|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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**

**Отчет**

**по лабораторной работе № 7**

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

**Дисциплина:** Языки интернет-программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-32Б |  | 29.11.2024 | Л.И. Заушников |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  | 29.11.2024 | В.Д. Шульман |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2024

**Цель работы** — изучить основы разработки SPA-приложение на JavaScript.

**Ход работы**

1. Было произведено ознакомление с материалами для подготовки перед выполнением лабораторной работы
2. Был сделан форк репозитория в GitHub (рисунок 1), копия была склонирована локально, была создана от мастера ветка dev и было произведено переключение на неё

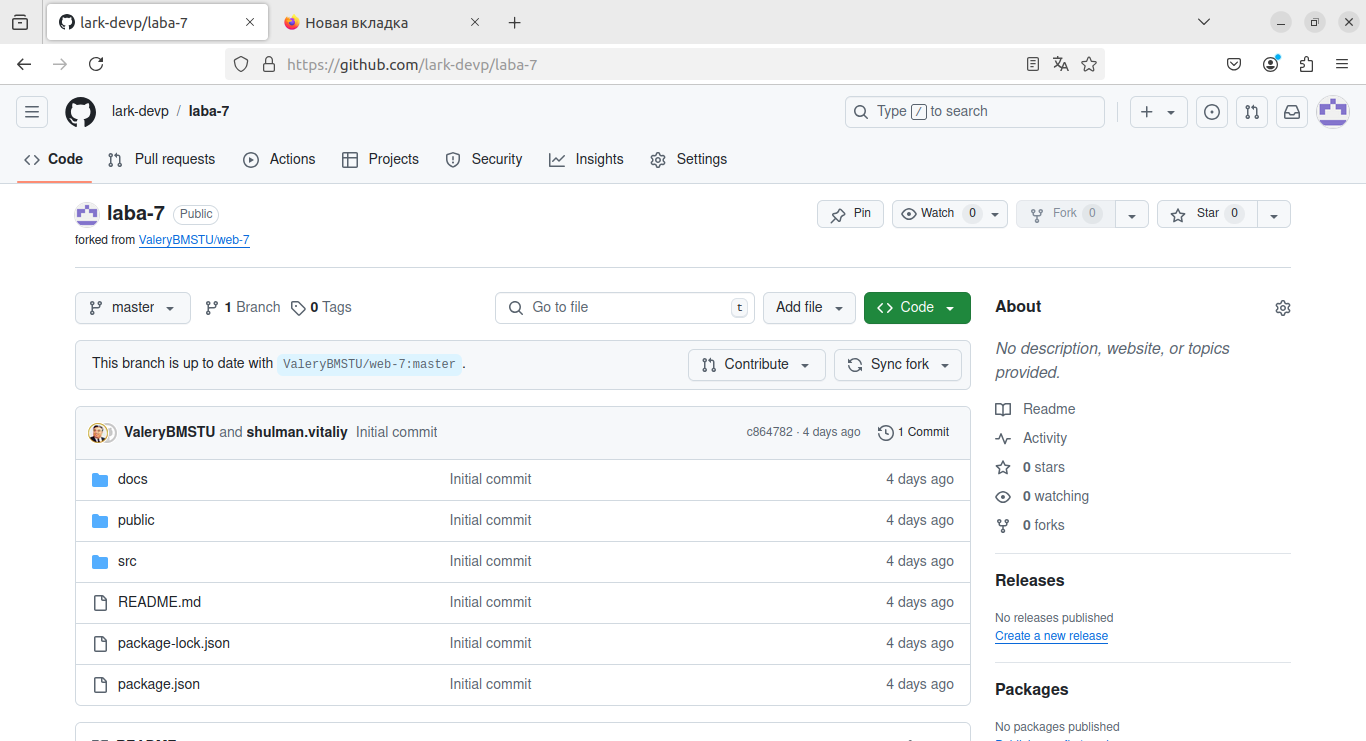


Рисунок 1 - Форкнутый репозиторий

Далее были написаны 3 компонента для React проекта. Код этих компонентов ниже:

**Компонент 1 (вывод строки)**

import React, { useState } from 'react';

const Hello = () => {

const [message, setMessage] = useState('');

const fetchHelloMessage = async () => {

try {

const response = await fetch('http://localhost:8082/getMessage');

if (!response.ok) {

throw new Error('Запрос не удался');

