Mika Suomi

WWP-PALVELUN MODERNISOINTI

Tietojenkäsittelyn koulutusohjelma

2019

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

WWP-PALVELUN MODERNISOINTI

Suomi, Mika
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Toukokuu 2019
Sivumäärä: 26
Liitteitä:

Asiasanat: reactjs, ohjelmistokehitys, javascript

Tämän opinnäytetyön tarkoituksena oli uudistaa vanhaa internetportaalia nimeltä WWP (Warehouse Web Portal). Tehtävän toimeksiantajana toimi Cimcorp Oy ja se toteutettiin yrityksen tuotekehitysosastolla.

Tarkoituksena oli tutustua uusiin web-tekniikoihin, niiden toimivuuteen yrityksen käytössä sekä uusia vanha WWP-palvelu. Uuden sivuston tavoitteena oli olla responsiivinen web-käyttöliittymäsovellus ja tyyliltään yhtenevä yrityksen nykyisen ulkoasun kanssa.

Tässä työssä käyn läpi käytetyt tekniikat, sovelluksen rakenteen sekä lopputuloksen.

MODERNIZATION OF THE WWP SERVICE

Suomi, Mika
Satakunta University of Applied Sciences
Degree Programme in Business Information Technology
May 2019
Number of pages: 26
Appendices:

Keywords: reactjs, software development, javascript

The purpose of this thesis was to renew old web portal called WWP (Warehouse Web Portal). The client was Cimcorp Oy and the thesis was completed in the company's software development department.

The purpose was to get information about new web technologies, how they fit the company's purposes and to renew old WWP service. The goal of the new site was to be a responsive web interface application and stylistically consistent with the company's current look.

This thesis covers the techniques used, the structure of the application and the result.

SISÄLLYS

1 JOHDANTO

Aihe työn toteuttamiseen tuli Cimcorp Oy:lta. Cimcorp on 1975 perustettu sisälogistiikan automaatiojärjestelmiin erikoistunut yritys ja sen pääkonttori sijaitsee Ulvilassa. Cimcorpin tytäryhtiöt sijaitsevat USA:ssa, Kanadassa ja nykyisin myös Intiassa. (Cimcorp Intranet 2019)

Työn tavoitteena oli uudistaa yrityksen vanhaa internetportaalia nimeltään WWP (Warehouse Web Portal). Samalla oli myös tarkoitus tutustua moderneihin web-tekniikoihin sekä niiden sopivuuteen yrityksen tarpeisiin. WWP on asiakkaille toimitettava internetportaali, johon käyttäjä kirjautuu tunnuksillaan. Palvelussa on alussa navigointisivu, jonka sisältö määräytyy käyttäjän oikeuksien mukaan. WWP-palveluun kuuluu kolme sivua. Ensimmäinen sisältää asiakkaalle toimitettavien robottien huolto- ja ohjekirjoja sekä mahdollisuuden lukea niitä sivustolla. Toisella sivulla voi ladata yrityksen tarjoamia sovelluksia tai päivityksiä. Kolmannella sivulla käyttäjä löytää tärkeitä linkkejä kuten erilaisten tulostimien, reitittimien ja muiden verkossa olevien laitteiden osoitteita. Palveluun on tarkoitus tulla myös asiakkaalle erikseen tarjottava raporttiosa, josta löytyy erilaisia raportteja asiakkaan hankkiman laitteiston toimivuudesta.

Työssä vaatimuksena oli, että sivusto toteutettaisiin JavaScriptillä ja siinä käytettäisiin React-käyttöliittymäkirjastoa sekä sitä tukevia apukirjastoja, kuten Reduxia.

2 TYÖSSÄ KÄYTETYT TEKNIIKAT

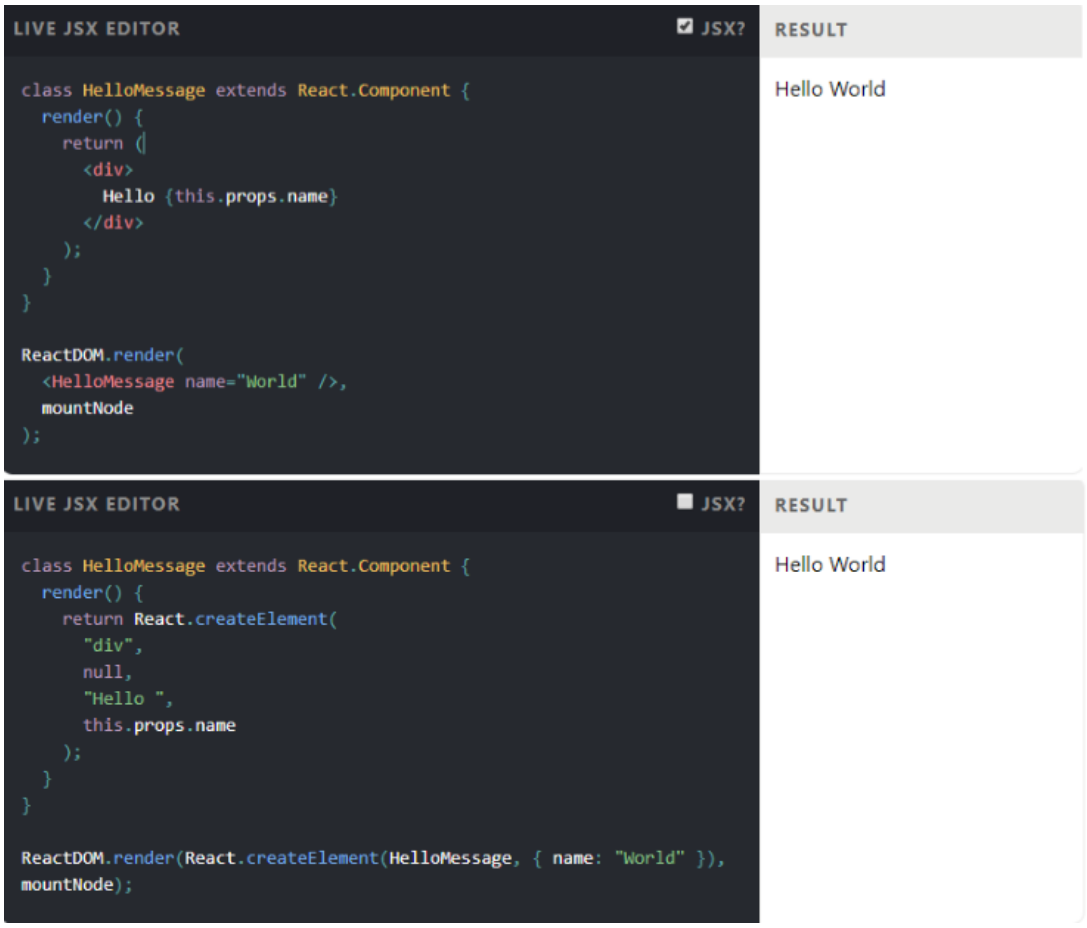
Tässä osiossa kuvataan työssä käytettyjä keskeisiä tekniikoita. Tekniikoita valittaessa pyrittiin siihen, että tekniikoita käytettäisiin tulevaisuudessa muissakin tuotekehityksen web-projekteissa.

