Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования «Ивановский государственный энергетический университет имени В.И. Ленина»

Кафедра программного обеспечения компьютерных систем

Отчёт по лабораторной работе №4

Конструирование интернет-приложений

Создание клиентской части

Выполнила студент гр. 3-42 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шарабанов Н.А.

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Садыков А.М.

Иваново 2022

**Цель лабораторной работы:** создать клиентское приложение с помощью библиотеки React с вызовом серверных методов.

**Задания:**

1. Настроить CORS в серверной части

1.1. Включить CORS в приложении

1.2. Включить CORS в котроллере

2. Создать шаблон клиентского приложения с библиотекой React

2.1. Установить Node.js

2.2. Создать шаблон приложение React

2.3. Открыть и запустить приложение React

3. Добавить вызов серверных методов в клиентской части

3.1. Создать компонент получения и отображения данных

3.2. Создать и подключить файл стилей

3.3. Подключить компонент отображения данных

3.4. Создать компонент добавления данных

3.5. Добавить удаление данных

**1. Результаты настройки CORS в серверной части**

**1.1. Включение CORS в приложение**

Cross-Origin Resource Sharing (CORS) — механизм, использующий дополнительные HTTP-заголовки, чтобы дать возможность агенту пользователя получать разрешения на доступ к выбранным ресурсам с сервера на источнике (домене), отличном от того, что сайт использует в данный момент. Говорят, что агент пользователя делает запрос с другого источника (cross-origin HTTP request), если источник текущего документа отличается от запрашиваемого ресурса доменом, протоколом или портом.

Был изменен код в program.cs для включения CORS в приложение. Результат показан на рисунке 1.

|  |
| --- |
|  |
| Рисунок 1 – Включение CORS в приложение |
| **1.2. Включение CORS в контроллере**  Также был изменен код в контроллерах OrderController для включения CORS в контроллер. Результат показан на рисунке 2.    Рисунок 2 – Включение CORS в контроллер  **2. Результат создания шаблона клиентского приложения с библиотекой React**  **2.1. Установка Node.js и создание шаблона приложения React**  Был установлен Node.js с официального сайта. Далее осуществлено создание шаблона приложения React через командную строку. Процесс и результат показан на рисунке 3. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

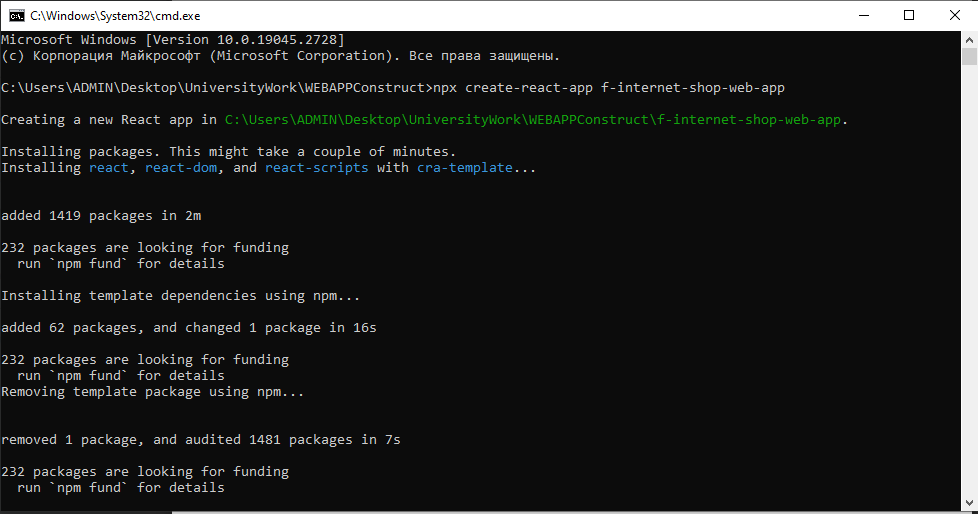


Рисунок 3 – Создание шаблона приложения

Далее через VS Code был открыт и запущен проект. Выполнена команда npm run start в терминале.

