ПРИЛОЖЕНИЕ

Листинг программы

Компонент index:

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './components/Layout/App.jsx';
import { BrowserRouter } from 'react-router-dom';

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

Компонент App:

```
import React, { Component } from 'react';
import './App.css';
import Content from './Content/index.jsx'
import Header from './Header/index.jsx'
import Footer from './Footer/index.jsx'
import CookieEventManager from '../../events.js'

class App extends Component {
  constructor(props) {
        super(props);
        CookieEventManager.initEvents();
  }

  render() {
    return (
        <>
          <Header/>
          <Content/>
          <Footer/>
      </>
    );
  }
}

export default App;
```

Компонент Header:

```
import React, { Component } from 'react';
import { Link } from "react-router-dom";
import { isUserAuthenticated, isCurRoleUser, isCurRoleAdmin } from '../../../utils.js'
import CookieEventManager from '../../../events.js'
import { getRandomNumber } from '../../../utils.js';

class Header extends Component {
    constructor(props) {
```

```
        super(props);
        this.state = {
            isCurUserLogged: isUserAuthenticated(),
            isCurRoleUser: isCurRoleUser(),
            isCurRoleAdmin: isCurRoleAdmin()
        };
        this.headerMenuUpdate = this.headerMenuUpdate.bind(this);
        CookieEventManager.addCookieChangedHandler(this.headerMenuUpdate);
    }
    headerMenuUpdate() {
        this.setState({ isCurUserLogged: isUserAuthenticated() });
        this.setState({ isCurRoleUser: isCurRoleUser() });
        this.setState({ isCurRoleAdmin: isCurRoleAdmin() });
    }
    render() {
        return (
            <header>
                <div className="container">
                    <h1 className="logo"><span>Z</span>IRO</h1>
                    {this.state.isCurRoleUser ? <UserMenu /> : null}
                    {this.state.isCurRoleAdmin ? <AdminMenu /> : null}
                    {!this.state.isCurUserLogged ? <LoginMenu /> : null}
                </div>
            </header>
        )
    }
}

class UserMenu extends Component {
    render() {
        const randomNumber = getRandomNumber(1, 5000);
        return (
            <React.Fragment>
                <div className="menu-wrap">
                    <ul className="menu">
                        <li className="menu__item" >
                            <Link to="/">Задачи</Link>
                        </li>
                        <li className="menu__item">
                            <Link to="/projects">Проекты</Link>
                        </li>
                        <li className="menu__item">
                            <Link to="/teams">Команды</Link>
                        </li>
                    </ul>
                </div>
                <div className="profile-wrap">
                    <Link to="/profile"><img className="short-img"
src={`/api/user/getAvatar?t=${randomNumber}`} alt="avatar" /></Link>
                </div>
            </React.Fragment>
        )
    }
}

class AdminMenu extends Component {
    render() {
        return (
            <div className="menu-wrap">
                <ul className="menu">
                    <li className="menu__item">
                        <Link to="/">Проекты</Link>
                    </li>
                    <li className="menu__item">
                        <Link to="/users">Пользователи</Link>
```

```
                </li>
                <li className="menu__item">
                    <Link to="/logout">Выйти</Link>
                </li>
            </ul>
        </div>
    )
  }
}

class LoginMenu extends Component {
  render() {
    return (
        <div className="menu-wrap">
            <ul className="menu">
                <li className="menu__item">
                    <Link to="/authorization">Авторизация</Link>
                </li>
            </ul>
        </div>
    )
  }
}
export default Header;
```

