



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)

20.11.24

(Подпись, дата)

Пономаренко
В.М.

(И.О. Фамилия)

Преподаватель

20.11.24

(Подпись, дата)

Шульман В.Д.

(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.

Ход работы:

ЗАДАНИЕ 1. Микросервис Hello

Необходимо, чтобы в результате взаимодействия пользовательского веб-интерфейса с сервером на страничке отображалась приветственная фраза.

Ниже представлены листинги

```
package main

import (
    "fmt"
    "net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Access-Control-Allow-Origin", "*") // Разрешить все источники (ДЛЯ ВЗАИМОДЕЙСТВИЯ С FRONT)
    w.Write([]byte("Hello web!"))
}

func main() {
    http.HandleFunc("/get", handler) // регистрация обработчика для пути "/get"
    err := http.ListenAndServe(":8082", nil) // запуск сервера на порту 8082(по условию)
    if err != nil {
        fmt.Println("error")
    }
}
```

Листинг файла server.go

```
"scripts": {
  "start": "set PORT=80 && react-scripts start",
```

Часть листинга package.json. Изменение данной строки проводится для запуска React-приложения на порте 80 (по требованию задания лабораторной работы)

Эта строка идентично изменена во всех подпроектах.

```
import React, { useEffect, useState } from 'react';
import './index.css'; // Импортируем CSS
function App() {
    const [message, setMessage] = useState('');
    // определение компонента App - основа приложения
    // message - состояние, setMessage - функция для его обновления
    useEffect(() => {
        //useEffect позволяет выполнять побочные эффекты
        fetch('http://localhost:8082/get')
        //fetch - метод для выполнения http запроса
        .then(response => response.text())//response.text(): Преобразует ответ в текст.
        .then(data => setMessage(data))//setMessage(data): Обновляет состояние message с полученными данными.
        .catch(error => console.error('Ошибка при получении данных:', error));
    }, []);

    return (
        <div className="App">
            <h1>{message}</h1>
        </div>
    );
}

export default App;
```

Листинг App.js

```
body {
```

```

margin: 0;
font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
  'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
  sans-serif;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}

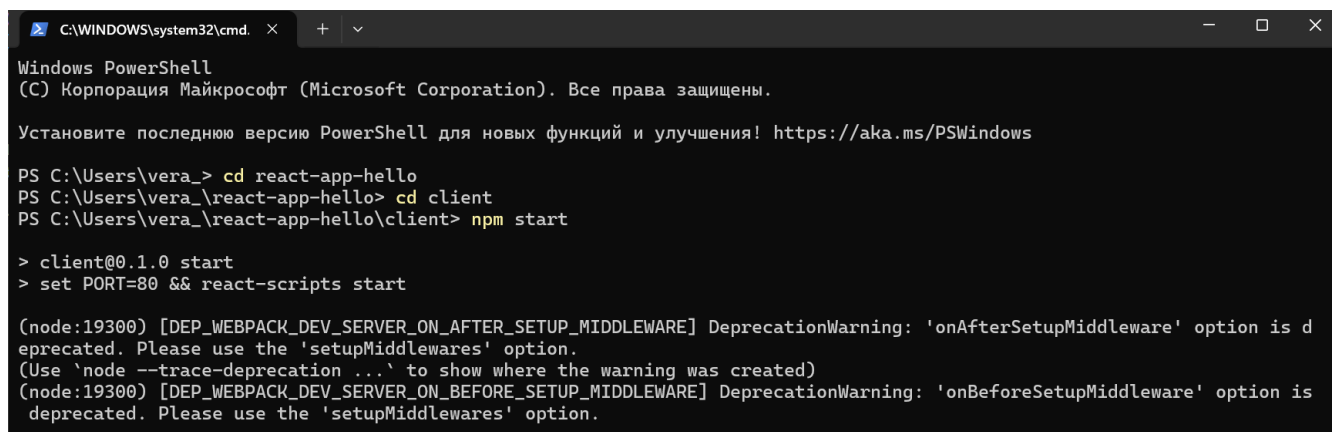
.App {
  text-align: center; /* Центрирование текста */
  font-size: 2em; /* Увеличение размера шрифта */
}

```

Листинг index.css

Продемонстрируем пример работы.

