Tampereen ammattikorkeakoulu

Käyttäjädatan kerääminen SPA-arkkitehtuurissa

Case AREA 21

Tomi Heino

OPINNÄYTETYÖ
Marraskuu 2020

Tieto- ja viestintätekniikan tutkinto-ohjelma
Ohjelmistotekniikan opintosuunta

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikan tutkinto-ohjelma
Ohjelmistotekniikan opintosuunta

HEINO, TOMI:
Käyttäjädatan kerääminen SPA-arkkitehtuurissa
Case AREA 21

Opinnäytetyö 38 sivua
Marraskuu 2020

Tässä työssä kerrotaan SPA-arkkitehtuurin toiminnasta ja ominaisuuksista, keskittyen erityisesti React-sovelluksiin sekä käyttäjäanalytiikan toteuttamiseen kyseisessä arkkitehtuurissa. Työssä toteutettiin AREA 21 -projektissa rakennettuun Energy App -sovellukseen TypeScript- ja GraphQL-kielien avulla tietojen hallintatoiminnot, datan visualisointia sekä palautetoiminto. Sovellus julkaistiin Progressive Web App -sovelluksena, jonka toimintaa Safari-selaimilla korjattiin.

Sovellukseen toteutettiin automaattinen pageview-tapahtumien lähetys Google Analytics -palveluun JavaScriptin avulla. Datan keräyksen toiminta varmistettiin palvelusta sovelluksen julkaisun jälkeen. Lopuksi kerrotaan jatkokehitystarpeista ja mahdollisuuksista.

Asiasanat: yhden sivun sovellus, web-analytiikka, react, evästeet

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Software Engineering

TOMI HEINO:
Collection of user data in SPA architecture
Case AREA 21

Bachelor's thesis 38 pages
November 2020

This work describes the functionality and properties of web applications implemented with Single Page Application architecture, with focus on React and implementation of user analytics. The AREA 21 project produced Energy App -web application, to which was implemented additional functionality regarding data management, data visualization and feedback with the use of TypeScript and GraphQL-languages. Web application was published as a Progressive Web Application, which had bugs on Safari browsers that were fixed.

An automatic pageview-event sending for Google Analytics was implemented with the use of JavaScript. Data collection was confirmed after the launch of the web application. Finally, there are requirements and suggestions for further development.

Key words: single page application, web analytics, react, cookies

SISÄLLYS

LYHENTEET JA TERMIT

CSR	Client Side Rendering, yhden sivun sovelluksen renderöintitapa
CSS	Cascading Style Sheets, verkkosivujen esitystapakieli
CSV	Comma Separated Values, datansiirtoformaatti
DOM	Document Object Model, selaimen tapa ymmärtää verkkosivu
HTML	Hypertext Markup Language, verkkosivujen kehityskieli
HTTP	Hypertext Transfer Protocol, verkkosivujen ja datan siirtoon selainten protokolla
REST	Representational State Transfer, HTTP-protokollaan perustuva arkkitehtuurimalli
SPA	Single Page Application, yhden sivun sovellus
SSG	Static Site Generation, sivu luodaan etukäteen sisällön muuttessa, jaetaan staattisena
SSR	Server Side Rendering, verkkosivun renderöinti web-palvelimella
UUID	Universally Unique Identifier, globaalisti uniikki tunniste
URL	Uniform Resource Locator, web-osoite

1 JOHDANTO

Tässä työssä tehtiin AREA 21-websovelluksen puuttuvat omaisuudet ja tarvittavat tutkimukset ja toimenpiteet, jotta automatisoitu käyttäjäanalytiikka saataisiin toteutettua Google Analytics-palvelun avulla.

Toisessa luvussa kerrotaan SPA-arkkitehtuurilla toteutetun web-sovelluksen toiminnasta, mitä etuja tai haittoja sillä on ja miten projektin sovelluksessa käytetty React-sovelluskehys toimii. Kolmannessa käydään läpi yleisesti selainpuolen tapoja seurata käyttäjän tilaa sivulatausten yli ja keskitytään huomioonotettaviin asioihin ja haasteisiin, kun halutaan seurata käyttäjän toimintaa web-sovelluksessa.

Area21-projektissa on toteutettu web-sovellus, jolla esitetään antureista kerättyä mittausdataa. Luvussa neljä kerrotaan sovelluksessa käytetyistä teknologioista ja selainpuoleen toteutetuista React-komponenteista ja API-operaatiosta sovelluksen datan hallintaa varten. Osana tätä on Google Analytics palvelu, jolla voidaan tutkia sovelluksen käyttäjämääriä ja toimintaa sovelluksen sisällä.

Toteutettujen ominaisuuksien jälkeen käydään läpi mahdollisia jatkokehitysideoita luvussa 5 – tärkeimpänä niistä muuttuneen lainsäädännön ja sen tulkinnan perusteella vaadittavat muutokset datan keräykseen.

Lopuksi pohditaan, millä tavoin ja mitä opittiin React-sovelluksen toimintojen täydentämisestä julkaisukuntoon ja mitä voitaisiin tulevaisuudessa tehdä toisin paremman lopputuloksen varmistamiseksi.

2 SINGLE PAGE APPLICATION-ARKKITEHTUURI

2.1 Yhden sivun sovellukset

Perinteisesti websivuston osat toimitetaan HTTP-palvelimelta valmiina HTML-sivuina, jotka renderöidään käyttäjän Internet-selaimessa. Tällaisessa arkkitehtuurissa siirretyn datan määrä on suoraan verrannollinen sivun pituuteen ja sen sisältämän median määrään. Sivu voidaan myös luoda dynaamisesti web-palvelimella sovelluskehyksen avulla tai sivua voidaan täydentää AJAX-tekniikoilla latauksen jälkeen. (SPA (Single-page application) - MDN Web Docs Glossary: Definitions of Web-related terms | MDN)

Modernimmassa web-sovelluskehityksessä sivuston luomiseen käyttäjälle on kehitetty muutamia eri tekniikoita, joilla voidaan optimoida verkossa siirrettävän tiedon määrää ja samalla sivuston ensilatauksen nopeutta tai web-sovelluspalvelimen kuormitusta. Kuormituksen määrää ei voida kuitenkaan suoraan yhdistää sivun luontitapaan, sillä välimuistin tai sisällönvälitysverkkojen käytöllä on myös huomattava vaikutus.

SPA-arkkitehtuurissa käyttäjän Internet-selaimelle toimitetaan kaikki sen tarvitseva HTML, JavaScript ja CSS-koodi ensimmäisellä sivulatauksella. HTML-koodilla kuvataan sivun pohja, johon JavaScriptillä ja SPA-sovelluskehyksellä toteutetaan web-sovelluksen logiikka. CSS-tyylikielellä määritellään sovelluksen ulkoasu. Äärimmäisessä tapauksessa tämä saattaa tarkoittaa kaiken web-sovelluksen tarvitseman JavaScript-- ja CSS-koodin toimittamista ensimmäisellä sivulatauksella, mukaan lukien SPA-sovelluskehyksen omien koodikirjastojen, voi isossa web-sovelluksessa ensilataus hidastua huomattavasti, joka vaikuttaa käyttäjän kokemukseen. (Single-page applications vs. multiple-page applications: pros, cons, pitfalls; How Page Load Time Affects Bounce Rate and Page Views)

Ensilatausta voidaan yrittää keventää lataamalla sivustolla olevia skriptejä tai osia sivustosta vasta kun niitä tarvitaan käyttäjän siirtyessä sivustolla niiden koh-

dalle. Latausnopeutta voidaan myös parantaa tehostamalla sivuston osien latausta mahdollisimman läheltä käyttäjän selainta käyttämällä hyödyksi selainten sisäistä välimuistia ja Internetin sisällönvälitysverkkoja. (Effect of first load)

SPA-sovelluksen edut saavutetaankin ensilatauksen jälkeen, jolloin kaikki käyttäjän seuraavat navigaatiot vaativat vain mahdollisesti uuden datan hakemista web-palvelimelta esimerkiksi REST- tai GraphQL-rajapinnan kautta. URL-osoitteen muuttuminen ei aiheuta uuden sivun renderöintiä, vaan sovelluksen tila päivitetään vastaamaan haluttua.

Arkkitehtuurin ongelmia ovat muun muassa hakukoneiden tai muiden sivuja ohjelmallisesti lukevien palvelujen vaikeudet sivuston sisällön ymmärtämisessä. Tällaiset palvelut ovat kiinnostuneita sivun metatiedoista ja sivun itse sisällöstä. Koska SPA-arkkitehtuurissa sivun varsinainen, esitettävä loppumuoto (DOM) luodaan suoritettavalla JavaScriptillä ensilatauksen jälkeen, ei pelkkä URL-osoitteen perusteella haettava sisältö edusta lopullista verkkosivua. Kuitenkin esimerkiksi Google renderöi ja käy läpi myös SPA-sovellusten sivujen URL-osoitteet, mutta tekee sen vasta myöhemmin erillisenä operaationa. (Understanding web pages better, 2014)

Ongelmaa voidaan ratkaista ottamalla käyttöön hakukoneystävälliset URL-osoitteet, sivun <head>-tagien hallinta ja Server Side Rendering. Hakukoneystävälliset osoitteet saavat sovelluksen muuttamaan sisäisen tilansa osoitteen mukaiseksi automaattisesti ja renderöimään odotetun sisällön käyttäjälle. Hallitsemalla sovelluksen head-tageja osana sovelluksen tilaa, saadaan ne edustamaan sovelluksen todellista tilaa. SSR-tekniikkaa käytettäessä ensilatauksella pyydetty sivu luodaan palvelimella ja se välitetään käyttäjälle muun sovelluskoodin mukana, joka jatkaa sivuston suorittamista CSR-sovelluksena.

