Samuel Ahjoniemi

Progressiiviset verkkosovellukset

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

17.12.2023

Tiivistelmä

Tekijä:	Samuel Ahjoniemi
Otsikko:	Progressiiviset verkkosovellukset
Sivumäärä:	36 sivua
Aika:	17.12.2023
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Ohjelmistotuotanto
Ohjaajat:	Lehtori, Ilpo Kuivanen

Sovellustyyppejä on paljon, ja eri tilanteet vaativat juuri niihin sopivat sovellustyypit niiden antamien etujen ja ominaisuuksien perusteella. Oikean sovellustyypin valinta on tärkeää, jotta voidaan tarjota optimaalinen käyttökokemus asiakkaille. Progressiiviset verkkosovellukset ovat suhteellisen uusi sovellustyyppi, jonka avulla voidaan luoda nopeita ja skaalautuvia verkkosovelluksia laitteesta riippumatta. Isoimpia avaintekijöitä progressiivisille verkkosovelluksille on web-alusta ja lähes natiivien sovellusten tasoinen käyttöliittymä.

Insinöörityön tavoitteena oli tutkia progressiivisten verkkosovellusten potentiaalia nykyaikana ja saada selville, tuovatko ne jotain uutta vaikuttavaa toiminnallisuutta web-kehitykselle. Insinöörityön aikana käydään läpi, mitä progressiiviset verkkosovellukset ovat, mitä niillä on tarjota ja mistä ne ovat peräisin.

Eri osapuolilla on hyvin eri näkemys siitä, mitä he haluavat sovellukselta, ja tässä opinnäytetyössä tutkitaan sitä, miten progressiiviset verkkosovellukset vaikuttavat käyttäjien, kehittäjien ja yrityksien näkökulmasta. Työssä käydään läpi, miten progressiiviset verkkosovellukset vertautuvat muihin suosittuihin sovellustyyppeihin.

Insinöörityön yhtenä tavoitteena oli muuntaa perinteinen verkkosovellus progressiiviseksi verkkosovellukseksi. Tämän ideana oli tutkia, onko se mahdollista, ja jos se on mahdollista, niin kuinka vaikeaa se on ja onko muutoksen tekeminen kannattava vaihtoehto missään tilanteessa.

Yhteenvetona saatiin tietoa progressiivisista verkkosovelluksista sovellustyyppinä, vaikutuksia monesta eri näkökulmasta ja muutettua perinteinen verkkosovellus progressiiviseksi verkkosovellukseksi onnistuneesti ja vastattiin kysymykseen, milloin ja missä tilanteessa mahdollinen muutos kannattaa tehdä.

Avainsanat: Progressiiviset verkkosovellukset, Verkkosovellukset

---

Abstract

Author:	Samuel Ahjoniemi
Title:	Progressive Web Applications
Number of Pages:	36 pages
Date:	17 December 2023
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Software Engineering
Supervisors:	Ilpo Kuivanen, Senior Lecturer

There are a lot of application types and different situations need their particular application type. Choosing the right application type is important to provide optimal user experience. Progressive web application is quite a new application type which can offer fast and scalable web experience for any device. Most noticeable features offered by progressive web applications are web as a platform and user interface that feels like native app.

The goal of the study was to research the potential of progressive web applications and figure out whether they provide something significant for web development. This thesis introduces progressive web applications and what they have to offer.

Users, developers and companies have different features they want from progressive web applications. This thesis provides comparisons between different application types and progressive web applications.

One of the goals was to create a traditional web application and convert that into a progressive web application. The idea was to discover if conversion is possible and if it was possible, would it still be worth it in any given situation.

In conclusion, information of progressive web applications as a type of application was obtained. A traditional web application was successfully converted to a progressive web application. The thesis answers the question in which situation it is worth converting an application to a progressive web application.

Keywords: Progressive Web applications, Web applications

Sisällys

Lyhenteet

URL: *Uniform Recourse Locator.* Merkkijono, jolla voidaan hakea tieto internetistä

HTML: *Hypertext Markup Language.* Merkintäkieli, jonka avulla voidaan kuvata hyperlinkkejä sisältävää tekstiä.

CSS: *Cascading Style Sheets.* Ohjelmointikieli, jolla voi tyylitellä verkkosivuja.

Ajax: *Asynchronous Javascript And XML.* Tekniikka, joka mahdollistaa verkkosivujen päivittämisen ilman, että koko sivua tarvitsee ladata uudelleen.

XML: *Extensible Markup Language*. Merkintäkieli, joka on ohjelmistosta ja laitteistosta riippumaton työkalu tietojen tallentamiseen ja siirtämiseen.

HTTPS: *Hypertext Transfer Protocol Secure* on protokolla turvalliseen tiedonsiirtoon internetissä.

JSON: *JavaScript Object Notation* on kevyt ja yksinkertainen tiedonvälitys ja tallennus formaatti.

SSL: *Secure Socket Layer* on protokolla, joka antaa mahdollisuuden laitteiden turvalliseen kommunikointiin.

TLS: *Transport Layer Security* on protokolla turvalliseen laitteiden kommunikointiin.

HTTP: *Hypertext Transfer Protocol* on protokolla tiedonsiirtoon Internetissä.

API: *Application Programming Interface* on ohjelmointirajapinta.

NFC: *Near Field Communication* on lyhyen matkan langaton laitteistotoiminnallisuus.

1 Johdanto

Verkkosovellus on jo pitkään ollut yksi suosittu tapa julkaista sovellus käyttäjien saataville. Progressiivinen verkkosovellus on uudenlainen tapa tuoda verkkosovellukseen natiivisovelluksen kaltaista käyttöliittymää. Progressiiviset verkkosovellukset herättävät mielenkiintoa, kun voidaan käyttää useita tuttuja ominaisuuksia natiivisovellusten puolelta kuten sovelluksen asennettavuus ja push-ilmoitukset.

Tässä insinöörityössä tutkitaan tarkemmin uudempaa progressiivisten verkkosovellusten konseptia verkkosovelluksen julkaisemiseen. Työssä esitellään, mistä progressiiviset verkkosovellukset ovat tulleet ja mitä uusia ominaisuuksia ne tarjoavat web-kehitykselle. Työssä tutkitaan, kuinka progressiiviset verkkosovellukset vertautuvat muihin sovellustyyppeihin, ja katsotaan, millaisia vaikutuksia progressiivisilla verkkosovelluksilla on eri osapuolien kannalta.

Insinöörityö sisältää pienen skaalan perinteisin verkkosovelluksen muuntamisen progressiiviseksi verkkosovellukseksi. Yksittäistapauksen avulla vastataan kysymykseen, minkälaisessa tilanteessa mahdollisesti muutos kannattaa tehdä.

Raportti aloitetaan progressiivisen verkkosovellusten esittelyllä. Seuraavassa luvussa tutkitaan, kuinka progressiiviset verkkosovellukset vertautuvat muihin sovellustyyppeihin. Näiden jälkeen neljännessä luvussa käydään läpi progressiivisten verkkosovellusten vaikutukset eri näkökulmista. Viidennessä luvussa käydään läpi prosessi sovelluksen muuntamisesta ja raportin lopuksi käydään läpi yhteenveto aiheesta.

2 Progressiiviset verkkosovellukset

Progressiiviset verkkosovellukset ovat verkossa olevia web-pohjaisia sovelluksia, jotka yhdistävät parhaat puolet perinteisistä web-sovelluksista ja natiiveista sovelluksista. Progressiivisia verkkosovelluksia voidaan käyttää selaimella, ja ne ovat haettavissa URL-osoitteiden avulla. Mahdollisuus, jonka URL-osoitteet

tarjoavat web-sovelluksille, on oiva tapa julkaista sovelluksia kaikille laitteille yhdestä koodikannasta. Progressiiviset verkkosovellukset erottuvat perinteisistä web-sovelluksista siten, että ne tarjoavat lähes natiivisovellustasoisen käyttökokemuksen laitteesta riippumatta. Progressiiviset verkkosovellukset tehdään käyttäen yleisiä web-sovelluskehityksen työkaluja kuten HTML:ää, CSS:ää ja JavaScriptiä. Nämä toiminnallisuudet on mahdollista saavuttaa modernien selaimien ominaisuuksia hyödyntämällä. (1.)

Progressiiviset verkkosovellukset ovat viimevuosina nostattanut suosiotaan. Suosion nousemiseen vaikuttavana tekijänä pidetään sitä, että suuret alustat ovat edistäneet progressiivisten verkkosovellusten tukea. (2.) Muiden verkkosovellusten tapaan progressiiviset verkkosovellukset ovat myös kykenevämpiä ja vaikuttavampia kuin ennen. Näiden ansiosta esimerkiksi progressiivisten verkkosovellusten latausmäärät ovat nousseet 270 prosenttia vuoden 2021 ja 2022 aikana (3). Latausmäärien nousun perusteella voidaan arvioida, että progressiiviset verkkosovellukset ovat tulleet tutummiksi käyttäjien keskuudessa. Tähän vaikuttaa se, että progressiivisten verkkosovellusten määrä on noussut ja jotkut suositut brändit ovat luoneet omat sovelluksensa progressiivisiksi.

Progressiiviset verkkosovellukset ovat hyvin toiminnassa ainakin Windows- ja Android-alustalla. Käyttöjärjestelmä iOS ei ole vielä saanut kaikkia ominaisuuksia toiminaan optimaalisella tavalla, ja se rajoittaa progressiivisia verkkosovelluksia tällä alustalla. Apple on kuitenkin pikkuhiljaa päivitellyt Safari -selainta hyödyntämään progressiivisten verkkosovellusten ominaisuuksia paremmin. (4.)

Näillä näkymin progressiiviset verkkosovellukset eivät ole katoamassa mihinkään, sillä responsiiviset verkkosivut vastaavat tällä hetkellä odotuksia, joka voidaan huomata siitä, että tukia lisätään eri alustoilla kuten iOS-käyttöjärjestelmässä. Syy tälle on se, että responsiiviset verkkosivut kuten progressiiviset verkkosovellukset ovat hyvä tapa julkaista sovellus monelle laitteelle yhdestä koodikannasta. Toinen iso syy, miksi ainakin progressiiviset verkkosovellukset pysyvät markkinoilla, on tukien lisääminen isoilta alustoilta. Isojen alustojen tukien lisääminen antaa tietoa siitä, että tätä teknologiaa halutaan tukea.

2.1 Alkuperä

Verkkosovellukset ovat alun perin lähtöisin 90-luvun alusta, jolloin Internet oli täynnä tekstitiedostoja, jotka sisälsivät HTML-sivuja. Tuohon aikaan ensimmäisten verkkosivujen toiminnallisuutena oli lähinnä linkit toisille sivuille, kun muita toiminnallisuuksia ei vielä ollut. Pian tämän jälkeen pystyi lisäämään kuvia, videoita ja ääniraitoja verkkosivuille, joiden avulla saatiin verkkosivuista eloisamman näköisiä. Näiden lisäyksien jälkeen verkkosivut olivat staattisia ja liikkumattomia ennen kuin niistä saatiin dynaamisempia, kun asiakkaan puolella (engl. client-side) pystyttiin koodaamaan JavaScriptillä vuonna 1995 (5). Dynaamisuus verkkosivulla tarkoittaa sitä, että sivu tai sivun sisältö voi muuttua esimerkiksi käyttäjän syötteen mukaan (6).

Ensimmäiset askeleet kohti modernimpaa web-sovelluskehitystä otetiin 2000-luvun alkupuolella, kun kehitettiin XMLHttpRequest-teknologia, jonka avulla pystyttiin hakemaan tietoa URL-lähteestä ilman koko sivun päivittämistä. Tämän avulla saatiin tehtyä dynaamisista verkkosivuista sulavampia, kun voidaan päivittää vain tarvittavat osat verkkosivusta koko verkkosivun päivittämisen sijaan. Tämän jälkeen saman vuosikymmenen aikana tuli AJAX-tekniikka, joka on nopeampi ja varmempi tapa hoitaa sama asia. AJAX:in avulla verkkosovellukset pystyvät lähettämään ja vastaanottamaan tietoa serveriltä asynkronisesti sovelluksen taustalla myös niin, että tiedonsiirto ei näy sovelluksessa millään tavalla ja seuraavalla interaktiolla saadaan aina päivitetyt tiedot. Tämän avulla verkkosivuista saatiin nopeampia, kun tiedonsiirto voidaan tehdä sovelluksen taustalla (7). Näiden teknologioiden avulla verkkosovelluksista tuli nopeampia ja dynaamisempia kuin ennen, mikä vaikutti positiivisesti käyttäjäkokemukseen.