Компонент Content:

```
import React, { Component } from 'react';
import { Switch, Route } from 'react-router-dom'
import Tasks from '../../Tasks/index.jsx'
import Task from '../../Task/index.jsx'
import Projects from '../../Projects/index.jsx'
import Project from '../../Project/index.jsx'
import Teams from '../../Teams/index.jsx'
import Profile from '../../Profile/index.jsx'
import Authorization from '../../Authorization/index.jsx'
import Logout from '../../Logout/index.jsx'
import Users from '../../AdminPanel/Users.jsx'
import NotFound from '../../NotFound/index.jsx'
import { isCurRoleUser, isCurRoleAdmin } from '../../../utils.js'
import CookieEventManager from '../../../events.js'

class Content extends Component {
  constructor(props) {
    super(props);
    this.state = {
        isCurRoleUser: isCurRoleUser()
    };
    CookieEventManager.addCookieChangedHandler(this.setHomeComponent);
  }
  setHomeComponent = () => {
    this.setState({ isCurRoleUser: isCurRoleUser() });
  }
  render() {
    return (
        <main>
            <Switch>
                {this.state.isCurRoleUser ? <Route exact path="/" component={Tasks} />
                    : <Route exact path="/" component={Projects} />
                }
                <Route exact path="/" component={Tasks} />
                <Route path="/task/:task_number" component={Task} />
                <Route path="/projects" component={Projects} />
```

```
                    <Route path="/project/:short_name" component={Project} />
                    <Route path="/teams" component={Teams} />

                    <Route path="/profile" component={Profile} />
                    <Route path="/authorization" component={Authorization} />
                    <Route path="/logout" component={Logout} />
                    <Route path="/users" component={Users} />
                    <Route component={NotFound} />
                </Switch>
            </main>
        )
    }
}
export default Content;
```

Компонент Authorization:

```
import React, { Component } from 'react'
import PropTypes from 'prop-types';
import {indigo, blueGrey} from '@material-ui/core/colors';
import CookieEventManager  from '../../events.js'

class Authorization extends Component {
    constructor(props) {
        super(props);
        this.state = {
            email: 'testUser@mail.com',
          password: '523',
            showPassword: false
        };
        this.successLogin = this.successLogin.bind(this);
    }

    handleClickShowPassword = () => {
        this.setState(state => ({ showPassword: !state.showPassword }));
    };

    handleEmailChange(e) {
        this.setState({ email: e.target.value });
    }
    handlePasswordChange(e) {
        this.setState({ password: e.target.value });
    }

  successLogin(response) {
        this.setState({ password: '' });
        if (!response.errors) {
                setCookies(this.state.email, response.data.role);
                if (isCurRoleUser()) {
                        this.props.history.push('/');
                }
                else if (isCurRoleAdmin()) {
                        this.props.history.push('/adminpanel');
                }
                CookieEventManager.invokeCookieChanged();
        }
  }

    errorLogin(error) {
        alert(error);
    }

    handleClick = (e) => {
```

```jsx
            e.preventDefault();
            var requestData = {
                email: this.state.email,
                password: this.state.password
            };
        fetchPostData('api/account/login', requestData, this.successLogin,
this.errorLogin);
        }

        render() {
            const { classes } = this.props;
            return (
                <div className="container">
                    <Paper className={classes.paper}>
                        <Avatar className={classes.avatar}>
                            <LockOutlinedIcon />
                        </Avatar>
                        <Typography color="textPrimary" component="h1"
variant="h5">Вход</Typography>
                        <form className={`${classes.form} authorization-form`} action="">
                            <FormControl margin="normal" required fullWidth>
                                <InputLabel htmlFor="email">Email</InputLabel>
                                <Input
                                    id="email"
                                    type="email"
                                    name="email"
                                    autoComplete="email"
                                    autoFocus
                                    value={this.state.email}
                                    onChange={this.handleEmailChange.bind(this)}
                                />
                                <Typography className={classes.errorMessage} component="span"
variant="p">error</Typography>
                            </FormControl>
                            <FormControl margin="normal" required fullWidth>
                                <InputLabel htmlFor="password">Пароль</InputLabel>
                                <Input
                                    id="password"
                                    name="password"
                                    type={this.state.showPassword ? 'text' : 'password'}
                                    autoComplete="current-password"
                                    value={this.state.password}
                                    onChange={this.handlePasswordChange.bind(this)}
                                    endAdornment={
                                        <InputAdornment position="end">
                                          <IconButton
                                            aria-label="Toggle password visibility"
                                            onClick={this.handleClickShowPassword}
                                            className={classes.eye}
                                          >
                                            {this.state.showPassword ? <Visibility /> :
<VisibilityOff />}
                                          </IconButton>
                                        </InputAdornment>
                                    }
                                />
                                <Typography className={classes.errorMessage} component="span"
variant="p">error</Typography>
                            </FormControl>
                            <FormControlLabel className={classes.remember}
                                control={<Checkbox value="remember" color="primary" />}
                                label="Запомнить"
                            />
                                <Button variant="contained" color="primary"
onClick={this.handleClick} className={classes.submit}>Войти</Button>
```

```
                </form>
              </Paper>
            </div>
        )
      }
   }

   Authorization.propTypes = {
      classes: PropTypes.object.isRequired,
   };
   export default withStyles(styles)(Authorization);
```

