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 Системы обработки информации и управления

Лабораторная работа №7

По курсу «Разработка интернет приложений»

Подготовил:

Студент группы ИУ5-55Б.

Турчин Д.С.

18.12.2020

Проверил:

Преподаватель кафедры ИУ5
Гапанюк Ю.Е.

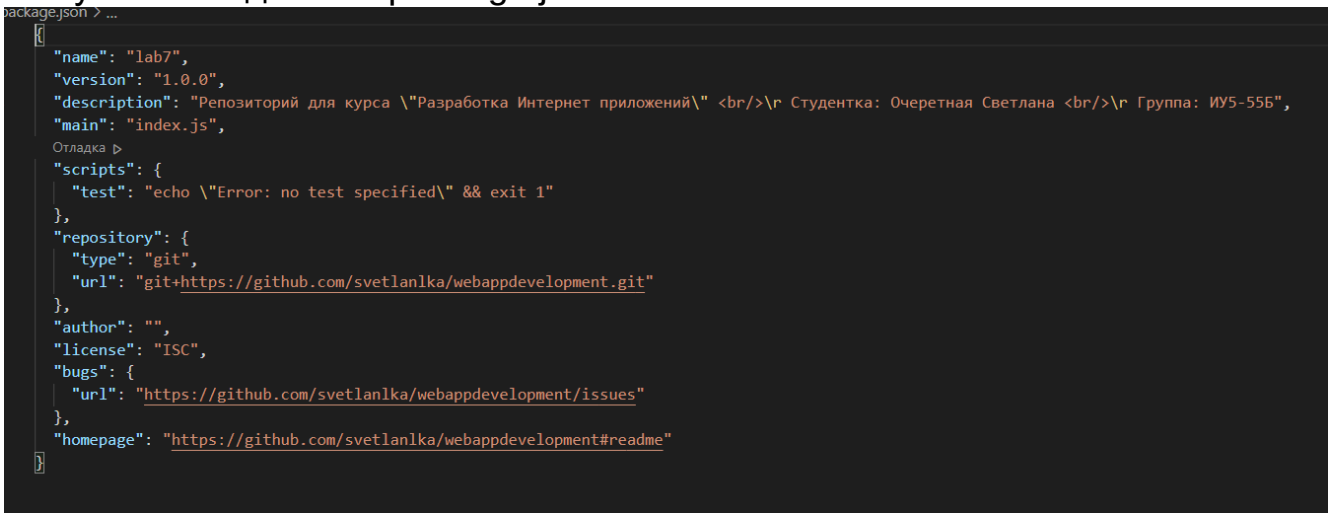
Москва, 2021 г.

Цель лабораторной работы: изучение возможностей создания пользовательского интерфейса в веб-приложениях.

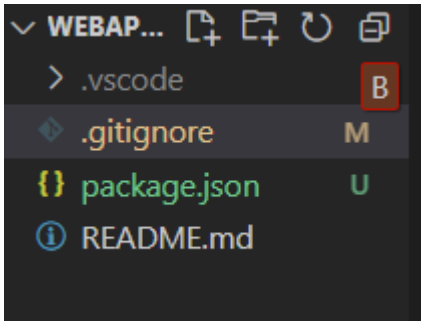
Задание: разработать пользовательский интерфейс для работы с REST API. Использовать REST API, разработанный в 6 лабораторной работе.

