Компонент отвечающий за маршрутизацию приложения.

export default function AppRoutes(props) {

return (

<div>

<NavigationBar setIsAuthorized={props.setIsAuthorized} />

<Switch>

<Route path='/' exact>

<BoardPage validateFetchResponse={props.validateFetchResponse} />

</Route>

<Route path='/team' exact>

<TeamPage validateFetchResponse={props.validateFetchResponse} />

</Route>

</Switch>

</div>

);

}

Пример одной из главных страниц приложения.

function BoardPage({ projects }) {

const dispatch = useDispatch();

const classes = useStyles();

return (

<div>

<div className={classes.projectToolsBar}>

<Toolbar />

</div>

<Board />

</div>

)

}

const useStyles = makeStyles((theme) => ({

projectToolsBar: {

marginTop: '1%',

}

}));

export default connect(

({ project }) => ({ projects: project.projects }),

{

}

)(BoardPage)

Пример компонента авторизации.  
  
export default function SignIn(props) {

const { enqueueSnackbar } = useSnackbar();

const classes = useStyles();

const history = useHistory();

const dispatch = useDispatch();

const [email, setEmail] = useState(null);

const [password, setPassword] = useState(null);

const login = async () => {

if (!validationHelper.isStringValid(email) || !validationHelper.isStringValid(password)) {

enqueueSnackbar('User Email or Password is empty!', { variant: 'error' });

}

let body = JSON.stringify(

{

'email': email,

'password': password

});

let userIdentity = await api.fetchPost('account/signIn', localStorage.token, body)

localStorage[localStorageConstants.token] = userIdentity.token;

localStorage[localStorageConstants.userName] = userIdentity.userName;

localStorage[localStorageConstants.userRole] = userIdentity.userRole;

localStorage[localStorageConstants.userId] = userIdentity.userId;

dispatch({ type: authorizationActionTypes.SET\_AUTHORIZE, payload: true });

history.push('/');

}

return (

<Container component="main" maxWidth="xs">

<CssBaseline />

<div className={classes.paper}>

<Avatar sx={{ m: 1, bgcolor: '#f50057' }}>

<LockOutlinedIcon />

</Avatar>

<Typography component="h1" variant="h5">

Sign in

</Typography>

<div className={classes.form} noValidate>

<TextField

variant="outlined"

margin="normal"

required

fullWidth

id="email"

label="Email"

name="email"

autoComplete="email"

autoFocus

value={email}

onChange={e => setEmail(e.target.value)}

/>

<TextField

variant="outlined"

margin="normal"

required

fullWidth

name="password"

label="Password"

type="password"

id="password"

autoComplete="current-password"

value={password}

onChange={e => setPassword(e.target.value)}

/>

<FormControlLabel

control={<Checkbox value="remember" color="primary" />}

label="Remember me"

/>

<Button

type="submit"

fullWidth

variant="contained"

color="primary"

className={classes.submit}

onClick={login}

>

Sign In

</Button>

<div className={classes.labels}>

<Grid container>

<Grid item xs>

<Link href="#" variant="body2">

Forgot password?

</Link>

</Grid>

<Grid item>

<Link to="signUp" variant="body2">

{"Don't have an account? Sign Up"}

</Link>

</Grid>

</Grid>

</div>

</div>

</div>

<Box mt={8}>

<Copyright />

</Box>

</Container>

);

}

const useStyles = makeStyles((theme) => ({

paper: {

marginTop: theme.spacing(8),

display: 'flex',

flexDirection: 'column',

alignItems: 'center',

},

avatar: {

margin: theme.spacing(1),

backgroundColor: theme.palette.secondary.main,

},

form: {

width: '100%',

marginTop: theme.spacing(1),

},

labels: {

marginTop: 10,

},

submit: {

margin: theme.spacing(3, 0, 2),

},

}));

Пример компонента, являющимся одним из инструментов приложения

function DeleteProjectTool({ selectedProject }) {

const dispatch = useDispatch();

const [isConfirmationOpen, setIsConfirmationOpen] = useState(false)

const deleteSelectedProject = () => {

dispatch(projectActions.deleteProject(selectedProject.id, localStorage.token));

}

if (selectedProject == null) {

return (

<div />

);

}

return (

<div>

<Button

variant="contained"

style={{

backgroundColor: "#d32f2f"

}}

size='large'

startIcon={<DeleteIcon />}

onClick={() => setIsConfirmationOpen(true)}

>

Delete

</Button>

<ProjectDeleteConfirmation isOpen={isConfirmationOpen} setIsOpen={setIsConfirmationOpen} deleteSelectedProject={deleteSelectedProject}/>

</div>

);

}

export default connect(

({ project }) => ({ selectedProject: project.selectedProject }),

{

}

)(DeleteProjectTool);