2.1 React

ReactJS on Facebookin vuonna 2013 kehittämä JavaScript-kirjasto, joka on tehty käyttöliittymien kirjoittamiseen. Reactin perusideana on, että se perustuu komponentteihin, jolloin isommissa projekteissa on mahdollista käyttää samoja komponentteja uudelleen ja selkeyttää sovelluksen rakennetta. React ei nykyisellään rajoitu pelkästään internetiin tehtäviin Single Page App -tyylisiin sovelluksiin vaan sitä on mahdollista käyttää myös mobiiliohjelmoinnissa sekä työpöytäsovelluksissa. Syyskuussa 2017 Facebook julkisti Reactin MIT-lisenssin alle. (React kotisivut 2018)

2.1.1 JSX

JSX on vapaavalintainen, mutta Facebookin vahvasti suosittelema syntaksi React-koodin kirjoittamiseen. JSX:n syntaksi näyttää HTML-koodilta eli se on XML:n kaltainen. JSX:n avulla on helppo jakaa komponenttien toiminnallisuus eri osa-alueisiin. Webkehittäjille tutun syntaksinsa ansiosta, JSX tekee koodista helppolukuisemman. (React kotisivut 2018)



Kuva 1. Ylempänä JSX-syntaksilla tehty komponentti ja alempana puhtaalla JavaScriptillä tehty.

2.1.2 Komponentit

Reactin komponentit on tapana jakaa kahteen osaan, jotka ovat: Presentational component(esityskomponentti) ja Container component(säilytyskomponentti). Tämä mahdollistaa selkeämmän käsitteiden erottamisen ja helpottaa sovelluksen toiminnan ymmärtämistä.

Presentational component:

- Määrittää miltä komponentti näyttää ruudulle renderöitynä.
- Ei sisällä logiikkaa datan tilan hallintaan.
- Sisältö tulee erilliseltä loogiselta komponentilta propseina.
- Ei sisällä riippuvuuksia muihin osiin sovellusta.

Container component:

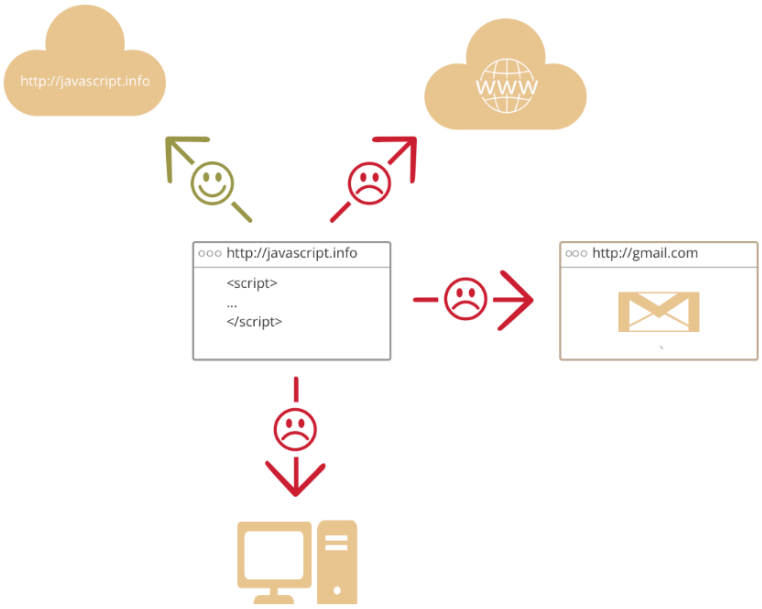
- Toiminnallinen komponentti, joka vastaa datan käsittelystä.
- Sisältää yhteyden erilliseen tilan hallintaan esim. Redux.
- Lähettää datan esityskomponentille.

(Medium 2018)

2.2 JavaScript

JavaScript luotiin alun perin tekemään www-sivuista dynaamisempia. Brendan Eich kehitti sen vuonna 1995 Netscape Navigator -selaimeen. JavaScriptin standardi on nimeltään ECMAScript ja toistaiseksi sen suurin versio on ES6 tai toiselta nimeltään ECMAScript2015.

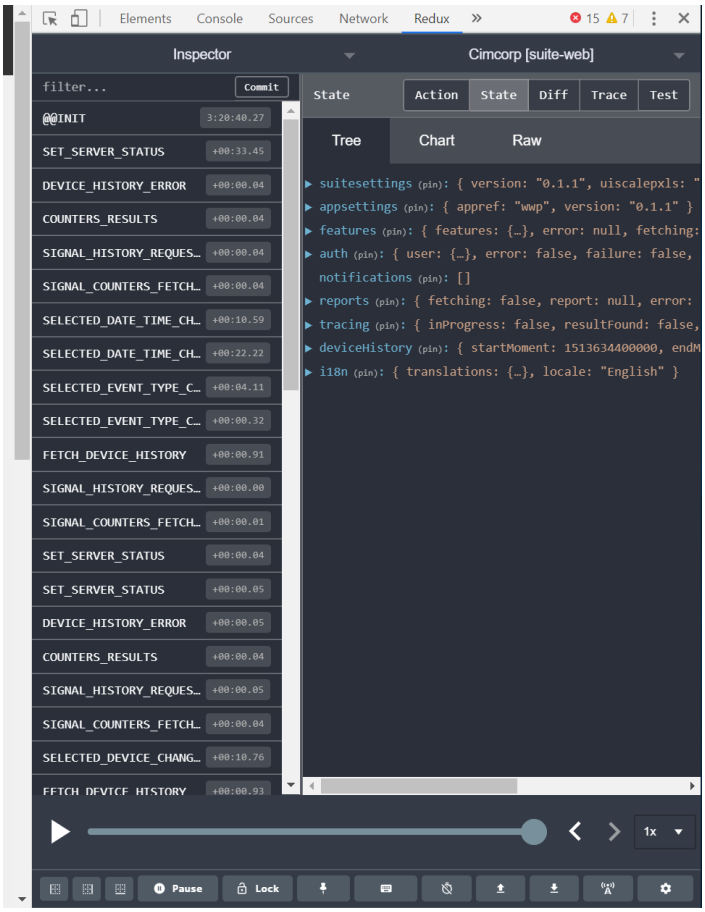
JavaScriptin suoritus tapahtuu yleensä selaimessa, joskin nykyisin node.js mahdollistaa myös back-end-kehityksen JavaScriptillä. JavaScriptin käyttämiseksi selaimessa on oltava tämän mahdollistava ”kone” eli tulkkaaja. Yleisin JavaScript -kääntäjä on Googlen Chromessa sekä Chromiumissa oleva V8. JavaScriptillä on rajoituksensa, kuten se ei esimerkiksi voi lukea tai tehdä muutoksia kovalevyllä oleviin tiedostoihin eikä kommunikoida toisten selainikkunoiden kanssa. (JavaScript info 2019)



Kuva 2. JavaScriptin rajoitukset. (JavaScript info 2019)