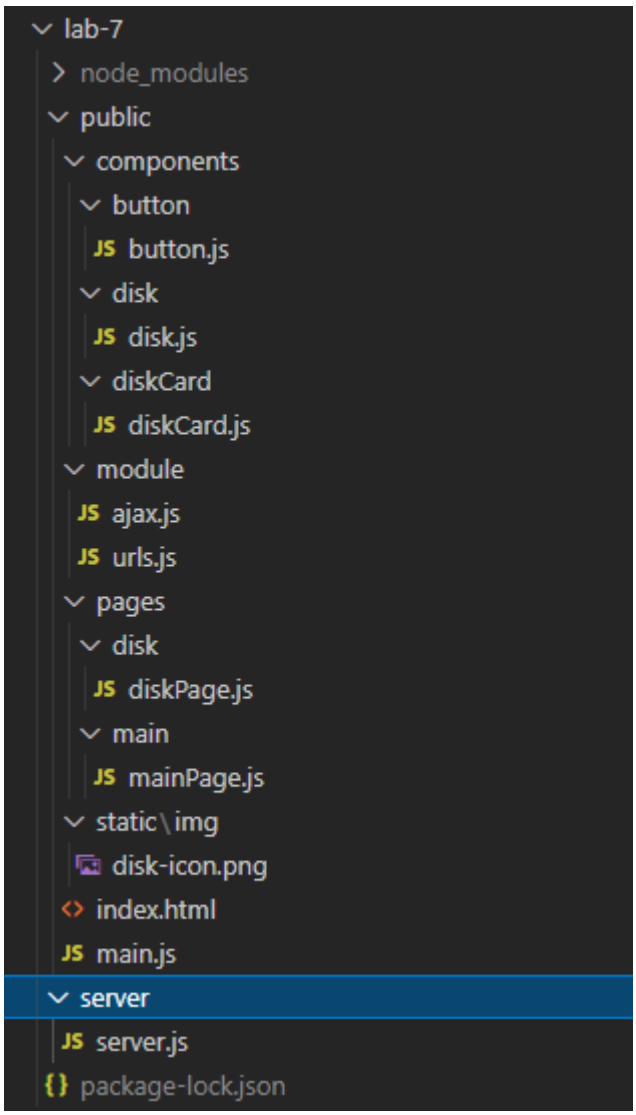
Ход работы

Создадим проект прописав npm init
Получаем созданный package.json



Добавим gitignore





Создаем файл index.html

```
lab-7 > public > index.html > ...
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Лабка-7 РИП</title>
8     <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css">
9   </head>
10  <body>
11    <div id="root"></div>
12    <script src="main.js" type="module"></script>
13    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/js/bootstrap.bundle.min.js"></script>
14  </body>
15  </html>
```

Чтобы отдавать этот файл, напишем свой сервер, для этого скачаем express через команду npm install express --save-dev
У нас добавился пакет в package.json и добавилась папка node_modules, где хранится наш пакет. Мы установили express, теперь нам нужно настроить наш сервер. Для этого создадим server/server.js. Напишу

следующий сервер статики с поддержкой строгого режима, который по запросу / или /index.html отдает базовый html, а по другим запросам отдает файл, лежащий по соответствующему пути в папке public/

lab-7 > server > JS server.js > ...

```
1   'use strict';
2
3   const express = require('express');
4   const path = require('path');
5   const staticPath = path.resolve(__dirname, '..', 'src');
6
7   const app = express();
8   const ip = 'http://localhost';
9   const port = 8080;
10
11  app.use('/', express.static(staticPath));
12
13  app.all('*', (req, res) => {
14    const path_ = req.path === '/' ? 'index.html' : req.path
15    res.contentType(path.basename(path_));
16    console.log(req.path)
17    res.sendFile(path.resolve(__dirname, '..', 'public/' + path_));
18  });
19
20  app.listen(port, () => {
21    console.log(`listening at ${ip}:${port}`);
22  });
23
```

Для запуска сервера используем команду node server/server.js. Теперь если мы перейдем на по урлу http://localhost:8080, то сможем увидеть наш сайт
Добавим bootstrap для верстки

Страницы на JS

lab-7 > public > pages > disk > JS diskPage.js > ...

```
import DiskComponent from '../../components/disk/disk.js';
import ButtonComponent from '../../components/button/button.js';
import MainPage from '../../pages/main/mainPage.js';
import urls from '../../module/urls.js';
import ajax from '../../module/ajax.js';

export default class DiskPage {
  constructor(root, id) {
    this.root = root;
    this.id = id;
  }

  getData() {
    // return {
    //   id: 1,
    //   src: "https://i.pinimg.com/originals/c9/ea/65/c9ea654eb3a7398b1f702c758c1c4206.jpg",
    //   title: "Акция",
    //   text: "Такой акции вы еще не видели"
    // }
    return ajax.get({url: urls.disk(this.id)});
  }

  get page() {
    return document.getElementById('disk-page');
  }

  getHTML() {
    return (
      `
        <div id="disk-page">
        </div>
      `
    );
  }

  render() {
      this.root.innerHTML = '';
      const html = this.getHTML();
      this.root.insertAdjacentHTML('beforeend', html);

      const disk = new DiskComponent(this.page);
      this.getData().then(({response}) => {
        disk.render(response);

        const backBtn = new ButtonComponent(this.page);
        backBtn.render({sign: 'Назад'}, (e) => {
          const mainPage = new MainPage(this.root);
          mainPage.render();
        });
      });

  }
}
```

