

```
1: /*
2:  *
3:  * University di Pisa - Master's Degree in Computer Science and Networking
4:  *
5:  * Final Project for the course of Peer to Peer Systems and Blockchains
6:  *
7:  * Teacher: Prof. Laura Ricci
8:  *
9:  * Candidate: Orlando Leombruni, matricola 475727
10:  *
11:  * File: index.js
12:  *
13:  * "Main" application script. It also creates a Material UI palette and applies it
to the webapp.
14:  *
15:  */
16:
17: import React from 'react';
18: import ReactDOM from 'react-dom';
19: import { HashRouter, Route, Switch } from 'react-router-dom';
20: import { MuiThemeProvider, createMuiTheme } from '@material-ui/core/styles';
21: import { blue, orange } from '@material-ui/core/colors';
22: import App from './components/App';
23: import './styles/global.css';
24: import 'typeface-roboto';
25:
26: const mTheme = createMuiTheme({
27:   palette: {
28:     primary: blue,
29:     secondary: orange,
30:   },
31:   typography: {
32:     useNextVariants: true,
33:   }
34: });
35:
36: ReactDOM.render(
37:   <MuiThemeProvider theme={mTheme}>
38:     <HashRouter>
39:       <Switch>
40:         <Route exact path="/catalog" render={(props) => <App {...props} isCa
talog={true} /> } />
41:         <Route render={(props) => <App {...props} isCatalog={false} /> } />
42:       </Switch>
43:     </HashRouter>
44:   </MuiThemeProvider>,
45:   document.getElementById('root')
46: );
```



```

1: /*
2:  *
3:  * University di Pisa - Master's Degree in Computer Science and Networking
4:  *
5:  * Final Project for the course of Peer to Peer Systems and Blockchains
6:  *
7:  * Teacher: Prof. Laura Ricci
8:  *
9:  * Candidate: Orlando Leombruni, matricola 475727
10:  *
11:  * File: MaterialCustomStyles.js
12:  *
13:  * Contains all the custom styles for each React Component, using the Material UI f
framework.
14:  *
15:  */
16:
17: import {blue, green, red} from "@material-ui/core/colors";
18: import {fade} from "@material-ui/core/styles/colorManipulator";
19:
20: export const AppStyle = theme => ({
21:   toolbar: theme.mixins.toolbar,
22:   centered: {
23:     display: "flex",
24:     flexDirection: "column",
25:     justifyContent: "center",
26:     alignItems: "center",
27:   },
28: });
29:
30: export const AppDrawerStyle = theme => {
31:   const drawerWidth = 240;
32:   return ({
33:     root: {
34:       display: 'flex',
35:     },
36:     mask: {
37:       display: 'flex',
38:       width: '100%',
39:       height: '100%',
40:     },
41:     appBar: {
42:       zIndex: theme.zIndex.drawer + 1,
43:       transition: theme.transitions.create(['width', 'margin'], {
44:         easing: theme.transitions.easing.sharp,
45:         duration: theme.transitions.duration.leavingScreen,
46:       }),
47:     },
48:     appBarShift: {
49:       marginLeft: drawerWidth,
50:       width: `calc(100% - ${drawerWidth}px)`,
51:       transition: theme.transitions.create(['width', 'margin'], {
52:         easing: theme.transitions.easing.sharp,
53:         duration: theme.transitions.duration.enteringScreen,
54:       }),
55:     },
56:     menuButton: {
57:       marginLeft: 12,
58:       marginRight: 36,
59:     },
60:     hide: {
61:       display: 'none',
62:     },
63:     drawer: {
64:       width: drawerWidth,
65:       flexShrink: 0,

```

```

66:       whiteSpace: 'nowrap',
67:     },
68:     drawerOpen: {
69:       width: drawerWidth,
70:       transition: theme.transitions.create('width', {
71:         easing: theme.transitions.easing.sharp,
72:         duration: theme.transitions.duration.enteringScreen,
73:       }),
74:     },
75:     drawerClose: {
76:       transition: theme.transitions.create('width', {
77:         easing: theme.transitions.easing.sharp,
78:         duration: theme.transitions.duration.leavingScreen,
79:       }),
80:       overflowX: 'hidden',
81:       width: theme.spacing.unit * 7 + 1,
82:       [theme.breakpoints.up('sm')]: {
83:         width: theme.spacing.unit * 9 + 1,
84:       },
85:     },
86:     toolbar: {
87:       display: 'flex',
88:       alignItems: 'center',
89:       justifyContent: 'flex-end',
90:       padding: '0 8px',
91:       ...theme.mixins.toolbar,
92:     },
93:     content: {
94:       flexGrow: 1,
95:       padding: theme.spacing.unit * 3,
96:     },
97:     linearProgress: {
98:       position: 'absolute',
99:       left: 0,
100:      zIndex: 1400,
101:      width: '100%',
102:      top: 0
103:    },
104:    darken: {
105:      backgroundColor: "rgba(0, 0, 0, 0.5)",
106:      position: 'fixed',
107:      zIndex: 1300,
108:      left: 0,
109:      right: 0,
110:      bottom: 0,
111:      top: 5,
112:    },
113:    grow: {
114:      flexGrow: 1,
115:    },
116:    logo: {
117:      maxHeight: 0,
118:      "@media (min-width:450px)": {
119:        maxHeight: theme.mixins.toolbar.minHeight - (theme.spacing.unit * 4),
120:      }
121:    },
122:    rightGutter: {
123:      paddingRight: theme.mixins.gutters().paddingRight,
124:      "@media (min-width:600px)": {
125:        paddingRight: theme.mixins.gutters() ["@media (min-width:600px)"].padding
Right,
126:      }
127:    }
128:  });
129:
130: export const BuyGiftDialogStyle = theme => ({

```

```
131:   formControl: {
132:     margin: theme.spacing.unit * 3,
133:   },
134:   group: {
135:     margin: `${theme.spacing.unit}px 0`,
136:   },
137:   root: {
138:     display: 'flex',
139:   },
140:   textField: {
141:     margin: theme.spacing.unit * 2,
142:     width: 200,
143:   },
144:   root2: {
145:     flexDirection: 'column',
146:     justifyContent: 'center',
147:     alignItems: 'center',
148:     padding: 0,
149:   }
150: });
151:
152: export const BuyGiftPremiumDialogStyle = theme => ({
153:   root: {
154:     display: 'flex',
155:   },
156:   textField: {
157:     margin: theme.spacing.unit,
158:     width: "80%",
159:   }
160: });
161:
162: export const CatalogManagerStyle = theme => ({
163:   typography: {
164:     padding: theme.spacing.unit,
165:   },
166:   holder: {
167:     display: 'flex',
168:     justifyContent: 'center',
169:   },
170:   warn: {
171:     color: theme.palette.getContrastText(red['A700']),
172:     backgroundColor: red['A700'],
173:     '&:hover': {
174:       backgroundColor: red[900],
175:     }
176:   },
177:   extendedIcon: {
178:     marginRight: theme.spacing.unit,
179:   },
180:   button: {
181:     margin: theme.spacing.unit,
182:   },
183:   elements: {
184:     display: 'flex',
185:     alignItems: 'center',
186:     padding: theme.spacing.unit,
187:   },
188:   grow: {
189:     flexGrow: 1,
190:   }
191: });
192:
193: export const ChartsStyle = theme => ({
194:   root: {
195:     flexGrow: 1,
196:   },
```

```
197:   avatarRed: {
198:     backgroundColor: red[500],
199:   },
200:   avatarGreen: {
201:     backgroundColor: green[500],
202:   },
203:   avatarBlue: {
204:     backgroundColor: blue[500],
205:   },
206:   card: {
207:     padding: theme.spacing.unit * 2,
208:   },
209:   button: {
210:     margin: theme.spacing.unit,
211:   },
212: });
213:
214: export const ContentListStyle = theme => ({
215:   root: {
216:     width: "100%",
217:   },
218:   heading: {
219:     fontSize: theme.typography.pxToRem(15),
220:     fontWeight: theme.typography.fontWeightRegular,
221:   },
222:   details: {
223:     alignItems: "center",
224:   }
225: });
226:
227: export const CreatorAppStyle = theme => ({
228:   notification: {
229:     color: theme.palette.text.secondary,
230:     backgroundColor: theme.palette.secondary.main,
231:     borderLeft: "8px solid " + theme.palette.secondary.dark,
232:   },
233:   textField: {
234:     margin: theme.spacing.unit,
235:     width: 400,
236:   },
237:   container: {
238:     display: "flex",
239:     flexDirection: "row",
240:     alignItems: "center",
241:   },
242:   sameline: {
243:     marginTop: "-12px",
244:   },
245:   warn: {
246:     margin: theme.spacing.unit,
247:     color: theme.palette.getContrastText(red['A700']),
248:     backgroundColor: red['A700'],
249:     '&:hover': {
250:       backgroundColor: red[900],
251:     }
252:   },
253:   extendedIcon: {
254:     marginRight: theme.spacing.unit,
255:   },
256:   button: {
257:     margin: theme.spacing.unit,
258:   },
259:   grow: {
260:     flexGrow: 1,
261:   }
262: });
```

```
263:
264: export const FeedbackFormStyle = theme => ({
265:   textField: {
266:     margin: theme.spacing.unit * 2,
267:     width: 100,
268:   },
269:   menu: {
270:     width: 100,
271:   },
272: });
273:
274: export const LoginPageStyle = theme => ({
275:   textField: {
276:     margin: theme.spacing.unit,
277:     width: '200px',
278:   },
279:   grow: {
280:     flexGrow: 1,
281:   }
282: });
283:
284: export const NewContentFormStyle = theme => ({
285:   root: {
286:     ...theme.mixins.gutters(),
287:     display: 'flex',
288:     flexWrap: 'wrap',
289:     flexDirection: 'row',
290:     alignItems: 'center',
291:     paddingTop: theme.spacing.unit * 3,
292:     paddingBottom: theme.spacing.unit * 3,
293:     height: 'auto',
294:     margin: theme.spacing.unit * 2,
295:   },
296:   textField: {
297:     margin: theme.spacing.unit,
298:     width: 180,
299:   },
300:   priceField: {
301:     marginTop: theme.spacing.unit,
302:     marginLeft: theme.spacing.unit,
303:     marginBottom: theme.spacing.unit,
304:     marginRight: 5,
305:     width: 95,
306:   },
307:   currency: {
308:     marginTop: theme.spacing.unit,
309:     marginRight: theme.spacing.unit,
310:     marginBottom: theme.spacing.unit,
311:     marginLeft: 0,
312:     width: 80,
313:   },
314:   menu: {
315:     width: 80,
316:   },
317:   formDiv: {
318:     display: "flex",
319:     alignItems: "center",
320:     flexWrap: "wrap",
321:   },
322:   buttonProgressPrimary: {
323:     color: theme.palette.secondary.main,
324:     position: 'absolute',
325:     top: '50%',
326:     left: '50%',
327:     marginTop: -12,
328:     marginLeft: -12,
```

```
329:   },
330:   wrapper: {
331:     display: "inline-block",
332:     float: "right",
333:     margin: theme.spacing.unit,
334:     position: 'relative',
335:   },
336: });
337:
338:
339: export const TransactionConfirmationStyle = theme => ({
340:   buttonOk: {
341:     margin: theme.spacing.unit,
342:   },
343:   buttonNo: {
344:     margin: theme.spacing.unit,
345:     color: red[700],
346:   }
347: });
348:
349: export const UserAppStyle = theme => ({
350:   notification: {
351:     color: theme.palette.text.secondary,
352:     backgroundColor: theme.palette.secondary.main,
353:     borderLeft: "8px solid " + theme.palette.secondary.dark,
354:   },
355:   button: {
356:     margin: theme.spacing.unit,
357:   },
358:   searchField: {
359:     width: 400,
360:     margin: theme.spacing.unit,
361:   },
362:   sameline: {
363:     marginTop: "-12px",
364:   },
365:   container: {
366:     display: "flex",
367:     flexDirection: "row",
368:     alignItems: "center",
369:   },
370:   search: {
371:     position: 'relative',
372:     borderRadius: theme.shape.borderRadius,
373:     backgroundColor: fade(theme.palette.common.white, 0.15),
374:     '&:hover': {
375:       backgroundColor: fade(theme.palette.common.white, 0.25),
376:     },
377:     marginRight: theme.spacing.unit * 2,
378:     marginLeft: 0,
379:     width: '100%',
380:     [theme.breakpoints.up('sm')]: {
381:       marginLeft: theme.spacing.unit * 3,
382:       width: 'auto',
383:     },
384:   },
385:   searchIcon: {
386:     width: theme.spacing.unit * 5,
387:     height: '100%',
388:     position: 'absolute',
389:     pointerEvents: 'none',
390:     display: 'flex',
391:     alignItems: 'center',
392:     justifyContent: 'center',
393:   },
394:   inputRoot: {
```

```
395:     color: 'inherit',
396:     width: '100%',
397:   },
398:   inputInput: {
399:     paddingTop: theme.spacing.unit,
400:     paddingRight: theme.spacing.unit,
401:     paddingBottom: theme.spacing.unit,
402:     paddingLeft: theme.spacing.unit * 6,
403:     transition: theme.transitions.create('width'),
404:     width: '100%',
405:     [theme.breakpoints.up('md')]: {
406:       width: 200,
407:     },
408:   },
409:   grow: {
410:     flexGrow: 1,
411:   },
412:   subheading: {
413:     marginTop: 0,
414:     width: "100%",
415:     display: "flex",
416:     alignItems: "center",
417:   },
418:   subMargin: {
419:     margin: theme.spacing.unit,
420:     marginTop: 0,
421:   },
422:   normalMargin: {
423:     margin: theme.spacing.unit,
424:   },
425:   prFetch: {
426:     color: theme.palette.secondary.main,
427:   },
428:   premium: {
429:     color: green.A400,
430:   },
431:   notPremium: {
432:     color: red.A400,
433:   },
434:   extendedIcon: {
435:     marginRight: theme.spacing.unit,
436:   },
437: });
438:
439: export const Web3LoginFormStyle = theme => ({
440:   root: {
441:     ...theme.mixins.gutters(),
442:     display: 'flex',
443:     flexWrap: 'wrap',
444:     flexDirection: 'column',
445:     justifyContent: 'center',
446:     alignItems: 'center',
447:     paddingTop: theme.spacing.unit * 2,
448:     paddingBottom: theme.spacing.unit * 2,
449:     height: 'auto',
450:     //minHeight: 500,
451:     "@media (min-height:550px)" : {
452:       minHeight: 500,
453:     }
454:   },
455:   container: {
456:     width: 'auto',
457:     display: 'flex',
458:     flexWrap: 'wrap',
459:     justifyContent: 'center',
460:   },
```

```
461:   textField: {
462:     margin: theme.spacing.unit * 2,
463:     width: 200,
464:   },
465:   menu: {
466:     width: 250,
467:   },
468:   extendedIcon: {
469:     marginRight: theme.spacing.unit,
470:   },
471:   button: {
472:     margin: 0,
473:   },
474:   buttonProgressPrimary: {
475:     color: theme.palette.secondary.main,
476:     position: 'absolute',
477:     top: '50%',
478:     left: '50%',
479:     marginTop: -12,
480:     marginLeft: -12,
481:   },
482:   buttonProgressSecondary: {
483:     color: theme.palette.primary.main,
484:     position: 'absolute',
485:     top: '50%',
486:     left: '50%',
487:     marginTop: -12,
488:     marginLeft: -12,
489:   },
490:   wrapper: {
491:     marginLeft: theme.spacing.unit,
492:     marginRight: theme.spacing.unit,
493:     position: 'relative',
494:   },
495:   fabProgress: {
496:     color: theme.palette.primary.main,
497:     position: 'absolute',
498:     top: 2,
499:     left: 2,
500:     zIndex: 1,
501:   },
502:   image: {
503:     maxWidth: 0,
504:     "@media (min-height:450px)" : {
505:       maxWidth: 200,
506:     }
507:   }
508: });
509:
```

```

1:  /*
2:  *
3:  * University di Pisa - Master's Degree in Computer Science and Networking
4:  *
5:  * Final Project for the course of Peer to Peer Systems and Blockchains
6:  *
7:  * Teacher: Prof. Laura Ricci
8:  *
9:  * Candidate: Orlando Leombruni, matricola 475727
10: *
11: * File: CatalogManager.js
12: *
13: */
14:
15: import React from 'react';
16: import classNames from 'classnames';
17: import { withStyles } from '@material-ui/core/styles';
18: import PropTypes from 'prop-types';
19: import { Typography,
20:         Button,
21:         TextField,
22:         Slide,
23:         Dialog,
24:         DialogContent,
25:         DialogActions,
26:         DialogTitle } from '@material-ui/core';
27: import { LibraryAdd, Delete } from '@material-ui/icons';
28: import Web3LoginForm from "../Web3LoginForm";
29: import json from '../solidity/compiled.json';
30: import TransactionConfirmation from "../TransactionConfirmation";
31: import AppDrawer from '../AppDrawer';
32: import { getTransactionParameters, makeTitle } from "../Utils";
33: import { CatalogManagerStyle as styles } from "../styles/MaterialCustomStyles";
34:
35: /*
36: * "Slide up" transition for the animation of a React component.
37: */
38: const Transition = (props) =>
39:   <Slide direction="up" {...props} />;
40:
41: /*
42: * CatalogManager Class
43: *
44: * A React Component that provides a very simple interface for creating and deleting
45: Catalog contracts.
46: */
47: class CatalogManager extends React.Component {
48:   constructor(props) {
49:     super(props);
50:     this.state = {
51:       loggedAccount: null,
52:       message: "",
53:       dialogOpen: false,
54:       catalogToClose: "",
55:       error: false,
56:       confirm: false,
57:       confirmProps: {},
58:       isClose: false,
59:     };
60:     this.abi = JSON.parse(json.contracts["Catalog.sol:Catalog"].abi);
61:     this.bin = "0x" + json.contracts["Catalog.sol:Catalog"].bin;
62:     this.props.manageCenter(true);
63:   }
64:
65:   /*

```

```

66:   * A callback function invoked when the user has chosen and unlocked an EOA from
67:   which to perform Catalog operations.
68:   */
69:   unlockCallback = account => {
70:     this.setState(oldState => ({...oldState, loggedAccount: account}));
71:     this.props.manageCenter(false);
72:   };
73:   /*
74:   * Begins the creation of a new Catalog contract, estimating the gas cost for th
75:   e operation.
76:   */
77:   createCatalog = () => {
78:     const { web3 } = this.props;
79:     const Catalog = new web3.eth.Contract(this.abi, {data: this.bin});
80:     const d = Catalog.deploy();
81:     this.setState(oldState => ({...oldState, isClose: false, dialogOpen: false,
82: error: false}));
83:     getTransactionParameters(web3, d, this.state.loggedAccount).then(
84:       (result) => this.setState(o => ({...o, confirmProps: result, confirm: tr
85: ue})),
86:       (error) => console.log(error)
87:     );
88:   };
89:   /*
90:   * Invoked when the user confirms the transaction (after reviewing it in an info
91:   rmative dialog), this function
92:   * effectively deploys a new Catalog and then queries it to check if it's been c
93:   orrectly created.
94:   */
95:   confirmCreate = () => {
96:     const {web3} = this.props;
97:     const Catalog = new web3.eth.Contract(this.abi, {data: this.bin});
98:     Catalog.deploy().send({
99:       from: this.state.loggedAccount,
100:      gasPrice: this.state.confirmProps.gasPrice,
101:      gas: this.state.confirmProps.gas,
102:    }).then(
103:      (result) => {
104:        result.methods.isActiveCatalog().call({from: this.state.loggedAccoun
105: t})
106:        .then(
107:          () => this.setState(oldState => ({...oldState,
108: message: 'Catalog created at address ${result.options.ad
109: dress}'})
110:          ),
111:          () => this.setState(oldState => ({...oldState, message: 'Err
112: or in creating catalog...'})
113:          );
114:        },
115:        (error) => {
116:          this.setState(oldState => ({...oldState, message: 'Error in creating
117: catalog...' }));
118:          console.log(error)
119:        }
120:      );
121:      this.setState(oldState => ({...oldState, message: "Creating catalog...", con
122: firm: false, confirmProps: {}}));
123:    };
124:   /*
125:   * Shows (or hides) the dialog window where the user can input the address of th
126:   e Catalog to close.
127:   */
128:   handleDialog = () =>

