Tampereen yliopisto

Arttu Moisio

PROGRESSIIVISEN WEB-SOVELLUKSEN OMINAISUUDET

Kandidaatintutkielma
Informaatioteknologian ja viestinnän tiedekunta
Maaliskuu 2022

TIIVISTELMÄ

Arttu Moisio: Progressiivisen web-sovelluksen ominaisuudet
Kandidaatintutkielma
Tampereen yliopisto
Informaatioteknologian ja viestinnän tiedekunta
Tietotekniikka
Maaliskuu 2023

Progressiiviset web-sovellukset hyödyntävät moderneja web-rajapintoja, joiden avulla web-sovellukseen voidaan sisällyttää natiiville applikaatiolle tyypillisiä ominaisuuksia. Tässä tutkielmassa perehdytään progressiivisen web-sovelluksen eli PWA:n määritelmään sekä verrataan progressiivista web-sovellusta perinteiseen verkkosivuun ja natiiviin sovellukseen.

Progressiivinen web-sovellus koostuu ainakin kolmesta tekijästä: salatusta yhteydestä, palvelutyöläisestä ja web-manifestista. Web-manifestissa määritetään PWA:n ulkoasu ja tiedot, joiden avulla sovellus voidaan asentaa laitteelle. Palvelutyöläinen on JavaScript-ohjelmakoodi, joka ajetaan omassa säikeessään. Se mahdollistaa resurssien varastoinnin, push-notifikaatiot ja taustaprosessoinnin eli tehtävien suorittamisen myös silloin, kun web-sovellus on suljettuna. Palvelutyöläisen käyttäminen on mahdollista vain käytettäessä salattua yhteyttä. Edellä mainittujen piirteiden lisäksi progressiiviset web-sovellukset hyödyntävät edistyneitä web-teknologioita natiivin sovelluksen kaltaisen kokemuksen tuottamiseksi.

Vertailun perusteella progressiiviset web-sovellukset ovat nopeampia ja käyttävät vähemmän dataa kuin perinteiset verkkosivut. Lisäksi progressiivisen web-sovelluksen kehittäminen ja levittäminen on vastaavaan natiiviin sovellukseen verrattuna helpompaa, ja ne vievät laitteelta vähemmän tallennustilaa. Puutteellinen ja huonosti ennustettava tuki, etenkin iOS:lla, ovat PWA:n heikkouksia. PWA ei pysty myöskään hyödyntämään kaikkia laitteen ominaisuuksia, minkä takia natiivi sovellus on usein suorituskykyisempi ja joihinkin käyttökohteisiin ainoa vaihtoehto.

Tutkielman perusteella perinteisen verkkosivun muuttaminen progressiiviseksi voi tuoda hyötyjä. Natiivin sovelluksen muuntaminen PWA:ksi ei usein ole kannattavaa, mutta yritykset ovat saavuttaneet taloudellista hyötyä rakentamalla PWA:n natiivin sovelluksen rinnalle. PWA soveltuu erityisesti kehittyville markkinoille, joissa laitteiden tallennustila ja datansiirto ovat rajoitettuja.

Avainsanat: PWA, Progressiivinen web-sovellus

SISÄLLYSLUETTELO

1. JOHDANTO

Moni organisaatio, yritys ja viranomainen on kehittänyt natiiveja mobiiliapplikaatioita, jotka laitteeseensa asentamalla käyttäjä pääsee käsiksi edistyneisiin toimintoihin ja joita käyttäjät käyttävät päivittäin. Mobiilisovelluksia kehitetään myös ratkaisemaan hyvin pieniä, rajattuja ja kertaluontoisia käyttötapauksia, minkä johdosta moniin mobiililaitteisiin on asennettu jopa kymmeniä applikaatioita, joita käyttäjät eivät käytä.

Moniin tällaisiin tilanteisiin soveltuisi erinomaisesti progressiivinen web-sovellus eli PWA. Progressiivinen web-sovellus kykenee vastaamaan samoihin tarpeisiin kuin natiivit sovellukset, mutta sitä ei tarvitse asentaa sovelluskaupasta ennen käyttöä. Perinteisistä web-sivuista poiketen PWA voidaan rakentaa toimimaan myös ilman verkkoyhteyttä. Esimerkiksi Starbucksin PWA:lla käyttäjä voi hallita lahjakortteja ja jopa maksaa offline-tilassa. (Grigsby, 2018)

PWA on laajasti käytetty tapa modernin mobiilin web-kokemuksen tuottamiseen. PWA voittaa perinteiset verkkosivut tarjoamalla kehittyneitä ominaisuuksia, jotka helpottavat sovellusten käyttöä ja joiden avulla käyttäjiä saadaan palaamaan sovelluksen pariin yhä uudelleen. Progressiivisten web-sovelluksien tarjoamat offline-ominaisuudet parantavat käytettävyyttä huonojen verkkoyhteyksien takaa (Hume & Osmani, 2018). PWA:n rakentaneet yritykset ovat raportoineet merkittävistä tuottavuusparannuksista (Grigsby, 2018).