Компонент Task:

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';

class Log extends Component {
   constructor(props) {
      super(props);
      this.state = {
         newLog: '',
         spentHours: 0
      }
   }

   onChangeInput = (e) => {
      this.setState({[event.target.name]: event.target.value})
   }

   onSuccsesAddedLog = (response) => {
      this.props.td_addLog(response.data);
  }
   onErrorAddedLog = (error) => {
      alert(error);
   }
   addLog = (e) => {
      e.preventDefault();
      var requestData = {
         taskId: this.props.td_id,
         text: this.state.newLog,
         spentHours: this.state.spentHours
      };
      if((this.state.newLog.length > 0) && (this.state.spentHours > 0)) {
         fetchPostData('api/task/addLogWork', requestData, this.onSuccsesAddedLog,
this.onErrorAddedLog);
         this.setState({newLog: ''})
         this.setState({spentHours: 0})
      } else {
         console.log('error');
      }
   }
   render() {
      return (
```

```jsx
            <div className="tab__content">
                {this.props.td_log.map((elem, index) =>
                    <div className="comment__item" key={index}>
                        <p className="comment__author"><img
src={`/api/user/getAvatar?userId=${elem.author.id}`} alt="avatar"
/>{elem.author.fullName}</p>
                        <p className="comment__text">{elem.text}</p>
                     <p className="comment__text">Заняло времени: {elem.spentTimeHours} ч.</p>
                        <p className="comment__date">{elem.leavingDate}</p>
                    </div>
                )}
                <div className="new-comment_wrap">
                    <textarea
                        name="newLog"
                        id="newLog"
                        placeholder="Добавить лог..."
                        value={this.state.newLog}
                        onChange={this.onChangeInput}
                        ></textarea>
                    <input
                        type="number"
                        name="spentHours"
                        placeholder="Затраченное время"
                        value={this.state.spentHours}
                        onChange={this.onChangeInput}
                        />
                    <Button onClick={this.addLog} variant="contained"
color="primary">Отправить</Button>
                </div>
            </div>
        )
    }
}

class Comments extends Component {
    constructor(props) {
        super(props);
        this.state = {
            newComment: ''
        }
    }

    onChangeInput = (e) => {
        console.log(e.target.value);
            this.setState({ newComment: e.target.value });
    }

    onSuccsesAddedComment = (response) => {
        this.props.td_addComment(response.data);
    }

    onErrorAddedComment = (error) => {
```

```
                alert(error);
            }

        addComment = (e) => {
            e.preventDefault();
            var requestData = {
                taskId: this.props.td_id,
                text: this.state.newComment
            };
            if(this.state.newComment.length > 0) {
                fetchPostData('api/task/addComment', requestData, this.onSuccsesAddedComment,
this.onErrorAddedComment);
                this.setState({newComment: ''})
            } else {
                console.log('error');
            }
        }

        render() {
            return (
                <div className="tab__content">
                    {this.props.td_comments.map((elem, index) =>
                        <div className="comment__item" key={index}>
                            <p className="comment__author"><img
src={`/api/user/getAvatar?userId=${elem.author.id}`} alt="avatar" />
{elem.author.fullName}</p>
                            <p className="comment__text">{elem.text}</p>
                            <p className="comment__date">{elem.leavingDate}</p>
                        </div>
                    )}
                    <div className="new-comment_wrap">
                        <textarea
                            name=""
                            id=""
                            placeholder="Добавить комментарий..."
                            value={this.state.newComment}
                            onChange={this.onChangeInput}
                        >
                        </textarea>
                        <Button onClick={this.addComment} variant="contained"
color="primary">Отправить</Button>
                    </div>
                </div>
            )
        }
    }

    class DeleteWin extends Component {
        constructor(props) {
            super(props);
            this.state = {
                //files: [],
```