}

const data = await response.text();

setMessage(data);

// Таймер, который очищает сообщение через 1,5 секунд

setTimeout(() => {

setMessage(''); // очищаем сообщение

}, 1500);

} catch (error) {

console.error('Ошибка получения:', error);

setMessage('Ошибка при получении сообщения');

}

};

return (

<div>

<h2>Заслуженное приветствие</h2>

<button onClick={fetchHelloMessage}>Получить привет</button>

<p>Сообщение: {message}</p>

</div>

);

};

export default Hello;

**Компонент 2 (вывод строки с ключом)**

import React, { useState } from 'react';

const User = () => {

const [name, setName] = useState('');

const [greeting, setGreeting] = useState('');

const fetchUserGreeting = async () => {

// Проверяем, является ли имя пустым

if (!name.trim()) { //Метод trim() удаляет пробельные символы с начала и конца строки.

// Если имя пустое, устанавливаем сообщение с просьбой ввести данные

setGreeting('Пожалуйста, введите Ваше имя');

return; // Выходим из функции раньше времени

}

try {

const response = await fetch(`http://localhost:8083/api/user?name=${name}`);

if (!response.ok) {

throw new Error('Сетевого ответа нет');

}

const data = await response.text();

setGreeting(data);

} catch (error) {

console.error('Ошибка:', error);

setGreeting('Ошибка при получении приветствия');

}

};

return (

<div>

<h2>Давайте знакомиться</h2>

<input

type="text"

placeholder="Введите своё имя"

value={name}

onChange={(e) => setName(e.target.value)}

/>

<button onClick={fetchUserGreeting}>Поприветствовать</button>

<p>Сообщение: {greeting}</p>

</div>

);

};

export default User;

**Компонент 3 (счётчик)**

import React, { useState } from 'react';

const Count = () => {

const [count, setCount] = useState(0); // Состояние для хранения значения счётчика

const [inputValue, setInputValue] = useState(''); // Состояние для хранения значения из input

const [error, setError] = useState(null); // Состояние для хранения ошибок

const [isCountVisible, setIsCountVisible] = useState(false); // Состояние для управления видимостью счётчика

// Функция для получения текущего значения счётчика

const fetchCount = async () => {

try {

const response = await fetch('http://localhost:8081/count');

if (!response.ok) {

throw new Error('Сетевое соединение не в порядке');

}

const data = await response.text();

setCount(parseInt(data, 10)); // Преобразуем строку в число

setIsCountVisible(true); // Показываем счётчик

setError(null); // Сбрасываем ошибку

// Прячем счётчик через 1,5 секунд

setTimeout(() => {

setIsCountVisible(false);

}, 1500);

} catch (error) {

console.error('Ошибка:', error);

setError('Ошибка при получении счётчика'); // Устанавливаем сообщение об ошибке

}

};

// Функция для увеличения счётчика

const incrementCount = async () => {

const valueToAdd = parseInt(inputValue, 10); // Преобразуем введённое значение в число

if (isNaN(valueToAdd) || valueToAdd <= 0) {

setError('Введите положительное число'); // Проверка на корректность введённого значения

return;

}

try {

await fetch('http://localhost:8081/count', {

method: 'POST',

headers: {

'Content-Type': 'application/x-www-form-urlencoded',

},

body: new URLSearchParams({ count: valueToAdd }), // Увеличиваем на введённое значение

});

setIsCountVisible(false); // Скрываем счётчик после увеличения

setInputValue(''); // Очищаем поле ввода

setError(null); // Сбрасываем ошибку

} catch (error) {

console.error('Ошибка:', error);

setError('Ошибка при увеличении счётчика'); // Устанавливаем сообщение об ошибке

}

};

return (

<div>

<h2>Счётчик</h2>

<button onClick={fetchCount}>Получить счётчик</button>

<div>

<input

type="number"

value={inputValue}

onChange={(e) => setInputValue(e.target.value)}

placeholder="Введите число"

/>

<button onClick={incrementCount}>Увеличить счётчик</button>

</div>

{isCountVisible && <p>Счётчик: {count}</p>} {/\* Отображаем счётчик только если он видим \*/}