Taulukkoon 1 on koottu muutamia erilaisia tapoja web-sivun luomiseen ja toimittamiseen sekä pääpiirteittäin heikkouksia ja hyviä puolia, joita on pyritty korostamaan – jokainen web-sivusto vaatii oman tarkastelun.

TAULUKKO 1. Eri tapoja luoda ja toimittaa web-sivu.

Tekniikka		Etuja	Haittoja
Staattinen HTML-sivu	Sivut jaetaan HTTP-palvelimilla	Yksinkertainen	Interaktiivisuuden puute
Static Site Generation	Sivut luodaan sisällöstä prosessin avulla	Sisältö voi muuttua	Vaatii sivuston luonnin uudelleen, kun sisältö muuttuu
Palvelimella luotu sivu	Sivu luodaan palvelimella joka kerta	Sisältöä voidaan hallita latauskohtaisesti	Ilman välimuistia kuormittaa palvelinta
Single Page Application	Sivusto toimitetaan selaimelle, sisältö haetaan erikseen	Ensilatausta seuraavat navigaatiot kevyitä	Ensilataus on raskas Hakukoneet eivät voi indeksoida sivua tehokkaasti
Single Page Application, Server-Side Rendering	Ensilataus luodaan palvelimella, selainpuolen koodi ottaa haltuun	Samat kuin SPA, mutta sivuston on indeksoitavissa, toimii myös osittain ilman JavaScriptiä	Ensilataus on raskas

2.2 React-sovelluskehys

React on Facebookin kehittämä deklaratiivinen sovelluskehys käyttöliittymien toteuttamiseen Web-ympäristöissä. Se perustuu komponentteihin, joiden päivitystä hallitaan niiden saaman tiedon ja tilan perusteella ja joita yhdistämällä voidaan rakentaa interaktiivisia käyttöliittymiä. Komponenttien logiikka ohjelmoidaan JavaScriptillä ja React huolehtii niiden esittämisestä haluttussa ympäristössä. (Why did we build React?, 2013)

Deklaratiivisessa paradigmassa sovellusten funktiot ohjelmoidaan tilattomiksi, eli ne palauttavat aina samalla syötteellä samat arvot. Tämä toimintatapa korostuu tavassa hallita sovelluksen tilaa React-sovelluksissa ja sovelluksen tilan muutos aiheuttaa komponenttien päivittymisen.

React-paradigman käyttö on laajentunut websovellusten ulkopuolelle. React Native -kirjaston avulla on mahdollista kehittää sovelluksia mobiililaitteisiin käyttämällä samoja tekniikoita kuin websovelluksien kehittämiseen. Mobiilisovellusten tapauksessa sovelluksen komponentin kirjoitetaan samantyylisesti kuin web-sivulla, mutta taustalla luodaan HTML-sivun sijaan mobiilisovelluksen käyttöliittymä. (React Native · A framework for building native apps using React)

2.2.1 Rakenne ja komponentit

React-sovelluskehyksen avulla luotava käyttöliittymä kuvataan React-komponentteina, jotka ovat taustalla JavaScript-objekteja. JavaScript-objektien luominen on päätelaitteen resurssien käytön osalta huomattavasti halvempaa kuin selaimen DOM-elementtien ja tällä on positiivinen vaikutus suorituskykyyn. (Rendering Elements – React)

Komponenttien kuvaamiseen käytetään erityistä JSX-syntaksia, joka ei ole suoraan JavaScriptiä tai HTML-kieltä, vaan laajennos JavaScriptiin. Syntaksi pyrkii yhdistämään käyttöliittymässä olevan komponentin esitystavan ja sen toiminnot yhdeksi loogiseksi yksiköksi sovelluksessa. (Introducing JSX – React)

Kuviossa 1 on koodiesimerkki JSX-syntaksilla kirjoitetusta elementistä, joka renderöitynä tulostaisi HTML-dokumenttiin otsikkoelementin sisällöllä ”Hello, world”. Itsessään tällaisen toiminnan päälle voitaisiin rakentaa tapa renderöidä staattista sisältöä ja siksi React tarjoaa komponenteille props-syötteen. (Components and Props – React)

```
1   const element = <h1>Hello, world</h1>;
2
```

KUVIO 1. Yksinkertainen JSX-elementti.

Komponentit eroavat pelkistä JSX-syntaksilla kirjoitetuista elementeistä tuomalla mukaan komponentin sisällön mukauttamisen props-syötteen avulla. Tällöin renderöitävä sisältö voidaan palauttaa JavaScript-funktion tuloksena tai ECMAScript 2015 -luokkasyntaksin avulla, joista molemmilla on kirjoitettu sama toiminnallisuus kuvion 2 kahdessa ylimmässä koodiesimerkissä. Kummastakin versiosta voidaan myös nähdä, miten JavaScriptillä luodun sisällön upottaminen JSX-elementin sisälle onnistuu, tässä tapauksessa syötteenä tuleva name-muuttuja. (Components and Props – React)

Kuvion 2 alimmassa koodiesimerkissä käytetään ylempänä määriteltyä komponenttia Hello. Komponentille annetaan sen tarvitsema syöte, jonka toimittamisesta sovelluskehys huolehtii itsenäisesti.

```
1   function Hello(props) {
2     return <h1>Hello, {props.name}</h1>;
3   }
4
5   class Hello extends React.Component {
6     render() {
7       return <h1>Hello, {this.props.name}</h1>;
8     }
9   }
10
11  const element = (
12    <div>
13      <Hello name="world" />
14    </div>
15  );
```

KUVIO 2. JSX-komponentti kahdella eri syntaksilla ja käyttöesimerkki.

2.2.2 Renderöinti ja tilanhallinta

JSX-elementtien muodostama puu ei sellaisenaan käy renderöitäväksi, vaan sen syntaksi on muunnettava Babel-kirjaston avulla tavalliseksi JavaScript-koodiksi. Muunnetun koodin renderöinti web-ympäristössä tehdään ReactDOM-kirjastolla. ReactDOM ottaa toiseksi parametriksi HTML-dokumentista löytyvän

DOM-elementin, johon se liittää renderöitävän sovelluksen juurielementin. (ReactDOM – React)

Muuttuva data tai tila aiheuttaa komponenttien uudelleenrenderöinnin tarpeen mukaan – tätä kutsutaan reconciliation-vaiheeksi. React tarjoaa sen käyttäjälle deklaratiiviseen rajapinnan ja huolehtii sisäisesti algoritminsa avulla, mitkä osat sovelluksesta on renderöitävä uudelleen. (Reconciliation – React)

Muuttuva tila onkin siis Reactissa interaktiivisuuden perusta. Käyttäjän toimet saavat aikaan muutoksia tilassa, jotka johtavat komponenttien uudelleenrenderöintiin. Tilaan voidaan varastoida JavaScript-tietotyyppejä ja se on muuttumaton, eli aikaisempi tila pitää aina kirjoittaa yli uudella, jotta React pystyy tunnistamaan muutokset. (State and Lifecycle – React)

Tilanhallinta on aikaisemmin Reactissa tehty luokkakomponenttien setState-metodin avulla, mutta versiosta 16.8 lähtien kehittäjät ovat suositelleet siirtymistä funktiokomponentteihin ja hook-rakenteiden käyttöä. Hook-rakenteet mahdollistavat helpomman toimintojen uudelleenkäytön komponenttien välillä. (Introducing Hooks – React)

3 KÄYTTÄJÄANALYTIIKKA

3.1 Analytiikan tarpeet

Data, jota kerätään tuotteen käytöstä, on vastapaino intuition luomille mielikuville asioista. Datalla siten voidaan saada kvantitatiivista ymmärrystä asioista, joita tuotteen tekijä tai omistaja ei välttämättä pystyisi ymmärtämään, mutta käyttäjät oppivat nopeasti. Dataan perustuvalla ymmärryksellä ohjattu oppiminen auttaa testaamaan rakennettua, mittamaan sen vaikutusta ja rakentamaan jotain parempaa. Tällainen iteroiva tuotekehityssykli on nopea ja helpottaa resurssien kohdentamista toimiviin ratkaisuihin. (Lean analytics, kappale 1, 2).