**3. Результат добавления серверных методов в клиентской части**

**3.1. Создание компонента получения и отображения данных**

В папке src создать папку Components. В ней создать папку с названием компонента OrderItem и в ней файл OrderItem.js. В данном файле создан компонент OrderItem для получения и отображения данных, а также методы get, set:

import React, { useEffect } from 'react';

import './Style.css';

const OrderItem = ({ OrderItems = [], setOrderItems, removeOrderItem }) => {

useEffect(() => {

const getOrderItems = async () => {

const requestOptions = {

method: 'GET',

};

try {

const response = await fetch('https://localhost:7194/api/OrderItem/', requestOptions);

const data = await response.json();

console.log('Data:', data);

setOrderItems(data);

} catch (error) {

console.log('Error:', error);

}

};

getOrderItems();

}, [setOrderItems]);

const deleteOrderItem = async ({ order\_Item\_Code }) => {

const requestOptions = {

method: 'DELETE'

}

return await fetch(`https://localhost:7194/api/OrderItem/${order\_Item\_Code}`,

requestOptions)

.then((response) => {

if (response.ok) {

removeOrderItem(order\_Item\_Code);

}

},

(error) => console.log(error)

)

}

return (

<React.Fragment>

<h3>Список блогов</h3>

{OrderItems && OrderItems.length > 0 ? (

OrderItems.map(({ order\_Item\_Code, order\_Sum, amount\_Order\_Item, product\_Code, order\_Code, status\_Order\_Item\_Table\_ID, products }) => (

<div className="OrderItem" key={order\_Item\_Code} id={order\_Item\_Code}>

<strong>

{order\_Item\_Code}: {amount\_Order\_Item}

</strong>

<button onClick={(e) => deleteOrderItem({ order\_Item\_Code})}>Удалить</button>

{products && products.map(({ product\_Code, orderItem\_Code, numberInStock, categoryID, dateOfManufacture, description, purchasePrice, marketPrice, bestBeforeDate, name, productOrderItem }) => (

<div className="OrderItemText" key={product\_Code} id={product\_Code}>

{name} <br />

{description}

<hr />

</div>

))}

</div>

))

) : (

<p>Загрузка данных...</p>

)}

</React.Fragment>

);

};

export default OrderItem;

**3.2. Создание и подключение файла стилей**

Также были подключены базовые стили через import:

.Cathegory {

font-size: large;

}

.CathegoryText {

padding: 10px;

}

**3.3.Подключение компонента отображения данных**

В index.js был изменен код для отображения полученных данных:

import React, { useState } from 'react'

import ReactDOM from "react-dom/client"

import OrderItem from './Components/OrderItem/OrderItem'

import OrderItemCreate from './Components/OrderItemCreate/OrderItemCreate'

const App = () => {

const [OrderItems, setOrderItems] = useState([])

const addOrderItem = (OrderItem) => setOrderItems([...OrderItems, OrderItem])

const removeOrderItem = (removeId) => setOrderItems(OrderItems.filter(({ order\_Item\_Code }) => order\_Item\_Code

!== removeId));

return (

<div>

<OrderItem

OrderItems={OrderItems}

setOrderItems={setOrderItems}

removeBlog={removeOrderItem}

/>

</div>

)

}

const root = ReactDOM.createRoot(document.getElementById("root"))

root.render(

// <React.StrictMode>

<App />

// </React.StrictMode>

)

**3.4. Создание компонента добавления данных**

В папке components создать папку OrderItemCreate в которой создать файл OrderItemCreate.js. В данном файле был создан компонент добавления данных.

import React from 'react'

const OrderItemCreate = ({ addOrderItem }) => {

const handleSubmit = (e) => {

e.preventDefault()

const { value } = e.target.elements.url

const OrderItem = { url: value }

// Изменить значение поля amount\_Order\_Item на 2

OrderItem.amount\_Order\_Item = 2;

const createOrderItem = async () => {

const requestOptions = {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify(OrderItem)

}

const response = await fetch("https://localhost:7194/api/OrderItem/",

requestOptions)

return await response.json()

.then((data) => {

console.log(data)

// response.status === 201 && addOrderItem(data)

if (response.ok) {

addOrderItem(data)

e.target.elements.url.value = ""

}

},

(error) => console.log(error)

)

}

createOrderItem()