```

```

120:         this.setState(oldState => ({...oldState, dialogOpen: !oldState.dialogOpen}));
121:
122:     /*
123:      * Handles the input of the address of the Catalog to close.
124:      */
125:     handleChange = event => {
126:         const { target } = event;
127:         this.setState(oldState => ({...oldState, catalogToClose: target.value}));
128:     };
129:
130:     /*
131:      * Begins the closing and deletion of a Catalog contract, estimating the gas cost of the operation.
132:      */
133:     deleteCatalog = () => {
134:         let error = (this.state.catalogToClose === "");
135:         const { web3 } = this.props;
136:         const abi = JSON.parse(json.contracts["Catalog.sol:Catalog"].abi);
137:         const Catalog = new web3.eth.Contract(abi);
138:         try {
139:             Catalog.options.address = this.state.catalogToClose;
140:         } catch (e) {
141:             error = true;
142:             console.log(e);
143:         }
144:         if (!error) {
145:             this.setState(oldState => ({...oldState, isClose: true, dialogOpen: false, error: false}));
146:             getTransactionParameters(web3, Catalog.methods.closeCatalog(), this.state.loggedAccount).then(
147:                 (result) => this.setState(o => ({...o, confirmProps: result, confirmed: true})),
148:                 (error) => console.log(error)
149:             );
150:         } else {
151:             this.setState(oldState => ({...oldState, error}));
152:         }
153:     };
154:
155:     /*
156:      * Invoked when the user confirms the transaction (after reviewing it in an informative dialog), this function
157:      * checks that the Catalog is valid and active and then proceeds to close and delete it.
158:      */
159:     confirmDelete = () => {
160:         const {web3} = this.props;
161:         const abi = JSON.parse(json.contracts["Catalog.sol:Catalog"].abi);
162:         const Catalog = new web3.eth.Contract(abi, this.state.catalogToClose);
163:         Catalog.methods.isActiveCatalog().call({from: this.state.loggedAccount})
164:             .then(
165:                 () => {
166:                     Catalog.methods.closeCatalog().send({
167:                         from: this.state.loggedAccount,
168:                         gasPrice: this.state.confirmProps.gasPrice,
169:                         gas: this.state.confirmProps.gas,
170:                     }).then(
171:                         (result) => {
172:                             Catalog.methods.isActiveCatalog().call({from: this.state.loggedAccount})
173:                                 .then(
174:                                     () => this.setState(oldState => ({...oldState, message: 'Catalog WAS NOT closed! Make sure you're logged in as the owner.'}))
175:                                 ),

```

```

177:         () => this.setState(oldState => ({...oldState, m
message: 'Catalog closed!'}))
178:             );
179:         },
180:         (error) => {
181:             this.setState(oldState => ({...oldState, message: 'Error
in closing catalog'}));
182:             console.log(error)
183:         }
184:     );
185:   },
186:   () => this.setState(oldState => ({...oldState, message: 'Catalog has
already been closed!'})))
187:   );
188:   this.setState(oldState => ({...oldState, message: "Closing catalog...", conf
irm: false, confirmProps: {})));
189: };
190:
191: render() {
192:   const {web3, classes, noLogin} = this.props;
193:   const {loggedAccount, message, dialogOpen, catalogToClose, error, confirm,
confirmProps, isClose} = this.state;
194:
195:   return (
196:     <AppDrawer drawer={false} loading={false} writeTitle={() => makeTitle("C
atalog Management", classes.grow)} render={() => ""} >
197:       {Boolean(loggedAccount) ?
198:         <div>
199:           <div className={classes.holder}>
200:             <div className={classes.elements}>
201:               <Typography variant={"body1"} color={"textPrimary"} clas
sName={classes.typography}>
202:                 Deploy a new Catalog
203:               </Typography>
204:               <Button variant={"extendedFab"} onClick={this.createCata
log} color={"secondary"} className={classes.button}>
205:                 <LibraryAdd className={classes.extendedIcon} />
206:                 Add New
207:               </Button>
208:             </div>
209:             <div className={classes.elements}>
210:               <Typography variant={"body1"} color={"textPrimary"} clas
sName={classes.typography}>
211:                 Remove a Catalog
212:               </Typography>
213:               <Button variant={"extendedFab"}
214:                 color={"inherit"}
215:                 className={classNames(classes.button, classes.wa
rn)}
216:                 onClick={this.handleDialog} >
217:                 <Delete className={classes.extendedIcon} />
218:                 Delete
219:               </Button>
220:             </div>
221:           </div>
222:           <div className={classes.holder}>
223:             <Typography variant={"body1"} color={"textPrimary"} classNam
e={classes.typography} >
224:               {message}
225:             </Typography>
226:           </div>
227:         <Dialog
228:           open={dialogOpen}
229:           keepMounted
230:           TransitionComponent={Transition}
231:           onBackPressed={this.handleDialog}

```



```
232:         onEscapeKeyDown={this.handleDialog} >
233:         <DialogTitle>Select Catalog to close</DialogTitle>
234:         <DialogContent>
235:             <TextField
236:                 id="catalogAddress"
237:                 required
238:                 name="Catalog"
239:                 label="Catalog"
240:                 className={classes.textField}
241:                 value={catalogToClose}
242:                 onChange={this.handleChange}
243:                 error={error}
244:                 helperText={error ? "Invalid COBrA Catalog address"
: ""}
245:             />
246:         </DialogContent>
247:         <DialogActions>
248:             <Button variant="extendedFab"
249:                 color={"inherit"}
250:                 className={classNames(classes.button, classes.wa
rn)}
251:                 onClick={this.deleteCatalog} >
252:                 Delete
253:             </Button>
254:         </DialogActions>
255:     </Dialog>
256:     {confirm ?
257:         <TransactionConfirmation
258:             {...confirmProps}
259:             ok={isClose? this.confirmDelete : this.confirmCreate}
260:             cancel={() => this.setState(oldState => ({...oldState, c
onfirm: false, confirmProps: {}}))} /> :
261:         ""}
262:     </div> :
263:     <Web3LoginForm noLogin={noLogin} web3={web3} onUnlock={this.unlockCa
llback}/>}
264:     </AppDrawer>
265:   );
266: }
267: }
268:
269: CatalogManager.propTypes = {
270:   noLogin: PropTypes.bool.isRequired,
271:   web3: PropTypes.object.isRequired,
272:   classes: PropTypes.object.isRequired,
273:   manageCenter: PropTypes.func.isRequired,
274: };
275:
276: export default withStyles(styles)(CatalogManager);
```



```

1:  /*
2:  *
3:  * University di Pisa - Master's Degree in Computer Science and Networking
4:  *
5:  * Final Project for the course of Peer to Peer Systems and Blockchains
6:  *
7:  * Teacher: Prof. Laura Ricci
8:  *
9:  * Candidate: Orlando Leombruni, matricola 475727
10:  *
11:  * File: LoginPage.js
12:  *
13:  */
14:
15: import React from 'react';
16: import PropTypes from 'prop-types';
17: import {
18:   withStyles,
19:   Dialog,
20:   DialogTitle,
21:   DialogContent,
22:   DialogContentText,
23:   DialogActions,
24:   TextField,
25:   Slide,
26:   Button,
27: } from '@material-ui/core';
28: import './styles/login.css';
29: import Web3LoginForm from '../Web3LoginForm';
30: import json from '../solidity/compiled.json';
31: import AppDrawer from '../AppDrawer';
32: import {makeTitle} from '../Utils';
33: import { LoginPageStyle as styles } from '../styles/MaterialCustomStyles';
34:
35: /*
36:  * "Slide up" transition for the animation of a React component.
37:  */
38: const Transition = (props) =>
39:   <Slide direction="up" {...props} />;
40:
41: /*
42:  * LoginPage Class
43:  *
44:  * A React Component that shows a login form to the user, allowing them also to sele
ct a Catalog contract to connect to.
45:  */
46: class LoginPage extends React.Component {
47:
48:   constructor(props) {
49:     super(props);
50:     this.state = {
51:       logged: false,
52:       loggedAccount: null,
53:       catalogAccount: "",
54:       error: false,
55:     };
56:     this.props.manageCenter(true);
57:   }
58:
59:   // Changes this component's state when the Web3 instance unlocks the selected EO
A.
60:   unlockCallback = account => {
61:     if (account === "error") this.props.onSubmit("error", "error", "error");
62:     else this.setState(oldState => ({...oldState, loggedAccount: account, lo
gged: true}));
63:   };

```

```

64:
65:   /*
66:    * After the desired EOA is unlocked by the Web3 instance and a Catalog account
is chosen, this function checks that
67:    * latter is a valid Catalog and then "reports" to the enclosing component the u
ser EOA, the Catalog account and
68:    * which app has been selected (Creator or User).
69:    */
70:   report = appType => () => {
71:     this.setState(oldState => ({...oldState, error: false}));
72:     let error = (this.state.catalogAccount === "");
73:     const { web3 } = this.props;
74:     const abi = JSON.parse(json.contracts["Catalog.sol:Catalog"].abi);
75:     const Catalog = new web3.eth.Contract(abi);
76:     try {
77:       Catalog.options.address = this.state.catalogAccount;
78:     } catch (e) {
79:       error = true;
80:     }
81:     if (!error) {
82:       Catalog.methods.isActiveCatalog().call({from: this.state.loggedAccount})
83:         .then(
84:           (result) => {
85:             this.props.onSubmit(this.state.loggedAccount, this.state.cat
alogAccount, appType)
86:           },
87:           (error) => {
88:             console.log(error);
89:             this.setState(oldState => ({...oldState, error: true}));
90:           }
91:         );
92:     } else {
93:       this.setState(oldState => ({...oldState, error: true}));
94:     }
95:   };
96:
97:   // Handles user changes in the form.
98:   handleChange = event => {
99:     const { target } = event;
100:    this.setState(oldState => ({...oldState, catalogAccount: target.value}));
101:  };
102:
103:   render() {
104:     const { web3, classes, noLogin } = this.props;
105:     const { logged, catalogAccount, error } = this.state;
106:
107:     return (
108:       <AppDrawer loading={false} render={() => ""} drawer={false} writeTitle={
() => makeTitle("Login", classes.grow)}>
109:         {logged} ?
110:         <Dialog
111:           open={true}
112:           disableBackdropClick={true}
113:           disableEscapeKeyDown={true}
114:           TransitionComponent={Transition}>
115:           <DialogTitle>{"Logged in successfully!"}</DialogTitle>
116:           <DialogContent>
117:             <DialogContentText>
118:               COBrA kai successfully connected to the Ethereum network
119:
120:               Please insert the desired Catalog's address, then
               choose if you want to use the service as an user or as a
content creator.
121:             </DialogContentText>
122:             <TextField
123:               id="catalogAddress"

```

```
124:         required
125:         name="Catalog"
126:         label="Catalog"
127:         className={classes.textField}
128:         value={catalogAccount}
129:         onChange={this.handleChange}
130:         error={error}
131:         helperText={error ? "Invalid COBrA Catalog address" : ""}
132:     }
133:     </DialogContent>
134:     <DialogActions>
135:         <Button color="secondary" onClick={this.report("user")}>
136:             User
137:         </Button>
138:         <Button color="secondary" onClick={this.report("creator")}>
139:             Creator
140:         </Button>
141:     </DialogActions>
142: </Dialog> :
143: <Web3LoginForm noLogin={noLogin} onUnlock={this.unlockCallback} web3
={ web3 }/>
144:     </AppDrawer>
145:   );
146: }
147: }
148:
149: LoginPage.propTypes = {
150:   noLogin: PropTypes.bool.isRequired,
151:   web3: PropTypes.object.isRequired,
152:   classes: PropTypes.object.isRequired,
153:   onSubmit: PropTypes.func.isRequired,
154:   manageCenter: PropTypes.func.isRequired,
155: };
156:
157: export default withStyles(styles)(LoginPage);
```

```

1:  /*
2:  *
3:  * University di Pisa - Master's Degree in Computer Science and Networking
4:  *
5:  * Final Project for the course of Peer to Peer Systems and Blockchains
6:  *
7:  * Teacher: Prof. Laura Ricci
8:  *
9:  * Candidate: Orlando Leombruni, matricola 475727
10: *
11: * File: AppDrawer.js
12: *
13: */
14:
15: import React from 'react';
16: import PropTypes from 'prop-types';
17: import classNames from 'classnames';
18: import {
19:   withStyles,
20:   Drawer,
21:   AppBar,
22:   Toolbar,
23:   CssBaseline,
24:   IconButton,
25:   LinearProgress,
26: } from '@material-ui/core';
27: import {
28:   Menu as MenuItem,
29:   ChevronLeft as ChevronLeftIcon,
30:   ChevronRight as ChevronRightIcon,
31: } from '@material-ui/icons';
32: import secondaryLogo from '../assets/logo2.png';
33: import { AppDrawerStyle as styles } from '../styles/MaterialCustomStyles';
34:
35: /*
36: * AppDrawer Class
37: *
38: * A React Component that provides a title bar and (optionally) a drawer menu to a p
age Component.
39: */
40:
41: class AppDrawer extends React.Component {
42:   state = {
43:     open: false,
44:   };
45:
46:   /*
47:   * Manage the opening and closing of the drawer. Functions used as callbacks for
the onClick properties of buttons.
48:   */
49:   handleDrawerOpen = () => {
50:     this.setState({ open: true });
51:   };
52:   handleDrawerClose = () => {
53:     this.setState({ open: false });
54:   };
55:
56:   render() {
57:     const { classes, theme, drawer, writeTitle, loading, children } = this.props
;
58:
59:     return (
60:       <div className={classes.root}>
61:         <CssBaseline />
62:         <AppBar
63:           position="fixed"

```

```

64:         className={classNames(classes.appBar, {
65:           [classes.appBarShift]: this.state.open,
66:         })}>
67:         <Toolbar className={classes.rightGutter} disableGutters={drawer
68:           {drawer &&
69:             <IconButton
70:               color="inherit"
71:               aria-label="Open drawer"
72:               onClick={this.handleDrawerOpen}
73:               className={classNames(classes.menuButton, {
74:                 [classes.hide]: this.state.open,
75:               })}>
76:             <MenuIcon />
77:           </IconButton>}
78:           {writeTitle()}
79:           <img src={secondaryLogo} alt="logo" className={classes.log
o} />
80:         </Toolbar>
81:       </AppBar>
82:       {drawer? <Drawer
83:         variant="permanent"
84:         className={classNames(classes.drawer, {
85:           [classes.drawerOpen]: this.state.open,
86:           [classes.drawerClose]: !this.state.open,
87:         })}
88:         classes={{
89:           paper: classNames({
90:             [classes.drawerOpen]: this.state.open,
91:             [classes.drawerClose]: !this.state.open,
92:           }),
93:         }}
94:         open={this.state.open}
95:       >
96:         <div className={classes.toolbar}>
97:           <IconButton onClick={this.handleDrawerClose}>
98:             {theme.direction === 'rtl' ? <ChevronRightIcon /> : <Che
vronLeftIcon />}
99:           </IconButton>
100:         </div>
101:         {this.props.render(this.state)}
102:       </Drawer> : null}
103:       <main className={classes.content}>
104:         {loading && <LinearProgress color="secondary" className={class
es.linearProgress}/>}
105:         {loading && <div className={classes.darken} />}
106:         <div className={classes.toolbar} />
107:         {children}
108:       </main>
109:     </div>
110:   );
111: }
112:
113: AppDrawer.propTypes = {
114:   classes: PropTypes.object.isRequired,
115:   theme: PropTypes.object.isRequired,
116:   drawer: PropTypes.bool.isRequired,
117:   render: PropTypes.func.isRequired,
118:   writeTitle: PropTypes.func.isRequired,
119:   loading: PropTypes.bool.isRequired,
120: };
121:
122: export default withStyles(styles, { withTheme: true })(AppDrawer);
123:

```



```

1:  /*
2:  *
3:  * University di Pisa - Master's Degree in Computer Science and Networking
4:  *
5:  * Final Project for the course of Peer to Peer Systems and Blockchains
6:  *
7:  * Teacher: Prof. Laura Ricci
8:  *
9:  * Candidate: Orlando Leombruni, matricola 475727
10: *
11: * File: Web3LoginForm.js
12: *
13: */
14:
15: import React from 'react';
16: import { withStyles } from '@material-ui/core/styles';
17: import PropTypes from 'prop-types';
18: import { Grid,
19:   Typography,
20:   TextField,
21:   MenuItem,
22:   FormControl,
23:   InputLabel,
24:   Input,
25:   InputAdornment,
26:   IconButton,
27:   Button,
28:   CircularProgress,
29:   Paper } from '@material-ui/core';
30: import { Visibility,
31:   VisibilityOff,
32:   Update,
33:   LockOpen } from "@material-ui/icons";
34: import mainLogo from '../assets/logo.png';
35: import { Web3LoginFormStyle as styles } from "../styles/MaterialCustomStyles";
36:
37: /*
38:  * Web3LoginForm Class
39:  *
40:  * A React Component that queries the Web3 instance for the list of available EOAs and
41:  * allows the user to unlock one
42:  * of them in order to be used as the main account for performing operations on the
43:  * blockchain.
44:  */
45: class Web3LoginForm extends React.Component {
46:   constructor(props) {
47:     super(props);
48:     this.state = {
49:       loading: false,
50:       account: "",
51:       password: "",
52:       accounts: [],
53:       loaded: false,
54:       bottomText: "",
55:       showPassword: false,
56:     };
57:   }
58:   /*
59:    * This is a React state function that will be called only once, after the compo
60:    * nent is mounted in the virtual
61:    * DOM but before it gets rendered.
62:    *
63:    * In particular, this function asks the underlying Web3 instance for the list o
64:    * f available EOAs.

