МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №6
по дисциплине «Web-технологии»
Тема: Модуль приложения «Покупка и продажа акций»

Студент гр.8303 _____ Рудько Д.Ю.
Преподаватель _____ Беляев С.А.

Санкт-Петербург
2020

Цель работы.

Изучить возможности применения библиотеки React (https://reactjs.org/) для разработки интерфейсов пользователя web-приложений.

Задание.

Необходимо создать web-приложение, обеспечивающее работу брокера, у него есть запас денежных средств, он имеет возможность купить или продать акции (любое доступное количество), а также контролировать изменение котировок акций. Брокеру должен отображаться баланс (запас денежных средств плюс стоимость акций), с которым он начал день, и текущее состояние.

Основные требования:

1. Приложение получает исходные данные из модуля администрирования приложения «Биржа акций» в виде настроек в формате JSON-файла.
2. В качестве сервера используется Node.JS с модулем express (либо ngnix в связке с PHP в качестве «back-end»).
3. Участники торгов подключаются к приложению «Покупка и продажа акций».
4. Предусмотрена HTML-страница администратора, на которой отображается перечень участников, для каждого участника отображается его баланс, количество акций каждого типа у каждого участника и количество, выставленное на торги.
5. Предусмотрена HTML-страница входа в приложение, где каждый участник указывает (или выбирает из допустимых) своё имя.
6. Предусмотрена HTML-страница, на которой участнику отображается общее количество доступных ему средств, количество и суммарная стоимость по каждому виду купленных акций. На ней же отображается количество выставленных на торги акций, их количество и стоимость. У участника есть возможность купить/продать интересующее его количество акций. Участнику отображается суммарный доход на начало торгов и на текущий момент времени.

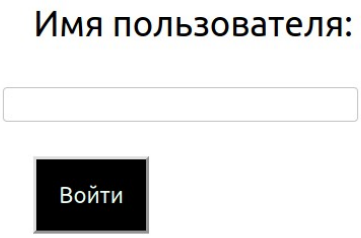
Для достижения поставленной цели требуется решить следующие задачи:

1) Разработка интерфейса web-приложения.
2) Создание web-сервера на основе express. Подготовка web-сокетов для обновления информации о стоимости у всех клиентов.
3) Создание каркаса web-приложения с использованием React.
4) Разработка перечня компонентов.
5) Создание статической версии интерфейса.
6) Определение минимального и достаточного набора состояний интерфейса.
7) Определение жизненного цикла состояний.
8) Программирование потока изменения состояний.

Ход выполнения.

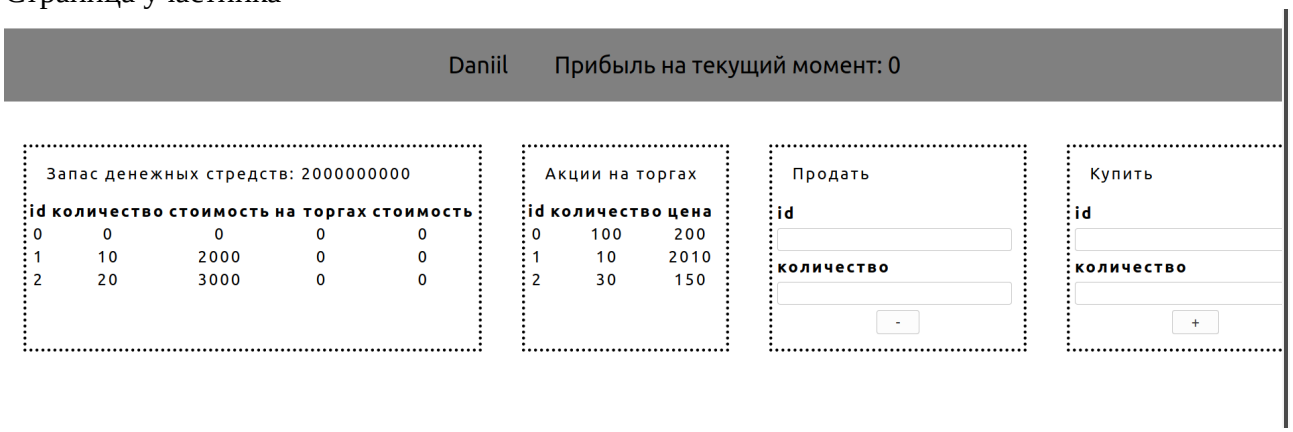
Создан web-сервер на основе express. Созданы шаблоны компонент страницы участников, админа и авторизации, реализован ввод из json файла. На сервере обработаны запросы на загрузку данных.