Näiden web-teknologioiden jälkeen natiivit sovellukset muuttivat internetin käyttöä ja veivät markkinat etenkin mobiilipuolella. Natiivit mobiilisovellukset ovat sovelluksia, jotka on tarkoitettu erilaisille mobiililaitteille. Nämä sovellukset vetivät mobiilikäyttäjiä hyvin puoleensa, koska verkkosovellukset olivat tahmaisia ja hitaita käyttää mobiililaitteilla ja suuret tekijät kuten Google ja Apple edistivät mobiilisovellusten tekoa lisäten esimerkiksi sovelluskaupat, joista oli helppo

etsiä tarvittavia sovelluksia. Tähän samaan aikaan verkkosovellusten suunnittelu alkoi muuttua responsiivisemmaksi, eli ne alkoivat mukautumaan hyvin eri näytön kokojen ja resoluutioiden mukaan, koska huomattiin, että esimerkiksi natiiveissa mobiilisovelluksissa tämä ominaisuus paransi tehokkaasti sovelluksen käyttökokemusta (7).

Natiivien sovellusten aikana idea universaaleista sovelluksista, joissa yksi sovellus toimisi lähes kaikilla laitteilla oli pysähtynyt melkein vuosikymmeneksi. Tämän jälkeen web-sovellukset lähtivät uudelleen nousuun, kun Frances Berriman ja Alex Russell esittelivät uudenlaisen sovellustyypin, jota he kutsuivat progressiivisiksi verkkosovelluksiksi vuonna 2015. Teknologian esittely herätti mielenkiintoa, kun teknologiasta löytyi potentiaalia, jolla pystyttiin esimerkiksi tuomaan huomattavasti parempaa käyttökokemusta web-sovelluksiin kuin ennen. Teknologian yksi suuri mielenkiinnon kohde oli myös huomattavasti parempi mobiilikäyttäminen, kun mobiililaitteet yleistyivät kovaa vauhtia ja natiivit mobiilisovellukset haalivat suurta käyttäjäkuntaa. Teknologia tarjosi myös ominaisuuksia, joita on aikaisemmin ollut vain natiiveissa sovelluksissa kuten ladattavuus ja offline-käyttö. Esittelyn jälkeen teknologia jätit Google ja Microsoft lähti yhdessä edistämään progressiivisia verkkosovelluksia ja pyrki tekemään siitä standardin. (7.)

2.2 Teknologian tavoitteet ja vaatimukset

Useimmat mobiilikäyttäjät käyttävät älypuhelimia ja mobiileja internetyhteyksiä päästäkseen erilaisiin verkkosivustoihin ja mediaresursseihin. Tästä syystä verkkosivujen kehittäjät haluavat luoda sovelluksia, jotka ovat käyttäjäystävällisiä, tehokkaita ja ne pyrkivät tarjoamaan optimaalisen käyttökokemuksen. (8.)

Teknologian tavoitteena on luoda parempi käyttökokemus verkkosovelluksissa laitteesta riippumatta. Teknologia haluaa varmistaa sen, että käyttäjien on myös turvallista käyttää sovelluksia HTTPS-yhteyden avulla. Teknologia pyrkii pienentämään eroja natiivien sovelluksien ja verkkosovelluksien välillä siten, että verkkosovellukset tuntuisivat, muistuttaisivat ja tarjoaisivat samat ominaisuudet kuin

natiivit sovellukset. Mobiililaitteiden yleistyttyä web-sovellusten käyttö mobiililaitteiden kanssa oli aikaisemmin hidasta ja tahmeaa, sillä web-sovellusten optimointi ja käyttöliittymät eivät olleet niin vakuuttavia. Progressiiviset verkkosovellukset pyrkivät siis vastaamaan tähän tuomalla huomattavasti parempaa käyttäjäkokemusta.

Progressiivisilla verkkosivuilla on vaatimuksia, jotka sovelluksen on täytettävä ollakseen progressiivinen verkkosovellus. Avainominaisuudet kuten offline-käyttö tai sovelluksen asennettavuus voidaan ottaa näiden minimivaatimusten avulla käyttöön. Vaatimuksia uusille web-tekniikoille on otettu käyttöön, jotta voidaan hyödyntää selaimien uusimpia ominaisuuksia ja pitää käyttäjien turvallisuudesta huolta. (8.)

Progressiivisten verkkosovellusten minimivaatimukset ovat

- Web app manifest -tiedosto
- HTTPS-yhteys
- Service Workers.

Näiden todentamiseksi voidaan käyttää Google Lighthouse -työkalua, jonka avulla voidaan tarkistaa, löytyvätkö vaadittavat kohdat täyttämään progressiivisen verkkosovelluksen määritelmän.

Google Lighthouse on avoimen lähdekoodin työkalu, jolla voidaan tutkia verkkosivuja ja etsiä sieltä kohtia, joita voidaan parantaa. Työkalun avulla voidaan tehdä tarkastuksia suorituskykyyn, saavutettavuuteen, progressiivisen verkkosovelluksen varmentamiseen, hakukoneoptimointiin ja tehdä muita tarkastuksia. Työkalu on automaattinen, jolloin tarkastus laitetaan päälle ja tarkastuksen jälkeen näytetään, kuinka hyvin sovellus pärjäsi tarkastuksesta. Jokainen tarkastuksen kohdalta löytyy viiteasiakirja, joka selittää, miksi tarkastus on tärkeä ja miten kohta korjataan tai miten kohdan tarkastamaa kohtaa voi parantaa. (9.)

2.2.1 Web app manifest -tiedosto

Web app manifest -tiedosto on JSON-tekstitiedosto, joka pitää sisällään metatietoja web-sovelluksesta. Metatiedot määrittelevät verkkosovelluksen tietoja kuten esimerkiksi verkkosovelluksen nimen, linkit kuvakkeisiin ja URL-osoitteen. Yleisesti tiedosto sisältää kaikki tarvittavat tiedot määrittämään, miltä progressiivinen verkkosovellus näyttää laitteen aloitusnäytössä ja valikoissa, kun se asennetaan laitteeseen. Tiedosto myös pitää sisällään tiedon siitä, miten sovellus käyttäytyy, kun se avataan asennuksen jälkeen. Sovelluksen kehittäjät voivat näiden metatietojen avulla luoda käyttäjäkokemuksen, joka vertautuu lähes natiivisovelluksiin (10). Progressiiviset verkkosovellukset ovat ladattavia, joten web app manifest -tiedosto on pakollinen, jotta se saadaan käyttöön. Minimivaatimukset web app manifestille on, että se sisältää seuraavat jäsenet: (engl. memebers) name, icons, start_url ja display. Näistä tarkemmat kuvaukset löytyvät taulukosta 1, jossa esitellään tiedoston mahdollisia jäseniä.

Taulukko 1. Web app manifest -tiedoston standardisoituja JSON-objekteja, joita kutsutaan jäseniksi (engl. members) ja niiden määritelmät (10).

Jäsen	**Määritelmä**
background_color	Määrittää sovelluksen taustan värin ennen kuin sovelluksen tyylitiedosto on ladattu.
categories	Lista kategorioista johon sovellus kuuluu tai mihin sovellus sisältyy esimerkiksi kirjoihin liittyvän sivun kategorioissa voi olla ”books” kategoria.
description	Kuvaus sovelluksesta tai sovelluksen toiminasta.
display	Määrittää kehittäjän valitseman ensisijaisen näyttötilan verkkosivustolle. Tällä voidaan muuttaa, kuinka suuri osa selaimen käyttöliittymästä näytetään käyttäjälle. Tämä kuitenkin vaihtelee selaimen mukaan.
display_override	Lista näyttötavoista, joista selain valitsee ensimmäisen tuetun näyttötilan.
file_handlers	Tietotyyppi, jota sovellus hyväksyy, kun sisältöä jaetaan käyttöjärjestelmän jakamisikkunan kautta.

Jäsen	**Määritelmä**
icons	Määrittelee sovelluksen kuvakkeet kuvatiedostoina, joita voidaan käyttää eri tilanteissa kuten työpöydällä tai sovellusvalikossa.
id	Määrittelee sovelluksen uniikin tunnisteen.
lang	Määrittää sovelluksen ensisijaisen käytetyn kielen.
launch_handler	Määrittää arvot, jotka ohjaavat sovelluksen käynnistystä. Tällä hetkellä tähän on valittavissa vain yksi vaihtoehto client_mode, jossa määritellään konteksti, jossa sovellus avataan. Esimerkiksi käytetäänkö selaimen nykyistä vai uutta ikkunaa, kun sovellus avataan.
name	Sovelluksen nimi, joka näytetään selaimen välilehdellä tai käyttöjärjestelmän sovelluksen kuvakkeen vieressä.
orientation	Määrittää sovelluksen oletetun näyttösuunnan.
prefer_related_applications	Arvo, joka määrittää related_applications listalta asennettavia sovelluksia, joita suositellaan käytettävän verkkosovelluksen sijaan.
protocol_handlers	Tarjoaa mahdollisuuden listata protokollia, joita sovellus rekisteröi tai käsittelee.
related_applications	Lista, jolla määritellään verkkosovellukseen liittyviä natiiveja sovelluksia, kuten esimerkiksi natiivi Android-sovellus, joka on saatavilla sovelluskaupasta.
scope	Määrittää sovelluksen navigointialueen.
screenshots	Lista kuvankaappaus sovelluksesta. Tarkoitettu käytettäväksi progressiivisten verkkosovellusten esittelyyn esimerkiksi sovelluskaupassa.
share_target	Voidaan mahdollistaa asennettujen progressiivisten verkkosovellus rekisteröidä jaettavaksi kohteeksi järjestelmän jakovalintaikkunassa.
short_name	Sovelluksen lyhyt nimi, jota voidaan käyttää tilanteessa, kun pidemmälle name kohdalle ei ole tilaa.
shortcuts	Lista pikakuvakkeista tai linkeistä, joilla pääsee sovelluksen tärkeisiin tehtäviin tai sivuille.
start_url	Ensisijainen URL-osoite, johon siirrytään tai yhdistetään kun sovellus avataan.

Jäsen	Määritelmä
theme_color	Määrittelee sovelluksen oletusteeman värin. Vaikuttaa siihen, kuinka käyttöjärjestelmä näyttää sivuston.

Esimerkkikoodi 1 on koodirivi, mikä pitää sisällyttää verkkosovelluksen index.html-tiedostoon pääelementin (engl. head) alle, jotta web app manifest saadaan käyttöön.

```
<link rel="manifest" href="/manifest.json">
```

Esimerkkikoodi 1. HTML-elementti, joka ottaa web app -manifestin käyttöön verkkosovelluksessa.

Esimerkkikoodi 2 on esimerkki siitä, miltä esimerkiksi web app manifest voi näyttää verkkosovelluksessa.

```
{
  "name": "Application name",
  "description": "Description for application",
  "icons": {
    "src": "favicon.ico",
    "sizes": "64x64 32x32 24x24 16x16",
    "type": "image/x-icon"
  },
  "start_url": "/",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff"
}
```

Esimerkkikoodi 2. Web app manifest on JSON-objekti, jossa on määritetty järjestyksessä, sovelluksen nimi, sovelluksen kuvaus, käytettävien kuvakkeiden objektit, aloitusosoite, sovelluksen ensisijainen näyttötila, sovelluksen oletusteema ja sovelluksen taustan väri ennen tyylitiedoston latausta.