```

```

63:  */
64:  componentDidMount() {
65:    const { web3, noLogin, onUnlock } = this.props;
66:    if (noLogin) {
67:      web3.eth.getAccounts().then(
68:        (list) => {
69:          if (list.length > 0) onUnlock(list[0]);
70:          else onUnlock("error");
71:        },
72:        (err) => console.log(err)
73:      );
74:    } else {
75:      web3.eth.getAccounts().then(this.showAccounts, (err) => console.log(err)
76:    );
77:    }
78:  }
79:  /*
80:   * Populates the list data structure containing the available EOAs.
81:   */
82:  showAccounts = accounts => {
83:    if (accounts && accounts.length > 0) {
84:      const accountList = accounts.map((account, index) => ({
85:        label: `Account #${index+1} (${account.replace(account.slice(5,
86:        value: account,
87:      }));
88:    });
89:    this.setState(oldState => ({...oldState, accounts: accountList, loaded:
90:    true}));
91:  } else {
92:    this.setState(oldState => ({...oldState, accounts: [{label: "No accounts
93:    found", value: null}]}));
94:  }
95:  };
96:  /*
97:   * Submits the form; asks the Web3 instance to unlock the selected account, then
98:   * reports back to the
99:   * enclosing component.
100:  */
101:  submit = event => {
102:    event.preventDefault();
103:    const { account, password } = this.state;
104:    if (!account || !password) return;
105:    const { web3 } = this.props;
106:    web3.eth.personal.unlockAccount(account, password, 0).then(
107:      () => {
108:        this.setState(oldState => ({...oldState, loading: false}));
109:        this.props.onUnlock(account);
110:      },
111:      (error) => {
112:        console.log(error);
113:        this.setState(oldState => ({...oldState, loading: false, bottomText:
114:        "Wrong username and/or password"}))
115:      }
116:    );
117:    this.setState(oldState => ({...oldState, loading: true}));
118:  };
119:  /*
120:   * Handles the show/hide password button.
121:   */
122:  handleShowPassword = () => {
123:    this.setState(oldState => ({...oldState, showPassword: !oldState.showPasswor
124:    d}));
125:  };

```

```

122: // Handles changes made in the form.
123: changedForm = event => {
124:   const { target } = event;
125:   this.setState(oldState => ({...oldState, [target.name]: target.value}));
126: };
127:
128: render() {
129:   const { classes } = this.props;
130:   const { account, password, accounts, showPassword, loading, loaded } = this.
state;
131:
132:   return (
133:     <Paper className={classes.root} id="paper" elevation={1}>
134:       <Grid
135:         container
136:         spacing={0}
137:         display="flex"
138:         direction="column"
139:         alignItems="center"
140:         justify="center">
141:         <Grid item style={{width: "fit-content"}} ><img src={mainLogo} alt="logo" className={classes.image} /></Grid>
142:         <Grid item style={{margin: "8px"}}>
143:           <Typography variant="h5" color="textPrimary">
144:             Welcome! Please login.
145:           </Typography>
146:         </Grid>
147:         {this.state.bottomText && <Grid item >
148:           <Typography variant={body1} color={error} id="bottomText"
">{this.state.bottomText}&nbsp;</Typography>
149:         </Grid>}
150:         <Grid style={{display: "flex", alignItems: "center", flexDirecti
on: "column"}} item xs={6}>
151:           <form className={classes.container} noValidate autoComplete=
"off" onSubmit={this.submit}>
152:             <TextField
153:               id="list-accounts"
154:               select
155:               required
156:               name="account"
157:               label="Account"
158:               className={classes.textField}
159:               value={account}
160:               onChange={this.changedForm}
161:               SelectProps={{MenuProps: {className: classes.men
u}}}
162:               margin="normal" >
163:               {accounts.map(option => (
164:                 <MenuItem key={option.value} value={option.v
alue}>
165:                   {option.label}
166:                 </MenuItem>
167:               ))}
168:             </TextField>
169:             <FormControl required className={classes.textField}>
170:               <InputLabel htmlFor="password">Password</InputLabel>
171:               <Input
172:                 id="password"
173:                 name="password"
174:                 value={password}
175:                 onChange={this.changedForm}
176:                 type={showPassword ? 'text' : 'password'}
177:                 endAdornment={
178:                   <InputAdornment position="end">
179:                     <IconButton
180:                       aria-label="Toggle password visibili

```

```

ty"
181:
182:       onClick={this.handleShowPassword} >
       {showPassword ? <Visibility/> : <Vis
ibilityOff/>}
183:
184:       </IconButton>
185:     }
186:   </InputAdornment>
187: </FormControls>
188: <div style={{display: "flex", alignItems: "center"}}>
189:   <div className={classes.wrapper}>
190:     <IconButton
191:       onClick={() => {
192:         this.setState(oldState => ({...oldState, loa
ded: false}));
193:         this.props.web3.eth.getAccounts().then(this.
showAccounts);
194:       }}
195:       disabled={!loaded}
196:       className={classes.button}>
197:       <Update/>
198:     </IconButton>
199:     {!loaded && <CircularProgress size={24} className={c
lasses.buttonProgressSecondary}/>}
200:   </div>
201:   <div className={classes.wrapper}>
202:     <Button
203:       variant="extendedFab"
204:       color="primary"
205:       aria-label="Unlock"
206:       onClick={this.submit}
207:       className={classes.button}
208:       disabled={loading || (!account || !password)}>
209:       <LockOpen className={classes.extendedIcon}/>
210:       Unlock
211:     </Button>
212:     {loading && <CircularProgress size={24} className={c
lasses.buttonProgressPrimary}/>}
213:   </div>
214: </div>
215: </form>
216: </Grid>
217: </Grid>
218: </Paper>
219:   );
220: }
221: }
222:
223:
224:
225: Web3LoginForm.propTypes = {
226:   noLogin: PropTypes.bool.isRequired,
227:   web3: PropTypes.object.isRequired,
228:   onUnlock: PropTypes.func.isRequired,
229: };
230:
231:
232: export default withStyles(styles)(Web3LoginForm);

```



```

1: /*
2:  *
3:  * University di Pisa - Master's Degree in Computer Science and Networking
4:  *
5:  * Final Project for the course of Peer to Peer Systems and Blockchains
6:  *
7:  * Teacher: Prof. Laura Ricci
8:  *
9:  * Candidate: Orlando Leombruni, matricola 475727
10:  *
11:  * File: BuyGiftDialog.js
12:  *
13:  */
14:
15: import React from 'react';
16: import PropTypes from 'prop-types';
17: import {
18:   withStyles,
19:   Dialog,
20:   DialogContent,
21:   DialogActions,
22:   DialogTitle,
23:   FormControl,
24:   FormControlLabel,
25:   DialogContentText,
26:   Radio,
27:   RadioGroup,
28:   TextField,
29:   Button,
30: } from '@material-ui/core';
31: import {prettifyWei} from "../Utils";
32: import {BuyGiftDialogStyle as styles} from "../styles/MaterialCustomStyles";
33:
34: /*
35:  * BuyGiftDialog Class
36:  *
37:  * A React Component that displays a dialog window, reviewing the purchase of a Cont
ent and asking the user
38:  * if they want to buy it for themselves or gift it to another user.
39:  */
40: class BuyGiftDialog extends React.Component {
41:
42:   state = {
43:     selected: 'me',
44:     account: '',
45:   };
46:
47:   handleChange = event => {
48:     const { target } = event;
49:     this.setState(oldState => ({...oldState, selected: target.value}));
50:   };
51:
52:   handleAccount = event => {
53:     const { target } = event;
54:     this.setState(oldState => ({...oldState, account: target.value}));
55:   };
56:
57:   render() {
58:     const {classes, callback, cancel, price} = this.props;
59:
60:     return (
61:       <div className={classes.root}>
62:         <Dialog
63:           open={true}
64:           keepMounted
65:           onBackdropClick={cancel}

```

```

66:           onEscapeKeyDown={cancel}>
67:             <DialogTitle>Purchase details</DialogTitle>
68:             <DialogContent>
69:               <DialogContentText>You need to pay {prettifyWei(price)} to acces
s this content.</DialogContentText>
70:               <FormControl component={"fieldset"} className={classes.formContr
ol}>
71:                 <RadioGroup
72:                   name={"recipient"}
73:                   className={classes.group}
74:                   value={this.state.selected}
75:                   onChange={this.handleChange}>
76:                   <FormControlLabel control={<Radio/>} label={"Buy it for
me"} value={"me"}/>
77:                   <FormControlLabel control={<Radio/>} label={
78:                     <div style={{display: "flex", alignItems: "center"}}>
79:                       Buy it for someone else:
80:                       <TextField
81:                         name={"account"}
82:                         type={"string"}
83:                         placeholder={"Account address..."}
84:                         value={this.state.account}
85:                         className={classes.textField}
86:                         onChange={event => this.handleAccount(event)}
87:                         onClick={() => this.setState(oldState => ({...ol
dState, selected: 'other'}})}>
88:                         />
89:                       </div>
90:                     } value={"other"}/>
91:                     </RadioGroup>
92:                   </FormControl>
93:                 </DialogContent>
94:                 <DialogActions>
95:                   <Button color={"inherit"} onClick={cancel}>Cancel</Button>
96:                   <Button color={"secondary"} onClick={this.state.selected === 'me
' ? callback() : callback(this.state.account)}>Ok</Button>
97:                 </DialogActions>
98:               </Dialog>
99:             </div>
100:           );
101:         }
102:       }
103:     }
104:
105:     BuyGiftDialog.propTypes = {
106:       classes: PropTypes.object.isRequired,
107:       callback: PropTypes.func.isRequired,
108:       cancel: PropTypes.func.isRequired,
109:       price: PropTypes.string.isRequired,
110:     };
111:
112:     export default withStyles(styles)(BuyGiftDialog);

```



```

1:  /*
2:  *
3:  * University di Pisa - Master's Degree in Computer Science and Networking
4:  *
5:  * Final Project for the course of Peer to Peer Systems and Blockchains
6:  *
7:  * Teacher: Prof. Laura Ricci
8:  *
9:  * Candidate: Orlando Leombruni, matricola 475727
10: *
11: * File: ContentList.js
12: *
13: */
14: import React from 'react';
15: import PropTypes from 'prop-types';
16: import CopyToClipboard from 'react-copy-to-clipboard';
17: import { withStyles } from '@material-ui/core/styles';
18: import {
19:   ExpansionPanel,
20:   ExpansionPanelActions,
21:   ExpansionPanelDetails,
22:   ExpansionPanelSummary,
23:   Typography,
24:   Button,
25:   Tooltip
26: } from '@material-ui/core';
27: import { ExpandMore } from '@material-ui/icons';
28: import { ContentListStyle as styles } from '../styles/MaterialCustomStyles';
29:
30: /*
31:  * CopyWrapper Class
32:  *
33:  * A React Component that wraps another component and makes it clickable; when click
34:  * gets copied into the clipboard
35:  */
36: const CopyWrapper = ({condition, placement, what, onCopy, children}) => (
37:   condition ?
38:     <Tooltip title={"Click to copy"} placement={placement}>
39:       <CopyToClipboard text={what} onCopy={onCopy}>
40:         {children}
41:       </CopyToClipboard>
42:     </Tooltip> :
43:     <div>
44:       {children}
45:     </div>
46: );
47:
48: /*
49:  * ContentList Class
50:  *
51:  * A React Component that displays a list where each item has several (displayable)
52:  * attributes.
53:  * The component is completely generic and can be used in a multitude of ways.
54:  */
55: class ContentList extends React.Component {
56:
57:   constructor(props) {
58:     super(props);
59:     this.divRef = React.createRef();
60:     this.handleResize = this.handleResize.bind(this);
61:   }
62:
63:
64:   state = {

```

```

65:     smallerBox: false,
66:   };
67:
68:   /*
69:    * Function that handles the resizing of the browser window.
70:    */
71:   handleResize() {
72:     if (this.divRef) {
73:       this.setState({smallerBox: (this.divRef.getBoundingClientRect().width <
531)});
74:     }
75:   }
76:
77:   // React lifecycle methods to add/remove a 'resize' event listener when the comp
onent gets mounted/unmounted.
78:   componentDidMount() {
79:     this.handleResize();
80:     window.addEventListener('resize', this.handleResize);
81:   }
82:
83:   componentWillUnmount() {
84:     window.removeEventListener('resize', this.handleResize);
85:   }
86:
87:   render() {
88:     const { classes, className, list, action, attributes, actionName, action2, a
ction2Name, notify } = this.props;
89:     const percent = (100/attributes.length) + "%";
90:     return (
91:       <div className={className ? className : classes.root} ref={element => th
is.divRef = element}>
92:         {list.map((item, key) =>
93:           <ExpansionPanel key={key}>
94:             <ExpansionPanelSummary expandIcon={<ExpandMore/>}>
95:               <Typography className={classes.heading}>{item.description}
n</Typography>
96:             </ExpansionPanelSummary>
97:             <ExpansionPanelDetails className={classes.details}>
98:               {attributes.map((attr, index) => (
99:                 <div key={index} style={{flexBasis: percent}}>
100:                   <Typography variant={"subtitle2"} style={{margin
Right: 12}}>{attr.description}</Typography>
101:                   <CopyWrapper
102:                     condition={attr.copy}
103:                     placement={"bottom"}
104:                     what={item[attr.name]}
105:                     onCopy={() => notify("Copied!", "Copied to c
lipboard", "success")}>
106:                     <Typography style={{cursor: attr.copy? "poin
ter" : "default"}}>
107:                       {attr.extra ?
108:                         (this.state.smallerBox ?
109:                           attr.makeSmall(attr.extra(item[a
ttr.name])) :
110:                           attr.extra(item[attr.name])) :
111:                         (this.state.smallerBox ?
112:                           attr.makeSmall(item[attr.name])
113:                           item[attr.name]))}
114:                     </Typography>
115:                   </CopyWrapper>
116:                 </div>
117:               )}}
118:             </ExpansionPanelDetails>
119:             <ExpansionPanelActions>
120:               {Boolean(action) &&

```

```
121:         <Button size={"small"} color={"secondary"} onClick={
action(item.index)}>
122:             {actionName}
123:         </Button>
124:         {Boolean(action2) &&
125:         <Button size={"small"} color={"default"} onClick={ac
tion2(item.index)}>
126:             {action2Name}
127:         </Button>}
128:         </ExpansionPanelActions>
129:     </ExpansionPanel>
130:   )}
131: </div>
132: );
133: }
134:
135: }
136:
137: ContentList.propTypes = {
138:   classes: PropTypes.object.isRequired,
139:   list: PropTypes.arrayOf(PropTypes.object).isRequired,
140:   attributes: PropTypes.arrayOf(PropTypes.object).isRequired,
141:   action: PropTypes.func,
142:   actionName: PropTypes.string,
143:   action2: PropTypes.func,
144:   action2Name: PropTypes.string,
145:   notify: PropTypes.func.isRequired,
146:   className: PropTypes.string,
147: };
148:
149: export default withStyles(styles)(ContentList);
```

```

1:  /*
2:  *
3:  *   University di Pisa - Master's Degree in Computer Science and Networking
4:  *
5:  *   Final Project for the course of Peer to Peer Systems and Blockchains
6:  *
7:  *   Teacher: Prof. Laura Ricci
8:  *
9:  *   Candidate: Orlando Leombruni, matricola 475727
10:  *
11:  *   File: Charts.js
12:  *
13:  */
14:
15: import React from 'react';
16: import PropTypes from 'prop-types';
17: import {
18:   withStyles,
19:   Grid,
20:   Card,
21:   Button,
22:   CardHeader,
23:   CardContent,
24:   CardActions,
25:   Chip,
26:   Typography,
27:   Avatar,
28:   List,
29:   ListItem,
30:   ListItemText,
31:   Divider,
32:   Menu,
33:   MenuItem,
34: } from '@material-ui/core';
35: import {
36:   DonutSmall,
37:   FavoriteBorder,
38:   RecordVoiceOver,
39:   AttachMoney,
40: } from '@material-ui/icons';
41: import { ChartsStyle as styles } from '../styles/MaterialCustomStyles';
42:
43: /*
44:  * Feedbacks Class
45:  *
46:  * A React Component that lists the top Contents for each feedback category.
47:  */
48:
49: class Feedbacks extends React.Component {
50:
51:   render() {
52:     const { avg, appreciation, fairness, suggest } = this.props;
53:
54:     return (
55:       <div style={{flexGrow: 1}}>
56:         <Typography variant="subtitle1">
57:           Top rated contents
58:         </Typography>
59:         <List>
60:           <Typography variant="subtitle2">
61:             Average
62:           </Typography>
63:           <ListItem>
64:             <Avatar><DonutSmall/></Avatar>
65:             <ListItemText primary={avg}/>
66:           </ListItem>

```

```

67:           <Typography variant="subtitle2">
68:             Appreciation
69:           </Typography>
70:           <ListItem>
71:             <Avatar><FavoriteBorder/></Avatar>
72:             <ListItemText primary={appreciation}/>
73:           </ListItem>
74:           <Typography variant="subtitle2">
75:             Price fairness
76:           </Typography>
77:           <ListItem>
78:             <Avatar><AttachMoney/></Avatar>
79:             <ListItemText primary={fairness}/>
80:           </ListItem>
81:           <Typography variant="subtitle2">
82:             Likeliness to suggest to other users
83:           </Typography>
84:           <ListItem>
85:             <Avatar><RecordVoiceOver/></Avatar>
86:             <ListItemText primary={suggest}/>
87:           </ListItem>
88:         </List>
89:       </div>
90:     );
91:   }
92: }
93:
94: Feedbacks.propTypes = {
95:   avg: PropTypes.string.isRequired,
96:   appreciation: PropTypes.string.isRequired,
97:   fairness: PropTypes.string.isRequired,
98:   suggest: PropTypes.string.isRequired,
99: };
100:
101: // Default text shown while waiting for the actual results.
102: const defaultValue = "Loading...";
103:
104: /*
105:  * Charts Class
106:  *
107:  * A React Component that shows to the user various information about Contents in the
108:  * Catalog, such as the top rated
109:  * or the most recent ones.
110:  */
111:
112: class Charts extends React.Component {
113:
114:   constructor(props) {
115:     super(props);
116:     this.state.author.name = this.props.authors[0];
117:     this.state.genre.name = this.props.genres[0];
118:
119:     state = {
120:       global: {
121:         recentContents: [defaultValue, defaultValue, defaultValue, defaultValue,
122:         defaultValue],
123:         topRatedAvg: defaultValue,
124:         topRatedAppreciation: defaultValue,
125:         topRatedFairness: defaultValue,
126:         topRatedSuggest: defaultValue,
127:       },
128:       author: {
129:         name: "",
130:         mostRecent: defaultValue,
131:         mostViewed: defaultValue,

