

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 контроллере
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.

```
1  using InternetShopWebApp.Context;  
2  using InternetShopWebApp.Data;  
3  using Microsoft.Extensions.DependencyInjection;  
4  using System.Text.Json.Serialization;  
5  
6  var builder = WebApplication.CreateBuilder(args);  
7  
8  builder.Services.AddCors(options =>  
9  {  
10     options.AddDefaultPolicy(builder =>  
11     {  
12         builder.WithOrigins("http://localhost:3000")  
13         .AllowAnyHeader()  
14         .AllowAnyMethod();  
15     });  
16 });  
17 });
```

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.

```

1  using Microsoft.AspNetCore.Mvc;
2  using Microsoft.EntityFrameworkCore;
3  using InternetShopWebApp.Models;
4  using Microsoft.AspNetCore.Cors;
5
6  namespace InternetShopWebApp.Controllers
7  {
8      [Route("api/[controller]")]
9      [EnableCors]
10     [ApiController]
11     public class OrderItemController : ControllerBase
12     {
13         private readonly Context.ShopContext _context;
14         public OrderItemController(Context.ShopContext context)
15         {
16             _context = context;
17             if (!_context.OrderItem.Any())
18             {
19                 _context.OrderItem.Add(new OrderItemModel
20                 {
21                     Order_Item_Code = 1,
22                     Order_Sum = 100,
23                     Amount_Order_Item = 1,
24                     Product_Code = 1,
25                     Order_Code = 1,
26                     Status_Order_Item_Table_ID = 1
27                 });
28                 _context.SaveChanges();
29             }
30         }
31     }

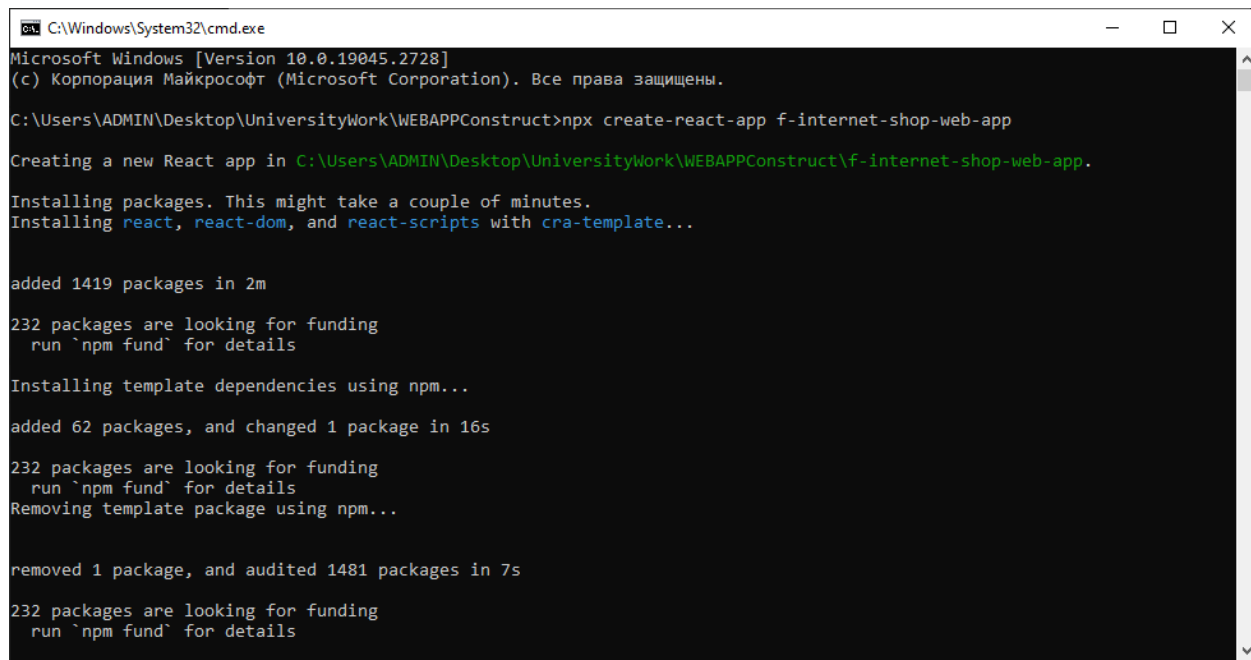
```