2.2.2 HTTPS-yhteys

HTTPS-yhteys on verkkosovelluksissa laajasti käytetty protokolla, joka tuo tiedonsiirtoon lisää turvallisuutta käyttäjän ja sovelluksen välille. HTTPS-yhteyden saamiseen verkkosivu vaatii SSL- tai TLS-sertifikaatin asentamisen. HTTPS-protokolla yksinkertaisesti salaa yhteyden, joka auttaa estämään tahallisia hyökkäyksiä sovelluksen käyttäjiä vastaan. HTTPS-yhteyden tuoma salaus pitää siis

huolta myös siitä, että henkilökohtaiset yksityiset tiedot pysyvät turvassa molemmille osapuolille. HTTPS-yhteys on molemmille käyttäjille ja verkkosovelluksen omistajille turvallisempi kuin tätä edeltävä HTTP-protokolla. Yhteyden salaamisen jälkeen yleiset huijaustavat kuten arkaluontoisten tietojen kalastelu ja tahalliset kolmannen osapuolen mainosten tuomat tietoturva-aukot voidaan eliminoida minimiin. (11.)

HTTPS-yhteyttä alun perin käytettiin vain tilanteissa, joissa turvallisuus oli erityisen tärkeää kuten sähköpostit, verkkopankkien ja verkkokaupoissa. HTTPS-yhteyteen vaativan sertifikaatin saaminen oli ennen huomattavasti vaikeampaa ja työläämpää, mikä vaikutti suuresti HTTPS-yhteyden suosioon. Toinen syy, mikä on vaikuttanut HTTPS-yhteyden yleistymiseen, on sen vaikutus verkkosivun nopeuteen. Nopeuden menetykset HTTPS-yhteyden kanssa ovat kuitenkin minimaalisen pieniä ja yleensä valitaan HTTPS-yhteys sen tarjoaman turvallisuuden vuoksi.

Nykyään HTTPS-yhteys on kaikille verkkosivuille saatavissa sertifikaatin avulla ja monet palvelinten ylläpitäjät tarjoavat sen ilmaisena palveluna verkkosovelluksen julkaisun yhteydessä. Vuonna 2018 esimerkiksi Googlen Chrome -selain aloitti merkitsemään HTTP-yhteydellä sivun ei turvalliseksi (12). Aikaisemmin vuonna 2014 Google ilmoitti, että HTTPS-yhteys tulee yhdeksi arvosteltavaksi kohteeksi, joka vaikuttaa hakukone tuloksiin (13). HTTPS-yhteys antaa siis pienen edun hakukoneoptimointiin. Samaan aikaan vuonna 2014 julkaistiin Let’s Encrypt -projekti julkisesti kaikkien käyttöön. Projektin tavoitteena oli saada perus HTTPS-sertifikaatti helpommin saataville. (14.)

HTTPS-yhteys on vaatimus uusille selaimien ominaisuuksille ja tarvittu etenkin progressiivisten verkkosovellusten kanssa. HTTPS-yhteys mahdollistaa ominaisuudet kuten service workerit, jotka mahdollistavat push-ilmoitukset. Nämä ovat avainominaisuuksia progressiivisille verkkosovelluksille ja vaativat HTTPS-yhteyttä, koska esimerkiksi haitallinen hyökkäys voi olla vielä vaarallisempi, kun nämä API:t tarjoavat laajat mahdollisuudet toiminnallisuuksille. (15.)

2.2.3 Service Workers API

Service Workers API toimii välityspalvelimena verkkosovelluksen, selaimen ja saatavilla olevan verkkoyhteyden välissä. Service worker on siis skripti, joka pyörii erillään verkkosovelluksesta sen taustalla. Toiminnallisuus vaatii selaimen olon päällä. Sovelluksen asennettavuus tarvitsee service workerin toimintaa. (15.)

Niiden yhtenä tarkoituksena on ottaa käyttöön tehokas offline-käyttö ja tehdä tarvittavat toimenpiteet verkon saatavuuden perusteella sekä yrittää hakea palvelimilla olevaa dataa. Offline-tilan saavuttamiseksi service worker voi hakea tietoa palvelimelta ja pistää tarvittavia resursseja välimuistiin ja tarjota näistä resursseista tietoja tarvittaessa ilman verkkoyhteyttä. Tämä voi myös nopeuttaa sovellusta, jos ei tietoa tarvitsekaan hakea palvelimelta enää uudestaan (15).

Tämä API antaa myös mahdollisuuden ottaa push-ilmoitukset käyttöön, koska service worker pyörii myös silloin, kun sovellus ei ole aktiivisena päällä. Toiminnallisuus kuitenkin vaatii selaimen käynnissä olemisen. Jotta sovellus voi saada push-ilmoituksia, niin sovelluksella pitää olla aktiivinen service worker. Sovelluksen pitää ottaa myös käyttöönsä Push API, jotta ilmoituksia voidaan lähettää ja käsitellä. Kun service worker on aktiivinen, se voi tilata push-ilmoituksia Push API:n avulla. Service worker käynnistetään tarpeen mukaan käsittelemään saapuvia push-ilmoituksia. Jos service worker saa tiedon ilmoituksesta, niin sen pitää ilmoittaa tästä sovellukselle, jotta sovellus voi näyttää push-ilmoituksen. Push-notifikaatio-ominaisuuden käyttöönotto voi lisätä resurssien käyttöä, joka näkyy etenkin akun kulutuksessa mobiililaitteilla.

Push-ilmoitukset ovat hyvä tapa pitää käyttäjät ajan tasalla uusista muutoksista tai tapahtumista. Jos sovellus ilmoittaa tärkeistä muutoksesta käyttäjälle ilmoituksen avulla, niin käyttäjän ei tarvitse itse aktiivisesti etsiä sovelluksesta tietoa. Ilmoitus, joka yhtäkkiä ilmestyy käyttäjän eteen, saa vedettyä käyttäjän huomion hetkellisesti ja ilmoitukset toimivat hyvänä markkinointikeinona. Ilmoituksen avulla voidaan saada käyttäjä palaamaan takaisin sovellukseen. (16.)

Service workerin tarjoamien toiminnallisuuksien avulla voidaan luoda käyttäjille parempi käyttökokemus ja kokemus muistuttamaan natiivia sovellusta.

2.3 Tyypilliset ominaisuudet

Progressiiviset verkkosovellukset sisältävät samat ominaisuudet kuin perinteiset verkkosovellukset. Samalla lisätään tuttuja natiivien sovellusten ominaisuuksia. Ne ovat tyypillisiä ominaisuuksia, jotka yleensä löytyvät progressiivisista verkkosovelluksista. Näistä jokaista ominaisuutta ei sovelluksen tarvitse toteuttaa ollakseen progressiivinen verkkosovellus.

Offline-käyttäminen on progressiivisten verkkosovellusten ominaisuus, joka tarjoaa sovelluksen toimivuuden osittain tai kokonaan ilman internetyhteyttä, mikä tekee niistä erittäin käteviä heikkojen yhteyksien tai offline-tilojen aikana. Tämä voidaan saavuttaa käyttämällä progressiivisille verkkosovelluksille tuttua Service Worker -teknologiaa, joka tallentaa sovelluksen toimivuuden kannalta kriittisiä tietoja välimuistiin. Käyttäjän näkökulmasta tällä voidaan myös pienemmän datan kulutuksen ansiosta säästää verkkoa.

Progressiiviset verkkosovellukset ovat hakukoneoptimoituja, mikä tarkoittaa sitä, että progressiiviset verkkosovellukset on suunniteltu olemaan paremmin haettavissa ja löydettävissä. Tämä saavutetaan niin, että jokainen progressiivinen verkkosovellus sisältää globaaleja web-sovellusstandardeja kuten turvallisempi HTTPS-tuki. Progressiiviset verkkosovellukset on tehty käyttäen HTTPS-standardia, joka tuo lisää turvallisuutta tiedonsiirron aikana (17). Perinteisten web-sovellusten tavoin myös progressiiviset verkkosovellukset ovat linkattavia URL-osoitteiden avulla.

Progressiiviset verkkosovellukset ovat asennettavia, joka tarkoittaa sitä, että progressiivisesta verkkosovelluksesta on mahdollista luoda kuvake, jota painamalla päästään suoraan käytettävään sovellukseen ilman selaimen hakuprosessia. Tämä ominaisuus saa progressiiviset verkkosovellukset tuntumaan samalta kuin natiivit sovellukset.

Progressiiviset verkkosovellukset ovat responsiivisia, sillä ne mukautuvat eri laitteiden ja näyttökoon mukaan ja tuovat samalla saumatonta käyttökokemusta sovelluksen sisällä. Progressiiviset verkkosovellukset ovat myös mobiilioptimoituja, joten niitä on mukavampi käyttää mobiililaitteilla.

2.4 Hyvät puolet ja huonot puolet

Kuten kaikilla erilaisilla sovellustyypeillä niin tälläkin on omat hyvät ja huonot puolensa. Edellisessä osassa käytiin läpi tyypillisiä ominaisuuksia, jotka voidaan laskea hyviksi puoliksi. Tässä osassa keskitytään progressiivisten verkkosovellusten hyviin ja huonoihin puoliin.

Saavutettavuus verkon välityksellä on iso plussa, koska sovelluksen avaamiselle heti ensimmäisestä kerrasta asti on asetettu pieni kynnys, kun ei tarvita erillistä asennusvaihetta vaan voidaan suoraan yhdistää URL-osoitteen kautta käyttövalmiiseen sovellukseen. Web-sovelluksilla on mahdollisuus olla etsittävissä selainten hakukoneissa, joka lisää sovelluksen löydettävyyttä. Progressiiviset verkkosovellukset pitävät sisällään paljon kohtia, joita hakukoneet priorisoivat. Näitä kohtia ovat esimerkiksi HTTPS-yhteys ja mobiilioptimointi. Hakukoneoptimointi ei kuitenkaan ole yksiselitteistä, koska Microsoftin hakukone Bing listaa sovellukset eri tavalla kuin Googlen hakukone. Sovellusten jakelu on helppoa, kun se voidaan jakaa yhdestä samasta koodikannasta. (17.)

Progressiiviset verkkosovellukset ovat yleisemmin verkosta haettavia sovelluksia, jotka ovat asennettavissa käyttäjän kotiruudulle tai työpöydälle. Responsiivisten verkkosovellusten huonona puolena on pidetty sitä, että niitä ei voida julkaista sovelluskauppoihin. Progressiivisia verkkosovelluksia voidaan julkaista sovelluskauppoihin, mutta se vaatii sovelluksen pakkausta, jonka voi esimerkiksi suorittaa siihen tarkoitetun sovelluksen avulla. (18.)

Vaikka progressiivisiin verkkosovelluksiin voidaan yhdistää URL-osoitteella selaimen avulla, niin nämä sovellukset kuitenkin tarjoavat mobiililaitteilla hyvin lähelle natiivisovellustason käyttäjäkokemusta. Progressiiviset verkkosovellukset

tarjoavat myös natiivisovellusominaisuuksia mobiililaitteilla kuten esimerkiksi ilmoitukset. Verkkosovellusten hyvänä puolena pidetään sitä, että ne ovat itsenäisiä sovelluksia eikä niiden tarvitse olla missään yhteydessä erilaisten sovelluskauppojen kanssa. Progressiiviset verkkosovellukset antavat käyttäjälle mahdollisuuden valita, kuinka käyttää sovellusta.

Päivitettävyys on web-sovellusten hyvä puoli ja huono puoli, kun voidaan olla varmoja siitä, että käyttäjille on aina uusin sovellusversio ja data käytössä. Natiivien sovellusten kanssa päivitysprosessi on manuaalinen. Verkkosovellusten tapauksessa päivitys prosessi on käyttäjän näkökulmasta automaattinen. Tällä päästään myös pois eri sovelluskauppojen tarkistuksista ja hyväksymisprosesseista, jolloin päivitykset saadaan nopeammin käyttäjien käsiin. Toisaalta automaattisten päivitysten huonona puolena pidetään sitä, että jos jokin muutos sovelluksessa ei miellytä osaa käyttäjistä esimerkiksi siten, että sovelluksesta poistetaan joitain ominaisuuksia tai sovellus menee rikki. Verkkosovellusten automaattiset päivitykset eivät tarjoa tapaa käyttää aikaisempaa sovelluksen versiota.