```

```
131:         topRatedAvg: defaultValue,
132:         topRatedAppreciation: defaultValue,
133:         topRatedFairness: defaultValue,
134:         topRatedSuggest: defaultValue,
135:     },
136:     genre: {
137:         name: "",
138:         mostRecent: defaultValue,
139:         mostViewed: defaultValue,
140:         topRatedAvg: defaultValue,
141:         topRatedAppreciation: defaultValue,
142:         topRatedFairness: defaultValue,
143:         topRatedSuggest: defaultValue,
144:     },
145:     authorAnchorEl: null,
146:     genreAnchorEl: null,
147: };
148:
149: /*
150:  * These functions control the behaviour of the "change author/genre" menus.
151:  */
152: showAuthors = event => {
153:     const target = event.currentTarget;
154:     this.setState(o => ({...o, authorAnchorEl: target}))
155: };
156:
157: showGenres = event => {
158:     const target = event.currentTarget;
159:     this.setState(o => ({...o, genreAnchorEl: target}))
160: };
161:
162: authorClose = () =>
163:     this.setState(o => ({...o, authorAnchorEl: null}));
164:
165: genreClose = () =>
166:     this.setState(o => ({...o, genreAnchorEl: null}));
167:
168: handleAuthorMenu = (event, index) => {
169:     this.setState(o => ({
170:         ...o,
171:         author: {
172:             name: this.props.authors[index],
173:             mostRecent: "Loading...",
174:             mostViewed: "Loading...",
175:             topRatedAvg: "Loading...",
176:             topRatedAppreciation: "Loading...",
177:             topRatedFairness: "Loading...",
178:             topRatedSuggest: "Loading...",
179:         }
180:     })), this.refreshAuthor);
181: };
182:
183: handleGenreMenu = (event, index) => {
184:     this.setState(o => ({
185:         ...o,
186:         genre: {
187:             name: this.props.genres[index],
188:             mostRecent: "Loading...",
189:             mostViewed: "Loading...",
190:             topRatedAvg: "Loading...",
191:             topRatedAppreciation: "Loading...",
192:             topRatedFairness: "Loading...",
193:             topRatedSuggest: "Loading...",
194:         }
195:     })), this.refreshGenre);
196: };
```

```
197:
198: /*
199:  * This function loads information about top/most recent contents of a specified
200:  genre.
201:  */
202: refreshGenre = () => {
203:     this.setState(o => ({...o, genreAnchorEl: null}));
204:     const {Catalog, web3, account} = this.props;
205:     const genre = web3.utils.asciiToHex(this.state.genre.name);
206:     Catalog.methods.getLatestByGenre(genre).call({from: account}).then(
207:         (result) => {
208:             this.setState(o => ({...o, genre: {...o.genre, mostRecent: web3.util
209: s.hexToAscii(result)}));
210:         },
211:         (error) => {
212:             console.log('Error in retrieving latest content with genre ${this.st
213: ate.genre.name}');
214:             this.setState(o => ({...o, genre: {...o.genre, mostRecent: "Error!"}
215: }));
216:         }
217:     );
218:     Catalog.methods.getMostPopularByGenre(genre).call({from: account}).then(
219:         (result) => {
220:             this.setState(o => ({...o, genre: {...o.genre, mostViewed: web3.util
221: s.hexToAscii(result)}));
222:         },
223:         (error) => {
224:             console.log('Error in retrieving most popular content with genre ${t
225: his.state.genre.name}');
226:             this.setState(o => ({...o, genre: {...o.genre, mostViewed: "Error!"}
227: }));
228:         }
229:     );
230:     Catalog.methods.getMostRatedByGenre(genre).call({from: account}).then(
231:         (result) => {
232:             this.setState(o => ({...o, genre: {...o.genre, topRatedAvg: web3.uti
233: ls.hexToAscii(result)}));
234:         },
235:         (error) => {
236:             console.log('Error in retrieving most rated (average) content with g
237: enre ${this.state.genre.name}');
238:             this.setState(o => ({...o, genre: {...o.genre, topRatedAvg: "Error!"}
239: }));
240:         }
241:     );
242:     ["Appreciation", "Fairness", "Suggest"].forEach((category, idx) => {
243:         Catalog.methods.getMostRatedByGenre(genre, idx+1).call({from: account}).
244: then(
245:         (result) => {
246:             this.setState(o => ({...o, genre: {...o.genre, ["topRated" + cat
247: egory]: web3.utils.hexToAscii(result)}));
248:         },
249:         (error) => {
250:             console.log('Error in retrieving most rated (${category}) conten
251: t with genre ${this.state.genre.name}');
252:             this.setState(o => ({...o, genre: {...o.genre, ["topRated" + cat
253: egory]: "Error!"}));
254:         }
255:     );
256: });
257:
258: /*
259:  * This function loads information about top/most recent contents of a specified
260:  author.
261:  */
```

```

248:   refreshAuthor = () => {
249:     this.setState(o => ({...o, authorAnchorEl: null}));
250:     const { Catalog, web3, account } = this.props;
251:     const author = web3.utils.asciiToHex(this.state.author.name);
252:     Catalog.methods.getLatestByAuthor(author).call({from: account}).then(
253:       (result) => {
254:         this.setState(o => ({...o, author: {...o.author, mostRecent: web3.ut
ils.hexToAscii(result)}));
255:       },
256:       (error) => {
257:         console.log(`Error in retrieving latest content with author ${this.s
tate.author.name}`);
258:         this.setState(o => ({...o, author: {...o.author, mostRecent: "Error!
"}));
259:       }
260:     );
261:     Catalog.methods.getMostPopularByAuthor(author).call({from: account}).then(
262:       (result) => {
263:         this.setState(o => ({...o, author: {...o.author, mostViewed: web3.ut
ils.hexToAscii(result)}));
264:       },
265:       (error) => {
266:         console.log(`Error in retrieving most popular content with author ${
this.state.author.name}`);
267:         this.setState(o => ({...o, author: {...o.author, mostViewed: "Error!
"}));
268:       }
269:     );
270:     Catalog.methods.getMostRatedByAuthor(author).call({from: account}).then(
271:       (result) => {
272:         this.setState(o => ({...o, author: {...o.author, topRatedAvg: web3.u
tils.hexToAscii(result)}));
273:       },
274:       (error) => {
275:         console.log(`Error in retrieving most rated (average) content with a
uthor ${this.state.author.name}`);
276:         this.setState(o => ({...o, author: {...o.author, topRatedAvg: "Error
!"}));
277:       }
278:     );
279:     ["Appreciation", "Fairness", "Suggest"].forEach((category, idx) => {
280:       Catalog.methods.getMostRatedByAuthor(author, idx+1).call({from: account
}).then(
281:         (result) => {
282:           this.setState(o => ({...o, author: {...o.author, ["topRated" + c
ategory]: web3.utils.hexToAscii(result)}));
283:         },
284:         (error) => {
285:           console.log(`Error in retrieving most rated (${category}) conten
t with author ${this.state.author.name}`);
286:           this.setState(o => ({...o, author: {...o.author, ["topRated" + c
ategory]: "Error!"}));
287:         }
288:       );
289:     });
290:   };
291:
292:   /*
293:   * This is a React state function that will be called only once, after the compo
nent is mounted in the virtual
294:   * DOM but before it gets rendered.
295:   *
296:   * In particular, this function asks the Catalog contract for information about
"global" (i.e. not author-
297:   * or genre-related) recent and top Contents, then loads the information about t
he default author's and genre's

```

```

298:   * Contents. (The default author and genre are usually the ones of the first Con
tent inserted in the Catalog).
299:   */
300:   componentDidMount() {
301:     const { Catalog, web3, account } = this.props;
302:     Catalog.methods.getNewContentsList(5).call({from: account}).then(
303:       (result) => {
304:         const list = result.map(item => web3.utils.hexToString(item));
305:         if (list.length < 5) {
306:           list.concat(new Array(5 - list.length).fill("N/A"));
307:         }
308:         this.setState(o => ({...o, global: {...o.global, recentContents: lis
t}));
309:       },
310:       (error) => {
311:         console.log("Error in retrieving recent contents");
312:         const list = [1,2,3,4,5].map(() => "Error!");
313:         this.setState(o => ({...o, global: {...o.global, recentContents: lis
t}));
314:       }
315:     );
316:     Catalog.methods.getMostRated().call({from: account}).then(
317:       (result) => {
318:         this.setState(o => ({...o, global: {...o.global, topRatedAvg: web3.u
tils.hexToString(result)}));
319:       },
320:       (error) => {
321:         console.log("Error in retrieving top rated content (average)");
322:         this.setState(o => ({...o, global: {...o.global, topRatedAvg: "Error
!"}));
323:       }
324:     );
325:     ["Appreciation", "Fairness", "Suggest"].forEach((category, idx) => {
326:       Catalog.methods.getMostRated(idx+1).call({from: account}).then(
327:         (result) => {
328:           this.setState(o => ({
329:             ...o,
330:             global: {
331:               ...o.global,
332:               ["topRated" + category]: web3.utils.hexToString(result),
333:             }
334:           }));
335:         },
336:         (error) => {
337:           console.log(`Error in retrieving top rated content (${category})
`);
338:           this.setState(o => ({...o, global: {...o.global, ["topRated" + c
ategory]: "Error!"}));
339:         }
340:       );
341:     });
342:     this.refreshAuthor();
343:     this.refreshGenre();
344:   }
345:
346:   render() {
347:     const { classes } = this.props;
348:     const {
349:       global,
350:       author,
351:       genre,
352:       authorAnchorEl,
353:       genreAnchorEl,
354:     } = this.state;
355:
356:     const authorOpen = Boolean(authorAnchorEl), genreOpen = Boolean(genreAnchorE

```

```
1);
357:
358:   return (
359:     <Grid container className={classes.root} spacing={24}>
360:       <Grid item xs={12}>
361:         <Card className={classes.card}>
362:           <CardHeader
363:             avatar={<Avatar className={classes.avatarRed}>O</Avatar>
364:               title="Overview"
365:               subheader={"Information about all contents"}
366:             />
367:           <CardContent>
368:             <div style={{display: 'flex', flexDirection: 'row'}}>
369:               <div style={{flexGrow: 1}}>
370:                 <Typography variant={"subtitle1"}>
371:                   Latest contents added to the catalog
372:                 </Typography>
373:                 <List>
374:                   {global.recentContents.map((item, idx) => (
375:                     <ListItem key={idx}>
376:                       <Avatar>{idx+1}</Avatar>
377:                       <ListItemText primary={item}/>
378:                     </ListItem>
379:                   ))}
380:                 </List>
381:               </div>
382:               <Divider/>
383:               <Feedbacks
384:                 avg={global.topRatedAvg}
385:                 appreciation={global.topRatedAppreciation}
386:                 fairness={global.topRatedFairness}
387:                 suggest={global.topRatedSuggest}
388:               />
389:             </div>
390:           </CardContent>
391:         </Card>
392:       </Grid>
393:       <Grid item xs={6}>
394:         <Card className={classes.card}>
395:           <CardHeader
396:             avatar={<Avatar className={classes.avatarGreen}>A</Avatar>
397:               title="Author"
398:               subheader={"Information about contents by a specific author"}
399:               action={
400:                 <Chip
401:                   onClick={this.showAuthors}
402:                   variant={"outlined"}
403:                   color={"secondary"}
404:                   label={author.name}
405:                   avatar={<Avatar>{author.name[0].toUpperCase()}</Avatar>} />
406:                 }
407:               />
408:             <Menu
409:               id={"-menu"}
410:               anchorEl={authorAnchorEl}
411:               open={authorOpen}
412:               onClose={this.authorClose}
413:               PaperProps={{
414:                 style: {
415:                   maxHeight: 48 * 4.5,
416:                   width: 200,
417:                 }
418:               }}
419:             >
420:               {this.props.authors.map((item, idx) => (
421:                 <MenuItem key={item} onClick={event => this.handleAuthorMenu(event, idx)}>
422:                   {item}
423:                 </MenuItem>
424:               ))}
425:             </Menu>
426:           <CardContent>
427:             <Grid container>
428:               <Grid item xs={6}>
429:                 <Typography variant={"subtitle1"}>
430:                   Latest
431:                 </Typography>
432:                 <List>
433:                   <ListItem>
434:                     <Avatar>{author.mostRecent[0].toUpperCase()}
435:                     <ListItemText primary={author.mostRecent[0].name}>
436:                   </ListItem>
437:                 </List>
438:               </Grid>
439:               <Grid item xs={6}>
440:                 <Typography variant={"subtitle1"}>
441:                   Most viewed
442:                 </Typography>
443:                 <List>
444:                   <ListItem>
445:                     <Avatar>{author.mostViewed[0].toUpperCase()}
446:                     <ListItemText primary={author.mostViewed[0].name}>
447:                   </ListItem>
448:                 </List>
449:               </Grid>
450:             </Grid>
451:             <Grid item xs={12}>
452:               <Feedbacks
453:                 avg={author.topRatedAvg}
454:                 appreciation={author.topRatedAppreciation}
455:                 fairness={author.topRatedFairness}
456:                 suggest={author.topRatedSuggest}/>
457:             </Grid>
458:           </CardContent>
459:           <CardActions style={{display: "block", width: "100%"}}>
460:             <Button
461:               className={classes.button}
462:               variant={"contained"}
463:               onClick={() => this.props.addSubAuthor(author.name)}
464:             >
465:               Notify about new content by {author.name}
466:             </Button>
467:           </CardActions>
468:         </Card>
469:       </Grid>
470:     </Grid>
471:   </Grid item xs={6}>
472:     <Card className={classes.card}>
473:       <CardHeader
474:         avatar={<Avatar className={classes.avatarBlue}>G</Avatar>
475:           title="Genre"
476:           subheader={"Information about contents of a specific genre"}
477:           action={
478:             <Chip
```



```
476:         onClick={this.showGenres}
477:         variant={"outlined"}
478:         color={"secondary"}
479:         label={genre.name}
480:         avatar={<Avatar>{genre.name[0].toUpperCase()}</A
vatar>}</>
481:     }
482:   </>
483:   <Menu
484:     id={"-menu"}
485:     anchorEl={genreAnchorEl}
486:     open={genreOpen}
487:     onClose={this.genreClose}
488:     PaperProps={{
489:       style: {
490:         maxHeight: 48 * 4.5,
491:         width: 200,
492:       }
493:     }}>
494:     {this.props.genres.map((item, idx) => (
495:       <MenuItem key={item} onClick={event => this.handleGe
nreMenu(event, idx)}>
496:         {item}
497:       </MenuItem>
498:     ))}
499:   </Menu>
500:   <CardContent>
501:     <Grid container>
502:       <Grid item xs={6}>
503:         <Typography variant={"subtitle1"}>
504:           Latest
505:         </Typography>
506:         <List>
507:           <ListItem>
508:             <Avatar>{genre.mostRecent[0].toUpperCase
()}</Avatar>
509:             <ListItemText primary={genre.mostRecent}
/>
510:           </ListItem>
511:         </List>
512:       </Grid>
513:       <Grid item xs={6}>
514:         <Typography variant={"subtitle1"}>
515:           Most viewed
516:         </Typography>
517:         <List>
518:           <ListItem>
519:             <Avatar>{genre.mostViewed[0].toUpperCase
()}</Avatar>
520:             <ListItemText primary={genre.mostViewed}
/>
521:           </ListItem>
522:         </List>
523:       </Grid>
524:       <Grid item xs={12}>
525:         <Feedbacks
526:           avg={genre.topRatedAvg}
527:           appreciation={genre.topRatedAppreciation}
528:           fairness={genre.topRatedFairness}
529:           suggest={genre.topRatedSuggest}>
530:       </Grid>
531:     </Grid>
532:   </CardContent>
533:   <CardActions style={{display: "block", width: "100%"}}>
534:     <Button
535:       variant={"contained"}
536:       onClick={() => this.props.addSubGenre(genre.name)}>
537:       Notify about new content of genre {genre.name}
538:     </Button>
539:   </CardActions>
540: </Card>
541: </Grid>
542: </Grid>
543: );
544: };
545: }
546: }
547:
548: Charts.propTypes = {
549:   classes: PropTypes.object.isRequired,
550:   authors: PropTypes.array.isRequired,
551:   genres: PropTypes.array.isRequired,
552:   addSubAuthor: PropTypes.func.isRequired,
553:   addSubGenre: PropTypes.func.isRequired,
554:   web3: PropTypes.object.isRequired,
555:   Catalog: PropTypes.object.isRequired,
556:   account: PropTypes.string.isRequired,
557: };
558:
559: export default withStyles(styles)(Charts);
```



```
1: /*
2: *
3: * University di Pisa - Master's Degree in Computer Science and Networking
4: *
5: * Final Project for the course of Peer to Peer Systems and Blockchains
6: *
7: * Teacher: Prof. Laura Ricci
8: *
9: * Candidate: Orlando Leombruni, matricola 475727
10: *
11: * File: NewContentForm.js
12: *
13: */
14:
15: import React from 'react';
16: import PropTypes from 'prop-types';
17: import BN from 'bn.js';
18: import {
19:   withStyles,
20:   Card,
21:   CardContent,
22:   CardActions,
23:   TextField,
24:   Button,
25:   MenuItem,
26:   CircularProgress,
27: } from '@material-ui/core';
28: import { NewContentFormStyle as styles } from "../styles/MaterialCustomStyles";
29:
30: /*
31: * NewContentForm Class
32: *
33: * A React Component that provides the user with a form for creating a new managed C
content.
34: */
35: class NewContentForm extends React.Component {
36:
37:   constructor(props) {
38:     super(props);
39:     this.currencies = [
40:       {label: "wei", value: "1"},
41:       {label: "kwei", value: "1000"},
42:       {label: "Mwei", value: "1000000"},
43:       {label: "Gwei", value: "1000000000"},
44:       {label: "szabo", value: "1000000000000"},
45:       {label: "finney", value: "1000000000000000"},
46:       {label: "ether", value: "1000000000000000000"}
47:     ];
48:     this.state = {
49:       name: "",
50:       genre: "",
51:       author: "",
52:       price: 1,
53:       currency: this.currencies[0].value,
54:     };
55:   }
56:
57:   // Handles user changes in the form.
58:   handleChange = event => {
59:     const { target } = event;
60:     this.setState(oldState => ({...oldState, [target.name]: target.value}));
61:   };
62:
63:   // Validates the form if all input is correct.
64:   checkForm = () => {
65:     const { name, genre, author, price, currency } = this.state;
```

```
66:     if (name && genre && author && (+price > 0) && currency)
67:       this.props.submitForm(name, genre, author, new BN(currency).muln(+price)
68:   );
69:
70:   render() {
71:     const { classes, loading } = this.props;
72:     const { name, genre, author, price, currency } = this.state;
73:     return (
74:       <Card className={classes.root}>
75:         <CardContent>
76:           <form onSubmit={this.checkForm} style={{width: "100%"}}>
77:             <div className={classes.formDiv}>
78:               <TextField
79:                 className={classes.textField}
80:                 name={ "name" }
81:                 label={ "Content description" }
82:                 InputLabelProps={{shrink: true}}
83:                 required
84:                 margin={ "normal" }
85:                 value={this.state.name}
86:                 type={ "string" }
87:                 onChange={this.handleChange}
88:               />
89:               <TextField
90:                 className={classes.textField}
91:                 name={ "genre" }
92:                 label={ "Genre" }
93:                 InputLabelProps={{shrink: true}}
94:                 required
95:                 margin={ "normal" }
96:                 type={ "string" }
97:                 value={genre}
98:                 onChange={this.handleChange}
99:               />
100:               <TextField
101:                 className={classes.textField}
102:                 name={ "author" }
103:                 label={ "Author" }
104:                 InputLabelProps={{shrink: true}}
105:                 required
106:                 margin={ "normal" }
107:                 type={ "string" }
108:                 value={author}
109:                 onChange={this.handleChange}
110:               />
111:               <div>
112:                 <TextField
113:                   className={classes.priceField}
114:                   name={ "price" }
115:                   label={ "Price" }
116:                   InputLabelProps={{shrink: true}}
117:                   inputProps={{min: 1, step: 1, style: {textAlign: "right"}}}
118:                   required
119:                   error={+price <= 0}
120:                   type={ "number" }
121:                   margin={ "normal" }
122:                   value={price}
123:                   onChange={this.handleChange}
124:                 />
125:                 <TextField
126:                   select
127:                   required
128:                   InputLabelProps={{shrink: true}}
129:                   className={classes.currency}
130:                   name={ "currency" }
```

```
131:         label={"Unit"}
132:         margin={"normal"}
133:         value={currency}
134:         SelectProps={{
135:           MenuProps: {
136:             className: classes.menu,
137:           }
138:         }}
139:         onChange={this.handleChange}>
140:         {this.currencies.map((option, item) =>
141:           <MenuItem key={item} value={option.value}>
142:             {option.label}
143:           </MenuItem>
144:         )}
145:       </TextField>
146:     </div>
147:   </div>
148: </form>
149: </CardContent>
150: <CardActions style={{display: "block", width: "100%}}>
151:   <div className={classes.wrapper}>
152:     <Button
153:       variant={"contained"}
154:       className={classes.button}
155:       color={"secondary"}
156:       disabled={! (name && genre && author && (+price > 0) && curr
ency)) || loading}
157:       onClick={this.checkForm}>Add</Button>
158:       {loading && <CircularProgress size={24} className={classes.b
uttonProgressPrimary}/>}
159:     </div>
160:   </CardActions>
161:
162:   </Card>
163: );
164: }
165: }
166:
167: NewContentForm.propTypes = {
168:   classes: PropTypes.object.isRequired,
169:   loading: PropTypes.bool.isRequired,
170:   submitForm: PropTypes.func.isRequired,
171: };
172:
173: export default withStyles(styles)(NewContentForm);
```

```

1:  /*
2:  *
3:  * University di Pisa - Master's Degree in Computer Science and Networking
4:  *
5:  * Final Project for the course of Peer to Peer Systems and Blockchains
6:  *
7:  * Teacher: Prof. Laura Ricci
8:  *
9:  * Candidate: Orlando Leombruni, matricola 475727
10:  *
11:  * File: FeedbackForm.js
12:  *
13:  */
14:
15: import React from 'react';
16: import PropTypes from 'prop-types';
17: import {
18:   withStyles,
19:   Dialog,
20:   DialogTitle,
21:   DialogContent,
22:   DialogContentText,
23:   DialogActions,
24:   TextField,
25:   MenuItem,
26:   Button
27: } from '@material-ui/core';
28: import { FeedbackFormStyle as styles } from "../styles/MaterialCustomStyles";
29:
30: /*
31:  * FeedbackForm Class
32:  *
33:  * A React Component that allows the user to input feedback (as three numerical grad
34:  es) for the Content.
35:  */
36: class FeedbackForm extends React.Component {
37:   state = {
38:     appreciation: -1,
39:     fairness: -1,
40:     suggest: -1
41:   };
42:   // Handles changes in the form.
43:   changeFields = event => {
44:     const { target } = event;
45:     this.setState(oldState => ({...oldState, [target.name]: target.value}))
46:   };
47:
48:   render() {
49:
50:     const { classes, cancel, confirm } = this.props;
51:
52:     const categories = [
53:       {
54:         label: "0 (worst)",
55:         value: 0,
56:       },
57:       {
58:         label: "1",
59:         value: 1,
60:       },
61:       {
62:         label: "2",
63:         value: 2,
64:       },
65:       {

