DAFTAR PUSTAKA

Adi, Laurensius, Rizky Januar Akbar, and Wijayanti Nurul Khotimah. 2017. "Platform E-Learning Untuk Pembelajaran Pemrograman *Web* Menggunakan Konsep Progressive *Web* Apps." 6(2):2–6.

Anonym. n.d. "Www.reactenlightenment.com."

Batubara, Febrin Aulia. 2015. "PERANCANGAN *WEB*SITE PADA PT . RATU ENIM PALEMBANG." 15–27.

Butkiewicz, Michael. 2011. "Understanding *Web*site Complexity : Measurements , Metrics , and Implications Categories and Subject Descriptors."

Colanus, Ivo, Rally Drajana, and Feature Selection. 2017. "METODE SUPPORT VECTOR MACHINE DAN FORWARD SELECTION PREDIKSI PEMBAYARAN PEMBELIAN BAHAN BAKU." 9:116–23.

Haidar Dzacko. 2007. "1 . BASIS DATA (DATABASE)."

Hannonen, Daria. 2017. "Development of *Web*site Solution for Association to Assist Young Professionals." (November).

Khuat, Tung. 2018. "Developing a Frontend Application Using ReactJS and Redux."

Mozahhebi, Massih. 2013. "Comparison of IndexedDB and SQLite Based on Developers ' Concerns."

Muhammad Agung Rizkyana, R.Sandhika Galih Amalga. 2014. "RANCANGAN ARSITEKTUR APLIKASI PENGUMPULAN TUGAS DENGAN PUSH NOTIFICATION REAL-TIME MENGGUNAKAN." 2014(semnasIF):70–75.

No, Vol and Apri Junaidi. 2016. "Studi Perbandingan Performansi Antara MongoDB Dan MySQL Dalam Lingkungan Big Data." 2(1):460–65.

Permana, Endang Cahya. 2016. "Penulisan Fungsi Pada Javascript." 1–27.

Tamire, Workneh Tefera. 2016. "HTML5 and Its Capability to Develop *Offline Web* Applications." (April).

Vanhala, Janne. 2017. "Implementing an *Offline* First *Web* Application."

LAMPIRAN

1. ***Source code service worker***

```
// In production, we register a service worker to serve
assets from local cache.

// This lets the app load faster on subsequent visits in
production, and gives
// it offline capabilities. However, it also means that
developers (and users)
// will only see deployed updates on the "N+1" visit to a
page, since previously
// cached resources are updated in the background.

// To learn more about the benefits of this model, read
https://goo.gl/KwvDNy.
// This link also includes instructions on opting out of
this behavior.

const isLocalhost = Boolean(
  window.location.hostname === 'localhost' ||
    // [::1] is the IPv6 localhost address.
    window.location.hostname === '[::1]' ||
    // 127.0.0.1/8 is considered localhost for IPv4.
    window.location.hostname.match(
      /^127(?:\.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-
9]?)){3}$/
    )
);

export default function register() {
  if (process.env.NODE_ENV === 'production' &&
'serviceWorker' in navigator) {
    // The URL constructor is available in all browsers
that support SW.
    const publicUrl = new URL(process.env.PUBLIC_URL,
window.location);
    if (publicUrl.origin !== window.location.origin) {
      // Our service worker won't work if PUBLIC_URL is on
different origin
```

```
      // from what our page is served on. This might
happen if a CDN is used to
      // serve assets; see
https://github.com/facebookincubator/create-react-
app/issues/2374
      return;
    }

    window.addEventListener('load', () => {
      const swUrl = `${process.env.PUBLIC_URL}/service-
worker.js`;

      if (isLocalhost) {
        // This is running on localhost. Lets check if a
service worker still exists or not.
        checkValidServiceWorker(swUrl);

        // Add some additional logging to localhost,
pointing developers to the
        // service worker/PWA documentation.
        navigator.serviceWorker.ready.then(() => {
          console.log(
            'This web app is being served cache-first by a
service ' +
              'worker. To learn more, visit
https://goo.gl/SC7cgQ'
          );
        });
      } else {
        // Is not local host. Just register service worker
        registerValidSW(swUrl);
      }
    });
  }
}

function registerValidSW(swUrl) {
  navigator.serviceWorker
    .register(swUrl)
    .then(registration => {
      registration.onupdatefound = () => {
        const installingWorker = registration.installing;
```

```
        installingWorker.onstatechange = () => {
          if (installingWorker.state === 'installed') {
            if (navigator.serviceWorker.controller) {
              // At this point, the old content will have
been purged and
              // the fresh content will have been added to
the cache.
              // It's the perfect time to display a "New
content is
              // available; please refresh." message in
your web app.
              console.log('New content is available;
please refresh.');
            } else {
              // At this point, everything has been
precached.
              // It's the perfect time to display a
              // "Content is cached for offline use."
message.
              console.log('Content is cached for offline
use.');
            }
          }
        };
      };
    })
    .catch(error => {
      console.error('Error during service worker
registration:', error);
    });
}

function checkValidServiceWorker(swUrl) {
  // Check if the service worker can be found. If it can't
reload the page.
  fetch(swUrl)
    .then(response => {
      // Ensure service worker exists, and that we really
      getting a JS file.
      if (
        response.status === 404 ||
```

```
        response.headers.get('content-
type').indexOf('javascript') === -1
      ) {
        // No service worker found. Probably a different
app. Reload the page.
        navigator.serviceWorker.ready.then(registration =>
{
          registration.unregister().then(() => {
            window.location.reload();
          });
        });
      } else {
        // Service worker found. Proceed as normal.
        registerValidSW(swUrl);
      }
    })
    .catch(() => {
      console.log(
        'No internet connection found. App is running in
offline mode.'
      );
    });
}

export function unregister() {
  if ('serviceWorker' in navigator) {
    navigator.serviceWorker.ready.then(registration => {
      registration.unregister();
    });
  }
}
```