Turvallisuudesta on myös pidetty huolta samalla tavalla kuin perinteisissä websovelluksissa, kun progressiiviset verkkosovellukset voivat myös hyödyntää HTTPS-yhteyksiä ja muita turvallisuusstandardeja, mikä tekee niistä turvallisia käyttää. Progressiiviset verkkosovellukset vaativat yleensä vähemmän rajoitettuja oikeuksia laitteilta, joka yleensä vähentää näkyviä turvallisuusuhkia. Tämän avulla pystytään takaamaan turvallisempi käyttökokemus käyttäjille (19).

Vaikka web on monilla laitteilla hyvin käytettävissä, niin eivät progressiiviset sadan prosentin toimintatakuuta voi taata, kun esimerkiksi vanhemmat selaimien versiot eivät välttämättä tue uusimpia ominaisuuksia niin, että sovellukset toimivat oikein. Jokaisen käyttöjärjestelmän täyttä tukea ei ole, joka voi vaikuttaa negatiivisesti käyttäjiä, jotka eivät pysty hyödyntämään sovelluksen kaikkia ominaisuuksia. (20.)

Progressiivisilla verkkosovelluksilla on oma paikkansa sovellustyyppinä eikä se voi tarjota kaikkia mahdollisia ominaisuuksia, joita esimerkiksi natiivit sovellukset voivat tarjota. Selaimet rajoittavat joidenkin ominaisuuksien lisäämistä verkkosovelluksiin, joten progressiiviset verkkosovellukset eivät ole aina paras valinta sovellustyypiksi.

Verkkosovelluksia käytetään selaimen avulla, joka tarkoittaa sitä, että selain on sovelluksen ja käyttöjärjestelmän välissä vaatimuksena sovelluksen toiminnalle. Tämä vaatimus on rajoittava tekijä verkkosovelluksille, koska selaimessa mahdollisesti tapahtuva virhe vaikuttaa sovelluksen toimintaan negatiivisesti.

3 Vertailut muihin samankaltaisiin sovellustyyppeihin

Tässä osiossa verrataan progressiivisia verkkosovelluksia muihin sovellustyyppeihin, joita on käytetty sovellusten tarjonnassa. On tärkeää valita oikeanlainen sovellustyyppi sen mukaan, minkälainen sovellus on tarkoitus tehdä. Valintaan vaikuttaa se, että minkälaisen käyttöön tuleva sovellus menee, jotta voidaan varmistaa optimaalinen käyttökokemus.

3.1 Perinteiset web-sovellukset

Perinteiset web-sovellukset ovat toimineet hyvin, mutta mobiililaitteiden yleistymisen aikana on huomattu, että perinteiset web-sovellukset voivat olla todella hitaita ja tahmaisia käyttää. Progressiiviset verkkosovellukset ovat vastanneet perinteisten verkkosovellusten huonoihin kohtiin ja parantanut niitä onnistuneesti. Progressiivisten verkkosovellusten lisäämät ominaisuudet antavat mahdollisuuden luoda verkkosovelluksista vähemmän verkkoa käyttäviä, nopeampia ja natiivisovellustasoisen käyttöliittymän.

Perinteisiin verkkosovelluksiin verrattuna turvallisen tiedonsiirron takaaminen progressiivisissa verkkosovelluksissa on parempi, koska progressiiviset verkkosovellukset vaativat HTTPS-yhteyden, kun taas perinteisille verkkosovelluksille se on vapaavalintaista mutta suotuista.

Progressiiviset verkkosovellukset tarjoavat paremman hakukone optimoinnin, jonka avulla niiden näkyvyys on parempi. Hakukoneet järjestelevät sovellusten listausjärjestyksen. Listausjärjestykseen vaikuttaa sovelluksen tarjoamat ominaisuudet kuten turvallisen tiedonsiirron HTTPS-yhteys ja mahdollisimman monelle käyttäjälle paremman käyttökokemuksen sisältämä mobiilioptimointi.

Tilanteita, joissa kannattaisi valita perinteinen verkkosovellus progressiivisen verkkosovelluksen sijaan ei pahemmin ole, koska progressiivinen verkkosovellus toimii samalla tavalla kuin perinteinen verkkosovellus mutta tuo samalla paremman käyttökokemuksen mobiilikäyttäjille ja tuo paljon uusia hyödynnettäviä ominaisuuksia. Tästä herääkin kysymys, miksi kaikki verkkosovellukset eivät ole progressiivisia verkkosovelluksia. Vastaus kysymykseen on, että verkkosovelluksia on olemassa jo paljon eikä kaikki tarvitse esimerkiksi parempaa mobiilitukea ja progressiiviset verkkosovellukset ovat vielä aika uusi konsepti web-kehityksessä. Perinteisen verkkosovelluksen kustantaminen on halvempaa, jolloin sitä voidaan pitää toimivana ratkaisu tietyissä tilanteissa. (21.)

3.2 Natiivit mobiilisovellukset

Mobiilikäytössä natiivit mobiilisovellukset ovat yleinen vaihtoehto tuoda sovellus markkinoille. Natiiveissa mobiilisovelluksissa yleensä otetaan mukaan sovelluskauppa, joka julkaisee sovelluksen.

Natiivit mobiilisovellukset pystyvät hyödyntämään paremmin laitteiden raakaa suorituskykyä, jolloin tilanteissa, missä tarvitaan kaikki mahdollinen laitteen teho, kuten esimerkiksi peleissä natiivit mobiilisovellukset ovat parempia. Tähän vaikuttaa suuresti se, että natiiveilla sovelluksilla ei ole mitään välikättä kuten selainta hidastamassa sovellusta. Progressiivisia verkkosovelluksia käytetään selaimen avulla, joka yleensä kuluttaa enemmän akkua kuin sama sovellus natiivina mobiilisovelluksena.

Jos halutaan luoda sovellus, joka vaatii laitteistotoimintoja, kuten antureita, NFC:tä tai geofencingiä, niin natiivit mobiilisovellukset ovat oikeastaan ainut

vaihtoehto, koska niillä voidaan ottaa nämä ominaisuudet käyttöön varmasti kaikille käyttäjille. Geofencing eli niin sanottu elektroninen aitaus, joka aitaa tietyn alueen, jonka sisällä tapahtuu jotain, jos laite on aidan sisällä ja tämä perustuu käyttäjän sijaintiin, kun esimerkiksi auto voi avata ovet, kun puhelin tulee tarpeeksi lähelle autoa. Raskaammat sovellukset ovat myös toinen syy valita natiivi mobiilisovellus progressiivisen verkkosovelluksen sijaan. (22.)

Molemmat sovellustyypit ovat mainioita tapoja tuoda sovellus käyttäjille, mutta päätös pitää tehdä vahvuuksien ja heikkouksien perusteella. Voidaan todeta, että tilanteeseen, jossa sovellus pitää saada mahdollisimman nopeasti käytettäväksi laajalle yleisölle ja monelle alustalle, niin progressiivinen verkkosovellus on parempi vaihtoehto, koska se vaatii vain yhden koodikannan. Tilanteeseen, jossa kaikki mahdollinen teho, virrankulutus halutaan pitää mahdollisimman pienenä tai halutaan käyttää laitteen omia antureita tai ominaisuuksia, niin natiivi mobiilisovellus on parempi vaihtoehto. (22.)

3.3 Työpöytäsovellukset

Verkkosovellukset ja työpöytäsovellukset pyrkivät yleisesti vastaamaan molemmat erilaisiin tilanteisiin ja näiden välillä valinta on yleensä hyvin yksinkertainen. Työpöytäsovellus yleensä ladataan kerran ja käytetään ilman verkkoyhteyttä, kun taas verkkosovellusta käytetään selaimen ja verkon avulla. Työpöytä sovellukset pystyvät hyödyntämään enemmän tietokoneen ominaisuuksia kuin Web-sovellus. Hyvänä esimerkkinä siitä, mitä työpöytäsovellus voi hyödyntää, on paikallinen tiedostojärjestelmän hyödyntäminen, jota ei voida hyödyntää web-sovelluksessa.

Progressiiviset verkkosovellukset eivät ole saaneet niin isoa huomiota työpöytäkäytössä, kun mobiilikäytössä, koska perinteiset verkkosovellukset ovat hoitaneet samat tehtävät aikaisemmin toimivilla käyttöliittymillä.

Web-sovelluksilla on hyvänä puolena se, että niitä ei tarvitse ladata. Progressiiviset verkkosovellukset kuitenkin antavat myös mahdollisuuden asentaa

sovellus ja asettaa se työpöydälle samalla tavalla kuin työpöytäsovellukset. Sovelluksen uusimman version saapuessa web-sovellus päivittyy automaattisesti, kun taas työpöytä sovelluksen joutuu aina lataamaan manuaalisesti.

Progressiiviset verkkosovellukset eivät pysty hyödyntämään kaikkea tietokoneiden tarjoamaa suorituskykyä. Koska toiminnallisuus on web-pohjaista, vaativammat sovellukset hoidetaan mieluummin optimoiduilla työpöytäsovelluksilla.

Hyvä työpöytäsovellus on natiivi sovellus. Käyttöjärjestelmät kuten Windows ja macOS ovat asettaneet käyttöjärjestelmän yleisiä kehittäjän ohjeistuksia, joissa käsitellään esimerkiksi, kuinka sovellus käyttäytyy eri näppäinyhdistelmien kanssa (23). Tämän ansiosta työsovellukset käyttäytyvät hyvin paljon samalla tavalla erilaisia näppäinkomentoja käytettäessä, mikä helpottaa huomattavasti käyttäjiä, kun samat näppäinkomennot toistuvat eikä tarvitse muistaa jokaiseen sovellukseen eri näppäinyhdistelmiä. Näin saadaan sovellukset tuntumaan tutuilta ja se helpottaa käyttäjien tottumista uuteen sovellukseen.

4 Vaikutukset eri osapuolien näkökulmasta

Progressiiviset verkkosovellukset tuovat laajasti uusia näkökulmia kaikille osapuolille. Tässä osiossa käydään läpi asioita, mitä vaikuttavia tekijöitä eri osapuolilla on.

4.1 Suunnitellut sovelluksen käyttäjät

Sovellusten käyttäjät ovat kohderyhmänä laaja ja vaativa. Jokainen käyttäjä on oma yksilönsä omilla sovellusvaatimuksilla. Yritysten tehtävänä on pyrkiä miellyttämään mahdollisimman hyvin potentiaalisia käyttäjiä, jotta sovellusta käytettäisiin. Yritysten on siis yleensä tehtävä jonkinlaisia kompromisseja, jotta saadaan aikaiseksi mahdollisimman hyvä sovellus ulos siihen määritellyillä resursseilla. Sovellusten käyttäjillä on erilaisia syitä käyttää tai olla käyttämättä joitain sovelluksia. Progressiiviset verkkosovellukset vastaavat hyvin käyttäjien tarpeeseen seuraavissa tilanteissa.

Progressiiviset verkkosovellukset tarjoavat toiminnallisuudet vähäisellä datan käytöllä. Vähäinen datan kulutus käyttäjille, joilla on esimerkiksi rajallinen datan käyttö tai hidas yhteys voi olla ratkaisevassa asemassa siinä vaiheessa, kun mietitään, tuleeko sovellus käyttöön vai ei.

Progressiiviset verkkosovellukset on mahdollista asentaa laitteelle hyvin pienessä tiedostokoossa, joka on yleensä huomattavasti pienempi kuin muissa sovellustyypeissä (23). Tämä mahdollisuus pienentää käyttäjien kynnystä asentaa sovellus, jos tallennustila on rajallista.

Sovellusten käyttäjille on hyvä tarjota kilpailullisia vaihtoehtoja sovellusten käyttämiseen, koska jokaisella potentiaalisella käyttäjällä on oma preferoitu tapa käyttää sovelluksia. Sovellusten käyttäjiä on erilaisia, ja jopa sovelluksen asennusvaihe voi olla liian suuri kynnys, joten progressiiviset verkkosovellukset ovat oiva vastaus tähän tilanteeseen, kun annetaan mahdollisuus molempiin. (24.)

