Joonas Leonsaari

Progressiiviset verkkosovellukset teollisessa digitalisaatiossa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

2.5.2022

Tiivistelmä

Tekijä:	Joonas Leonsaari
Otsikko:	Progressiiviset verkkosovellukset teollisessa digitalisaatiossa
Sivumäärä:	37 sivua
Aika:	2.5.2022
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Mobile Solutions
Ohjaaja:	Lehtori Toni Spännäri

Insinöörityön tarkoituksena oli progressiivisten verkkosovellusteknologioiden tutkiminen, ja tavoitteena oli luoda kyvykkyys ja tietopohja käyttää näitä teknologioita hyväksi tilaajayrityksen asiakassovellusprojekteissa. Teknologiapainotteisella tutkimustyöllä saavutettiin kattava tietoperusta, jonka kautta käytännönläheinen kehitystyö mahdollistettiin. Lisäksi työssä tutkittiin progressiivisten verkkosovellusteknologioiden mahdollisia käyttökohteita ja hyötyjä teollisessa digitalisaatiossa.

Tietopohja progressiivisista verkkosovelluksista luotiin käyttäen erilaisia verkkolähteitä ja tietotekniikan alan yritysten aineistoja. Progressiivisia verkkosovellusteknologioita tutkittiin käytännönläheisesti sillä ajatuksella, että nämä teknologiat otettaisiin sovelluspohjan kehityksessä käyttöön. Progressiivisten verkkosovellusten hyötyjä ja ominaisuuksia tarkasteltiin niiden arvolupausten pohjalta tutkien mahdollisia käyttökohteita sekä hyötyjä, joita tämä teknologia tarjoaa.

Insinöörityössä selvitettiin käytännön sovelluskehityksen kautta, miten asiakassovelluspohjaa voitiin kehittää tukemaan progressiivisten verkkosovellusteknologioiden käyttöä tulevaisuuden projekteissa. Asiakassovelluspohja oli Angular JavaScript -kirjastolla luotu sovelluskehys, joka työn seurauksena kykeni progressiivisten verkkosovellusten pohjustamiseen ja tukemaan eri päätelaitteiden natiiviominaisuuksia. Työssä kehitetyn asiakassovelluspohjan ominaisuuksia ja kykyjä testattiin Lighthouse-kehitystyökalulla.

Tutkimuksen avulla luotiin uusi kyvykkyys ja tietoperusta vastata tilaajayrityksen teollisuusasiakkaiden tarpeisiin. Progressiiviset verkkosovellusteknologiat otettiin onnistuneesti käyttöön asiakassovelluspohjassa, ja erilaisten päätelaitteiden tuki kyettiin todentamaan tutkimuksen aikana. Jatkossa asiakassovelluspohjasta luodut työkalut, sovellusratkaisut ja erilaiset näkymät kyetään tutkimuksen ja toteutuksen pohjalta tarjoamaan progressiivisena verkkosovelluksena kehitettyinä.

Avainsanat: progressiivinen verkkosovellus, teollinen digitalisaatio

Abstract

Author:	Joonas Leonsaari
Title:	Progressive web applications within industrial digitalization
Number of Pages:	37 pages
Date:	2 May 2022
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Mobile Solutions
Supervisor:	Toni Spännäri, Senior Lecturer

The subject of this bachelor's thesis was to study the technologies behind progressive web applications and to establish a practical capability and a theoretical basis to use these technologies as a part of Siemens Finland's client's software projects. The technology focused research enabled establishing a comprehensive knowledge base, through which practical development work was made possible. Furthermore, the potential applications and benefits of progressive web application technologies within industrial digitalization were investigated.

The theoretical basis regarding progressive web applications was built up using various online resources and publications made by different information technology companies. Progressive web application technologies were studied in a practical way, to ensure that these technologies could be implemented within a software development application template. The benefits and features of progressive web applications were studied through their value propositions, potential use cases as well as the different applications that this technology offers.

While doing the functional section of this thesis via practical software development, it was investigated whether the application template could be developed to support progressive web application technologies in future projects. The application template was developed using an Angular Javascript framework, which as a result of this research was able to support the creation and development of progressive web applications as well as different types of end devices. The qualities and capabilities of this application template was tested using a development tool called Lighthouse.

The research and practical development created a capability and a theoretical basis to meet new requirements and needs of Siemens Finland's industrial customers. The progressive web application technologies were successfully implemented within the application template and the support for different types of end devices was verified during the process. In the future new software projects, tools and applications created using the application template can be developed as progressive web applications.

Keywords: Progressive web application, industrial digitalization

Sisällys

Lyhenteet

PWA	*Progressive Web Application*. Verkkosovellus laajennetuin ominaisuuksin.
HTTPS	*Hypertext Transfer Protocol Secure.* Protokolla, jonka ansiosta tiedon siirto on salattua palvelinten välillä.
JSON	*Javascript Object Notation*. Javascript-ohjelmointikieleen perustuva tiedostomuoto, jota käytetään tiedonvälitykseen.
API	*Application Programming Interface.* Ohjelmointirajapinta.
JS	*Javascript.* Javascript-ohjelmointikieli.
TS	*Typescript.* Typescript-ohjelmointikieli, Javascriptiin perustuva kieli.
HTML	*Hypertext Markup Language.* Merkintäkieli, joka toimii verkkosivustojen kirjoituskielenä.
CSS	*Cascasding Style Sheets.* Tyylityskieli, joka toimii verkkosivustojen tyyliohjeina.
IOT	*Internet of Things.* Järjestelmä, joka perustuu laitteiden tuottamaan tietoon, sen siirtoon ja etäseurantaan ja -ohjaukseen verkon välityksellä.
GPS	*Global Positioning System.* Satelliittipohjainen radionavigointijärjestelmä.

1 Johdanto

Insinöörityössä tutkitaan progressiivisten verkkosovellusten (progressive web applications, PWA) soveltuvuutta teollisen digitalisaation maailmaan. Työssä perehdytään progressiivisten verkkosovellusten teknologioihin, ominaisuuksiin, vaatimuksiin ja niihin hyötyihin, joita tämä teknologia tarjoaa.

Työn tilaajana toimii Siemens Osakeyhtiö. Työn tavoitteena on keventää progressiivisten verkkosovellusteknologioiden käyttöönottamista sovelluskehityksen yhteydessä. Toivottavaa on, että tutkimuksen pohjalta tulevissa sovelluspohjaisissa asiakasprojekteissa on optio käyttää PWA-teknologioita hyväksi ja täten tarjota kattavampia, useita päätelaitteita tukevia ratkaisuja tulevaisuuden digitalisaatiohankkeissa.

Osana työtä kehitetään Siemens Osakeyhtiön asiakassovelluspohjasta mobiiliyhteensopivan ja progressiivisen verkkosovelluksen kriteerit täyttävä. Tämä tarkoittaa sitä, että sovelluksella on kyvykkyys toimia alustana progressiivisten verkkosovellusten kehittämisessä ja sovelluksen käyttämisessä eri näyttökokoisilla päätelaitteilla. Kriteerien todentamiseen käytetään Lighthouse-nimistä Googlen kehittämää Chrome-kehitystyökalua. Asiakassovelluspohja on kehitetty Googlen kehittämällä Angular JavaScript -kirjastolla, joten työssä keskitytään ohjelmointiperiaatteisiin ja arkkitehtuuriratkaisuihin sekä ohjelmointikieliin, joita tämä kirjasto tukee. Progressiivisen verkkosovelluksen vaatimukset täytetään käyttäen Angularin ja selaimen tukemia työkaluja ja kehitysmenetelmiä.

Työ alkaa progressiivisten verkkosovellusten yleisellä esittelyllä, josta siirrytään yksittäisten PWA-teknologioiden tarkempaan tutkimiseen. Tästä päästään selvittämään teknologioiden hyötyjä ja mahdollisia käyttötarkoituksia. Lopussa esitellään kehitetyn asiakassovelluspohjan ratkaisut, PWA-teknologioiden käyttöönotto Angular JavaScript -kirjastolla ja yhteenveto insinöörityön päätelmistä.

2 Progressiiviset verkkosovellukset

2.1 Määritelmä

Progressiiviset verkkosovellukset, joihin jatkossa tässä insinöörityössä viitataan lyhenteellä PWA (engl. progressive web application), ovat verkon kautta tarjottavia kyvykkäitä, luotettavia ja asennettavia verkkosivuja [1]. PWA-sovellus mielletään usein vaihtoehtoiseksi ratkaisuksi natiiville mobiilisovellukselle.

Kyvykkyys

Verkkosovellusten kyvykkyys muodostuu verkon monipuolisista ohjelmointirajapinnoista [1]. Nämä ohjelmointirajapinnat mahdollistavat monien aiemmin vain natiiveissa ja alustakohtaisissa mobiilisovelluksissa olleiden eri ominaisuuksien hyödyntämisen verkkosovelluksissa. Esimerkiksi laitteen tiedostojärjestelmän, mediakontrollien ja push-ilmoitusten käyttö verkkosovelluksesta onnistuu erilaisten selaimessa olevien ohjelmointirajapintojen kautta. [2.]

Luotettavuus

Ohjelmistojen luotettavuus on määriteltävissä sillä todennäköisyydellä, millä ohjelmisto toimii vioitta määritellyn ajan määritellyssä ympäristössä [3]. Verkkosovelluksille tyypillistä on niiden toimivuus vain ympäristöissä, jossa internetyhteys on saatavilla. Tästä syystä verkkosovelluksen luotettavuus on ollut suoraan verrannollinen käyttäjän verkkoyhteyden luotettavuuteen. PWA-teknologia tarjoaa tähän ongelmaan palvelunvälittäjäteknologiaan (engl. service worker) perustuvaa ratkaisua, joka takaa sovellukselle riippumattomuuden verkkoyhteydestä ja tekee verkkosovelluksesta luotettavamman. [4.]

Asennettavuus

Asennettavuus tarkoittaa, että verkkosovellus voidaan ajaa verkkoselaimen sijasta itsenäisessä näkymässä. Asennettavuus saavutetaan PWA-sovelluksessa lisäämällä sen tietoja sisältävä, JSON-muodossa kirjoitettu

sovellusluettelotiedosto (engl. web app manifest) verkkosovellukseen. [5.] Asennuksen jälkeen sovelluksen voi käynnistää käyttäjän aloitusnäytöltä tai tehtäväpalkista, mikä luo tunteen siitä, että sovellus on osa laitetta, jolle se on asennettu [1].

2.2 Ominaisuudet

PWA-sovellukset koostuvat monista avainominaisuuksista (kuva 1). Nämä ominaisuudet tekevät tavallisista verkkosovelluksista progressiivisia ja toimivat erottavana tekijänä natiivisovelluksiin verrattuna. [6.]

Kuva 1. PWA-sovelluksen tärkeimmät ominaisuudet.

Yhtenä tärkeimpänä PWA-ominaisuutena on sovellusten kehittäminen käyttäen web-suunnittelustrategiaa nimeltä asteittainen parannus (engl. progressive enhancement). Tämä strategia mahdollistaa sen, että sovelluksen perustoiminallisuudet ja -ominaisuudet ovat saatavilla kaikille riippumatta käytössä olevasta

selaimesta tai päätelaitteesta. [7.] Tätä strategiaa tukee myös PWA-sovelluksen verkoton toimintakyky. Toisin sanoen sovellus kykenee toimimaan verkon epäluotettavassa, hitaassa tai jopa kokonaan verkottomassa tilassa. [6.]