2.3 Redux

Web-sovelluksissa komponenttien tilanhallinta tapahtuu välittämällä tila loogiselta komponentilta niitä tarvitseville näyttökomponenteille. Tilanhallinta käy nopeasti työlääksi, joten alun perin Facebook kehitti Flux-arkkitehtuurin helpottamaan tilanhallintaa. Nykyisin käytössä on kuitenkin Fluxiin perustuva, mutta yksinkertaisempi malli nimeltään Redux. Selaimeen saatava Redux devtools -selainlaajennus auttaa paljon tilan hallinnassa, kun sovelluksen muuttuvia tiloja voi havainnoida reaaliaikaisesti. (Flux -dokumentit 2018)

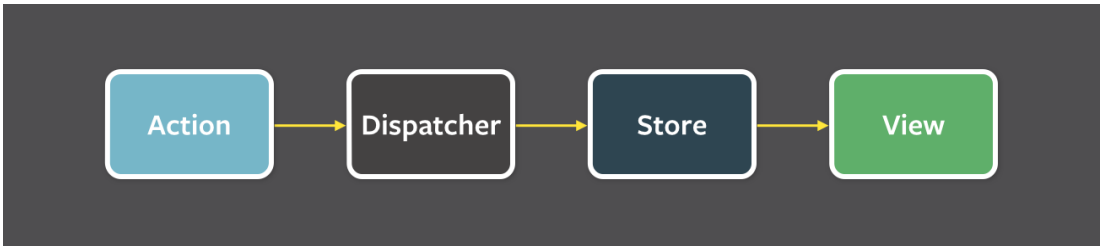


Kuva 3. Redux devtools.

2.3.1 Flux

Fluxissa ja Reduxissa komponenttien tila säilytetään tilavarastossa, storessa. Fluxissa voi olla useita storeja ja myös sovelluksen logiikka säilytetään storessa. Tilaa storessa

ei pysty suoraan muuttamaan, vaan siihen vaaditaan tapahtuman eli actionin lähettäminen, joka muuttaa tilan storessa ja se puolestaan päivittää näkymän uudelleen. (Flux -dokumentit 2018)



Kuva 4. Tilan yksisuuntainen liike Flux-mallissa. (Flux-dokumentit 2018)

2.3.2 Reduxin periaatteet

Reduxissa on vain yksi store käytössä koko sovelluksessa, jotta sovelluksen tilaa olisi mahdollisimman helppo seurata. Reduxissa tilaa voi vain lukea ja ainoa keino muuttaa sitä on lähettää actionin avulla pyyntö reducerille, joka palauttaa uuden halutun tilan.

2.4 React router

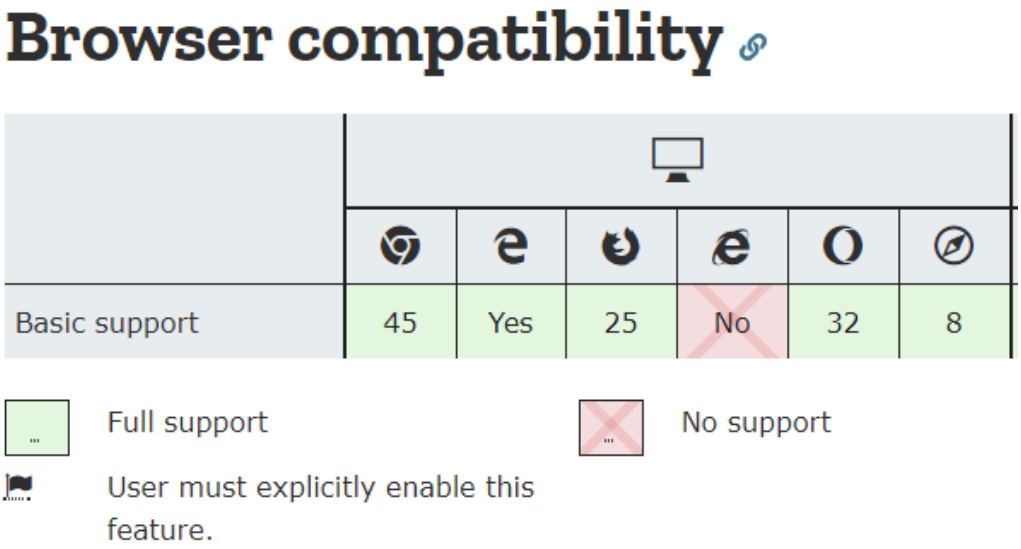
React router on kokoelma komponentteja, jotka mahdollistavat sovelluksessa navigoinnin, erilaisten komponenttien lataamisen tietyssä tilanteessa tai kirjanmerkkeihin laitettavan osoitteen. SPA-tyyliset sovellukset koostuvat vain yhdestä HTML-tiedostosta, johon JavaScript renderöi uutta sisältöä. Tällöin navigointi on hieman erilaista, sillä React router lataa halutun komponentin sille annetulla URL-osoitteella. (React router kotisivu 2019)

```
<Route exact path="/" component={Home} />
<Route path="/about" component={About} />
<Route path="/topics" component={Topics} />
```

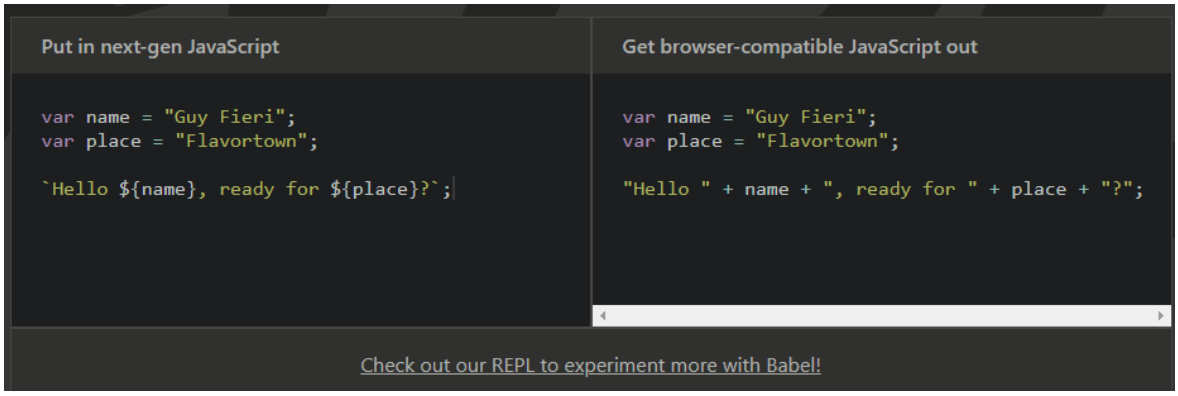
Kuva 5. React routerilla ohjaaminen haluttuun komponenttiin käyttäen annettua URL-osoitetta.