Mobiililaitteiden käyttöjärjestelmien markkinat ja sen myötä mobiiliapplikaatioiden kauppapaikat ovat keskittyneet vahvasti kahden toimijan, Applen ja Googlen, ympärille (lähde). Nämä kaksi toimijaa käyttävät merkittävää valtaa ja pyrkivät myös hyötymään määräävästä markkina-asemasta taloudellisesti. Kaikkien näiden kahden toimijan kauppapaikoilla applikaatioitaan tarjoavien tahojen tulee täyttää näiden toimijoiden asettamat vaatimukset ja kauppapaikka veloittaa maksuina 15–30 % liikevaihdosta. PWA:n asentaminen ei riipu kauppapaikasta, mutta PWA voi olla tarjolla myös tällaisessa kauppapaikassa (Grigsby, 2018). PWA:n edistyneet toiminnot eivät edes vaadi sovelluksen asentamista laitteelle.

Perinteiset mobiiliapplikaatiot kirjoitetaan natiivilla koodilla, mikä on tyypillisesti laitelähеistä ja sen myötä suorituskykyistä. Progressiiviset web-sovellukset toteutetaan nimensä mukaisesti web-teknologioilla, joiden suorituskyky voi olla rajoittunutta. Modernit web-teknologiat, kuten WebAssembly, voivat kuitenkin tuoda tähän parannusta.

Tämän tutkielman tarkoituksena on selvittää, miten progressiivinen web-sovellus määritellään kirjallisuudessa sekä miten PWA:t vertautuvat natiiveihin mobiiliapplikaatioihin ja perinteisiin verkkosivuihin. PWA ei pysty peittoamaan natiivia sovellusta raa'assa laskentatehossa, mutta haluan selvittää, onko olemassa tapauksia, joissa PWA on natiivia sovellusta parempi vaihtoehto.

2. TUTKIMUSMENETELMÄ

Tämä tutkielma suoritettiin kirjallisuuskatsauksena, jonka tarkoituksena on rakentaa hyvä kokonaiskuva tutkimuskysymystä käsittelevästä kirjallisuudesta.

Aineiston keräämisessä hyödynnetään Andor-, Scopus-, Diva-portal-, ja ACM Digital Library -tietokantoja sekä Googlen Scholar -hakukonetta. Progressiivinen web-sovellus -käsite on muodostunut vuonna 2015, joten hakutulokset on rajattu käsittämään vain vuoden 2015 jälkeen ilmestynyttä aineistoa. Suomenkielistä kirjallisuutta ei alustavien hakujen perusteella ollut juurikaan saatavilla, joten hakusanoina on käytetty vain englanninkielisiä sanoja. Englanninkieliset hakusanat tuottivat kuitenkin jonkin verran suomenkielistäkin kirjallisuutta.

Tarvittavien käsitteiden määrittelemiseen pyrittiin löytämään kirjallisuutta, johon on viitattu laajasti aiheen tutkimuksessa. Muutoin hakusanat on pyritty rajaamaan siten, että saadaan riittävä määrä tutkimuskysymyksen kannalta relevanttia kirjallisuutta ilman, että hakutulosjoukko muodostuu liian suureksi, jolloin hakutuloksen läpikäyminen ei ole enää mahdollista. Alustavien hakujen perusteella kirjallisuutta ei ole vielä saatavilla kovin paljoa, joten systemaattiseen tiedonhakuun käytetyt hakusanat jätettiin tarkoituksella melko yleisluontoisiksi riittävien hakutulosten löytämiseksi.