```

```

66:         label: "3",
67:         value: 3,
68:       },
69:       {
70:         label: "4",
71:         value: 4,
72:       },
73:       {
74:         label: "5 (best)",
75:         value: 5,
76:       }
77:     ];
78:
79:     return (
80:       <Dialog
81:         open={true}
82:         keepMounted
83:         onBackdropClick={cancel}
84:         onEscapeKeyDown={cancel} >
85:         <DialogTitle>Rate your experience</DialogTitle>
86:         <DialogContent>
87:           <ul>
88:             <li>
89:               <div>
90:                 <DialogContentText>How much did you enjoy the conten
91:                 <TextField
92:                   id={"appreciation"}
93:                   select
94:                   required
95:                   name={"appreciation"}
96:                   className={classes.textField}
97:                   value={this.state.appreciation}
98:                   onChange={this.changeFields}
99:                   SelectProps={{
100:                     MenuProps: {
101:                       className: classes.menu,
102:                     }
103:                   }}
104:                   margin={"normal"} >
105:                   {categories.map((item) => (
106:                     <MenuItem key={item.value} value={item.value
107:                     ))}
108:                 </TextField>
109:               </div>
110:             </li>
111:             <li>
112:               <div>
113:                 <DialogContentText>Do you think that the price was f
114:                 <TextField
115:                   id={"fairness"}
116:                   select
117:                   required
118:                   name={"fairness"}
119:                   className={classes.textField}
120:                   value={this.state.fairness}
121:                   onChange={this.changeFields}
122:                   SelectProps={{
123:                     MenuProps: {
124:                       className: classes.menu,
125:                     }
126:                   }}
127:                   margin={"normal"} >
128:                   {categories.map((item) => (

```

```
129:                                     <MenuItem key={item.value} value={item.value
}>{item.label}</MenuItem>
130:                                     )))
131:                                     </TextField>
132:                               </div>
133:         </li>
134:         <li>
135:           <div>
136:             <DialogContentText>How likely are you to recommend t
his content to others?</DialogContentText>
137:             <TextField
138:               id={"suggest"}
139:               select
140:               required
141:               name={"suggest"}
142:               className={classes.textField}
143:               value={this.state.suggest}
144:               onChange={this.changeFields}
145:               SelectProps={{
146:                 MenuProps: {
147:                   className: classes.menu,
148:                 }
149:               }}
150:               margin={"normal"} >
151:               {categories.map((item) => (
152:                 <MenuItem key={item.value} value={item.value
}>{item.label}</MenuItem>
153:                 )))
154:               </TextField>
155:             </div>
156:           </li>
157:         </ul>
158:       </DialogContent>
159:       <DialogActions>
160:         <Button
161:           color={"inherit"}
162:           className={classes.button}
163:           onClick={cancel}>Cancel</Button>
164:         <Button
165:           color={"secondary"}
166:           disabled={!((this.state.appreciation > -1) && (this.state.fa
irness > -1) && (this.state.suggest > -1))}
167:           className={classes.button}
168:           onClick={confirm(this.state)}>Ok</Button>
169:       </DialogActions>
170:     </Dialog>
171:   );
172: }
173: }
174:
175: FeedbackForm.propTypes = {
176:   classes: PropTypes.object.isRequired,
177:   cancel: PropTypes.func.isRequired,
178:   confirm: PropTypes.func.isRequired,
179: };
180:
181: export default withStyles(styles)(FeedbackForm);
```

```

1:  /*
2:  *
3:  *   University di Pisa - Master's Degree in Computer Science and Networking
4:  *
5:  *   Final Project for the course of Peer to Peer Systems and Blockchains
6:  *
7:  *   Teacher: Prof. Laura Ricci
8:  *
9:  *   Candidate: Orlando Leombruni, matricola 475727
10:  *
11:  *   File: CreatorApp.js
12:  *
13:  */
14:
15: import React from 'react';
16: import { withStyles } from '@material-ui/core';
17: import PropTypes from 'prop-types';
18: import ReactNotification from 'react-notifications-component';
19: import AppDrawer from './AppDrawer';
20: import ContentList from './ContentList';
21: import TransactionConfirmation from './TransactionConfirmation';
22: import NewContentForm from './NewContentForm';
23: import {
24:   List,
25:   ListItemIcon,
26:   ListItemText,
27:   ListItem,
28:   Tooltip,
29:   Divider,
30:   TextField,
31:   Button,
32:   Dialog,
33:   DialogTitle,
34:   DialogContent,
35:   DialogContentText,
36:   DialogActions,
37:   Slide,
38: } from '@material-ui/core';
39: import {
40:   List as ListIcon,
41:   LibraryAdd,
42:   Delete,
43:   Update,
44:   Publish,
45: } from '@material-ui/icons';
46: import CopyToClipboard from 'react-copy-to-clipboard';
47: import 'react-notifications-component/dist/theme.css';
48: import 'animate.css';
49: import json from '../solidity/compiled.json';
50: import { getTransactionParameters, makeTitle, prettifyWei } from '../Utils';
51: import { CreatorAppStyle as styles } from '../styles/MaterialCustomStyles';
52:
53: /*
54:  * "Slide up" transition for the animation of a React component.
55:  */
56: const Transition = (props) =>
57:   <Slide direction="up" {...props} />;
58:
59: /*
60:  * Enum-like object for the various sections of the application.
61:  */
62: const Panels = Object.freeze({
63:   LIST_CONTENT: Symbol("list"),
64:   NEW_CONTENT: Symbol("new"),
65:   DEL_CONTENT: Symbol("delete"),
66:   PUBLISH: Symbol("publish"),

```

```

67: });
68:
69: /*
70:  * CreatorApp Class
71:  *
72:  * A React Component that represents the entire Creators portion for the app.
73:  * It allows a Creator to publish new contents, delete existing ones, and request mo
netization for their work.
74:  */
75: class CreatorApp extends React.Component {
76:
77:   constructor(props) {
78:     super(props);
79:     this.notificationRef = React.createRef();
80:     this.notify = this.notify.bind(this);
81:     this.web3 = this.props.web3;
82:     this.props.manageCenter(false);
83:     const abi = JSON.parse(json.contracts["Catalog.sol:Catalog"].abi);
84:     this.Catalog = new this.web3.eth.Contract(abi, this.props.catalog);
85:     this.newContent = {};
86:     this.registered = [];
87:   }
88:
89:   state = {
90:     panel: Panels.LIST_CONTENT,
91:     contentList: [],
92:     transactionProps: {},
93:     transactPopup: false,
94:     error: false,
95:     selectedContent: "",
96:     creatingContent: false,
97:     publishingContent: false,
98:     closing: false,
99:     createdContent: "",
100:    createdPopup: false,
101:    loading: false,
102:    monetize: false,
103:  };
104:
105:  // Changes the current panel (section of the app).
106:  changePanel = (panel) => () => {
107:    this.setState(oldState => ({...oldState, panel: panel}));
108:  };
109:
110:  // Displays a notification in the upper right portion of the screen.
111:  notify(title, message, type, time) {
112:    this.notificationRef.current.addNotification({
113:      title,
114:      message,
115:      type,
116:      insert: "top",
117:      container: "top-right",
118:      animationIn: ["animated", "fadeIn"],
119:      animationOut: ["animated", "fadeOut"],
120:      dismiss: { duration: time || 2000 },
121:      dismissable: { click: true },
122:    });
123:  }
124:
125:  // Callback function to handle user input for the publishing/deletion of a conte
nt.
126:  handleContent = (event) => {
127:    const { target } = event;
128:    this.setState(oldState => ({...oldState, selectedContent: target.value}));
129:  };
130:

```

```

131: // Closes the "new Content created" popup dialog.
132: handleDialog = () =>
133:   this.setState(oldState => ({...oldState, createdPopup: false, createdContent
: ""}));
134:
135: /*
136:  * Begins the deletion of a ContentManagementBase contract, checking the correct
ness of the operation
137:  * (i.e. if the content exists and is owned by the current selected EOA) and est
imating the gas cost.
138:  */
139: initiateDelete = () => {
140:   let error = (this.state.selectedContent === "");
141:   const abi = JSON.parse(json.contracts["ContentManagementBase.sol:ContentMana
gementBase"].abi);
142:   const CMB = new this.web3.eth.Contract(abi);
143:   try {
144:     CMB.options.address = this.state.selectedContent;
145:   } catch (err) {
146:     error = true;
147:     console.log(err);
148:   }
149:   if (error) {
150:     this.setState(oldState => ({...oldState, error: true}));
151:     this.notify("Error", "Invalid Content address", "danger");
152:   } else {
153:     this.setState(oldState => ({...oldState, closing: true, error: false}));
154:     getTransactionParameters(this.web3, CMB.methods.closeContract(), this.pr
ops.account).then(
155:       (result) => this.setState(o => ({...o, transactionProps: result, tra
nsactPopup: true})),
156:       (error) => {
157:         console.log("Error in getting parameters for initiateDelete");
158:         console.log(error);
159:         this.notify("Error", error.stackTrace.message.toString(), "dange
r");
160:       }
161:     );
162:   }
163: };
164:
165: /*
166:  * Invoked when the user confirms the transaction (after reviewing it in an info
rnative dialog), this function
167:  * effectively closes and deletes a content, checking afterwards that the operat
ion was successful.
168:  */
169: confirmDelete = () => {
170:   const abi = JSON.parse(json.contracts["ContentManagementBase.sol:ContentMana
gementBase"].abi);
171:   const CMB = new this.web3.eth.Contract(abi, this.state.selectedContent);
172:   CMB.methods.closeContract().send({
173:     from: this.props.account,
174:     gasPrice: this.state.transactionProps.gasPrice,
175:     gas: this.state.transactionProps.gas,
176:   }).then(
177:     (result) => {
178:       CMB.methods.isActiveContent().call({from: this.props.account}).then(
179:         () => this.notify("Error", "Contract NOT closed! Are you the own
er?", "danger"),
180:         () => this.notify("Success!", "Content contract was closed!", "s
uccess")
181:       );
182:       this.setState(o => ({...o, loading: false}));
183:     },
184:     (error) => {

```

```

185:     console.log(error);
186:     this.notify("Error", "Could not close contract...", "danger");
187:     this.setState(o => ({...o, loading: false}));
188:   }
189:   );
190:   this.setState(oldState => ({...oldState, loading: true, closing: false, tran
sactionProps: {}, transactPopup: false}));
191: };
192:
193: // Starts a request for monetization of a Content to the Catalog, estimating the
gas cost.
194: requestMonetization = (key) => () => {
195:   const address = this.state.contentList[key].address;
196:   getTransactionParameters(this.web3, this.Catalog.methods.collectPayment(addr
ess), this.props.account).then(
197:     (result) => this.setState(o => ({...o, transactionProps: result, transac
tPopup: true})),
198:     (error) => {
199:       console.log("Error in getting parameters for requestMonetization");
200:       console.log(error.stackTrace);
201:       switch (error.type) {
202:         case "estimateGas":
203:           this.notify("Error", "Could not request payment, maybe there
's no payment available", "danger");
204:           break;
205:         case "getBalance":
206:           this.notify("Error", "Could not check account balance, maybe
connection is down", "danger");
207:           break;
208:         case "getGasPrice":
209:           this.notify("Error", "Could not check gas price from the blo
ckchain, maybe connection is down", "danger");
210:           break;
211:         default:
212:           this.notify("Error", error.stackTrace.message.toString(), "d
anger");
213:       }
214:       this.setState(o => ({...o, monetize: ""}));
215:     }
216:   );
217:   this.setState(o => ({...o, monetize: address}));
218: };
219:
220: /*
221:  * Invoked when the user confirms the transaction (after reviewing it in an info
rnative dialog), this function
222:  * effectively sends a request for payment to the Catalog and (upon success) sho
ws the Creator their new balance.
223:  */
224: confirmMonetize = () => {
225:   this.Catalog.methods.collectPayment(this.state.monetize).send({
226:     from: this.props.account,
227:     gas: this.state.transactionProps.gas,
228:     gasPrice: this.state.transactionProps.gasPrice
229:   }).then(
230:     () => {
231:       this.web3.eth.getBalance(this.props.account).then(
232:         (result) => {
233:           this.notify("Success!",
234:             `You've been paid for views of your content, balance is
now ${prettyWei(result.toString())}`,
235:             "success");
236:         },
237:         (error) => {
238:           this.notify("Success",
239:             `You've been paid for views of your content, but could n

```



```

ot fetch new balance`,
240:             "material");
241:         }
242:     });
243:     this.setState(o => ({...o, loading: false}));
244: },
245: (error) => {
246:     console.log("Could not monetize", error);
247:     this.notify("Error", "There's been a problem with monetization, chec
k console logs", "danger");
248:     this.setState(o => ({...o, loading: false}));
249: }
250: );
251: this.setState(o => ({...o, transactionProps: {}, transactPopup: false, monet
ize: "", loading: true}));
252: };
253:
254: // Begins the creation of a new ContentManagementBase contract, estimating its g
as cost.
255: createContent = (name, genre, author, price) => {
256:     const abi = JSON.parse(json.contracts["ContentManagementBase.sol:ContentMana
gementBase"].abi);
257:     const bin = "0x" + json.contracts["ContentManagementBase.sol:ContentManageme
ntBase"].bin;
258:     const CMB = new this.web3.eth.Contract(abi);
259:     this.newContent = CMB.deploy({
260:         data: bin,
261:         arguments: [
262:             this.web3.utils.asciiToHex(name),
263:             this.web3.utils.asciiToHex(genre),
264:             this.web3.utils.asciiToHex(author),
265:             price.toString(),
266:         ]});
267:     getTransactionParameters(this.web3, this.newContent, this.props.account).the
n(
268:         (result) => this.setState(o => ({...o, transactionProps: result, transac
tPopup: true})),
269:         (error) => {
270:             console.log("Error in getting parameters for createContent");
271:             console.log(error.stackTrace);
272:             this.notify("Error", 'Could not complete operation (error in ${error
.type})', "danger");
273:             this.setState(o => ({...o, creatingContent: false}));
274:         }
275:     );
276:     this.setState(oldState => ({...oldState, closing: false, creatingContent: tr
ue}));
277: };
278:
279: /*
280:  * Invoked when the user confirms the transaction (after reviewing it in an info
rnative dialog), this function
281:  * effectively creates a new ContentManagementBase contract and checks if it has
been correctly deployed.
282:  *
283:  * This function WILL NOT publish the new Content on the Catalog, but will ask t
he user if they want to.
284:  */
285: confirmCreate = () => {
286:     this.newContent.send({
287:         from: this.props.account,
288:         gasPrice: this.state.transactionProps.gasPrice,
289:         gas: this.state.transactionProps.gas,
290:     }).then(
291:         (result) => {
292:             result.methods.isActiveContent().call({from: this.props.account}).th

```

```

en(
293:         () => {
294:             this.setState(oldState => ({...oldState, createdContent: res
ult.options.address, createdPopup: true, creatingContent: false}));
295:             this.newContent = result;
296:         },
297:         () => {
298:             this.notify("Error", "Contract not created...", "danger");
299:             this.setState(o => ({...o, loading: false, creatingContent:
false}));
300:         }
301:     )
302: },
303: (error) => {
304:     console.log(error);
305:     this.notify("Error", "Could not create contract...", "danger");
306:     this.setState(o => ({...o, loading: false, creatingContent: false}));
307: }
308: );
309: this.setState(oldState => ({...oldState, loading: true, closing: false, tran
sactionProps: {}, transactPopup: false}));
310: };
311:
312: /*
313:  * Begins the publishing of a Content (either a freshly-created or a user-provid
ed one) to the Catalog,
314:  * estimating the operation's gas cost.
315:  */
316: publishContent = () => {
317:     this.setState(oldState => ({...oldState, publishingContent: true}));
318:     getTransactionParameters(this.web3, this.newContent.methods.publish(this.pro
ps.catalog), this.props.account).then(
319:         (result) => this.setState(o => ({...o, transactionProps: result, transac
tPopup: true})),
320:         (error) => {
321:             console.log("Error in getting parameters for publishContent");
322:             console.log(error);
323:             if (error.type === "estimateGas")
324:                 this.notify("Error", "Could not publish content, maybe it's alre
ady in the catalog?", "danger");
325:             else
326:                 this.notify("Error", 'Could not complete operation (error in ${e
rror.type})', "danger");
327:             this.setState(o => ({...o, loading: false, publishingContent: false,
createdPopup: false, createdContent: ""}));
328:         }
329:     );
330: };
331:
332: /*
333:  * Invoked when the user confirms the transaction (after reviewing it in an info
rnative dialog), this function
334:  * effectively publishes a Content to the Catalog.
335:  */
336: confirmPublish = () => {
337:     this.newContent.methods.publish(this.props.catalog).send({
338:         from: this.props.account,
339:         gas: this.state.transactionProps.gas,
340:         gasPrice: this.state.transactionProps.gasPrice,
341:     }).then(
342:         (result) => {
343:             this.notify("Success!", 'Contract published! Refresh the list to see
it', "success");
344:             this.setState(o => ({...o, loading: false}));
345:         },

```

```

346:         (error) => {
347:             this.notify("Error", "Error in publishing contract!", "danger");
348:             this.setState(o => ({...o, loading: false}));
349:         }
350:     );
351:     this.newContent = {};
352:     this.setState(oldState => ({...oldState, loading: true, transactionProps:{},
publishingContent: false, transactPopup: false, createdPopup: false, createdContent: ""}))
353:     };
354:
355:     // Renders the drawer menu.
356:     menu = () =>
357:         <List>
358:             <Tooltip title="List this account's Contents" placement="right">
359:                 <ListItem button key="list" onClick={this.changePanel(Panels.LIST_
CONTENT)}>
360:                     <ListItemIcon><ListIcon /></ListItemIcon>
361:                     <ListItemText secondary="My Contents"/>
362:                 </ListItem>
363:             </Tooltip>
364:             <Tooltip title="Create a new Content" placement="right">
365:                 <ListItem button key="new" onClick={this.changePanel(Panels.NEW_CO
CONTENT)}>
366:                     <ListItemIcon><LibraryAdd /></ListItemIcon>
367:                     <ListItemText secondary="New Content"/>
368:                 </ListItem>
369:             </Tooltip>
370:             <Tooltip title="Publish an already-existing content" placement="right
">
371:                 <ListItem button key="publish" onClick={this.changePanel(Panels.PU
BLISH)}>
372:                     <ListItemIcon><Publish/></ListItemIcon>
373:                     <ListItemText secondary="Publish a Content"/>
374:                 </ListItem>
375:             </Tooltip>
376:             <Divider/>
377:             <Tooltip title="Delete a Content from the Catalog" placement="right">
378:                 <ListItem button key="delete" onClick={this.changePanel(Panels.DEL
_CONTENT)}>
379:                     <ListItemIcon><Delete/></ListItemIcon>
380:                     <ListItemText secondary="Delete a Content"/>
381:                 </ListItem>
382:             </Tooltip>
383:             {(this.state.panel === Panels.LIST_CONTENT) && <Divider/>}
384:             {(this.state.panel === Panels.LIST_CONTENT) &&
385:                 <Tooltip title="Update Content list" placement="right">
386:                     <ListItem button key="update" onClick={this.updateContentList}
>
387:                         <ListItemIcon><Update/></ListItemIcon>
388:                         <ListItemText secondary="Update List"/>
389:                     </ListItem>
390:                 </Tooltip>
391:             </List>;
392:
393:     // Renders the app bar title (depending on the current panel).
394:     renderTitle = () => {
395:         const { classes } = this.props;
396:         switch (this.state.panel) {
397:             case Panels.LIST_CONTENT: return [makeTitle("My Contents", classes.grow
)];
398:             case Panels.NEW_CONTENT: return [makeTitle("Add a new Content", classes.
grow)];
399:             case Panels.PUBLISH: return [makeTitle("Publish a Content", classes.grow
)];
400:             case Panels.DEL_CONTENT: return [makeTitle("Delete a Content", classes.g

