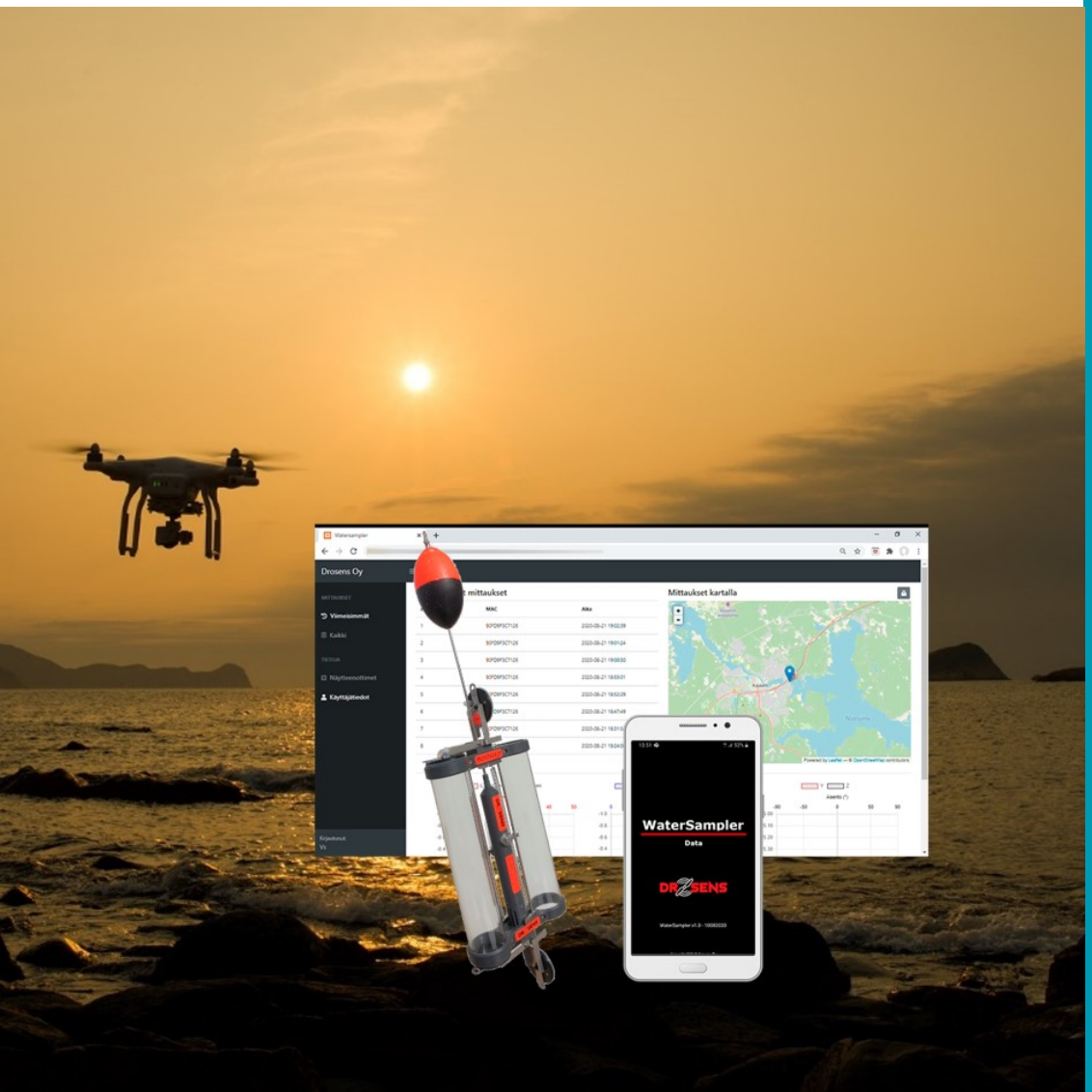
Isto Kinnunen

Progressiivisen Web-applikaation soveltuvuus Bluetooth-pohjaiseen mittalaitteeseen



Insinööri (AMK)

Tieto- ja viestintätekniikka

Kevät 2021

KAMK • University of Applied Sciences

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä(t): Kinnunen Isto

Työn nimi: Progressiivisen Web-applikaation soveltuvuus Bluetooth-pohjaiseen mittalaitteeseen

Tutkintonimike: Insinööri (AMK), tieto- ja viestintätekniikka

Asiasanat: PWA, Vue, progressiivinen web-applikaatio, Web Bluetooth API, Bluetooth

Opinnäytetyön tarkoituksena oli selvittää, voisiko progressiivinen web-applikaatio korvata jo käytössä olevan natiivin Android-sovelluksen Bluetooth-pohjaisessa mittalaitteessa, jolla tehdään veden laadun mittauksia. Työn idea oli lähtöisin Drosens Oy:ltä, koska heillä oli tarve saada iOS-laitteet mukaan veden laadun mittauksiin.

Työssä käsiteltiin, mikä Progressiivinen Web-applikaatio on ja miksi sitä lähdettiin harkitsemaan Android-sovelluksen korvaajaksi. Progressiivinen Web-applikaatio eli PWA, on verkkosivu, joka näyttää ja toimii kuten natiivisovellus. Työssä myös käsiteltiin, mikä Vue on ja miksi se valittiin sovelluksen kehitykseen. Vue on JavaScript-ohjelmistokehys, joka on tarkoitettu verkkosivustojen ja verkkosovellusten kehitykseen. Lopuksi työssä käsiteltiin, mikä Web Bluetooth API on ja kuinka sitä käytetään. Web Bluetooth API on verkkoselaimessa oleva rajapinta, minkä avulla verkkosovellus pystyy muodostamaan Bluetooth-yhteyden halutun Bluetooth-laitteen kanssa.

Työn aikana kehitettiin toimiva PWA-sovellus, jolla oli samat ominaisuudet kuin Android-sovelluksessa. Työssä esitellään sovelluksen kehityksen eri vaiheita, kuinka sovellus muutettiin PWA-sovellukseksi, mitä eri Vuen ominaisuuksia sovelluksen kehityksen aikana käytettiin, miten Web Bluetooth API:lla sovelluksessa Bluetooth-yhteyden muodostaminen tehtiin sekä tarkastellaan, minkälaisia ongelmia sovelluksen kehityksen aikana ilmeni.

Lopuksi tarkastellaan Progressiivisen Web-applikaation soveltuvuutta Android-sovelluksen korvaajaksi. Samalla myös tarkastellaan työn lopussa tehtyjä erilaisia testejä, missä vertailtiin PWA-sovellusta Android-sovellukseen. Työn aikana saatiin selville eri syitä, miksi PWA-sovellus ei tällä hetkellä sovellu korvaamaan Android-sovellusta. Yhdeksi pääsyyksi paljastui se, että sovelluksessa käytetty Web Bluetooth API ei tue iOS-laitteita millään suositulla verkkoselaimella. Samalla myös tarkastellaan, onko työn aikana ilmenneisiin ongelmiin ratkaisuja.

KAMK • University of Applied Sciences

Abstract

Author(s): Kinnunen Isto

Title of the Publication: Compatibility of a Progressive Web Application with a Bluetooth-based measuring device

Degree Title: Bachelor of Engineering, Information and Communication Technology

Keywords: PWA, Vue, progressive web application, Web Bluetooth API, Bluetooth

The purpose of the thesis was to find out whether a progressive web application could replace an already used native Android application in a Bluetooth-based measuring device that makes water quality measurements. The idea for the work came from Drosens Oy because they needed to involve iOS devices in water quality measurements.

The work discussed what a Progressive Web application is and why it was considered as a replacement for an Android application. A progressive web application, or PWA, is a web page that looks and works like a native application. The work also discussed what Vue is and why it was chosen for application development. Vue is a JavaScript framework for the development of websites and web applications. Finally, the work discussed what the Web Bluetooth API is and how to use it. The Web Bluetooth API is an interface in a web browser that allows a web application to establish a Bluetooth connection to a desired Bluetooth device.

During the work, a working PWA application with the same features as the Android application was developed. The work presents the different stages of application development, how the application was converted to a PWA application, what different Vue features were used during application development, how the Web Bluetooth API was used to establish a Bluetooth connection in the application, and what kind of problems occurred during application development.

Finally, we consider the suitability of the Progressive Web application as a replacement for an Android application. At the same time, we also look at various tests done at the end of the work comparing the PWA app to the Android app. During the work, various reasons were identified as to why the PWA application is not currently suitable to replace the Android application. One of the main reasons revealed is that the Web Bluetooth API used in the app does not support iOS devices with any popular web browser. At the same time, it also looks at whether there are solutions to problems that have arisen during the work.

Sisällys

Symboliluettelo

Android	Mobiilikäyttöjärjestelmä
AngularJS	JavaScript-ohjelmistokehys
Bluetooth	Lyhyen kantaman langaton tiedonsiirtotekniikka
CSS	Cascading Style Sheets, WWW-dokumenteille kehitetty tyyliohjeiden laji
HTML	Hypertext Markup Language, hypertekstin merkintäkieli
HTTPS	Hypertext Transfer Protocal Secure, suojattu versio HTTP:stä, joka on verkon pääsääntöinen tapa lähettää dataa selaimen ja sivun välillä.
iOS	Mobiilikäyttöjärjestelmä
JavaScript	Dynaaminen verkko-ohjelmointikieli
JSON	JavaScript Object Notation, tiedonsiirtomuoto
npm	Node Package Manager, paketinhallintajärjestelmä JavaScriptille
PWA	Progressiivinen Web-applikaatio
React	JavaScript-ohjelmistokehys
URL	Uniform Resource Locator, jonkin yksilöllisen resurssin osoite verkossa
UUID	Universally unique identifier, yksilöintijärjestelmä
Vue	JavaScript-ohjelmistokehys
Web Bluetooth API	Rajapinta verkkoselaimessa, jonka avulla pystytään luomaan Bluetooth-yhteys verkkosivun ja Bluetooth-laitteen välille

1 Johdanto

Opinnäytetyön tarkoituksena oli tarkastella, voisiko Bluetooth-pohjaiselle mittalaitteelle tehdyn natiivin Android-sovelluksen korvata progressiivisella web-applikaatiolla eli PWA:lla. PWA on verkkoselaimessa pyörivä natiivin sovelluksen kaltainen verkkosivu, jonka pystyy halutessaan lisäämään tietokoneen tai puhelimen aloitussivulle. Työn idea lähti Drosens Oy:ltä, joka on vuonna 2020 perustettu vedenlaadun mittauksia sekä näytteenottoja dronea hyödyntäen tekevä yritys. Heillä oli jo entuudestaan tehty Android-sovellus mittausten tekemiseen, ja lähdimme siitä miettimään, voisiko Android-sovelluksen korvata PWA-sovelluksella, koska muuten joutuisi kehittämään toisen sovelluksen iOS:lle, että mittauksia voisi suorittaa esimerkiksi iPhonella. Tällä hetkellä iOS-laitteilla mittauksia ei voi tehdä, koska Android-sovellukset eivät toimi iOS-laitteilla.