Страница входа



Страница администратора

Страница участника



Выводы.

В ходе лабораторной работы были изучены возможности применения библиотеки React для разработки интерфейсов пользователя web-приложений.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД

Сервер

www

```
#!/usr/bin/env node

/**
 * Module dependencies.
 */

var app = require('../app');
var debug = require('debug')('backend:server');
var http = require('http');

/**
 * Get port from environment and store in Express.
 */

var port = normalizePort(process.env.PORT || '3001');
//app.set('port', port);

/**
 * Create HTTP server.
 */

var server = http.createServer(app);

/**
 * Listen on provided port, on all network interfaces.
 */

server.listen(3001);
server.on('error', onError);
server.on('listening', onListening);

/**
 * Normalize a port into a number, string, or false.
 */

function normalizePort(val) {
  var port = parseInt(val, 10);

  if (isNaN(port)) {
    // named pipe
    return val;
  }

  if (port >= 0) {
    // port number
    return port;
  }

  return false;
}

/**
 * Event listener for HTTP server "error" event.
 */

function onError(error) {
  if (error.syscall !== 'listen') {
    throw error;
  }

  var bind = typeof port === 'string'
    ? 'Pipe ' + port
    : 'Port ' + port;

  // handle specific listen errors with friendly messages
  switch (error.code) {
    case 'EACCES':
      console.error(bind + ' requires elevated privileges');
```

```
      process.exit(1);
      break;
    case 'EADDRINUSE':
      console.error(bind + ' is already in use');
      process.exit(1);
      break;
    default:
      throw error;
  }
}

/**
 * Event listener for HTTP server "listening" event.
 */

function onListening() {
  var addr = server.address();
  var bind = typeof addr === 'string'
    ? 'pipe ' + addr
    : 'port ' + addr.port;
  debug('Listening on ' + bind);
}
```

app.js

```
const app = require('express')();
var path = require('path');
const bodyParser = require('body-parser');
const cors = require('cors');

var stocks = require('./json/stocks');
var persons = require('./json/person');

const corsOptions = {
    'credentials': true,
    'origin': true,
    'methods': 'GET,HEAD,PUT,PATCH,POST,DELETE',
    'allowedHeaders': 'Authorization,X-Requested-With,X-HTTP-Method-Override,
Content-Type, Cache-Control, Accept',
};

app.use(cors(corsOptions));
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());
app.use('/', require('./routes/routes'));

const server = require('http').createServer(app);

const io = require('socket.io')(server, {
    origins: "http://localhost:3000"
});

io.on('connection', function(socket) {

    socket.on('login', function(data) {
        for(let person of persons) {
            if(person.name === data.name || data.name === 'admin') {
                socket.json.emit('update', {stocks: stocks, persons: persons});
                socket.json.emit('welcome', {uname:data.name});

                return
            }
        }
        //socket.broadcast.json.emit('welcome', { uname: 'Nobody',
message:'Пользователь не найден'});
        socket.json.emit('welcome', { uname: 'Nobody', message:'Пользователь не
найден'});
    });

    socket.on('start', function(data) {
        socket.broadcast.json.emit('start_ex');
        socket.json.emit('start_ex');
        let timerId = setInterval(() => {
```

```
            for (let st of data.stocks) {
                if (st.distribution === 'равномерный'){
                    st.price = Math.round(st.st_price + ((Math.random()-
0.5)*st.max*2));
                } else {
                    st.price = Math.round(st.st_price + ((randNorm()-
0.5)*st.max*2));
                }
            }

            for (let br of persons) {
                for(let i = 0; i< data.stocks.length; i++) {
                    br.price[i] = br.stocks[i] * data.stocks[i].price;
                    br.ontorg_price[i] = br.ontorg_stocks[i] *
data.stocks[i].price;
                }
            }
            socket.json.emit('update', {stocks: data.stocks, persons: persons});
            socket.broadcast.json.emit('update', {stocks: data.stocks, persons:
persons});
        }, 10000)
    });

    socket.on('sell', function(data) {
        let ind = Number(data.index);
        let ct = Number(data.count);
        for (let br of persons) {
            if (br.name === data.name){
                br.stocks[ind] -= ct;
                br.ontorg_stocks[ind] += ct;
                br.price[ind] = br.stocks[ind] * stocks[ind].price;
                br.ontorg_price[ind] = br.ontorg_stocks[ind] *
stocks[ind].price;
                stocks[ind].in_torg +=ct;
                socket.json.emit('update', {stocks: stocks, persons: persons});
                socket.broadcast.json.emit('update', {stocks: stocks, persons:
persons});
            }
        }
    });

    socket.on('buy', function(data) {
        let ind = Number(data.index);
        let ct = Number(data.count);
        for (let br of persons) {
            if (br.name === data.name){
                br.money -= ct * stocks[ind].price;
                br.stocks[ind] += ct;
                br.price[ind] = br.stocks[ind] * stocks[ind].price;
                stocks[ind].in_torg -=ct;
                buy_stocks(ct, ind);
                socket.json.emit('update', {stocks: stocks, persons: persons});
                socket.broadcast.json.emit('update', {stocks: stocks, persons:
persons});
            }
        }
    });
});

function buy_stocks(count, index) {
    for (let br of persons) {
        if(br.ontorg_stocks[index] > 0) {
            let cur_count = br.ontorg_stocks[index] - count;
            if (cur_count < 0) {
                count = -cur_count;
                br.money += br.ontorg_stocks[index] * stocks[index].price;
                br.ontorg_stocks[index] = 0;
            }
            if (cur_count >= 0) {
                br.money += count * stocks[index].price;
                br.ontorg_stocks[index] -= count;
            }
```

```
            br.ontorg_price[index] = br.ontorg_stocks[index] *
stocks[index].price;
        }
    }
}

function randNorm() {
    let t = 0;
    let n = 3;
    for (let i = 0; i < n; ++i)
        t += Math.random();
    return t/n;
}

server.listen(8080);
```