{error && <p style={{ color: 'red' }}>{error}</p>} {/\* Отображаем ошибки \*/}

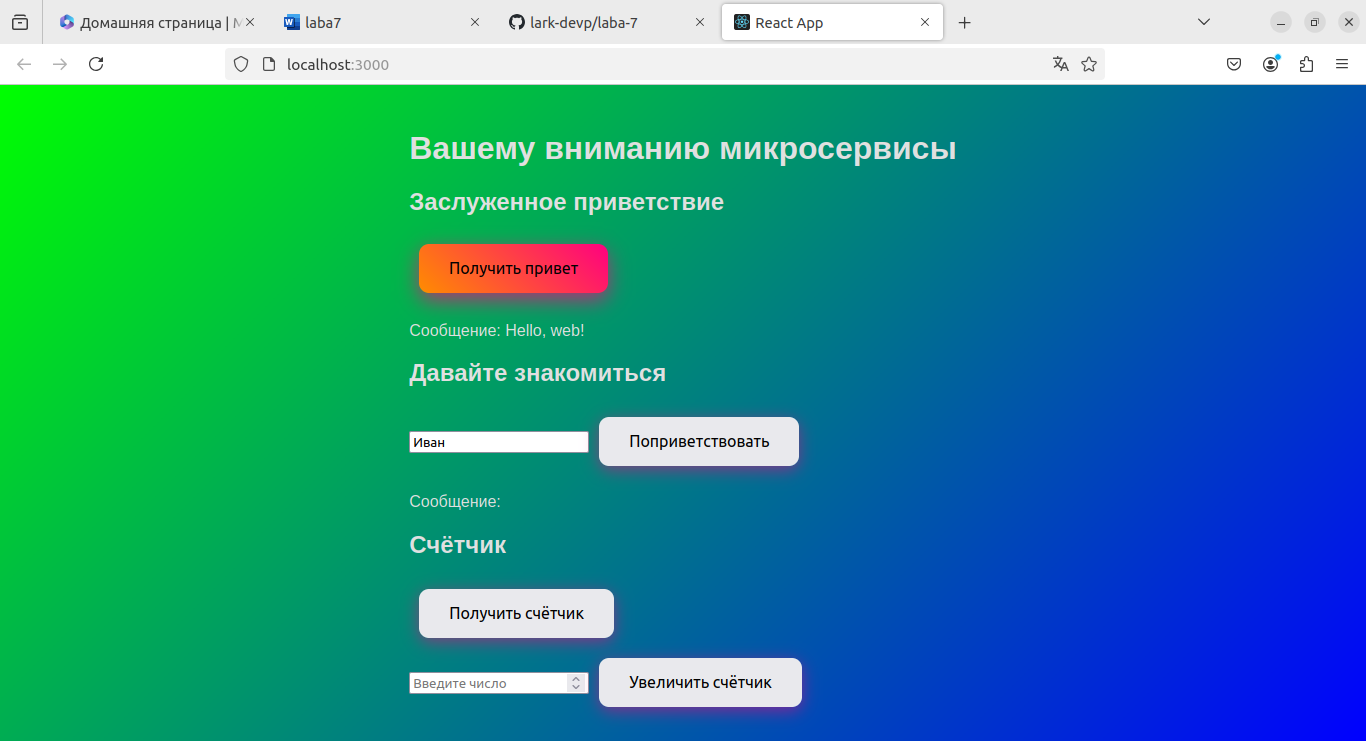