```

```

row)];
401:         default: return [makeTitle("", classes.grow)];
402:     }
403: };
404:
405: /*
406:  * Updates the list of content belonging to the current Creator, also registerin
g callbacks for the "Content
407:  * Provided" Solidity events.
408:  */
409:     updateContentList = () => {
410:         this.Catalog.methods.getCreatorContentList().call({from: this.props.account}
)
411:         .then(
412:             (result) => {
413:                 if (!result[0] || !result[1]) {
414:                     this.notify("Error", "Couldn't fetch Catalog content", "dang
er");
415:                     console.log(result);
416:                 }
417:                 else {
418:                     const list = [];
419:                     for (let i = 0; i < result[0].length; i++) {
420:                         list.push({
421:                             index: i,
422:                             description: this.web3.utils.hexToAscii(result[0][i]
),
423:                             address: result[1][i],
424:                         });
425:                         this.registerProvide(result[1][i]);
426:                     }
427:                     this.setState(oldState => ({...oldState, contentList: list})
);
428:                     this.notify("List updated!", "List of content has been updat
ed", "material");
429:                 }
430:             },
431:             (error) => {
432:                 this.notify("Error", "Couldn't fetch Catalog content", "danger")
;
433:                 console.log(error);
434:             }
435:         )
436:     };
437:
438:     // Registers a callback for the "Provide Content" event relative to the selected
Content.
439:     registerProvide = (address) => {
440:         if (!this.registered.some(item => item === address)) {
441:             const abi = JSON.parse(json.contracts["ContentManagementBase.sol:Conte
ntManagementBase"].abi);
442:             const CMB = new this.web3.eth.Contract(abi, address);
443:
444:             CMB.events.ProvideContent({filter: {content: address}, fromBlock: "lates
t"}).
445:                 .on('error', error => console.log("Error in event Providecontent", e
rror))
446:                 .on('data', event => {
447:                     const content = this.state.contentList.find((item) => item.addre
ss === address);
448:                     if (content) {
449:                         this.notify("Notice!", 'Content ${address} provided to user
${event.returnValues.user}!', "material", 10000);
450:                     }
451:                 });
452:

```



```

574:         onChange={this.handleContent}
575:         error={error}
576:       />
577:     </Tooltip>
578:     <Button className={classes.warn} variant="extendedFab" onClick={this.initiateDelete}>
579:       <Delete className={classes.extendedIcon} />
580:       Delete
581:     </Button>
582:   </div>
583: );
584: default:
585:   return "";
586: }
587: })();
588: {transactPopup ? <TransactionConfirmation
589:   {...transactionProps}
590:   ok={(createdPopup || publishingContent)? this.confirmPublish :
591:     (monetize? this.confirmMonetize :
592:       (closing? this.confirmDelete : this.confirmCreate))}
593:   cancel={() => this.setState(oldState => ({...oldState, closing:
false, transactionProps: {}, transactPopup: false}))}
594: /> : ""}
595:   <Dialog
596:     open={createdPopup}
597:     keepMounted
598:     TransitionComponent={Transition}
599:     onEscapeKeyDown={this.handleDialog}
600:     onBackdropClick={this.handleDialog} >
601:     <DialogTitle>Content contract created!</DialogTitle>
602:     <DialogContent>
603:       <DialogContentText>The desired Content contract has been cre
ated at address:</DialogContentText>
604:       <Tooltip title="Click to copy" placement="right">
605:         <CopyToClipboard text={createdContent} onCopy={() => thi
s.notify("Success!", "Copied to clipboard", "success")}>
606:           <DialogContentText style={{cursor: "pointer"}}>{crea
tedContent}</DialogContentText>
607:         </CopyToClipboard>
608:       </Tooltip>
609:       <DialogContentText>Do you want to publish it right away?</Di
alogContentText>
610:       <DialogContentText>
611:         Make sure to hold on to the contract address if you want
to publish it later.
612:       </DialogContentText>
613:       <DialogContentText>
614:         You can click on the address in this window to copy it i
nto the clipboard.
615:       </DialogContentText>
616:     </DialogContent>
617:     <DialogActions>
618:       <Button
619:         color="primary"
620:         disabled={publishingContent}
621:         onClick={this.handleDialog}
622:         className={classes.button}>
623:         Cancel
624:       </Button>
625:       <Button
626:         color="secondary"
627:         disabled={publishingContent}
628:         onClick={this.publishContent}
629:         className={classes.button}>
630:         Publish
631:       </Button>

```

```

632:     </DialogActions>
633:   </Dialog>
634: </AppDrawer>
635: );
636: }
637: }
638:
639: CreatorApp.propTypes = {
640:   classes: PropTypes.object.isRequired,
641:   web3: PropTypes.object.isRequired,
642:   catalog: PropTypes.string.isRequired,
643:   account: PropTypes.string.isRequired,
644:   manageCenter: PropTypes.func.isRequired,
645: };
646:
647: export default withStyles(styles)(CreatorApp);

```

```

1:  /*
2:  *
3:  * University di Pisa - Master's Degree in Computer Science and Networking
4:  *
5:  * Final Project for the course of Peer to Peer Systems and Blockchains
6:  *
7:  * Teacher: Prof. Laura Ricci
8:  *
9:  * Candidate: Orlando Leombruni, matricola 475727
10:  *
11:  * File: BuyGiftPremiumDialog.js
12:  *
13:  */
14:
15: import React from 'react';
16: import PropTypes from 'prop-types';
17: import {
18:   withStyles,
19:   Dialog,
20:   DialogTitle,
21:   DialogContent,
22:   DialogContentText,
23:   DialogActions,
24:   TextField, Button,
25: } from '@material-ui/core';
26: import { prettifyWei } from '../Utils';
27: import { BuyGiftPremiumDialogStyle as styles } from "../styles/MaterialCustomStyles";
28:
29: /*
30:  * BuyGiftPremiumDialog Class
31:  *
32:  * A React Component that displays a dialog window, reviewing the purchase or renewal
33:  * of a Premium subscription
34:  * and asking the user if they want to buy it for themselves or gift it to another user.
35:  */
36: class BuyGiftPremiumDialog extends React.Component {
37:
38:   constructor(props) {
39:     super(props);
40:     this.state = {
41:       giftTo: '',
42:       retrieving: true,
43:       price: 0,
44:     };
45:   }
46:
47:   handleChange = (event) => {
48:     const { target } = event;
49:     this.setState(oldState => ({...oldState, giftTo: target.value}));
50:   };
51:
52:   /*
53:    * This is a React state function that will be called only once, after the component
54:    * is mounted in the virtual
55:    * DOM but before it gets rendered.
56:    * In particular, this function asks the Catalog contract for the price of a Premium
57:    * subscription.
58:    */
59:   componentDidMount() {
60:     this.props.catalog.methods.getPremiumPrice().call({from: this.props.account})
61:   }.then(
62:     (result) => {
63:       this.setState(oldState => ({giftTo: oldState.giftTo, retrieving: false,

```

```

64:       price: result}))
65:     },
66:     (error) => {
67:       console.log(error);
68:       this.props.cancel(true);
69:     }
70:   );
71: }
72:
73: render() {
74:   const { classes, gifting, confirm, cancel } = this.props;
75:   const { giftTo, retrieving, price } = this.state;
76:
77:   return (
78:     <div className={classes.root}>
79:       <Dialog
80:         open={true}
81:         keepMounted
82:         onBackdropClick={cancel(false)}
83:         onEscapeKeyDown={cancel(false)}>
84:         <DialogTitle>Purchase Premium subscription</DialogTitle>
85:         <DialogContent>
86:           <DialogContentText>
87:             {retrieving ?
88:               "Fetching subscription price..." :
89:               "The subscription costs ${prettifyWei(price)}."}
90:           </DialogContentText>
91:           <div style={{display: (gifting? "block" : "none")}}>
92:             <DialogContentText>Insert the account to gift.</DialogContentText>
93:             <form onChange={this.handleChange}>
94:               <TextField
95:                 name={"account"}
96:                 type={"string"}
97:                 placeholder={"Account address..."}
98:                 value={giftTo}
99:                 className={classes.textField}
100:               />
101:             </form>
102:           </div>
103:           <DialogContentText>Purchase?</DialogContentText>
104:           <DialogActions>
105:             <Button color={"inherit"} onClick={cancel(false)}>Cancel</Button>
106:             {gifting ? <Button
107:               disabled={retrieving || !giftTo}
108:               color={"secondary"}
109:               onClick={confirm(price, giftTo)}>
110:                 Gift
111:             </Button> :
112:             <Button
113:               disabled={retrieving}
114:               color={"secondary"}
115:               onClick={confirm(price)}>
116:                 Ok
117:             </Button>
118:           }
119:         </DialogActions>
120:       </Dialog>
121:     </div>
122:   );
123: }
124: BuyGiftPremiumDialog.propTypes = {

```

```
125:     classes: PropTypes.object.isRequired,  
126:     gifting: PropTypes.bool.isRequired,  
127:     confirm: PropTypes.func.isRequired,  
128:     cancel: PropTypes.func.isRequired,  
129:     catalog: PropTypes.object.isRequired,  
130:     account: PropTypes.string.isRequired,  
131:   };  
132:  
133: export default withStyles(styles)(BuyGiftPremiumDialog);
```

```
1: /*
2:  *
3:  * University di Pisa - Master's Degree in Computer Science and Networking
4:  *
5:  * Final Project for the course of Peer to Peer Systems and Blockchains
6:  *
7:  * Teacher: Prof. Laura Ricci
8:  *
9:  * Candidate: Orlando Leombruni, matricola 475727
10:  *
11:  * File: TransactionConfirmation.js
12:  *
13:  */
14:
15: import React from 'react';
16: import PropTypes from 'prop-types';
17: import BN from 'bn.js';
18: import {
19:   withStyles,
20:   Dialog,
21:   DialogTitle,
22:   DialogContent,
23:   DialogContentText,
24:   DialogActions,
25:   Button,
26: } from '@material-ui/core';
27: import { prettifyWei } from '../Utils';
28: import { TransactionConfirmationStyle as styles } from '../styles/MaterialCustomStyl
es";
29:
30: /*
31:  * TransactionConfirmation Class
32:  *
33:  * A React Component that shows to the user a dialog window that contains informatio
n about the transaction
34:  * they're going to perform: estimated gas cost, estimated transaction cost and curr
ent account balance.
35:  * The user can then choose to perform or decline the transaction.
36:  */
37: class TransactionConfirmation extends React.Component {
38:
39:   render() {
40:     const { classes, gas, gasPrice, balance, ok, cancel } = this.props;
41:     const valid = gas * gasPrice < balance;
42:     return (
43:       <Dialog
44:         open={true}
45:         keepMounted
46:         onBackdropClick={cancel}
47:         onEscapeKeyDown={cancel} >
48:         <DialogTitle>{valid ? "Perform the transaction?" : "Insufficient fun
ds"}</DialogTitle>
49:         <DialogContent>
50:           <ul>
51:             <li><DialogContentText>Estimated gas is {gas}</DialogCon
tentText></li>
52:             <li><DialogContentText>Estimated transation cost is {pre
ttifyWei(new BN(gasPrice).mul(new BN(gas)).toString())}</DialogContentText></li>
53:             {valid ?
54:               <li><DialogContentText>Account balance is {prettifyW
ei(balance)}</DialogContentText></li>:
55:               <li><DialogContentText>Insufficient funds in the acc
ount ({prettifyWei(balance)})</DialogContentText></li>}
56:           </ul>
57:         </DialogContent>
58:         <DialogActions>
```

```
59:           <Button color={"primary"} disabled={!valid} className={classes.b
uttonOk} onClick={ok}>Ok</Button>
60:           <Button color={"inherit"} className={classes.buttonNo} onClick={
cancel}>Cancel</Button>
61:         </DialogActions>
62:       </Dialog>
63:     );
64:   }
65:
66: }
67:
68: TransactionConfirmation.propTypes = {
69:   classes: PropTypes.object.isRequired,
70:   gas: PropTypes.string.isRequired,
71:   gasPrice: PropTypes.string.isRequired,
72:   balance: PropTypes.string.isRequired,
73:   ok: PropTypes.func.isRequired,
74:   cancel: PropTypes.func.isRequired,
75: };
76:
77: export default withStyles(styles)(TransactionConfirmation);
```





```

1:  /*
2:  *
3:  * University di Pisa - Master's Degree in Computer Science and Networking
4:  *
5:  * Final Project for the course of Peer to Peer Systems and Blockchains
6:  *
7:  * Teacher: Prof. Laura Ricci
8:  *
9:  * Candidate: Orlando Leombruni, matricola 475727
10:  *
11:  * File: UserApp.js
12:  *
13:  */
14: import React from 'react';
15: import PropTypes from 'prop-types';
16: import {
17:   withStyles,
18:   List,
19:   ListItem,
20:   ListItemText,
21:   ListItemIcon,
22:   Tooltip,
23:   Divider,
24:   Typography,
25:   InputBase,
26:   Button,
27: } from '@material-ui/core';
28: import {
29:   List as ListIcon,
30:   Search,
31:   Stars,
32:   Subscriptions,
33:   BarChart,
34:   ShopTwo,
35:   Update,
36:   CardGiftcard,
37:   Timelapse,
38:   QueuePlayNext
39: } from '@material-ui/icons';
40: import ReactNotification from 'react-notifications-component';
41: import 'react-notifications-component/dist/theme.css';
42: import AppDrawer from './AppDrawer';
43: import ContentList from './ContentList';
44: import json from '../solidity/compiled.json';
45: import {getTransactionParameters, makeTitle, prettifyWei} from '../Utils';
46: import BuyGiftDialog from './BuyGiftDialog';
47: import TransactionConfirmation from './TransactionConfirmation';
48: import BuyGiftPremiumDialog from './BuyGiftPremiumDialog';
49: import FeedbackForm from './FeedbackForm';
50: import Charts from './Charts';
51: import { UserAppStyle as styles } from '../styles/MaterialCustomStyles';
52:
53: /*
54:  * Enum-like object for the various sections of the application.
55:  */
56: const Panels = Object.freeze({
57:   FULL_LIST: Symbol("list"),
58:   CHARTS: Symbol("charts"),
59:   PLAY: Symbol("play"),
60:   FEEDBACK: Symbol("feedback"),
61:   BUY: Symbol("buy")
62: });
63:
64: /*
65:  * UserApp Class
66:  */

```

```

67:  * A React Component that represents the entire User (consumer) portion of the app.
68:  *
69:  * It allows Users to check trending content, purchase premium subscriptions and acc
ess to Contents, and leave
70:  * feedback.
71:  */
72: class UserApp extends React.Component {
73:
74:   constructor(props) {
75:     super(props);
76:     this.notify = this.notify.bind(this);
77:     this.notificationRef = React.createRef();
78:     this.props.manageCenter(false);
79:     this.web3 = this.props.web3;
80:     const abi = JSON.parse(json.contracts["Catalog.sol:Catalog"].abi);
81:     this.Catalog = new this.web3.eth.Contract(abi, this.props.catalog);
82:     this.premiumUpdates = null;
83:   }
84:
85:   state = {
86:     authorList: [],
87:     genreList: [],
88:     loading: false,
89:     panel: Panels.FULL_LIST,
90:     contentList: [],
91:     searchTerm: "",
92:     filteredList: [],
93:     boughtContent: [],
94:     feedbackToLeave: [],
95:     feedbackLeaving: null,
96:     feedback: null,
97:     feedbackOpen: false,
98:     buyingPrice: 0,
99:     buyingAction: false,
100:    consuming: null,
101:    gifting: "",
102:    premiumFetching: true,
103:    isPremium: false,
104:    buyingPremium: false,
105:    premiumMethod: null,
106:    giftingPremium: false,
107:    transactPopup: false,
108:    transactionProps: {},
109:  };
110:
111:  // State not used in rendering (doesn't have to be changed by React.Component.se
tState)
112:  otherState = {
113:    contentBuying: -1,
114:    isPremiumTransaction: false,
115:    authorSubs: [],
116:    genreSubs: [],
117:  };
118:
119:  // Displays a notification in the upper right portion of the screen.
120:  notify(title, message, type, time) {
121:    this.notificationRef.current.addNotification({
122:      title,
123:      message,
124:      type,
125:      insert: "top",
126:      container: "top-right",
127:      animationIn: ["animated", "fadeIn"],
128:      animationOut: ["animated", "fadeOut"],
129:      dismiss: { duration: time || 2000 },
130:      dismissable: { click: true },

```

```

131:     });
132:   }
133:
134:   // Changes the current panel (section of the app).
135:   changePanel = (panel) => () => {
136:     this.setState(oldState => ({...oldState, panel: panel}));
137:   };
138:
139:   // Renders the drawer menu.
140:   menu = () =>
141:     <List>
142:       <Tooltip title="List all contents in the Catalog" placement="right">
143:         <ListItem button key="list" onClick={this.changePanel(Panels.FULL_
LIST)}}>
144:           <ListItemIcon><ListIcon /></ListItemIcon>
145:           <ListItemText secondary="Show all Contents"/>
146:         </ListItem>
147:       </Tooltip>
148:       <Tooltip title="Discover trending content" placement="right">
149:         <ListItem button key="charts" onClick={this.changePanel(Panels.CHA
RTS)}}>
150:           <ListItemIcon><BarChart/></ListItemIcon>
151:           <ListItemText secondary="Trending"/>
152:         </ListItem>
153:       </Tooltip>
154:       <Divider/>
155:       <Tooltip title="Obtain a Content that you've bought" placement="right"
">
156:         <ListItem button key="play" onClick={this.changePanel(Panels.PLAY)}
>
157:           <ListItemIcon><Subscriptions/></ListItemIcon>
158:           <ListItemText secondary="Consume Content"/>
159:         </ListItem>
160:       </Tooltip>
161:       <Tooltip title="Leave feedback for consumed Contents" placement="right"
">
162:         <ListItem button key="feedback" onClick={this.changePanel(Panels.F
EEDBACK)}}>
163:           <ListItemIcon><Stars/></ListItemIcon>
164:           <ListItemText secondary="Leave Feedback"/>
165:         </ListItem>
166:       </Tooltip>
167:       <Tooltip title="Buy or gift a Premium subscription" placement="right"
">
168:         <ListItem button key="buy" onClick={this.changePanel(Panels.BUY)}>
169:           <ListItemIcon><ShopTwo/></ListItemIcon>
170:           <ListItemText secondary="Buy Premium"/>
171:         </ListItem>
172:       </Tooltip>
173:       {(this.state.panel === Panels.FULL_LIST) && <Divider/>}
174:       {(this.state.panel === Panels.FULL_LIST) &&
175:         <Tooltip title="Update Content list" placement="right">
176:           <ListItem button key="update" onClick={this.updateContentList}
>
177:             <ListItemIcon><Update/></ListItemIcon>
178:             <ListItemText secondary="Update List"/>
179:           </ListItem>
180:         </Tooltip>
181:       }
182:     </List>;
183:
184:   // Renders the app bar title (depending on the current panel).
185:   renderTitle = () => {
186:     const { classes } = this.props;
187:     const { panel, searchTerm } = this.state;
188:     switch(panel) {