PWA-sovellukset ovat myös helposti löydettävissä. Koska PWA on pohjimmiltaan verkkosivu progressiivisilla ominaisuuksilla, se voidaan löytää tavallisia hakukoneita käyttäen. Sovellusta ei tarvitse etsiä sovelluskaupoista, vaan sen asennus tapahtuu taustalla ensimmäistä kertaa verkkosivulla vierailtaessa. [8.]

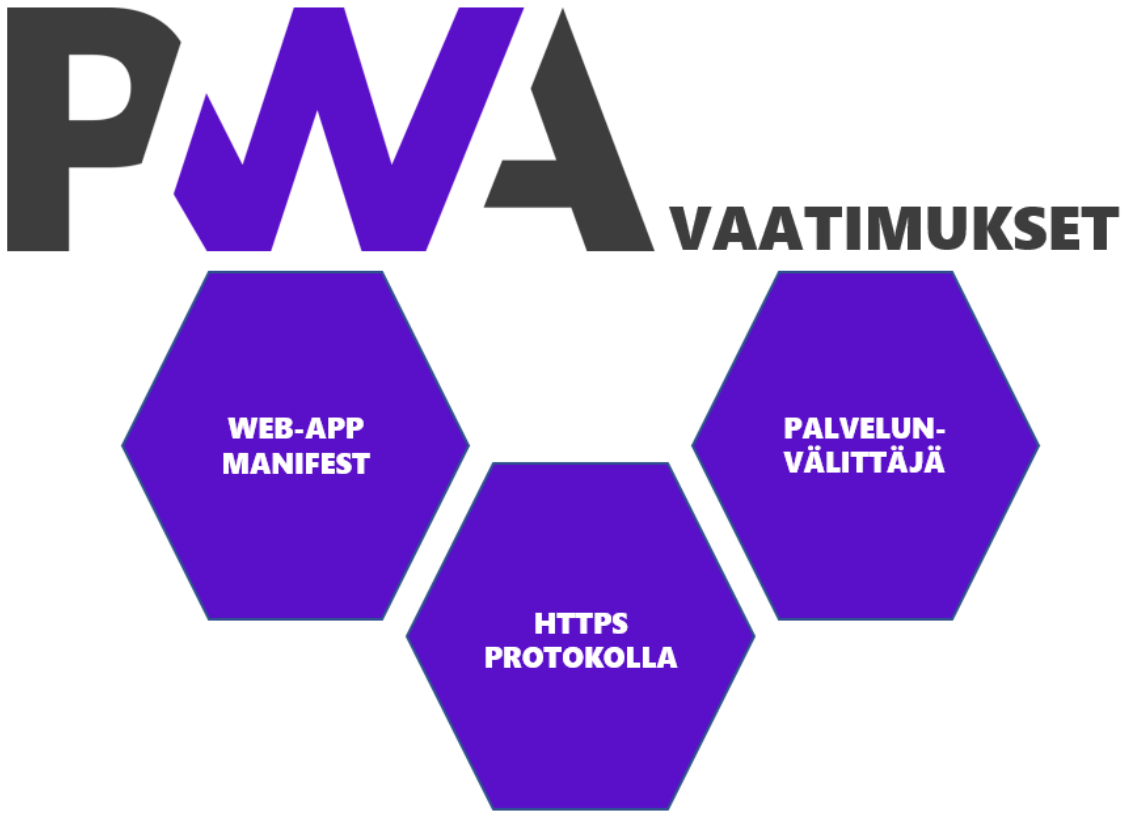
PWA-sovelluksille tyypillistä on monien, aikaisemmin vain natiiveissa ja alustakohtaisista sovelluksissa olleiden ominaisuuksien käyttäminen. Tällaisia puhelinominaisuuksia ovat esimerkiksi laitteen tiedostojärjestelmä, mediakontrollit, push-ilmoitukset, kamera, GPS, bluetooth ja sormenjälkilukija. [9.]

Uuden sovellusversion julkaisu on PWA-sovelluksilla hyvin yksinkertaista. PWA-teknologia sallii sovelluksen julkaisijoiden toteuttaa uusia ominaisuuksia ja korjauksia saumattomasti. Sovelluksen uusi versio on saatavilla käyttäjille heti, kun päivitetyt tiedostot on ladattu palvelimelle. Tämä takaa sen, että käyttäjillä on aina ajankohtainen versio sovelluksesta käytössä eikä kehittäjien tarvitse odottaa sovelluskauppojen ylläpitäjien hyväksyntää julkaistessaan uutta sovellusversiota. [8.]

Näiden avainominaisuuksien lisäksi PWA-sovellukset hyödyntävät HTTPS-protokollaa luomaan salatun ja turvallisen tavan liikuttaa dataa sovelluksen ja eri palvelinten välillä [5].

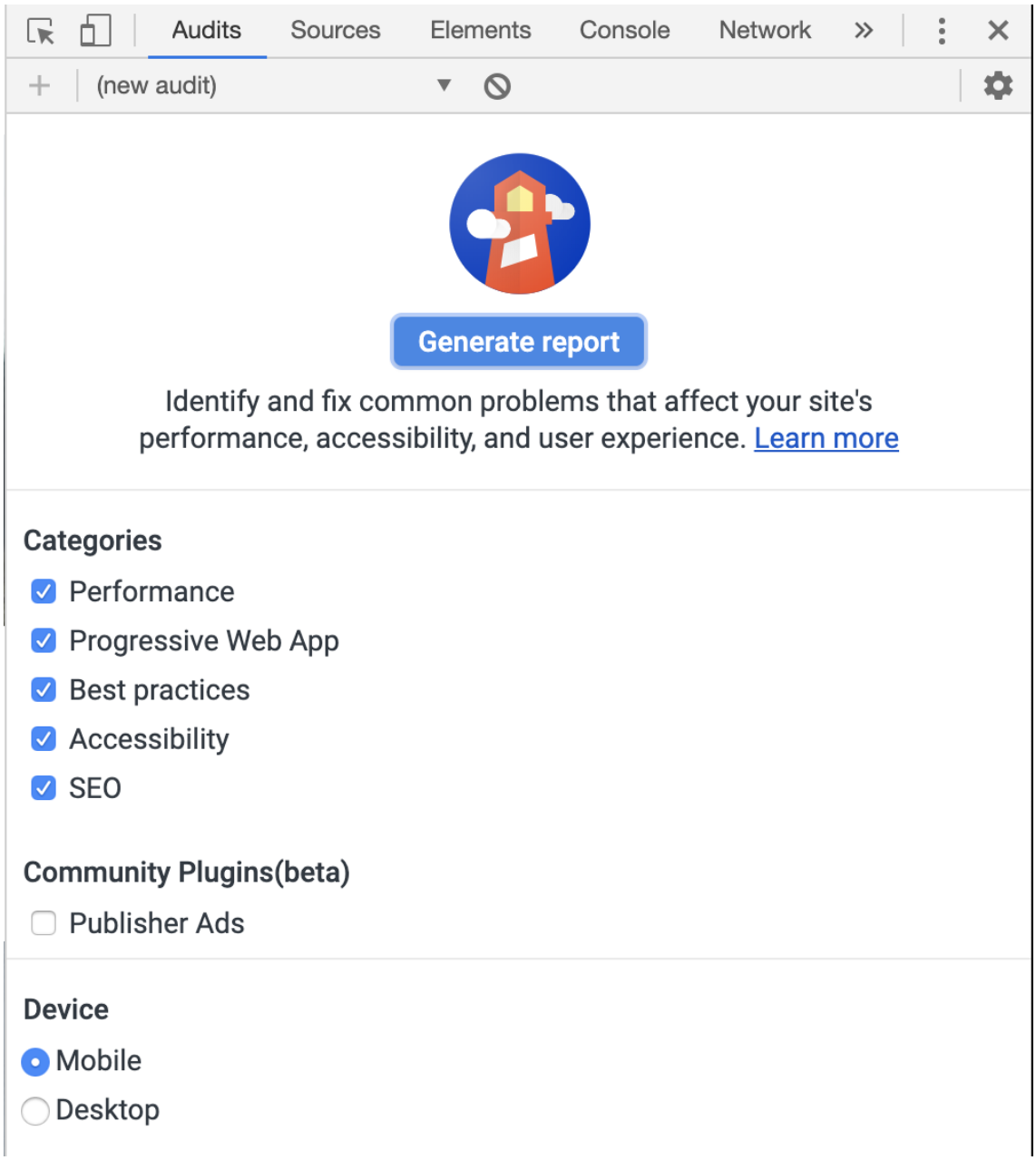
2.3 Vaatimukset

Jotta verkkosovellus voitaisiin määritellä progressiiviseksi, sen on täytettävä PWA-teknologian sille asettamat vaatimukset. Nämä vaatimukset koostuvat vähimmillään kolmesta teknisestä attribuutista (kuva 2): verkkosovelluksen luettelotiedostosta (engl. web app manifest), HTTPS-protokollan käytöstä ja palvelunvälittäjästä (engl. service worker). [5.]



Kuva 2. PWA-vaatimukset.

Verkkosovelluksen täyttäessä nämä kolme kriteeriä sitä voidaan teknisesti kutsua progressiiviseksi verkkosovellukseksi. Käytännössä sovellukselle voidaan kuitenkin asettaa useampia vaatimuksia, jotta se kykenisi tarjoamaan saumattoman käyttökokemuksen. Tässä opinnäytetyössä PWA-vaatimusten todentamiseen käytetään hyväksi Lighthouse-kehitystyökalua. Vähimmäisvaatimusten lisäksi tämä Googlen kehittämä työkalu testaa, kykeneekö sovellus latautumaan verkottomassa tilassa, onko sovellus riittävän nopea, käyttääkö sovellus turvallista tapaa siirtää dataa, tarjotaanko sovellus turvallisesta sijainnista ja onko sovellus kehitetty käyttäen saavutettavuuden hyviä käytäntöjä. [29.] Kuvassa 3 esitellään eri kategoriat, joita Lighthouse-työkalu tarkistaa testatessaan verkkosovelluksen ominaisuuksia.



Kuva 3. Lighthouse-työkalun testaamat kategoriat.

2.4 Erot verrattuna natiivisovelluksiin

Taulukossa 1 vertaillaan natiivien mobiilisovellusten avaintekijöitä PWA-sovellusten vastaaviin ominaisuuksiin. Taulukko havainnollistaa yksinkertaistettuna erot ja yhtäläisyydet sovellusten ominaisuuksien ja niiden tarjoamien hyötyjen välillä.

Taulukko 1. Erot natiivien sovellusten ja progressiivisten verkkosovellusten välillä [6].