Процесс запуска микросервера:



```

C:\WINDOWS\system32\cmd. X + v
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Установите последнюю версию PowerShell для новых функций и улучшения! https://aka.ms/PSWindows

PS C:\Users\vera_> cd react-app-hello
PS C:\Users\vera_\react-app-hello> cd client
PS C:\Users\vera_\react-app-hello\client> npm start

> client@0.1.0 start
> set PORT=80 && react-scripts start

(node:19300) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is d
eprecated. Please use the 'setupMiddlewares' option.
(Use 'node --trace-deprecation ...' to show where the warning was created)
(node:19300) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is
deprecated. Please use the 'setupMiddlewares' option.

```

Рисунок 1 - процесс запуска через командную строку

```

Compiled successfully!

You can now view client in the browser.

  Local:            http://localhost:80
  On Your Network:  http://192.168.56.1:80

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully

```

Рисунок 2 - запуск через командную строку

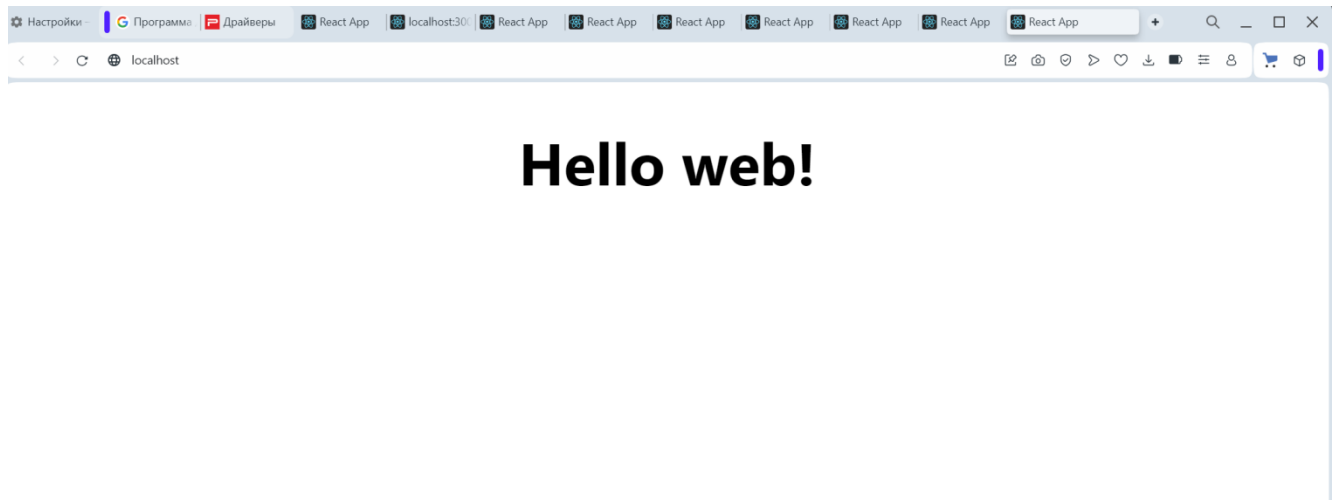


Рисунок 3 - результат в браузере

ЗАДАНИЕ 2. Микросервис Query

На примере данного микросервера подробно представим процесс создания React-приложения

1) Создание папки для проекта и переход в папку

```
mkdir react-app – query
```

```
cd react-app – query
```

2) Создание React-приложения

```
npx create-react-app client
```

```
cd client
```

3) Изменить файл src/App.js

4) Изменить строку в файле package.json (“set PORT = 80 && react-scripts start”)

```
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Установите последнюю версию PowerShell для новых функций и улучшения! https://aka.ms/PSWindows

PS C:\Users\vera_> mkdir react-app-query

Каталог: C:\Users\vera_

Mode                LastWriteTime         Length Name
----                -
d-----          19.11.2024   19:54                react-app-query

PS C:\Users\vera_> cd react-app-query
PS C:\Users\vera_\react-app-query> go mod init my-go-react-app
go: creating new go.mod: module my-go-react-app
go: to add module requirements and sums:
    go mod tidy
PS C:\Users\vera_\react-app-query> npx create-react-app client

Creating a new React app in C:\Users\vera_\react-app-query\client.

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