2.5 Babel

Babel on tällä hetkellä johtava työkalu JavaScriptin sekä Reactin käyttämän JSX:n kääntämiseen. Kääntämistä tarvitaan, jotta ohjelmaa tehdessä voidaan kirjoittaa uudempaa JavaScriptin versiota, jonka Babel lopuksi kääntää ES5 yhteensopivaksi, jota käytännössä kaikki selaimet tukevat. Koodissa saatetaan myös käyttää jotakin sellaista tekniikkaa, jota osa selaimista kuten Internet Explorer ei osaa edes käännettynä, kuten Promiset. Babel tarjoaa myös käyttöön polyfillit, joka lisää koodin tarvitsemat toiminnallisuudet vanhoihin selaimiin. (Babel kotisivut 2019)



Kuva 6. MDN-sivulta otettu esimerkki siitä miten selaimet tukevat Array.find -komentoa. (MDN network 2019)



Kuva 7. Vasemmalla uudempaa JavaScriptiä ja oikealla Babelin kääntämää ja tämän myötä vanhempien selainten ymmärtämää JavaScriptiä.

2.6 HTML5

Hypertext Markup Language eli hypertekstin merkintäkieli on internetsivun rakenteen tärkein osa. Se on avoimesti standardoitu kuvauskieli, jota käytetään nykyisin pääasiassa kielenä, jolla internetsivut kirjoitetaan. Alun perin HTML:n oli tarkoitus kuvata sivujen rakennetta, mutta kehittäjien toivomuksesta kieleen lisättiin myös mahdollisuus dokumentin ulkoasun muokkaamiseen. HTML5-standardi julkaistiin vuonna 2014. (MDN web docs 2019)

2.7 CSS/SASS

CSS eli Cascading Style Sheets on erityisesti HTML-dokumenteille tarkoitettu syntaksiltaan yksinkertainen tyyliohje. CSS-säännöt ehdottavat, miten dokumentti voitaisiin esittää eli ne eivät ole ehdottomia. CSS:n suurimpia etuja on, että dokumentin rakenne ja asiasisältö voidaan erotella täysin tyyleistä, mikä mahdollistaa sen, että dokumentti on helpompi luoda ja ylläpitää.
(webbipakki css intro 2019)

2.8 NPM

NPM on alun perin node.js:lle tarkoitettu paketinhallintaohjelma, mutta nykyisin se on käytössä myös muissa JavaScript-kirjastoissa. NPM:n kautta on mahdollista ladata ohjelman tarvitsemat kirjastot. Avoimen lähdekoodin kehittäjät käyttävät sitä jakaakseen ohjelmia ja yrityksillä saattaa olla maksullinen tili, jolla hallitaan yksityisiä ohjelmia. (w3schools 2019)

2.9 Webpack

Webpack on staattinen moduulien paketoija nykyajan moderneille pääosin JavaScriptillä tehdyille sovelluksille, mutta se osaa myös HTML:n ja CSS:n paketoinnin. Webpackilla ohjelma voidaan paketoida yhteen tai useampaan pakettiin. Käännösvaiheessa tarkastetaan myös, että projektissa olevat riippuvuudet menevät oikein. Yksi

suurimpia syitä Webpackin suosioon on myös sen modulaarisuus, joka mahdollistaa erilaisten pluginien eli liitännäisten lisäämisen esimerkiksi ohjelmakoodin minimoimiseksi tai kääntämiseksi vanhempaan JavaScriptiin samalla kun ohjelma paketoidaan, jolloin näitä vaiheita ei jouduta tekemään erikseen. (Webpack 2019)

2.10 Jest

Jest on JavaScriptille tarkoitettu testauskirjasto, joka pyrkii tekemään ohjelmakoodin testaamisesta helppoa. Jestillä voi suorittaa kuvan 8 mukaisia perinteisiä testejä.

```
const sum = require('./sum');

test('adds 1 + 2 to equal 3', () => {
  expect(sum(1, 2)).toBe(3);
});
```

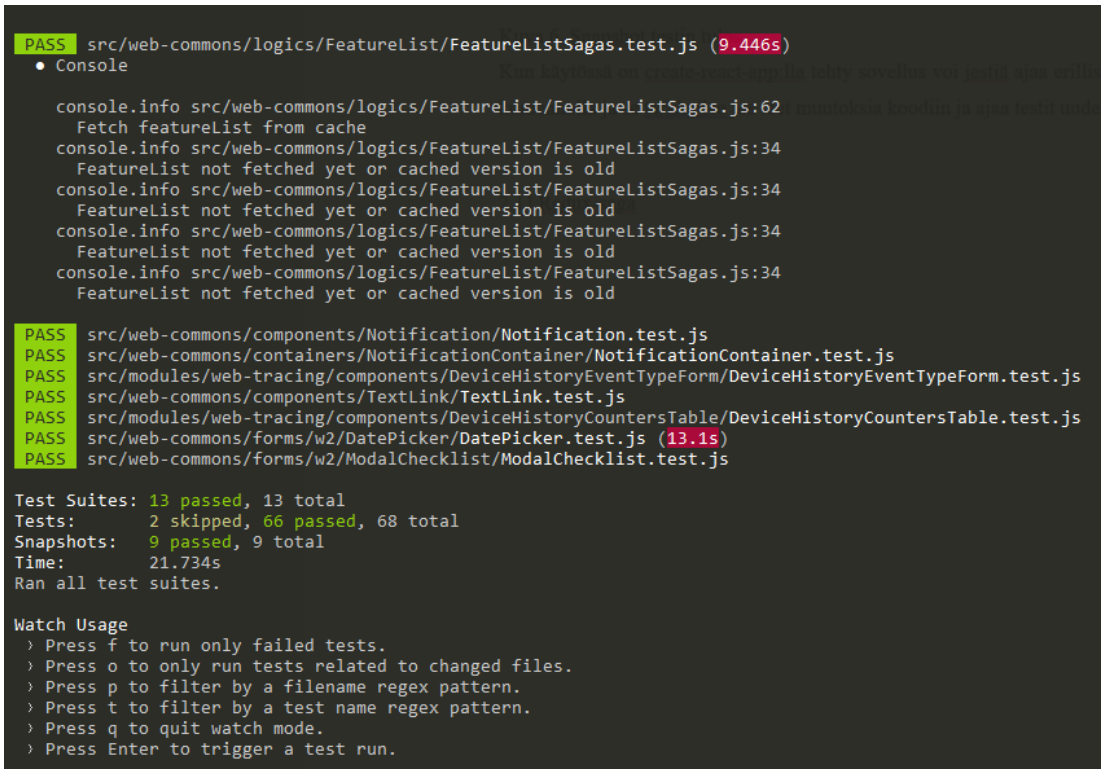
Kuva 8. Jest-testi.

Jestillä voi myös suorittaa niin kutsuttuja snapshot-testejä, joissa luodaan komponentin kaikista tiedoista ja toiminnallisuuksista niin sanottu snapshot (kuva 9.). (Jest kotisivut 2019)

```
exports[`renders correctly 1`] = `
<a
  className="normal"
  href="http://www.facebook.com"
  onMouseEnter={[Function]}
  onMouseLeave={[Function]}
>
  Facebook
</a>
`;
```

Kuva 9. Snapshot-testin tulos.

Kun käytössä on create-react-app:lla tehty sovellus, voi Jestiä ajaa erillisellä npm test -komennolla ja se tarkkailee koodiin tehtäviä muutoksia ajaen tällöin testit uudestaan. (kuva 10.)



Kuva 10. Konsoli, jossa testit ajettu.