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.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.2728]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\ADMIN\Desktop\UniversityWork\WEBAPPConstruct>npx create-react-app f-internet-shop-web-app

Creating a new React app in C:\Users\ADMIN\Desktop\UniversityWork\WEBAPPConstruct\f-internet-shop-web-app.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

added 1419 packages in 2m

232 packages are looking for funding
  run `npm fund` for details

Installing template dependencies using npm...

added 62 packages, and changed 1 package in 16s

232 packages are looking for funding
  run `npm fund` for details
Removing template package using npm...

removed 1 package, and audited 1481 packages in 7s

232 packages are looking for funding
  run `npm fund` for details
```

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}
```

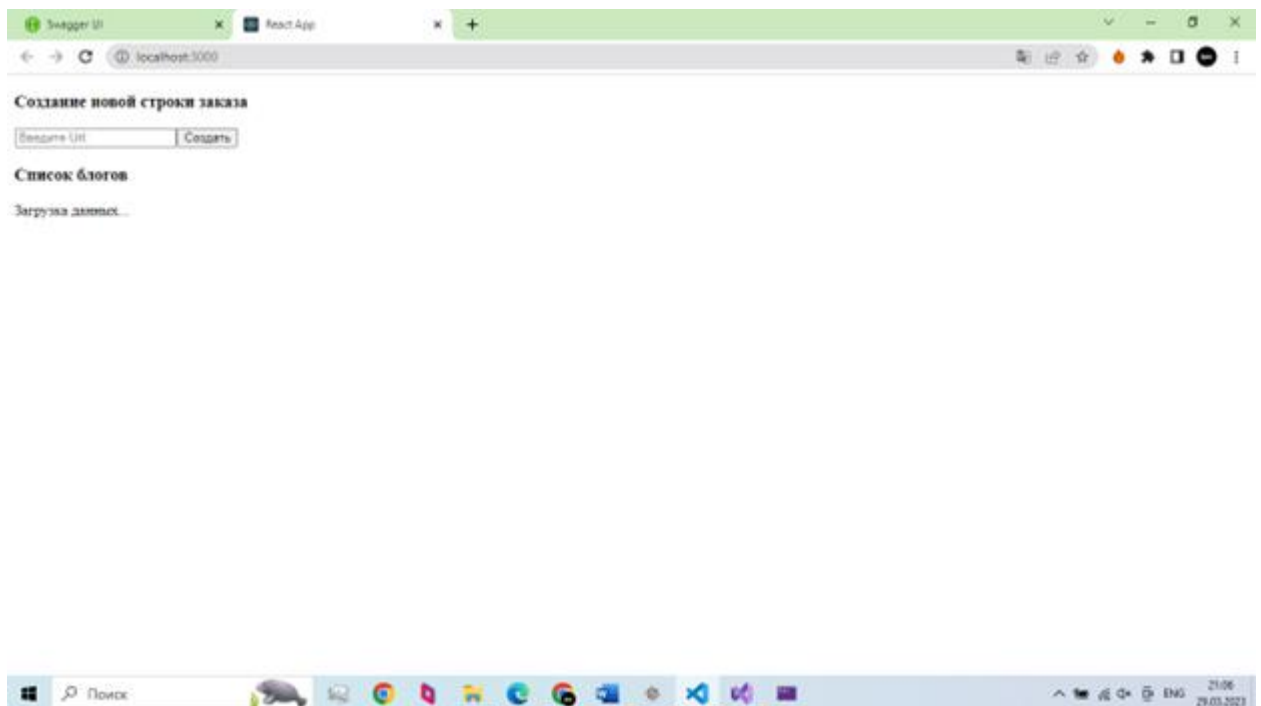


Рисунок 4 – Результат запуска приложения

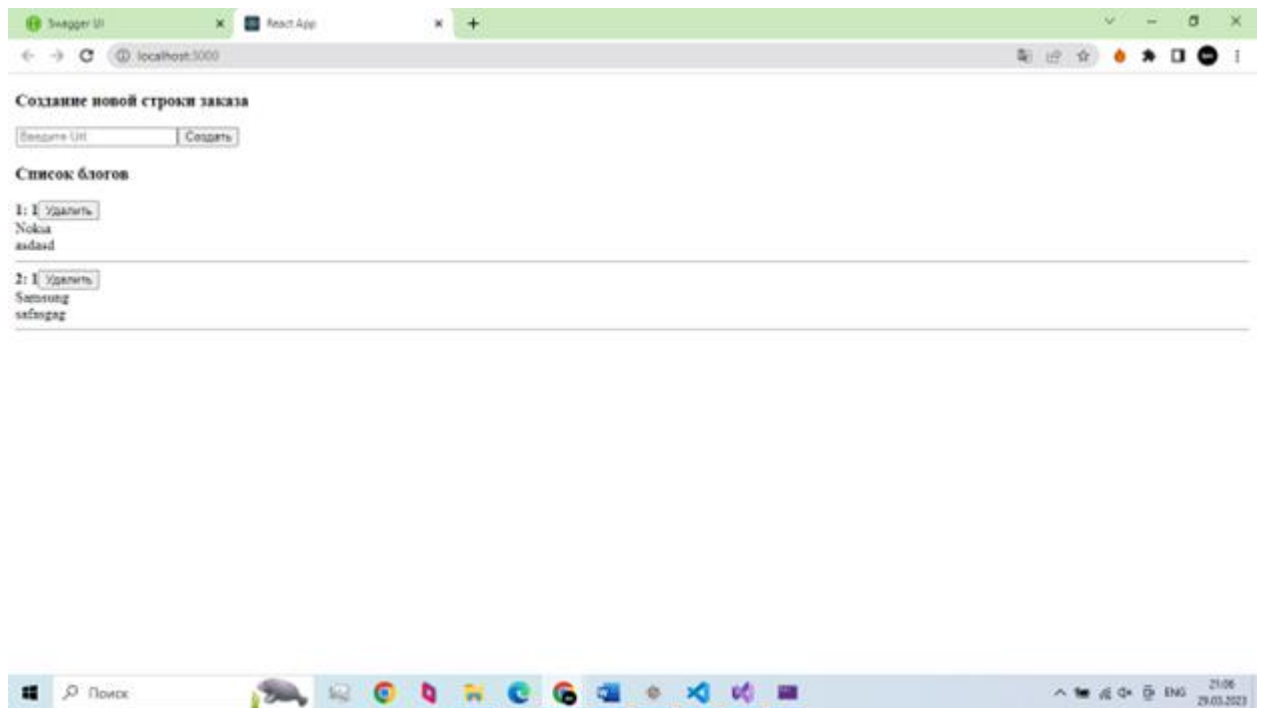


Рисунок 5 – Результат загрузки данных

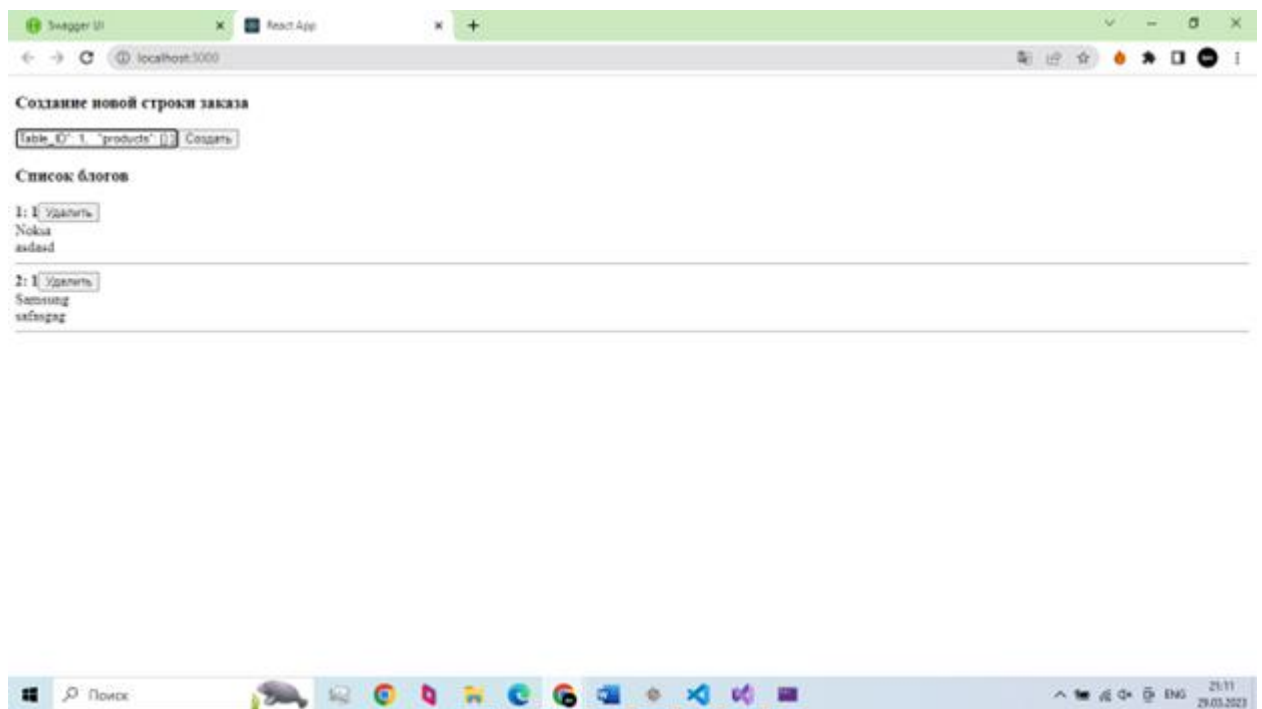


Рисунок 6 – Добавление данных

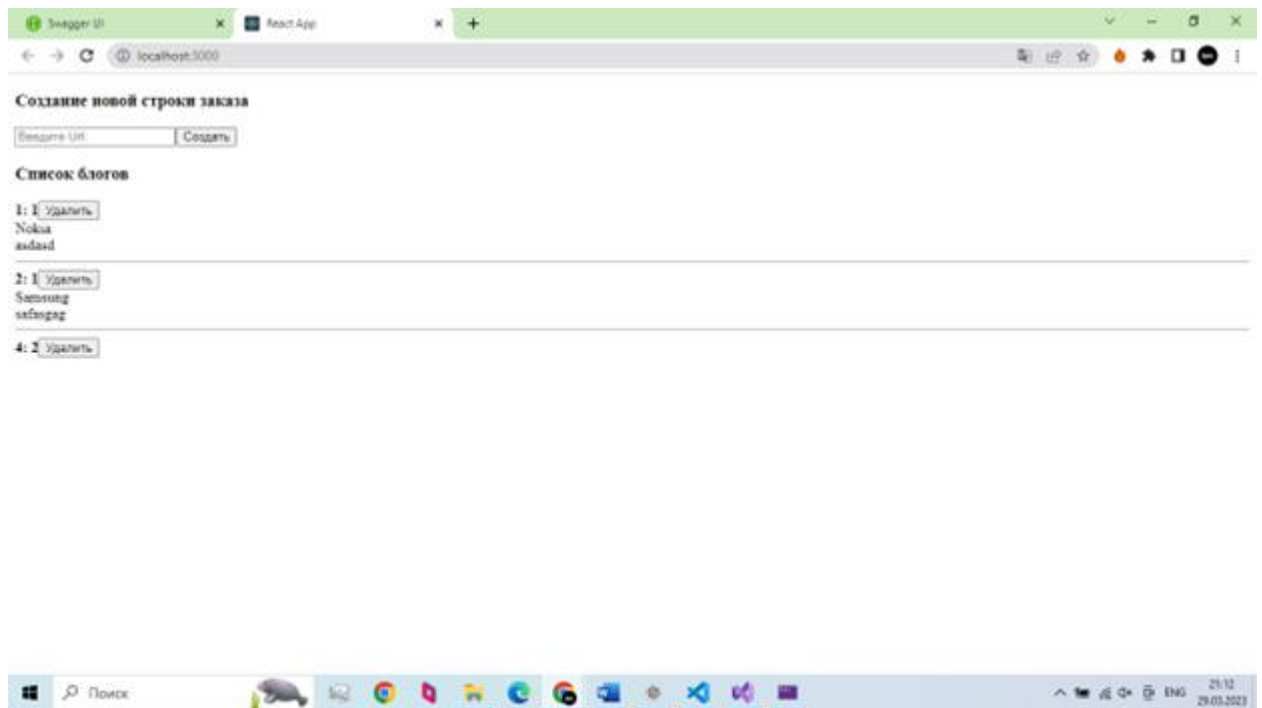


Рисунок 7 – Результат добавления данных

Вывод

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