```

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

added 1314 packages in 53s

259 packages are looking for funding
  run `npm fund` for details
Git repo not initialized Error: Command failed: git --version
    at genericNodeError (node:internal/errors:983:15)
    at wrappedFn (node:internal/errors:537:14)
    at checkExecSyncError (node:child_process:888:11)
    at execSync (node:child_process:960:15)
    at tryGitInit (C:\Users\vera_\react-app-query\client\node_modules\react-scripts\scripts\init.js:46:5)
    at module.exports (C:\Users\vera_\react-app-query\client\node_modules\react-scripts\scripts\init.js:276:7)
    at [eval]:3:14
    at runScriptInThisContext (node:internal/vm:209:10)
    at node:internal/process/execution:118:14
    at [eval]-wrapper:6:24 {
  status: 1,
  signal: null,
  output: [ null, null, null ],
  pid: 14368,
  stdout: null,
  stderr: null
}

Installing template dependencies using npm...
└─
8 vulnerabilities (2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

Success! Created client at C:\Users\vera_\react-app-query\client
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd client
  npm start

Happy hacking!
PS C:\Users\vera_\react-app-query> |

```

Рисунок 4 - создание React-приложения

Необходимо реализовать пользовательский веб-интерфейс, чтобы при вводе имени в специальное поле сервер выводил приветственную фразу для данного пользователя.

```

package main

import (
    "fmt"
    "net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Access-Control-Allow-Origin", "*") // Разрешить все источники
    name := r.URL.Query().Get("name")
    ans := "Hello, " + name + "!"
    w.Write([]byte(ans))
}

func main() {
    http.HandleFunc("/api/user", handler)
    err := http.ListenAndServe(":8083", nil)
    if err != nil {
        fmt.Println("error")
    }
}

```

```
}  
}
```

Листинг server.go

```
import React, { useState } from 'react';  
import './App.css';  
  
function App() {  
  const [name, setName] = useState(''); // name: состояние для хранения введенного имени, setName: функция для  
  // обновления состояния name  
  const [message, setMessage] = useState(''); // message: состояние для хранения сообщения от сервера или  
  // сообщения об ошибке  
  
  //handleInputChange: Функция, которая вызывается при изменении значения в поле ввода.  
  // Она обновляет состояние name с текущим значением из поля ввода  
  const handleInputChange = (event) => {  
    setName(event.target.value);  
  };  
  // handleSubmit: Функция, которая вызывается при отправке формы  
  const handleSubmit = (event) => {  
    event.preventDefault(); // Предотвращаем перезагрузку страницы  
  
    if (name.trim() === '') {  
      setMessage('Пожалуйста, заполните поле.');      // Сообщение об ошибке  
      return; // Выходим из функции, если поле пустое  
    }  
    //fetch: Метод для выполнения HTTP-запроса на сервер  
    fetch(`http://localhost:8083/api/user?name=${encodeURIComponent(name)}`)  
      .then(response => {  
        if (!response.ok) {  
          throw new Error('Network response was not ok');  
        }  
        return response.text(); // Получаем текст ответа  
      })  
      .then(data => {  
        setMessage(data); // Устанавливаем сообщение из ответа  
      })  
      .catch(error => {  
        console.error('Ошибка при получении данных:', error);  
        setMessage('Ошибка при получении данных');  
      });  
  };  
  
  return (  
    <div className="App">  
      <h1>Приветствие</h1>  
      <form onSubmit={handleSubmit}>  
        <input  
          type="text"  
          value={name}  
          onChange={handleInputChange}  
          placeholder="Введите ваше имя"  
          required  
        />  
        <button type="submit">Отправить</button>  
      </form>  
      {message} && <h2>{message}</h2> { /* Отображаем ответ от сервера */ }  
    </div>  
  );  
}
```