```
        };
    }

    handleDeleteTask = () => {
        this.props.history.push('/');
        return;
    }

    render() {
        return (
            <Dialog
                onClose={this.handleClose}
         open={this.props.openDeleteDialog}
                maxWidth="xs"
            >
                <DialogContent>
                    <Typography variant="body2">Вы действительно хотите удалить
задачу?</Typography>
                </DialogContent>
                <DialogActions>
                    <Button onClick={this.props.handleCloseDeleteDialog}
color="default">Отмена</Button>
                    <Button onClick={this.handleDeleteTask} color="primary">Да</Button>
                </DialogActions>
            </Dialog>
        )
    }
}

class Task extends Component {
    constructor(props) {
        super(props);
        this.state = {
            taskDetails: {
                project: {},
                assignee: {},
                owner: {},
                comments: [{ author: {} }],
                logWorks: []
            },
            currentAssignee: {},
            isOwner: true, //test field
            isActiveStatus: false,
            value: 0,
            openEditDialog: false,
            openAssigneeDialog: false,
            openDeleteDialog: false,
        }
        if (!isUserAuthenticated()) {
            this.props.history.push('/authorization');
            return;
        }
```

```
        var requestData = {
            taskNumber: this.props.match.params.task_number
        };
        fetchPostData('api/task/getTaskDetailsByNumber', requestData,
this.successGetTaskData, this.errorGetTaskData);
    }

    successGetTaskData = (response) => {
        if (!response.errors) {
            this.setState({
                taskDetails: response.data,
                currentAssignee: response.data.assignee
            })
        }
    }

    errorGetTaskData = (error) => {
        alert(error);
    }

    toggleStatus = (e) => {
        var taskD = this.state.taskDetails;
        taskD.statusNum = e.currentTarget.getAttribute('statusNum');
        this.setState({ taskDetails: taskD });
    }

    tabChange = (event, value) => {
        this.setState({ value });
    };

    handleClickAssignee = () => {
        this.setState({ openAssigneeDialog: true });
    };

    handleCloseAssigneeDialog = currentAssignee => {
        this.setState({ currentAssignee, openAssigneeDialog: false });
    };
    handleCloseEditFormDialog = () => {
        this.setState({ openEditDialog: false });
    };
    handleCloseDeleteDialog = () => {
        this.setState({ openDeleteDialog: false });
    };

    addLog = (newLog) => {
        let newTaskDetails = this.state.taskDetails;
        newTaskDetails.logWorks.push(newLog);
        this.setState({ taskDetails: newTaskDetails})
    }
    addComment = (newComment) => {
        let newTaskDetails = this.state.taskDetails;
        newTaskDetails.comments.push(newComment);
```

```
                        this.setState({ taskDetails: newTaskDetails})
            }

        render() {
            const { classes } = this.props;
            var task_d = this.state.taskDetails;
            var { value } = this.state;
            return (
                <div className="container">
                    <div className="dialog-block">
                        {/* <AssigneeWin
                            open={this.state.openAssigneeDialog}
                            handleClose={this.handleCloseAssigneeDialog}
                            value={this.state.currentAssignee}
                        /> */}
                        <EditTaskForm
                            open={this.state.openEditDialog}
                            handleClose={this.handleCloseEditFormDialog}
                            taskData={this.state.taskDetails}
                        />
                        <DeleteWin
                            open={this.state.openDeleteDialog}
                            handleClose={this.handleCloseDeleteDialog}
                        />
                    </div>
                    <div className="task__wrap">
                        <Paper className="task__block">
                            <div className="task__section-wrap">
                                <h1 className="task__title"><span
className="task__number">{task_d.number}</span>{task_d.title}</h1>
                                <div className="task__section task__owner-block">
                                    <p className="task__owner-text">
                                        <span
className="task__creator">{task_d.owner.fullName}</span>открыл эту задачу
                                        <span className="task__creation-
date">{task_d.creationDate}</span> - <span className="task__comments-
number">{task_d.comments.length}</span> комментариев</p>
                                </div>