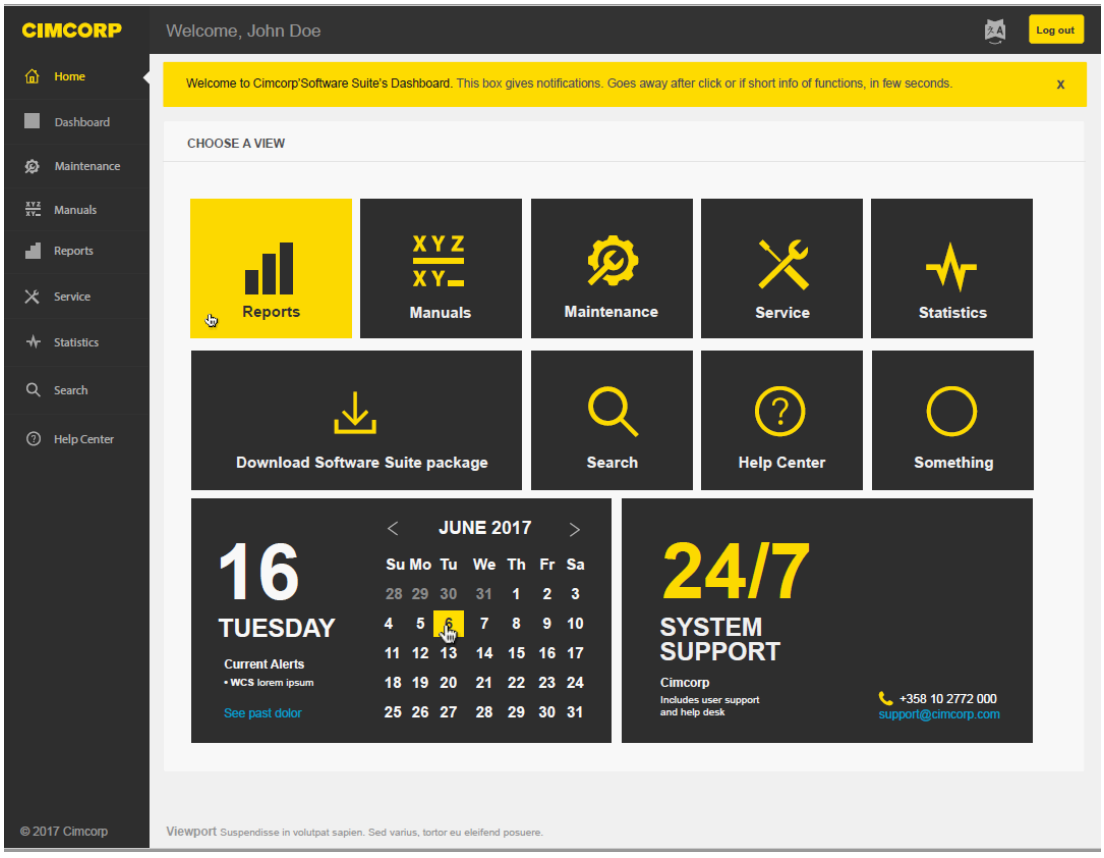
2.11 Redux-saga

Redux-saga -kirjaston tarkoitus on tehdä epäsynkronisia asioita kuten hakea tietoa serveriltä. Se on Reduxin middleware, eli se toimii Reduxin actioneiden kautta ja sillä on pääsy Reduxin hallitsemaan tilaan. Redux-saga käyttää generaattorifunktioita, eli koodia ei välttämättä suoriteta heti, jos dataa ei ole saatavilla. Generaattorifunktio laittaa funktion odottamaan vastausta ja jatkaa muun ohjelman suoritusta, jotta ohjelma ei jäätyisi.
(Redux-saga kotisivu 2019)

3 JÄRJESTELMÄN SUUNNITTELU JA TOTEUTUS

Toiveena oli modernisoida vanha WWP eli Warehouse Web Portal. Uuden järjestelmän toteutuksen lähtökohtana oli toteuttaa helposti erilaisiin projekteihin konfiguroitava järjestelmä, moderneilla työkaluilla sekä yrityksen nykyisellä ulkoasulla. Työssä haluttiin myös saada kokemusta uusista web-tekniikoista, joita yrityksessä voitaisiin myöhemmin hyödyntää muissakin tilanteissa.

Yrityksellä oli vaatimuksena, että työ tehdään Single Page App -tyylisesti moderneilla web-tekniikoilla sekä React-käyttöliittymäkirjastoa käyttäen. Vanhasta järjestelmästä ei sen erilaisen toteutuksen vuoksi saanut tähän työhön mitään apuja, mutta yrityksellä oli Photoshop-ohjelmalla tehtyjä kuvia, miltä suunnilleen käyttöliittymän haluttiin näyttävän.

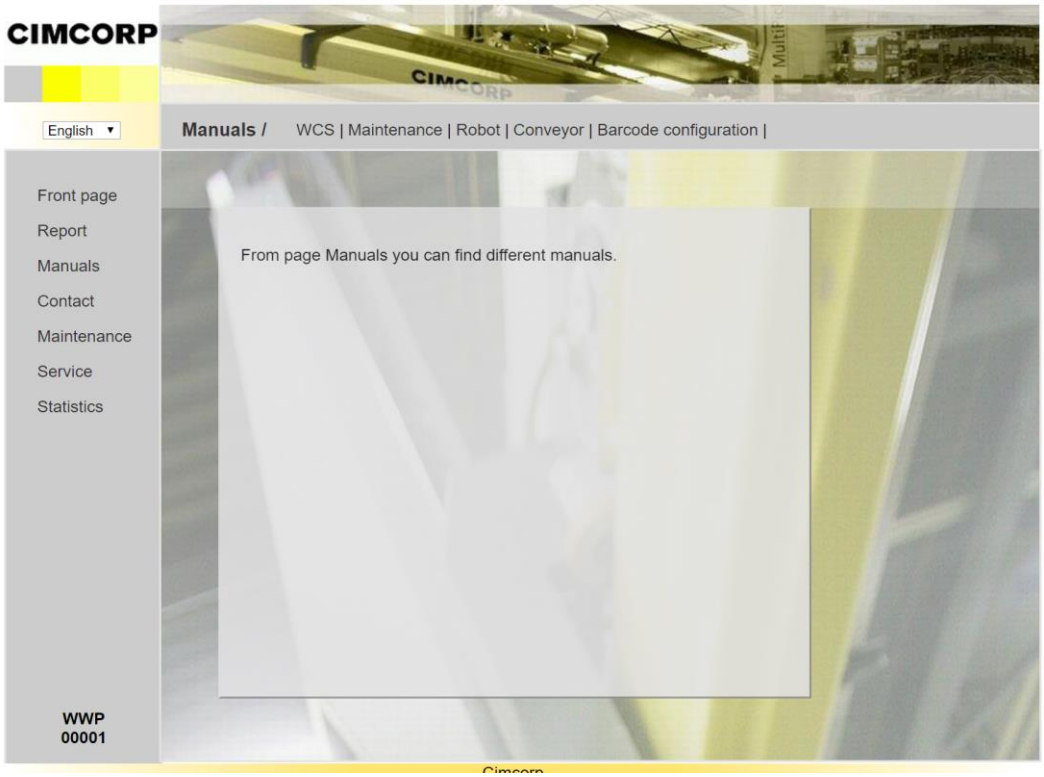


Kuva 11. Kuva suunnitelmasta.