</div>

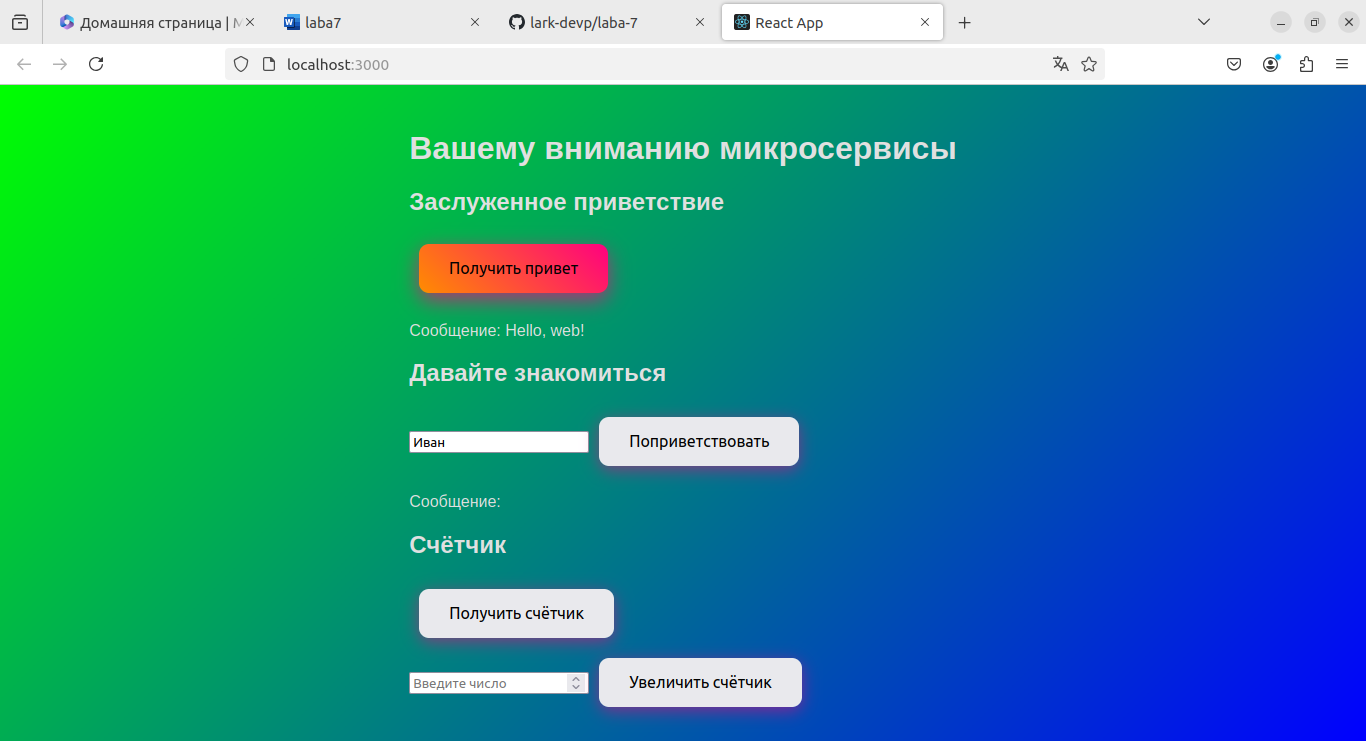
);

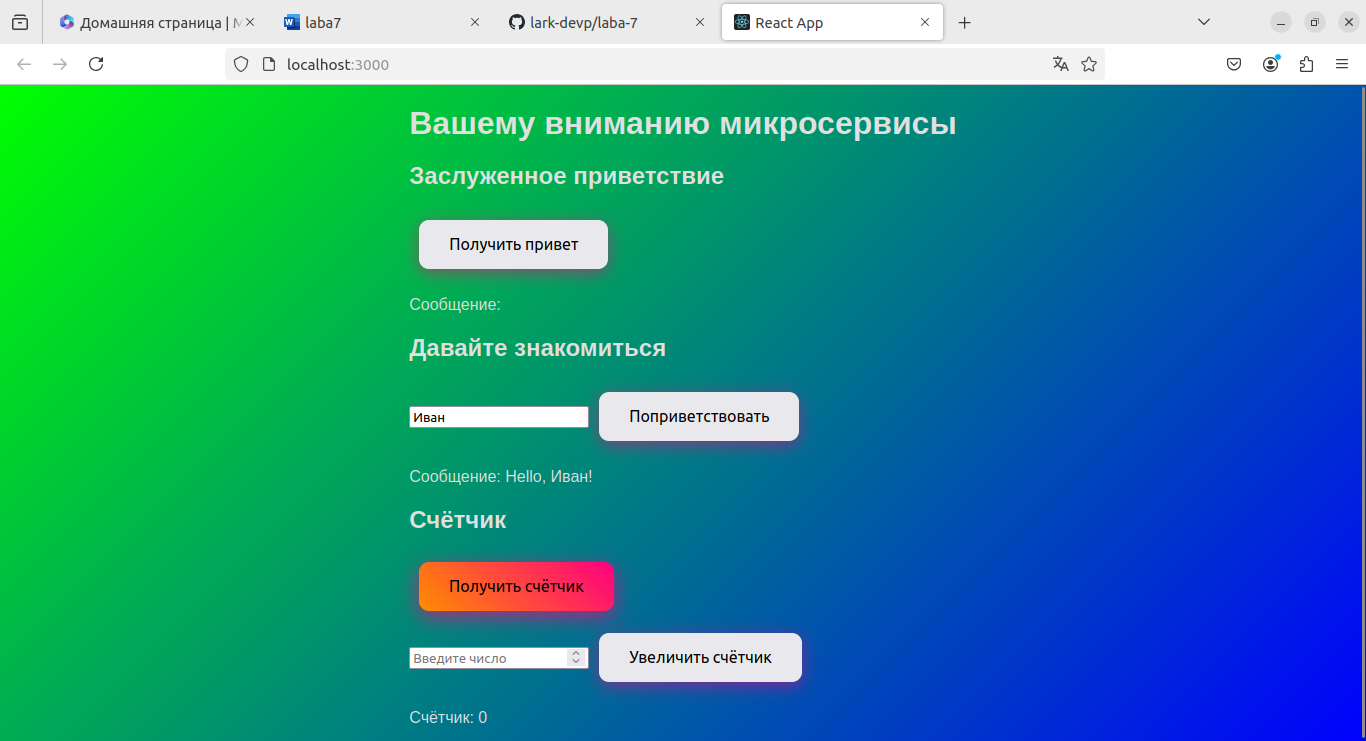
};