Työn aikana tarkoituksena oli kehittää PWA-sovellus, missä tulisi olemaan samat perusominaisuudet kuin Android-sovelluksessa. Ominaisuudet tulisivat olemaan Bluetooth-yhteyden muodostaminen mittalaitteeseen, mittalaitteen infon tulostaminen, mihin kuuluu muun muassa mittalaitteen paine, lämpötila, ohjelmistoversio sekä muuta informaatiota mittalaitteesta. Muita ominaisuuksia tulisi olemaan näytteenoton aloittaminen sekä viimeisimmän näytteenoton tulosten haku mittalaitteelta. Sovelluksen kehityksen jälkeen tarkoituksena oli vielä testata, kuinka hyvin PWA-sovellus pärjää Android-sovellusta vastaan erilaisissa testeissä, sekä kuinka luotettavasti data liikkuu siinä Bluetooth-yhteyden välityksellä.

Opinnäytetyössä tavoitteena oli se, että kehitetyn PWA-sovelluksen avulla pystyttäisiin selvittämään voisiko se korvata Bluetooth-pohjaisen mittalaitteen, jo käytössä olevan Android-sovelluksen. Vaatimuksina sovellukselle olivat ne, että sovellus pystyy muodostamaan Bluetooth-yhteyden mittalaitteen kanssa, se pystyy hakemaan mittalaitteelta info tulostuksen, se pystyy suorittamaan näytteenoton sekä se, että sovellus pystyy hakemaan viimeisimmän näytteenoton tulokset mittalaitteelta sekä tulostamaan ne.

Opinnäytetyön teoriaosuuden aikana käydään läpi, mikä on PWA sekä mitä se tarvitsee toimiakseen. Sen jälkeen tarkastellaan mikä on Vue ja miksi se valikoitui sovelluksen kehitykseen. Samalla myös tarkastellaan paria Vue:en liittyvää asiaa. Lopuksi vielä tarkastellaan mikä on Web Bluetooth API, mitä se tarvitsee toimiakseen sekä miten sillä voidaan muodostaa yksinkertainen Bluetooth-yhteys.

Opinnäytetyön työosuuden aikana käydään läpi, kuinka PWA-sovellus on tehty. Tarkemmin tarkastellaan, miten projekti muutettiin PWA-sovellukseksi, mitä Vue:n eri ominaisuuksia siinä käytettiin sekä kuinka Web Bluetooth APIa sovelluksessa käytettiin. Sen jälkeen tarkastellaan paria ongelmaa, mitkä ilmenivät sovelluksen kehityksen aikana. Sen jälkeen tarkastellaan, soveltuuko PWA-sovellus korvaamaan Android-sovelluksen vaiko ei, sekä testituloksia, missä vertaillaan Android-sovellusta PWA-sovellukseen. Lopuksi vielä tarkastellaan, päästiinkö tavoitteisiin ja vaatimuksiin.

2 Progressiivinen Web-applikaatio

Progressiivinen Web-applikaatio eli PWA on verkkoselaimessa toimiva verkkosivu, joka on perinteiseen tapaan kehitetty käyttäen HTML:ää, CSS:ää sekä JavaScriptiä, mutta toimii ja tuntuu kuten natiivisovellus. Nykyään melkein minkä tahansa verkkosivun pystyy halutessaan muuttamaan PWA:ksi. Tänä päivänä PWA-sovelluksen käyttö voi olla jopa parempi vaihtoehto kuin natiivin sovelluksen. PWA-sovellus on nopeampi kehittää kuin natiivisovellus, sekä jos PWA:ta käyttää, ei tarvitse tehdä kuin yksi sovellus, sillä PWA toimii yhtä lailla kaikilla mobiilikäyttöjärjestelmillä, oli se sitten iOS tai Android. Lisäksi PWA-sovelluksessa on mahdollista käyttää täysin samoja ominaisuuksia kuin natiivissa sovelluksessa, joita ovat esimerkiksi push-ilmoitukset sekä sovelluksen käyttö ilman verkkoyhteyttä. [1.] PWA:ta lähdettiin miettimään Android-sovelluksen tilalle sen takia, että siihen saataisiin mukaan myös iOS, eikä tarvitsisi kehittää kuin yksi sovellus.

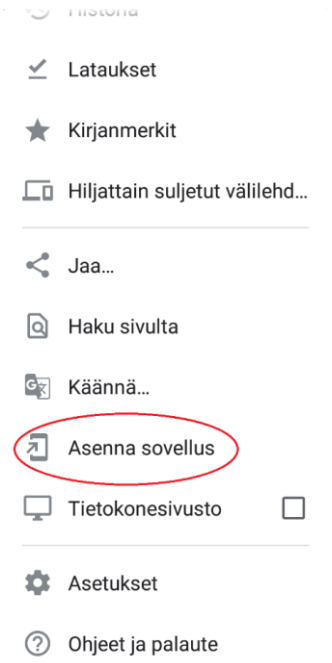
Ensimmäisen kerran idean PWA:n kaltaisesta teknologiasta heitti ilmoille Steve Jobs vuonna 2007, ensimmäisen iPhonen julkaisun yhteydessä. Jobs puhui uudesta tavasta kehittää verkkosovelluksia, jotka näyttävät ja käyttäytyvät samoin kuin natiivit sovellukset. Vasta vuonna 2015 nimi Progressiivinen Web-applikaatio esiteltiin kahden Googlen työntekijän, Alex Russelin sekä Frances Berrimanin toimesta. Tämän jälkeen isot firmat kuten Google ja Microsoft, ovat erityisesti mainostaneet, että uusi PWA-teknologia olisi nykyaikainen tapa pienentää verkko- ja natiivisovellusten välistä eroa. [2.]

Monet suositut verkkosivut omaksuivat PWA-teknologian sen ollessa vielä todella uusi juttu, mikä osoittautui valtavaksi menestykseksi verkkosivujen kannalta. Näitä verkkosivuja olivat muun muassa Alibaba, Twitter sekä Aliexpress. [2.] Alla kuva, jossa muutamia verkkosivuja, jotka omaksuivat PWA-teknologian sen ollessa vielä uutta.

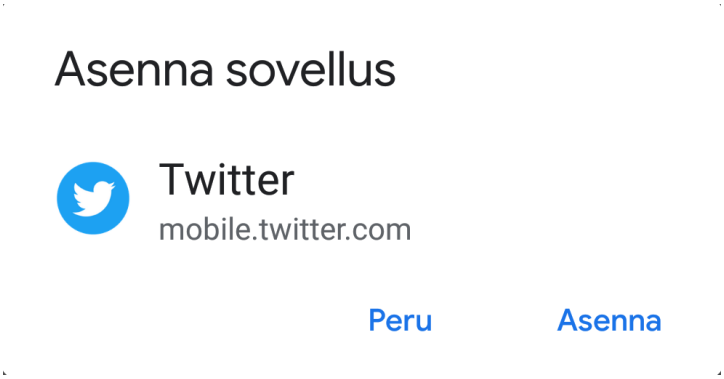
Kuva 1. Verkkosivuja, jotka omaksuivat PWA-teknologian. Mukana myös asia missä nähtiin parannusta, kun PWA:ta alettu käyttää. [2]

Kolme tärkeintä asiaa, mitä PWA tarvitsee toimiakseen ovat HTTPS-yhteys, service worker sekä manifest-tiedosto. Ensimmäinen asia mitä PWA tarvitsee toimiakseen, on HTTPS-yhteys (Hypertext Transfer Protocal Secure), joka on suojattu versio HTTP:stä, joka taas on verkon pääsääntöinen tapa lähettää dataa verkkoselaimen ja verkkosivun välillä. HTTPS-yhteys on salattu sen takia, että se kasvattaisi turvallisuutta datan lähettämisessä. Tämä taas on erittäin tärkeää verkkosivuilla, joissa on arkaluontoista dataa, esimerkiksi verkkopankkiin kirjautumista. [3.] Seuraava asia mitä PWA tarvitsee toimiakseen, on service worker. Service worker on JavaScript-tiedosto. Service worker toimii välityspalvelimena verkkosovelluksen, verkkoselaimen sekä verkkoyhteyden välillä. [4] Tarkemmin service workeristä alla. Viimeinen asia mitä PWA tarvitsee toimiakseen, on manifest-tiedosto. Manifest on JSON-tiedosto. JSON eli JavaScript Object Notation, on tiedonsiirtomuoto. Monet ohjelmointikielet tukevat JSONia, mutta se on erityisen hyödyllinen JavaScript-pohjaisissa sovelluksissa. [5.] Manifest-tiedostosta löytyy tarvittavat tiedot verkkosovelluksesta, että sovelluksen voi ladata sekä että sovellus näkyy käyttäjälle kuten natiivisovellus [6]. Tarkemmin manifest-tiedostosta alla.

Google Chrome -selaimessa PWA-sovelluksen voi helposti asentaa puhelimeen seuraavasti.



Kuva 2. Google Chrome -selaimen sivu asetukset avattuna, mistä löytää ”Asenna sovellus” -napin, mitä painamalla aukeaa ikkuna, mistä pystyy asentamaan PWA-sovelluksen.



Kuva 3. Ikkuna, joka aukeaa ”Asenna sovellus” -nappia painamalla. Ikkunassa pystyy joko asentamaan PWA-sovelluksen tai perumaan asennuksen.

Myös tietokoneelle pystytään asentamaan PWA-sovelluksia. PWA-sovelluksen asennus nappi tietokoneella löytyy verkko-osoitteen oikealta puolelta. Asennus nappi on näkyvissä vain niissä verkko-osoitteissa, jotka ovat PWA-sovelluksia.

Kuva 4. Kuva verkko-osoitteesta, joka on PWA-sovellus ja jonka pystyy halutessaan asentamaan työpöydälle. Asennus nappi ympyröity punaisella.