                                {this.state.isOwner ?
                                    <div className="task__section task__buttons-block">
                                        <Button variant="contained" className={`${classes.eventBtn}
event__btn`}>
                                            <Icon>edit_icon</Icon>
                                            Редактировать
                                            </Button>
                                        <Button variant="contained" className={`${classes.eventBtn}
event__btn`}
                                            onClick={this.handleClickAssignee}
                                        >
                                            <NavigationIcon />
                                            Назначить
```

```
                                          </Button>
                                        <Button variant="contained" className={`${classes.eventBtn}
event__btn`}>

                                          <DeleteIcon />
                                          Удалить
                                          </Button>
                                    </div>
                                    : null}

                            <div className="task__section task__details-block">
                                <h2 className="section__title">Детали</h2>
                                <p className="task__status"><span
className="key">Статус:</span> <span className="value">{task_d.status}</span></p>
                                <p className="task__type"><span className="key">Тип:</span>
<span className="value">{task_d.type}</span></p>
                                <p className="task__priority"><span
className="key">Приоритет:</span> <span className={`value
value${task_d.priorityNum}`}>{task_d.priority}</span></p>
                            </div>
                            <div className="task__section task__description-block">
                                <h2 className="section__title">Описание</h2>
                                <p className="task__description">{task_d.description}</p>
                                <img className="task__img" src="https://user-
images.githubusercontent.com/2376915/31859948-e42fd92e-b71b-11e7-9383-3459f8aa3af5.png" />
                            </div>
                        </div>
                        <div className="task__set-status-block">
                            <h2 className="section__title">Установить статус</h2>
                            <Button
                                className={`${classes.statusBtn} status__btn`}
                                variant={task_d.statusNum == 0 ? "contained" : "outlined"}
                                color={task_d.statusNum == 0 ? "primary" : null}
                                onClick={this.toggleStatus}
                                statusNum="0"  >
                                Открыто
                            </Button>
                            <Button
                                className={`${classes.statusBtn} status__btn`}
                                variant={task_d.statusNum == 1 ? "contained" : "outlined"}
                                color={task_d.statusNum == 1 ? "primary" : null}
                                onClick={this.toggleStatus}
                                statusNum="1">
                                Выполняется
                            </Button>
                            <Button
                                className={`${classes.statusBtn} status__btn`}
                                variant={task_d.statusNum == 2 ? "contained" : "outlined"}
                                color={task_d.statusNum == 2 ? "primary" : null}
                                onClick={this.toggleStatus}
                                statusNum="2"  >
                                Верификация
                            </Button>
```

```
                        <Button
                            className={`${classes.statusBtn} status__btn`}
                            variant={task_d.statusNum == 3 ? "contained" : "outlined"}
                            color={task_d.statusNum == 3 ? "primary" : null}
                            onClick={this.toggleStatus}
                            statusNum="3"      >
                            Тестирование
                        </Button>
                        <Button
                            className={`${classes.statusBtn} status__btn`}
                            variant={task_d.statusNum == 4 ? "contained" : "outlined"}
                            color={task_d.statusNum == 4 ? "primary" : null}
                            onClick={this.toggleStatus}
                            statusNum="4" >
                            Завершено
                        </Button>
                    </div>
                </Paper>
                <Paper className="comments__block">
                    <div className={classes.tabWrap}>
                        <Tabs
                            value={value}
                            onChange={this.tabChange}
                            variant="scrollable"
                            // scrollButtons="auto"
                            classes={{ root: classes.tabsRoot, indicator:
classes.tabsIndicator }}
                        >
                            <Tab
                                classes={{ root: classes.tabRoot, selected:
classes.tabSelected }}
                                label="Журнал работ" />
                            <Tab
                                classes={{ root: classes.tabRoot, selected:
classes.tabSelected }}
                                label="Комментарии" />
                        </Tabs>