```
export default App;
```

Листинг App.js

Приведем примеры работы

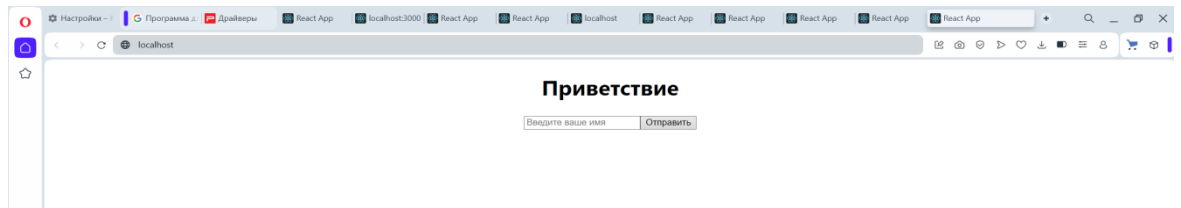


Рисунок 5 - вид формы

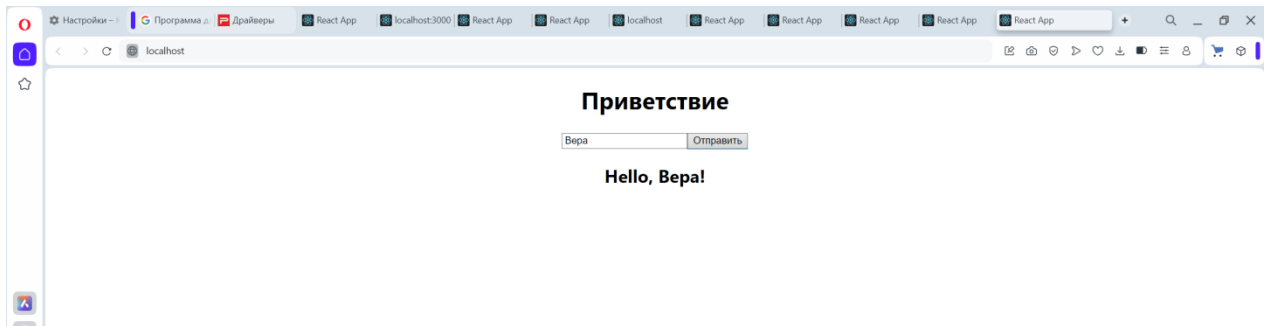


Рисунок 6 - работа с корректными данными

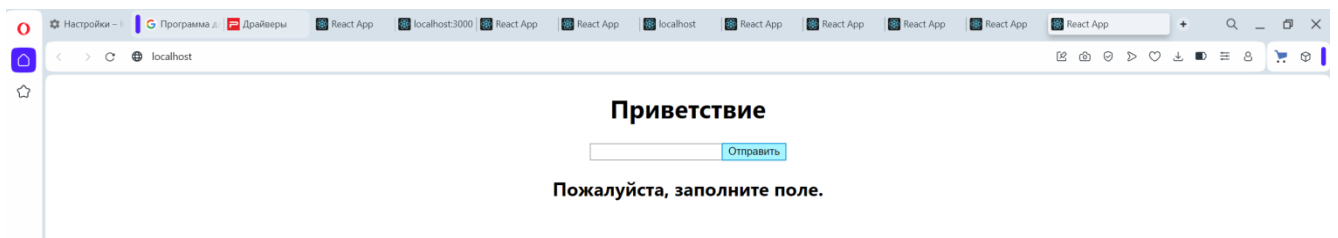


Рисунок 7 - работа с невведенным значением

ЗАДАНИЕ 3. Микросервис Count

Требуется создать пользовательский интерфейс, который будет отображать значение счетчика и позволит добавлять к нему введенное пользователем значение

Ниже приведены листинги

```
package main

import (
    "fmt"
    "net/http"
    "strconv"
)

var counter = 0

func handler(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Access-Control-Allow-Origin", "*")
    if r.Method == "GET" {
        w.Write([]byte(strconv.Itoa(counter)))
        return
    } else if r.Method == "POST" {
        r.ParseForm() // в POST передаются данные, собираем их в структуру, доступную через r.Form
        count := r.Form.Get("count") //получаем значение по ключу count
        if countInt, err := strconv.Atoi(count); err != nil {
            w.WriteHeader(http.StatusBadRequest)
            w.Write([]byte("Это не число"))
            return
        } else {
            counter += countInt
            w.WriteHeader(http.StatusOK)
            w.Write([]byte(strconv.Itoa(counter)))
            return
        }
    }
}
```