Verrattava tekijä	Natiivi	PWA
Kuvaus	Kehitetty tietylle käyttöjärjestelmälle.	Verkkosivu, jossa natiivisovelluksen kaltainen käyttöliittymä ja ominaisuudet
Asennus	Apple App Store / Google Play Store. Asennettu suoraan laitteeseen.	Asennus ei ole pakollinen, mutta sovellus voidaan lisätä aloitusnäytölle.
Pääsy laitteen ominaisuuksiin	Täysi pääsy laitteen ominaisuuksiin, laitteistokomponentit mukaan luettuna.	Pääsy suurimpaan osaan ominaisuuksista.
Käyttö verkottomassa tilassa	Toimii verkottomassa tilassa.	Verkoton tila on saatavilla välimuistiteknologian (engl. caching) ansiosta.
Päivitykset	Päivitykset sovelluskaupan kautta käyttäjän toimesta.	Välittömät ja automatisoidut päivitykset. Ei vaadi käyttäjältä toimia.

Kehitysaika	Vaatii käyttöjärjestelmäkohtaisia kehittäjiä. Sovelluksen jakelijan hyväksyntä vaadittu julkaisun yhteydessä.	Nopeampi kehittää. Yksi sovellus soveltuu kaikille käyttöjärjestelmille. Vaatii web-kehittäjiä.
Sovellusten välinen kommunikaatio	Saatavilla	Ei saatavilla

2.5 PWA-teknologioiden hyödyt

Progressiivisten verkkosovellusten hyödyt rakentuvat opinnäytetyön luvussa 2.2 läpikäydyistä ominaisuuksista ja seuraavassa luvussa läpikäytävistä teknologioista. Niitä hyväksikäyttäen PWA kykenee tarjoamaan useita hyötyjä sovelluksen käyttäjille, kehittäjille ja liiketoiminnan harjoittajille. [10.]

Hyödyt käyttäjille

Tärkeimpänä hyötynä PWA tarjoaa ratkaisua kaikille erilaisten sovellusten käyttäjille tuttuun ongelmaan: hitauteen. PWA-sovelluksen lyhyt latausaika, nopeat päivitykset ja saumaton toimivuus huonoissakin verkko-olosuhteissa tarjoavat käyttäjälle sujuvan ja ennen kaikkea nopean käyttökokemuksen. [8.]

Nopeutta voidaan helposti mitata ajalla, joka sovelluksella kestää päästä avaamisesta pisteeseen, jossa käyttäjä kykenee kanssakäymiseen sovelluksen kanssa. Tätä aikaa kutsutaan lyhenteellä TTI (engl. time to interactive). [12.] Esimerkiksi Pinterest-kuvien jakopalvelu kykeni nopeuttamaan tätä TTI-aikaa 23 sekunnista 5,6 sekuntiin ottamalla käyttöön PWA-teknologiat verkkosovelluksessaan [11].

Nopeuden lisäksi käyttäjät hyötyvät progressiivisten verkkosovellusten pienestä sovelluskoosta. Kuten Pinterest-esimerkistä kävi ilmi, progressiivinen verkkosovellus vie tilaa puhelimesta vain 150 kilotavua, kun taas Pinterestin natiivit Android- ja iOS-sovellukset vievät vastaavasti 10 ja 56 megatavua puhelimen muistia [11]. Tämä on huomattava ero, joka korostuu erityisesti markkinoiden huokeimmissa älypuhelinmalleissa, joissa muistin määrä on rajoitettu.

Hyödyt kehittäjille

PWA-teknologia perustuu yleisiin web-standardeihin, kuten JavaScript-ohjelmointikieleen ja HTML5- ja CSS3-kieliin [8]. Hallitessaan tämän yhden verkkokehityskokonaisuuden kehittäjä kykenee kehittämään perinteisiä verkkosovelluksia ja mobiiliyhteensopivia, progressiivisia verkkosovelluksia. Kehittäjä pystyy siis tarjoamaan palvelunsa tai tuotteensa sekä tietokoneita että älypuhelimia hyödyntäville käyttäjille.

Ilman PWA-teknologiaa älypuhelimille tarjottava versio sovelluksesta toimisi vain älypuhelimen selaimen kautta, mutta täyttämällä PWA-teknologian vaatimukset voidaan sovellus ladata myös natiivisovelluksen tavoin älypuhelimen aloitusnäytölle. [13.] Mikäli kehittäjä päätyisi kehittämään erillisen natiivisovelluksen eri älypuhelinten käyttöjärjestelmille, kuten Androidille ja iOS:lle, se vaatisi se kehittäjältä asiantuntijuutta käyttöjärjestelmistä ja järjestelmille ominaisista ohjelmointikielistä [8]. Tämän lisäksi kehitettäessä käyttöjärjestelmäkohtaista sovellusta on sovellus riippuvainen siitä, että laite käyttää juuri kyseistä käyttöjärjestelmää. Progressiivinen verkkosovellus on riippumaton käyttäjän laitteesta tai käyttöjärjestelmästä, joten sitä voidaan käyttää missä tahansa laitteessa, jossa on pääsy verkkoselaimeen. [13.]

Työskennellessään verkkoympäristössä kehittäjä voi myös vapaammin valita työkalunsa eikä kehittäjän tarvitse omistaa maksullisia kehitystilejä eri käyttöjärjestelmien sovelluskauppoihin [14].

Hyödyt liiketoiminnan harjoittajille

Liiketoiminnat, joita tukee tai joiden perustana toimii sovellus, hyötyvät siitä, että potentiaaliset käyttäjät löytävät ja asentavat sovelluksen laitteelleen sekä jatkavat sovelluksen käyttöä ensimmäisen latauksen jälkeen. Helposti löydettävissä, ladattavissa ja käytettävissä oleva sovellus mahdollistaa yrityksille saumattoman kanavan vaikuttaa asiakkaisiin. [8.]

PWA-sovellus tarjoaa tähän tarpeeseen hyvän lähtökohdan. Koska progressiiviset verkkosovellukset ovat pohjimmiltaan verkkosivuja, ne voidaan löytää yleisimpiä hakukoneita, kuten Google-hakua, käyttäen. Hakualgoritmit ja hakukoneoptimointi takaavat näkyvyyttä verkkosovellukselle, eikä sovellusta tarvitse erikseen etsiä sovelluskaupoista miljoonien sovellusten joukosta, vaan sen asennus tapahtuu taustalla ensimmäistä kertaa verkkosivulla vierailtaessa. [8; 15.]

Kun käyttäjä on saavutettu, on seuraavana haasteena saada käyttäjä pysymään sovelluksessa riittävän kauan. Keskimääräinen käyttäjä muodostaa päätöksen sovelluksen käytön jatkosta ensimmäisten kymmenien sekuntien aikana [16]. Tähän PWA-teknologiat vastaavat nopeuttamalla sovelluksen sisällön latausaikaa huomattavasti [11]. Progressiivisen verkkosovelluksen sisällön latautuessa yksittäisissä sekunneissa on käyttäjällä enemmän aikaa tutustua palvelun tai tuotteen arvolupaukseen. Jos se on selkeä käyttäjä jatkaa sovelluksen käyttöä. [15.]

Tämän lisäksi sovelluskauppoihin ladattavista natiivisovelluksista kertyy automaattisesti kolmasosa kaikista sovelluksen tuotoista. Tämä osuus päätyy sovelluskaupan ylläpitäjälle. Suositut Google Play Store- ja Apple Appstore -sovelluskaupat ovat tästä hyvä esimerkki. Tähän osuuteen lasketaan kaikki tuotot sovelluksen hinnasta, sovelluksen sisällä tapahtuvista ostoista ja jatkuvista sovellustilauksista. [17.] Progressiivinen verkkosovellus toimii selaimen kautta, eikä se vaadi latausta sovelluskaupasta, mikä tarkoittaa, että mahdolliset sovelluksen

rahalliset tuotot ovat täysin sovelluksen omistajan. Näin ollen sovelluskauppojen ylläpitäjät putoavat välimiehen roolista.

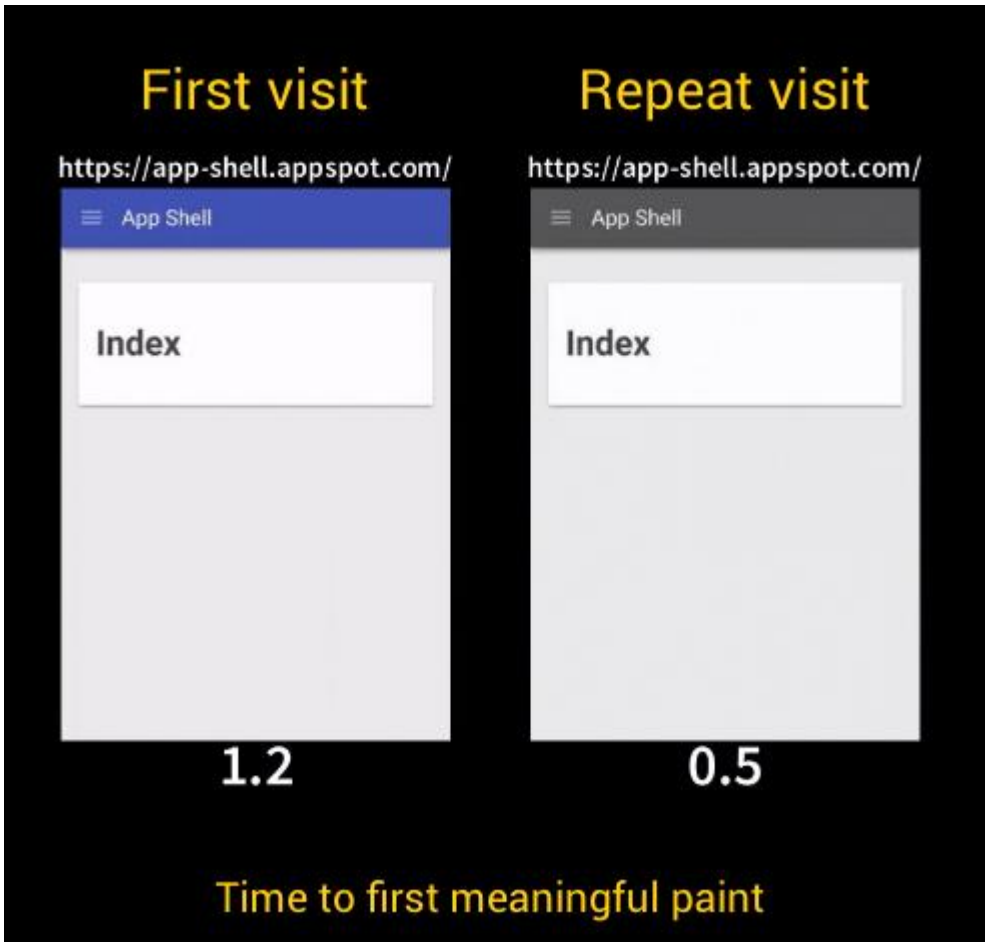
Liiketoiminnalle merkityksellistä ovat myös kehittäjille olennaiset hyödyt. PWA-sovelluksen kehittämisessä riittää, että kehittäjä hallitsee yleiset verkkokehityksessä käytettävät kielet ja työkalut, eikä kehittäjiltä tai liiketoiminnalta vaadita erityistä osaamista natiivisovellusten kehittämisestä, ylläpidosta tai käyttöjärjestelmästä. [13.] Näin ollen yritys kykenee kattamaan suuren laitevalikoiman kustannustehokkaasti ja täten saavuttamaan suuren määrän potentiaalisia käyttäjiä pienemmällä alkuinvestoinnilla.