3.2 Metriikoiden valinta

Analytiikassa seurattavaa asiaa kutsutaan metriikaksi. A Review of Methodologies for Analyzing Websites antaa esimerkkeinä erilaisista metriikoista seuraavat:

- Sivuston käyttö
    - Vierailijoiden määrä
    - Istuntojen määrä
    - Palaavien käyttäjien määrä
    - Käyttäjien fyysinen sijainti
    - Hakukoneaktiivisuus
- Viittaukset
    - Mistä kävijät tulevat
    - Kävijöiden käyttämät hakusanat
    - Kuinka monet käyttäjät lisäävät sivustolle kirjanmerkin
- Sivuston sisällön analyysi
    - Suosituimmat sisääntulosivut
    - Suosituimmat sivut
    - Suosituimmat sivut yhden sivun istunnoissa
    - Suosituimmat lähtösivut
    - Suosituimmat reitit sivuston läpi

    - Avainsisällön toimivuus
- Laadunvarmistus
    - Rikkinäiset sivut
    - Palvelimen virhetilat
    - Kävijöiden vaste virheisiin

Metriikka on syytä valita tuotteen tarpeiden mukaan, mutta kaikkien metriikoiden paremmuutta voidaan arvioida Lean Analyticsin mukaan seuraavasti:

- **Vertailtavuus:** tuloksia voidaan vertailla eri ajanjaksoihin, eri käyttäjiin tai kilpailijoihin, jotta voidaan ymmärtää asioiden suunta
- **Ymmärrettävyys:** tuloksista on voitava keskustella ja ne jäävät ihmisten mieleen
- **Metriikka on asioiden suhde tai määrä:**
    - suhteessa oleviin asioihin on helpompi reagoida
    - suhteet ovat vertailtavia – voidaan erottaa lyhyen aikavälin piikit pitkän aikavälin trendeistä
    - suhteilla on mahdollista vertailla tekijöitä, jotka normaalisti olisivat vastakkain

Analytiikassa etsitään mittaustulosta, jonka perusteella voidaan tehdä luotettavia päätöksiä. Kvantitatiivisella, lukuihin perustuvalla, metriikalla on mahdollista todeta tuotteen tilan kehittymisen suunta, onko se haluttu vai ei. Kvalitatiivisella, laatua tutkivalla, metriikalla voidaan testata ominaisuuden onnistumista – mikäli kukaan ei käytä sitä, on resurssien käyttöä siihen harkittava tarkasti. (Lean analytics, kappale 2).

Mitattavat asiat on siis syytä valita tarpeen mukaan – sisältösivulla arvokas tieto saattaa olla, mikä on sivuston suosituin sivu, jolloin kehitystoimenpiteet voidaan keskittää sen parantamiseen.

3.3 Datan keräysprosessi

Taulukossa 2 on esitelty A review of Methodologies for Analyzing Websites -teoksen jaottelu taustajärjestelmästä tehtävän analytiikan ja selainpuolen skriptien avulla tehtävän mittauksen välille ja miten etuja tai haittoja kummallakin on. Mikäli tavoitteeksi otetaan mahdollisimman tarkka ymmärrys käyttäjien toiminnasta web-sivustolla, voidaan pelkästään lokitiedostoihin perustuvat ratkaisut sivuuttaa kokonaan. Palvelimella saadaan SPA-arkkitehtuurissa tietoa käyttäjän vuorovaikutuksesta sovelluksen kanssa vain, jos käyttäjä aiheuttaa uuden datan lataamisen palvelimelta.

TAULUKKO 2. Logitiedostojen ja sivunvalvonnan edut ja haitat (A review of Methodologies for Analyzing Websites).

Lokitiedostot		**Sivunvalvonta**	
Edut	Haitat	Edut	Haitat
Ei vaadi muutoksia sivustoon	Voi tallentaa vain palvelimella tapahtuvat interaktiot	Lähes reaaliaikainen raportointi	Vaatii muutoksia etupäähän
Ei vaadi ylimääräistä resursseja	Palvelimen on asetettava käyttäjille eväste	Helppo muuttaa tallennettavaa tietoa	Käyttää resursseja sivulatauksen aikana
Vapaus muuttaa työkaluja helposti	Saatavilla vain, mikäli ylläpitäjä hallitsee palvelinta	Mahdollisuus tallentaa monimutkaista tietoa JavaScriptin avulla	Voi tallentaa tietoa vain onnistuneista sivulatauksista
Tallentaa tietoa myös epäonnistuneista sivulatauksista	Ei voi tallentaa käyttäjän fyysistä sijaintia		Sitoutuminen tiettyyn analytiikkakehykseen

Loppuun analytiikkasuunnitteluun voidaankin ottaa tavoitteeksi kaiken mittaamisen tekeminen selainpuolella JavaScriptin avulla, koska vain sen avulla voidaan saada tarkkaa tietoa käyttäjien vuorovaikutuksesta sivuston kanssa.

3.4 Selaintekniikat

Tietoa käyttäjän toimista web-sovelluksessa voidaan tallentaa muutamilla eri tavoilla, jotka säilyvät sivun latauskerran elinkaaren yli. Perinteisin tapa on käyttää evästeitä (cookies): tekstimuotoisia avain-arvo-pareja, joiden sisältö on liitetty web-sivuston verkkotunnukseen. Evästeitä voidaan asettaa sivustoa jakelevan web-palvelimen toimesta HTTP-protokollan Set-Cookie -otsikkotiedon avulla tai JavaScriptin avulla käyttämällä document.cookie -komennon avulla. (Using HTTP cookies - HTTP | MDN)

Sivulle asetetut evästeet voidaan toimittaa takaisin palvelimelle seuraavien HTTP-kutsujen mukana – tämä mahdollistaa sen, että sama selain voidaan tunnistaa eri sivulatausten välillä. Evästeet jakautuvat istuntoevästeisiin ja pysyviin evästeisiin riippuen siitä, onko niille asetettu voimassaoloaikaa (Expires). Mikäli voimassaoloaikaa ei ole asetettu, selain käsittelee evästettä istuntoevästeenä ja poistaa sen, kun käyttäjä sulkee web-selaimensa tai kaikki verkkotunnukseen liittyvät avoimet sivut. (Using HTTP cookies - HTTP | MDN)

Mahdollisuuksia lukea ja vaikuttaa evästeisiin hallinnoidaan eri lippujen avulla, jotka evästeelle voidaan asettaa ja täten vaikuttavat sivuilla suorittavan JavaScriptin pääsyyn niihin. Selainten tukemia evästelippuja on esitelty taulukossa 3.

TAULUKKO 3. Eri lippuja evästeen käyttöoikeushallintaan.

Lippu	Vaikutus
HttpOnly	Selaimessa suoritettava JavaScript ei voi lukea tai vaikuttaa evästeeseen. Muokkaus- ja lukeminen mahdollista vain web-palvelimella, joka asetti evästeen.
Secure	Eväste toimitetaan ja sitä voidaan lukea ja käsitellä vain, jos yhteys on muodostettu suojatulla HTTPS-protokollalla
SameSite	Hallitsee evästeen toimitusta, kun selain navigoi verkkotunnukselta toiselle (cross-domain). Kolme mahdollista arvoa: • Strict – eväste toimitetaan vain verkkotunnukselle, joka asetti sen • Lax – evästä toimitetaan, mikäli käyttäjä navigoi esimerkiksi käyttämällä linkkiä • None – eväste toimitetaan aina

Toinen tapa jakaa evästeet riippuu siitä, mistä kontekstista niitä yritetään lukea tai asettaa. Web-sovelluksen käsitellessä omaan verkkotunnukseensa asetettuja evästeitä, kutsutaan niitä ensimmäisen osapuolen evästeiksi (first party cookies). Tässä tapauksessa sivustolla on täysi oikeus, aiemmin kuvattujen sääntöjen puitteissa, lukea ja asettaa evästeitä omaan tunnukseensa. (Using HTTP cookies - HTTP | MDN)

Mikäli sivusto tekee HTTP-kutsuja muihin verkkotunnuksiin, jotka asettavat evästeitä, kutsutaan näitä kolmannen osapuolen evästeiksi (third party cookies). Tähän vaikuttaakin erityisesti aiemmin mainittu SameSite-lippu. Kolmannen osapuolen evästeet sopivatkin erinomaisesti käyttäjän toiminnan seuraamiseen hänen siirtyessään verkkotunnuksien yli ja juurikin tämän yksityisyydensuoja-uhkan ansiosta osa selainvalmistajista ovat jo estäneet pääsyn kolmannen osapuolen evästeisiin tai aikovat tehdä niin lähitulevaisuudessa. (Using HTTP cookies - HTTP | MDN)

Myös ensimmäisen osapuolen evästeet saatetaan poistaa selaimen muistista aikaisemmin, kuin Expires-kenttä määrää. Näin toimii esimerkiksi Applen Safari-

selain, joka poistaa JavaScriptin document-cookie-kutsun avulla asetetut evästeet 24 tunnin kuluttua, mikäli käyttäjä ei vieraile sivustolla uudelleen samassa aikaikkunassa. (Using HTTP cookies - HTTP | MDN)

Tietoa voidaan myös tallentaa moderneimmilla tavoilla, kuten selaimen localStorageen, joka on pitkälti samoin kuin evästeet, avain-arvo-pareihin perustuva tietovarasto, mutta sitä ei lähetetä takaisin http-palvelimelle ilman lisäratkaisuja. Selainten sessionStorage vastaa istuntoevästettä, koska myös se poistetaan käyttäjän lopetettua istuntonsa. (Using HTTP cookies - HTTP | MDN)

3.5 Tietosuoja

3.5.1 Selainvalmistajien toimenpiteet

Koska evästeet ja muut mahdolliset käyttäjän seurannan mahdollistavat tekniikat suoritetaan selaimessa, on osa selainvalmistajista ottanut proaktiivisen otteen käyttäjien yksityisyyden suojaamisessa. Applen Safari on versiosta 11 (2017) lähtien sisältänyt toimintoja, jotka vaikeuttavat käyttäjän toiminnan seuraamista verkkotunnusten yli. Versiossa 14 Safari antaa käyttäjälle yksityisyysraportin, jossa on listattuna sen tunnistamat ja estämät seurantatavat. (Apple, Safari Privacy Overview, 2019; WebKit Blog, Full Third-Party Cookie Blocking and More, 2020)

