



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 КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Информатика и вычислительная техника

О Т Ч Е Т

по лабораторной работе № 7

Название: **Основы Front-End разработки на JavaScript**

Дисциплина: **Языки интернет программирования**

Студент

ИУ6-33Б

(Группа)

(Подпись, дата)

Д.А. Лазутин

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

В.Д. Шульман

(И.О. Фамилия)

Москва, 2024

Цель работы — изучение основ разработки SPA-приложение на JavaScript.

Задание. Реализовать пользовательский веб-интерфейс для взаимодействия с микросервисами, которые были получены в ходе выполнения предыдущей лабораторной работы. Взаимодействие с Back-End частью веб-приложения должно осуществляться с помощью AJAX-запросов.

Напишем код, реализующий фронт сервер.

// index.js

```
import React, { StrictMode } from 'react';
import ReactDOM from 'react-dom/client';
import App from './components/App';
```

```
const root = ReactDOM.createRoot(document.getElementById('root'));
```

```
root.render(
  <StrictMode>
    <App />
  </StrictMode>
);
```

// App.js

```
import React from "react";
import HelloService from "./HelloService";
import QueryService from "./QueryService";
import CountService from "./CountService";
```

```
const App = () => {
  console.log("App is rendering...");
```

```
  return (
    <div>
      <h1>Микросервисы</h1>
      <HelloService />
      <QueryService />
      <CountService />
    </div>
  );
};
```

```
export default App; // Экспорт по умолчанию
```

CountServer.js

```
import React, { useState, useEffect } from "react";
import { getCount, postCount } from "../services/api";
```

// Функциональный компонент для работы с сервисом Count

```
const CountService = () => {
```

// Состояние для хранения текущего значения счётчика

```
const [count, setCount] = useState(0);
```

// Состояние для хранения введённого пользователем значения инкремента

```
const [increment, setIncrement] = useState("");
```

// Хук useEffect срабатывает при первом рендере компонента, вызывая fetchCount

```

useEffect(() => {
  fetchCount(); // Загружаем текущее значение счётчика
}, []); // Пустой массив зависимостей — хук выполнится только при монтировании

// Асинхронная функция для получения значения счётчика с сервера
const fetchCount = async () => {
  try {
    const response = await getCount(); // Вызов API для получения данных
    setCount(parseInt(response, 10)); // Парсим текстовый ответ как число и обновляем состояние
  } catch (error) {
    console.error("Ошибка при получении счётчика:", error);
    alert("Не удалось загрузить значение счётчика.");
  }
};

// Асинхронная функция для отправки нового значения инкремента на сервер
const updateCount = async () => {
  try {
    await postCount(increment); // Вызов API для обновления данных
    fetchCount(); // Обновляем значение счётчика после изменения
  } catch (error) {
    console.error("Ошибка при обновлении счётчика:", error);
    alert("Не удалось обновить значение счётчика.");
  }
};

return (
  <div>
    <h2>Сервис Count</h2>
    <p>Текущий счётчик: {count}</p>
    { /* Поле ввода для значения инкремента */ }
    <input
      type="number" // Тип ввода: числовое поле
      value={increment} // Текущее значение инкремента
      onChange={e => setIncrement(e.target.value)} // Обновляем состояние при вводе
      placeholder="Введите число"
    />
    { /* Кнопка для отправки нового значения инкремента */ }
    <button onClick={updateCount}>Обновить счётчик</button>
  </div>
);
};

export default CountService; // Экспорт компонента для использования в других местах

// HelloService.js
import React, { useState } from "react";
import { getHello } from "../services/api";

const HelloService = () => {
  const [message, setMessage] = useState("");

```

```
const fetchHello = async () => {
  try {
    const response = await fetch("http://localhost:8080/get");
    if (!response.ok) {
      throw new Error("Ошибка при получении данных");
    }
    // Читаем ответ как текст
    const data = await response.text();
    setMessage(data); // Устанавливаем полученное текстовое сообщение
  } catch (error) {
    console.error("Ошибка:", error);
  }
};

return (
  <div>
    <h2>Сервис Hello</h2>
    <button onClick={fetchHello}>Получить приветствие</button>
    <p>{message}</p>
  </div>
);

export default HelloService; // Экспорт по умолчанию
```

QueryService.js

```
import React, { useState } from "react";
import { getUser } from "../services/api";

const QueryService = () => {
  const [name, setName] = useState("");
  const [message, setMessage] = useState("");

  const fetchUser = async () => {
    try {
      const response = await getUser(name);
      if (!response.ok) {
        throw new Error("Ошибка при получении данных");
      }
      // Читаем ответ как текст
      const data = await response.text();
      setMessage(data); // Устанавливаем полученное текстовое сообщение
    } catch (error) {
      console.error("Ошибка:", error);
      setMessage("Ошибка при запросе данных");
    }
  };

  return (
    <div>
      <h2>Сервис Query</h2>
      <input
        type="text"

```

```

value={name}
placeholder="Введите имя"
onChange={(e) => setName(e.target.value)}
/>
<button onClick={fetchUser}>Получить приветствие</button>
<p>{message}</p>
</div>
);
};