```

```

189:       case Panels.FULL_LIST:
190:         return (
191:           [
192:             makeTitle("List of available Content", classes.grow),
193:             <div key={"search"} className={classes.search}>
194:               <div className={classes.searchIcon}>
195:                 <Search />
196:               </div>
197:               <InputBase
198:                 placeholder="Searchâ\200|"
199:                 classes={{
200:                   root: classes.inputRoot,
201:                   input: classes.inputInput,
202:                 }}
203:                 onChange={this.filterContent}
204:                 value={searchTerm}
205:               />
206:             </div>
207:           ]
208:         );
209:       case Panels.PLAY:
210:         return [makeTitle("Consume Content", classes.grow)];
211:       case Panels.BUY:
212:         return [makeTitle("Premium Subscriptions", classes.grow)];
213:       case Panels.FEEDBACK:
214:         return [makeTitle("Rate your Contents", classes.grow)];
215:       case Panels.CHARTS:
216:         return [makeTitle("Discover trending", classes.grow)];
217:       default:
218:         return [makeTitle("Something went wrong", classes.grow)];
219:     }
220:   };
221:
222:   // Filters the content list in real-time according to user input in the search b
ar.
223:   filterContent = (event) => {
224:     const { target } = event;
225:     const filtered = this.state.contentList.filter(value => (
226:       value.description.toLowerCase().includes(target.value.toLowerCase())
227:       || value.genre.toLowerCase().includes(target.value.toLowerCase())
228:       || value.author.toLowerCase().includes(target.value.toLowerCase())
229:     ));
230:     this.setState(oldState => ({...oldState, filteredList: filtered, searchTerm:
target.value}));
231:   };
232:
233:   // Refreshes the content list (and also the list of authors and genres).
234:   updateContentList = () => {
235:     this.setState(o => ({...o, loading: true}));
236:     this.Catalog.methods.getStatistics().call({from: this.props.account})
237:       .then(
238:         (result) => {
239:           if (!result[0] || !result[1] || !result[2] || !result[3] || !res
ult[4] ) {
240:             this.notify("Error", "Couldn't fetch Catalog content", "dang
er");
241:             console.log(result);
242:           } else {
243:             const list = [];
244:             for (let i = 0; i < result[0].length; i++) {
245:               list.push({
246:                 index: i,
247:                 description: this.web3.utils.hexToAscii(result[0][i]),
248:                 genre: this.web3.utils.hexToAscii(result[1][i]),
249:                 author: this.web3.utils.hexToAscii(result[2][i]),

```

```

250:         price: result[3][i],
251:         views: result[4][i],
252:     })
253: }
254: const authors = new Set(), genres = new Set();
255: for (let j = 0; j < list.length; j++) {
256:     authors.add(list[j].author);
257:     genres.add(list[j].genre);
258: }
259: this.setState(oldState => ({
260:     ...oldState,
261:     loading: false,
262:     contentList: list,
263:     filteredList: list,
264:     authorList: [...authors],
265:     genreList: [...genres],
266: }));
267: this.notify("List updated!", "List of content has been updated", "material");
268:     }
269: },
270: (error) => {
271:     this.notify("Error", "Couldn't fetch Catalog content", "danger");
272:     console.log(error);
273:     this.setState(o => ({...o, loading: false}));
274: }
275: )
276: };
277:
278: // Refreshes the list of Content bought by the User.
279: refreshBought = () => {
280:     this.Catalog.methods.grantsAvailable().call({from: this.props.account}).then
281: (
282:     (result) => {
283:         if (!result) {
284:             this.notify("Error", "Could not retrieve bought content", "danger");
285:         } else {
286:             const list = [];
287:             for (let i = 0; i < result.length; i++) {
288:                 list.push({
289:                     index: i,
290:                     description: this.web3.utils.hexToAscii(result[i]),
291:                 })
292:             }
293:             this.setState(oldState => ({...oldState, loading: false, boughtContent: list}));
294:             this.notify("Success!", "Bought Content list updated!", "success");
295:         }
296:     },
297:     (error) => {
298:         console.log(error);
299:         this.notify("Error", "Could not retrieve bought content", "danger");
300:         this.setState(o => ({...o, loading: false}));
301:     }
302: );
303: this.setState(o => ({...o, loading: true}));
304: };
305: // Refreshes the list of Content for which the User can leave feedback.
306: refreshFeedback = () => {
307:     this.Catalog.methods.feedbackAvailable().call({from: this.props.account}).then
308: (
309:     (result) => {

```

```

310:         if (!result[0] || !result[1]) {
311:             this.notify("Error", "Could not retrieve list of content to leave feedback for", "danger");
312:         } else {
313:             const list = [];
314:             for (let i = 0; i < result[0].length; i++) {
315:                 list.push({
316:                     index: i,
317:                     description: this.web3.utils.hexToAscii(result[0][i]),
318:                     address: result[1][i],
319:                 })
320:             }
321:             this.setState(oldState => ({...oldState, loading: false, feedbackToLeave: list}));
322:             this.notify("Success!", "List of feedbacks to leave updated!", "success");
323:         }
324:     },
325:     (error) => {
326:         console.log(error);
327:         this.notify("Error", "Could not retrieve list of content to leave feedback for", "danger");
328:     }
329: );
330: this.setState(o => ({...o, loading: true}));
331: };
332:
333: // Starts the "buy" operation (for both premium and non-premium users), estimating its gas cost.
334: buy = acc => () => {
335:     const { contentList } = this.state;
336:     const { contentBuying, isPremiumTransaction: premium } = this.otherState;
337:     if (contentBuying < 0) {
338:         this.notify("Error", "No selected content", "danger");
339:     } else {
340:         const content = this.web3.utils.asciiToHex(contentList[contentBuying].description);
341:         const method = (premium)? this.Catalog.methods.getContentPremium(content) : this.Catalog.methods.getContent(content);
342:         ((acc) ? this.Catalog.methods.giftContent(content, acc) : this.Catalog.methods.getContent(content));
343:         const value = premium ? 0 : contentList[contentBuying].price;
344:         getTransactionParameters(this.web3, method, this.props.account, {value})
345:     }.then(
346:         (result) => {
347:             this.setState(o => ({...o, transactionProps: result, transactionUp: true}));
348:         },
349:         (error) => {
350:             console.log("Error in getting parameters for buy");
351:             console.log(error.stackTrace);
352:             switch(error.type) {
353:                 case "estimateGas":
354:                     this.notify("Error",
355:                         (premium) ? "Unable to obtain access to Content, check if you have a subscription" :
356:                         "Unable to buy Content, check if already bought or if recipient has a premium subscription",
357:                     );
358:                     break;
359:                 case "getGasPrice":
360:                     this.notify("Error", "Could not get gas price from the blockchain, check the connection", "danger");
361:                     break;

```

```

362:         case "getBalance":
363:             this.notify("Error", "Could not get balance for the curr
ent account, check the connection", "danger");
364:             break;
365:             default:
366:                 this.notify("Error", error.stackTrace.message.toString()
, "danger");
367:             }
368:             this.otherState.contentBuying = -1;
369:             this.otherState.isPremiumTransaction = false;
370:             this.setState(oldState => ({
371:                 ...oldState,
372:                 transactionProps: {},
373:                 transactPopup: false,
374:                 buyingAction: false,
375:                 gifting: "",
376:             }));
377:         }
378:     });
379:     this.setState(oldState => ({...oldState, gifting: acc, buyingPrice: 0, b
uyingAction: true}));
380:     }
381:     };
382:
383:     /*
384:     * Invoked when the user confirms the transaction (after reviewing it in an info
rmatve dialog), this function
385:     * effectively asks the Catalog for access to the content, paying for it if the
user is not a Premium subscriber.
386:     */
387:     confirmBuy = () => {
388:         const { contentList, gifting, transactionProps } = this.state;
389:         const { contentBuying, isPremiumTransaction } = this.otherState;
390:         if (contentBuying < 0) {
391:             this.notify("Error", "No selected content", "danger");
392:         } else {
393:             const content = this.web3.utils.asciiToHex(contentList[contentBuying].de
scription);
394:             const method = (isPremiumTransaction)? this.Catalog.methods.getContentPr
emium(content) :
395:                 ((gifting) ? this.Catalog.methods.giftContent(content, gifting) : th
is.Catalog.methods.getContent(content));
396:
397:             method.send({
398:                 from: this.props.account,
399:                 gas: transactionProps.gas,
400:                 gasPrice: transactionProps.gasPrice,
401:                 value: (isPremiumTransaction ? 0 : contentList[contentBuying].price)
,
402:             }).then(
403:                 (result) => this.setState(o => ({...o, loading: false})),
404:                 (error) => {
405:                     console.log(error);
406:                     this.notify("Error", "Could not complete purchase", "danger");
407:                     this.setState(o => ({...o, loading: false}));
408:                 }
409:             );
410:
411:         }
412:         this.otherState = {contentBuying: -1, isPremiumTransaction: false};
413:         this.setState(oldState => ({...oldState, loading: true, gifting: "", buyingA
ction: false, transactionProps: {}, transactPopup: false}))
414:     };
415:
416:     // Starts the "leave feedback for a content" operation, estimating its gas cost.
417:     leaveFeedback = (idx) => () => {
418:         const content = this.state.feedbackToLeave[idx];
419:         if (content.address) {
420:             this.setState(o => ({...o, feedbackLeaving: idx, feedbackOpen: true}));
421:         }
422:     };
423:
424:     feedbackFormSubmitted = (feedback) => () => {
425:         const content = this.state.feedbackToLeave[this.state.feedbackLeaving];
426:         if (content.address) {
427:             getTransactionParameters(this.web3, this.Catalog.methods.leaveFeedback(c
ontent.address, 5, 5, 5), this.props.account).then(
428:                 (result) => this.setState(o => ({
429:                     ...o,
430:                     feedback: feedback,
431:                     transactionProps: result,
432:                     transactPopup: true
433:                 })),
434:                 (error) => {
435:                     console.log("Error in getting parameters for leaveFeedback");
436:                     console.log(error);
437:                     this.notify("Error", 'Could not complete operation (error in ${e
rror.type})', "danger");
438:                     this.setState(o => ({...o, feedbackLeaving: null, feedbackOpen:
false}));
439:                 }
440:             )
441:         }
442:     };
443:
444:     /*
445:     * Invoked when the user confirms the transaction (after reviewing it in an infor
mative dialog), this function
446:     * effectively writes the user's feedback for the current Content to the Catalog.
447:     */
448:     confirmedFeedback = () => {
449:         const { feedbackLeaving, feedbackToLeave, transactionProps, feedback } = thi
s.state;
450:         const { appreciation, fairness, suggest } = feedback;
451:         this.Catalog.methods
452:             .leaveFeedback(feedbackToLeave[feedbackLeaving].address, appreciation, f
airness, suggest)
453:             .send({from: this.props.account, gas: transactionProps.gas, gasPrice: tr
ansactionProps.gasPrice})
454:             .then(
455:                 (result) => {
456:                     this.notify("Success!", 'Left feedback for content ${feedbackToL
eave[feedbackLeaving].description}', "success");
457:                     const feedList = feedbackToLeave.filter((item, idx) => idx !== f
eedbackLeaving).map((item, idx) =>
458:                         ({...item, index: idx})
459:                     );
460:                     this.setState(o => ({...o, loading: false, feedbackLeaving: null
, feedback: null, feedbackToLeave: feedList}));
461:                 },
462:                 (error) => {
463:                     console.log("error in confirm feedback");
464:                     console.log(error);
465:                     this.notify("Error", 'Could not leave feedback for ${feedbackToL
eave[feedbackLeaving].description}', "danger");
466:                     this.setState(o => ({...o, loading: false, feedbackLeaving: null
, feedback: null}));
467:                 }
468:             );
469:             this.setState(o => ({...o, loading: true, feedbackOpen: false, transactionPr
ops: {}, transactPopup: false}));
470:         };

```

```

471:
472: // Starts the consumption of a bought Content, estimating its gas cost.
473: consume = (idx) => () => {
474:   const content = this.state.boughtContent[idx];
475:   if (content.address) {
476:     const abi = JSON.parse(json.contracts["ContentManagementBase.sol:Content
ManagementBase"].abi);
477:     const CMB = new this.web3.eth.Contract(abi, content.address);
478:     getTransactionParameters(this.web3, CMB.methods.consumeContent(), this.p
rops.account).then(
479:       (result) => this.setState(o => ({...o, transactionProps: result, tra
nsactPopup: true})),
480:       (error) => {
481:         console.log("Error in getting parameters for consume");
482:         console.log(error);
483:         this.notify("Error", 'Could not complete operation (error in ${e
rror.type})', "danger");
484:         this.setState(o => ({...o, consuming: null}))
485:       }
486:     );
487:     this.setState(oldState => ({...oldState, consuming: CMB}));
488:   } else {
489:     this.Catalog.methods.getContentAddress(this.web3.utils.asciiToHex(conten
t.description)).call(
490:       {from: this.props.account}
491:     ).then(
492:       (result) => {
493:         this.setState(oldState => ({
494:           ...oldState,
495:           boughtContent: oldState.boughtContent.map((v, i) =>
496:             (i === idx) ? {...v, address: result.toString()} : v
497:           )
498:         }));
499:         this.consume(idx)();
500:       },
501:       (error) => {
502:         console.log(error);
503:         this.notify("Error", "Could not get Content's address", "danger"
);
504:       }
505:     )
506:   }
507: };
508:
509: /*
510:  * Invoked when the user confirms the transaction (after reviewing it in an infor
mative dialog), this function
511:  * effectively asks the Content for consumption, also registering callbacks for t
he "Content Provider" and "Can
512:  * Leave Feedback" solidity events.
513:  */
514: confirmConsume = () => {
515:   this.state.consuming.once('ProvideContent', {filter: {user: this.props.accou
nt}, fromBlock: "latest"},
516:     (error, event) => {
517:       if (error) console.log("provideContent error: " + error);
518:       else {
519:         const contentName = this.web3.utils.hexToAscii(event.returnValue
s.content);
520:         this.notify("Success!", 'Content "${contentName}" has been provi
ded!', "success", 10000);
521:         const bought = this.state.boughtContent
522:           .filter(content => content.description !== contentName)
523:           .map((item, idx) => ({...item, index: idx}));
524:         this.setState(oldState => ({...oldState, boughtContent: bought}
);

```

```

525:       }
526:     }
527:   );
528:   this.state.consuming.once('CanLeaveFeedback', {filter: {user: this.props.acc
ount}, fromBlock: "latest"},
529:     (error, event) => {
530:       if (error) console.log("canLeaveFeedback error: " + error);
531:       else {
532:         const contentName = this.web3.utils.hexToAscii(event.returnValue
s.content);
533:         this.notify("Leave feedback", 'You can now leave feedback for ${
contentName}', "material", 10000);
534:         const feedbacks = [...this.state.feedbackToLeave, {
535:           index: this.state.feedbackToLeave.length,
536:           description: contentName,
537:           address: event.address,
538:         }];
539:         this.setState(oldState => ({...oldState, feedbackToLeave: feedba
cks}));
540:       }
541:     }
542:   );
543:   this.state.consuming.methods.consumeContent().send({
544:     from: this.props.account,
545:     gas: this.state.transactionProps.gas,
546:     gasPrice: this.state.transactionProps.gasPrice,
547:   }).then(
548:     (result) => {
549:       this.setState(o => ({...o, loading: false}));
550:       this.notify("Success!", "Request for access sent", "success");
551:     },
552:     (error) => {
553:       console.log(error);
554:       this.setState(o => ({...o, loading: false}));
555:       this.notify("Error", "Request for access denied or other error, chec
k console log", "danger");
556:     }
557:   );
558:   this.setState(oldState => ({...oldState, loading: true, consuming: null, tra
nsactionProps: {}, transactPopup: false}));
559: };
560:
561: // Starts the purchase or renewal of a Premium subscription, estimating the gas
cost of the operation.
562: buyPremium = (price, acc) => () => {
563:   const method = (acc) ? this.Catalog.methods.giftPremium(acc) : this.Catalog.
methods.buyPremium();
564:   getTransactionParameters(this.web3, method, this.props.account, {value: pric
e}).then(
565:     (result) => this.setState(o => ({...o, transactionProps: {price, ...resu
lt}, transactPopup: true})),
566:     (error) => {
567:       console.log("Error in getting parameters for buyPremium");
568:       console.log(error);
569:       this.notify("Error", 'Could not complete operation (error in ${error
.type})', "danger");
570:       this.setState(o => ({...o, giftingPremium: false, premiumMethod: nul
l}));
571:     }
572:   );
573:   this.setState(o => ({...o, buyingPremium: false, premiumMethod: method}));
574: };
575:
576: /*
577:  * Invoked when the user confirms the transaction (after reviewing it in an infor
mative dialog), this function

```

```

578:    * effectively purchases or renews a Premium subscription (either for the user or
gifted to another user) from the
579:    * Catalog.
580:    */
581:    confirmPremium = () => {
582:      this.state.premiumMethod.send({
583:        from: this.props.account,
584:        gas: this.state.transactionProps.gas,
585:        gasPrice: this.state.transactionProps.gasPrice,
586:        value: this.state.transactionProps.price
587:      }).then(
588:        (result) => {
589:          let premium;
590:          if (this.state.giftingPremium) {
591:            this.notify("Success!", "Gifted a premium subscription to the ot
her user", "success");
592:            premium = this.state.isPremium;
593:          } else {
594:            this.notify("Success!",
595:              'You ${this.state.isPremium ? "extended your" : "bought a"}
Premium subscription!', "success");
596:            premium = true;
597:          }
598:          this.setState(o => ({...o, loading: false, isPremium: premium, premi
umMethod: null, giftingPremium: false}));
599:        },
600:        (error) => {
601:          console.log(error);
602:          this.notify("Error", "Could not buy/gift content", "danger");
603:          this.setState(o => ({...o, loading: false, premiumMethod: null, gift
ingPremium: false}));
604:        }
605:      );
606:      this.setState(o => ({...o, transactionProps: {}, transactPopup: false, loadi
ng: true}));
607:    };
608:
609:    // Adds an author to the list of subscriptions for notifications
610:    addSubAuthor = (author) => {
611:      if (this.otherState.authorSubs.includes(author)) {
612:        this.notify("Warning", 'You was already subscribed to new content by ${a
uthor}', "material");
613:      } else {
614:        this.otherState.authorSubs.push(author);
615:        this.notify("Success!", 'You've subscribed to new content by ${author}!'
, "success");
616:      }
617:    };
618:
619:    // Adds a genre to the list of subscriptions for notifications
620:    addSubGenre = (genre) => {
621:      if (this.otherState.genreSubs.includes(genre)) {
622:        this.notify("Warning", 'You was already subscribed to new content with g
enre ${genre}', "material");
623:      } else {
624:        this.otherState.genreSubs.push(genre);
625:        this.notify("Success!", 'You've subscribed to new content with genre ${g
enre}!', "success");
626:      }
627:    };
628:
629:    /*
630:     * This is a React state function that will be called only once, after the compo
nent is mounted in the virtual
631:     * DOM but before it gets rendered.
632:     *