routes.js

```
var express = require('express');
var router = express.Router();
const fs = require('fs');
var stocks = require('../json/stocks');
var person = require('../json/person');


router.get('/get_stocks', function(req,res,next) {
  res.json(stocks);
});

module.exports = router;
```

Клиент (React)

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(<App />, document.getElementById('root'));


serviceWorker.unregister();
```

App.js

```
 import React from 'react';
import Login from "./components/login";
import Admin from "./components/admin"
import User from "./components/user"
import * as io from 'socket.io-client'
import './App.css';

let socket;

class App extends React.Component{
  state = {
    username: 'Nobody',
    stocks: null,
    brokers: null,
    start_exchange: false,
    message: ''
  };

  login = (event) => {
    event.preventDefault();
    let name = event.target.elements.name.value;
    console.log(name);

    socket = io('http://localhost:8080');
    socket.on('connect',function(){
```

```
    socket.emit('login', {name:name});
  });

  socket.on('welcome', (data) => {
    this.setState({username: data.uname, message: data.message});
  });

  socket.on('update', (data) => {
    this.setState({
      stocks: data.stocks,
      brokers: data.persons,
    })
  });

  socket.on('start_ex', data =>{
    this.setState({
      start_exchange: true
    });
    console.log(this.state.username, this.state.start_exchange)
  });
};

start_torgs = () => {
  socket.emit('start', {stocks: this.state.stocks});
};

setS = (data) =>{
  this.setState({stocks: data});
}

sell = (name, index, count) => {
  socket.emit('sell', {name:name, index:index, count:count});
};

buy = (name, index, count) => {
  console.log(name, index, count);
  socket.emit('buy', {name:name, index:index, count:count});
};

render() {
  return (
      <div className="App">
        {this.get_content()}
      </div>
  );
}

get_content() {
  let content;
  if (this.state.username === 'Nobody')
    content = <Login
        login={this.login}
        message = {this.state.message}
    />;
  else if (this.state.username === 'admin') {
    content = <Admin
        start={this.state.start_exchange}
        brokers={this.state.brokers}
        stocks={this.state.stocks}
        start_torgs={this.start_torgs}
        setState={this.setS}
    />;
  }
  else
    content = <User
        start={this.state.start_exchange}
        brokers={this.state.brokers}
        username ={this.state.username}
        stocks ={this.state.stocks}
        sell ={this.sell}
        buy = {this.buy}
    />;
```