3 PWA-teknologiat

3.1 Sovelluskuori

Sovelluskuori on PWA-sovelluksen arkkitehtuurimalli. Tässä arkkitehtuurimallissa keskeistä on ajatus siitä, että sovellus toimittaa vain vähimmäismäärän kriittisiä resursseja lataamista ja ensimmäistä avausta varten. [19.] Käytännössä siis sovellus pitää kuoren ja sovelluksen sisällön erillään, mikä mahdollistaa sovelluskuoren pienen koon ja sen tallennuksen välimuistiin (engl. cache). Tällöin sovellus voi ladata sisältöä käyttäjän sitä tarvitessa, ja sovelluksen perustoiminnalle epäkriittiset resurssit voidaan ladata taustalla käyttäjän jo ollessa vuorovaikutuksessa sovelluksen kanssa. Teoreettisesti tämä vähentää lataus- ja TTI-aikaa (engl. time to interactive) ja täten parantaa käyttäjän käsitystä sovelluksen suorituskyvystä.

Ensimmäistä kertaa PWA-sovellusta avattaessa välimuistiin tallennetaan sovelluskuoren vaatimat HTML-, CSS- ja JavaScript-tiedostot [18]. Koska sovelluskuori vaatii näistä tiedostoista hyvin pelkistetyt versiot, sovelluskuori latautuu välimuistista lähes välittömästi sovelluksen ensimmäisen käyttökerran jälkeen. Kuvassa 4 on esiteltynä Googlen kehittämän testisovelluksen latausnopeudet. Välimuistista ladatessa sovellus on käytettävissä jopa puolitetussa ajassa ensimmäiseen vierailuun verrattuna.



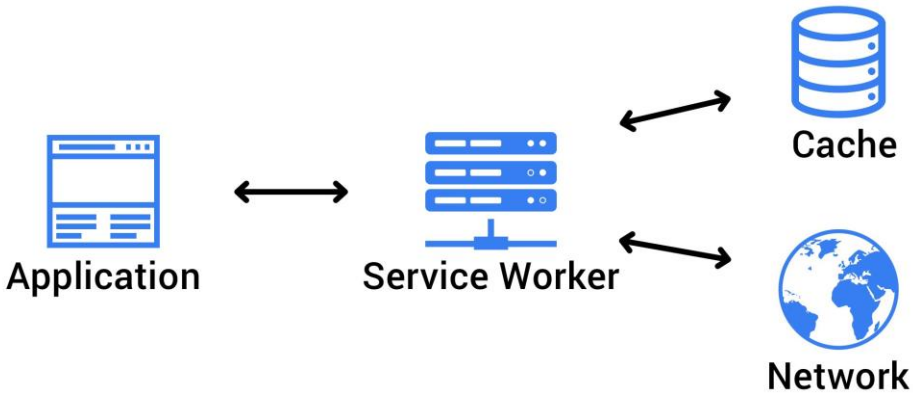
Kuva 4. Sovelluskuoren latausaika ensimmäisellä kerralla verrattuna sovelluskuoren lautaukseen välimuistista [20].

Sovelluskuoren latauduttua välimuistista voidaan sovellukselle ryhtyä lataamaan sen sisältöä. Dynaamista sisältöä voidaan hakea erilaisten ohjelmointirajapintojen kautta, ja data voidaan esittää käyttöliittymässä asynkronisesti latauksen valmistuttua. Näin käyttäjä ei esty käyttämästä sovellusta sillä aikaa, kun sovelluksen raskaampikin sisältö latautuu. [18.]

3.2 Service worker -teknologia

Service worker -teknologia perustuu palvelun välittämiseen. Pohjimmiltaan service worker on JavaScript-ohjelmointikieltä sisältävä palvelunvälittäjätiedosto.

Palvelunvälittäjä on itsenäisesti toimiva oma entiteettinsä, joka mahdollistaa verkkopyyntöjen sieppauksen sovelluksen ja internetin välillä. Palvelunvälittäjä vastaa siepattuihin verkkopyyntöihin joko verkosta tai välimuistista haetulla datalla (kuva 5). [20.] Käytännössä tämä toiminallisuus on service worker -teknologian päätoiminto, sillä se mahdollistaa sovelluksen toiminnan verkottomassa tilassa. Palvelunvälittäjän välimuistiin tallennetaan myös luvussa 3.1 kuvattu sovelluskuori. Palvelunvälittäjä tarjoaa sovelluskuoren välimuististaan, mikäli verkkoyhteyttä ei ole olemassa. Välimuistiin voidaan lisäksi tallentaa sovelluksen perustoiminnalle olennaisinta sisältöä ja resursseja, mikä vähentää verkkoyhteyden tarvetta. [18.]



Kuva 5. Service worker -teknologian toiminta palvelunvälittäjänä sovelluksen ja datan välillä.

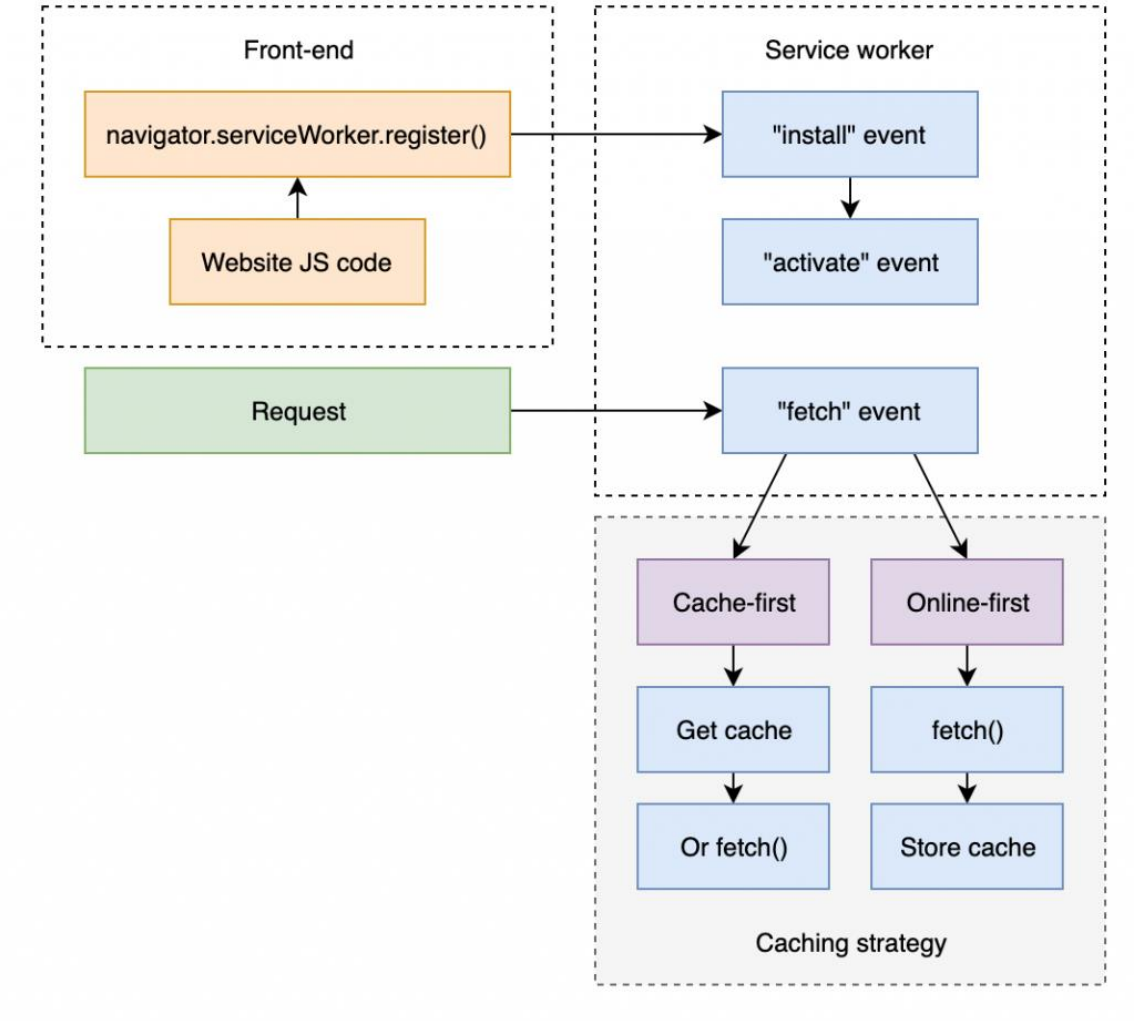
Palvelunvälittäjä toimii teknologisena perustana erilaisille ominaisuuksille, jotka tekevät web-sovelluksesta natiivin sovelluksen kaltaisen. Palvelunvälittäjä mahdollistaa muun muassa Cache storage-, Notifications-, Push-, Background Sync- ja Channel Messaging -ohjelmointirajapintojen käytön sovelluksessa. [21.] Nämä ohjelmointirajapinnat mahdollistavat insinöörityön luvussa 2.2 kuvattujen puhelinominaisuuksien käytön PWA-sovelluksessa.

Jotta palvelunvälittäjä voidaan ottaa käyttöön sovelluksessa, on sen käytävä läpi kolme eri tilaa: rekisteröinti, asennus ja aktivointi. Näistä askelista muodostuu palvelunvälittäjän elinkaari. Palvelunvälittäjän elinkaari on erillinen muun sovelluksen elinkaaresta, ja kun palvelunvälittäjä on käynyt läpi kaikki sen kolme eri tilaa, se on valmis käytettäväksi (kuva 6). [22.]

Rekisteröinti on ensimmäinen askel palvelunvälittäjän käyttöönotossa. Rekisteröinti onnistuu, jos käyttäjän selain tukee palvelunvälittäjä rajapintaa (engl. service worker API). [21.] Käytännössä kaikki modernit selaimet, Chrome, Firefox, Opera, Safari ja Edge, tukevat palvelunvälittäjiä. [23.]

Rekisteröinnissä sovellus kertoo selaimelle, mistä palvelunvälittäjätiedosto löytyy. Rekisteröinnin onnistuessa siirrytään elinkaaren toiseen kohtaan: asennukseen. Asennus on palvelunvälittäjän ensimmäinen vastaanottama tapahtuma (engl. event). Tämän tapahtuman päästyä lopputulokseen voi palvelunvälittäjä siirtyä elinkaaren viimeiseen vaiheeseen, mikäli asennus on tapahtunut ongelmitta. [22.]

Viimeisessä vaiheessa palvelunvälittäjä on valmis täyttämään toimintatarkoituksensa välittäjänä sovelluksen ja verkon välisten tapahtumakutsujen välillä. Tässä vaiheessa palvelunvälittäjä on valmiustilassa ja se on täysin toiminnallinen. Jos uusia tapahtumakutsuja ei havaita, siirtyy palvelunvälittäjä lopetettuun tilaan, josta se kykenee palaamaan takaisin valmiustilaan havaitessaan uuden verkkopyyntötapahtuman. [22.]



Kuva 6. Palvelunvälittäjän elinkaari [24].