2. ***Source code* Mesin.js**

```
import React from 'react';
import PropTypes from 'prop-types';
import classNames from 'classnames';
import { withStyles } from '@material-ui/core/styles';
rt CssBaseline from '@material-ui/core/CssBaseline';
rt Drawer from '@material-ui/core/Drawer';
rt AppBar from '@material-ui/core/AppBar';
rt Toolbar from '@material-ui/core/Toolbar';
rt List from '@material-ui/core/List';
```

```
import Typography from '@material-ui/core/Typography';
import Divider from '@material-ui/core/Divider';
import IconButton from '@material-ui/core/IconButton';
import Badge from '@material-ui/core/Badge';
import MenuIcon from '@material-ui/icons/Menu';
import ChevronLeftIcon from '@material-
ui/icons/ChevronLeft';
import NotificationsIcon from '@material-
ui/icons/Notifications';
import { mainListItems, secondaryListItems } from
'@material-ui/core/ListItem';
import ListItem from '@material-ui/core/ListItem';
import ListItemText from '@material-
ui/core/ListItemText';
import Avatar from '@material-ui/core/Avatar';
import ImageIcon from '@material-ui/icons/Image';
import TextField from '@material-ui/core/TextField';
import WorkIcon from '@material-ui/icons/Work';
import {Link} from 'react-router-dom';
import Button from '@material-ui/core/Button';
import Grid from '@material-ui/core/Grid';
import ListItemIcon from '@material-
ui/core/ListItemIcon';
import Paper from '@material-ui/core/Paper';
//import ListItemText from '@material-
ui/core/ListItemText';
import ListSubheader from '@material-
ui/core/ListSubheader';
import AssignmentIcon from '@material-
ui/icons/Assignment';
import * as moment from 'moment';
import FormControlLabel from '@material-
ui/core/FormControlLabel';
import Checkbox from '@material-ui/core/Checkbox';
import * as RxDB from 'rxdb';
import {QueryChangeDetector} from 'rxdb';
import { skema } from './Schema';
import { BrowserRouter, Route} from 'react-router-dom';
rt { ToastContainer, toast } from 'react-toastify';
rt Icon from '@material-ui/core/Icon';
rt Visibility from '@material-ui/icons/Visibility';
```

```
import VisibilityOff from '@material-
ui/icons/VisibilityOff';
import InputAdornment from '@material-
ui/core/InputAdornment';
import SaveIcon from '@material-ui/icons/Save';
import 'react-toastify/dist/ReactToastify.css';
import Table from '@material-ui/core/Table';
import TableBody from '@material-ui/core/TableBody';
import TableCell from '@material-ui/core/TableCell';
import TableHead from '@material-ui/core/TableHead';
import TableRow from '@material-ui/core/TableRow';
import View from './View';
import Edit from './Edit';
import Home from './Home';
const drawerWidth = 240;


QueryChangeDetector.enable(); // kita membutuhkan query
change detector untuk meningkatkan performa query
QueryChangeDetector.enableDebugging(); //karena RxDb yang
akan meminta request data ketika terjadi perubhan maka
dia akan memberatkan query makanya pake kode ini

RxDB.plugin(require('pouchdb-adapter-idb')); //
inisialisasi database local using rxdb
RxDB.plugin(require('pouchdb-adapter-http'));//
inisialisasi remote db

const syncURL = 'https://98855c8d.ngrok.io/'; // remote
db
const dbName = 'mesinfixfix'; // name local db

const styles = theme => ({
  root: {
    display: 'flex',
    flexWrap: 'wrap',
    ...theme.mixins.gutters(),
    paddingTop: theme.spacing.unit * 2,
    paddingBottom: theme.spacing.unit * 2,
    width: '100%',
    marginTop: theme.spacing.unit * 3,
    overflow:'hidden !important',
```

```
  },
  table: {
    minWidth: 700,
  },
  container: {
    display: 'flex',
    flexWrap: 'wrap',
  },
  margin: {
    margin: theme.spacing.unit,
  },
  textField: {
    flexBasis: 200,
  },
  dense: {
    marginTop: 16,
  },
  menu: {
    width: 200,
  },
  toolbar: {
    paddingRight: 24, // keep right padding when drawer
closed
  },
  toolbarIcon: {
    display: 'flex',