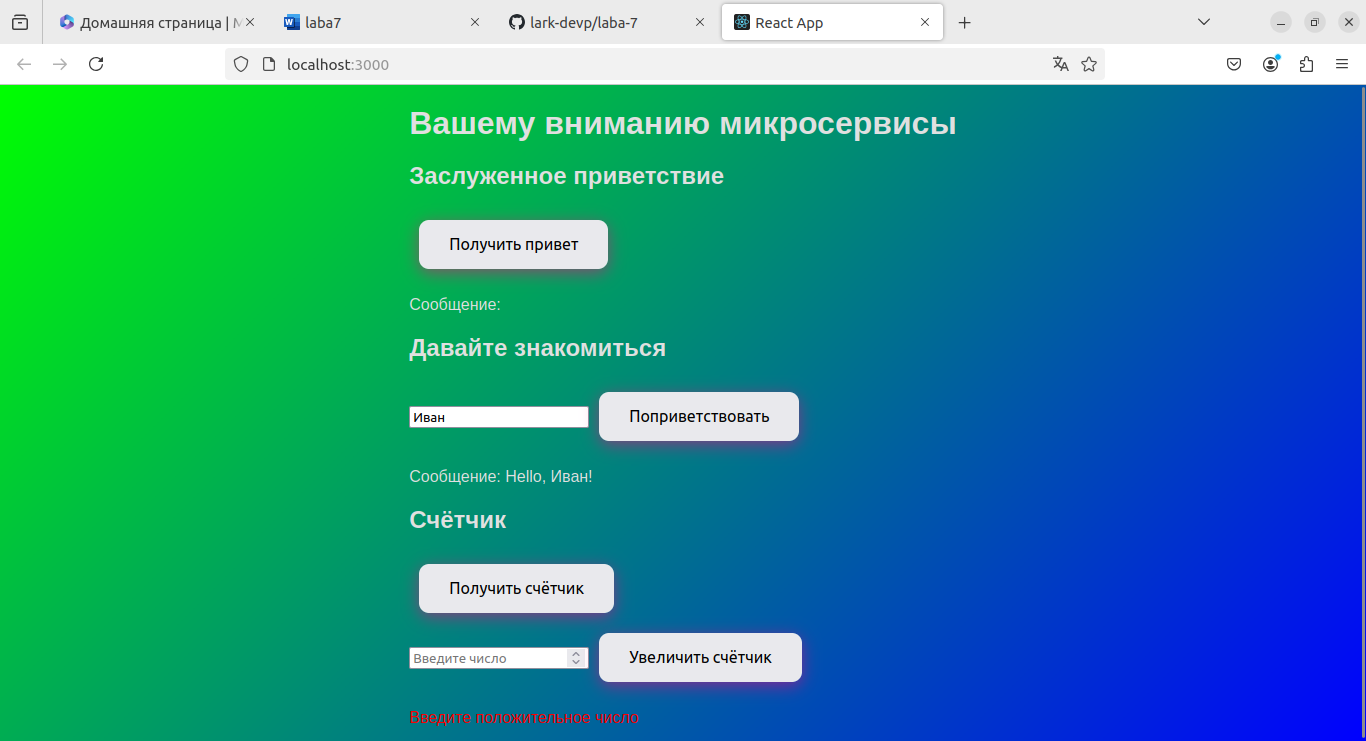
export default Count;