3.3 Verkoton tila

Toimiakseen verkottomassa tilassa PWA-sovellus vaatii onnistuneesti rekisteröidyn palvelunvälittäjän. Ollessaan elinkaaren toisessa vaiheessa, asennuksessa, service worker -teknologia käyttää Cache Storage -ohjelmointirajapintaa avatakseen välimuistista tilaa PWA-sovellusta varten. [25.] Sovelluksen toiminnalle tärkeitä resursseja, kuten sovelluksen kuori, CSS-tyylitiedostot ja sovelluslogiikkaa sisältäviä Javascript-tiedostoja, tallennetaan tähän välimuistiin (esimerkkikoodi 1). Tätä kutsutaan ennaltaehkäiseväksi tiedostojen tallentamiseksi. Ennaltaehkäisevä tiedostojen tallennus mahdollistaa toistuvilla

sovellusvierailuilla sovelluksen alustamisen suoraan välimuistiin tallennettujen tiedostojen avulla. [25.]

```
const cacheName = 'pwa-cache-v1';

// Lista tiedostoista, jotka tallennetaan
const precacheResources = ['/', '/index.html',
'/css/styles.css', '/js/main.js'];

// Avaa välimuisti ja tallenna precacheResource listan tiedostot
self.addEventListened('install', (event) => {
    console.log('Service worker install event');
    event.waitUntill(caches.open(cacheName).then((cache) =>
    cache.addAll(precacheResources)));
});
```

Esimerkkikoodi 1. Resurssien tallennus palvelunvälittäjän välimuistiin.

Kun palvelunvälittäjä on saavuttanut kontrollin verkkosovelluksesta, se on aktivoitunut. Tällöin kaikki verkkosovelluksen internetyhteyden välityksellä pyytämät resurssit kulkevat palvelunvälittäjän läpi. Jokainen pyyntö aiheuttaa palvelunvälittäjässä tapahtuman, johon palvelunvälittäjä vastaa ensin välimuististaan etsimällä, pyyntöön vastaavalla resurssilla. [25.]

```
self.addEventListener('fetch', (event) => {
    console.log('Fetch intercepted for:', event.request.url);
    event.respondWith(
        caches.match(event.request).then((cachedResponse) => {
            if (cachedResponse) {
                return cachedResponse;
            }
            return fetch(event.request);
        }),
    );
});
```

Esimerkkikoodi 2. Palvelunvälittäjän toiminta uuden pyyntötapahtuman yhteydessä.

Jotta välimuistissa olisi mahdollisimman suuri osa sovelluksen tarvitsemia resursseja, tallennetaan sinne kaikkien onnistuneiden verkkopyyntöjen vastaukset [25]. Tämä takaa kattavan valikoiman resursseja suoraan välimuistista tarjottuna sovelluksen seuraavaa käyttökertaa varten, mikä mahdollistaa sovelluksen toiminnan verkottomassa tilassa.

3.4 Taustasynkronointi

Verkkopyyntöjen toiminta verkottomassa tilassa vaatii, että sovellus kykenee synkronoimaan tilansa yhdessä palvelunvälittäjän kanssa. Taustasynkronointi on oleellinen osa palvelunvälittäjäteknologiaa, joka mahdollistaa sovelluksen jatkuvan käytön verkottomassa tilassa.

Kun palvelunvälittäjä havaitsee sovelluksesta lähetetyn verkkopyynnön, se yrittää lähettää pyynnön normaalisti verkossa eteenpäin. Verkottomassa tilassa tämä pyyntö epäonnistuu ja vastuu pyynnön hallinnasta siirtyy. Palvelunvälittäjä käyttää taustasynkronointiohjelmointirajapintaa (engl. background sync API) pyynnön tallentamiseen ja lykkäämiseen, jolloin palvelunvälittäjä voi vastata pyyntöön välimuistissaan olevalla, yhteensopivalla, aiemmin tallennetulla vastauksella. Sovelluksen saavuttaessa verkkoyhteyden uudelleen saadaan pyyntöön päivitetty vastaus, ja tällöin taustalla sovelluksen tila ja välimuistiin tallennettu vastaus päivittyy verkon kautta saadun vastauksen sisältöön. Saavutettuaan päivitetyn sisällön taustasynkronointioperaatio poistaa tallennetun pyynnön eikä yritä enää lykätä pyyntöä. [26.] Taustasynkronointi siis varmistaa, että sovellus saa joka skenaariossa vastauksen pyyntöönsä ja että välimuistiin tallennettu vastaus on pyynnön uusin mahdollinen versio.

3.5 Verkkosovelluksen luettelo

Verkkosovelluksen luettelo (engl. web app manifest) on JSON-muodossa kirjoitettu tiedosto, joka määrittää, kuinka käyttöjärjestelmän tulee käsitellä PWA:ta asennettuna sovelluksena. Tämä tiedosto on yksi kolmesta vähimmäisvaatimuksesta PWA:n toiminnalle. Ilman tätä tiedostoa verkkosivu ei kykene asentumaan sovelluksena. [27.]

Sovellusluettelo sisältää selaimelle ja käyttöjärjestelmälle olennaisia tietoja verkkosovelluksesta. Sovelluksen perustiedot, kuten nimi, kuvake, väriteema ja aloituspolku, ovat luettelosta löytyvää, vaadittua sisältöä. Näiden perustietojen lisäksi tiedostoon voidaan tallentaa tarkennettuja sovellusasetuksia ja -

toiminnallisuuksia. Sovelluksen ensisijainen orientaatio, sovelluksen pikakomennot ja tarkemmat kuvaukset ovat tyypillisiä sovelluksen käytettävyyttä parantavia sovellusluettelossa olevia lisäasetuksia. [28.]

Kuvassa 7 on esiteltynä tyypillinen sovellusluettelo. Luettelossa on avain-arvo-pareina sekä vähimmäisvaatimukset että käytettävyyttä parantavia ominaisuuksia.

```
{
  "name": "ProgressiveWebApplication", //Täydellinen nimi
  "short_name": "PWA", //Lyhennetty nimi
  "start_url": ".", //Aloituspolku
  "display": "standalone", //Antaa sovellukselle natiivin käyttökokemuksen
  "background_color": "#fff", //Sovelluksen taustaväri
  "description": "A PWA test application", //Sovelluksen kuvaus
  "orientation": "portrait-primary", //Sovelluksen orientaatio
  "icons": [{ //Lista sovelluksen logoista erikokoisina
    "src": "images/touch/logo48.png",
    "sizes": "48x48",
    "type": "image/png"
  }, {
    "src": "images/touch/logo72.png",
    "sizes": "72x72",
    "type": "image/png"
  }, {
    "src": "images/touch/logo96.png",
    "sizes": "96x96",
    "type": "image/png"
  }],
  "shortcuts" : [ //Sovelluksen pikakomennot
  {
    "name": "Admin dashboard",
    "url": "/admin",
    "description": "Admin acessible only"
  },
  {
    "name": "New event",
    "url": "/create/event"
  },
  {
    "name": "New reminder",
    "url": "/create/reminder"
  }],
}
```

Kuva 7. Sovellusluettelotiedosto.

Vuonna 2022 lähes kaikki valtaselaimet, Google Chrome, Mozilla Firefox ja Apple Safari, tukevat sovellusluettelotiedostoa jollain tasolla. Googlen kehittämässä Chrome-selaimessa on kuitenkin kattavin tuki kaikille PWA-ominaisuuksille, mukaan lukien sovellusluettelon ominaisuuksille. [23.]

4 PWA-teknologia teollisessa digitalisaatiossa

4.1 Digitaalinen murros teollisuudessa

Digitaalinen murros perinteisen teollisuuden aloilla on alkanut vuosia sitten. Pilvipalveluiden, esineiden internetin, datan ja ohjelmistopohjaisten palveluiden käyttö on nykyaikaisessa modernissa teollisuudessa yleisluontoista. 2000-luvun alussa näiden teknologisten osa-alueiden kehitys kiihtyi ja samalla digitalisaatio yleistyi teollisuuden aloilla.

Vuonna 2021 Suomen teollisuuden yrityksistä 85 prosenttia hyödynsi pilvipalveluita erilaisiin käyttötarkoituksiin. Verkkoon yhdistettyjä tehdaslaitteita oli lähes puolella yrityksistä, ja niissä dataa käytettiin hyväksi muun muassa tuotantoprosesseissa, kuntoon perustuvissa huolloissa ja energian kulutuksen hallinnassa (taulukko 2). [30.]

Taulukko 2. Esineiden internetin käyttötarkoitukset teollisuudessa [30].

Käyttötarkoitus	Osuus
Energian kulutuksen hallinnassa	17 %
Toimitilojen turvallisuuden valvonnassa	38 %

Tuotantoprosessissa	17 %
Logistiikassa	9 %
Kuntoon perustuvassa huollossa	13 %
Asiakaspalveluun liittyen	3 %
Muihin tarkoituksiin	9 %

Digitalisaatio on mahdollistanut syvemmän ymmärryksen erilaisten prosessien kulusta ja laitteiden toiminnasta, ja sen kautta teollisuuden harjoittajat ovat kyenneet luomaan kokonaisvaltaisemman kuvan työtehtävistään ja niiden vaikutuksista eri osa-alueisiin. Tämän ymmärryksen ja osaamisen ehtona on työkalujen toimivuus. Erilaiset sovellusratkaisut, niistä saatu hyöty ja niiden käytettävyys ovat kehittäneet teollisuuden digitalisaatiota eteenpäin. Tämän hyödyn maksimointi on oleellinen osa-alue, johon uudet teknologiset ratkaisut, kuten progressiiviset verkkosovellukset, tarjoavat uusia perspektiivejä.

4.2 PWA-ominaisuudet teollisessa digitalisaatiossa

Progressiivinen verkkosovellus pyrkii lähtökohtaisesti tarjoamaan samat perustoiminnallisuudet ja -ominaisuudet käyttäjälle riippumatta käytössä olevasta laitteesta tai selaimesta [7]. Teollisissa laitoksissa työskentelee laaja kirjo erilaisia työntekijöitä eri prosessivaiheissa ja -tehtävissä. Tämä tarkoittaa, että teollisuuslaitosta palveleva digitaalinen ratkaisu on todennäköisesti käytössä monenlaisissa ympäristöissä ja päätelaitteissa. Sovellus voi olla käytössä toimistotyöntekijällä verkkoon liitetyn tietokoneen selaimen kautta ja samaan aikaan

tehdastiloissa toimivan operaattorin mukana taskussa löytyvästä älypuhelimesta. Tavallinen verkkosovellus soveltuu toimistotyöntekijän tarpeisiin hyvin, mutta käyttökokemus pienemmällä älypuhelimen tai tabletin näytöllä on vaikeampi toteuttaa ja käyttää. PWA:n natiivisovelluksen kaltainen käyttöliittymä mahdollistaa sovelluksen toiminnan saumattomasti myös vaativimmilla päätelaitteilla, kuten älypuhelimilla. Sovellusta ei tarvitse hakea verkkoselaimen kautta, vaan se voidaan avata suoraan älypuhelimen aloitusnäytöstä. Operaattorille sovellus toimii todennäköisesti päätyötä tukevana ratkaisuna, näkemyksiä ja yhteenvetoja tarjoavana työkaluna, jonka päätehtävänä on operaattorin tehtävien helpottaminen ja valistuneiden päätösten tukeminen. Sovelluksen nopea avaaminen, käyttö ja edellisestä istunnosta käytön jatkaminen ovat tarpeita, jotka tukevat näitä tehtäviä ja joihin PWA-teknologiat tarjoavat ratkaisuja.