Monen alustan tuki on käyttäjille iso plussa, kun käyttäjän laitteella ei ole väliä. Sovelluksen käyttöönotto vaatii pelkän sovelluksen ja sillä voidaan taata ainakin jonkunlainen käyttökokemus jokaiselle laitteelle. Käyttäjille on iso helpotus, kun tietää sovelluksen toimivan, vaikka laitteet vaihtuvat ja alustat vaihtuvat.

Responsiivinen toiminnallisuus ja nopeat latausajat sovelluksissa yleensä antavat positiivisen kuvan käyttäjälle. Progressiiviset verkkosovellukset on luotu optimaalisesti, ja ne tarjoavat saumatonta vuorovaikutusta.

4.2 Sovelluksen kehittäjät

Sovellusten kehittäjiä on paljon ja jokaisella on omat osaamisalueet teknologioissa. Progressiiviset verkkosovellukset toteutetaan pääosin perinteisiä web-teknologioita käyttäen, joten kehittäjien ei välttämättä tarvitse opetella täysin uusia kieliä tai viitekehyksiä. Tällainen tilanne on sovelluskehittäjillä, jotka ovat erikoistuneet web-teknologioihin. Progressiivisista verkkosovelluksista ei ole vain yhtä mielipidettä sovelluskehittäjien keskuudessa, koska se voi vaikuttaa

positiivisesti osalle kehittäjistä ja toisaalta taas negatiivisesti joillekin kehittäjille esimerkiksi työllistymisen kannalta.

Progressiivisten verkkosovellusten tapauksessa yhdestä koodikannasta voidaan jakaa koko sovellus, mikä tarkoittaa sitä, että erillisiä sovelluksia eri alustoille ei välttämättä tarvitse tehdä.

Nopeat muutokset ja päivitykset ovat yleisesti verkkosovelluksissa yksinkertaista saada ulos käyttäjien saataville, kun jokaiseen muutokseen ei tarvita esimerkiksi sovelluskauppojen omia tarkistuskierroksia tai muita kolmansia osapuolia (25).

4.3 Ohjelmistoyritykset

Ohjelmistoyritysten on tärkeää huomioida käyttäjien vaatimukset sovellusta tehtäessä ja vastata näihin vaatimuksiin parhaalla tavalla. Käyttäjät vaativat, että sovellukset toimivat moitteettomasti ilman stressiä siitä, että sovellus esimerkiksi kaatuisi tai hävittäisi tärkeitä tietoja kesken käytön. Jokaisen sovelluksen käyttämisen kynnyksen pienentäminen vaikuttaa positiivisesti sovelluksen käyttökokemukseen. Ohjelmistoyritysten kannattaa tarkasti miettiä, minkälainen sovellustyyppi soveltuu parhaiten heidän tarpeisiinsa. Progressiiviset verkkosovellukset ovat hyvä vaihtoehto, mutta jokainen kuluttaja on erilainen eikä jokainen välttämättä tiedä tai osaa käyttää kaikkia sovellustyypin tarjoamia ominaisuuksia, joita heille tarjotaan.

Vuonna 2015 mobiililaitteiden osuus kaikesta verkkoliikenteestä oli noin kolmannes ja nyt vuoden 2023 ensimmäisellä kvartaalilla se oli yli 55 % (26). Tämän statistiikan perusteella voidaan arvioida, että verkkosivujen mobiilikäyttäminen kasvanut. Verkkosivujen mobiilikäyttämisen yleistymisen myötä verkkosivujen pitää pystyä vastaamaan käyttäjien kovia odotuksia nopeuden ja turvallisuuden suhteen. Mahdollisimman suuren kävijämäärän saamiseen vaaditaan mahdollisimman monen laitteen tuki, koska muuten menetetään potentiaalisia käyttäjiä.

4.3.1 Kustannukset

Erilaiset kustannukset ovat yrityksille arkipäivää ja niitä halutaan aina pienentää, jotta kulut pysyisivät mahdollisimman pieninä. Sovelluksen kustannuksiin liittyy myös paljon eri osa-alueita kuten luonti, suurempi päivittäminen tai ylläpito.

Sovelluksen ylläpitäminen ja luonti monelle eri alustalle voi olla työlästä, kun mahdollisessa tilanteessa yhden kohdan korjaaminen voi vaatia saman kohdan korjaamista sovelluksen eri alustojen koodikannoissa. Saman sovelluksen eri alustojen koodia voi olla tekemässä monta eri tiimiä, joka voi pahimmassa tapauksessa hidastaa esimerkiksi korjauksien tekemistä ja näin ollen eri alustoilla voi olla saman sovelluksen eri versiot. Progressiiviset verkkosovellukset tarjoavat yhden sovelluksen monelle alustalle yhdestä koodikannasta. Tämän avulla korjauksia tai päivityksiä tarvitsee tehdä enää yhdelle koodikannalle, jolla vähennetään tarvittavaa työtä. Yhden koodikannan hyötynä on myös se, että progressiiviset verkkosovellukset luodaan käyttäen perinteisiä web-teknologioita. Tämä mahdollisesti vähentää yritysten tarvetta hankkia erillisiä sovelluskehittäjiä tai tiimejä, jotka ovat erikoistuneet eri alustoille.

Progressiiviset verkkosovellukset ovat pohjimmiltaan hyvin samanlaisia kuin perinteiset verkkosovellukset, mikä tarkoittaa sitä, että ne voidaan julkaista helposti ja nopeasti ilman erilaisia sovelluskauppoja. Tämä mahdollistaa sen, että sovelluksen erilaiset tarkistukset ja vaatimukset hoidetaan täysin itse. Kun sovelluksen ei tarvitse mennä kolmansien osapuolien tarkistusten läpi, voidaan tehdä päätelmä siitä, että ei synny tilannetta, jossa sovellus jää julkaisematta, koska se ei läpäissyt tai miellyttänyt jotain kolmannen osapuolen moderointia. Pahimmassa tapauksessa on myös mahdollista, että sovellus poistetaan ilman syytä sovelluskaupasta, joka ei ole hyvä asia sovelluksen omistajalle. Kuitenkin halutessaan progressiiviset verkkosovellukset voi myös julkaista sovelluskauppoihin, mutta tämän jälkeen pitää täyttää heidän ehtonsa sovelluksen julkaisemiseen.

Aikaisemmat kohdat vastaavat tilanteeseen, jossa mietitään uuden sovelluksen luontia ja tapaa, jolla tämä halutaan toteuttaa. Tilanteessa, jossa yritykseltä löytyy jo erilliset mobiilisovellukset tai hyvin optimoitu ja skaalautuva perinteinen verkkosovellus, niin verkkosovelluksen muuntaminen tai uuden progressiivisen verkkosovelluksen luominen ei välttämättä ole kannattava vaihtoehto.

4.3.2 Käyttäjien käyttäytyminen sovelluksessa

Sovelluksen aktiiviseen käyttämiseen vaaditaan mahdollisimman monen asian menevän oikein käyttäjän näkökulmasta. Sovelluksen pitää olla helposti käytettävissä, tuntua hyvältä ja olla turvallinen.

Yleisesti verkkosovelluksia on helppo hakea ja käyttää selaimen avulla. Tämä hakemisprosessi on hyvä, mutta voi tuottaa osalle käyttäjistä kynnystä avata sovellus uudelleen esimerkiksi siitä syystä, että selainta käytetään verkon selaamiseen, jonka jälkeen jonkun tietyn sovelluksen olemassaolo saattaa unohtua. Jos halutaan kasvattaa lojaalien käyttäjien määrää, jotka käyttävät sovellusta aktiivisesti, niin sovelluksen avaaminen pitää olla mahdollisimman vaivatonta. Progressiiviset verkkosovellukset voivat pienentää kynnystä avata sovellus uudelleen asennettavuusominaisuuden avulla. Asennettavan sovelluksen yksi suurimpia hyötyjä on yksinkertaisesti se, että se on näkyvissä laitteen aloitusnäytöllä tai sovellusvalikossa, ja se on avattavissa yhdellä painalluksella. Tämä näkyvyys ja helppo avattavuus pienentää kynnystä avata sovellus.

Push-ilmoitukset ovat voimakas tapa markkinoida ja lisätä aktiivisuutta. Push-ilmoituksilla on monta hyvää positiivista vaikutusta sovellukselle. Push-ilmoitukset nostavat sovelluksen uudelleen avaamismäärää, koska ilmoituksen avulla saavutetaan laitteen käyttäjän huomio ilmoitukseen. Uudelleenavaamismäärä nousee tämän avulla, koska käyttäjillä voi mahdollisesti olla monia sovelluksia, joten yhden sovelluksen unohtuminen voi tapahtua helposti, kun jotain uutta ilmestyy.

Tilanteet, joissa yritykset voivat vaikuttaa potentiaalisten käyttäjien käyttäjäkokemukseen positiivisesti, voivat lisätä kävijämäärää ja keskimääräistä ajankäyttöä sovellusten sisällä.

Progressiivisilla verkkosovelluksilla on paljon hyviä ominaisuuksia web-puolella. Tämä on vielä kuitenkin suhteellisen tuore sovellusmuoto, niin käyttäjät eivät välttämättä tiedä, että tällainen sovellustyyppi on olemassa. Vaikutus tällä ilmiöllä on se, että käyttäjät eivät mahdollisesti osaa tai tajua käyttää ominaisuuksia, mitä progressiiviset verkkosovellukset tarjoavat.

4.3.3 Onnistumistarinoita

Progressiiviset verkkosovellukset ovat saaneet hyvän vastaanoton, ja ne ovat nostaneet merkittävästi joidenkin sovellusten statistiikkoja. Yleisimpinä syinä onnistumisille on huomattu olevan sovelluksen luotettavuus ja nopeus.

Ilmoitustaulutyyppinen sosiaalinen median sovellus Pinterest tarjosi ennen progressiivista verkkosovellusta perinteiset verkkosivut, iOS- ja Android-mobiilisovellukset. Ennen progressiivista verkkosovellusta heidän verkkosivunsa olivat hitaat, ja he huomasivat, että vain murto-osa käyttäjistä latasi mobiilisovelluksen. Progressiivisen verkkosovelluksen jälkeen verkkosivulla kävijät viettivät huomattavasti enemmän aikaa verrattuna edelliseen verkkosovellukseen. Huomattiin myös, että käyttäjät ovat enemmän sitoutuvia sovelluksen käyttöön eli käyttivät sovellusta useamman kerran (27). Tästä onnistumistarinasta voidaan ottaa huomioon se, kuinka paljon sovelluksen nopeus vaikuttaa käyttäjien haluun käyttää sovellusta ja sitoutumiseen sovellusta kohtaan.

Kahvilaketju Starbucks julkaisi heidän progressiivisen verkkosovelluksensa vuonna 2017 tuodakseen huomattavasti paremman käyttökokemuksen heidän verkkosivuillensa. Starbucks:in natiivisovelluksen kaltainen toteutus tarjoaa käyttäjille saumatonta käyttökokemusta myös ilman verkkoa. Julkaisun jälkeen Starbucks kertoo tuplanneensa päivittäiset käyttäjät ja lisänneensä lähes neljänneksen tilausten monimuotoisuutta. (28.) Tästä onnistumistarinasta voidaan

ottaa irti se, että saumaton käyttökokemus ja mahdollisuus käyttää sovellusta ilman verkkoa voi antaa huomattavia parannuksia sovelluksen menestymiselle.

Viestintäpalvelu Twitter tai X myös julkaisi heidän progressiivisen verkkosovelluksensa vuonna 2017 kutsuen tätä sovellusta nimellä Twitter Lite. Viestintäpalvelu huomasi käyttäjien olevan 80 % mobiilikäyttäjiä, jolloin he saivat idean luoda paremman käyttökokemuksen käyttäjilleen. Nimen lisäys Lite annettiin sovellukselle, koska sen koko verrattuna heidän mobiilisovelluksiinsa oli murto-osan ja sovellus käytti myös huomattavasti vähemmän verkkoa, koska voitiin päivittää vain tiedot, jotka muuttuivat näkymässä (29). Tämän muutoksen jälkeen Twitter Lite -verkkosivujen mobiilikäyttäjistä tuli aktiivisempia, ja he viettivät enemmän aikaa sovelluksen parissa.

Muistetaan kuitenkin, että tässä kerrottiin vain onnistumisia. Sovellustyypin muuttaminen ei takaa suoraa onnistumista ja jotkut sovellukset hyötyvät muutoksista muita sovelluksia paremmin, koska muutokset auttavat eri luonteisia sovelluksia eri tavalla.

5 Web-sovelluksen muuttaminen progressiiviseksi verkkosovellukseksi

Tässä osioissa käydään läpi sitä, miten tässä yksittäistapauksessa olemassa oleva verkkosovellus muutetaan progressiiviseksi verkkosovellukseksi. Osiossa kerrotaan siitä, miten muuntamisprosessi etenee päiväkirja tyyliseen tapaan. Osio antaa vastauksen myös siihen, milloin kannattaa mahdollisesti muutos tehdä.

5.1 Alkuasettelu ja käytetyt teknologiat

Opinnäytetyön tavoitteeksi asetettiin se, että opinnäytetyön aikana luodaan perinteinen verkkosovellus käyttäen perinteisiä web-tekniikoita ja muutetaan se progressiiviseksi verkkosovellukseksi.

Tämän sovelluksen toteuttamiseen valikoitui ReactJS-kirjasto, jolla toteutettiin perustoiminnallisuus ja Tailwind CSS -viitekehys, jolla toteutettiin sovelluksen tyylittely. Teknologiat valikoituivat, koska tekijällä oli aikaisempaa kokemusta web-kehittämisestä kyseisillä teknologioilla.

ReactJS on frontend-kehittämiseen Facebookin toimesta vuonna 2013 luotu avoimen lähdekoodin JavaScript-kirjasto käyttöliittymäkehittämiseen. ReactJS:n etuna nähdään sen rakentama virtuaalinen DOM, joka on nopeampi ja kevyempi kuin tavanomainen DOM. Virtuaalinen DOM mahdollistaa sen, että päivitetään vain ne objektit, jotka muuttuvat eikä kaikkia mahdollisia objekteja. ReactJS:n kehittäminen perustuu komponentteihin, jotka sisältävät aina tietyn toiminnallisuuden logiikan, ja yksi sovellus yleensä koostuu monesta komponentista. Komponentit ovat uudelleenkäytettäviä, joten yhtä komponenttia voidaan hyödyntää useassa paikassa, mikä vähentää ylimääräistä duplikaattikoodin määrää. ReactJS seuraa yksisuuntaista tietovirtaa eli alikomponentit (engl. child component) sijoitetaan yläkomponenttien (engl. parent component) sisälle, jolloin on helpompi selvittää ja tietää, missä kohdassa ongelmat sovelluksessa esiintyvät. (30.)

Tailwind CSS -viitekehys on suosittu työkalu, joka tarjoaa valmiita matalan tason CSS-tyylittelyjä verkkosivun luomiseen. Viitekehys on luotu hyödyllisyys ensin lähestymistavalla, joka eroaa perinteisistä CSS-viitekehyksistä siten, että Tailwind ei tarjoa valmiiksi rakennettuja erillisiä komponentteja vaan valmiiksi tyyliteltyjä HTML-elementtejä, jotka ovat muokattavissa. Tämän tavoitteena on pitää koodi yksinkertaistettuna. Viitekehyksen tavoitteena on helpottaa ja nopeuttaa sovelluksen kehittämistä säästämällä aikaa, joka menisi tyylittelyyn ilman viitekehystä. (31.)

5.2 Tavoitteet

Projektin tavoitteena on saada luotua perinteinen verkkosovellus, josta muunnetaan progressiivinen verkkosovellus. Lopullisen sovelluksen tulee täyttää seuraavia vaatimuksia. Progressiivisen verkkosovelluksen minimivaatimukset eli

web app manifest -tiedosto pitää löytyä, service worker pitää saada toimimaan, HTTPS-yhteyden käyttäminen. Tämän kokonaisuuden toteamiseen käytetään Google Lighthouse -työkalua. Google Lighthouse ajaa testin sovelluksella ja tarkistaa sen, että siitä löytyvät kaikki vaadittavat kohdat, jotta se täyttää progressiivisen verkkosovelluksen minimivaatimukset. Sovelluksen asennettavuus ja perustoiminnallisuus pitää varmistaa useamman laite- ja selainyhdistelmän avulla.

5.3 Prosessi

Suunnittelun jälkeen tämä projekti alkoi siitä, että aluksi luotiin ReactJS-sovelluksen pohja luontiskriptin avulla, jonka jälkeen alettiin toteuttamaan itse sovellusta. Sovellus luotiin käyttämällä esimerkkikoodin 3 esiteltyä skriptiä.

```
npx create-react-app sovellus
```

Esimerkkikoodi 3. Yleinen luontiskripti, jonka avulla voidaan luoda ReactJS-sovelluksen pohja.

Tämän jälkeen sovellukseen asennettiin Tailwind CSS. Tailwind CSS:n asennus aloitettiin laittamalla esimerkkikoodin 4 asennusskriptit sovelluksen terminaaliin, joka asentaa Tailwind CSS:n ja lisää tailwind.config.js-tiedoston, joka vaaditaan Tailwind CSS -viitekehyksen toimintaan.

```
npm install -D tailwindcss
npx tailwindcss init
```

Esimerkkikoodi 4. Asennusskriptit Tailwind CSS-viitekehykselle ja tailwind.config-js-tiedoston luontiin.

Tailwind CSS:n asennus vaatii tyyliteltävien tiedostojen polun ja tiedostojen tyyppien lisäämisen tailwind.config.js-tiedostoon. Esimerkkikoodi 5 näyttää sen, mitä lopullinen tailwind.config.js-tiedosto pitää sisällään.

```
 /** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    "./src/**/*.{js,jsx,ts,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

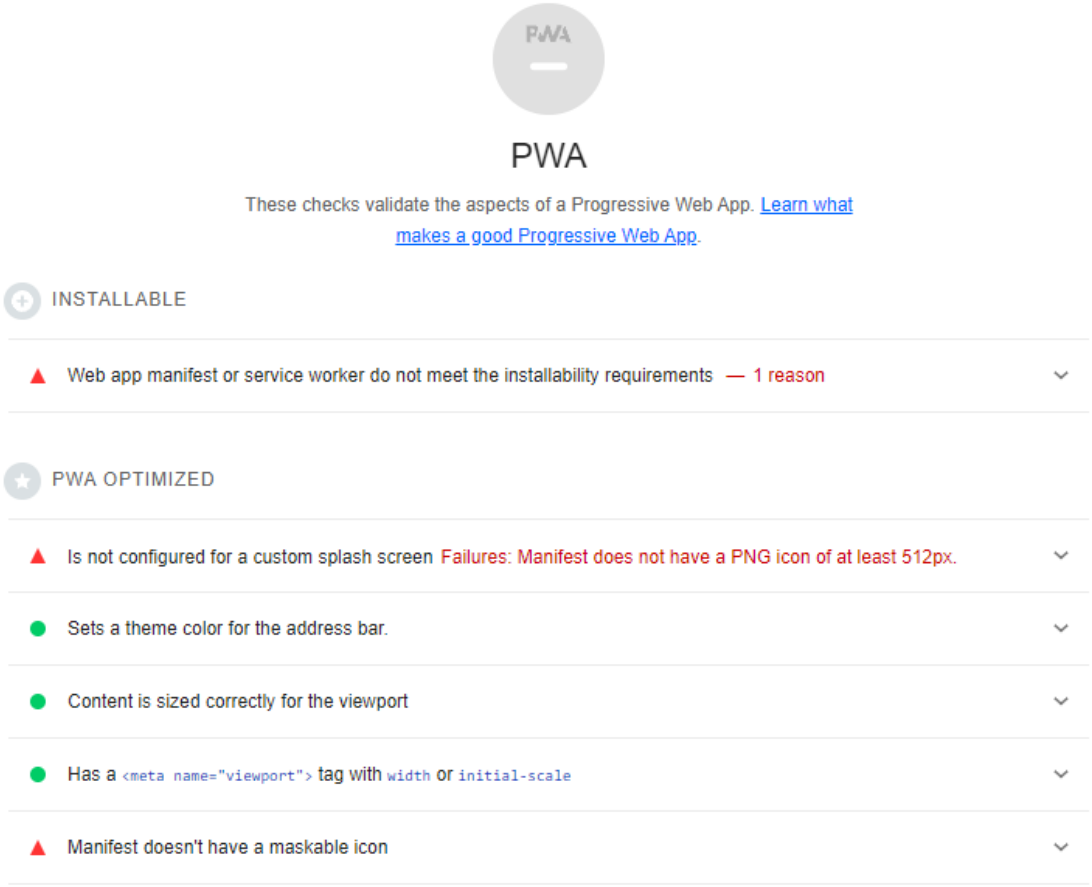
Esimerkkikoodi 5. Näyttää mitä Tailwind CSS-viitekehyksen tarvitseman tailwind.config.js-tiedoston sisältö on.

Tailwind CSS:n asentamisen viimeinen vaihe on Tailwind-ohjeiden lisääminen ./src/index.css-tiedoston alkuun. Esimerkkikoodi 6 lisätään ./src/index.css-tiedoston alkuun.

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Esimerkkikoodi 6. Tarvittavat CSS-ohjeet Tailwindin käyttöönottoon ./src/index.css-tiedostoon.

Projektin ensimmäinen vaihe oli hyvin suoraviivainen, sillä projektin tekijällä oli aikaisempaa kokemusta sovelluksen teosta tällä viitekehyksellä. Kun sovelluksen perustoiminnallisuus todettiin toimivaksi, niin katsottiin Lighthouse-tarkistuksen mukaan, mitä puuttuu siitä, että sovellus olisi progressiivinen verkkosovellus. Tarkistuksen perusteella sovelluksesta ei paljoa puutu, jotta se täyttäisi progressiivinen verkkosovelluksen vaatimukset (kuva 1).



Kuva 1. Lighthouse-tarkistuksen raportti ennen sovelluksen päivittämistä

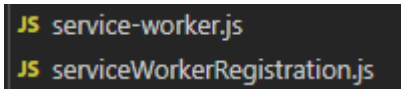
Muuntamisvaihe alkoi siitä, että katsottiin Lighthousen antamaa analyysiä ja tutkittiin, mitä kaikkea tarvitaan siihen, että sovellus voidaan muuttaa progressiiviseksi verkkosovellukseksi. Tutkinnan jälkeen huomattiin, että perinteisen verkkosovellukseen luotu sovelluksen pohja luotiin käyttämällä yleistä ReactJS-viitekehyksen luontiskriptiä, johon oli helposti lisättävissä progressiivisen verkkosovelluksen ominaisuudet. Aloitettiin siitä, että muokattiin skriptin mukana tullutta web app manifest -tiedostoa päivittämään sellaiseen kuntoon, että se täyttää tarvittavat vaatimukset. Manifest-tiedosto ei vaatinut tässä tapauksessa, kuin tarpeellisten kuvakkeiden lisäämistä, jotta sovelluksella olisi jokin kuvake erilaisiin tilanteeseen. Manifest-tiedosto vaati maskattavissa (engl. maskable) olevan kuvakkeen, jota voidaan näyttää eri muotojen sisällä, kuten esimerkiksi Android-sovelluksen kuvakkeet ovat yleensä ympyrän muotoisia.

Service worker -toiminnallisuuden lisääminen tässä tapauksessa onnistui helposti, kun projektissa käytetystä luontiskriptistä löytyy toinen versio, jolla voi luoda progressiivisen verkkosovelluksen ReactJS pohjan. Esimerkkikoodi 7 esittelee luontiskriptin, jolla luotiin toinen sovelluksen pohja, josta otettiin tarvittavat tiedostot muunnettavaan sovellukseen.

```
npx create-react-app sovellus –template cra-template-pwa
```

Esimerkkikoodi 7. ReactJS-sovelluksen pohjan luontiskripti, jolla voidaan luoda progressiivisen verkkosovelluksen pohja.

Tiedostot, jotka otettiin tästä progressiivisen verkkosovelluksen pohjasta muunnettavaan perinteiseen verkkosovellukseen (kuva 2).



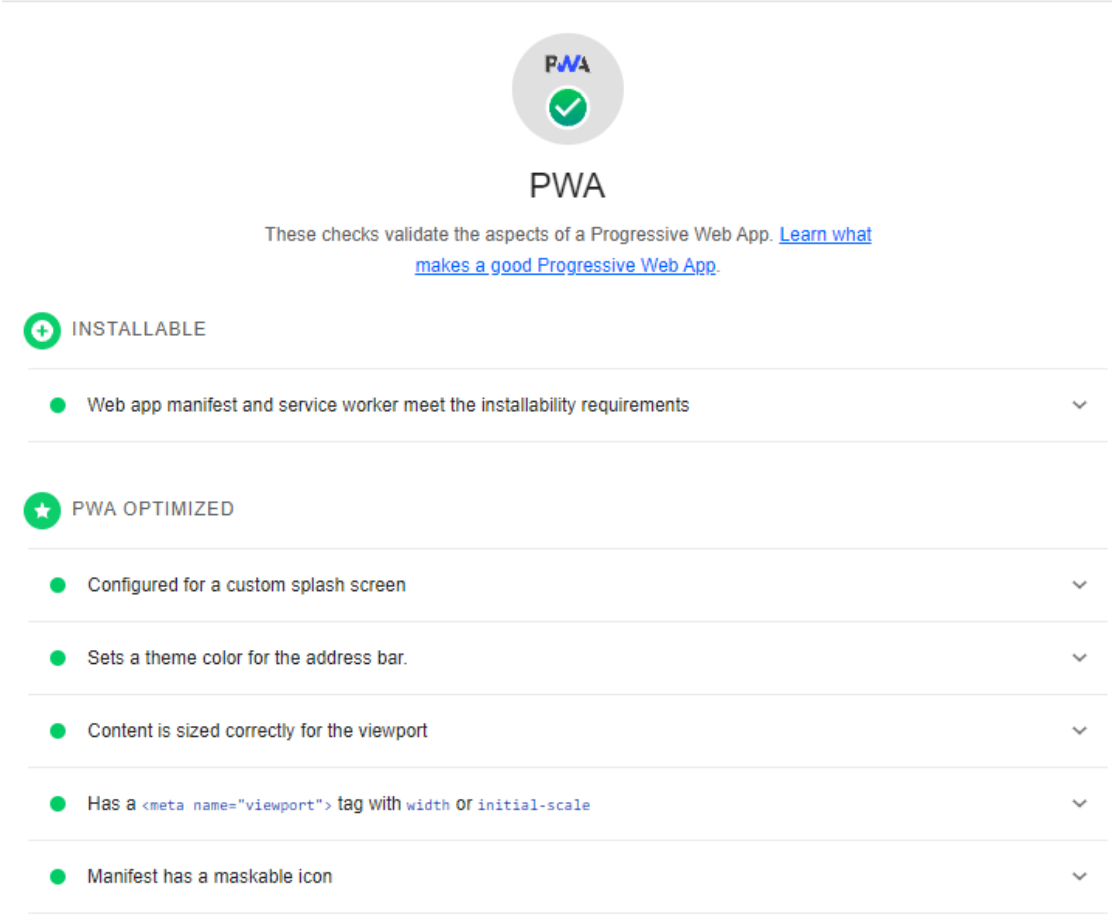
Kuva 2. Kuva tiedostoista, jotka otettiin muunnettavaan sovellukseen.

Nämä tiedostot lisättiin muunnettavaan sovellukseen, jonka jälkeen piti lisätä koodirivi sovelluksen index.js-tiedostoon, jotta service worker saadaan päälle. Esimerkkikoodi 8 on koodinpätkä, joka piti lisätä sovelluksen index.js-tiedostoon, jotta service worker saatiin päälle ja toimintaan.

```
serviceWorkerRegistration.register();
```

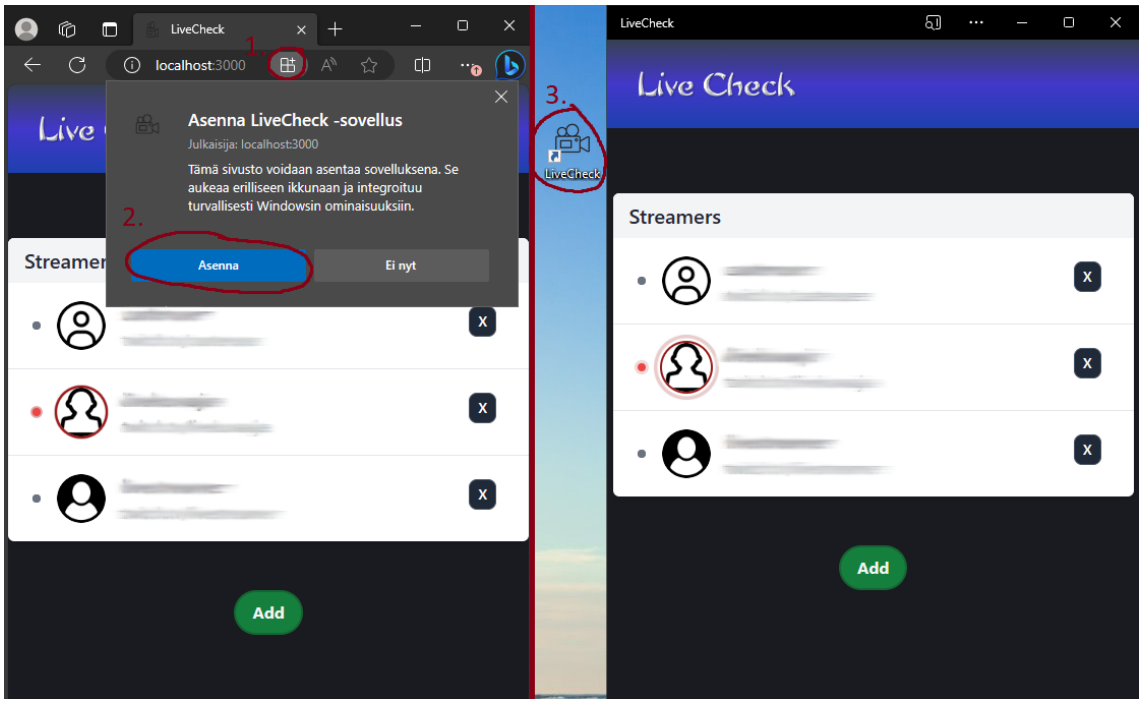
Esimerkkikoodi 8. JavaScript koodinpätkä, joka tarvittiin tässä tilanteessa service workerin laittamiseen päälle.

Tämän jälkeen aloitettiin testaaminen. Testit ajettiin lokaalisti, jolloin sovelluksen ympäristö käyttäytyy samalla tavalla kuin siinä olisi käytetty HTTPS-yhteyttä, jolloin sitä ei erikseen otettu tässä huomioon. Näiden muutosten jälkeen sovellus saatiin Lighthouse-tarkituksen mukaan muutettua progressiiviseksi verkkosovellukseksi (kuva 3).



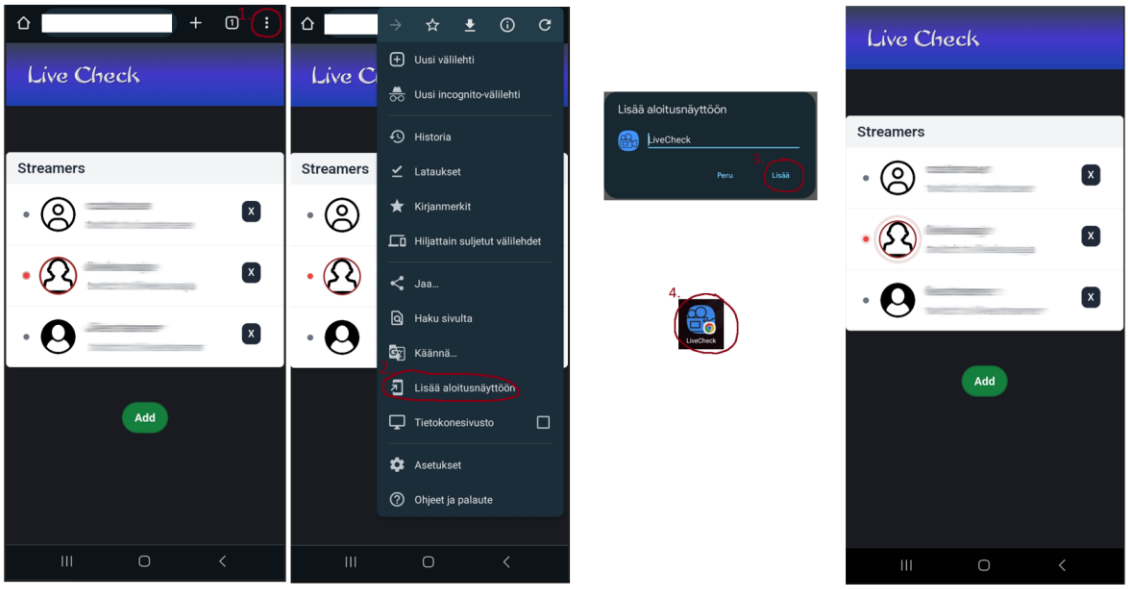
Kuva 3. Lighthouse-raportti sovelluksen päivityksen jälkeen

Tarkistuksen jälkeen todettiin, että sovellus on nyt muutettu progressiiviseksi verkkosovellukseksi ja varmistettiin vielä, että halutut toiminnallisuudet toimivat oikealla tavalla muutamassa eri käyttöjärjestelmässä. Sovelluksen pystyi nyt ladaamaan, ja ladatun sovelluksen kanssa kokeiltiin myös, että kaikki toiminnallisuudet toimisivat vielä niin kuin pitäisi. Ladattavan sovelluksen testaus testattiin siten, että sovellus avattiin ensin selaimessa ja sitten asennettiin laitteelle, avattiin, kokeiltiin sovelluksen ominaisuuksien toimivuus, jotta ne toimisivat samalla tavalla kuin selaimessa. Toimivuuksien kokeileminen selaimissa tehtiin myös tässä samassa vaiheessa. Kuvassa 4 näytetään visuaalisesti se, kuinka sovellus asennettiin Windows-käyttöjärjestelmässä Microsoft Edge -selaimen kanssa.



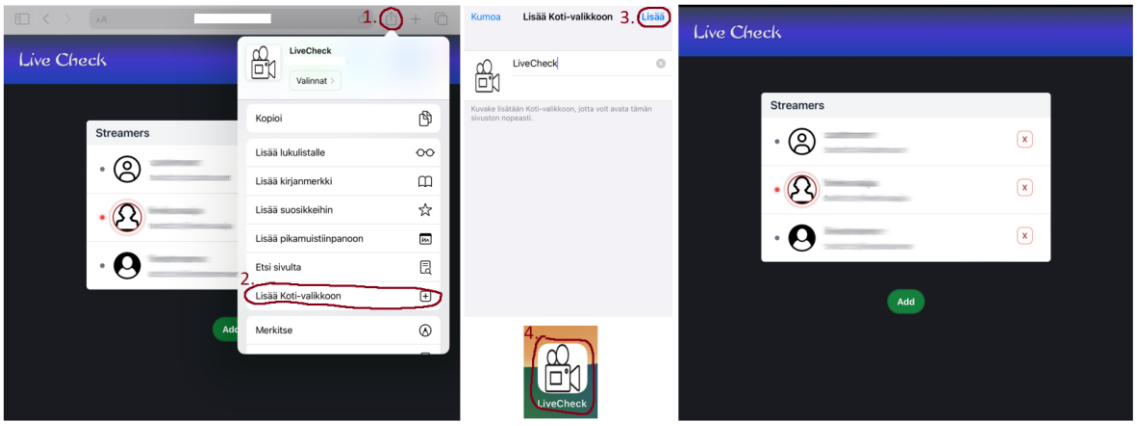
Kuva 4. Progressiivisen verkkosovelluksen asennuksen visualisointi ja avaaminen Windows-käyttöjärjestelmässä Microsoft Edge -selaimen kanssa.

Sovellus todettiin toimivaksi Microsoft Edge -selaimella ja Windows-käyttöjärjestelmässä asennettuna. Haluttiin vielä varmistaa, että sama sovellus toimii myös muilla käyttöjärjestelmillä. Android-käyttöjärjestelmällä testaaminen tehtiin samalla tavalla, eli sovellus avattiin selaimessa ja sitten asennettiin, avattiin ja kokeiltiin sovelluksen ominaisuuksien toimivuus. Kuvassa 5 näytetään visuaalisesti, kuinka asennus tehtiin Android-käyttöjärjestelmässä Chrome-selaimen kanssa.



Kuva 5. Progressiivisen verkkosovelluksen asennuksen visualisointi ja avaaminen Android-käyttöjärjestelmässä Chrome-selaimen kanssa.

Sovellus toimi molemmissa selaimessa ja asennettuna juuri niin kuin haluttiin. Sovelluksen asentaminen ja toimivuus kokeiltiin myös iOS-käyttöjärjestelmässä. iOS-käyttöjärjestelmässä käytettiin Safari-selainta. Kuten Android-käyttöjärjestelmässä myös iOS-käyttöjärjestelmällä testaaminen tehtiin samalla tavalla, eli sovellus avattiin selaimessa ja sitten asennettiin, avattiin ja kokeiltiin sovelluksen ominaisuuksien toimivuus (kuva 6).



Kuva 6. Progressiivisen verkkosovelluksen asennuksen visualisointi ja sovelluksen avaaminen iOS-käyttöjärjestelmässä Safari-selaimen kanssa.

Sovellus todettiin toimivaksi iOS-käyttöjärjestelmässä molemmissa asennettuna ja selaimessa. Perinteisen verkkosovelluksen tyylittely oli toteutettu skaalautuvasti. Näin ollen muutoksia ei tarvinnut tehdä, ja sovellus oli hyvin käytettävissä myös mobiililaitteilla.

5.4 Lopputuloksen arviointi

Lopputuloksena saatiin muutettua perinteinen verkkosovellus progressiiviseksi verkkosovellukseksi onnistuneesti. Sovellus onnistui täyttämään kaikki annetut vaatimukset. Projektissa käytetty Google Lighthousen tarjoama automaattinen tarkistus tarkisti, että sovellus täyttää progressiivisen verkkosovelluksen minimivaatimukset ja näin ollen tarjoaa hyvän pohjan tulevaisuuden sovelluskehitykseen.

Vaikka sovellus todettiin hyvin toimivaksi näillä testatuilla laite- ja selainyhdistelmillä, niin sovelluksen täyttä toimivuutta jokaisella mahdollisella laitteella, selaimella ja selaimen versiolla ei kuitenkaan voida taata.

Jälkeenpäin voidaan arvioida, että muutos tässä yksittäistapauksessa tapahtui suhteellisen vaivattomasti. Tähän vaikuttaa suuresti se, että projektin skaala oli pieni ja saatiin suuri apu siitä, että perinteiseen sovellukseen oli helppo tehdä vaadittavat muutokset siinä käytetyn luontiskriptin ansiosta.

Vaikka tämän yksittäistapauksen tilanteessa muutos onnistui hyvin ilman, että suurempia muutoksia tarvitsi tehdä, niin ei voida varmaksi sanoa, miten kaikissa tilanteissa käy. Näin ollen todetaan, että muutoksen tekemistä kannattaa miettiä tarkkaan ennen kuin siihen sitoutuu, koska yllätyksellisiä ongelmia voi ilmetä. Harkita kannattaa etenkin siinä tapauksessa, jos ei ole tietoa paljoa aiheesta tai kiinnostusta muutosta kohtaan.

6 Yhteenveto

Progressiiviset verkkosovellukset pyrkivät olemaan tämän hetken universaali sovellustyyppi, jota pystytään käyttämään lähes laitteesta riippumatta. Progressiiviset verkkosovellukset tarjoavat hyvän välimallin perinteisten ja natiivien sovellusten väliin ottamalla molemmista hyviä puolia.

Tilanteiden ollessa suotuisat progressiiviset verkkosovellukset ovat erittäin potentiaalinen vaihtoehto sovellustyypiksi, sillä sen avulla voidaan tarjota lähes samankaltaista käytettävyyttä kuin natiiveilla sovelluksilla. Eri näkökulmista katsottuna jokainen osapuoli hyötyy ainakin, jollain tavalla progressiivisista verkkosovelluksista.

Perinteinen verkkosovellus pystyttiin muuntamaan onnistuneesti progressiiviseksi verkkosovellukseksi. Vastaus kysymykseen, milloin perinteinen verkkosovellus kannattaa muuntaa progressiiviseksi verkkosovellukseksi, ei ole yksiselitteinen, koska muuttujia on paljon ja jokainen tilanne on erilainen. Yksittäistapauksen perusteella voidaan kuitenkin sanoa, että muuntaminen on mahdollista.

Lähteet

1 Bulut, Muge, Evren. 2021. PWA – Combining the Bests of Mobile and Web Worlds. Verkkoaineisto. Medium. <https://medium.com/quick-code/pwa-combining-the-bests-of-mobile-and-web-worlds-51d20d074a0>. Luettu 21.9.2023.

2 Nagar, Tarun. 2023. Why are progressive web apps becoming the future of web development?. Verkkoaineisto. TechTarget. <https://www.datas-ciencecentral.com/why-are-progressive-web-apps-becoming-the-future-of-web-development/> Luettu 5.11.2023.

3 Gvak, Natalia & Mistry Vivek. 2022. Powerful apps fueled by the web: How developers are engaging an expanding ChromeOS audience. Verkkoaineisto. Google <https://chromeos.dev/en/posts/powerful-apps-fueled-by-the-web> Luettu 2.11.2023.

4 Tatis, Mario. 2023. Progressive Web Apps for iOS. Verkkoaineisto. koombea <https://www.koombea.com/blog/progressive-web-apps-for-ios/> Luettu 5.11.2023.

5 Uryutin, Oleg. 2018. A brief history of web app. Verkkoaineisto. Medium. <https://oleg-uryutin.medium.com/a-brief-history-of-web-app-50d188f30d> Luettu 5.11.2023.

6 Kwiecień, Aleksandra & Karwatka, Piotr & Grzybowska, Kaja & Rakowski, Filip. 2020. The PWA Book. E-kirja. Divante <https://www.divante.com/reports/pwabook/what-are-progressive-web-apps> Luettu 21.9.2023.

7 Tomasis, Rebecca. 2018. Static vs dynamic websites: the key differences and which to use. Verkkoaineisto. WixBlog. <https://www.wix.com/blog/static-vs-dynamic-website> Luettu 14.11.2023.

8 Rana, Mansi. 2023. Progressive Web Apps (PWAs): How do they Help?. Verkkoaineisto. Medium. <https://oleg-uryutin.medium.com/a-brief-history-of-web-app-50d188f30d> Luettu 5.11.2023.

9 Lighthouse, 2016 päivitetty 2022. Verkkoaineisto. Google Developers. <https://developer.chrome.com/docs/lighthouse/overview/> Luettu 2.11.2023.

10 Web app manifests. 2023. Verkkoaineisto. MDN. <https://developer.mozilla.org/en-US/docs/Web/Manifest> Luettu 2.11.2023.

11 Basques, Kayce. 2015 päivitetty 2020. Verkkoaineisto. Web dev. <https://web.dev/why-https-matters/> Luettu 2.11.2023.

12 Schechter, Emily. 2018. A milestone for Chrome security: marking HTTP as “not secure”. Verkkoaineisto. Google. < https://blog.google/products/chrome/milestone-chrome-security-marking-http-not-secure/> Luettu 2.11.2023.

13 Google Developers. 2014. HTTPS as a ranking signal. Verkkoaineisto. Google. < https://developers.google.com/search/blog/2014/08/https-as-ranking-signal> Luettu 2.11.2023.

14 Aas, Josh. 2014. Let’s Encrypt: Delivering SSL/TLS Everywhere. Verkkoaineisto. Let’s Encrypt. <https://letsencrypt.org/2014/11/18/announcing-lets-encrypt.html> Luettu 2.11.2023.

15 Service Worker API. 2023. Verkkoaineisto. MDN. <https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API> Luettu 2.11.2023.

16 Push API. 2023. Verkkoaineisto. MDN. <https://developer.mozilla.org/en-US/docs/Web/API/Push_API> Luettu 5.11.2023.

17 Kremer, Meryl. 2023. What is PWA?. Verkkoaineisto. OneSignal. <https://onesignal.com/blog/what-is-a-pwa/> Luettu 2.11.2023.

18 Steiner, Thomas. 2023. PWAs in app stores. Verkkoaineisto. webdev. <https://web.dev/articles/pwas-in-app-stores> Luettu 2.11.2023.

19 Richard, Sam & LePage, Pete. 2020 päivitetty 2022. What makes a good Progressive Web App. <https://web.dev/pwa-checklist/> Luettu 2.11.2023.

20 Warcholinski, Matt. 2023. Progressive Web Apps: Advantages and Disadvantages [2023]. Verkkoaineisto. Brainhub. <https://brainhub.eu/library/progressive-web-apps-advantages-disadvantages> Luettu 2.11.2023.

21 Adhav, Raj. 2023. Progressive Web Apps Vs Responsive Websites: How to Choose. Verkkoaineisto. designial. <https://designial.com/blogs/progressive-web-apps-vs-responsive-websites-how-to-choose> Luettu 2.11.2023.

22 Chaudhary, Anjali. 2023. Progressive Web Apps vs Native Apps: What Should You Pick?. Verkkoaineisto. Turing <https://www.turing.com/blog/progressive-web-apps-vs-native-apps/> Luettu 2.11.2023.

23 Gomez, Jose. 2023. Web Apps Vs. Desktop Apps: Understanding the Differences. Verkkoaineisto. koombea. <https://www.koombea.com/blog/web-apps-vs-desktop-apps/> Luettu 2.11.2023.

24 Singh, Hemendra. 2023. Advantages & Benefits of Progressive Web Apps (PWA). Verkkoaineisto. NineHertz. <https://theninehertz.com/blog/benefits-of-progressive-web-apps#Benefits_of_PWA_For_Developers> Luettu 2.11.2023.

25 Fourault, Sébastien. 2020. How Progressive Web Apps can drive business success. Verkkoaineisto. Web.dev. <https://web.dev/drive-business-success/> Luettu 2.11.2023.

26 Howarth, Josh. 2023. Internet Traffic from Mobile Devices (Oct 2023). Verkkoaineisto. Exploding Topics. <https://explodingtopics.com/blog/mobile-internet-traffic> Luettu 2.11.2023.

27 Comeau, Lesley. 2021. The Most Popular PWAs of 2021: An Intro to Progressive Web Apps. readwrite. <https://readwrite.com/the-most-popular-pwas-an-intro-to-progressive-web-apps/> Luettu 2.11.2023.

28 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 2.11.2023.

29 Gallagher, Nicolas. 2017. How we built Twitter Lite. Verkkoainesto. Open source. <https://blog.twitter.com/engineering/en_us/topics/open-source/2017/how-we-built-twitter-lite> Luettu 2.11.2023.

30 Deshpande, Chinmaysee. 2023. The Best Guide to Know What Is React. Verkkoaineisto. Simplilearn <https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs> Luettu 5.11.2023.

31 Fitzgeral, Anna. 2022. Tailwind CSS: What It Is, Why Use It & Examples. Verkkoaineisto. HubSpot <https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs> Luettu 5.11.2023.