Samalle linjalle ovat lähteneet Mozilla Firefox ja Microsoftin Chromium-pohjainen Edge. Kumpikin selain estää tunnettujen seurantatekniikoiden ja verkkotunnusten toiminnan sekä kertoo käyttäjälle listan niistä. (Tracking Prevention in Microsoft Edge (Chromium) - Microsoft Edge Development | Microsoft Docs, 2020; Firefox Now Available…, 2019)

3.5.2 Suostumus evästeisiin

Yleisestä ohjeistusta evästeiden tai käyttötietojen tallentamisesta ja käytöstä saadaan Kyberturvallisuuskeskuksen Luottamuksellinen viestintä (2020) -sivulta, joka opastaa, että sivuston käyttäjille pitää kertoa

- evästeiden ja käyttötietojen tallentamisesta ja niiden käyttötarkoituksesta on kerrottava selkeästi ja kattavasti
- tiedot evästeen toiminta-ajasta ja kolmansien osapuolien mahdollisuus käyttää evästettä

Käyttäjältä on saatava suostumus evästeiden avulla kerättävien tietojen tallentamiseen ja käyttöön, mutta myös kieltäytyminen on tehtävä käyttäjälle vaivattomaksi. (Luottamuksellinen viestintä, 2020).

Käyttäjäanalytiikassa tarvittavien evästeiden käyttöön saadaan lisätietoa Liikenne- ja viestintäviraston huhtikuussa 2020 antamasta päätöksestä evästeiden käytöstä innowise.fi-sivustolla. Päätöksessä on käsitelty kyseisellä sivustolla käytettyjä evästeitä ja niiden tietosuojasääntelyn suhdetta niihin, joista tärkeimpänä nousee esiin Google Analyticsin käyttöön liittyvä kohta:

> Saadun selvityksen perusteella Innowise-sivustolla käytössä olevat tilastointi- ja analytiikkaevästeet, eli Google Analytics -kävijäseurantapalvelun eväste ja Facebook pixel -seurantaeväste, eivät kuulu välttämättömiin evästeisiin, jolloin niistä tulee antaa käyttäjälle selkeät tiedot ja pyytää käyttäjän suostumus. (Liikenne- ja viestintäviraston päätös käyttäjän suostumuksen antamisesta evästeiden käyttämiseksi innowise.fi-sivustolla 2020, 3.)

Tilastointi- ja analytiikkaevästeiden käytöstä pitää kysyä käyttäjältä erikseen lupa eikä pelkkä selaimen salliva evästeasetus riitä suostumukseksi. Google Analytics-palvelun asettama eväste on myös sellainen, jota kolmas osapuoli (Google) voi käyttää ja tällainen eväste ei silloin ole välttämätön palvelun tarjoamiseksi. (Liikenne- ja viestintäviraston päätös käyttäjän suostumuksen antamisesta evästeiden käyttämiseksi innowise.fi-sivustolla 2020, 4.)

Päätöksessä otetaan myös kantaa suostumuksen kysymiseen palvelun kannalta välttämättömien evästeiden tallentamiseen, joihin riittää selaimen asetusten antama lupa. AREA 21-sovelluksen kehyksessä tarkoittaa istuntoevästettä, jolla käyttäjän kirjautuminen tunnistetaan ja on siten välttämätön palvelun tarjoamiseksi, jota käyttäjä on pyytänyt.

4 AREA21-SOVELLUS

4.1 Web-sovellus

Projektissa toteutetun sovelluksen tarkoituksena on avustaa sovelluksen käyttäjiä vähentämään resurssien kulutusta antamalla välineet kulutuksen lähes-reaaliaikaiseen seurantaa. Area21-sovelluksen käyttöliittymä ja sen perustoiminnot, joilla pystyttiin lukemaan järjestelmään ulkopuolelta syötettyä dataa, oli toteutettu osana aikaisempaa opinnäytetyötä TAMKissa.

Sovelluksesta kuitenkin puuttui julkaisua varten komponentteja tai toteutuksia, joilla pystyttäisiin hallitsemaan sovelluksen antureita, käyttäjä ja mittauspaikkoja. Eri käyttäjillä ja käyttäjätasoilla pitäisi olla mahdollisuus nähdä heille sallitut toiminnot ja käyttää niitä.

4.2 Sovelluksen ohjelmistoratkaisut

4.2.1 Datan siirto

Sovelluksessa palvelinpuolen ja selainpuolen yhdistämiseen HTTP-sovellusrajapinnan kautta on käytetty Apollo-palvelinta. Apollo on avoimen lähdekoodin toteutus, joka sisältää projektissa käytetyn Express-kirjaston kanssa yhteensopivan GraphQL-palvelintoteutuksen. Vastaavasti projektin React-sovelluskehyksellä toteutetussa selainpuolen toiminnoissa käytettiin Apollo Client-kirjastoa.

GraphQL on Facebookin vuonna 2012 sisäisesti kehittämä data-kyselykieli, joka julkaistiin avoimena lähdekoodina vuonna 2015. Sen tavoitteena on tarjota vaihtoehto REST-pohjaiselle arkkitehtuurille lisäten kehittäjien tuottavuutta ja minimoiden siirretyn datan määrää. GraphQL on otettu tuotantokäyttöön useissa erikokoisissa organisaatioissa. (The GraphQL foundation | An open and neutral home to GraphQL community).

GraphQL koostuu data-kyselykielestä ja ajoympäristöstä, joka suorittaa kyselyt käyttäen tyyppijärjestelmää, joka määritellään datalle. GraphQL ei ole sidottu mihinkään tiettyyn tietokantaan tai tietokantamoottoriin. GraphQL-palvelu luodaan määrittelemällä tyypit, tyyppien avulla kentät ja luomalla funktiot, jotka palauttavat kenttien sisältämät arvot. Näiden kenttien sisältöä kysellään query-operaation avulla ja niiden tilaa voidaan muuttaa mutation-operaation avulla (Taulukko 4). Tämä on GraphQL-spesifikaation luoma käsite-ero ja käytännössä yksittäisen GraphQL-palvelun toiminta perustuu sen toteutukseen.

TAULUKKO 4. GraphQL-käsitteitä.

Käsite	Selitys
type	Esittelee GraphQL-tyypin, jota voidaan pyytää kyselyissä.
query	Pyytää GraphQL-palvelulta haluttujen kenttien sisältöä.
mutation	Muuttaa resurssin tilaa. Kuten query, palauttaa haluttujen kenttien sisällön.

Kuviossa n. on projektissa määritelty GraphQL-tyyppi Sensor. Sillä on kenttiä (devEUI ja notes), joiden tyyppinä on GraphQL-perustyyppi String. Kentän properties tyyppinä on String-primitiivien taulukko. Muiden kenttien tyyppinä on muualla määriteltyjä tyyppejä kuten SensorType tai Apartment. (Kuvio 3)

```
type Sensor {
  devEUI: String
  type: SensorType
  location: String
  properties: [String]
  data(
    dataLimit: Int
    dataFrom: String
    dataTo: String
  ): [SensorData]
  notes: String
  apartment: Apartment
}
```

KUVIO 3. GraphQL-tyyppi Sensor.

Kyselyt lähetetään isännöidylle GraphQL-palvelulle, jotka ajoympäristö tarkistaa ja suorittaa. Kyselyä tehdessä määritellään kentät, joiden sisältö halutaan vastaukseen – tällä voidaan toteuttaa GraphQLn tavoitetta palauttaa ja siirtää vain dataa, josta käyttäjä on kiinnostunut. Kysely voi sisältää vaadittuja tai vapaaehtoisia argumentteja. Kuvion 4. kyselyn datakenttä edellyttää lisäargumentteja, joilla hallitaan palautettavan datan määrää ja aikaikkunaa, josta haku tehdään

```
export const SENSORS_WITH_APARTMENT_QUERY = gql`
  query Sensors($devEUI: ID) {
    sensors(devEUI: $devEUI) {
      ...sensorInfo
      data(dataLimit: 100) {
        time
        temperature
        humidity
        co2
        power
        voltage
      }
    }
  }
  ${sensorInfoFragment}
`;
```

KUVIO 4. GraphQL-kysely, joka palauttaa antureiden kaikki tiedot.

4.2.2 TypeScript-ohjelmointikielen käyttö

Sovelluksen selain- ja palvelinpuolen toteutukseen käytettiin TypeScript-ohjelmointikieltä. TypeScript on JavaScriptin tyypitetty ylijoukko, joka muunnetaan tavalliseksi JavaScriptiksi. Kieli lisää JavaScriptiin ominaisuuksia ”kypsemmistä” ohjelmointikielistä, kuten C++ tai Java. Kielen on kehittänyt Microsoft, joka on julkaissut kielen spesifikaation ilmaiseksi Open Web Foundation Final Specification-sopimuksen perusteella lokakuussa 2012. TypeScript on kasvattanut suosiotaan viime vuosien aikana muiden tyyppiturvallisten kielien mukana (The state of the Octoverse, 2019).