                        {value === 0 && <Log
                                        td_log={task_d.logWorks}
                                        td_id={task_d.id}
                                        td_number={task_d.number}
                                        td_addLog={this.addLog}
                                        />}
                        {value === 1 && <Comments
                                        td_comments={task_d.comments}
                                        td_id={task_d.id}
                                        td_number={task_d.number}
                                        td_addComment={this.addComment}
                                        />}
                    </div>
                </Paper>
```

```
            </div>
        </div>
    )
}
}
Task.propTypes = {
    classes: PropTypes.object.isRequired,
};
export default withStyles(styles)(Task)
```

Компонент Projects:

```
import React, { Component } from 'react'
import { Link } from "react-router-dom"

const styles = theme => ({
    paper: {
        'max-width': '900px',
        marginLeft: 'auto',
        marginRight: 'auto',
        padding: `${theme.spacing.unit * 4}px ${theme.spacing.unit * 5}px
${theme.spacing.unit * 5}px`,
    },
    search: {
        width: '60%',
    },
    addBtn: {
        color: '#fff',
        backgroundColor: green[600],
        '&:hover': {
            backgroundColor: green[400],
        },
    }
});

class Projects extends Component {
    constructor(props) {
        super(props);
        this.state = {
            projects: [],
            isCurRoleAdmin: false,
        }
        if (!isUserAuthenticated()) {
            this.props.history.push('/authorization');
            return;
        }
        fetchGetData('api/project/getCurrentProjectsInfos', this.successGetData,
this.errorGetData);
    }

    successGetData = (response) => {
        var projects = response.data.projects;
        if (!response.errors) {
            this.setState({
                projects: projects
            })
        }
    }

    errorGetData = (error) => {
        alert(error);
```

```jsx
    }

    render() {
        const { classes } = this.props;
        return (
            <div className="container">
                <div className="search-block">
                    <TextField
                        id="outlined-search"
                        label="Поиск проекта..."
                        type="search"
                        className={classes.search}
                        margin="normal"
                        variant="outlined"
                    />
                    {this.state.isCurRoleAdmin ?
                        <div className="buttons-block">
                            <Button variant="contained" className={classes.addBtn}>
                                <AddIcon />
                                Создать проект
                            </Button>
                        </div>
                        : null}
                </div>

                <Paper className="projects-block">
                    <div className="projects-header">
                        <div className="item__col">
                            <p>Название</p>
                        </div>
                        <div className="item__col">
                            <p>Краткое название</p>
                        </div>
                        <div className="item__col">
                            <p>Описание</p>
                        </div>
                        <div className="item__col">
                            <p>Текущие задачи</p>
                        </div>
                        <div className="item__col">
                            <p>Человек на проекте</p>
                        </div>
                    </div>

                    <div className="projects-list">
                        {this.state.projects.map((proj, index) =>
                            <Link to={`/project/${proj.shortName}`} className="project__item"
key={proj.id}>
                                <div className="item__col">
                                    <p className="project__title">{proj.name}</p>
                                </div>
                                <div className="item__col">
                                    <p className="project__short-title">{proj.shortName}</p>
                                </div>
                                <div className="item__col">
                                    <p className="project__description">{proj.description}</p>
                                </div>
                                <div className="item__col">
                                    <p
className="project__count">{proj.nonClosedTasksCount}</p>
                                </div>
                                <div className="item__col">
                                    <p className="project__count">{proj.totalUsersCount}</p>
                                </div>
                            </Link>
```

```
                    )}
                </div>

            </Paper>
        </div>
    )
  }
}

Projects.propTypes = {
    classes: PropTypes.object.isRequired,
};
export default withStyles(styles)(Projects);
```

Компонент Profile:

