**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Разработка интернет приложений»

Отчет по лабораторной работе №7

«Разработка пользовательского интерфейса для работы с REST API»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-52Б |  | преподаватель каф. ИУ5 |
| Барышников Михаил |  | Гапанюк Ю.Е. |
|  |  |  |

Москва, 2021 г.

**Описание задания:**

На основе методических указаний разработайте пользовательский интерфейс для работы с REST API. Используйте REST API, разработанный Вами в предыдущей лабораторной работе.

**Файл server/index.js:**

'use strict';

const express = require('express');

const path = require('path');

const app = express();

const htmlPath = path.resolve(\_\_dirname, '..', 'src/index.html')

const staticPath = path.resolve(\_\_dirname, '..', 'src')

const port = 3000;

app.use('/', express.static(staticPath));

app.all('\*', (req, res) => {

res.sendFile(htmlPath);

});

app.listen(port, () => {

console.log(`Server listening http://localhost:${port}`);

});

**Файл src/components/back-button/BackButtonComponent.js:**

export class BackButtonComponent {

    constructor(parent) {

        this.parent = parent

    }

    getHTML() {

        return '<button id="back-button" class="btn btn-primary" type="button">Назад</button>'

    }

    addListeners(listener) {

        document.getElementById("back-button").addEventListener('click', listener)

    }

    render(listener) {

        const html = this.getHTML()

        this.parent.insertAdjacentHTML('beforeend', html)

        this.addListeners(listener)

    }

}

**Файл src/components/cpu-card/CpuCardComponents.js:**

export class CpuCardComponent {

constructor(parent) {

this.parent = parent;

}

getHTML(data) {

return (

`

<div class="card" style="width: 300px;">

<div class="card-body">

<h5 class="card-title">${data.name}</h5>

<!--<p class="card-text">${data.frequency}</p>architecture-->

<!--<p class="card-text">${data.architecture}</p>-->

<button class="btn btn-primary" id="click-card-${data.id}" data-id="${data.id}">Подробнее</button>

</div>

</div>

`

)

}

addListeners(data, listener) {

document

.getElementById(`click-card-${data.id}`)

.addEventListener("click", listener)

}

render(data, listener) {

const html = this.getHTML(data)

this.parent.insertAdjacentHTML('beforeend', html)

this.addListeners(data, listener)

}

}

**Файл src/components/Cpu/CpuComponent.js:**

export class CpuComponent {

constructor(parent) {

this.parent = parent

}

getHTML(data) {

return (

`

<div class='card text-white bg-dark' style='width: 300px'>

<div class="card-header">

Процессор

</div>

<div class='card-body'>

<h4 class='card-title'>${data.name}</h4>

<p class='card-text'>Рабочая частота: ${data.frequency}</p>

<p class='card-text'>Архитектура: ${data.architecture}</p>

</div>

</div>

`

)

}

render(data) {

const html = this.getHTML(data)

this.parent.insertAdjacentHTML('beforeend', html)

}

}

**Файл src/modules/ajax.js:**

class Ajax {

async get(url) {

const response = await fetch(url);

const responseData = await response.json();

return {status: response.status, data: responseData};

}

}

export const ajax = new Ajax();

**Файл src/modules/urls.js:**

class Urls {

constructor() {

this.url = 'http://localhost:8000/api/';

}

cpus() {

return `${this.url}computercomp/`

}

cpu(id) {

return `${this.url}computercomp/${id}/`

}

}

export const urls = new Urls()

**Файл src/pages/main/MainPage.js:**

import {CpuCardComponent} from "../../components/cpu-card/CpuCardComponents.js";

import {CpuPage} from "../../Cpu/CpuPage.js";

import { urls } from "../../modules/urls.js";

import { ajax } from "../../modules/ajax.js";

export class MainPage {

constructor(parent) {

this.parent = parent;

}

async getData() {

return ajax.get(urls.cpus())

}

get page() {

return document.getElementById('main-page')

}

getHTML() {

return (

`

<div id="main-page" class="d-flex flex-wrap"><div/>

`

)

}

clickCard(e) {

const cardId = e.target.dataset.id

const cpuPage = new CpuPage(this.parent, cardId)

cpuPage.render()

}

async render() {

this.parent.innerHTML = ''

const html = this.getHTML()

this.parent.insertAdjacentHTML('beforeend', html)

const data = await this.getData()

data.data.forEach((item) => {

const CpuCard = new CpuCardComponent(this.page)

CpuCard.render(item, this.clickCard.bind(this))

})

}

}

**Файл src/Cpu/CpuPage.js:**

import {CpuComponent} from "../components/Cpu/CpuComponent.js";

import {BackButtonComponent} from "../components/back-button/BackButtonComponent.js";

import {MainPage} from "../pages/main/MainPage.js";

import { urls } from "../modules/urls.js";

import { ajax } from "../modules/ajax.js";

export class CpuPage {

constructor(parent, id) {

this.parent = parent

this.id = id

}

async getData() {

return ajax.get(urls.cpu(this.id))

}

get page() {

return document.getElementById('stock-page')

}

getHTML() {

return (

`

<div id="stock-page">

</div>

`

)

}

clickBack() {

const mainPage = new MainPage(this.parent)

mainPage.render()

}

async render() {

this.parent.innerHTML = ''

const html = this.getHTML()

this.parent.insertAdjacentHTML('beforeend', html)

const backButton = new BackButtonComponent(this.page)

backButton.render(this.clickBack.bind(this))

const data = await this.getData()

const cpu = new CpuComponent(this.page)

cpu.render(data.data)

}

}

**Файл src/index.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Lab7</title>

    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css">

</head>

<body>

    <div id="root"></div>

    <script src="main.js" type="module"></script>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>

**Файл src/main.js:**

import {MainPage} from "./pages/main/MainPage.js";

const root = document.getElementById('root');

const mainPage = new MainPage(root)

mainPage.render()

**Пользовательский интерфейс:**