TypeScript kehitettiin parantamaan JavaScriptin käyttöä laajoissa, monimutkaisissa ja modulaarisissa ohjelmistoissa. Yksi sen ominaisuuksista on staattiset

tyypit, jolloin toisin kuin JavaScriptissä, joka on dynaamisesti tyypitetty kieli, TypeScriptissä jokaisella muuttujalla on tyyppi. Kielen käyttäjä voi asettaa muuttujan tyypin itse tai antaa TypeScriptin päätellä sen muuttujaan tallennetun tiedon perusteella. Kehitysympäristöt voivat käyttää tyyppejä käyttäjän työn helpottamiseen antamalla ehdotuksia ja tunnistamalla virhetilanteita. (Microsoft augments JavaScript for large-scale development)

Muita TypeScriptin tuomia laajennuksia, joita tässä projektissa on käytetty ovat luokat (classes), numeroituvat tyypit (enumerable types) ja rajapinnat (interfaces). Luokat toimivat samalla tavalla kuin aikaisemmin mainituissa ”kypsemmissä” ohjelmointikielissä – ne tukevat perintää, luokan jäsenten näkyvyyssääntöjä ja rakentajia. Numeroituvat tyypit ovat TypeScriptissä Number-primitiviin erityistapaus, jolla voidaan määritellä sarja nimettyjä vakioita, joilla on arvo. Rajapinta on puhtaasti käännösaikainen rakenne, jolla voidaan määritellä olioiden muoto, eli jäsenten tyypit ja nimet. Osa näistä ominaisuuksista on tuotu JavaScriptiin ECMAScript 2015-standardin myötä. (TypeScript Documentation

TypeScriptin tuomat lisäominaisuudet poistetaan, kun se käännetään JavaScriptiksi TypeScript-kääntäjällä. Tämä tuotettu koodi on siten ajettavissa ECMAScript-standardin mukaisessa ajoympäristössä (TypeScript Documentation

AREA21-projektissa TypeScriptiä käytetään palvelimen sovelluslogiikan sekä asiakasohjelmiston toteuttamiseen. Palvelimen koodi käännetään julkaisuvaiheessa TypeScript-kääntäjällä JavaScriptiksi, jota ajetaan NodeJS-ajoympäristössä. Asiakasohjelman käännöksestä selaimille vastaa Create React App-sovelluskehys, joka käyttää sisäisesti koodin lataamiseen ja kääntämiseen Babel-kirjastoa.

4.3 Google Analytics

4.3.1 Käyttäjäanalytiikkatarpeet

Area21 projektin sovelluksen mittaustarpeet keskittyvät sovelluksen käyttäjien, käyttäjämäärien ja käyttötapojen tunnistamiseen. Selainpuolen ratkaisussa käyttäjät tunnistetaan evästeiden avulla ja tarvittavat suureet pageview-tapahtumien perusteella. Koska projektin painopisteenä ei ollut mittausalustan kehittäminen, tarpeeksi muodostui valmiin ratkaisun käyttäminen.

Mahdollisina vaihtoehtoina pohdittiin aluksi palvelinpuolen toteutusta, jossa käyttöä seurattaisiin käyttäjien synnyttämien palvelimen logiviestien avulla. Tämä kuitenkin vaatisi huomattavan määrän muutoksia palvelinpuolen toteutuksessa eikä sillä päästäisi samaan yksityiskohtaisuuteen ja tarkkuuteen kuin selainpuolen mittauksella.

4.3.2 Yleistä ja ominaisuudet

Google Analytics on osa Googlen Marketing Platform-tuoteperhettä ja sen ilmainen, rajoitettu versio tarjoaa tarvittavat perusominaisuudet web-sivuston käyttäjäanalytiikan tarpeisiin. Datan keräys on mahdollista web-sivustojen lisäksi esimerkiksi mobiilisovelluksista, jota tässä projektissa ei kuitenkaan tarvittu koska Progressive Web App-tekniikka käsittelee asennusta web-sovelluksena. (Compare Free & Enterprise Analytics Options - Google Analytics)

Perusversio tarjoaa projektin tavoitteiden ja jatkokehityksen kannalta tärkeimmät raporttivaihtoehdot, joihin kuuluvat Audience Report (Yleisöraportti), josta voidaan tarkastella esimerkiksi käyttäjien, sivulatausten määrää sekä tietoa käyttäjien päätelaitteista. Behavior Report (Käyttäytymisrapotti) kertoo, miten sivustoa käytetään – miltä sivulta käyttäjät saapuvat sovellukseen ja mille sivulle navigoidaan seuraavaksi vai tapahtuu esimerkiksi poistuminen sovelluksesta. (Analytics & Data Analysis Features List – Analytics)

Datan tarkempi analysointi on mahdollista ottamalla tarkasteluun tietty segmentti käyttäjistä. Segmentti voidaan suoraan kerättävän tiedon, esimerkiksi päätelaitteen tyypin perusteella tai se voidaan luoda custom dimension-tietojen perusteella.

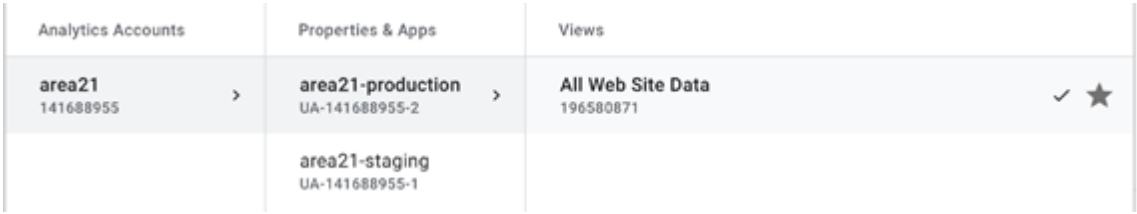
Teknisellä tasolla Google Analytics käyttää tiedon keräämiseen JavaScript-tiedostoa, joka kerää tietoa käyttäjän selaimen tilasta, sivun tilasta ja käyttäjästä itsestään asettamansa ensimmäisen osapuolen evästeen perusteella. Kerättyään tarvittavat tiedot, skripti lähettää HTTP GET-kutsun GIF-kuvapikseliin Googlen palvelimilla. Pikselin kutsuparametreiksi asetetaan kerätyt tiedot. (Tracking Code Overview | Google Analytics | Google Developers)

Google tarjoaa palvelustaan myös maksullista 360-versiota, jossa lisäominaisuuksina tarjotaan esimerkiksi datan vientiä Googlen BigQuery-tietokantaan tai muihin tuettuihin integraatioihin.

4.3.3 Tilin luonti

Google Marketing Platform -alustan käyttöä varten vaaditaan Google-tili. Tätä varten luotiin yhteiskäyttöön tarkoitettu Google-tili. Tilille luotiin kaksi Google Analytics-seurantatunnusta, yksi testikäyttöön ja yksi tuotantokäyttöön, jolla saadaan estettyä sovelluksen kehityksen aikana kertyneen datan kulkeutuminen tuotantoanalytiikkaan.

Jokaisella Google Analytics-seurantatunnuksella on uniikki UA-merkkijono (kuvio 5), jota käytetään sivustolla mittaustapahtumien lähettämiseen. Tämä UA-tunnus upotetaan sovellukseen ympäristömuuttujan kautta jatkuvan integraation ja toimituksen prosessissa, kun sovelluksesta julkaistaan uusi versio.



KUVIO 5. AREA 21 sovelluksen Google Analytics-seurantatunnukset.

4.3.4 Mittauskirjaston käyttö

Perinteisellä websivustolla Google Analytics-kirjaston upotus sivustolle onnistuisi lisäämällä tarvittava script-tagi sivupohjaan ja alustamalla kirjasto, jolloin lähetettäisiin pageview-tapahtuma. Kuten aiemmin on mainittu, SPA-sovelluksessa asia ei ole niin yksinkertainen, sillä tällöin saataisiin kerättyä tietoa vain käyttäjän laskeutumissivuista ja kaikki myöhemmät navigaatiot jäisivät seuraamatta.

Google Analytics-tapahtumia voidaan lähettää suoraan JavaScriptillä ja eräs tapa toteuttaa tämä React-sovelluksessa on liittää tapahtuma sovelluksen selainpään reitittimeen, joka on vastuussa sovelluksen tilanhallinnasta URL-osoitteen perusteella. Tästä seuraa haluttu toiminta: kun käyttäjä navigoi sovelluksessa toiselle sivulle, eli selaimen URL-osoite muuttuu ja sovelluksen reititin päivittää sen perusteella käyttäjän selaimeen uuden tilan, lähetetään pageview-tapahtuma.

AREA 21-sovelluksessa on selainpuolen reitittimenä käytetty React Router-kirjastoa, jossa eri sivuja kuvataan Route-komponentteina. Route-komponentti saa props-syötteenä automaattisesti selaimen osoiterivillä olevan URL-osoitteen ja tähän tietoon voidaan liittää pageview-tapahtuman lähetys.

Rajapintana Google Analytics-kirjastoon sovelluksessa on käytetty JavaScriptillä kirjoitettua react-ga-kirjastoa. Kirjasto ei nimestään huolimatta sisällä React-koodia, mutta sen tarjoama rajapinta on helppokäyttöinen. Kirjasto alustetaan ennen React-sovelluksen renderöintiä seurantatunnuksella, joka on syötetty ympäristömuuttujan kautta.