```
    return content;
  }

}

export default App;
```

admin.js

```
import React from 'react';
import "./admin.css"


export default class Admin extends React.Component{
    constructor(props){
        super(props);
        this.state = {
            stocks : this.props.stocks
        };
    }

    render() {
        return (
            <div className="Admin">
                <nav className="one">
                    <ul>
                        <li> Admin</li>
                        <li id="b" onClick={this.props.start_torgs}> Начать
торги</li>
                    </ul>
                </nav>

                <div className="stocks ad">
                    {this.get_stocks(this.props.stocks)}
                </div>

                <div className="brokers ad">
                    {this.get_brokers(this.props.brokers)}
                </div>
            </div>
        );
    }

        get_stocks(st){
            let stocks=[];
            let table = [];
            table.push(
                <tr>
                    <th>id</th>
                    <th>Количество</th>
                    <th>Цена</th>
                    <th>Закон распределения</th>
                </tr>
            )
            for(let i = 0; i < st.length; i++){
                table.push(
                    <tr>
                        <td>{st[i].id}</td>
                        <td>{st[i].in_torg}</td>
                        <td>{st[i].price}</td>
                        <td>{st[i].distribution}</td>
                        <button id='change' className={'change' +
this.props.start} value={i} onClick={this.distChange}>&#8634;</button>
                    </tr>
                )
            }
            stocks.push(<p>Акции:</p>)
            stocks.push(<table className="table">{table}</table>)
            return <div>{stocks}</div>
        }

        distChange = (event) => {
            console.log(event.target.value);
```

```
            let stocks = this.props.stocks;
            let law = stocks[event.target.value].distribution;
            if (law === 'нормальный') {
                law = 'равномерный';
            } else {
                law = 'нормальный';
            }
            stocks[event.target.value].distribution = law;
            this.setState(stocks);
            this.props.setState(stocks);
        };

        get_brokers(br){
            let brokers=[];
            for (let i = 0; i < br.length; i++){
                let broker = [];
                broker.push(<p> Имя: {br[i].name} </p>);
                broker.push(<p> Денежный счет: {br[i].money} </p>);
                let table = [];
                table.push(
                    <tr>
                        <th>id</th>
                        <th>количество</th>
                        <th>стоимость</th>
                    </tr>
                );
                for (let j = 0; j < br[i].stocks.length; j++){
                    table.push(
                        <tr>
                            <td>{j}</td>
                            <td>{br[i].stocks[j]}</td>
                            <td>{br[i].price[j]}</td>
                        </tr>
                    )
                }
                broker.push(<table className="tab">{table}</table>)
                brokers.push(<div className = "broker" key = {br[i].id}>
{broker}</div>)
            }
            return <div>{brokers}</div>
        }

}
```

login.js

```
import React from "react";
import "./login.css"

export default class Login extends React.Component{
    render() {
        return(
            <div className="Login">
                <form onSubmit={this.props.login}>
                    <p>Имя пользователя:</p>
                    <p id="warning">{this.props.message}</p>
                    <input name="name"/>
                    <p><button id="a">Войти</button> </p>
                </form>

            </div>
        );
    }
}
```

user.js

```
import React from 'react';
import "./user.css"

var pr = 0;

export default class User extends React.Component{
```

```
    constructor(props){
        super(props);
        this.state = {
            start: props.start_exchange,
            username: props.username,
            brokers: props.brokers,
            stocks: props.stocks,
            count: -1,
            index: -1,
            pr: 0
        };
    }

    render() {
        return (
            <div className="User">
                <nav className="two">
                    <ul>
                        <li>{this.props.username}</li>
                        <li>Прибыль на текущий момент: {pr}</li>
                    </ul>
                </nav>