Myös teollisuuslaitosten verkkoyhteydet voivat erilaisissa tiloissa ja tilanteissa olla vaihtelevia. Hitaat tai hetkellisesti jopa olemattomat verkkoyhteydet ovat yleisiä teollisuuslaitoksissa. PWA-sovelluksen kyky toimia verkottomassa tilassa on tarpeellinen ominaisuus tällaisessa tilanteessa. Rekisteröity palvelunvälittäjä varmistaa, että sovelluksen käyttöä voidaan jatkaa verkon epäluotettavissa olosuhteissa ja näin työnkulku voi jatkua esteettömästi. [6.]

PWA-sovelluksien natiiviominaisuuksista, kuten push-ilmoituksista, älypuhelimen kamerasta ja geolokaatiosta eli puhelimen maantieteellinen sijainnista, voidaan rakentaa uusia sovellusominaisuuksia teollisuuden käytössä oleviin ratkaisuihin. Näistä teknologioista voidaan digitalisaatioratkaisulle rakentaa kyvykkyys vastata yhä useampiin teollisuuden tarpeisiin. Push-ilmoitukset herättävät huomiota prosessikriittisessä tilanteessa ja voivat tuoda operaattorille ja tehdastyöntekijälle tärkeitä havaintoja esille ilman, että työntekijän tarvitsee katsoa päätelaitetta herkeämättä. Geolokaation avulla voidaan paikantaa erilaisten tehdaslaitteiden sijainti, avustaa mahdollisten huoltotoimien tekoa, opastaa työntekijä oikeaan paikkaan ja kartoittaa monimutkaisia tehdastiloja toimistotyöntekijöiden ja ulkopuolisten työntekijöiden kanssa. Kameraa taas voidaan käyttää hyväksi esimerkiksi laadun varmistamisessa tai eri prosessivaiheiden dokumentoinnissa.

PWA-teknologiat luovat mahdollisuuden näiden hyötyjen valjastamiselle, ja kun uudet digitalisaation ratkaisut kykenevät tarjoamaan käyttäjilleen enemmän hyötyä, yleistyy niiden käyttö myös osana teollista digitalisaatiota.

4.3 Pilvipalvelut osana teollista digitalisaatiota

Teollisuustuotantoon erikoistunut laitos muodostuu suuresta määrästä erilaisia koneita, laitteita ja järjestelmiä. Tämä kokonaisuus ja laitekanta tuottaa suunnattoman määrän dataa, jota voidaan hyödyntää erilaisiin käyttötarkoituksiin. Prosessitehokkuuden parantaminen on näistä käyttötarkoituksista yksi tärkeimmistä. Koneista, laitteista ja järjestelmistä kerätty prosessidata sisältää valtavan määrän yksityiskohtaista prosessitietoa, joka voidaan erilaisilla digitalisaation menetelmillä ottaa käyttöön prosessiohjauksen parantamiseksi.

Prosessidatan hyödyntäminen vaatii, että data on tallennettu ja saatavilla paikassa, josta sitä voidaan käyttää hyväksi edellä mainituilla osa-alueilla (kuva 8). Erilaiset pilvipalvelut tarjoavat tietoteknistä tallennuskapasiteettia kaikelle tälle tiedolle. Käytännössä tallennuskapasiteetti on saatavilla verkkoyhteyden välityksellä, ja pilvipalvelut kykenevät skaalautumaan tarpeen ja datanmäärän mukaan. Pilvipalveluiden avulla prosessidataa voidaan käyttää milloin ja missä tahansa, mikä tekee siitä ihanteellisen hajallaan tapahtuvien toimintojen yhteydessä. Erilaiset pilviratkaisut tukevat tiedonkeruuta reaaliaikaisella ohjauksella. Lisäksi nämä palvelut tarjoavat erilaisia ohjelmointirajapintoja datan käyttämistä varten. Nämä ominaisuudet ovat toimineet pilvipalveluiden suurimpana etuna niiden noustessa yhdeksi tärkeimmistä teollisen digitalisaation mahdollistajista. Nykypäivänä yli neljä viidestä suomalaisesta teollisuuden yrityksestä hyötyy erilaisten pilvipalveluiden tarjoamista eduista. [31.]

Kuva 8. Prosessidatan siirtyminen teollisuuslaitoksesta pilvipalveluun.

Prosessidatan hyödyntäminen PWA-sovelluksessa

Jotta prosessitieto saataisiin prosessinohjausta tukevassa PWA-sovelluksessa käyttöön, tarvitaan rajapinta, josta sovellus kykenee sitä käyttämään. Pilvipalveluiden avulla prosessidataa voidaan käyttää lähes missä ja milloin tahansa, myös PWA-sovelluksessa. Pilvipalveluiden tallennuskapasiteetin käyttö vaatii kuitenkin toimivan verkkoyhteyden, jotta prosessidata saadaan siirrettyä teollisuuslaitoksessa sijaitsevilta laitteilta palvelun omistajan palvelimelle. Prosessidatan kerääjänä toimii laitteessa kiinni oleva pieni keruulaite (engl. edge device), joka välittää dataa laitoksen paikallisen verkon ja pilvipalvelun välillä. Tavallisesti näihin laitteisiin on rakennettu kyky myös tallentaa pieniä määriä tietoa omaan puskuriin, mikäli verkkoyhteys paikallisen verkon ja pilven välillä katkeaisi. Tämä vähentää mahdollista tiedon katoamista ja parantaa sen säilyvyyttä. Vaihtelevissa laitosolosuhteissa onkin elintärkeää, että mahdolliset verkkokatkokset on otettu huomioon. [32; 33.] Prosessidatan siirryttyä ja tallennuttua pilvipalvelun palvelimille, voi PWA-sovellus hakea sitä ja käyttää hyväkseen. Mikäli prosessidataa käyttävä progressiivinen verkkosovellus menettäisi väliaikaisesti laitostiloissa verkkoyhteytensä, on hyödyllistä, että PWA-sovelluksen palvelunvälitykseen perustuva service worker -teknologia kykenee lykkäämään

mahdollisia verkkopyyntöjä ja takaamaan, että taustasynkronoinnin avulla pyyntöihin saadaan välimuistista välittömästi tallennettu vastaus ja verkkoyhteyden palautuessa saadaan verkosta pyynnöille päivitetty vastaus.

5 Asiakassovelluspohja progressiivisena verkkosovelluksena

5.1 Sovelluspohjan toteutus

Osana insinöörityötä Siemens Osakeyhtiön asiakassovelluspohja kehitettiin paremmin tukemaan teollisuusasiakkaiden digitaalisia palveluita ja projekteja. Asiakassovelluspohja on kehitetty helpottamaan uusien asiakasprojektien aloittamista luomalla sovelluskehys tärkeimmistä projekteille olennaisista komponenteista. Kehityksen aikana asiakassovelluspohjaan lisättiin mahdollisuus käyttää hyväksi PWA-teknologioita. Kun lisättiin optio käyttää PWA-teknologioita hyväksi asiakasprojekteissa, se mahdollisti kattavampien ratkaisujen ja uusien ominaisuuksien kehittämisen ja teollisen digitalisaation uusien perspektiivien tutkimisen. Tämän lisäksi sovelluspohjan komponenteista kehitettiin mobiiliyhteensopivia, eli pienille näyttöko'oille soveltuvia, jotta tuki ja toimiva käyttökokemus erilaisille päätelaitteille saavutettaisiin.

5.2 Käytetyt teknologiat

Asiakassovelluspohja on kehitetty Angular JavaScript -kirjastolla. Angular on avoimen lähdekoodin Typescript-ohjelmointikieltä käyttävä ohjelmistokehys. Google on kehittänyt Angularin dynaamisten verkkosovellusten kehittämistä varten. Angularin kehitystä johtaa Googlen Angular-kehitysryhmä. Alustus asiakassovelluspohjalle on tehty Angularin komentorivityökalulla (engl. angular command line interface). Tätä työkalua käyttäen voi alustaa, kehittää, rakentaa ja ylläpitää Angular-sovelluksia suoraan komentotulkista. [34.] Tämä Typescript-pohjainen kirjasto soveltuu PWA-sovellusten kehitystyökaluksi erinomaisesti, sillä konseptit ja teknologiat progressiivisten verkkosovellusten taustalla ovat myös Googlen kehittämiä. [8.]

Angular JavaScript -kirjaston valinta kehitystyökaluksi insinöörityössä oli luonnollinen, koska Siemens Osakeyhtiön asiakkaiden sovelluksia on kehitetty Angularilla vuodesta 2018 lähtien. Angularin alkuperäinen valinta ensisijaiseksi kehitystyökaluksi asiakasprojekteihin perustui Googlen kehitysryhmän takaamaan johdonmukaiseen tukeen ja kehitystyöhön. Asiakkaille kehitettävissä digitaalisissa ratkaisuissa sovelluksen laajeneminen ja pitkä elinkaari puolsivat valintaa, jossa kehitystyökalujen stabiliteetti ja taattu jatkuvuus ovat tärkeässä roolissa.

Asiakassovelluspohjaan on vuosien aikana kehitetty paljon valmiuksia ja komponentteja erilaisten asiakastarpeiden täyttämiseen. Osana insinöörityötä asiakassovelluspohjaan lisättiin myös tuki progressiivisten verkkosovellusten teknologioiden käyttöön ja kaikki valmiit komponentit kehitettiin tukemaan useita erilaisia päätelaitteita.

5.3 PWA-teknologioiden käyttöönotto

Jotta asiakassovelluspohja saataisiin tukemaan PWA-teknologioita, täytyy sovellukseen lisätä palvelunvälitykseen perustuva service worker -teknologia (esimerkkikoodi 3). Sen lisäys onnistuu Angularin komentorivityökalulla (engl. angular CLI). Tämä komentotulkin kautta käytettävä työkalu määrittää sovelluksen niin, että sovellus kykenee käyttämään modernissa selaimessa olevaa palvelunvälittäjärajapintaa (engl. service worker API). Tämän lisäksi komentorivityökalu lisää sovellukseen verkkosovellusluettelotiedoston (engl. web app manifest) ja ngsw-config.json-nimisen palvelunvälittäjän määritystiedoston. Verkkosovellusluettelotiedosto määrittelee, kuinka päätelaitteen käyttöjärjestelmän tulee käsitellä PWA:ta asennettuna sovelluksena. Ngsw-config.json-tiedoston kautta voidaan hallita välimuistin käyttäytymistä verkkokutsujen ja sovellusresurssien tallennuksen yhteydessä. [35.]

```
ng add @angular/pwa --project template-project
```

Esimerkkikoodi 3. Angular-komentorivityökalun komento, joka määrittää sovelluksen tukemaan PWA-teknologioita.

Oletuksena komentorivityökalun lisäämä ngsw-config.json-tiedosto on määritelty tallentamaan kaikki sovelluksen näyttämistä varten vaaditut resurssitiedostot välimuistiin. Nämä sovelluksen koonnin yhteydessä luodut resurssitiedostot koostuvat sovelluksen index.html-oletustiedostosta, paketoiduista JavaScript- ja CSS-artifaktitiedostoista ja sovelluksen kuva ja fonttitiedostoista. Näiden resurssien tallentuessa välimuistiin kykenee service worker -teknologia alustamaan sovelluksen avautumaan ja latautumaan täysin verkottomassa tilassa (esimerkkikoodi 4). [35.]