Integraatio pageview-tapahtumien lähettämiseen URL-osoitteen muuttuessa tehtiin react-ga-kirjaston esimerkkitoteutuksen perusteella (React Router v4 withTracker). Sovelluksessa oletusreittiin liitettiin withTracker-HOC-komponentti, jossa Reactin useEffect-hook tarkkailee URL-osoitteen muuttumista. Mikäli se havaitsee osoitteen muuttuneen, lähetetään react-ga kirjaston avulla pageview-tapahtuma uudella osoitteella.

4.4 Sovelluskomponenttien toteutus

Websovelluksesta puuttuvia toteutuksia olivat:

- Käyttäjien, rakennusten, asuntojen ja anturien hallinta
- Käyttäjän omien tietojen hallinta
- Sähkö- ja vesimittaukset
- Palautelomake

Näiden lisäksi toteutettiin muita käyttöliittymämuutoksia ja -lisäyksiä. Hallintaominaisuuksien toteutus muodostui tarvittavien käyttöliittymäkomponenttien tekemisestä, puuttuvien GraphQL-operaatioiden tekemisestä ja näiden toimintojen manuaalisesta testaamisesta.

Sähkö- ja vesimittausten toteutukseen sisältyivät tarvittavat käyttöliittymäkomponentit sekä halutun kuvaajan toteuttaminen ECharts-kirjaston avulla. Raakaa mittausdataa varten rakennettiin tarvittavat suodattimet sen muokkaamiseksi sopivammaksi esitystä varten. Lisäksi sähködataa haluttiin esittää edellisen ja kuluvan viikon kumulatiivisina summina vertailua varten.

Jotta sovelluksesta voitaisiin kerätä palautetta, toteutettiin sisään kirjautuneille käyttäjille palautelomake. Lomakkeella voi antaa vapaamuotoista palautetta sovelluksesta, joka toimitetaan GraphQL-operaation avulla sähköpostilla sovelluksen ylläpidolle.

Lisäksi websovelluksella oli ongelmia PWA-toiminnollisuuksien kanssa – sovellus ei päivittynyt automaattisesti uusimpaan versioon Safari-selaimilla ja saattoi ajaa selaimen uudelleenohjaussilmukkaan, joka esti sovelluksen käytön. Tämän korjaamiseksi sovelluksen Service Workerin toimintaa paranneltiin paremman yhteensopivuuden varmistamiseksi.

4.4.1 Sovellusdatan muokkaus ja päivitys

Eräs sovelluksen toimintoja on antureiden, asuntojen, rakennusten ja käyttäjien hallinta ja esitys. Projektissa oli jo toteutettu asioiden esitys listana, joten toteutettava ominaisuudet olivat luominen, muokkaaminen ja poisto. Sovellukseen toteutettiin tarvittavat muokkaukset olemassa oleviin komponentteihin lisäämällä listoihin muokkaus- ja poistovalinnat sekä toteuttamalla lisätietonäkymään muokkaustoiminnot. Uusien tietojen tallennusta varten tehtiin tarvittavat lomakkeet. Poistotoimintoon toteutettiin varmistustoiminto, joka vaatii poistettavan asian nimen kirjoittamisen ennen hyväksymistä.

Hallintapaneelien tietojen tallennus ja poisto suoritetaan GraphQL-kielen Mutation-operaatoiden avulla. Operaatiot toteutettiin tarvittavien muutosten tekemiseen ja liitettiin sovelluksen käyttöliittymään Apollo Client-kirjaston Mutation-komponenttien avulla. Komponentti asetettiin päivittämään tiedot hakeva Query-operaatio automaattisesti tallennuksen jälkeen.

4.4.2 Mittausdatan esitys

Sovellusta käytetään antureista tulevan datan esittämiseen. Antureiden data varastoidaan ElasticSearch-tietokantaan, josta sovelluksen palvelin hakee tiedon yksilöidyn devEUI-tunnisteen avulla. Antureiden esitys ja hallinta sovelluksessa tapahtuu Sensors-näkymästä. Näkymässä anturit esitetään listana. Mikäli käyttäjällä on ADMIN-tason oikeudet, voi hän myös muokata tai poistaa antureita.

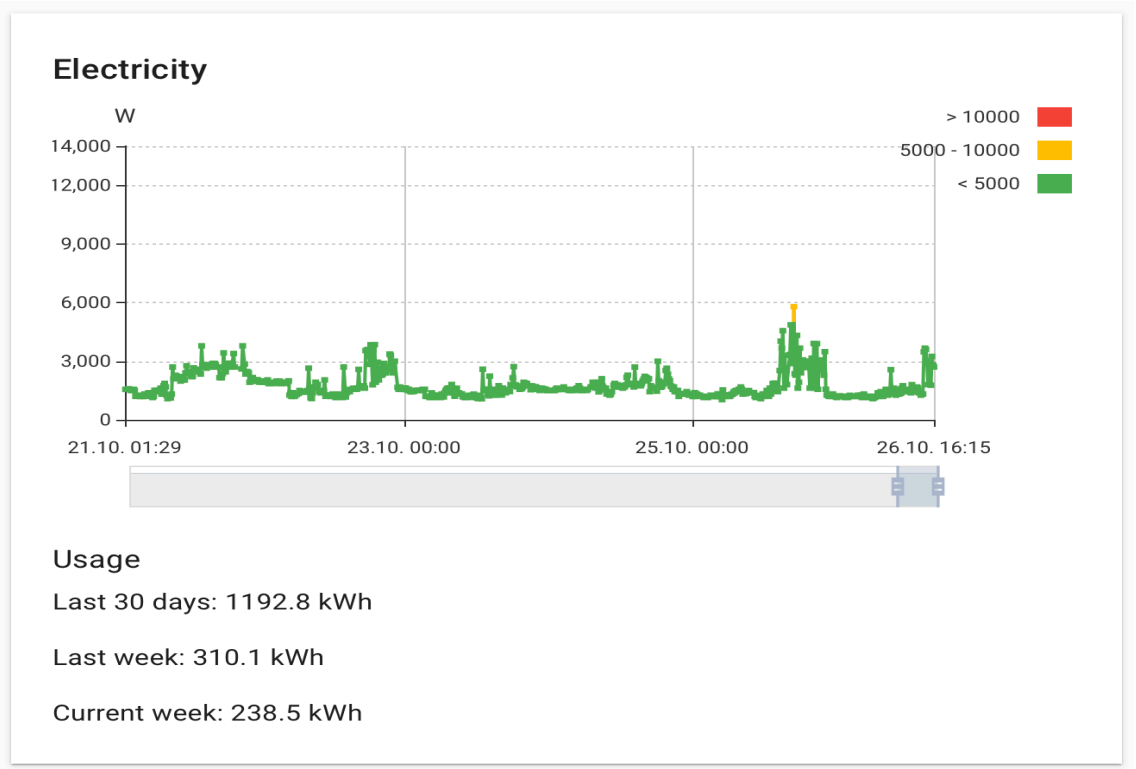
Puuttuvien mittausdatakuvaajien toteutuksessa noudatettiin samaa toteutustapaa- ja tyyliä kuin aikaisemmissa, lämpötilan-, ilmankosteus- ja hiilidioksidimittausten toteutuksissa. Material UI-kirjaston Paper-komponentin päälle toteutettiin React ECharts-kirjaston avulla mittauskuvaaja, jossa y-akselilla on mitattava suure ja x-akselilla aika. Useimmissa tapauksissa mittausdataa pyydetään palvelimelta pidemmältä ajanjaksolta ja kuvaajassa esitetään 5 päivän ajanjakso mittauksia. Kuvaajan alapuolelle on lisätty valitsin, josta voidaan muuttaa kuvaajassa esitettyä ”ikkunaa” mittausdatasta muuttamalla sen paikkaa tai pituutta.

Kuvaajan otsikkona käytetään esitettävän suureen nimeä ja y-akselin otsikkona esitettävän suureen yksikköä. Kuvaajaa voidaan värittää tarvittaessa suureen suurusluokan mukaisesti – näille arvoilla on lisätty selitteet kuvaajan oikealle puolelle. Y-akselin asteikko skaalataan automaattisesti esitettävän datan suurusluokan mukaan.

4.4.3 Sähkömittaus

Sähkömittaustieto saadaan tietokannasta mittauspulsseina, jotka muunnetaan palvelimella wateiksi esitystä varten. Lisäksi tiedoista suodatetaan pois tyhjät mittaukset. Aika-akselille mittauspisteet suhteutetaan mittausajankohdan mukaisesti, koska mittaustiedoissa voi olla puutteita tai mittausajankohtien välillä vaihtelua. Sähkömittauksen mittausväli on kymmenen minuuttia, joka luo kuvaajalle mittauspisteitä niin tiheästi, että ECharts-kirjastosta otettiin käyttöön kuvaajan pisteiden automaattinen tarkkuussuodatus kuvaajan suorituskyvyn parantamiseksi.

Kuviosta 6 näkyy ote sähkömittausdatasta. Vasemmalla näkyy 14,000 wattiin skaalattu y-akseli, jonka arvo perustuu aikaisempaan, korkeampaan mittausarvoon. Oikealla on esitetty kolme arvoluokkaa, joiden perusteella kuvaaja värin valitaan. Kuvaajan alapuolella on esitetty lasketut edellisen 30 päivän, esitysajanjaksoa edeltävän viikon ja kuluvan viikon sähkön kulutusarvot.