    alignItems: 'center',
    justifyContent: 'flex-end',
    padding: '0 8px',
    ...theme.mixins.toolbar,
  },
  appBar: {
    zIndex: theme.zIndex.drawer + 1,
    transition: theme.transitions.create(['width',
'margin'], {
      easing: theme.transitions.easing.sharp,
      duration: theme.transitions.duration.leavingScreen,
    }),

    pBarShift: {
    marginLeft: drawerWidth,
    width: `calc(100% - ${drawerWidth}px)`,
```

```
    transition: theme.transitions.create(['width',
'margin'], {
      easing: theme.transitions.easing.sharp,
      duration:
theme.transitions.duration.enteringScreen,
    }),
  },
  menuButton: {
    marginLeft: 12,
    marginRight: 36,
  },
  menuButtonHidden: {
    display: 'none',
  },
  title: {
    flexGrow: 1,
  },
  drawerPaper: {
    position: 'relative',
    whiteSpace: 'nowrap',
    width: drawerWidth,
    transition: theme.transitions.create('width', {
      easing: theme.transitions.easing.sharp,
      duration:
theme.transitions.duration.enteringScreen,
    }),
  },
  drawerPaperClose: {
    overflowX: 'hidden',
    transition: theme.transitions.create('width', {
      easing: theme.transitions.easing.sharp,
      duration: theme.transitions.duration.leavingScreen,
    }),
    width: theme.spacing.unit * 7,
    [theme.breakpoints.up('sm')]: {
      width: theme.spacing.unit * 9,
    },
  },
    pBarSpacer: theme.mixins.toolbar,
    ntent: {
    flexGrow: 1,
    padding: theme.spacing.unit * 3,
```

```
    height: '100vh',
    overflow: 'auto',
  },
  chartContainer: {
    marginLeft: -22,
  },
  tableContainer: {
    height: 320,
  },
});
let id = 0;
function createData(name, calories, fat, carbs, protein)
{
  id += 1;
  return { id, name, calories, fat, carbs, protein };
}

const rows = [
createData('Frozen yoghurt', 159, 6.0, 24, 4.0),
createData('Ice cream sandwich', 237, 9.0, 37, 4.3),
createData('Eclair', 262, 16.0, 24, 6.0),
createData('Cupcake', 305, 3.7, 67, 4.3),
createData('Gingerbread', 356, 16.0, 49, 3.9),
];
class Mesin extends React.Component {
  state = {
    open: true,
  };


  constructor(props) {
    super(props);
    this.state = {
      nama_mesin: '',
      status: '',
      downTime:'',
      costDate:'',
      data : '',
      contoh:[],
      mesin: [],

    };
```

```
    this.subs = [];
    this.addMessage = this.addMessage.bind(this);
    // this.DeleteMessage =
this.DeleteMessage.bind(this);
    this.handleMessageChange =
this.handleMessageChange.bind(this);
    //console.log("ini "+ window.location.href)
  }
  // 1) buat method pembuatan db dengan cara awit

  handleChange = prop => event => {
    this.setState({ [prop]: event.target.value });
  };

  async createDatabase() {
    // password must have at least 8 characters dan ini
fungsinya untuk enkripsi data yang masuk ke dalam koleksi
data
    const db = await RxDB.create(
      {name: dbName, adapter: 'idb', password:
'12345678', ignoreDuplicate: true}
      );
    console.dir(db);

    // show who's the leader in page's title
    db.waitForLeadership().then(() => {
      document.title = 'Home ' + document.title;
    });
    // leader ellection algorithm, dia membuat satu tab
hanya me manage remote db

    // create collection
    const mesinCollection = await db.collection({
      name: 'mesin',
      schema: skema
    });

    // set up replication
    const replicationState = mesinCollection.sync({
    te: syncURL + dbName + '/' });
    this.subs.push(
```

```
      replicationState.change$.subscribe(change => {
        toast('Replication change');
        console.dir(change)
      })
      );
    this.subs.push(
      replicationState.docs$.subscribe(docData =>
console.dir(docData))
      );
    this.subs.push(
      replicationState.active$.subscribe(active =>
toast(`Replication active: ${active}`))
      );
    this.subs.push(
      replicationState.complete$.subscribe(completed =>