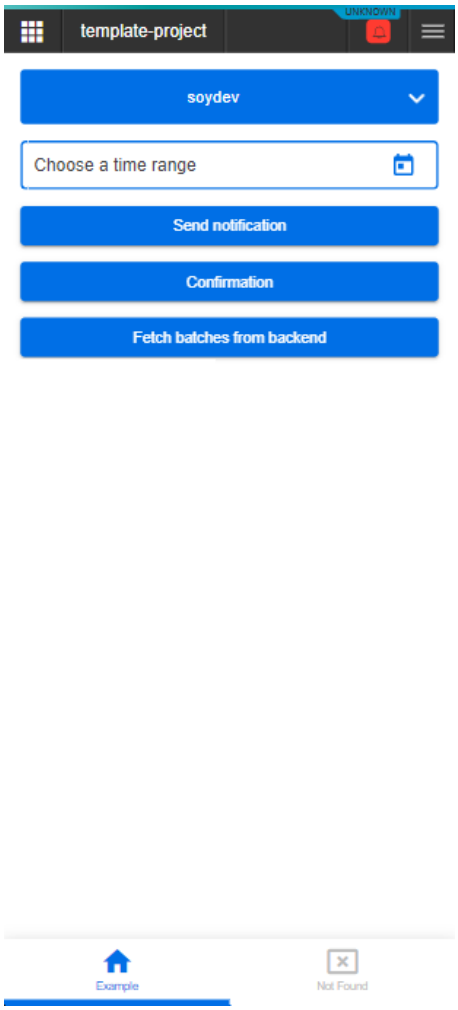
```
{
  "$schema":"../../node_modules/@angular/service-worker/con-
  fig/schema.json",
  "index": "/index.html",
  "assetGroups": [
    {
      "name": "app",
      "installMode": "prefetch",
      "resources": {
        "files": [
          "/favicon.ico",
          "/index.html",
          "/manifest.webmanifest",
          "/*.css",
          "/*.js"
        ]
      }
    }, {
      "name": "assets",
      "installMode": "lazy",
      "updateMode": "prefetch",
      "resources": {
        "files": [
          "/assets/**",
          "/*.(eot|svg|cur|jpg|png|webp|gif|otf|ttf|woff|woff2)"
        ]
      }
```

```
      }
    ]
}
```

Esimerkkikoodi 4. Komentorivityökalun lisäämä ngsw-config.json-tiedoston sisältö.

5.4 Eri päätelaitteiden tuki

Asiakassovelluspohjan käyttämät komponentit ja elementit kehitettiin myös täysin responsiivisiksi, jotta käyttäjät kykenevät hyödyntämään sovellusta älypuhelimilla, tableteilla ja muilla eri näyttökokoisilla päätelaitteilla. Tämä luo myös pohjan aikaisemmin vain natiiveissa ja alustakohtaisissa sovelluksissa olevien ominaisuuksien hyödyntämiseen. Kun sovellus tarjoaa hyvän käyttökokemuksen pienillä näytöillä, voidaan eri natiiveja mobiiliominaisuuksia, kuten tiedostojärjestelmää, mediakontrolleja, push-ilmoituksia, kameraa, GPS:ää, Bluetoothia ja sormenjälkilukijaa, käyttää asiakastarpeisiin. [9.] Responsiivisuus saavutettiin kehityksen aikana käyttäen CSS-mediakyselyitä, joiden avulla eri elementit ja komponentit kyettiin asettelemaan pienille näytöille sopivankokoisiksi (kuva 9).



Kuva 9. Responsiivinen asiakassovelluspohja älypuhelimen näyttökoossa.

Sovelluspohjaan lisättiin myös Angular service -tiedosto (esimerkkikoodi 5), jonka vastuulla on käyttöympäristön tunnistaminen.

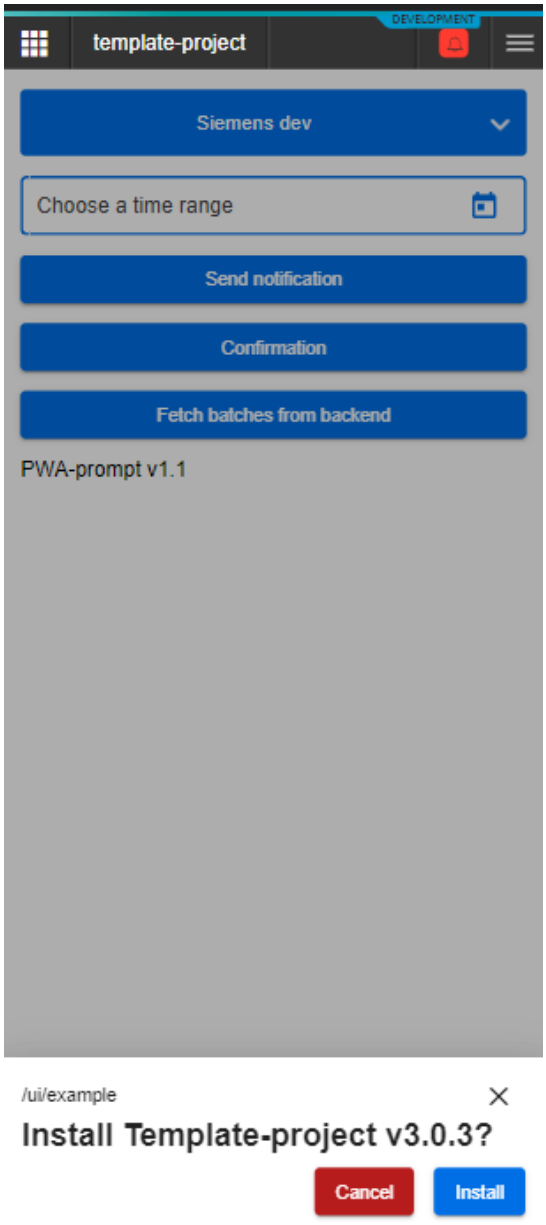
```
export class PwaService {
  private promptEvent: Event;

  constructor(private bottomSheet: MatBottomSheet, private plat-
  form: Platform){}

  public initPwaPrompt() {

    //ANDROID
    if (this.platform.ANDROID) {
```

```
      this.openPromptComponent(MobileType.Android);
          window.addEventListener('beforeinstallprompt',  (event:
          Event) => {
          event.preventDefault();
          this.promptEvent = event;
          this.openPromptComponent(MobileType.Android);
          });
    }

    //iOS
    if (this.platform.IOS) {
      const isInStandaloneMode =
      'standalone' in window.navigator &&
      window.navigator['standalone'];

      if (!isInStandaloneMode) {
        this.openPromptComponent(MobileType.IOS);
      }
    }
  }

  private openPromptComponent(mobileType: MobileType) {
    timer(3000)
      .pipe(take(1))
      .subscribe(() => (
      this.bottomSheet.open(PwaPromptComponent,
      { data: { mobileType, promptEvent: this.promptEvent } })));
   }
 }
```

Esimerkkikoodi 5. Singleton-luokkamallia käyttävä tiedosto, joka tiedottaa sovellusta käyttäjän käyttöjärjestelmästä ja avaa käyttöjärjestelmäkohtaisen asennuskehotuksen.

Tämä singleton-luokkamallia käyttävä tiedosto kykenee tiedottamaan koko sovelluksen komponenteille siitä, mitkä päätelaitteen näyttökoko ja

käyttöjärjestelmä ovat. Tämä mahdollisti myös käyttöjärjestelmästä riippuvan mukautetun asennuskokemuksen tarjoamisen (kuva 10). [36.]



Kuva 10. Android-käyttöjärjestelmän asennuskehotus.

PWA-teknologiat ja asennuskehotus yhdessä verkkosovellusluettelotiedoston kanssa mahdollistavat verkkosivun lataamisen ja asentamisen älypuhelimen

aloitusnäytölle. Tämä tukee sovelluksen älypuhelun käyttöä ilman erillistä selaimen avaamista.

5.5 Mindsphere-pilvipalvelu

Täyttääkseen erilaisia asiakastarpeita teollisessa digitalisaatiossa, PWA-sovellukseksi muutettu asiakassovelluspohja vaatii palvelun, josta se kykenee käyttämään prosessidataa hyväkseen.

Siemensin kehittämä Mindsphere-palvelu on kehitetty tunnettujen pilvipalvelutarjoajien, kuten Amazon AWS ja Microsoft Azure, päälle. Mindsphere käyttää edellä mainittuja pilvipalveluita tarjotakseen prosessi- ja anturidatalle tallennustilaa ja analysointia. [37.] Asiakassovelluspohja kehitettiin käyttämään tätä palvelua pääasiallisena asiakasdatan tarjoajana. Sovelluspohjaan lisätty service worker -teknologia kykenee välittämään verkossa tapahtuvia kutsuja Mindsphere-palvelun rajapintojen ja sovelluksen välillä ja tallentamaan kutsujen vastauksia välimuistiinsa. Tarjoamalla kutsujen vastauksissa ollutta dataa suoraan välimuistista progressiiviseksi verkkosovellukseksi kehitetty asiakassovelluspohja kykenee toimimaan verkon epäluotettavassa, hitaassa tai jopa kokonaan verkottomassa tilassa. Verkottomassa tilassa välimuistissa oleva prosessidata on kuitenkin historiadataa, mikä tarkoittaa, että täysin reaaliaikaista dataa vaativat sovellustoiminnot ja päätelmät vaativat service worker -teknologian välittämille pyynnöille päivitettyjä vastauksia, jotta välimuistissa oleva tieto olisi ajan tasalla.

5.6 Ongelmat

Vaikka PWA-teknologia tarjoaa erilaisia hyötyjä digitaalisille palveluille, on siinä vielä osa-alueita, joissa on puutteita. Apple-yhtiön kehittämä Safari-selain ei vielä täysin tue PWA-teknologioita, ja tästä syystä käyttäjät, jotka käyttävät iOS-käyttöjärjestelmän oletusselaimeksi asetettua Safari-selainta, eivät saa progressiivisten verkkosovellusten hyötyjä käyttöönsä. [23.] Googlen kehittämä Chrome-selain on kuitenkin ladattavissa lähes kaikille päätelaitteille, mikä toimii kiertotienä tämän ongelman ratkaisemiseksi. Apple on myös todennut

tulevaisuudessa kehittävänsä PWA-teknologioiden tukea eteenpäin omissa käyttöjärjestelmissään ja selaimissaan. Todennäköistä on, että lähitulevaisuudessa myös Safari-selain tukee palvelunvälittäjiä ja muita PWA-teknologioita. [38.]

Asiakassovelluspohjaa kehittäessä haasteeksi muodostui myös käyttäjän todennuksessa käytössä olevan evästeen vanheneminen tietoturvasyistä. Todennuksen vanhentuessa sovellus piti ohjata avattaessa kirjautumissivulle, jotta käyttäjän todennus voitaisiin päivittää. Tämä kirjautuminen vaati reititystä päätelaitteen verkkoselainsovellukseen ja Siemensin Mindsphere-palveluun kirjautumista. Käytännössä tämä tarkoittaa sitä, että vaikka progressiiviseksi verkkosovellukseksi kehitetty sovellus sijaitsee käyttäjän aloitusnäytöltä ja on sieltä käytettävissä, täytyy käyttäjän tietyin aikavälein päivittää sovelluksen käyttämä käyttöoikeustietue selaimen avaamisen avulla. Tämä osittain rikkoo natiivin käyttökokemuksen tunteen, joskin tämäkin ongelma on jatkokehityksen yhteydessä ratkaistavissa, esimerkiksi avaamalla upotettu selain progressiivisen verkkosovelluksen sisällä ja hoitamalla kirjautuminen sekä käyttöoikeustietueen päivitys tämän upotetun selaimen kautta.