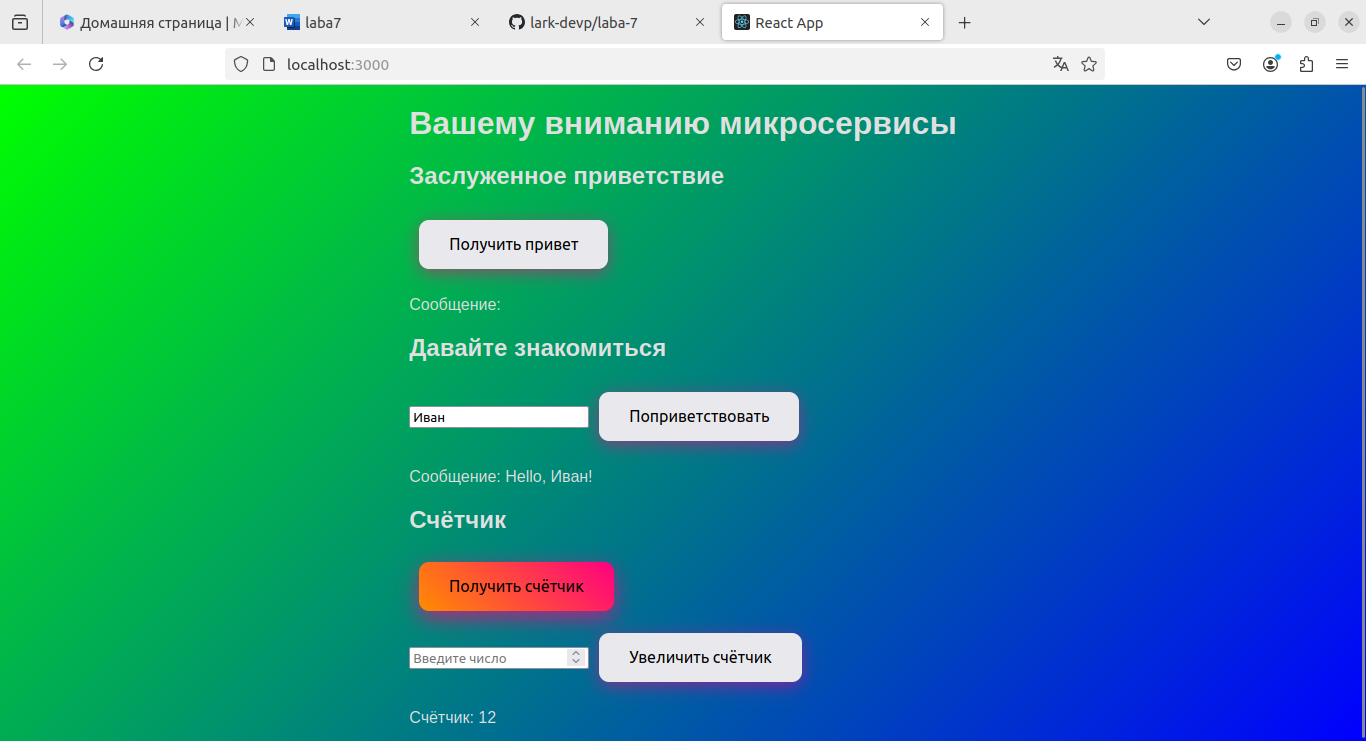
Проект с тремя микросервисами был протестирован, результаты тестирования на рисунках 2-7.