Tässä opinnäytetyössä keskityn kertomaan lähinnä vain Front-End puolen toteutuksesta, konfiguroinnista ja muista siihen liittyvistä seikoista.

3.1 Vanha järjestelmä

Vanha järjestelmä oli aikanaan toteutettu Cimcorpille opinnäytetyönä ja siinä käytetty tekniikka oli vanhentunutta. Järjestelmä oli toteutettu PHP-ohjelmointikielellä ja tyylillisesti vanhentunut, eikä se ollut responsiivinen erilaisille laitteille, joten yritys päätti, että uusi järjestelmä tehdään täysin puhtaalta pöydältä. Vanhassa sivustossa oli ongelmana myös, että sen konfigurointi oli liian työlästä, minkä vuoksi uudessa järjestelmässä kiinnitettiin erityistä huomiota siihen, että asiat olisivat helposti määritettävissä.



Kuva 12. Vanhan järjestelmän ohjekirjat-sivu.

3.2 Projektin konfigurointi

Projektin aloittamiseen päätin käyttää Create React App:ia(CRA), joka yhdistää Webpackin, Babelin sekä muita käteviä sovelluksenhallinnan työkaluja. CRA mahdol-

listaa ohjelmoinnin nopean aloittamisen, sekä erittäin miellyttävän ja nopeasti nähtävän edistymisen. NPM-ohjelman npm start -komennolla voidaan ajaa kehitysserveriä samalla koneella käyttöliittymän kanssa ja näin pystyt näkemään tekemäsi muutokset muutamassa sekunnissa. Kaikki konfiguroinnit olisi myös voinut tehdä manuaalisesti, vaikka se olisi hieman monimutkaisempaa, mutta toisaalta tällä hetkellä projektin käyttöön purettu CRA on kaikista kätevin.

3.3 Komponenttien suunnittelu ja toteutus

3.3.1 Root

Ohjelman ylin taso on Root.js -niminen tiedosto, jossa sovellukseen haetaan Reduxin tarjoama tilavarasto tilanhallintaan sekä asetetaan joitakin tietoja, kuten versionumero ja sovelluksen nimi.

3.3.2 WwpApp

WwpApp-taso yhdistää sovelluksen ylimmän tason komponentit eli yläpalkin, sivupalkin, alapalkin sekä keskitason, joihin React router renderöi halutun komponentin. Halutessa tässä komponentissa voi myös määrittää onko kirjautuminen päällä. Sovellusta kehitettäessä on käytännöllistä pitää kirjautuminen pois päältä, jottei jokaisessa testaustilanteessa jouduta kirjoittamaan tunnuksia uudestaan.

```
export class WwpApp extends Component {
    render() {


        if (noAuth || this.props.token) {
            return (
                <div>
                    <TopBar />
                    <SideBar />
                    <div className="MiddleContent">
                        <NotificationContainer />
                        <WwpRouter />
                        <DevFooter />
                    </div>
                </div>
            );
        } else {
            return (
                <div>
                    <AuthForm />
                </div>
            );
        }
    }
};
```

Kuva 13. Sovelluksen ylimmän tason komponentit.

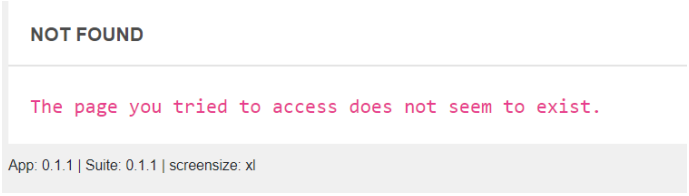
3.3.3 WwpRouter

Router-sivulla määritellään, mitkä komponentit muodostavat oman “sivunsa” ohjelmassa eli käytännössä React router piirtää sen komponentin, jonka osoite tai navigointivalikon kohta osoittaa. Koska sovellus on melko laaja, käytössä on myös Reactin mahdollistama ”laiska” lataus eli kyseinen komponentti ladataan vasta kun käyttäjä haluaa mennä siihen, eikä sitä ladata taustalla valmiiksi kuten yleensä. Periaatteessa tämä lisää hieman sovelluksessa näkyviä latausikoneja, mutta etenkin hitaammilla yhteyksillä käytettäessä myös nopeuttaa sovelluksen aukeamista.

```
<Router>
    <Redirect from="wwp" to="/" noThrow />
    <TilesDemo path="/" />
    {props.features.dashboard && <DashBoardPage path="dashboard" />}
    {props.features.downloads && <DownloadsPage path="downloads" />}
    {props.features.IT && <MaintenancePage path="IT" />}
    {props.features.manuals && <ManualsPage path="manuals" />}
    {props.features.tracking && <TracingRouter path="tracking/*" />}
    {props.features.reports && <ReportsRouter path="reports/*" />}
    <NotFoundPage default />
</Router>
```

Kuva 14. WwpRouterin Router -osio.

Kuvassa 14. näkyy että sovelluksessa oletuksena navigoidaan TilesDemo-nimiselle komponentille. Jos käyttäjä haluaa jollekin muulle sivulle, voi hän joko painaa navigointiin tarkoitettua tiiltä tai kirjoittaa halutun osoitteen selaimen osoiteriville. Jos haluttua sivua ei olekaan olemassa tai käyttäjä kirjoittaa väärin, näytetään NotFoundPage-komponentti.



Kuva 15. Reactin NotFoundPage-komponentti.

3.3.4 TopBar

TopBar eli yläpalkki, josta muodostui sekä tyylillisesti että datan suhteen melko haastava komponentti, koska siihen tehtiin hyvin monta toiminnallisuutta. Yläpalkin on oltava responsiivinen, jolloin pienillä näytöillä myös navigoinnille näytetään kuvake, kun taas suurella näyttökoolla riittää, että, kun taas suurella näyttökoolla riittää, että käytössä on vain kielen valinta sekä käyttäjähallinta.

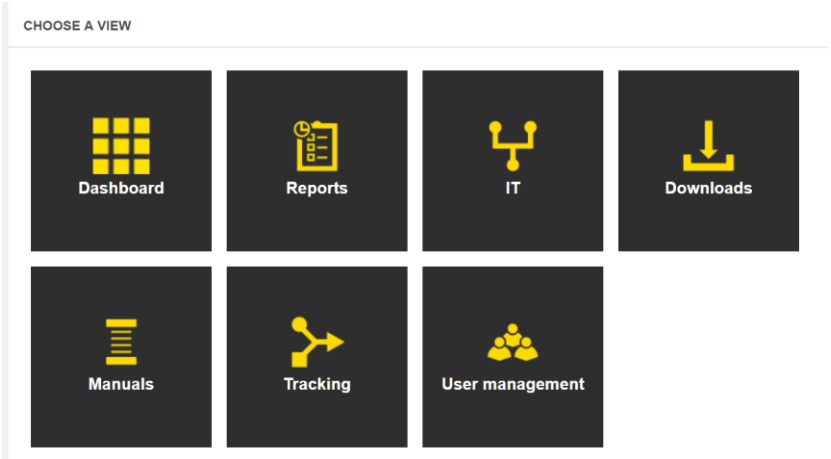
3.3.5 SideBar

Sidebar eli sivupalkkikomponentti toteutettiin ainoastaan navigointia varten. Se toteutettiin responsiivisesti, eli suurella näyttökoolla sivupalkki piirretään ja tätä pienemmillä piilotetaan käyttäen sen sijaan navigointiin joko tiiliä tai yläpalkkia. Sivupalkki

saa tiedon siitä, mitä sivustoja ohjelmistossa voi olla, datan hakemista varten toteutetulta FeatureList-komponentilta.