```

```
export default QueryService;
```

```
api.js
```

```

const BASE_URLS = {
  hello: "http://localhost:8080",
  query: "http://localhost:9000/api/user",
  count: "http://localhost:3333"
};
export async function getHello() {
  const response = await fetch(`${BASE_URLS.hello}/get`);
  return response.json(); // Используем .json() для парсинга данных в объект
}

```

```

export async function getUser(name) {
  const response = await fetch(`http://localhost:9000/api/user?name=${name}`);
  return response;
}

```

```

export async function getCount() {
  const response = await fetch(`${BASE_URLS.count}/count`);
  return response.json(); // То же самое для счётчика
}

```

```

export async function postCount(value) {
  const response = await fetch(`${BASE_URLS.count}/count`, {
    method: "POST",
    headers: { "Content-Type": "application/x-www-form-urlencoded" },
    body: `count=${value}`, // Тело запроса содержит данные для инкремента
  });
}

```

```

if (!response.ok) {
  throw new Error("Ошибка обновления счётчика"); // Генерация исключения, если запрос не успешен
}

```

```

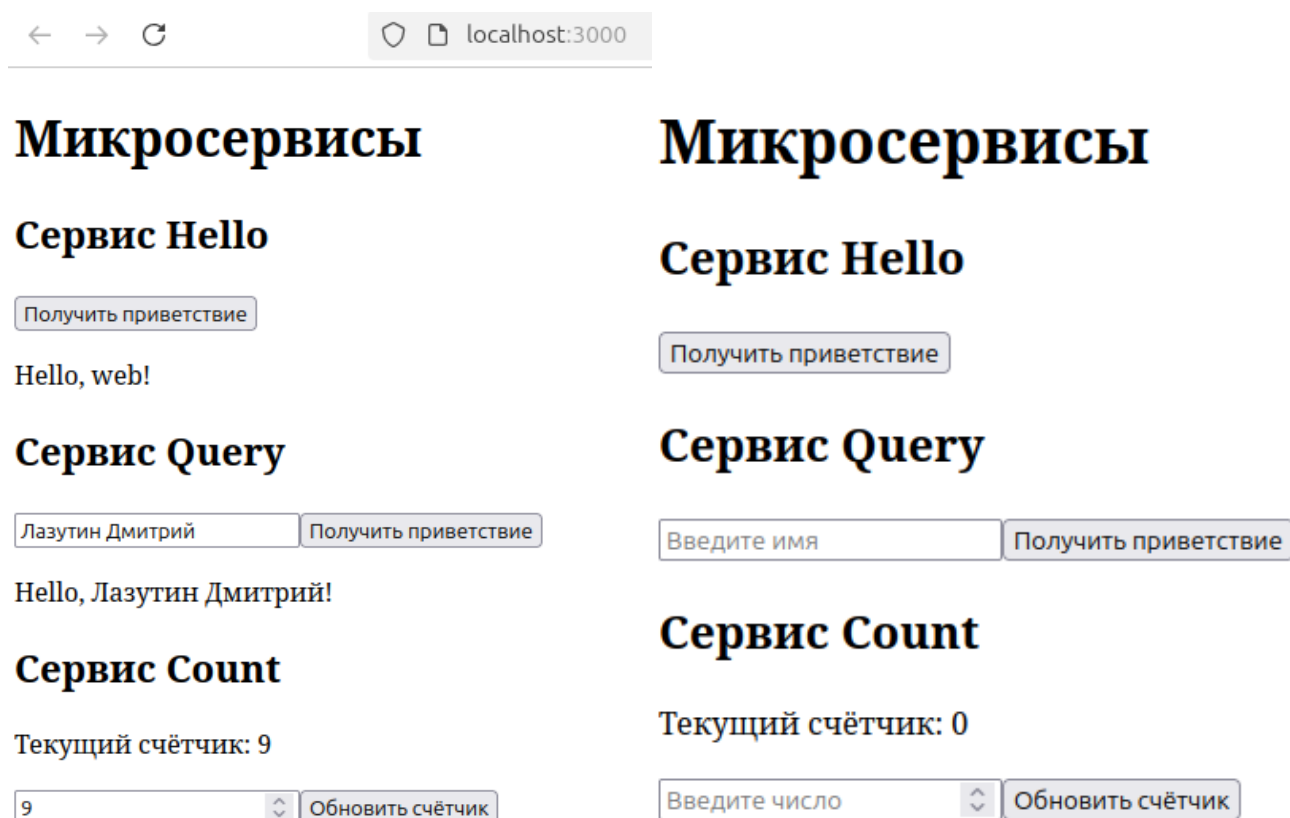
// Пытаемся обработать ответ как текст, так как сервер не возвращает JSON
const textResponse = await response.text();
// Преобразуем текстовый ответ в число (если нужно)
return parseInt(textResponse, 10); // Возвращаем число для дальнейшей обработки
}

```

```
index.html
```

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>React App</title>
</head>
<body>
<div id="root"></div>
</body>
</html>
```

Запустим код и проверим его (рис 1)



А

Б

Рисунок 1Б — запуск сервера. Рисунок 1А — результаты тестирования команд
Вывод. В результате программы был изучен метод создания фронтэнд
сервера и его взаимодействие с бекэнд кодом на go.