}

return (

<React.Fragment>

<h3>Создание новой строки заказа</h3>

<form onSubmit={handleSubmit}>

<label>URL: </label>

<input type="text" name="url" placeholder="Введите Url:" />

<button type="submit">Создать</button>

</form>

</React.Fragment >

)

}

export default OrderItemCreate

После создания он был подключен в index.js.

return (

<div>

<OrderItemCreate

OrderItem={addOrderItem}

/>

<OrderItem

OrderItems={OrderItems}

setOrderItems={setOrderItems}

removeBlog={removeOrderItem}

/>

</div>

)

**3.5. Удаление данных**

Был добавлен компонент удаления данных. В OrderItem.js и index.js были внесены изменения:

OrderItem.js:

<button onClick={(e) => deleteOrderItem({ order\_Item\_Code})}>Удалить</button>

index.js:

removeBlog={removeOrderItem}

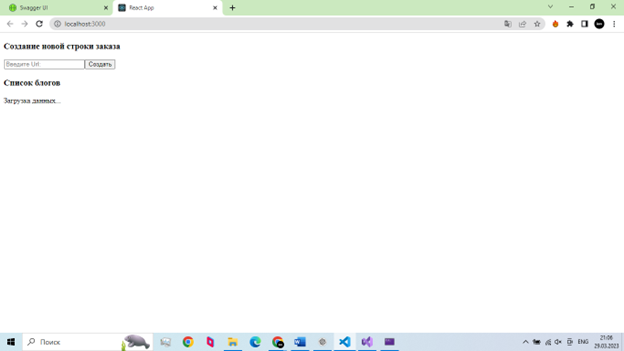


Рисунок 4 – Результат запуска приложения

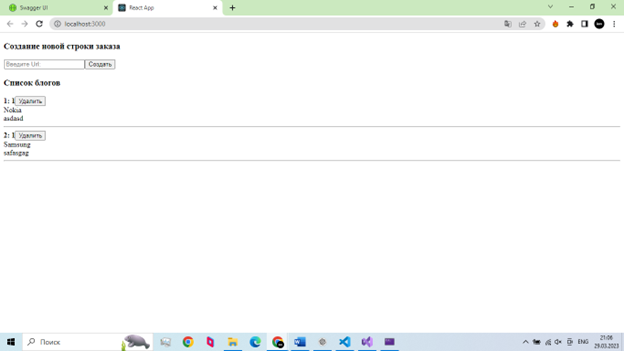


Рисунок 5 – Результат загрузки данных

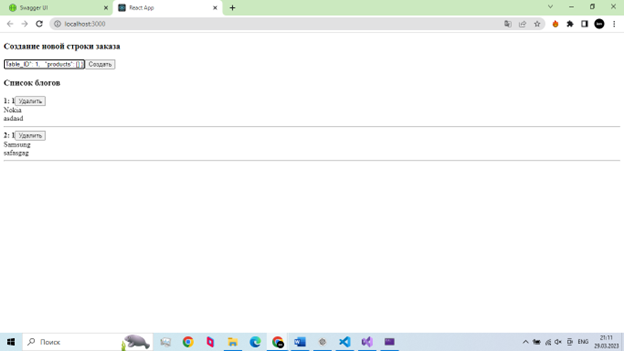


Рисунок 6 – Добавление данных

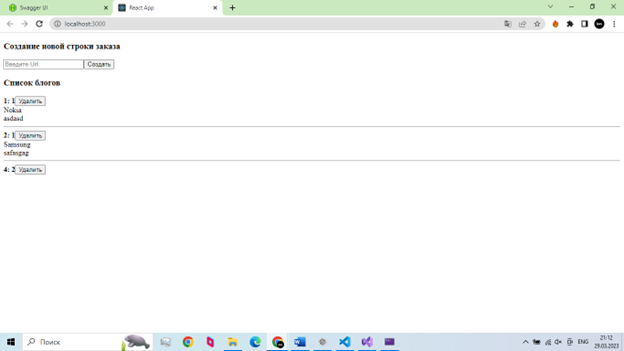


Рисунок 7 – Результат добавления данных

**Вывод**

В ходе лабораторной работы создала клиентское приложение с помощью библиотеки React с вызовом серверных методов.