3.3.6 TilesDemo

TilesDemo-komponentti muodostaa sovelluksen näkyvän aloitussivun ja koostuu navigoinnin sovelluksessa mahdollistavista tiilistä. TilesDemo-komponentti käyttää samaa tietoa näytettävistä ominaisuuksista kuin edellä kuvatut sivupalkki ja yläpalkki.



Kuva 16. TilesDemo ja toteutetut tiilet.

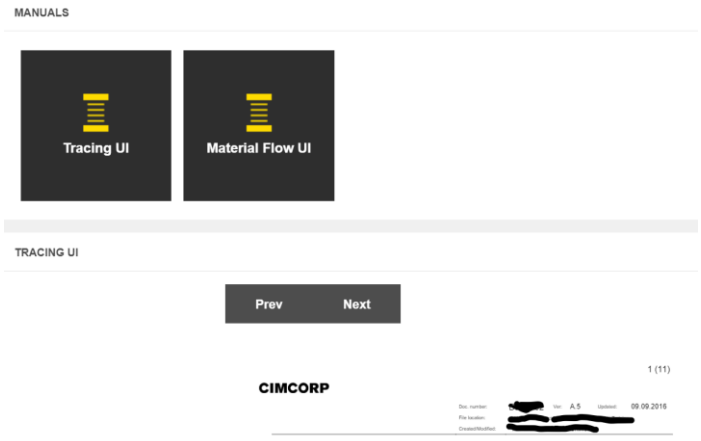
TilesDemo-komponentti lähettää proppina tiedot TileContainer-komponentille, joka kasaa yksittäisistä Tile-komponenteista TilesDemo:n näkymän lähettämällä kulloinkin tarvittavat tiedot propsina.

```
export const TileContainer = (props) => (
    <div className="TileContainer">
        {props.tiles.map((tile, index) => (
            <Tile
                bgImage={tile.icons.base}
                bgImageHover={tile.icons.hover}
                key={tile.ref}
                textContent={tile.text}
                link={tile.link}
                isExternal={tile.isExternal} />
        ))}
    </div>
);
```

Kuva 17. TileContainer-komponentti.

3.3.7 ManualsPage

ManualsPage-sivulla käyttäjä voi lukea laitteiden ohjekirjoja ja muita materiaaleja järjestelmästä. Sivu on toteutettu siten, että työpöydällä ei tarvita erillistä lukijaa pdf-tiedostoille vaan ohjelma kääntää sen luettavaksi selaimella. Tällä vältetään tarve erillisille mahdollisesti tietoturvaa vaarantaville ohjelmille. Mobiililaitteilla pdf-tiedostot avautuvat laitteen oletuslukijalla, sillä pdf-tiedostojen kääntäminen oli liian hidas ja raskas prosessi mobiilille.



Kuva 18. ManualPage, jossa auki yksi manuaali.

3.4 Redux-saga ja datan hakeminen palvelimelta

Koska ohjelmasta haluttiin mahdollisimman helposti muokattava, vältettiin kaiken mahdollisen datan ”kovakoodaamista” esimerkiksi JSON-tiedostoina sovellukseen. Sen sijaan käytössä on Redux-saga -kirjasto ja sen mahdollistama datan hakeminen asynkronisesti palvelimelta.

```
export function* featuresFetcher(action, webConfigApiGetter = getWebConfigApi) {
    const { features, cacheAge, fetched } = yield select(selectFeatureListFromStore);

    if (features.length === 0 || fetched + cacheAge <= Date.now()) {
        console.info('FeatureList not fetched yet or cached version is old');

        yield put(signalFeatureListLoading());

        if (action.appRef) {

            const featuresRequestFunc = (callback) => {
                webConfigApiGetter().getFeatures(action.appRef, callback);
            };

            const { error, data/*, response*/ } =
                yield call(craftResolvingPromisedApiRequestFunc(featuresRequestFunc));

            if (error) {
                if (isDevBuild() && !isTestEnv()) { // Simply isDevBuild() here does not suffice
                    console.warn(error.message);
                    console.warn('FeatureList fetch failed, will use preloaded feature set');
                    yield put(signalFeatureListLoading(false));
                } else {
                    yield put(setFeatureListError(error.message));
                }
            } else {
                yield put(setFeatureList(data));
            }
        } else {
            yield put(setFeatureListError('appref not set'));
        }
    } else {
        console.info('Fetch featureList from cache');
    }
}
```

Kuva 19. FeaturesFetcher Redux-saga -komponentti datan hakemiseen.

Aluksi määritetään mistä osoitteesta dataa koitetaan hakea mitä varten käytetään webconfigApiGetter funktiota. Sen jälkeen tehdään asynkroninen call kutsu palvelimelle ja riippuen palvelimen vastauksesta, joko ilmoitetaan tapahtuma virheelliseksi tai asetetaan saadut ominaisuudet Reduxiin.

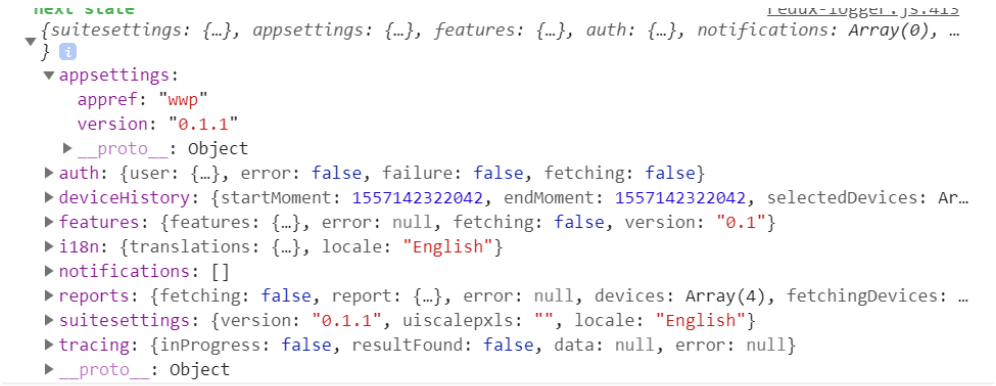
3.5 Datan hallinta Reduxilla

Riippuen siitä onnistuiko datan haku, Redux-kirjaston tarjoama reducer-komponentti palauttaa uuden tilan ja storeen asetetaan jokaisessa tapauksessa erilaiset tiedot (kuva 19.).

```
function featureList(state = defaultState, action) {
    switch (action.type) {
        case SET_FEATURE_LIST:
            const featuresToSet = action.featureList || defaultFeatures;
            return {
                ...state,
                fetching: false,
                features: sanitizeFeatureList(featuresToSet),
                error: null,
                fetched: Date.now()
            }
        case SET_FEATURE_LIST_ERROR:
            return {
                ...state,
                fetching: false,
                features: defaultFeatures,
                error: action.error,
                fetched: null
            }
        case FETCH_FEATURE_LIST:
            return {
                ...state,
                fetching: true,
                error: null
            }
        case SIGNAL_FEATURE_LIST_LOADING:
            return {
                ...state,
                fetching: action.isLoading || false,
                error: action.isLoading ? null : state.error
            }
        default:
            return state;
    }
}
```

Kuva 20. Reducer-komponentti.