                <div className="Ubrokers">
                    {this.get_info(this.props.username, this.props.brokers)}
                </div>
                <div className="torgs">
                    {this.get_torgs(this.props.stocks)}
                </div>
                <div className="UDeal">
                    <p> Продать </p>
                    <table className="Sell">
                        <tr>
                            <th>id</th>
                        </tr>
                        <tr>
                            <td><input onChange={this.get_index}/></td>
                        </tr>
                        <tr>
                            <th>количество</th>
                        </tr>
                        <tr>
                            <td><input onChange={this.get_count}/></td>
                        </tr>
                    </table>
                    <button className={"butDeal" + this.props.start}
onClick={this.sell}> - </button>
                </div>
                <div className="UDeal">
                    <p> Купить </p>
                    <table className="Buy">
                        <tr>
                            <th>id</th>
                        </tr>
                        <tr>
                            <td><input id="indexx" onChange={this.get_index}/></
td>
                        </tr>
                        <tr>
                            <th>количество</th>
                        </tr>
                        <tr>
                            <td><input onChange={this.get_count}/></td>
                        </tr>
                    </table>
                    <button className={"butDeal"+ this.props.start}
onClick={this.buy}> + </button>
                </div>
            </div>
        );
    }
```

```
    get_info(name, br) {
        for (let i = 0; i < br.length; i++) {
            if (br[i].name === name) {
                let broker = [];
                pr = br[i].money - br[i].start_money;
                broker.push(<p> Запас денежных стредств: {br[i].money} </p>);
                let table = [];
                table.push(
                    <tr>
                        <th>id</th>
                        <th>количество</th>
                        <th>стоимость</th>
                        <th>на торгах</th>
                        <th>стоимость</th>
                    </tr>
                );
                for (let j = 0; j < br[i].stocks.length; j++) {
                    table.push(
                        <tr>
                            <td>{j}</td>
                            <td>{br[i].stocks[j]}</td>
                            <td>{br[i].price[j]}</td>
                            <td> {br[i].ontorg_stocks[j]} </td>
                            <td>{br[i].ontorg_price[j]}</td>
                        </tr>
                    )
                }
                broker.push(<table>{table}</table>);
                return <div className="Ubroker">{broker}</div>
            }
        }
    }

    get_torgs(st) {
        let table = [];
        table.push(
            <tr>
                <th>id</th>
                <th>количество</th>
                <th>цена</th>
            </tr>
        );

        for (let i = 0; i < st.length; i++) {
            table.push(
                <tr>
                    <td>{i}</td>
                    <td>{st[i].in_torg}</td>
                    <td>{st[i].price}</td>
                </tr>
            )
        }
        return <div className="Ubroker" id="tt"><p> Акции на
торгах</p><table>{table}</table></div>
    }

    sell = (event) => {
        console.log(this.state.index , this.state.count, this.state.username);
        if(this.state.index > this.state.stocks.length) {
            console.log(this.state.index ,">", this.state.stocks.length);
            return;
        }
        let br = this.props.brokers;
        for (let i = 0; i < br.length; i++) {
            if (br[i].name === this.state.username) {
                if(this.state.count === -1  || this.state.index === -1 ||
this.state.count < 0 || this.state.count > br[i].stocks[this.state.index]) {
                    console.log("NO%%%%%%");
                    console.log(this.state.count)
                    console.log(br[i].stocks[this.state.index])
                    return;
                }
```

```
            }
        }
        this.props.sell(this.props.username, this.state.index, this.state.count,
this.props.brokers, this.props.stocks, this.state.pr);
    };

    buy = (event) => {
        console.log(event.target)
        console.log(this.state.count);
        let br = this.props.brokers;
        if(this.state.index > this.state.stocks.length) {
            console.log("NOT SUCCESS");
            console.log("index > stocks.length");
            return;
        }
        for (let i = 0; i < br.length; i++) {
            if (br[i].name === this.state.username) {
                if(this.state.count === -1  || this.state.index === -1 ||
this.state.count < 0 || this.state.count >
this.props.stocks[this.state.index].in_torg) {
                    // console.log(this.state.index);
                    // console.log(this.state.stocks[this.state.index].in_torg);
                    console.log("NOT SUCCESS");
                    return;
                }
                if(br[i].money < this.state.stocks[this.state.index].price *
this.state.count) {
                    return
                }
            }
        }
        console.log("SUCCESS");

        this.props.buy(this.props.username, this.state.index, this.state.count,
this.props.brokers, this.props.stocks, this.state.pr);

    };

    get_index = (event) => {
        this.setState({index: Number(event.target.value)});
    };

    get_count = (event) => {
        this.setState({count: Number(event.target.value)});
    };

}
```