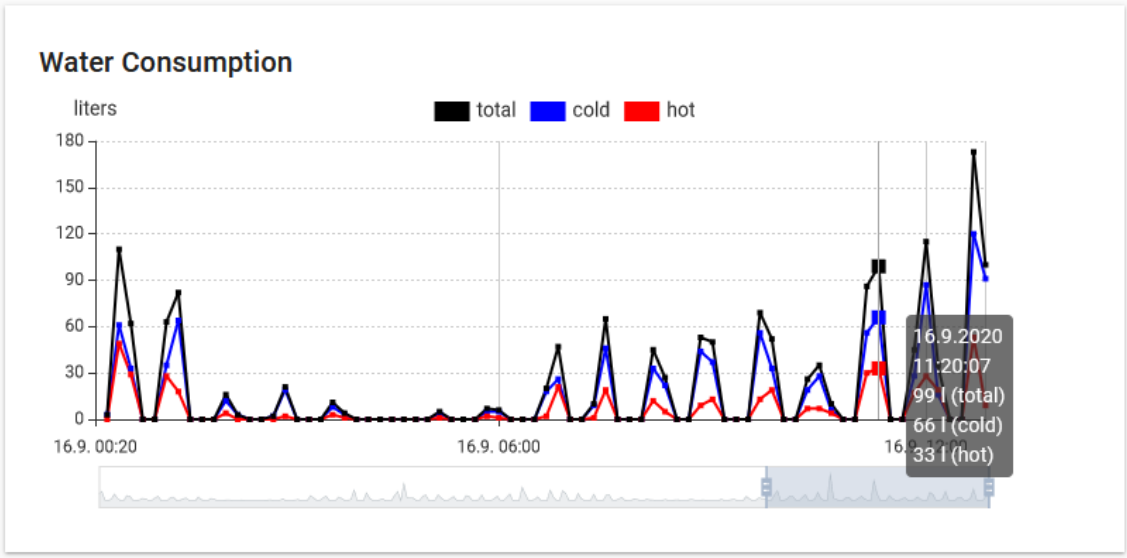


KUVIO 6. Sähkön kulutusmittauksen esimerkkikuvaaja.

4.4.4 Vesimittaus

Veden kulutusmittaustietoa pyydettäessä palvelin hakee anturidatan sisältävästä Elasticsearch-tietokannasta tarvittavat mittapisteet, joista lasketaan edellisestä mittauksesta kuluneen veden määrä litroissa. Datassa on eritelty kuuman ja kylmän veden käyttö erikseen.

Veden kulutuksen seuraamista varten sovelluksen selainpuoleen toteutettiin tarvittava kuvaaja. Kuvaajassa on esitetty veden kulutus kokonaisuudessaan sekä eriteltynä kuuman ja kylmän veden osuutena litroina – selainpuolen koodi laskee yhteiskulutuksen haettuaan datan palvelimelta. Kuviossa 7 on esimerkki vedenkulutusdataa esittävästä kuvaajasta.



KUVIO 7. Veden kulutusmittauksen esimerkkikuvaaja.

Oletuksena esitetään kokonaiskulutus, mutta kuvaajan yläreunasta on mahdollista valita tarkasteltavaksi halutut tiedot erikseen tai yhtä aikaa. Kuvaajan y-akselilla esitetty kulutus litroina skaalataan automaattisesti esitettävän datan arvoihin. Yksittäisestä mittauspisteestä voidaan tarkastella osoittimen avulla mittaushetken aikaleimaa sekä arvoja mittaushetkellä.

5 JATKOKEHITYS

5.1 Suostumuksen kysyminen

Kohdassa 3.5.2 mainittiin siitä, miten evästeiden asettaminen ja käyttö on muuttunut kehittyvän tietosuojasääntelyn myötä suostumuksen alaiseksi toiminnaksi. Vain palvelun toiminnan kannalta välttämättömät evästeet, jotka asetetaan seurauksena käyttäjän toimista sivustolla, voidaan asettaa selaimen evästeasetusten perusteella.

Lisäksi jokaisesta käytössä olevasta evästeestä pitäisi kertoa käyttäjälle käyttötarkoitus, evästeen kesto ja onko kolmannella osapuolella mahdollisuus käyttää evästettä. Sivustolla käytössä oleva Google Analytics on täten päätöksen perusteella ei-välttämätön eväste ja käyttäjälle on tarjottava mahdollisuus kieltäytyä siitä.

Suostumuksenhallinta AREA 21-sovelluksessa voitaisiin täyttää esittämällä käyttäjälle modaalinen dialogi, joka kertoo tarvittavat tiedot sovelluksessa käytössä olevista evästeistä ja mahdollistaa niistä kieltäytymisen tai hyväksymisen. Mikäli käyttäjä kieltäytyy Google Analytics-evästeestä, estetään react-ga-kirjaston lataaminen sovelluksen koodissa.

5.2 Tapahtumien käyttö

Google Analytics-palveluun on mahdollista lähettää myös muita kuin pageview-tapahtumia. Tapahtumat ovat käyttäjän vuorovaikutusta sovelluksen kanssa, joka ei ole uuden näkymän tai sivun lataus, vaan esimerkiksi tietyn nappulan tai toiminnan käyttöä. Tapahtumien mukana lähetetään vähintään sen kategoria ja tapahtuman toiminta, mutta lisäkenttinä sallitaan myös nimikenttä ja mahdollinen arvo. (Event Measurement | Analytics for Web (analytics.js))

AREA 21-sovelluksessa hyötyä voitaisiin saada lisäämällä tapahtumia esimerkiksi kuvaajien käyttötapojen tutkimiseksi. Esimerkiksi kuvaajien aikajakson

muuttamisesta voitaisiin lähettää tapahtuma, jossa kategoriana olisi kuvaajan tyyppi, toimintana kuvaajan aikajakson muutos ja nimikenttänä uusi aikajakso. Tällaisten tapahtumien keräämisen avulla voitaisiin tehdä päätelmiä siitä, käyttääkö kukaan kuvaajan aikajakson muutostoimintoa ja mikä esimerkiksi olisi sopivampi aikajakso esitettäväksi oletuksena.

5.3 Datan yhdistely ja uuden johtaminen

Vaikka Google Analyticsin ilmaisversio ei salli sinne kerätyn datan vientiä integraatioiden kautta, on datan vienti silti mahdollista Export-toiminnon kanssa. Vienti on mahdollista CSV-muodossa, jolloin datan liittäminen muihin järjestelmiin onnistuu standardin CSV-tuonnin avulla.

Esimerkiksi yhdistämällä käyttäytymisestä kerättyä dataa käyttäjien asuntojen kulutusdataan voitaisiin saada selville käyttäjät, joilla kulutus on tavallista suurempaa. Tämän perusteella voitaisiin suostutella tähän joukkoon kuuluvia käyttäjiä tarkastelemaan kuvaajia tai suureita, joissa olisi parantamisen varaa. Lisäämällä tapahtuma suostuttelun toimivuuteen voitaisiin valita käyttöön parhaiten toimiva.

5.4 User ID-seuranta

Koska AREA21-sovellusta on mahdollista käyttää useassa laitteessa samaan aikaan ja sen käyttöön vaaditaan sisäänkirjautuminen, voitaisiin analytiikkadatan tarkkuutta parantaa huomattavasti ottamalla käyttöön Google Analyticsin User ID-toiminto.

User ID-toiminnolla voidaan web-sovelluksen käyttämä sisäinen tunniste, AREA21-sovelluksen tapauksessa käyttäjän UUID, syöttää Google Analyticsille. Tämän tunnisteen avulla Google Analytics pystyy rekisteröimään käyttäjän aiheuttamat tapahtumat jopa silloin, kun käyttäjä käyttää palvelua eri laitteelta.