```jsx
import React, { Component } from 'react';
import PropTypes from 'prop-types';
import Paper from '@material-ui/core/Paper';
import Typography from '@material-ui/core/Typography';
import Button from '@material-ui/core/Button';
import Icon from '@material-ui/core/Icon';
import withStyles from '@material-ui/core/styles/withStyles';
import './profile.css';
import { isUserAuthenticated, fetchGetData, getRandomNumber } from '../../utils.js'
import EditForm from './ProfileEditForm.jsx'

const styles = theme => ({
    paper: {
        ' max-width': '900px',
        marginLeft: 'auto',
        marginRight: 'auto',
        //spacing.unit = 8
        padding: `${theme.spacing.unit * 4}px ${theme.spacing.unit * 5}px ${theme.spacing.unit * 5}px`,
    },
    ava__wrap: {
        width: '150px',
        height: '150px',
    },
    ava: {
        width: '100%',
        borderRadius: '50%',
    },
    editBtn: {
        marginTop: '10px'
    }
});

class Profile extends Component {
    constructor(props) {
        super(props);
        if (!isUserAuthenticated()) {
            this.props.history.push('/authorization');
            return;
        }
        this.state = {
            isEditFormVisible: false,
            name: null,
            lastName: null,
            email: null,
            skype: null,
            phoneNumber: null,
```

```jsx
            position: null,
            dateOfBirth: null,
            data: null,
        };
        this.editFormSetVisibility = this.editFormSetVisibility.bind(this);
        this.successGetData = this.successGetData.bind(this);
        this.errorGetData = this.errorGetData.bind(this);
        fetchGetData('api/user/getProfile', this.successGetData, this.errorGetData);
    }

    successGetData(response){
        var data = response.data;
        if (!response.errors) {
            this.setState({
                name: data.name,
                lastName: data.lastName,
                email: data.email,
                skype: data.skype,
                phoneNumber: data.phoneNumber,
                position: data.position.name,
                dateOfBirth: data.dateOfBirth,
                data: data
            })
        }
    }

    errorGetData(error) {
        alert(error);
    }

    editFormSetVisibility(e) {
        e.preventDefault();
        this.setState({ isEditFormVisible: !this.state.isEditFormVisible });
    }

    logout = () => {
        this.props.history.push('/logout');
    }

    render() {
        const { classes } = this.props;
        const randomNumber = getRandomNumber(1, 5000);
        return (
            <div className="container">
                <Paper className={`${classes.paper} profile__wrap`}>
                    <div className="profile__image-block col">
                        <div className={classes.ava__wrap}>
                            <img className={classes.ava}
src={`/api/user/getAvatar?t=${randomNumber}`} alt="user avatar" />
                        </div>
                        <Button variant="contained" color="primary"
className={`${classes.editBtn} profile-edit__btn`} onClick={this.editFormSetVisibility}>
                        {this.state.isEditFormVisible ? 'Отмена' : 'Редактировать'}
                            <Icon>edit_icon</Icon>
                        </Button>
                    </div>
                    <div className="profile__info-block col">
                        <Typography className="profile__name" color="textPrimary"
component="h1" variant="h5">{this.state.lastName} {this.state.name}</Typography>
                        <Typography className="profile__position" color="textPrimary"
component="h2" variant="h6">{this.state.position}</Typography>
                        <br />
                        <p className="personal-data">email: {this.state.email}</p>
                        <p className="personal-data">skype: {this.state.skype}</p>
```

```jsx
                    <p className="personal-data">Номер телефона:
{this.state.phoneNumber}</p>
                    <p className="personal-data">Дата рождения:
{this.state.dateOfBirth}</p>

                    <p className="personal-data">Technology and skills: <span>C#,
ASP.NET, DB, javascript</span></p>
                    <p className="personal-data">Current projects: <span>Ziro, Machine
learning startap</span></p>
                </div>
                <div className="profile__exit col">
                    <Button variant="contained" color="primary" onClick={this.logout}>
                        Выйти
                            <Icon>open_in_new</Icon>
                    </Button>
                </div>
            </Paper>

            {this.state.isEditFormVisible ? <EditForm data={this.state.data}/> : null}

        </div>
        );
    }
}

Profile.propTypes = {
    classes: PropTypes.object.isRequired,
};
export default withStyles(styles)(Profile);
```