```

```

633:     * First, this function populates the content list. Then it registers some callb
acks for Solidity events
634:     * fired when there's a new content available, when the Catalog grants access to
a Content, and when a(n extension
635:     * to the) Premium subscription is made available for the user. Finally, it regi
sters a repeating function that
636:     * checks if the user is still a Premium subscriber every 20 seconds.
637:     */
638:    componentDidMount() {
639:      this.updateContentList();
640:      this.Catalog.events.NewContentAvailable({fromBlock: "latest"}).on('data', (e
vent) => {
641:        let show = false;
642:        if (this.otherState.genreSubs.includes(this.web3.utils.hexToAscii(event.
returnValues.genre))
643:          || this.otherState.authorSubs.includes(this.web3.utils.hexToAscii(ev
ent.returnValues.author)))
644:          show = true;
645:        if (show) {
646:          this.notify("New content!",
647:            'Content ${this.web3.utils.hexToAscii(event.returnValues.descr)}
has been published on the Catalog!',
648:            "material", 10000);
649:        }
650:        if (!this.state.contentList.some((content) =>
651:          content.description === this.web3.utils.hexToAscii(event.returnValue
s.descr)
652:        )) {
653:          const list = [...this.state.contentList, {
654:            index: this.state.contentList.length,
655:            description: this.web3.utils.hexToAscii(event.returnValues.descr
),
656:            genre: this.web3.utils.hexToAscii(event.returnValues.genre),
657:            author: this.web3.utils.hexToAscii(event.returnValues.author),
658:            price: event.returnValues.price,
659:            views: 0,
660:          }];
661:          this.setState(oldState => ({...oldState, contentList: list, filtered
List: list}));
662:        }
663:      }).on('error', console.log);
664:      this.Catalog.events.GrantedAccess({filter: {user: this.props.account}, fromB
lock: "latest"})
665:        .on('error', error => console.log("Error in GrantedAccess: " + error))
666:        .on('data', event => {
667:          const { boughtContent, contentList } = this.state;
668:          const description = this.web3.utils.hexToAscii(event.returnValues.de
scription);
669:          this.notify("Access!", 'Obtained access to "${description}"!', "succ
ess", 10000);
670:          const refresh = [...boughtContent, {
671:            description: description,
672:            index: boughtContent.length,
673:            address: event.returnValues.requested
674:          }];
675:          const newList = contentList.map((item) =>
676:            (item.description === description) ?
677:              {...item, address: event.returnValues.requested} :
678:              item
679:          );
680:          this.setState(oldState => ({...oldState, contentList: newList, bough
tContent: refresh}));
681:        });
682:      this.Catalog.events.GotPremium({filter: {user: this.props.account}, fromBloc
k: "latest"})
683:        .on('error', error => console.log("GotPremium: " + error))

```

```

684:         .on('data', event => {
685:             console.log(event);
686:             this.notify("Premium!", 'Obtained or extended a Premium subscription
!', "success", 10000);
687:             this.setState(oldState => ({...oldState, premiumFetching: false, isP
remium: true}));
688:         });
689:         const premiumfun = () => {
690:             this.setState(oldState => ({...oldState, premiumFetching: true}));
691:             this.Catalog.methods.isPremium(this.props.account).call({from: this.prop
s.account}).then(
692:                 (result) => {
693:                     this.setState(oldState => ({...oldState, premiumFetching: false,
isPremium: result}));
694:                 },
695:                 (error) => {
696:                     this.notify("Error", "Could not retrieve premium subscription st
atus", "danger");
697:                     this.setState(oldState => ({...oldState, premiumFetching: false}
))
698:                 }
699:             );
700:         };
701:         this.premiumUpdates = window.setInterval(premiumfun, 20000);
702:         premiumfun();
703:     }
704:
705:     // Unregisters the "check for premium status" function.
706:     componentWillUnmount() {
707:         if (this.premiumUpdates) window.clearInterval(this.premiumUpdates);
708:     }
709:
710:     render() {
711:         const {
712:             classes,
713:         } = this.props;
714:
715:         const {
716:             authorList,
717:             genreList,
718:             panel,
719:             filteredList,
720:             contentList,
721:             boughtContent,
722:             feedbackToLeave,
723:             buyingPrice,
724:             buyingAction,
725:             consuming,
726:             premiumFetching,
727:             isPremium,
728:             transactionProps,
729:             transactPopup,
730:             buyingPremium,
731:             premiumMethod,
732:             giftingPremium,
733:             feedbackOpen,
734:             feedback,
735:         } = this.state;
736:
737:         const defaultFun = x => x;
738:
739:         const attributes = [
740:             {
741:                 name: "genre",
742:                 description: "Genre",
743:                 copy: false,

```

```

744:                 makeSmall: defaultFun
745:             },
746:             {
747:                 name: "author",
748:                 description: "Author",
749:                 copy: false,
750:                 makeSmall: defaultFun,
751:             },
752:             {
753:                 name: "price",
754:                 description: "Price",
755:                 copy: false,
756:                 makeSmall: defaultFun,
757:                 extra: prettifyWei,
758:             },
759:             {
760:                 name: "views",
761:                 description: "Number of views",
762:                 copy: false,
763:                 makeSmall: defaultFun,
764:             },
765:         ],
766:     ];
767:
768:     return (
769:         <div>
770:             <AppDrawer drawer={true} writeTitle={this.renderTitle} render={this.
menu} loading={this.state.loading}>
771:                 <ReactNotification
772:                     ref={this.notificationRef}
773:                     types={[{
774:                         htmlClasses: classes.notification,
775:                         name: "material",
776:                     }]}
777:                 />
778:                 {(() => {
779:                     switch (panel) {
780:                         case Panels.FULL_LIST:
781:                             return(
782:                                 <div>
783:                                     <ContentList
784:                                         attributes={attributes}
785:                                         action={
786:                                             (i) => () => {
787:                                                 this.otherState.contentBuying = i;
788:                                                 this.setState(oldState => ({...oldSt
ate, buyingPrice: contentList[i].price}));
789:                                             }
790:                                         }
791:                                         actionName={"Buy"}
792:                                         action2={
793:                                             (i) => () => {
794:                                                 this.otherState = { contentBuying: i
795:                                                 this.buy() () }
796:                                             }
797:                                         }
798:                                         list={filteredList}
799:                                         notify={this.notify}
800:                                     />
801:                                 </div>
802:                             );
803:                         case Panels.PLAY:
804:                             return(
805:                                 <div>
806:                                     <div className={classes.subheading}>
807:                                         <Typography variant={"body1"} className=
The list may not be synced with the

```

```

server.
806:         </Typography>
807:         <Button variant={"contained"}
808:             size={"small"}
809:             onClick={this.refreshBought}
810:             color={"secondary"}
811:             className={classes.subMargin}>
812:             Refresh now
813:         </Button>
814:     </div>
815:     <ContentList
816:         className={classes.normalMargin}
817:         attributes={[]}
818:         action={this.consume}
819:         actionName={"Consume"}
820:         list={boughtContent}
821:         notify={this.notify}
822:     />
823: </div>
824: );
825: case Panels.FEEDBACK:
826:     return (
827:         <div>
828:             <div className={classes.subheading}>
829:                 <Typography variant={"body1"} className=
{classes.subMargin}>
830:                     The list may not be synced with the
server.
831:                 </Typography>
832:                 <Button variant={"contained"}
833:                     size={"small"}
834:                     onClick={this.refreshFeedback}
835:                     color={"secondary"}
836:                     className={classes.subMargin}>
837:                     Refresh now
838:                 </Button>
839:             </div>
840:             <ContentList
841:                 className={classes.normalMargin}
842:                 attributes={[]}
843:                 action={this.leaveFeedback}
844:                 actionName={"Leave Feedback"}
845:                 list={feedbackToLeave}
846:                 notify={this.notify}
847:             />
848:         </div>
849:     );
850: case Panels.BUY:
851:     return (
852:         <div>
853:             <div style={{display: "flex"}}>
854:                 <Typography variant={"subtitle2"}>Premiu
m subscription:</Typography>
855:                 <Typography
856:                     style={{marginLeft: 16}}
857:                     variant={"subtitle2"}
858:                     color={"inherit"}
859:                     className={premiumFetching ?
860:                         classes.prFetch :
861:                         (isPremium ? classes.premium : c
lasses.notPremium)}>
862:                     {premiumFetching ?
863:                         "Fetching..." :
864:                         (isPremium ? "Active" : "Not act
ive")}
865:                 </Typography>
866:             </div>
867:             <div style={{display: "flex", justifyContent
: "center"}}>
868:                 <Button
869:                     color={"primary"}
870:                     variant={"extendedFab"}
871:                     className={classes.button}
872:                     onClick={() => this.setState(o => ({
...o, buyingPremium: true, giftingPremium: false}))} >
873:                     {isPremium ?
874:                         <Timelapse className={classes.ex
tendedIcon}/> :
875:                         s.extendedIcon}/>}
876:                 <Button
877:                     color={"primary"}
878:                     variant={"extendedFab"}
879:                     className={classes.button}
880:                     onClick={() => this.setState(o => ({
...o, buyingPremium: true, giftingPremium: true}))} >
881:                     <CardGiftcard className={classes.ext
endedIcon}/>
882:                     Gift
883:                 </Button>
884:             </div>
885:         </div>
886:     );
887: case Panels.CHARTS:
888:     return (
889:         <Charts
890:             authors={authorList}
891:             genres={genreList}
892:             web3={this.web3}
893:             addSubAuthor={this.addSubAuthor}
894:             addSubGenre={this.addSubGenre}
895:             Catalog={this.Catalog}
896:             account={this.props.account}/>
897:         </div>
898:     );
899: default:
900:     return (
901:         <div>
902:             Not implemented.
903:         </div>
904:     );
905: }
906: })()
907: {feedbackOpen ?
908:     <FeedbackForm
909:         cancel={() => this.setState(o => ({...o, transactionPr
ops: {}, feedback: null, feedbackOpen: false, feedbackLeaving: null})) }
910:         confirm={this.feedbackFormSubmitted}
911:     /> : null
912: }
913: {buyingPrice ?
914:     <BuyGiftDialog
915:         price={buyingPrice}
916:         cancel={() => this.setState(oldState => ({...oldState, buyin
gPrice: 0, buyingAction: false}))}
917:         callback={this.buy}
918:     /> : null}
919: {buyingPremium &&
920:     <BuyGiftPremiumDialog
921:         account={this.props.account}
922:         catalog={this.Catalog}

```



```
924:         confirm={this.buyPremium}
925:         gifting={giftingPremium}
926:         cancel={error => () => {
927:             this.setState(o => ({...o, buyingPremium: false, gifting
Premium: false}));
928:             error && this.notify("Error", "Could not fetch Premium s
ubscription price from Catalog", "danger");
929:         }}
930:     />
931:     {transactPopup &&
932:     <TransactionConfirmation
933:         {...transactionProps}
934:         ok={buyingAction ? this.confirmBuy :
935:             (premiumMethod? this.confirmPremium :
936:                 (consuming? this.confirmConsume :
937:                     (feedback ? this.confirmedFeedback : null)))}
938:         cancel={() => this.setState(oldState => ({...oldState, buyin
gAction: false, consuming: false, transactionProps: {}, transactPopup: false}))}
939:     />
940:     </AppDrawer>
941: </div>
942:     );
943: }
944: }
945:
946: UserApp.propTypes = {
947:     classes: PropTypes.object.isRequired,
948:     web3: PropTypes.object.isRequired,
949:     account: PropTypes.string.isRequired,
950:     catalog: PropTypes.string.isRequired,
951:     manageCenter: PropTypes.func.isRequired,
952: };
953:
954: export default withStyles(styles)(UserApp);
```



```

1:  /*
2:  *
3:  * University di Pisa - Master's Degree in Computer Science and Networking
4:  *
5:  * Final Project for the course of Peer to Peer Systems and Blockchains
6:  *
7:  * Teacher: Prof. Laura Ricci
8:  *
9:  * Candidate: Orlando Leombruni, matricola 475727
10: *
11: * File: App.js
12: *
13: */
14:
15: import React from 'react';
16: import { withStyles } from '@material-ui/core/styles';
17: import { Dialog, Typography } from '@material-ui/core';
18: import classNames from 'classnames';
19: import LoginPage from './LoginPage';
20: import UserApp from './UserApp';
21: import CreatorApp from './CreatorApp';
22: import Web3 from 'web3';
23: import PropTypes from "prop-types";
24: import CatalogManager from './CatalogManager';
25: import { AppStyle as styles } from '../styles/MaterialCustomStyles';
26:
27: /*
28: *
29: * App Class
30: *
31: * A React Component that acts as the "foundation" for the web application. It manages the connection
32: * to a Web3 instance and provides the other Components with needed data such as the EOA used for the operations.
33: *
34: */
35:
36: class App extends React.Component {
37:
38:   constructor(props) {
39:     super(props);
40:     this.state = {
41:       centerContent: true,
42:       connecting: true,
43:       account: "",
44:       catalog: "",
45:       app: "none",
46:       noLogin: false,
47:       error: false,
48:     };
49:
50:     // Initialize the Web3 instance
51:     var web3;
52:     if (window.ethereum) { // Web3 provided by latest versions of browsers/extensions (e.g. Metamask, Mist)
53:       web3 = new Web3(window.ethereum);
54:       window.ethereum.enable().then(
55:         () => this.setState(oldState => ({...oldState, noLogin: true, connecting: false})),
56:         () => this.setState(oldState => ({error: true}))
57:       );
58:     } else { // Web3 injected by old browsers/extensions, or with an instance of a compatible client
59:       web3 = new Web3(Web3.givenProvider || new Web3.providers.WebsocketProvider("ws://localhost:8545"));
60:       web3.eth.net.isListening()

```

```

61:         .then(() => this.setState(oldState => ({...oldState, connecting: false}))),
62:         () => this.setState(oldState => ({error: true})));
63:     }
64:     /*
65:     * Workaround for latest versions of Web3 ( >= 1.0.0-beta.33 )
66:     * This fixes a bug where Solidity events or functions that return 0 or empty arrays cause Web3 to throw error
67:     * See https://github.com/ethereum/web3.js/issues/1916
68:     * Should be fixed in Web3.js version 1.0.0-beta.37
69:     */
70:     console.log("using web3 ver. ", web3.version);
71:     const version = +web3.version.slice(-2);
72:     if (version >= 33 && version < 37) {
73:       console.log("Applying fix patch");
74:       web3.eth.abi.decodeParameters = function(outputs, bytes) {
75:         if (bytes === '0x') {
76:           bytes = '0x00';
77:         }
78:         return web3.eth.abi.__proto__.decodeParameters(outputs, bytes)
79:       }
80:     }
81:     this.web3 = web3;
82:     this.submit = this.submit.bind(this);
83:   };
84:
85:   /*
86:   * Takes values input by the user (which EOA to use, which Catalog to connect to, which application to use)
87:   * and triggers the start of the chosen application (user interface/creator interface).
88:   * Used as a callback by the LoginPage component.
89:   */
90:   submit(account, catalog, app) {
91:     if (app === "error") this.setState(o => ({...o, error: true}));
92:     this.setState(oldState => ({...oldState, account, catalog, app}));
93:   };
94:
95:   /*
96:   * Centers the graphic elements in the window, if required.
97:   * Used as a callback by various components.
98:   */
99:   fixCenter = (what) =>
100:     this.setState(oldState => ({...oldState, centerContent: what}));
101:
102:   render() {
103:     const { connecting, account, catalog, app, error, centerContent, noLogin } = this.state;
104:     const { isCatalog, classes } = this.props;
105:     return (
106:       <div className={classNames("pagewide", classes.centered)} : "pagewide">>
107:         {error ?
108:           <Dialog open={true} disableBackdropClick={true} disableEscapeKeyDown={true}>
109:             <Typography id="error" variant={"body1"} color={"textPrimary"}>
110:               Could not connect to an Ethereum node. Check your provider (geth, Metamask, ...).
111:             </Typography>
112:           </Dialog> :
113:           (connecting ?
114:             <Typography variant={"body1"} color={"textPrimary"}>
115:               Connecting to Ethereum node...
116:             </Typography> :
117:             (isCatalog ?

```

```
118:         <CatalogManager web3={this.web3} noLogin={noLogin} manag
eCenter={this.fixCenter} /> :
119:         (app === "none") ?
120:         <LoginPage onSubmit={this.submit} noLogin={noLogin}
web3={this.web3} manageCenter={this.fixCenter} /> :
121:         (app === "user") ?
122:         <UserApp web3={this.web3} account={account} cata
log={catalog} manageCenter={this.fixCenter} /> :
123:         <CreatorApp web3={this.web3} account={account} c
atalog={catalog} manageCenter={this.fixCenter} />
124:         }
125:     </div>
126:   );
127: };
128: }
129:
130: App.propTypes = {
131:   classes: PropTypes.object.isRequired,
132:   isCatalog: PropTypes.bool.isRequired,
133: };
134:
135: export default withStyles(styles)(App);
```

```

1:  /*
2:  *
3:  *   University di Pisa - Master's Degree in Computer Science and Networking
4:  *
5:  *   Final Project for the course of Peer to Peer Systems and Blockchains
6:  *
7:  *   Teacher: Prof. Laura Ricci
8:  *
9:  *   Candidate: Orlando Leombruni, matricola 475727
10:  *
11:  *   File: Utils.js
12:  *
13:  *   Collection of utility functions, used throughout the webapp.
14:  */
15:
16: import BN from 'bn.js';
17: import Typography from "@material-ui/core/Typography/Typography";
18: import React from "react";
19:
20: // Given a numerical amount of wei (e.g. 5003000000000000000) returns a human-readable string (e.g. "5.003 ether").
21: export const prettifyWei = wei => {
22:   const weiBN = new BN(wei);
23:   const mweiBN = weiBN.divn(1000000);
24:   const finneyBN = mweiBN.div(new BN(1000000000));
25:   if (finneyBN.gtn(0)) {
26:     return (finneyBN.toNumber() / 1000) + " ether";
27:   } else
28:   if (mweiBN.gtn(0)) {
29:     return (mweiBN.toNumber() / 1000) + " gwei";
30:   } else return weiBN.toString() + " wei";
31: };
32:
33: // Bundles together text and style for app bar titles.
34: export const makeTitle = (title, className) =>
35:   <Typography key={"title"} variant="h6" color="inherit" className={className} noWrap>
36:     {title}
37:   </Typography>;
38:
39: /*
40: * Given a Solidity method (accessed through a Web3 instance), its parameters and the EOA from which the operation
41: * must be performed, asks the blockchain for a gas estimate, current average gas cost and EOA balance, then delivers
42: * it to the asking function with a Promise.
43: */
44: export const getTransactionParameters = (web3, method, account, otherParams) =>
45:   new Promise((resolve, reject) => {
46:     let gas = null, balance = null, gasPrice = null;
47:     const params = {
48:       from: account,
49:       ...otherParams
50:     };
51:     method.estimateGas(params).then(
52:       (result) => {
53:         gas = result.toString();
54:         if (balance && gasPrice) resolve({balance, gas, gasPrice});
55:       },
56:       (error) => {
57:         reject({type: "estimateGas", stackTrace: error});
58:       }
59:     );
60:     web3.eth.getBalance(account).then(
61:       (result) => {
62:         balance = result.toString();

```

```

63:         if (gas && gasPrice) resolve({balance, gas, gasPrice});
64:       },
65:       (error) => {
66:         reject({type: "getBalance", stackTrace: error});
67:       }
68:     );
69:     web3.eth.getGasPrice().then(
70:       (result) => {
71:         gasPrice = result.toString();
72:         if (gas && balance) resolve({balance, gas, gasPrice});
73:       },
74:       (error) => {
75:         reject({type: "getGasPrice", stackTrace: error});
76:       }
77:     )
78:   });

```