Sovelluksen dynaamisuuden vuoksi on tärkeää, että palvelimelta saadaan oikeanlaiset tiedot tallennettavaksi Reduxin storeen, koska ilman onnistunutta hakua ei sovellus ole käytettävissä lainkaan.



Kuva 21. Redux store.

3.5.1 Lokalisointi

Koska sovellusta saatetaan käyttää missä suunnalla maailmaa tahansa, oli tärkeää myös saada lokalisointi helpoksi. Siinä apuna oli React-Redux-I18n -kirjasto.

Kirjaston avulla jokaisesta kielestä voitiin luoda oma helposti muokattava tiedostonsa. Sovellusta tehdessä kiinnitettiin myös huomiota lokalisointiin, sillä erilaiset tekstit ovat ohjelmassa muuttujina ja niiden teksti haetaan Reduxin kautta (kuva 21).

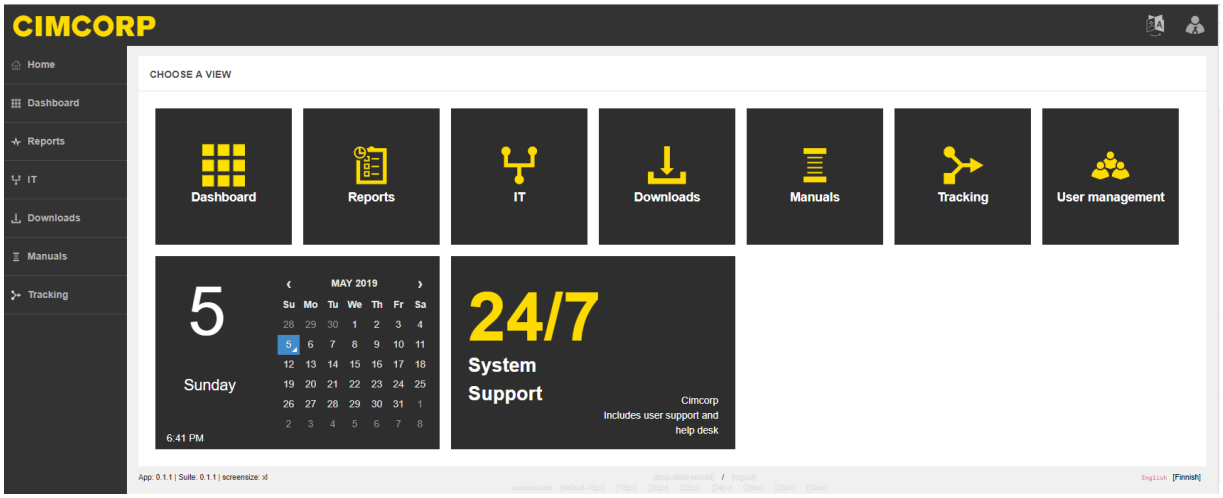
```
        <TitledContainer title={this.props.localization.titleName} collapsible={true}>
          {tiles}
        </TitledContainer>
      </div>
    );
  }


/ This is the Redux "container"
xport default connect(
 /* mapStateToProps */
 function (state) {
   let downloadsFromState = [];
   const downloadsFound = state.features.features;
   if (!isNullOrUndefined(downloadsFound)) {
     downloadsFromState = state.features.features.downloads.items;
   }
   const localization = {
     titleName: I18n.t('downloadsPage.title')
```

Kuva 22. Esimerkki otsikon lokalisoinnista.

4 YHTEENVETO

Kokonaisuutena projekti aikaisemmin tuntemattomilla web-tekniikoilla onnistui hyvin ja olen moniin ratkaisuihin tyytyväinen. Projektin edetessä, kun taidot kehittyivät, tuli huomattua useampiakin alussa tehtyjä virheitä, jotka työllistivät myöhemmin lisää. Projektin muiden osa-alueiden kehitys jatkuu vielä, mutta ne olivat tämän opinnäytetyön rajauksen ulkopuolella.

Sovelluksen ulkoasu työpöydällä ja mobiilissa vastaa haluttua. Projektissa käytetyistä tekniikoista, sekä yrityksen valitsemista että itse päättämistäni, jäi positiivinen kuva, enkä edes jälkikäteen lähtisi vaihtamaan käytettyjä tekniikoita. Ohjelma kirjoitettiin JavaScriptillä, mutta projektin edetessä tuli myös TypeScript-tuki mahdolliseksi lisätä CRA:lla tehtyyn sovellukseen. Olen tyytyväinen kokonaisuuteen ja aionkin jatkaa työssä käytetyillä web-tekniikoilla ohjelmointia tulevaisuudessa.



Kuva 23. Valmiin sovelluksen etusivu.

LÄHTEET

Babel kotisivut. 2018. Viitattu 30.9.2018. https://babeljs.io/docs/en/

Cimcorp intranet. 2019. Viitattu 6.5.2019. Vain yrityksen työntekijöille.

Flux-dokumentit. 2018. Viitattu 29.9.2018. https://facebook.github.io/flux/docs/in-depth-overview.html#content

JavaScript info. 2019. Viitattu 18.4.2019. https://JavaScript.info/intro

Jest kotisivu. 2019. Viitattu 4.5.2019. https://jestjs.io/docs/en/getting-started

Medium. 2018. Viitattu 25.8.2018. https://medium.com/@dan_abramov/smart-and-dumb-components-7ca2f9a7c7d0

MDN web docs. 2019. Viitattu 9.3.2019. https://developer.mozilla.org/en-US/docs/Web/HTML

MDN network. 2019. Viitattu 9.5.2019. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/find

ReactJS kotisivu. 2018. Viitattu 25.8.2018. https://reactjs.org/

ReactJS kotisivu JSX. 2018. Viitattu 25.8.2018. https://reactjs.org/docs/introducing-jsx.html

React router Kotisivut. 2019. Viitattu 28.4.2019. https://reacttraining.com/react-router/web/example/basic

Redux kotisivut. 2018. Viitattu 29.9.2018. https://Redux.js.org/introduction/threeprinciples

Redux-saga kotisivut. 2019. Viitattu 4.5.2019 https://Redux-saga.js.or

Webbipakki css intro. 2019. Viitattu 9.3.2019. http://www.weppipakki.com/css/tekstit/cssintro.html

Webpack. 2019. Viitattu 4.5.2019. https://github.com/webpack/webpack

W3Schools. 2019. Viitattu 18.4.2019. https://www.w3schools.com/whatis/whatis_npm.asp