2.1 Manifest

Manifest on JSON-tiedosto, joka kertoo verkkoselaimelle, kuinka Progressiivisen web-applikaation tulee toimia, kun se on asennettuna tietokoneelle tai puhelimelle. Verkkoselaimet, jotka tukevat manifest-tiedostoa, ovat Chrome, Edge, Firefox, UC Browser, Opera ja Samsung Browser. Saferille on myös tuki, mutta se on osittainen. [7.] Manifestistä tulee vähintään löytyä sovelluksen koko nimi sekä sovelluksen käyttämät kuvakkeet. Muita tietoja, joita manifestistä voi löytyä, ovat esimerkiksi sovelluksen aloitus-URL, sovelluksen lyhyt nimi, taustan väri tai vaihtoehtoisesti teeman väri. [8.] Manifestissä koko nimen (name) kohdalle laitetaan vain sovellukselle haluttu nimi, esimerkiksi ”testisovellus”. Sovelluksen koko nimeä käytetään silloin, kun sovellusta asennetaan. [7] Manifestissä sovelluksen käyttämien kuvakkeiden (icons) kohdalle tulee määrittää kaikki applikaation käyttämät kuvakkeet. Jos verkkoselaimena käytetään Chromea, tulee manifestiin määrittää vähintään kahden kokoiset kuvakkeet, että sovellus toimii oikein. Kuvakkeet ovat kokoa 192x192 ja 512x512. [7.] Kuvakkeet ovat kuvia, jotka näkyvät käyttäjälle puhelimen aloitusnäytöllä tai tiekoneen työpöydällä, kun sovellus on ladattuna laitteelle. Aloitus-URL kertoo verkkoselaimelle, miltä sivulta sovelluksen tulee aueta, kun se käynnistetään. Aloitus-URL myös estää sovellusta aukeamasta siltä sivulta, millä sivulla käyttäjä on ollut, kun sovellus on lisätty aloitusnäytölle. [7.] URL eli Uniform Resource Locator, on vain jonkin yksilöllisen resurssin osoite verkossa. [9] Suomessa URL:stä yleensä käytetään nimeä verkko-osoite. Sovelluksen lyhyt nimi eroaa sovelluksen koko nimestä siten, että se on käytössä puhelimen aloitusnäytöllä tai jossain muualla missä tila voi olla rajattu. [7] Sovelluksen lyhyeksi nimeksi voi laittaa mitä haluaa, vaikka käyttää samaa nimeä, kuin koko nimessä. Jos koko nimi on todella pitkä, tulisi silloin käyttää lyhennettyä versiota nimestä.

Alla esimerkkikuva manifest-tiedostosta.

```
{
    "name":"drosens-app",
    "short_name":"drosens-app",
    "icons":[
        {
            "src":"./img/icons/drosens-192x192.png",
            "sizes":"192x192",
            "type":"image/png",
            "purpose": "any"
        },
        {
            "src":"./img/icons/drosens-maskable-192x192.png",
            "sizes":"192x192",
            "type":"image/png",
            "purpose": "any maskable"
        },
        {
            "src":"./img/icons/drosens-512x512.png",
            "sizes":"512x512",
            "type":"image/png",
            "purpose":"any"
        },
        {
            "src":"./img/icons/drosens-maskable-512x512.png",
            "sizes":"512x512",
            "type":"image/png",
            "purpose":"any maskable"
        }

    ],
    "start_url":"/",
    "display":"standalone",
    "background_color":"#000000",
    "theme_color":"#be0000",
    "prefer_related_applications": true

}
```

Kuva 5. Sovelluksessa käytetty manifest-tiedosto.

2.2 Service Worker

Service worker on JavaScript-tiedosto, joka toimii välityspalvelimena verkkosovelluksen, verkkoselaimen sekä verkkoyhteyden välillä. Se on tarkoitettu muun muassa tehokkaan offline-kokemuksen luontiin, verkkopyyntöjen sieppaamiseen, se ryhtyy tarvittaviin toimiin verkon saatavuuden perusteella sekä se päivittää palvelimella olevia resursseja. Sillä on myös pääsy push-ilmoituksiin. [4.]

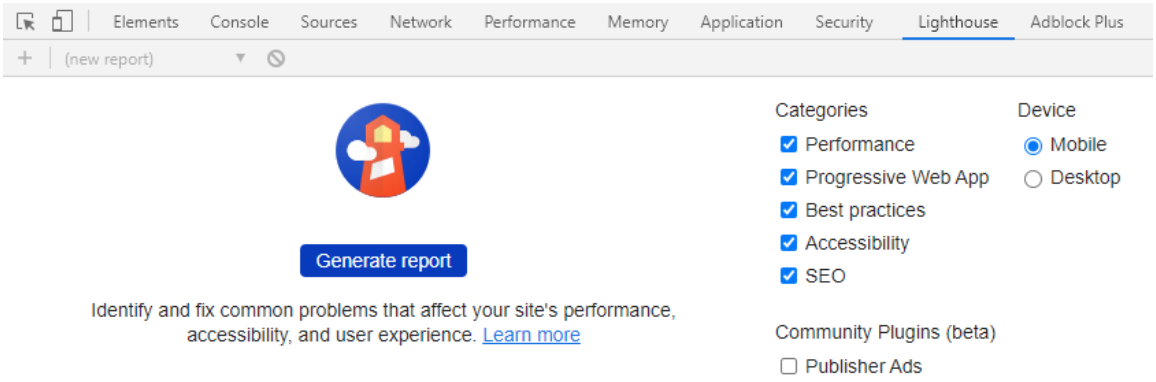
Kuten PWA, myös service worker tarvitsee HTTPS-yhteyden toimiakseen. Koska service worker pystyy sieppaamaan verkkopyyntöjä sekä muokkaamaan vastauksia, ilman HTTPS-yhteyttä olisi vaarana joutua ”Mies välissä” -hyökkäyksen kohteeksi. [10.] Mies välissä -hyökkäys tarkoittaa sitä, että hyökkääjä keskeyttää keskustelun tai tiedonsiirron, joka on jo meneillään käyttäjän ja palvelimen välillä. Kun hyökkääjä on saanut lisättyä itsensä tiedonsiirron ”keskelle”, alkaa hän esittämään molempia osapuolia. Näin hyökkääjä saa kaapattua informaatiota ja dataa molemmilta osapuolilta. [11.]

Service workerin elinkaari koostuu kolmesta vaiheesta, jotka ovat rekisteröinti, asennus ja aktivointi. Elinkaari alkaa rekisteröinnillä. Service workerin rekisteröinti tapahtuu JavaScript-tiedostossa, jossa käytännössä kerrotaan verkkoselaimelle, missä service worker sijaitsee. Kun verkkoselain on rekisteröinyt service workerin, voidaan asennusta yrittää. Service workerin asennuksen yhteyteen voidaan laittaa tehtäviä, joita service worker voi suorittaa asennuksen aikana. Tehtävänä voi olla esimerkiksi, että service worker laittaa osan verkkosovelluksesta välimuistiin, jotta seuraavan kerran kun käyttäjä avaa sovelluksen, latautuu sovellus hetkessä. Kun service worker on asennettu, siirtyy se aktivointivaiheeseen. Jos verkkosovelluksella on jo käytössä service worker, siirtyy uusi service worker odotustilaan, muulloin uusi service worker on valmis suoraan aktivoitumaan. Kun service worker on aktivoitunut, on se valmiina käyttöön. [10.]

2.3 Lighthouse

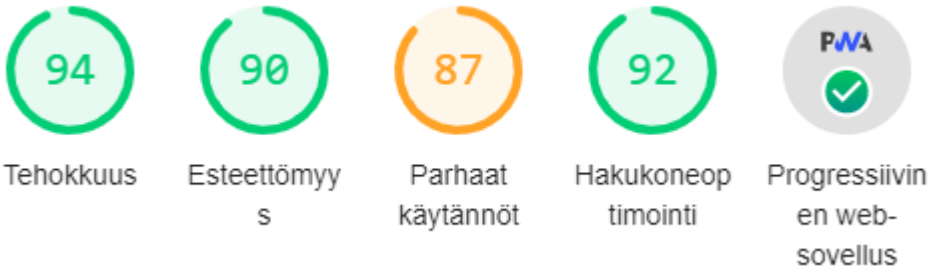
Lighthouse on Google Chromesta löytyvä web-kehitys työkalu, jonka helpoiten löytää Chromessa avaamalla kehittäjätyökalut painamalla näppäimistön F12 nappia. Muita tapoja, kuinka Lighthousen saa käyttöön on esimerkiksi asentaa se Google Chromeen lisäosana, jonka löytää Chromen webstoresta. [12.] Lighthousea käytetään apuna, kun halutaan tunnistaa ja korjata verkkosivulla havaittuja ongelmia. Lighthouse on loistava tapa esimerkiksi seurata sivun suorituskykyä. Työn

aikana käytettiin Lighthousea apuna silloin, kun pyrittiin muuttamaan sovellusta PWA-sovellukseksi. Lighthouse kertoo tarkasti, että mitä sivulta puuttuu ennen kuin sivun pystyy luokittelemaan PWA-sovellukseksi. Alla esimerkkikuva Google Chromen Lighthouse näkymästä.



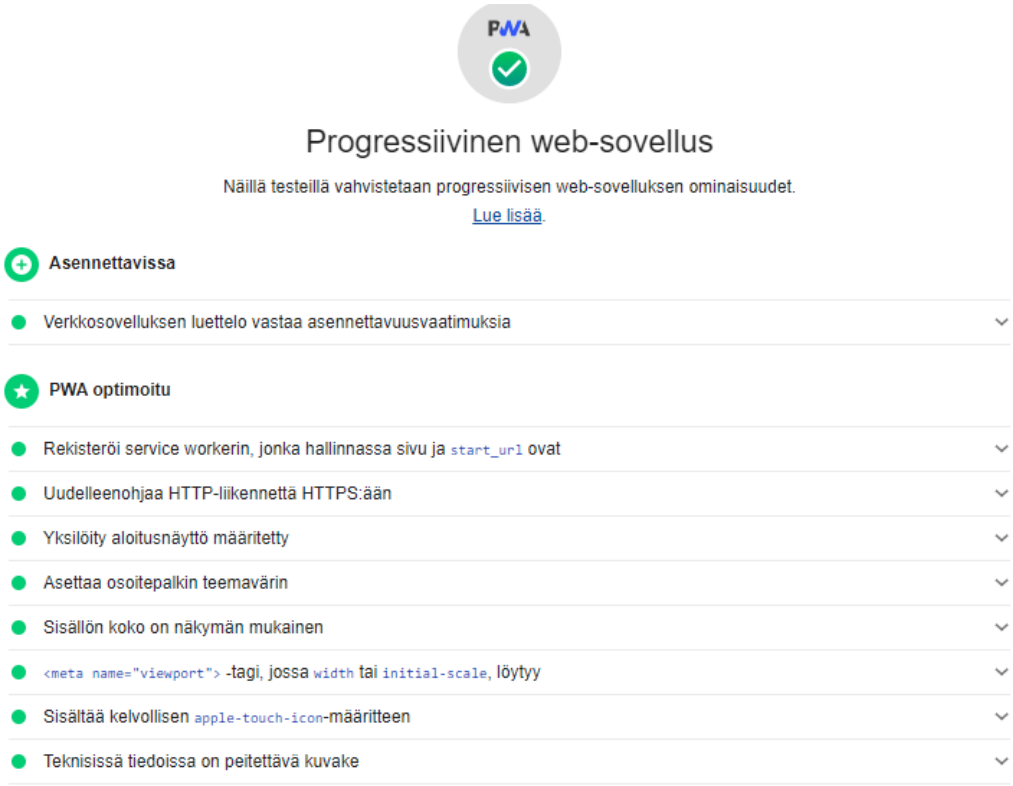
Kuva 6. Google Chromen kehittäjätyökalut avattuna ja Lighthouse valittuna. Generate-reporttia painamalla Lighthouse suorittaa sivun skannauksen ja antaa listan asioista, mitkä ovat sivulla hyvin ja mitkä tarvitsevat korjausta.

