Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Разработка Интернет-Приложений»

Отчет по лабораторной работе №8

Выполнил:
студент группы ИУ5-53Б
Ветошкин Артём
Подпись и дата:
20.12.21

Проверил:
Юрий Евгеньевич Гапанюк
Подпись и дата:

Москва, 2021 г.

Цель лабораторной работы: изучение возможностей создания пользовательского интерфейса в веб-приложениях с использованием библиотеки React. З
адание:
Разработать React-приложение. Для создания приложения необходимо решить следующие задачи:
1. Создать стартовый React-проект. Удалить неиспользуемый код. Организовать директории для страниц, компонентов, утилит и работы с сетью.
2. Организовать роутниг в веб-приложении.
3. Разработать базовые страницы, на которых будут отображаться сущности из выбранной вами предметной области:
    i. Стартовая страница.
    ii. Страница просмотра списка объектов.
    iii. Страница просмотра конкретного объекта.
4. Вынести переиспользуемые компоненты в отдельные файлы:
    i. Для навигации по приложению можно добавить header.
    ii. Для отображения дополнительной информации (данные о студенте и предметной области) можно использовать footer.
    iii. Источники ввода-вывода (поля ввода (inputs)/формы/текстовые блоки).
    iv. Переиспользуемые таблицы/гриды.
5. Добавить асинхронные запросы в разработанный API, чтобы страница получала данные с сервера.
6. Если в Вашем проекте реализована сложная логика работы с состоянием приложения, то рекомендуется добавить пользовательские хуки.
7. Страницы приложения должны хорошо отображаться как на больших, так и на маленьких экранах.

Выполнение задания лабораторной работы

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import 'bootstrap/dist/css/bootstrap.min.css';
import { BrowserRouter } from 'react-router-dom';

const app = (
    <BrowserRouter>
        <App />
    </BrowserRouter>
);

ReactDOM.render(
    app,
    document.getElementById('root')
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

App.js

```
import './App.css';
import {Route, Routes, Navigate} from "react-router-dom";
import MainPage from "./pages/main_pages.js";
import StockPage from "./pages/stock_page.js";

function App() {
```

```
    return (
        <div className="App">
            <Routes>
                <Route path="/" element={<Navigate to={"/stocks"}/>}/>
                <Route exact path="/stocks" element={<MainPage/>}/>
                <Route path="/stock/:id" element={<StockPage/>}/>
            </Routes>
            <hr/>
        </div>
    );
}

export default App;
```

stock_info.js

```
import {Card} from "react-bootstrap";
import React from "react";

const StockInfo = (data) => {
    return <Card className="card">
        <Card.Body>
            <div className="textStyle">
                <Card.Title>{data.company_name}</Card.Title>
            </div>
            <div  className="textStyle">
                <Card.Text>
                    {data.price}
                </Card.Text>
            </div>
            <div  className="textStyle">
                <Card.Text>
                    {data.is_growing ?"Растёт":"Не растёт"}
                </Card.Text>
            </div>
            <div  className="textStyle">
                <Card.Text>
                    {data.date_modified}
                </Card.Text>
            </div>
        </Card.Body>
    </Card>
}

export default StockInfo;
```

info_field.js

```
import {Button} from "react-bootstrap";
import React from "react";

const InputField = ({ value, setValue, onSubmit, loading, placeholder,
buttonTitle = 'Поиск'}) => {
    return <div className="inputField">
        <input value={value} placeholder={placeholder} onChange={(event =>
setValue(event.target.value))}/>
        <Button disabled={loading} onClick={onSubmit}>{buttonTitle}</Button>
    </div>
}

export default InputField
```

stock_card.js

```
import {Button, Card} from "react-bootstrap";
import React from "react";
import {useNavigate} from "react-router-dom";
```

```
const StockCard = (data) => {
    const history = useNavigate();

    function to_project() {
        history(`/stock/${data.pk}/`);
    }

    return <Card className="card">
        <Card.Body>
            <div className="textStyle">
                <Card.Title>{data.company_name}</Card.Title>
            </div>
            <div  className="textStyle">
                <Card.Text>
                    {data.price}
                </Card.Text>
            </div>
            <Button className="cardButton" onClick={to_project}
target="_blank" variant="primary">Подробнее...</Button>
        </Card.Body>
    </Card>
}

export default StockCard;
```

