**Лабораторная работа 4**

**Тема: Создание клиентского приложения**

**Глоссарий**

* **Node.js** - программная платформа, превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода.
* **React** (React.js или ReactJS) — JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. React разрабатывается и поддерживается Facebook, Instagram и сообществом отдельных разработчиков и корпораций. React может использоваться для разработки одностраничных и мобильных приложений. Его цель — предоставить высокую скорость, простоту и масштабируемость.
* **Create React App** —инструмент для быстрого старта React-приложений.

**Ход работы**

1. Установите платформу разработки node.js (<https://nodejs.org/en/>).
2. Откройте терминал (нажмите на клавиатуре сочетание клавиш Win+R, введите «cmd» и нажмите клавишу Enter).

Откройте в терминале папку, в которой будет расположена папка проекта.

Пример команды:

pushd F:\Development

1. Для разработки пользовательского интерфейса рекомендуется использовать JavaScript-библиотеку React.

Установите инструмент для быстрого старта React-приложений create-react-app с помощью команды:

npm install -g create-react-app

1. Создайте приложение с названием «deanery-ivbo-02-15-«номер бригады»:

create-react-app deanery-ivbo-02-15-«номер бригады»

Выполнение команды может занять несколько минут.

1. Перейдите в папку только что созданного проекта:

cd deanery-ivbo-02-15-«номер бригады»

1. Запустите приложение:

npm start

После запуска откроется браузер с вкладкой шаблонного приложения (рис 1). Если браузер не открылся, после завершения команды перейдите по адресу, указанному в терминале. В большинстве случаев это «http://localhost:3000/».



Рисунок 1. Шаблонное приложение

1. По указанному пути (F:\Development\deanery-ivbo-02-15-«номер бригады») было автоматически создано шаблонное приложение. В процессе разработки нас в первую очередь будет интересовать папка \src.

Откройте файл \src\App.js в текстовом редакторе (рекомендуется использовать Intellij Idea, рассмотренную в предыдущих лабораторных работах).

1. Подключите набор инструментов Bootstrap для стилизации приложения. Для этого в терминале введите команду:

npm install --save bootstrap react-bootstrap

Файл стилей необходимо импортировать в начало файла App.js

import "bootstrap/dist/css/bootstrap.css";

Следом импортируйте компоненты react-bootstrap, которые планируете использовать при вёрстке страницы. Полный перечень компонентов можно найти по ссылке: <https://react-bootstrap.github.io/components/alerts/>.

import { Navbar, Nav, Dropdown, Tabs, Tab } from "react-bootstrap";

1. Можно приступать к вёрстке.

Мы видим, что приложение содержит единственный компонент App. Его метод render() возвращает содержимое, которое мы наблюдаем на веб-странице http://localhost:3000/.

Для наглядности измените содержимое html-тегов внутри метода render() и посмотрите, как изменилась страница в браузере. Например, замените строку «Edit src/App.js and save to reload» на «My favorite app».

1. Создайте новый компонент JournalTable для отображения журнала группы.

class JournalTable extends Component {  
 render() {  
 return <Table bordered>  
 <thead>  
 <tr>  
 <th>#</th>  
 <th>ФИО</th>  
 <th>ПрИС</th>  
 <th>СИИ</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr>  
 <td>1</td>  
 <td>Иванов Иван Иванович</td>  
 <td>5</td>  
 <td>4</td>  
 </tr>  
 </tbody>  
 </Table>  
 }  
}

Чтобы отобразить новый компонент, измените компонент App так, чтобы он приобрёл следующий вид:

class App extends Component {

render() {

return (

<div className="App">

<JournalTable/>

</div>

);

}

}

1. Передайте данные о студентах компоненту JournalTable через свойства (props) и отобразите их. Пример:

class App extends Component {  
 render() {  
 return (

const students = [

{name: 'Иванов Иван Иванович', markPrIS: 5, markSII: 4},

{name: 'Петров Пётр Петрович', markPrIS: 3, markSII: 2}

];