Alla esimerkkikuva Lighthouseskannauksesta.



Kuva 7. Lighthouseskannaus tehty työn aikana tehdylle PWA-sovellukselle.

Alla esimerkkikuva listasta, jonka saa sivulle tehdyn skannauksen jälkeen.



Kuva 8. Skannauksen PWA-osio, josta ilmenee, että sivustolla on vaaditut asiat, että sen voi luokitella PWA-sovellukseksi.

3 Vue

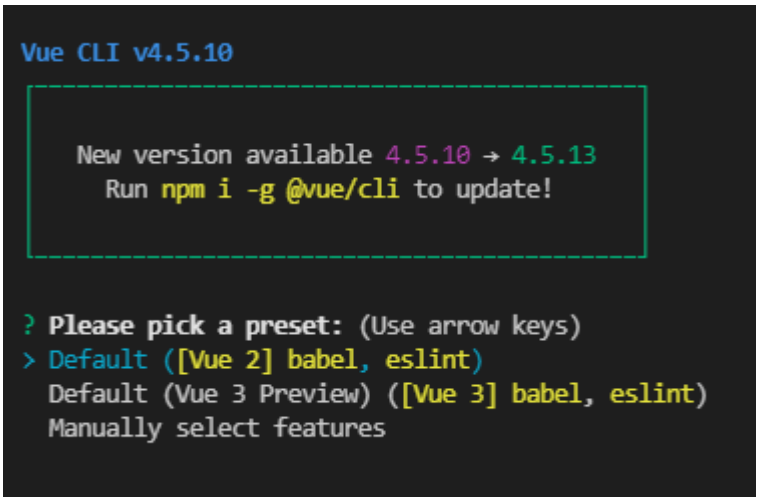
Vue, joka tunnetaan myös nimellä Vue.js, on progressiivinen JavaScript-ohjelmistokehys (JavaScript Framework), käyttöliittymien rakentamiseen. [13] JavaScript-ohjelmistokehys tarkoitta JavaScript koodikirjastoa, jossa on valmiiksi kirjoitettu koodi rutiininomaisten ohjelmointiominaisuuksien ja tehtävien käyttöön. Se on kirjaimellisesti kehys verkkosivustojen ja verkkosovelluksien luontiin. [14.] Vue on ensimmäisen kerran julkaistu helmikuussa vuonna 2014. [15] Tällä hetkellä Vuesta on käytössä kaksi stabiilia versiota, Vue 2 sekä Vue 3. Toisin kuin muut ohjelmistokehykset, Vue on suunniteltu omaksuttavaksi asteittain. Vuen ydinkirjasto on keskittynyt vain näkymäkerrokseen ja se on helppo hakea ja integroida muihin kirjastoihin tai jo käynnissä oleviin projekteihin. Vue on myös täysin kykenevä rakentamaan yhden sivun sovelluksia (Single-Page Applications), kun sitä käytetään yhdessä nykyaikaisten työkalujen ja tukikirjastojen kanssa. [13.] Vue on yleisesti kolmanneksi suosituin JavaScript-ohjelmistokehys heti Reactin ja AngularJS:än jälkeen. [16]

Vuen on luonut entinen Googlen työntekijä, Evan You, joka sai idean Vuesta ollessaan töissä Googlella ja käytettyään siellä toista JavaScript-ohjelmistokehystä, AngularJS:ää monissa eri projekteissa. You mietti voisiko hän tehdä uuden JavaScript-ohjelmistokehyksen, jossa olisi parhaat puolet AngularJS:stä. [17.]

Ennen työn aloittamista työn ohjaajan kanssa ei ollut päätetty, millä sovellusta aletaan kehittämään. Heti alussa oli kaksi ohjelmistokehystä, joista jommallakummalla voitaisiin alkaa sovellusta kehittämään. Ne olivat React sekä Vue. Molemmissa ohjelmistokehyksissä oli hyvät sekä huonot puolet. Vaikka Reactin hyvänä puolena oli se, että työpaikalla oli menossa toinen projekti mitä kehitettiin Reactilla ja missä oli mukana myös, sekä PWA että Web Bluetooth API, valittiin loppujen lopuksi kehitykseen Vue, koska yksinkertaisesti Vue:sta oli entuudestaan paljon enemmän käyttökokemusta kuin Reactista, joten sen osalta hommaksi jäisi vain vanhan kertaaminen, eikä täysin uuden ohjelmistokehyksen opettelu. Vuen huonoksi puoleksi jäi vain se, että Vue ei ole yleisesti niin suosittu kuin React, joten kun lähdetään kehittämään sovellusta, jossa käytetään jotain tiettyä asiaa, malliesimerkkejä löytyy paljon vähemmän.

3.1 Vue CLI

Vue asennettiin käyttäen Vue CLIä, joka on asennettava npm paketti. Npm eli Node Package Manager on paketinhallintajärjestelmä JavaScriptille, jonka avulla JavaScript-kehittäjät pystyvät jakamaan hyödyllisiä verkkosovellusten kehitykseen tarkoitettuja paketteja helposti ja vaivattomasti. [18.] Asennus komennot löytyvät Vue CLI:n viralliselta sivulta. Vue CLI:llä saa käyttöön ”vue” komennon terminaaliin, jolla pystyy helposti ”vue create (projektin nimi)” komennolla tekemään Vue projekti pohjan, josta on helppo lähteä rakentamaan mieluista sovellusta. Vue create komento antaa valita kolmesta vaihtoehtoisesta projekti pohjasta mieluisimman. Tällä hetkellä ne ovat vakio pohja Vue 2:lla, vakio pohja Vue 3:lla tai voi manuaalisesti valita listalta haluamansa ominaisuudet projektiin. Ominaisuuksia ovat esimerkiksi projektin muuntaminen PWA:ksi tai Vue Routerin lisääminen projektiin. Jos valitsee vakio pohjan ja jälkeenpäin haluaa lisätä jonkun ominaisuuden projektiin, onnistuu se käyttäen ”vue add” komentoa. Alla kuva, kun käyttää vue create komentoa.



Kuva 9. Käytetty vue create komentoa. Vue CLI antaa valita kolmesta vaihtoehdosta mieluisimman projekti pohjan.

3.2 Vue Router

Verkkosovelluksissa router on se osa, joka synkronoi nykyisen näkymän selaimen osoiterivin kanssa. Toisin sanoen router on se osa, joka muuttaa URL-osoitteen, kun sivulla klikataan jotain ja näin ollen se osaa näyttää oikean näkymän, kun saavutaan tiettyyn URL-osoitteeseen. [19.]

Sovelluksessa käytettiin Vuen omaa virallista routeria, Vue Routeria. Vue Routeri on npm paketti eli sen saa asennettua käyttäen npm:ää tai sen voi myös asentaa käyttäen Vue CLI:n ”vue add router” komentoa. Jos Vue Routerin asentaa käyttäen Vue CLIä, tulee ottaa huomioon, että samalla ylikirjoitetaan App.vue tiedosto. [20] Vue Router:ssa on index.js JavaScript-tiedosto, missä asetetaan sovelluksessa käytettäville näkymille (component) URL-osoitteet (path), jotta voidaan siirtyä tietylle näkymälle muuttamalla pelkästään URL-osoitetta.

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Home from '../views/Home.vue'

Vue.use(VueRouter)



const routes = [
  {
    path: '/',
    name: 'Home',
    component: Home,
  },
  {
    path: '/about',
    name: 'About',
    component: () => import(/* */ '../views/About.vue')
  },
  {
    path: '/tiedot',
    name: 'Tiedot',
    component: () => import(/* */ '../views/Tiedot.vue')
  },
  {
    path: '/naytteenotto',
    name: 'Naytteenotto',
    component: () => import(/* */ '../views/Naytteenotto.vue')
  },
]

const router = new VueRouter({
  mode: 'history',
  base: '/',
  routes,
})