toast(`Replication completed: ${completed}`))
      );
    this.subs.push(
      replicationState.error$.subscribe(error => {
        toast('Replication Error');
        console.dir(error)
      })
      );

    return db;
  }

  async componentDidMount() {
    this.db = await this.createDatabase();

    // Subscribe to query to get all messages
    const sub =
    this.db.mesin.find().where('tipe').eq('Mesin
Tools').sort({id: 1}).$.subscribe(mesin => {
      if (!mesin)
        return;
      toast('Reloading Data');
      this.setState({mesin: mesin});
    });
    this.subs.push(sub);
```

```
  componentWillUnmount() {
    // Unsubscribe from all subscriptions
    this.subs.forEach(sub => sub.unsubscribe());
  }

  render() {

    const
{data,status,contoh,mesin,downTime,costDate,date} =
this.state;
    return (

      <div>

      <div>
      <Link to="/mesin">Mesin</Link>
      <Link to="/maintenance">Maintenance</Link>
      </div>

      <div>
      <p>{this.addData()}</p>

      </div>

      </div>
      );
  }
  addData = () => {
    const { classes } = this.props;
    const
{data,status,contoh,mesin,downTime,costDate,date} =
this.state;
    return (
      <main className={classes.content}>
      <div className={classes.appBarSpacer} />
      <Paper className={classes.root} elevation={1}>
      <Typography variant="headline" component="h3">
      Tambah Data Mesin
      </Typography>

      </Paper>
      <br />
```

```
      <br />
      <form className={classes.container} noValidate
autoComplete="off">

      <TextField
      id="outlined-adornment-weight"
      className={classNames(classes.margin,
classes.textField)}
      variant="outlined"
      name="nama_mesin"
      label="Nama Mesin"
      value={this.state.nama_mesin}
      onChange={(e)=>this.handleMessageChange(e)}
      helperText="Nama Mesin"
      InputProps={{
        endAdornment: <InputAdornment
position="end"></InputAdornment>,
      }}
      />
      <TextField
      id="outlined-adornment-weight"
      className={classNames(classes.margin,
classes.textField)}
      variant="outlined"
      name="status"
      label="Status Mesin"
      value={status}
      onChange={(e)=>this.handleMessageChange(e)}
      helperText="Status"
      InputProps={{
        endAdornment: <InputAdornment
position="end"></InputAdornment>,
      }}
      />
      <TextField
      id="outlined-adornment-weight"
      className={classNames(classes.margin,
classes.textField)}
      variant="outlined"
      name="downTime"
      type="date"
      label=" "
```

```
      value={downTime}
      onChange={(e)=>this.handleMessageChange(e)}
      helperText="Down Time"
      InputProps={{
        endAdornment: <InputAdornment
position="end"></InputAdornment>,
      }}
      />
      <TextField
      id="outlined-adornment-weight"
      className={classNames(classes.margin,
classes.textField)}
      variant="outlined"
      name="costDate"
      label="Cost"
      value={costDate}
      onChange={(e)=>this.handleMessageChange(e)}
      helperText="Down Time"
      InputProps={{
        endAdornment: <InputAdornment
position="end"></InputAdornment>,
      }}
      />


      </form>
      <br />
      <br/>
      <Button onClick={()=>this.addMessage()}
variant="contained" color="primary" >
      <SaveIcon className={classNames(classes.leftIcon,
classes.iconSmall)} />
      Save
      </Button>
      <br />
      <br/>
      <Paper className={classes.root} elevation={1}>
      <Typography variant="headline" component="h3">
      Data Mesin
      </Typography>

      </Paper>
```

```
      <br />
      <br />
      <TableHead>
      <TableRow>
      <TableCell>Machine Name</TableCell>
      <TableCell align="right">Status Mesin</TableCell>
      <TableCell align="right">Down Time</TableCell>
      <TableCell align="right">Cost (g)</TableCell>