```

    } else {
        w.WriteHeader(http.StatusMethodNotAllowed)
        w.Write([]byte("этот метод не разрешен"))
        return
    }
}

func main() {
    http.HandleFunc("/count", handler)
    err := http.ListenAndServe(":8081", nil)
    if err != nil {
        fmt.Println("error")
    }
}

```

Листинг server.go

```

import React, { useState, useEffect } from 'react';
import './App.css';

function App() {
    const [counter, setCounter] = useState(0);
    const [inputValue, setInputValue] = useState('');
    const [errorMessage, setErrorMessage] = useState('');

    // Функция для получения текущего значения счетчика
    const fetchCounter = () => {
        fetch('http://localhost:8081/count')
            .then(response => {
                if (!response.ok) {
                    throw new Error('Ошибка при получении данных');
                }
                return response.text();
            })
            .then(data => {
                setCounter(parseInt(data)); // Устанавливаем текущее значение счетчика
                setErrorMessage(''); // Сбрасываем сообщение об ошибке
            })
            .catch(error => {
                console.error('Ошибка:', error);
                setErrorMessage('Ошибка при получении данных');
            });
    };

    // Получаем значение счетчика при загрузке компонента
    useEffect(() => {
        fetchCounter();
    }, []);

    // Обработчик изменения значения в поле ввода
    const handleInputChange = (event) => {
        setInputValue(event.target.value);
    };

    // Обработчик отправки формы
    const handleSubmit = (event) => {
        event.preventDefault(); // Предотвращаем перезагрузку страницы

        fetch('http://localhost:8081/count', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/x-www-form-urlencoded',
            },
            body: new URLSearchParams({ count: inputValue }), // Форматируем данные для отправки
        })
            .then(response => {
                if (!response.ok) {

```



```

        throw new Error('Ошибка при обновлении счетчика');
    }
    return response.text();
  })
  .then(data => {
    setCounter(parseInt(data)); // Устанавливаем новое значение счетчика
    setInputValue(''); // Очищаем поле ввода
    setErrorMessage(''); // Сбрасываем сообщение об ошибке
  })
  .catch(error => {
    console.error('Ошибка:', error);
    setErrorMessage('Ошибка при обновлении счетчика');
  });
});

return (
  <div className="App">
    <h1>Счетчик: {counter}</h1>
    <form onSubmit={handleSubmit}>
      <input
        type="number"
        value={inputValue}
        onChange={handleInputChange}
        placeholder="Введите число"
        required
      />
      <button type="submit">Добавить</button>
    </form>
    {errorMessage && <h2 style={{ color: 'red' }}>{errorMessage}</h2>} { /* Отображаем сообщение об ошибке */ }
  </div>
);
}

export default App;

```

Листинг App.js

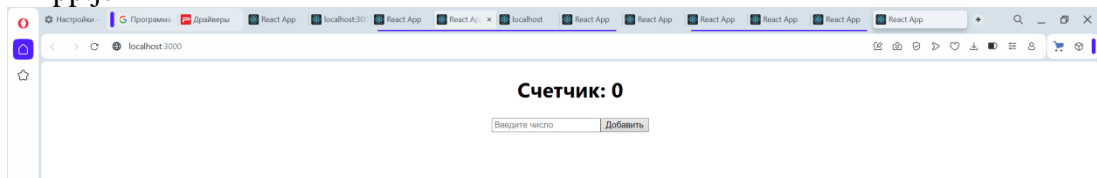


Рисунок 8 - вид формы

Счетчик: 109

Счетчик: 97

Рисунок 9 - результат работы

Выводы: В ходе данной лабораторной работы были реализованы 3 веб-интерфейса для взаимодействия с микросервисами, которые были получены в ходе выполнения предыдущей лабораторной работы. Был изучен фреймворк React, используемый для разработки фронтальной части веб-приложения.

Список использованных источников:

- 1) https://docs.google.com/presentation/d/1_7jB69j8u57FGMtli4f71GeS_TFgcz2mTYM_tlmbb-Y/edit#slide=id.p
- 2) <https://github.com/coreybutler/nvm-windows>
- 3) <https://create-react-app.dev/docs/getting-started/>