export default router
```

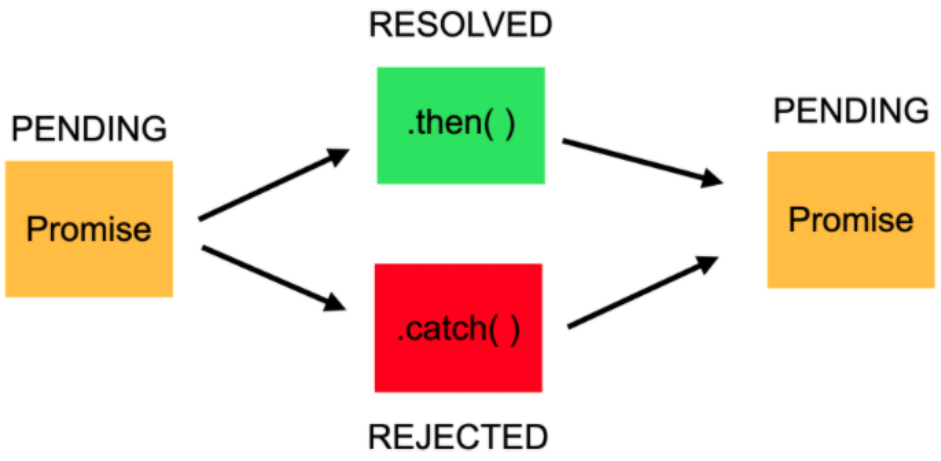
Kuva 10. Esimerkki kuva index.js-tiedostosta.

4 Web Bluetooth API

Web Bluetooth API on verkkoselaimessa oleva rajapinta, minkä avulla verkkosivu pystyy luomaan Bluetooth-yhteyden verkkosovelluksen sekä halutun Bluetooth-laitteen välille. Web Bluetooth API:n kehitys alkanut vuonna 2014 ja on edelleen kehityksen alla.

Web Bluetooth API:ssa on muutamia turvallisuusvaatimuksia, joita tulee noudattaa, että API toimii oikein. Ensimmäinen on, että verkkosivun yhteys tulee olla HTTPS-yhteys (Hypertext Transfer Protocal Secure). Toinen turvallisuusvaatimus on, että metodi *navigator.bluetooth.requestDevice*, jolla aloitetaan Bluetooth-laitteiden etsintä koodi puolella, tulee olla piilotettuna user gesture:n taakse, joka tarkoittaa että käyttäjän tulee itse tehdä jotain ennen kuin funktiota voidaan kutsua, jonka sisällä *navigator.bluetooth.requestDevice* metodi on. User gesturena voi toimia, vaikka napin painallus tai ihan pelkkä hiiren klikkaus. [21.]

Web Bluetooth API perustuu laajalti JavaScript lupauksiin (promise). JavaScript lupaus toimii kuten lupaus toimisi oikeassakin elämässä. Joko lupaus pidetään tai sitä ei pidetä. Kun JavaScriptissä lupaus määritellään ja aika on oikea, se joko ratkaistaan tai hylätään. JavaScript lupaus on olio, jolla on kolme eri tilaa. Se joko on vireillä (pending), ratkaistu (resolved) tai hylätty (rejected). [22.] Kun lupaus on vireillä ja se ratkaistaan, homma jatkuu .then() metodilla uuteen lupaukseen ja jos lupaus hylätään, hylkäyksen syy napataan .catch() metodin avulla. Alla esimerkkikuva, kuinka JavaScript lupaus toimii.



Kuva 11. JavaScript lupauksen toiminta. [22]

4.1 Käyttö

Web Bluetooth API:n käyttö on helppoa, sillä netistä löytyy paljon hyviä ohjeita, joissa opastetaan yksinkertaisen Bluetooth-yhteyden rakentamisen. Koska Web Bluetooth API on asennettuna suoraan selaimen sisään, ei sitä tarvitse alkaa lataamaan mitenkään vaan sitä pääsee käyttämään, kunhan vain seuraa käyttöohjeita. Ainoa asia, joka tulee ottaa huomioon on, että käyttöselain tukee APIa. Parhaimman käyttökokemuksen antaa Google Chrome, mutta muita selaimia, joissa Web Bluetooth API toimii tietokoneella, ovat Edge sekä Opera. Tietokoneen käyttöjärjestelmistä Web Bluetooth API toimii Windows 10:llä, Linux:lla sekä macOs:lla. Mobiilipuolen selaimilla Web Bluetooth API toimii Chrome Androidilla, Opera Androidilla sekä Samsung Internetillä. Valitettavasti iOS:lla Chromessa, Operassa tai sen omassa Safarissa ei ole minkäänlaista tukea Web Bluetooth API:lle. Eli mobiilikäyttöjärjestelmissä Web Bluetooth API toimii Androidilla, mutta ei iOS:lla.

Web Bluetooth API:ssa yhteyden muodostaminen seuraa alussa tiettyä kaavaa ja loppua kohden rakenne vaihtelee käyttökohtaisesti. Kaikki alkaa metodista nimeltä *navigator.bluetooth.requestDevice,* joka sijoitetaan funktion sisälle. Tämä metodi kertoo selaimelle, että Web Bluetooth APIa halutaan alkaa käyttää. Metodin sisään tulee laittaa yksi neljästä suodatin tavasta. Tavoilla yhteistä on se, että jokaisessa tulee määrittää Bluetooth-laitteelta haluttu palvelu (service). Alla kuva funktiosta ”Esimerkki”, jonka sisään laitettu kaikki vaihtoehtoiset tavat, kuinka Bluetooth-laitteita voi etsiä Web Bluetooth API:lla. Suodattimeksi voi laittaa, että API etsii Bluetooth-laitteita laitteen palvelun nimen perusteella (tapa 1), että API etsii palvelut tietyllä 16-bittisellä, 32-bittisellä tai täydellä UUID:lla (tapa 2), että API etsii Bluetooth-laitteita tietyllä nimellä, jonka mukaan tulee myös määrittää laitteelta halutut palvelut (tapa 3) tai että API etsii kaikki lähellä olevat laitteet, joissa on etukäteen määritetty palvelu (tapa 4). [21.]

```
Esimerkki(){
  // Tapa 1
  navigator.bluetooth.requestDevice({
  filters: [{
  services: ['battery_service'] }]
  })
  // Tapa 2
  navigator.bluetooth.requestDevice({
  filters: [{
  services: [0x1234, 0x12345678, '99999999-0000-1000-8000-00805f9b34fb']
  }]
  })
  // Tapa 3
  navigator.bluetooth.requestDevice({
  filters: [{
  name: 'Esimerkki nimi'
  }],
  optionalServices: ['battery_service']
  })
  // Tapa 4
  navigator.bluetooth.requestDevice({
  acceptAllDevices: true,
  optionalServices: ['battery_service']
  })
},
```

Kuva 12. Funktio nimeltä ”Esimerkki”, jonka sisällä näkyvillä kaikki vaihtoehtoiset tavat, kuinka Bluetooth-laitteita voi etsiä.

Seuraavaksi yhteyden muodostaminen Web Bluetooth API:n ja Bluetooth-laitteen välille. Yhteyden muodostus tapahtuu seuraavasti.

```
.then(device => {device.gatt.connect()
return device.gatt.connect()

})
```

Kuva 13. Yhteyden muodostaminen API:n ja Bluetooth-laitteen välille.

Sen jälkeen Bluetooth-laitteelta haetaan ensisijainen palvelu. Tämä määritellään samaksi kuin *navigator.bluetooth.requestDevice* kohdan palvelu. Palvelun voi hakea joko nimen tai UUID:n perusteella. Palvelun hakeminen onnistuu seuraavalla koodilla. [21.]

```
.then(server => {

return server.getPrimaryService('battery_service');
})
```

Kuva 14. Ensisijaisen palvelun hakeminen nimen perusteella.

Yleensä Bluetooth-laitteen palveluilla on monia eri karaktereja (characteristics). Riippuen siitä mitä Bluetooth-laitteella halutaan tehdä, oli se sitten laitteelta lukemista tai laitteelle kirjoittamista, tapahtuu se karaktereja käyttäen. Yksinkertaisessa tapauksessa, jossa halutaan käyttää vain yhtä karakteria, haetaan se joko nimen tai UUID:n perusteella. Jos halutaan käyttää useampaa kuin yhtä karakteria, tulee hakea kaikki palvelun karakterit kerralla. Karakterien haku tapahtuu seuraavasti. [21.]

```
// Haetaan tietty characteristic
.then(service => {

return service.getCharacteristic('battery_level');
})
// Haetaan kaikki characteristicit
.then(service => {

return service.getCharacteristics();
})
```

Kuva 15. Ylemmässä kohdassa haetaan tietty karakteri nimeltä ”battery_level” ja alemmassa kohdassa esitelty tapa, jolla voidaan hakea kaikki palvelun karakterit.

Nyt kun on haettu haluttu karakteri tai halutut karakterit, seuraavaksi tehdään haluttu toiminto sille, esimerkiksi lukea dataa siltä tai kirjoittaa siihen. Yksinkertainen karakterin lukeminen tapahtuu seuraavasti. [21.]

```
.then(characteristic => {
// Luetaan akun varaustaso
return characteristic.readValue();
})
.then(value => {
console.log(`Akkua on jäljellä ${value.getUint8(0)}`);
})
```

Kuva 16. Yläpuolella luetaan karakterin arvo ja alapuolella arvo tulostetaan konsolilokissa 8-bittisenä etumerkittömänä kokonaislukuna.

Karakteriin kirjoittaminen tapahtuu siten, että itse nimetystä vakiosta, tässä tapauksessa esimerkki, tehdään 8-bittinen etumerkitön kokonaislukutaulukko (Uint8Array), jolle on annettu arvo 1. Vakio sen jälkeen kirjoitetaan karakteriin. Lopuksi vielä voi tehdä vaihtoehtoisen konsolilokauksen, jonka avulla tiedetään, että kirjoitus on mennyt perille onnistuneesti. Alla kuva karakteriin kirjoittamisesta. [21.]

```
.then(characteristic => {
const esimerkki = Uint8Array.of(1);
return characteristic.writeValue(esimerkki);
})
.then(_ => {
console.log('Onnistui');
})
```

Kuva 17. Yksinkertainen tapa kirjoittaa karakteriin.

Kaiken lopuksi vielä kannattaa laittaa toiminto, joka antaa virheilmoituksen, jos jokin asia on mennyt pieleen funktion sisällä. Alla esimerkkikuva toiminnosta.

```
.catch(error => { console.log(error.message); });
```

Kuva 18. Toiminto, jolla napataan funktion sisällä tapahtuvat virheet ja tulostetaan ne konsolilokiin.

Joskus voi olla tilanne, missä halutaan katkaista Bluetooth-yhteys. Bluetooth-yhteyden katkaisu onnistuu Vue:ssa seuraavasti. Ensin tehdään uusi data, jolle annetaan tyhjä arvo. Alla esimerkkikuva tehdystä uudesta datasta.

```
data() {
  return {
    bluetoothDevice:null,
  }
},
```

Kuva 19. Tehty uusi data nimeltä bluetoothDevice.

Kun uusi data on tehty, voidaan asettaa API:n ja Bluetooth-laitteen välille muodostettu yhteys bluetoothDevice:ksi, jota pystytään jatkossa kutsumaan muiden funktioiden sisällä this.bluetoothDevice:llä. Alla esimerkkikuva, jossa muodostettu yhteys asetetaan bluetoothDevice:ksi.

```
.then(device => {device.gatt.connect()
this.bluetoothDevice=device;

return this.bluetoothDevice.gatt.connect()
})
```

Kuva 20. Muodostettu yhteys asetetaan bluetoothDevice:ksi.

Lopuksi vielä tehdään funktio nimeltä lopetus, jota kutsumalla ohjelma tarkistaa, onko Bluetooth-yhteyttä muodostettu ja jos on, se sammutetaan. Alla esimerkkikuva funktiosta.