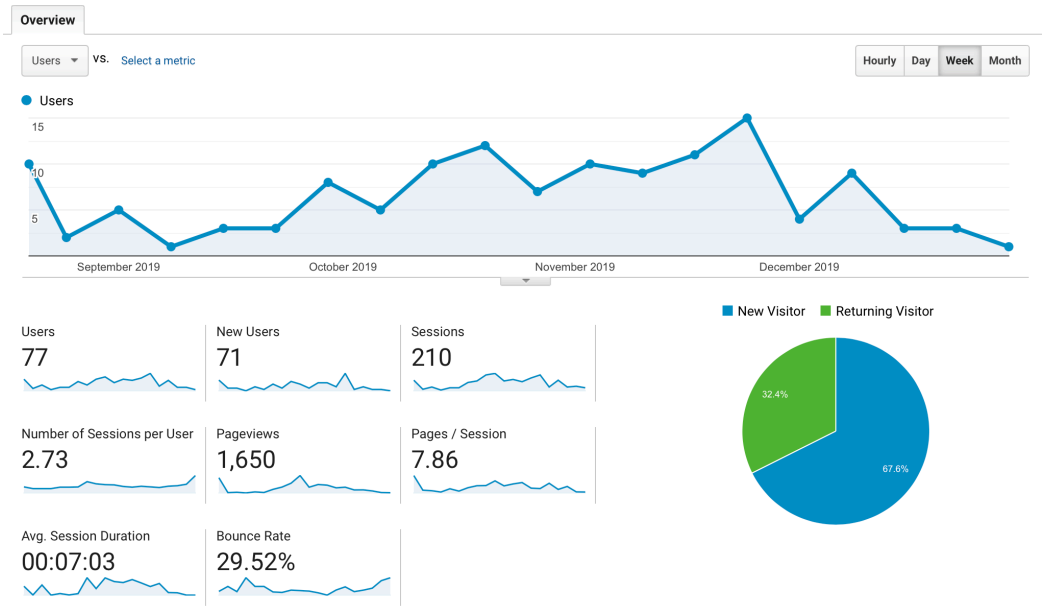
Session unification-toiminnon avulla tapahtumat, jotka käyttäjä aiheuttaa ennen sisäänkirjautumista ja siten UUIDn liittämistä Google Analyticsin istuntoon, voidaan liittää mukaan sen jälkeen, kun ID ilmoitetaan analytics.js-kirjastolle. (About the User-ID feature)

6 POHDINTA

Työssä rakennettiin loppuun AREA 21-sovelluksen hallintaominaisuudet, puuttuvat mittausesitykset, palautelomake ja korjattiin muutamia sovellusta vaivanneita ohjelmistovirheitä. Käyttäjäanalytiikka toteutettiin lisäämällä Google Analytics-kirjasto, joka asetettiin lähettämään perustapahtumat automaattisesti liitettynä sovelluksen selainpuolen reitittimeen.

Näiden toteuttamista varten tutkittiin käytössä olevien teknologioiden ja ympäristöjen käyttöoppaita, toteutusesimerkkejä ja yleisesti internetin ohjelmistokehitysläheitä. Huomattavan tietomäärän tarpeesta huolimatta työn välittömät tavoitteet, jotka vaadittiin sovelluksen julkaisua varten, saatiin täytettyä julkaisupäivään mennessä. Työssä käytetty Kanban-taulu vaatimusmäärittelyssä, työn jakamisessa sopiviin toteutuskappaleisiin ja työn etenemisen seuraamisessa oli erinomainen apu.

Google Analytics keräsi onnistuneesti tietoa käyttäjien vierailuista sovelluksessa (kuvio 8). Kuvassa on Audience Overview-näkymä sovelluksen julkaisupäivästä saman vuoden loppuun olevalta jaksolta viikkotasolla, jolloin sovellusta ovat käyttäneet 77 käyttäjää tehden 1650 sivulatausta. 30 % käyttäjistä poistui ensilatauksen jälkeen ja keskimäärin käyttäjä kävi 2,7 sivulla istuntonsa aikana, jonka perusteella voidaan päätellä sovelluksen käytön onnistuneen.



KUVIO 8. Audience Overview 20.08.2019-31.12.2019.

Tulevaisuudessa web-sovellusprojekteissa olisi syytä tehdä määrittely käyttäjädatan keräämisestä yhteistyössä projektin muiden sidosryhmien kanssa. Tällöin on mahdollista kerätä kaikki mahdollinen ymmärrys sovelluksen toiminnasta ja käyttäjien kvantitatiivisesta kokemuksesta heti projektin alusta alkaen.

Puhtaasti web-sovellusprojektina automaattisen integraatiotestauksen tekeminen React-sovellukselle olisi hyvä lisä vaatimusmäärittelyyn. Erinomaista AREA 21-projektin sovelluksessa olikin CI/CD-ympäristössä automaattisesti suoritettava smoke testing, joka onnistui TypeScriptin käytön ansiosta.

LÄHTEET

Apple Inc. 2019. Safari Privacy Overview, 2019. White paper. Luettu 28.11.2020. https://www.apple.com/safari/docs/Safari_White_Paper_Nov_2019.pdf

Beasley, M. 2013. Practical Web Analytics for User Experience. How Analytics Can Help You Understand Your Users.

Booth, D., Jansen, J. B. 2010. A Review of Methodologies for Analyzing Websites.

Camp, D. 2019. Firefox Now Available with Enhanced Tracking Protection by Default Plus Updates to Facebook Container, Firefox Monitor and Lockwise. Blogikirjoitus. Luettu 28.11.2020. https://blog.mozilla.org/blog/2019/06/04/firefox-now-available-with-enhanced-tracking-protection-by-default/

Croll, A., Yoskovitz, B. 2013. Lean Analytics. O'Reilly.

Facebook. 2020. Components and Props. Dokumentaatio. Luettu 29.11.2020. https://reactjs.org/docs/components-and-props.html

Facebook. 2020. Introducing Hooks. Dokumentaatio. Luettu 29.11.2020. https://reactjs.org/docs/hooks-intro.html

Facebook. 2020. Introducing JSX. Dokumentaatio. Luettu 29.11.2020. https://reactjs.org/docs/introducing-jsx.html

Facebook. 2020. ReactDOM. Dokumentaatio. Luettu 29.11.2020. https://reactjs.org/docs/react-dom.html

Facebook. 2020. Reconciliation. Dokumentaatio. Luettu 29.11.2020. https://reactjs.org/docs/reconciliation.html

Facebook. 2020. Rendering Elements. Dokumentaatio. Luettu 29.11.2020. https://reactjs.org/docs/rendering-elements.html

Facebook. 2020. State and Lifecycle. Dokumentaatio. Luettu 29.11.2020. https://reactjs.org/docs/state-and-lifecycle.html

Facebook. n.d. React Native. Luettu 29.11.2020. https://reactnative.dev/

GitHub. The state of the Octoverse. Raportti. Luettu 17.11.2019. https://octoverse.github.com/

Google. 2014. Understanding web pages better. Blogikirjoitus. Luettu 29.11.2020. https://developers.google.com/search/blog/2014/05/understanding-web-pages-better

Google. 2018. Tracking Code. Dokumentaatio. Luettu 29.11.2020. https://developers.google.com/analytics/resources/concepts/gaConceptsTrackingOverview

Google. 2019. Event Measurement. Dokumentaatio- Luettu 29.11.2020. https://developers.google.com/analytics/devguides/collection/analyticsjs/events
Google. 2020. About the User-ID feature. Käyttöohje. https://support.google.com/analytics/answer/3123662?hl=en

Google. n.d. Compare Free & Enterprise Analytics Options. Dokumentaatio. Luettu 29.11.2020. https://marketingplatform.google.com/about/analytics/compare/

Google. n.d. Analytics & Data Analysis Features List. Dokumentaatio. Luettu 29.11.2020. https://marketingplatform.google.com/about/analytics/features/

Hunt, P. 2013. Why did we build React?. Blogikirjoitus. Luettu 29.11.2020. https://reactjs.org/blog/2013/06/05/why-react.html

Infoworld. 2012. Microsoft augments JavaScript for large-scale development. Artikkeli. Luettu 17.11.2019. https://www.infoworld.com/article/2614863/microsoft-augments-javascript-for-large-scale-development.html

Liikenne- ja viestintävirasto. Kyberturvallisuuskeskus. 2020. Liikenne- ja viestintäviraston päätöskäyttäjän suostumuksen antamisesta evästeiden käyttämiseksi innowise.fi-sivustolla. Päätös. Luettu 28.11.2020. https://www.dataguidance.com/sites/default/files/0_3.pdf

Liikenne- ja viestintävirasto. Kyberturvallisuuskeskus. n.d. Luottamuksellinen viestintä. Ohjesivu. Luettu 28.11.2020. https://www.kyberturvallisuuskeskus.fi/fi/toimintamme/saantely-ja-valvonta/luottamuksellinen-viestinta

Microsoft. 2020. Tracking Prevention in Microsoft Edge (Chromium). Dokumentaatio. Luettu 28.11.2020. https://docs.microsoft.com/en-us/microsoft-edge/web-platform/tracking-prevention

Microsoft. n.d. TypeScript Documentation. Dokumentaatio. Luettu 17.11.2019. https://www.typescriptlang.org/docs/

MDN Contributors. 2020. SPA (Single-page application). Sanasto. Luettu 26.11.2020. https://developer.mozilla.org/en-US/docs/Glossary/SPA

MDN Contributors. 2020. Using HTTP cookies – HTTP. Dokumentaatio. Luettu 26.11.2020 | https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

Qiu, J. 2020. React Router v4 withTracker. Wiki-artikkeli. Luettu 08.06.2019. https://github.com/react-ga/react-ga/wiki/React-Router-v4-withTracker

Section. 2017. How Page Load Time Affects Bounce Rate and Page Views. Blogikirjoitus. Luettu 23.11.2020. https://www.section.io/blog/page-load-time-bounce-rate/

The GraphQL foundation. n.d. An open and neutral home to GraphQL community. Luettu 17.11.2019. https://foundation.graphql.org/
The GraphQL foundation. n.d. GraphQL | A query language for your API. Dokumentaatio. Luettu 17.11.2019. https://graphql.org/

OZiTAG. 2017. Single-page applications vs. multiple-page applications: pros, cons, pitfalls. Blogikirjoitus. Luettu 29.11.2020. https://ozitag.com/blog/spa-advantages

Wilander, J. 2020. Full Third-Party Cookie Blocking and More. Blogikirjoitus. Luettu 28.11.2020 https://webkit.org/blog/10218/full-third-party-cookie-blocking-and-more/