lab-7 > public > pages > main > JS mainPage.js > ...

```js
import DiskCardComponent from '../../components/DiskCard/DiskCard.js';
import DiskPage from '../../pages/disk/diskPage.js';
import urls from '../../module/urls.js';
import ajax from '../../module/ajax.js';

export default class MainPage {
  constructor(root) {
    this.root = root;
  }

  get page() {
    return document.getElementById('main-page')
  }

  getData() {
    return ajax.get({url: urls.disks()});
  }

  render() {
    this.root.innerHTML = '';
    this.root.insertAdjacentHTML('beforeend', `
      <div id="main-page" class="d-flex flex-wrap"><div/>
    `);

    this.getData().then(({response}) => {
      response.forEach(element => {
        const diskCard = new DiskCardComponent(this.page)
        diskCard.render(element, (e) => {
          const cardId = e.currentTarget.dataset.id;
          console.log(cardId);
          const diskPage = new DiskPage(this.root, cardId)
          diskPage.render();
        });
      });
    });
  }
}

```

Модуль для работы с сетью

Урлы апи аккуратно вынесены в отдельный файлик

Сам ajax-модуль, основанный на fetch-API. Преимущество такого подхода в том, что можно безболезненно и относительно просто написать, какие действия надо выполнить по выполнению запроса. Логирование запросов позволит контролировать процесс дальнейшей разработки АПИ.

lab-7 > public > module > JS ajax.js > ajaxCall > then() callback

```
import urls from './urls.js';

const APIurl = urls.url;
const ajaxDebug = true;

/**
 * Поддерживаемые методы: GET и POST
 * @typedef {Object} ajaxMethod
 * @property {string} post
 * @property {string} get
 */
const ajaxMethods = {
  post: 'POST',
  get: 'GET',
};

/**
 * Поддерживаемые http-статусы ответа: 200, 400, 404
 * @typedef {Object} ajaxStatus
 * @property {number} ok
 * @property {number} notFound
 * @property {number} badRequest
 */
const ajaxStatuses = {
  ok: 200,
  notFound: 404,
  badRequest: 400,
  invalidSession: 424,
};

/**
 * Выполняет ajax-запрос на сервер. При успешном выполнении вызывает callback
 * @param {Object} requestParams
 * @property {ajaxMethod} [method = "GET"]
 * @property {Url} [url = '/']
 * @property {any} body
 * @return {Promise}
 */
function ajaxCall(requestParams) {
  const url = APIurl + (requestParams.url || '/');
  const fetchParams = {
    body: JSON.stringify(requestParams.body),
    mode: 'cors',
    credentials: 'include',
    headers: {
      'Content-Type': 'application/json',
    },
    method: requestParams.method,
  };

  if (ajaxDebug) {
    console.log('ajax request', {url}, ': ' + fetchParams);
  }
```

```
54
55    let status = 0;
56    return fetch(url, fetchParams)
57        .then((response) => {
58          status = response.status;
59          return response.json();
60        })
61        .then((response) => {
62          if (ajaxDebug) {
63            console.log('ajax resolved ' + status + ': ');
64            console.log(response);
65          }
66          return {
67            status,
68            response,
69          };
70        })
71        .catch((error) => {
72          console.warn(error);
73        });
74  }
75
76  // Плагин для общения с API
77  const ajax = {
78    AJAX_METHODS: ajaxMethods,
79    STATUS: ajaxStatuses,
80    get: (requestParams) => ajaxCall({method: ajaxMethods.get, ...requestParams}),
81    post: (requestParams) => ajaxCall({method: ajaxMethods.post, ...requestParams}),
82  };
83
84  export default ajax;
```

Компоненты

Кнопка (либо переход на страницу диска, либо со страницы просмотра диска обратно, на главную)

lab-7 > public > components > button > JS button.js > ButtonComponent

```
1   export default class ButtonComponent {
2     constructor(root) {
3       this.root = root;
4     }
5
6     render(props, onClick) {
7       const btnWrapper = document.createElement('div');
8       btnWrapper.innerHTML = `<button type="button" class="btn btn-primary">${props.sign}</button>`;
9       const btn = btnWrapper.firstChild;
10      btn.addEventListener('click', onClick);
11      this.root.insertAdjacentElement('beforeend', btn);
12    }
13  }
14
```

«карточка» представления диск в списке дисков

lab-7 > public > components > diskCard > JS diskCard.js > DiskCardComponent

```
export default class DiskCardComponent {
  constructor(root) {
      this.root = root;
  }

  addListeners(data, listener) {
    document
        .getElementById(`click-card-${data.id}`)
        .addEventListener("click", listener);
}

  render(props, listener) {
  this.root.insertAdjacentHTML('beforeend', `
    <div class="card" style="width: 300px;">
      <img class="card-img-top" src="${props.src || '../../static/img/disk-icon.png'}" alt="${props.title}">
      <div class="card-body">
        <h5 class="card-title">${props.title}</h5>
        <p class="card-text">${props.description}</p>
        <button class="btn btn-primary" id="click-card-${props.id}" data-id="${props.id}">Нажми на меня</button>
      </div>
    </div>
    `);
    this.addListeners(props, listener);
  }
}