<div className="App">  
 <JournalTable students={students}/>  
 </div>  
 );  
 }  
}  
  
class JournalTable extends Component {  
 render() {  
 return <Table bordered>  
 <thead>  
 <tr>  
 <th>#</th>  
 <th>ФИО</th>  
 <th>ПрИС</th>  
 <th>СИИ</th>  
 </tr>  
 </thead>  
 <tbody>  
 {this.props.students.map((student, index) => {  
 return <tr>  
 <td>{index + 1}</td>  
 <td>{student.name}</td>  
 <td>{student.markPrIS}</td>  
 <td>{student.markSII}</td>  
 </tr>  
 })}  
 </tbody>  
 </Table>  
 }  
}

На текущий момент страница должна приобрести вид, аналогичный рис 2:



1. Добавьте возможность выбора группы, журнал которой необходимо отобразить. В методе render() можно использовать данные из this.state (рис 3). Изменять такие данные можно с помощью метода setState React-компонента. Он изменяет состояние компонента, а также перезапускает render() для изменения интерфейса пользователя. Пример (изменился только компонент App):

class App extends Component {  
 constructor() {  
 super();  
 this.state = {  
 activeGroup: 0  
 };  
 }  
  
 render() {  
 const students = [  
 [

{name: 'Иванов Иван Иванович', markPrIS: 5, markSII: 4},

{name: 'Петров Пётр Петрович', markPrIS: 3, markSII: 2}

],

[

{name: 'Валиева Лидия Павловна', markPrIS: 3, markSII: 5},

{name: 'Илюшин Игорь Витальевич', markPrIS: 5, markSII: 5}

],

[

{name: 'Голутвин Семён Юрьевич', markPrIS: 4, markSII: 2},

{name: 'Сёмина Анна Николаевна', markPrIS: 3, markSII: 5}

]  
 ];  
  
 return (  
 <div className="App">  
 <JournalTable students={students[this.state.activeGroup]}/>  
 <button onClick={() => {  
 this.setState({activeGroup: 0});  
 }}>Группа 1</button>  
 <button onClick={() => {  
 this.setState({activeGroup: 1});  
 }}>Группа 2</button>  
 <button onClick={() => {  
 this.setState({activeGroup: 2});  
 }}>Группа 3</button>  
 </div>  
 );  
 }  
}

1. Создайте файл \src\Services\Request.js для хранения запросов к серверу.

Добавьте в него следующее содержимое:

const Request = {

getStudents() {

return fetch("/getStudents").then(res => res.json);

}

};

export default Request

Метод getStudents обращается к серверу по адресу /getStudents для получения данных о студентах, заполненных вами во 2-й лабораторной работе. Адрес запроса к серверу (/getStudents) замените на соответствующий, созданный Вами в 3-й лабораторной работе.

Импортируйте объект Request в файле App.js:

import Request from "./Services/Request"

Реализуйте получение данных о студентах с помощью запроса к серверу. Пример:

class App extends Component {

constructor() {

super();

this.state = {

activeGroup: 0,

students: null

};

}

componentDidMount() {

Request.getStudents().then((students) => {

this.setState({students: students});

});

}

render() {

return (

<div className="App">

{this.state.students ? <JournalTable students={this.state.students[this.state.activeGroup]}/> : null}

<button onClick={() => {

this.setState({activeGroup: 0});

}}>Группа 1</button>

<button onClick={() => {

this.setState({activeGroup: 1});

}}>Группа 2</button>

<button onClick={() => {

this.setState({activeGroup: 2});

}}>Группа 3</button>

</div>

);

}

}

Обратим внимание на метод componentDidMount(). Это один из этапов жизненного цикла компонентов. Данный метод будет вызван автоматически после отображения компонента на странице. Подробнее об этапах жизненного цикла компонентов можно узнать здесь: <https://reactjs.org/docs/state-and-lifecycle.html>

1. Соберите финальную версию проекта с помощью ввода в консоль команды

npm run build

После завершения команды содержимое папки /build необходимо расположить внутри серверной части приложения (лабораторная работа №3) по адресу /src/main/resources/static

1. Самостоятельно реализуйте в интерфейсе пользователя:
2. отображение типа итоговой аттестации для каждого из предметов в таблице (зачёт/экзамен),
3. выделение цветом ячеек с неудовлетворительной оценкой,
4. отображение количества пересдач