```
lopetus(){
  if (this.bluetoothDevice.gatt.connected) {
  this.bluetoothDevice.gatt.disconnect();
  }else{
  console.log('> Bluetooth Device is already disconnected');
  }
  },
```

Kuva 21. Funktion nimeltä ”lopetus”, jota kutsumalla Bluetooth-yhteys lopetetaan.

5 Sovelluksen toteutus

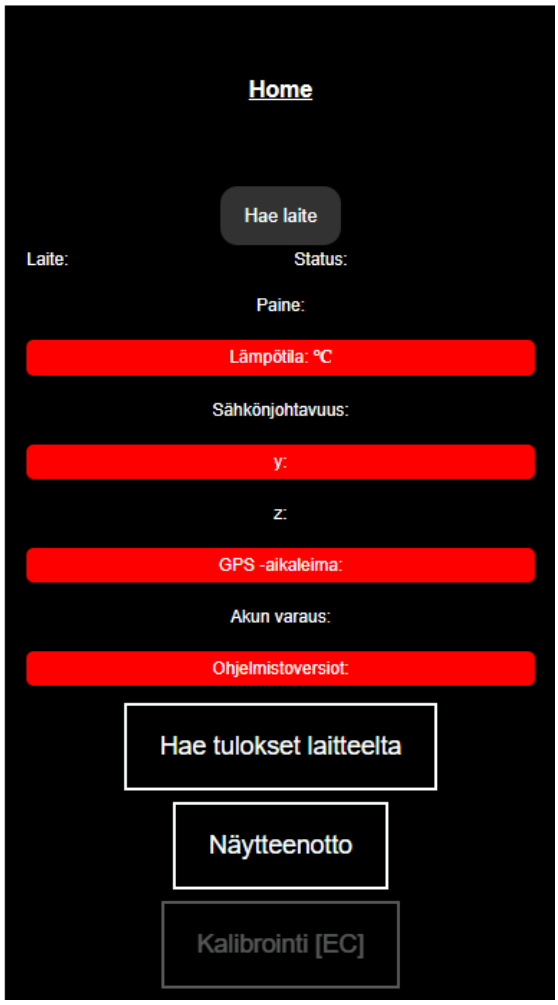
Sovellus kehitettiin Android-sovelluksen pohjalta. Sovelluksen kehitykseen valittiin Vue, koska siitä löytyi kokemusta jo entuudestaan. Sovelluksessa pyrittiin pitämään käyttöliittymä kohtuullisen samanlaisena, kuin Android-sovelluksessa ilman sen suurempaa viilaamista vaan pääsääntöisesti keskityttiin sovelluksessa sen toiminnallisuuteen. Sovelluksen valmiiksi saantiin meni noin kaksi kuukautta, mutta tämä johtui pääosin siitä, että sovelluksessa käytetty Web Bluetooth API oli täysin uusi tuttavuus ja näin ollen jouduttiin opettelemaan sen toiminnallisuudet kokonaan. Myös PWA itsessään oli täysin uusi tuttavuus, mutta sovelluksen muuntaminen PWA:ksi ei tuottanut juurikaan sen suurempia ongelmia. Sovelluksen kehityksen alussa ensimmäiseksi lähdettiin kehittämään sovellukselle käyttöliittymän. Projekti pohjana käytettiin Vue CLI:n vakio projekti pohjaa, mistä oli helppo lähteä kehittämään sovellukselle käyttöliittymää sekä sovellukselle etusivua. Sovelluksen kehitys aloitettiin etusivusta, mistä poistettiin kaikki vanha vakio pohjasta ja lisäiltiin sitä mukaan uutta tilalle. Uusien sivujen lisäilyssä käytettiin apuna Vue Routeria. Aina kun edellinen sivu oli valmis, siirryttiin seuraavan kehitykseen. Sovellukseen sivuja yhteensä tuli neljä kappaletta. Etusivu, info tulostus sivu, näytteenotto sivu sekä viimeisimmän näytteenoton tulosten haku sivu. Kaikki muut sivut pyrittiin mallintamaan Android-sovelluksen sivuista, paitsi viimeisimmän näytteenoton tulosten haku sivun, sillä se sivu tehtiin nopeasti parissa päivässä vain näytteenotto tulosten raa’an datan tarkasteluun. Sivun tarkoituksena oli, että siihen saadaan tulostettua viimeisimmän näytteenoton raaka data, mitä sen jälkeen voidaan vertailla Android-sovelluksella saatuun raakaan dataan ja tarkastella täsmäävätkö eri sovelluksilla saadut datat keskenään.

Sovelluksen muuntamista PWA:ksi apuna käytettiin Vue CLI:n omaa "vue add pwa" toimintoa, joka lisää projektiin PWA:han tarvittavat tiedostot, joita ovat esimerkiksi service workerin rekisteröinti tiedosto tai vakio kuvakkeet, joita voi käyttää PWA-sovelluksessa, jos ei halua lisätä omia. Lisäksi, jotta sovelluksen saa toimimaan kuten PWA-sovelluksen, siihen täytyy lisätä vielä manifest-tiedosto sekä määrittää service workerille asetukset, joko lisäämällä ne vue.config.js tiedostoon, joka on JavaScript-tiedosto tai package.json tiedostoon, joka on JSON-tiedosto. Tarkemmat ohjeet löytyvät Vue CLI PWA:n virallisilta sivuilta. Pienen viilauksen jälkeen sovellus saatiin muutettua PWA:ksi ja se toimi kuten pitikin. Puhelimella sekä tietokoneella pystyi palvelimelta lataamaan sovellus version, joka toimi kuten pitääkin. Ainoaksi pikku ongelmaksi jäi se, että joka kerta kun palvelimelle lisättiin uusi sovellusversio, olisi PWA:n pitänyt ilmoittaa siitä, että saatavilla on uusi versio ja ladata se, mutta emme saaneet sitä toimimaan vaan joka kerta, kun halusimme

uuden version käyttöön, täytyi evästeet ja välimuisti poistaa selaimesta, että uusin versio päivittyi palvelimelle sekä sovellus versioon. Ongelmaan olisi varmaan löytynyt nopea korjaus, mutta ongelma ei ollut niin kriittinen PWA-sovelluksen soveltuvuuden testaamisen kannalta, että sitä tarvitsisi lähteä korjaamaan.

5.1 Bluetooth-yhteyden muodostus sovelluksessa

Web Bluetooth API kohdassa jo kerrottiin, kuinka yksinkertainen Bluetooth-yhteys muodostetaan selaimen sekä Bluetooth-laitteen välille. Sovelluksessa tällä hetkellä muodostetaan kolmella eri sivulla Bluetooth-yhteys ja yhteyden muodostus eroaa joka kerta hieman toisistaan. Ensimmäinen sivu missä voidaan muodostaa yhteys, on heti etusivun jälkeinen sivu, missä laitteelta haetaan info tulostus. Info tulostuksen haussa täytyy käyttää Web Bluetooth API:n ilmoitukset toimintoa, sillä karakteri, jolta info tulostus haetaan, on laitettu lukeminen (read) pois päältä ja laitettu indikoiminen (indicate) päälle. Indikoimista halutaan käyttää lukemisen sijaan sen takia, koska se on tarkoitettu nimenomaan sellaiseen tilanteeseen, missä halutaan seurata karakterin arvoja, jotka muuttuvat koko ajan. Alla kuva info tulostussivusta.



Kuva 22. Info tulostussivu.

Karakterin lähettämään notifikaatioita saadaan seuraavasti. Aluksi tehdään uusi data, jolle annetaan tyhjä arvo. Alla kuva uuden datan teosta.

```
data() {
  return {
    mycharacteristic:null,
  }
},
```

Kuva 3. Tehdään uusi data nimeltä mycharacteristic, jolle annetaan tyhjä arvo.

Sen jälkeen haetaan haluttu karakteri, johon laitetaan notifikaatiot päälle startNotifications-toiminnolla. Sen jälkeen karakteri asetetaan mycharacteristiciksi, jotta sitä voidaan kutsua myöhemmin muiden funktioiden sisällä this.mycharacteristic:llä. Lopuksi karakteriin lisätään tapahtuma kuuntelija (add event listener), joka kuuntelee karakterin arvon muuttumista (characteristic value

changed) ja loppuun vielä funktio (esimerkissä nimellä esimerkki), missä seuraavaksi pääsee käyttämään karakterin arvoa haluttuun tarkoitukseen.

```
.then(characteristic =>
characteristic.startNotifications())
.then(characteristic => {
  this.mycharacteristic=characteristic

  this.mycharacteristic.addEventListener('characteristicvaluechanged', this.esimerkki)
})
```

Kuva 24. Esitelty tapa, kuinka karakteriin laitetaan notifikaatiot päälle sekä kuinka karakteriin lisätään tapahtuma kuuntelija.

Seuraavaksi edellä nimettyyn funktioon, tässä tapauksessa esimerkki, annetaan parametri nimeltä event. Sen jälkeen funktion sisälle tehdään muuttuja (alla olevassa kuvassa nimellä value) ja asetetaan se karakterilta saaduksi arvoksi. Sen jälkeen tehdään uusi muuttuja (alla olevassa kuvassa nimellä utf8decoder) ja se asetetaan uudeksi tekstin dekoodaajaksi. Lopuksi tehdään vielä yksi muuttuja (alla olevassa kuvassa nimellä u8arr), joka asetetaan karakterin arvon taulukkopuskuriksi (buffer), mikä voidaan lopuksi dekoodata käyttäen tekstin dekoodaajaa ja asettaa se sen jälkeen muuttujaksi (alla olevassa kuvassa nimellä infodata). Nyt arvo on muutettu selkokielelle ja sitä voidaan käyttää haluttuun tarkoitukseen.

```
esimerkki(event){
  let value = event.target.value;
  let utf8decoder = new TextDecoder();
  let u8arr = value.buffer
  var infodata = utf8decoder.decode(u8arr)
  },
```

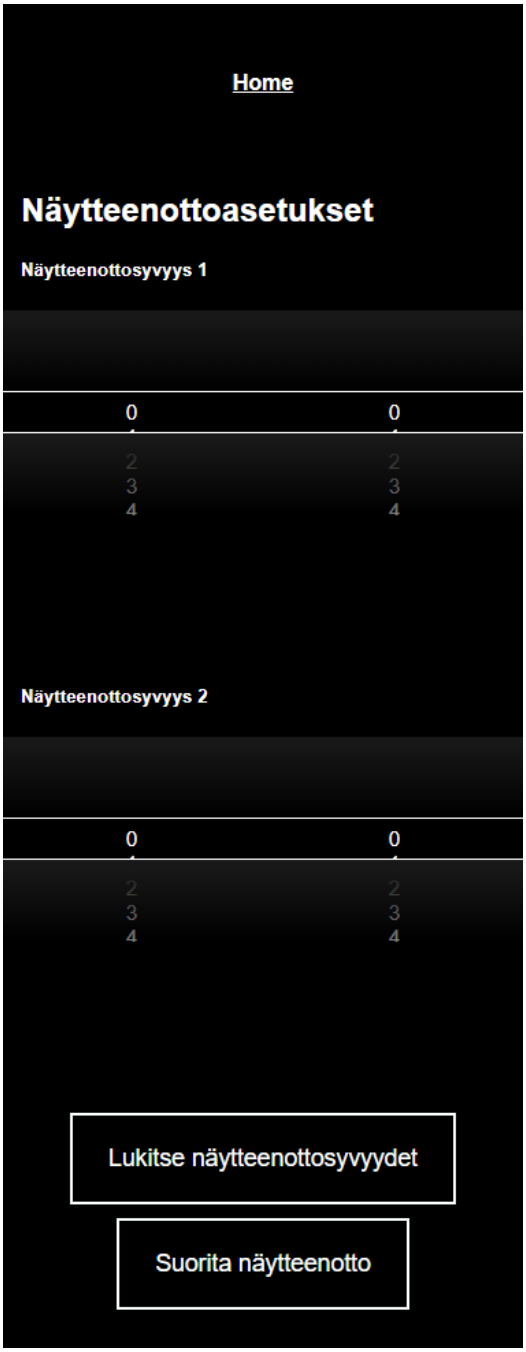
Kuva 25. Funktio nimeltä esimerkki, jonka sisällä tapahtuu karakterin arvon muuntaminen selkokielelle.

Lopuksi vielä karakterin notifikaatioiden lopetus. Lopetus aloitetaan antamalla lopetus käsky stopNotifications ja lopuksi vielä poistamalla tapahtuma kuuntelija (remove event listener). Karakterin notifikaatioiden lopetus voidaan laittaa oman funktion sisään tai sen voi vaikka liittää Bluetooth-yhteyden katkaisun mukaan.

```
lopetus(){
this.mycharacteristic.stopNotifications()
.then(characteristic => {
characteristic.removeEventListener('characteristicvaluechanged', this.esimerkki);
})
},
lopetus(){
this.mycharacteristic.stopNotifications()
.then(characteristic => {
characteristic.removeEventListener('characteristicvaluechanged', this.esimerkki);
if (this.bluetoothDevice.gatt.connected) {
this.bluetoothDevice.gatt.disconnect();
} else {
console.log('> Bluetooth Device is already disconnected');
}
})
},
```

Kuva 26. Esitelty tapa, jolla karakterin notifikaatiot lopetetaan. Yläpuolella oman funktion sisällä ja alapuolella Bluetooth-yhteyden katkaisun yhteydessä.

Seuraavaksi Bluetooth-yhteyden muodostus sovelluksessa tehdään, kun halutaan aloittaa näytteenotto. Sivulla aluksi valitaan näytteenottosyvyydet. Näytteenottosyvyyksiksi tulee asettaa yli 0.2 m tai muuten sovellus ilmoittaa, että syvyydet ovat virheelliset ja joka samalla myös estää näytteenoton aloittamisen.



Kuva 27. Näytteenoton aloitussivu.

Kun syvyydet ovat laitettu oikein ja näytteenotto aloitettu, alkaa Bluetooth-yhteyden muodostus normaaliin tapaan. Ensin haetaan laitetta nimen perusteella, muodostetaan siihen yhteys ja haetaan ensisijainen palvelu. Sen jälkeen alkaa karakterien haku. Näytteenotossa karakterien haku eroaa info tulostuksesta sen verran, että siinä käytetään useampaa kuin yhtä karakteria. Karakterit tulee hakea oikeassa järjestyksessä sekä kaikki karakterit yksitellen, koska muuten sovelluksen kanssa ilmenee ongelmia, kun se laitetaan palvelimelle ja sitä testataan puhelimella. Eli toimintoa, joka hakee kaikki karakterit yhdessä, ei voi käyttää. Kun sovellus on hakenut kaikki karakterit ja

tehnyt niille tarvittavat asiat, sovellus palaa takaisin info tulostus sivulle ja katkaisee sovelluksen ja mittalaitteen välisen Bluetooth-yhteyden, ettei mitään ongelmia tule näytteenoton suorittamisen yhteydessä.

Viimeinen sivu missä Bluetooth-yhteys muodostetaan, on viimeisimmän näytteenoton tulosten haku sivu. Sivu tehtiin nopeasti parissa päivässä vain testi tarkoitukseen. Yhteyden muodostus alkaa normaaliin tapaan eli haetaan mittalaitetta nimen perusteella, muodostetaan yhteys ja haetaan ensisijainen palvelu. Sen jälkeen haetaan kaikki karakterit, koska näytteenoton tulosten haussa käytetään useampaa kuin yhtä karakteria. Tässä kohtaa toimintoa, joka hakee kaikki karakterit kerralla, voidaan käyttää, koska siitä ei synny mitään ongelmia sovelluksen kanssa, kun sen laitetaan palvelimelle.

Kuva 28. Viimeisimmän näytteenoton tulosten hakusivu.

Näytteenoton tulosten haussa yhdelle karakterille laitetaan notifikaatiot päälle sekä tapahtuma kuuntelija (event listener), jonka tarkoituksena on kuunnella karakterin arvojen muuttumista. Kun tapahtuma kuuntelija kuuntelee karakterin arvojen muuttumista, toiseen karakteriin kirjoitetaan samaan aikaan. Tätä tapahtuu niin pitkään, että sovellus ilmoittaa, että viimeisimmän näytteenoton tulokset on saatu haettua ja tulostaa ne sivulle. Nyt tuloksia voidaan käyttää testaamiseen.

5.2 Ongelmia sovelluksen kehityksessä

Sovelluksen kehityksen aikana oli pari isompaa ongelmaa. Yleisesti ongelma olivat Web Bluetooth API:n kanssa. Ensimmäinen isompi ongelma tuli silloin, kun alettiin ensimmäisen kerran Bluetooth-yhteyden muodostamista tekemään info tulostus sivulla. Alussa oli vaikeuksia saada mittalaitteelta minkäänlaista dataa tulemaan sovellukselle. Lopulta kun mittalaitteelta saatiin dataa tulemaan sovellukselle, oli se täyttä sekamelskaa. Sen jälkeen täytyi alkaa etsimään keinoa muuntaa sekamelska selkokielelle. Lopulta löytyi keino, millä sekamelskan sai muutettua selkokielelle ja ongelma saatiin ratkaistua. Ongelman ratkaisemiseen kului monta päivää, ehkä jopa viikko.

Seuraava ongelma ilmeni heti seuraavalla sivulla, näytteenoton aloituksen yhteydessä. Ongelmana oli saada lähetettyä mittalaitteelle näytteenottosyvyyksiä. Alussa ei saatu lähetettyä mittalaitteelle minkäänlaisia arvoja. Sitten selvisi, että lähetettävien arvojen tulee olla tiettynä muotona, että ne menevät perille. Lopulta saatiin lähetettyä kokonaislukuja, mutta ongelmana oli vielä se, että desimaaliluvut eivät menneet perille asti. Pienen taistelun jälkeen, desimaalilukuongelmakin saatiin ratkaistua ja sovellus kerettiin jo julistaa olevan valmis, mutta kun sovellus laitettiin palvelimelle ja sillä testattiin näytteenoton suorittamista, ei se enää toiminutkaan. Ilmeni, että koodissa oli jotain vikaa, mikä esti puhelimella näytteenoton aloittamisen. Niinpä osa koodista täytyi rakentaa uudelleen. Koodin kanssa tappeluun meni todella kauan ja alkoi näyttää, että saadaanko sovellusta ollenkaan valmiiksi. Onneksi työpaikalla olevasta toisesta projektista tuli todella hyvä vinkki, mikä osoittautui toimivaksi ratkaisuksi ongelmaan. Lopultakin sovelluksella pystyi suorittamaan näytteenottoja.

6 PWA:n soveltuvuuden tarkastelua

Opinnäytetyön tarkoituksena oli tutkia voisiko PWA-sovellus korvata mittalaitteelle tehdyn Android-sovelluksen. Työn aikana kehitettiin täysin toimiva sovellus, missä oli samat ominaisuudet kuin Android-sovelluksessa. PWA ei tällä hetkellä sovellu täysin siihen, mitä lähdettiin hakemaan, sillä vaikkakin sovellus toimii kuten pitääkin sekä vaikkakin sillä pystyy suorittamaan näytteenoton kuten pitääkin, esiintyi siinä muutamia suuria ongelmia.

Ensimmäinen ongelma oli se, että PWA-sovellus ei ollut täysin käyttäjäystävällinen. PWA-sovelluksessa käytetyssä Web Bluetooth API:ssa oli ongelmallista se, että joka kerta kun halusi sovelluksen kommunikoida mittalaitteen kanssa oli se sitten mittalaitteelle kirjoittamista tai mittalaitteelta lukemista, oli Bluetooth-yhteys muodostettava joka kerta uudestaan. Android-sovelluksessa tätä ongelmaa ei ollut, sillä ensimmäisen kerran kun Bluetooth-yhteys muodostetaan, on kaikki sovelluksen ominaisuudet käytettävissä niin pitkään kunnes Android-sovellus suljetaan. Esimerkiksi jos PWA-sovelluksessa ensimmäisellä sivulla halusi suorittaa info tulostuksen ja sen jälkeen seuraavalla sivulla näytteenoton, oli ensiksi muodostettava Bluetooth-yhteys ensimmäisellä sivulla ja vaikkakin Bluetooth-yhteys oli päällä siirtyessä seuraavalle sivulle, seuraavalla sivulla sitä ei ollut enää ns. ”olemassa” vaan yhteys piti muodostaan taas uudelleen. Osa syy tähän oli se, että ohjelmointi puolella sivut ovat eri tiedostoina ja näin ollen, kun Bluetooth-yhteys muodostetaan ensimmäisellä sivulla eri tiedostossa, ei siihen pysty enää viittaamaan seuraavalla sivulla toisessa tiedostossa vaan se on muodostettava ns. kokonaan uudelleen. Tämä johtuu siitä, että Web Bluetooth API:ssa tällä hetkellä joka kerta kun Bluetooth-yhteys halutaan, on se muodostettava alusta asti uudelleen, vaikka se olisikin jo päällä. Ongelmaan saattaa olla tulossa ratkaisu tulevaisuudessa, sillä Web Bluetooth API:ssa on tällä hetkellä testauksessa metodi nimeltä getDevices, minkä tarkoituksena olisi tulostaa lista laitteista joiden kanssa Bluetooth-yhteys on jo muodostettu, eli kun ensimmäisen kerran ensimmäisellä sivulla Bluetooth-yhteys muodostetaan, että saadaan esimerkiksi mittalaitteelta info tulostus, pitäisi sitten seuraavalla sivulla pystyä kutsumaan suoraan getDevices-metodilla jo muodostettua yhteyttä, minkä avulla ei tarvitsisi muodostella enää yhteyksiä uudestaan.

Toisena ongelmana ilmeni se, että tällä hetkellä Web Bluetooth API:ssa ei ole minkäänlaista tukea suosituimmissa selaimissa iOS:lle eli Web Bluetooth API ei toimi tällä hetkellä iOS:lla yhdelläkään suosituimmalla selaimella, joita ovat Google Chrome, Safari, Opera tai FireFox. Tämä oli ongelmista suurin, koska alun perin PWA:ta lähdettiin harkitsemaan Android-sovelluksen korvaajaksi

juuri sen takia, että siihen saataisiin mukaan iOS, Androidin lisäksi. Ongelmaan on pari kelpo ratkaisua, jotka ovat, että PWA:ta harkitaan uudelleen, jos joskus Web Bluetooth API alkaa tukemaan iOS:ia tai vaihtoehtoisesti käyttää sellaista selainta, jossa olisi tuki Web Bluetooth API:lle, mutta joka ei olisi niin laajalti käytössä, kuin edellä mainitut selaimet. App Store:sta löytyy muutama tähän tarkoitukseen tehty sovellus. Ne ovat Cider Connect ja WebBLE. Cider Connect on ilmainen ja WebBLE maksaa $1.99. [23.] [24.] Sovellusten toimivuudesta projektin yhteydessä ei osata sanoa, sillä sovelluksia ei päästy testaamaan, vaan ne olivat ennemminkin vain yksi vaihtoehto, jolla iOS:in Web Bluetooth API tuki ongelman voisi käytännössä kiertää.

6.1 Testituloksia

Vaikka selvisi, että PWA ei tällä hetkellä ole hyvä vaihtoehto korvaamaan Android-sovellusta, PWA-sovelluksella testattiin Bluetooth-yhteyden toimivuutta, luotettavuutta sekä vasteaikoja erityyppisillä datoilla. Tuloksia verrattiin Android-sovelluksella saatuihin tuloksiin. Vertailussa selvisi esimerkiksi, että PWA-sovelluksessa viimeisimmän näytteenoton tulosten haku oli noin 50% nopeampi kuin Android-sovelluksessa. Syy miksi PWA-sovellus oli noin 50% nopeampi kuin Android-sovellus jäi tuntemattomaksi. Bluetooth-yhteyden toimivuutta testattiin viimeisimmän näytteenoton tulosten haulla ja testauksessa huomattiin, että esteettömässä tilassa Bluetooth-yhteyden kantavuus oli noin 20-25 metriä, ennen kuin Bluetooth-yhteys alkoi heikentyä sen verran, että tulosten haku ei onnistunutkaan joka kerta. PWA-sovelluksella myös testattiin näytteenottoa mittalaitteen kanssa ja näytteenotto toimi kuten pitääkin. Bluetooth-yhteyden luotettavuutta testattiin siten, että molemmilla sovelluksilla, PWA:lla sekä Androidilla haettiin viimeisimmän näytteenoton tulokset ja verrattiin datoja keskenään. Android-sovelluksen ollessa se ns. virallinen sovellus, testissä verrattiin, että PWA:lla saatu data täsmää Android-sovelluksen dataan ja datat täsmäsivät keskenään.

6.2 Tavoitteet ja vaatimukset

Opinnäytetyössä tavoitteena oli selvittää, että pystyykö PWA-sovellus korvaamaan Bluetooth-pohjaiselle mittalaitteelle tehdyn Android-sovelluksen. Tavoitteeseen päästiin, sillä opinnäytetyön aikana saatiin selville, että tällä hetkellä PWA-sovellus ei sovellu korvaamaan Android-sovellusta, koska PWA-sovelluksessa käytetyssä Web Bluetooth API:ssa oli pari isoa ongelmaa.

Vaatimukset sovellukselle olivat ne, että siihen saadaan kehitettyä kaikki samat perusominaisuudet kuin Android-sovelluksessa. Vaatimuksiin päästiin, sillä PWA-sovellukseen saatiin kehitettyä kaikki perusominaisuudet, mitkä haluttiinkin. Perusominaisuuksia olivat Bluetooth-yhteyden muodostaminen mittalaitteeseen, mittalaitteelta infon tulostaminen, näytteenoton aloittaminen sekä viimeisimmän näytteenoton tulosten hakeminen.

7 Yhteenveto

Opinnäytetyön tavoitteena oli kehittää PWA-sovellus ja samalla selvittä, voiko sillä korvata jo käytössä olevan Android-sovelluksen. Tavoitteeseen päästiin, sillä sovelluksen kehityksen aikana saatiin selvyys sille, että soveltuuko PWA-sovellus korvaamaan Android-sovelluksen vaiko ei. Valitettavasti tällä kertaa PWA-sovellus ei soveltunut korvaamaan Android-sovellusta, koska sovelluksen kehityksessä käytetty Web Bluetooth API ei tukenut iOS-laitteita millään suositulla verkkoselaimella. Sovelluksen kehityksen aikana kylläkin löydettiin ratkaisu iOS-laite tuki ongelman kiertoon, mutta itse ratkaisua ei päästy testaamaa missään välissä.

Sovelluksen kehitys onnistui hyvin, vaikkakin sovelluksen kehityksen aikana ilmeni pari isompaa ongelmaa, joiden takia sovelluksen valmiiksi saanti venyi hieman. Työn alussa määritettyihin sovellus vaatimuksiin päästiin, sillä valmiissa sovelluksessa oli juuri ne ominaisuudet mitkä pitikin. Sovelluksen kehitys itsessään oli mukavaa ja haastavaa, vaikkakin välillä tuntui, että mikään ei etene mihinkään. Työharjoitteluni aikana pääsin ensimmäisen kerran kehittämään verkkosovelluksia ja tykästyin niihin hommiin ja nyt opinnäytetyön aikana tajusin, että verkkosovellus kehitys voisi olla homma, mitä ehkä haluisin tulevaisuudessa tehdä työkseni.

Lopuksi vielä iso kiitos toimeksiantajalle Drosens Oy:lle sekä kaikille työssä auttaneille henkilöille. Sovelluksen kehitykseen lähdin tietäen jonkin verran vain Vue:sta, mutta sovelluksen kehityksen jälkeen olin saanut uutta tietämystä minulle täysin uusista teknologioista, PWA:sta sekä Web Bluetooth API:sta. Samalla sain myös lisää tietämystä itse Vue:sta.

Lähteet

1. Toonen E. What is progressive web app (PWA)? Why would you want one? [Internet]. [Viitattu 25.6.2021]. Saatavilla: https://yoast.com/what-is-a-progressive-web-app-pwa/

2. ScandiPwa History of Progressive Web Apps [Internet]. [Viitattu 25.5.2021]. Saatavilla: https://medium.com/progressivewebapps/history-of-progressive-web-apps-4c912533a531

3. What is HTTPS? [Internet]. [Viitattu 24.3.2021]. Saatavilla: https://www.cloudflare.com/learning/ssl/what-is-https/

4. Service Worker API [Internet]. [Viitattu 25.5.2021]. Saatavilla: https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API

5. JSON [Internet]. [Viitattu 25.5.2021]. Saatavilla: https://developer.mozilla.org/en-US/docs/Glossary/JSON

6. Web app manifests [Internet]. [Viitattu 25.5.2021]. Saatavilla: https://developer.mozilla.org/en-US/docs/Web/Manifest

7. Lepage P, Beaufort F, Steiner T. Add a web app manifest [Internet]. [Viitattu 27.5.2021]. Saatavilla: https://web.dev/add-manifest/

8. How to make PWAs installable [Internet]. [Viitattu 27.5.2021]. Saatavilla: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Installable_PWAs

9. What is a URL? [Internet]. [Viitattu 27.5.2021]. Saatavilla: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL

10. Introduction to Service Worker [Internet]. [Viitattu 31.5.2021]. Saatavilla: https://developers.google.com/web/ilt/pwa/introduction-to-service-worker

11. Man in the Middle (MITM) Attack [Internet]. [Viitattu 31.5.2021]. Saatavilla: https://www.veracode.com/security/man-middle-attack

12. Lighthouse [Internet]. [Viitattu 31.5.2021]. Saatavilla: https://developers.google.com/web/tools/lighthouse

13. What is Vue.js? [Internet]. [Viitattu 31.5.2021]. Saatavilla: https://vuejs.org/v2/guide/index.html

14. Gupta A. What is a JavaScript Framework? [Internet]. [Viitattu 31.5.2021]. Saatavilla: https://www.tutorialspoint.com/what-is-a-javascript-framework

15. You E. FIRST WEEK OF LAUNCHING VUE.JS [Internet]. [Viitattu 31.5.2021]. Saatavilla: https://blog.evanyou.me/2014/02/11/first-week-of-launching-an-oss-project/

16. NehaMali Top 10 Most Popular JavaScript Frameworks for Web Development [Internet]. [Viitattu 31.5.2021]. Saatavilla: https://www.geeksforgeeks.org/top-10-most-popular-javascript-frameworks-for-web-development/

17. Vue.js [Internet]. [Viitattu 31.5.2021]. Saatavilla: https://en.wikipedia.org/wiki/Vue.js

18. Nguyen S. What is npm? A Node Package Manager Tutorial for Beginners [Internet]. [Viitattu 31.5.2021]. Saatavilla: https://www.freecodecamp.org/news/what-is-npm-a-node-package-manager-tutorial-for-beginners/

19. Copes F. The Vue Router [Internet]. [Viitattu 1.6.2021]. Saatavilla: https://flaviocopes.com/vue-router/

20. Vue Router Installation [Internet]. [Viitattu 1.6.2021]. Saatavilla: https://router.vuejs.org/installation.html#vue-cli

21. Beaufort F. Communicating with Bluetooth devices over JavaScript [Internet]. [Viitattu 3.6.2021]. Saatavilla: https://web.dev/bluetooth/

22. Eygi C. JavaScript Promise Tutorial: Resolve, Reject, and Chaining in JS and ES6 [Internet]. [Viitattu 3.6.2021]. Saatavilla: https://www.freecodecamp.org/news/javascript-es6-promises-for-beginners-resolve-reject-and-chaining-explained/

23. Cider Connect [Internet]. [Viitattu 3.6.2021]. Saatavilla: https://www.ciderapp.io/

24. WebBLE [Internet]. [Viitattu 3.6.2021]. Saatavilla: https://apps.apple.com/us/app/webble/id1193531073