api.js

```
export const getStocks = async () =>{
    return fetch(`http://localhost:8000/stocks/`)
        .then((response) => {
            return response.json();
        }).catch(() => {
            return []
        })
}

export const getStockById = async (id) =>{
    return await fetch(`http://localhost:8000/stocks/${id}/`)
        .then((response) => {
            return response.json();
        }).catch(() => {
            return {}
        })
}
```

huk.js

```
// eslint-disable-next-line no-unused-vars
import React, {useState, useEffect} from 'react';

// Пример пользовательского хука (Называйте пользовательские хуки с use в
начале!)
export default function useWindowSize() {
    // в данном пользовательском хуке мы используем хук состояния и хук
эффекта
    const [windowSize, setWindowSize] = useState({
        width: undefined,
        height: undefined,
    });
    useEffect(() => {
        // при вызове этой функции, мы будем "класть" в состояние актуальную
высоту и ширину экрана
        function handleResize() {
            setWindowSize({
                width: window.innerWidth,
                height: window.innerHeight,
```

```
            });
        }
        // В данном примере мы будет подписываться на изменение размеров
экрана, чтобы всегда иметь актуальные данные
        window.addEventListener("resize", handleResize);
        handleResize();
        // После того, как компонент "уничтожается", желательно избавиться от
всех "слушателей", чтобы не тратить ресурсы браузера
        return () => window.removeEventListener("resize", handleResize);
    }, []);

    return windowSize;
}
```

main_page.js

```
import React, {useEffect, useState} from 'react';
import {Col, Row, Spinner} from "react-bootstrap";
import StockCard from "../components/stock_card.js";
import {getStocks} from '../modules/api.js'
import useWindowSize from '../modules/huk.js'

function MainPage() {
    const [loading, setLoading] = useState(false)
    const [value, setValue] = useState([])

    useEffect(() => {
        (async () => {
            // Ставим загрузку
            await setLoading(true);
            const results = await getStocks();
            // Добавляем в состояние только треки
            await setValue(results);
            // Убираем загрузку
            await setLoading(false)
        })()
    }, [])

    const {width} = useWindowSize();
    const isMobile = width && width <= 600;

    return (
        <div className={`container ${loading && 'containerLoading'}`}>
            {loading && <div className="loadingBg"><Spinner
animation="border"/></div>}
            {!value.length ? <h1>К сожалению, пока ничего не найдено :(</h1>
:
                <Row xs={1} md={isMobile ? 1 : 4} className="g-4">
                    {value.map((item, index) => {
                        return isMobile ? <StockCard {...item} key={index}/>
: (
                            <Col key={index}>
                                <StockCard {...item}/>
                            </Col>
                        )
                    })}
                </Row>
            }
        </div>
    );
}

export default MainPage;
```

stock_page.js

```
import React, {useEffect, useState} from 'react';
import {Button, Spinner} from "react-bootstrap";
import StockInfo from "../components/stock_info.js";
import {getStockById} from '../modules/api.js'
import {useNavigate, useParams} from "react-router-dom";

function StockPage() {
    const history = useNavigate();
    const pk = useParams();

    const [loading, setLoading] = useState(false)
    const [value, setValue] = useState()

    function to_project() {
        history(`/stocks`);
    }

    useEffect(() => {
        (async () => {
            // Ставим загрузку
            await setLoading(true);
            const results = await getStockById(pk["id"]);
            // Добавляем в состояние только треки
            await setValue(results);
            // Убираем загрузку
            await setLoading(false)
        })()
    }, [pk])

    return (
        <>
        <Button className="cardButton" onClick={to_project} target="_blank"
variant="primary">Назад</Button>
        {loading && <div className="loadingBg"><Spinner
animation="border"/></div>}
        <StockInfo {...value}/>
        </>
    );
}

export default StockPage;
```

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on
a
      user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
```

```
      It will be replaced with the URL of the `public` folder during the
build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run
build`.
    -->
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```

main.js

```
import {MainPage} from "./pages/main_page.js";

document.addEventListener('DOMContentLoaded', () => {
    const root = document.getElementById('root');
    const mainPage = new MainPage(root)
    mainPage.render()
});
```

Пример работы программы: