
    console.log(`${req.url} ${req.method}`);

    if (req.method === "GET") {
        console.log(req.query);
    } else if (req.method === "POST") {
        console.log(req.body);
    }

    next();
}
```

src/router.js

```
const Router = require("express").Router;

const controllers = require("./controllers");

const router = Router();

router.get("/create-car", controllers.render_create_car);
router.post("/create-car", controllers.create_car);

router.get("/search-car", controllers.search_car);
router.get("/car", controllers.show_car);

router.get("/create-stock", controllers.render_create_stock);
router.post("/create-stock", controllers.create_stock);

router.get("/search-stock", controllers.search_stock);
router.get("/stock", controllers.show_stock);

router.post("/create-car", controllers.create_car);
router.use(controllers.default_controller);

module.exports = router;
```

Задание 2

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.

Код программы

Язык: **Javascript**

main.js

```
"use strict";

const exec = require("child_process").execSync;

const fact = (num) => {
    const value = exec(`node fact.js ${num}`);
    return parseInt(value);
}

const result = process.argv.slice(2, process.argc).map(num => parseInt(num));
result.map(num => console.log(`${num}! = ${fact(num)}`))
```

fact.js

```
"use strict";

const fact = (num) => {
    return Array.from(Array(parseInt(num)).keys()).reduce((acc, i) => acc * (i + 1), 1)
}

console.log(fact(process.argv[2]));
```

Результаты тестирования

Все тесты были пройдены успешно.

Вывод

В результате работы было освоено взаимодействие между серверами.

Отчет по разделу #8

Цель работы

Ознакомиться с особенностями и базовыми принципами программирования на Prolog.

Задание

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

Код программы

Язык: **Prolog**

is_fib.js

```
ok.
isFib(0, 1, 0, B) :-
    write(0), nl,
    isFib(1, 1, 0, B).
isFib(F1, F2, A, B) :-
    F3 is (F1 + F2),
    F2 < A,
    isFib(F2, F3, A, B).
isFib(F1, F2, A, B) :-
    F3 is (F1 + F2),
    F2 >= A,
    F2 =< B,
    write(F2), nl,
    isFib(F2, F3, A, B);
ok.
input(A, B) :- read(A), read(B); ok.
f :- input(A, B), isFib(0, 1, A, B); ok.
```

Результаты тестирования

Все тесты были пройдены успешно.

Вывод

В результате работы были изучены базовые приемы работы с Prolog, освоены особенности работы с данным языком программирования.