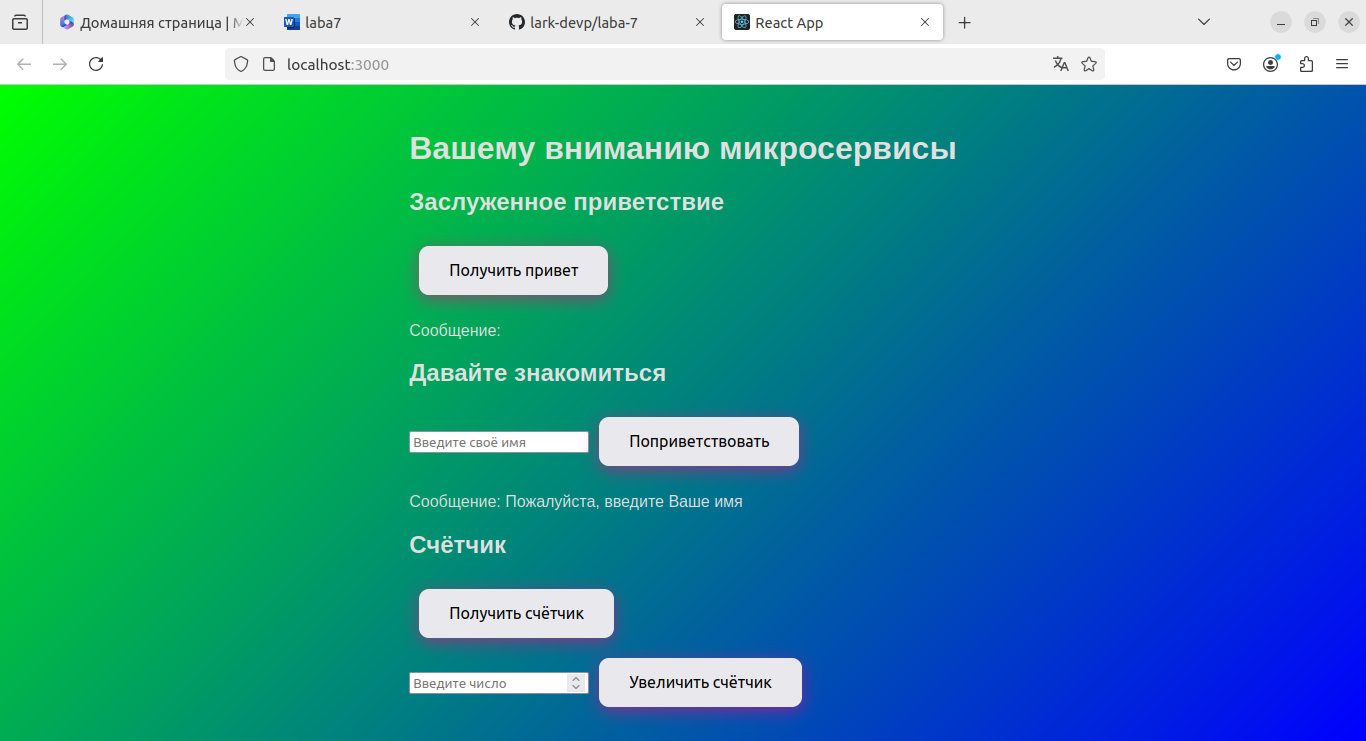
Рисунок 2 - Тестирование проекта с 3-мя микросервисами

Рисунок 3 - Тестирование проекта с 3-мя микросервисами

Рисунок 4 - Тестирование проекта с 3-мя микросервисами

Рисунок 5 - Тестирование проекта с 3-мя микросервисами

Рисунок 6 - Тестирование проекта с 3-мя микросервисами

Рисунок 7 - Тестирование проекта с 3-мя микросервисами

1. Все изменения были зафиксированы, был сделан commit и произошла отправка в удалённый репозиторий GitHub. Через интерфейс GitHub был создан Pull Request dev --> master.

**Заключение:** в ходе выполнения лабораторной работы были изучены основы разработки SPA-приложение на JavaScript.

**Список источников**

1. <https://github.com/coreybutler/nvm-windows>
2. <https://github.com/coreybutler/nvm-windows>
3. <https://tproger.ru/articles/ponimanie-raznicy-mezhdu-npm-i-npx>
4. <https://create-react-app.dev/docs/getting-started/>
5. <https://jsonplaceholder.typicode.com/>
6. <https://habr.com/ru/companies/ruvds/articles/418085/>
7. <https://yandex.ru/video/preview/7850899982348140117>
8. <https://www.youtube.com/watch?v=OtAlPwW8DNU&t=409s>
9. <https://reactrouter.com/en/main/start/tutorial>
10. https://www.youtube.com/playlist?list=PLiZoB8JBsdznY1XwBcBhHL9L7S\_shPGVE