```

Блок, содержащий подробную информацию о диске, находящийся на странице просмотра диска:

lab-7 > public > components > disk > JS disk.js > DiskComponent

```
export default class DiskComponent {
  constructor(root) {
      this.root = root;
  }

  render(props) {
  this.root.insertAdjacentHTML('beforeend', `
    <div class="card" style="width: 300px;">
      <img class="card-img-top" src="${props.src || '../../static/img/disk-icon.png'}" alt="Диск ${props.title}">
      <div class="card-body">
        <h5 class="card-title">${props.title}</h5>
        <p class="card-text">Совсем другая верстка! ${props.description}</p>
      </div>
    </div>
    `);
  }
}

```

И главный скрипт main.js

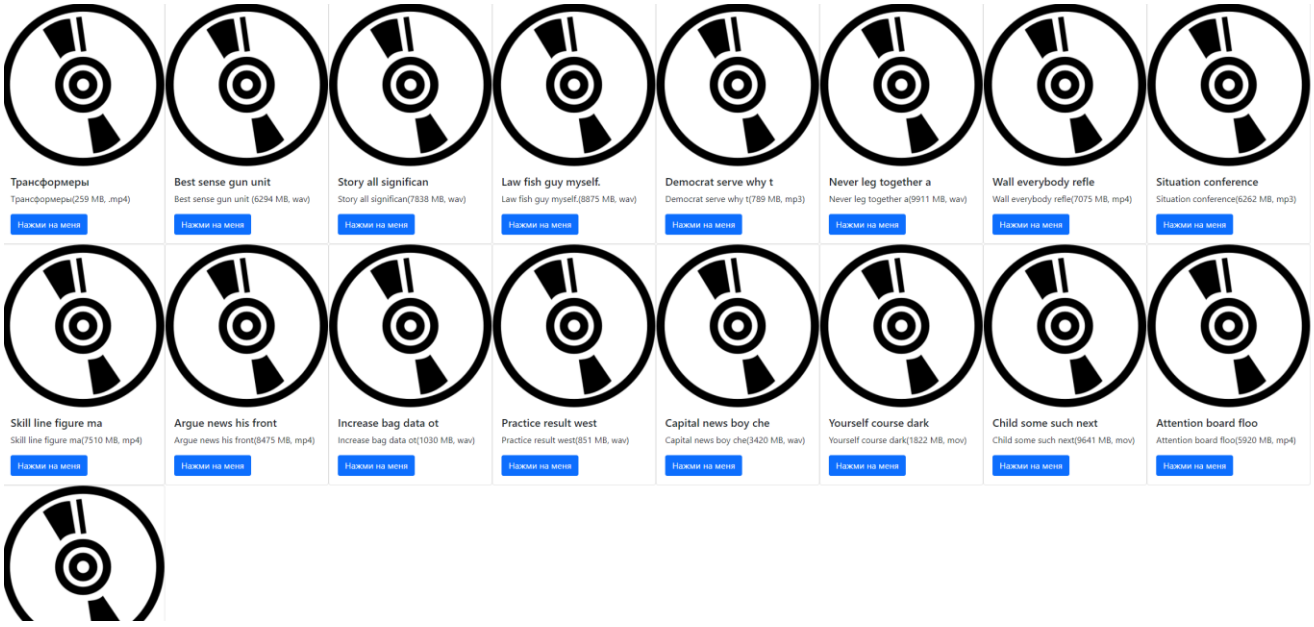
lab-7 > public > JS main.js > ...

```
import MainPage from "./pages/main/MainPage.js";

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

const mainPage = new MainPage(root);
mainPage.render();

```

Однако путешествие с ресурса localhost:8080 на localhost:8000 (API) является кросс-доменным запросом. Следовательно, надо настроить поддержку CORS на API сервере. Для этого использую библиотеку corseheaders

```
31  # Application definition
32
33  INSTALLED_APPS = [
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40      'rest_framework', # REST Framework
41      "corsheaders",    # CORS
42      'app',
43  ]
```

Результат



Страница диска

Law fish guy myself.

Совсем другая верстка! Law fish guy myself.(8875 MB, wav)

Назад