6 Yhteenveto

Insinöörityön tavoitteena oli luoda kyvykkyys ja tietopohja käyttää PWA-teknologioita hyväksi Siemens Osakeyhtiön asiakassovellusprojekteissa. Työssä tutkittiin progressiivisia verkkosovelluksia erilaisista näkökulmista. Tutkimalla progressiivisten verkkosovellusten taustalla olevaa teknologiaa, vaatimuksia, ominaisuuksia ja eroja natiivisovelluksien kanssa, kyettiin luomaan teoreettinen ymmärrys teknologian hyödyistä digitaalisessa teollisuudessa. Perehtymällä tutkittujen teknologioiden mahdollistamiin sovellusominaisuuksiin ja valjastamalla teknologiat asiakassovelluspohjaan luotiin kehittäjille ja sovelluspohjalle kompetenssi uudenlaisten digitalisaatioratkaisuiden kehittämiseen. Tämä valmius kehittää uudenlaisia, monipuolisia ja kyvykkäitä ratkaisuja palvelee asiakastarpeita muuttamalla epävarmat olosuhteet ja erilaiset päätelaitteet seikoiksi, joista asiakassovelluspohja on riippumaton. Riippumattomuus näistä olosuhteista takaa

sovelluksen perustoiminallisuudet ja -ominaisuudet käyttäjille vaihtelevissakin olosuhteissa ja tarpeissa.

Teollisuusasiakkaiden erilaisten pöytätietokoneiden, tablettitietokoneiden, älypuhelinten ja suurien operointinäyttöjen käyttö erilaisissa olosuhteissa ja tehtävissä luo laajat ja vaativat edellytykset sovellukselle täyttää nämä tarpeet. Tämän tutkimuksen avulla luotiin uusi pohja vastata näihin tarpeisiin uudella tavalla. PWA-teknologiat otettiin onnistuneesti käyttöön asiakassovelluspohjassa, ja erilaisten päätelaitteiden tuki kyettiin todentamaan kehityksen aikana. Jatkossa asiakassovelluspohjasta luodut työkalut, sovellusratkaisut ja erilaiset näkymät kyetään tutkimuksen ja toteutuksen pohjalta tarjoamaan progressiivisena verkkosovelluksena kehitettyinä.

Jatkokehityksen yhteydessä PWA-teknologioiden mahdollistamia natiiviominaisuuksia voidaan ottaa käyttöön uusia ideoita kehitettäessä. Erityisesti push-ilmoitusten käyttäminen avaa mahdollisuuden tuoda ajankohtaisia, prosessikriittisiä havaintoja sovelluksen käyttäjien saataville. Natiiviominaisuuksien käyttöön ottaminen vaatii vielä hiukan sovelluspohjan jatkokehitystä ja tutustumista erilaisiin ohjelmointirajapintoihin. Asiakastarpeiden määrityksiä tuleekin päivittää ja tarkentaa, kun uusia asiakkuuksia aloitetaan. Asiakassovelluspohja on lähtökohtaisesti projekti, jonka kehitystä jatketaan ja jota päivitetään aina, kun uusia ominaisuuksia kehitetään eri asiakkaita palveleviin sovellusratkaisuihin.

Lähteet

1 Richard, Sam & LaPage, Pete. 2020. What are Progressive Web Apps? Verkkoaineisto. Web dev. <https://web.dev/what-are-pwas/>. Luettu 5.7.2020.

2 Introduction to web APIs. 2022. Verkkoaineisto. Mozilla. <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction>. Luettu 11.1.2022.

3 Pan, Jiantao. 1999. Software Reliability. Ohjelmistojen luotettavuustutkimus. Carnegie Mellon University. <https://users.ece.cmu.edu/~koopman/des_s99/sw_reliability/>. Luettu 10.7.2020.

4 Introduction to Service Worker. 2019. Verkkoaineisto. Google. <https://developers.google.com/web/ilt/pwa/introduction-to-service-worker>. Luettu 5.7.2020.

5 How to make PWAs installable. 2022. Verkkoaineisto. Mozilla. <https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Installable_PWAs>. Luettu 18.1.2022.

6 Progressive Web Apps: Core Features, Architecture, Pros and Cons. 2018. Verkkoaineisto. Altexsoft. <https://www.altexsoft.com/blog/engineering/progressive-web-apps>. Luettu 18.7.2020.

7 Craig, William. Progressive Enhancement 101: Overview and Best Practices. Verkkoaineisto. Webfx. <https://www.webfx.com/blog/web-design/progressive-enhancement>. Luettu 19.7.2020.

8 Rakowski, Filip & Grzybowska, Kaja & Karwatka, Piotr & Kwiecien, Aleksandra. 2020. The PWA Book. E-kirja. Divante.

9 Bar, Adam. What web can do today? Verkkoaineisto. <https://whatwebcando.today>. Luettu 20.7.2021.

10 PWA Stats. Verkkoaineisto. Cloud four. <https://www.pwastats.com/>. Luettu 5.8.2021.

11 Osmani, Addy. 2017. A Pinterest Progressive Web App Performance Case Study. Verkkoaineisto. Medium. <https://medium.com/dev-channel/a-pinterest-progressive-web-app-performance-case-study-3bd6ed2e6154>. Luettu 6.8.2021.

12 Time to Interactive. 2019. Verkkoaineisto. Web dev. <https://web.dev/interactive/>. Luettu 6.8.2021.

13 Introduction to progressive web apps. 2022. Verkkoaineisto. Mozilla. <https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Introduction>. Luettu 3.2.2022.

14 Choosing a Membership. Verkkoaineisto. Apple. <https://developer.apple.com/support/compare-memberships>. Luettu 1.9.2021.

15 Curry, David. 2022. App Store Data. Verkkoaineisto. Businessofapps. <https://www.businessofapps.com/data/app-stores/>. Luettu 3.2.2022.

16 Nielsen, Jakob. 2011. How Long Do Users Stay on Web Pages? Verkkoaineisto. Nielsen Norman Group. <https://www.nngroup.com/articles/how-long-do-users-stay-on-web-pages/>. Luettu 3.9.2021.

17 Ling, Grete. 2021. Google Play Store and Apple App Store fees, (+12 other stores). Verkkoaineisto. App Radar. <https://appradar.com/blog/google-play-apple-app-store-fees>. Luettu 5.9.2021.

18 Firtman, Maximiliano & Andrew, Rachel; Jara, Adriana & LePage, Pete. 2021. Learn PWA. Verkkoaineisto. Web dev. <https://web.dev/learn/pwa/progressive-web-apps/>. Luettu 7.11.2020.

19 Majumdar, Anurag. 2018. Progressive Web App Shell: The Key To Responsive & Engaging User Experiences. Verkkoaineisto. Medium. <https://medium.com/udacity-google-india-scholars/build-your-own-reusable-app-shell-from-scratch-7823f65e1fbd>. Luettu 7.11.2020.

20 Osmani, Addy & Gaunt, Matt. 2020. Instant Loading Web Apps with an Application Shell Architecture. Verkkoaineisto. Google. <https://developer.chrome.com/blog/app-shell/>. Luettu 7.11.2020.

21 Osmani, Addy & Gaunt, Matt. 2020. Making PWAs work offline with Service workers. Verkkoaineisto. Mozilla. <https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Offline_Service_workers>. Luettu 16.12.2020.

22 Lab: Scripting the Service Worker. 2020. Verkkoaineisto. Mozilla. <https://developers.google.com/web/ilt/pwa/lab-scripting-the-service-worker>. Luettu 16.12.2020.

23 Can I Use. Verkkoaineisto. <https://caniuse.com/>. Luettu 16.12.2021.

24 Service Worker best practices – Life cycle. 2020. Verkkoaineisto. The Code Guides. <https://thecodeguides.com/service-worker-life-cycle/>. Luettu 16.12.2021.

25 Progressive Web Apps: Going Offline. 2021. Verkkoaineisto. Google. <https://developers.google.com/codelabs/pwa-training/pwa03--going-offline>. Luettu 16.12.2021.

26 Brosset, Patrick & Hoffman, Michael. 2022. Synchronize and update a PWA in the background. Verkkoaineisto. Microsoft. <https://docs.microsoft.com/en-us/microsoft-edge/progressive-web-apps-chromium/how-to/background-syncs>. Luettu 4.4.2022.

27 Web app manifest. Verkkoaineisto. Web dev. <https://web.dev/learn/pwa/web-app-manifest/>. Luettu 4.4.2022.

28 Brosset, Patrick & Hoffman, Michael. 2022. Use a Web App Manifest to integrate a Progressive Web App into the operating system. Verkkoaineisto. Microsoft. <https://docs.microsoft.com/en-us/microsoft-edge/progressive-web-apps-chromium/how-to/web-app-manifests >. Luettu 04.04.2022.

29 Lighthouse PWA Analysis Tool. 2021. Verkkoaineisto. Google. <https://developers.google.com/web/ilt/pwa/lighthouse-pwa-analysis-tool>. Luettu 20.11.2020.

30 Esineiden internet. 2021. Verkkoaineisto. Tilastokeskus. <https://tilastokeskus.fi/til/icte/2021/icte_2021_2021-12-03_kat_005_fi.html>. Luettu 10.3.2022.

31 Liitetaulukko 3. Maksullisten pilvipalvelujen käyttötarkoitukset yrityksissä vuonna 2021. 2021. Verkkoaineisto. Tilastokeskus. <https://www.stat.fi/til/icte/2021/icte_2021_2021-12-03_tau_003_fi.html>. Luettu 10.3.2021.

32 Pettersson, Andreas. 2021. IoT data in the cloud and on the edge. Verkkoaineisto. Techtarget. <https://www.techtarget.com/iotagenda/post/IoT-data-in-the-cloud-and-on-the-edge>. Luettu 1.3.2022.

33 Bither, Bill. 2021. What is and edge device and why is it essential for IoT? Verkkoaineisto. Machinemetrics. <https://www.machinemetrics.com/blog/edge-devices>. Luettu 1.3.2022.

34 Getting started with Angular. 2022. Verkkoaineisto. Google. <https://angular.io/start>. Luettu 1.3.2022.

35 Getting started with service workers. 2022. Verkkoaineisto. Google. <https://angular.io/guide/service-worker-getting-started>. Luettu 5.3.2022.

36 Introduction to services and dependency injection. 2022. Verkkoaineisto. Google. <https://angular.io/guide/service-worker-getting-started>. Luettu 7.3.2022.

37 MindSphere. 2020. Verkkoaineisto. Siemens. <https://siemens.mindsphere.io/en>. Luettu 5.7.2020.

38 Firtman, Maximiliano. 2022. Push Notifications, WebXR, and better PWA support coming to iOS. Verkkoaineisto. <https://firt.dev/ios-15.4b>. Luettu 15.4.2022.