      <TableCell align="right">Date</TableCell>
      </TableRow>
      </TableHead>
      {this.renderMessages()}

      </main>




      )
  }

  renderMessages = () =>{
    return this.state.mesin.map(({id,
nama_mesin,status,downTime,costDate,date}) => {
      const waktu Input = moment(id, 'x').fromNow();


      return (

        <Paper>



        <TableBody>


        <TableRow>
        <TableCell component="th" scope="row">
        {nama_mesin}
        </TableCell>
        <TableCell align="right">
```

```
        {status}
        </TableCell>
        <TableCell numeric>
        {downTime}
        </TableCell>
        <TableCell numeric>
        {costDate}
        </TableCell>
        <TableCell numeric>
        {date}
        </TableCell>
        <TableCell numeric>
        <Link to={`/mesin/view/${id}`}><Button
variant="contained" color="primary">
        View
        </Button></Link>
        </TableCell>
        <TableCell numeric>
        <Button variant="contained" color="secondary"
onClick={()=>{this.DeleteMessage(id)}}>Delete</Button>
        </TableCell>
        </TableRow>


        </TableBody>

        </Paper>
        );
      });
    }
    handleMessageChange(e) {
      this.setState({[e.target.name]: e.target.value});
    }



    async addMessage() {
      var today = new Date();
      var dd = today.getDate();
      var mm = today.getMonth() + 1; //January is 0!
      var yyyy = today.getFullYear();
```

```
      if (dd < 10) {
        dd = '0' + dd;
      }

      if (mm < 10) {
        mm = '0' + mm;
      }
      const hari = mm + '/' + dd + '/' + yyyy;

      const id = Date.now().toString();
      const {nama_mesin,status,downTime,costDate} =
this.state
      const mesin = nama_mesin;
      const tipe = 'Mesin Tools';
      const date = hari.toString();
      const dataMesin =
{id,nama_mesin,status,downTime,costDate,date,tipe};
      await this.db.mesin.insert(dataMesin);

      this.setState({nama_mesin: ''});
      this.setState({status:''});
      this.setState({downTime: ''});
      this.setState({costDate:''});


    }


    DeleteMessage = async (id) =>{
      await this.db.mesin.findOne().where('_id').eq(id)
      .remove();


      // this.db.messages.find().exec() // <- find all
documents
      //.then(documents => console.log(id));
      // console.log(this.state.messages.map);
    }

    editData = async (id) => {
```

```
      //this.setState({newMessage: ''});
      const query =
this.db.mesin.find().where('_id').gt(id);
      // await query.update({
        //  $inc: {
          //        me: 1 // increases age of every found
document by 1
          //       }
          console.log(query);
          //});

        }

        //-----------------------------------------------
-------------------------------------------

        getData = async (id) => {
          //await
this.db.messages.findOne().where('_id').eq(id).exec().the
n(doc => console.log(doc));;
          await this.db.mesin.findOne(id).exec().then(doc
=> {

            console.log(doc._data)
            this.setState({contoh : doc._data})
            setTimeout(()=>{
              console.log(this.state.contoh)
            },2000)
          });
        }

        //-----------------------------------------------
-------------------------------------------
        sgetData = async (id) =>{
          //  await
this.db.messages.findOne().where('_id').eq(id).exec().the
n(doc => console.log(doc));;
          await this.db.mesin.findOne(id).exec().then(doc
onsole.log(doc._data)) ;

        }
```

```
    updateData = async (id) => {
      this.getData(id);
      console.log(id)
    }

    dataContoh = () => {
      const {data} = this.state
      data === [] ? <p>Kosong</p> : <p>Ada</p>
    }

  }

  Mesin.propTypes = {
    classes: PropTypes.object.isRequired,
};

export default withStyles(styles)(Mesin);
```