Hakulausekkeita:

- Progressive web app
- "Progressive web app*" AND (evalua* OR performance OR compar* OR analysis)

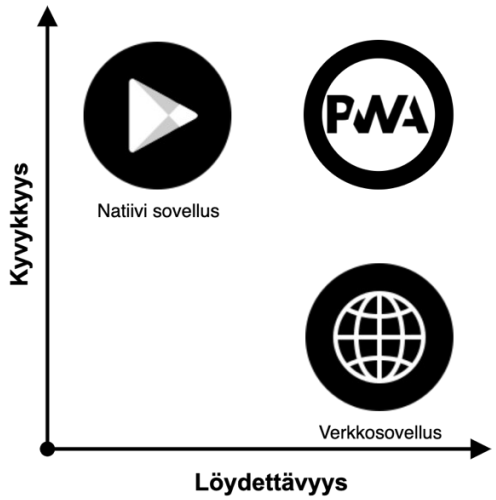
Yleisluontoiset hakusanat aiheuttivat sen, että hakutuloksia tuli suodattaa manuaalisesti riittävän laadukkaan ja relevantin kirjallisuuden valitsemiseksi. Suodattaminen tapahtui hakutulosten otsikoiden, tiivistelmien ja yhteenvetojen perusteella. Näiden pohjalta muodostui hyvä käsitys siitä, tarjoaako hakutulos tämän kirjallisuuskatsauksen kannalta olennaista tutkimustietoa.

3. PROGRESSIIVINEN WEB-SOVELLUS

Progressiivinen web-sovellus -käsitettä on käytetty vuodesta 2015 alkaen (Grigsby, 2018). Tässä luvussa määritellään, mitä progressiivisella web-sovelluksella tarkoitetaan tässä tutkielmassa.

Perinteisillä web-sovelluksilla on puutteita, joiden takia ne eivät sovellu kaikkiin käyttötarkoituksiin, mutta progressiiviset web-sovellukset poistavat joitain näistä puutteista (Fauzan et al., 2022). Progressiivisten web-sovelluksien tarkoituksena on moderneja web-rajapintoja hyödyntämällä tuottaa natiivin sovelluksen kaltainen kokemus, mutta silti saavuttaen käyttäjät missä tahansa, millä laitteella tahansa ilman riippuvuutta yksittäisen yrityksen hallitsemasta kauppapaikasta. PWA:n ajoalustana on laitteen selain, joka toimii hyvin samalla tavalla alustasta riippumatta. Tämän takia natiivin sovelluksen kehittäminen usealle alustalle vie 2-3 kertaa enemmän resursseja, kuin PWA:n kehittäminen (Khan et al., 2019).

Kuva 1 esittää diagrammin, missä nämä kolme sovellustyyppiä on asetettu karkeasti diagrammiin löydettävyyden ja kyvykkyyden mukaan.



Kuva 1. *Natiivin sovelluksen, verkkosovelluksen ja progressiivisen web-sovelluksen erot kyvykkyydessä ja löydettävyydessä (LePage & Richard, 2020).*

Kyvykkyydellä tässä tarkoitetaan laajasti ominaisuuksia, joita sovellustyyppi pystyy hyödyntämään. Löydettävyydellä tarkoitetaan sitä, kuinka helppoa sovellus on jakaa käyttäjille.

Web-sovelluksia jaetaan vapaassa verkossa ja ne ovat kaikkien saatavilla. Natiiveja sovelluksia jaetaan sovelluskaupasta, joka on alustakohtainen. Sovelluskaupan omistaja suodattaa julkaistavat sovellukset omien kriteeriensä mukaan ja käyttää merkittävää valtaa valitessaan julkaistavat sovellukset (Ambrasaitė & Smagurauskaitė, 2021). Tämä vallankäyttö on voi haitata vapaata kilpailua ja sovellusten vapaata saatavuutta.

3.1 Progressiivisen web-sovelluksen määritelmä

Progressiivisen web-sovelluksen määritelmä vaihtelee merkittävästi eri organisaatioiden välillä. Google, joka on merkittävä progressiivisia web-sovelluksia eteenpäin vievä toimija, on muuttanut määritelmää useita kertoja viime vuosien aikana. Progressiivisen web-sovelluksen ominaisuuksiksi myös kuvaillaan usein moderneja web-teknologioita, jotka eivät virallisesti kuulu progressiivisen web-sovelluksen määritelmään. (Grigsby, 2018)

Frances Berriman ja Alex Russel kehittivät progressiivinen web-sovellus käsitteen vuonna 2015 (Grigsby, 2018). Berrimanin ja Russelin mukaan progressiivinen web-sovellus koostuu yhdeksästä tunnusmerkistä:

- responsiivinen
- yhteydestä riippumaton
- tuntuu sovellukselta
- tuore
- turvallinen
- löydettävissä
- sitouttava
- asennettava
- linkattava

Näiden tunnusmerkkien lisäksi progressiivisen web-sovelluksen tulee olla rakennettu Progressive Enchancement -menetelmällä (Berriman & Russel, 2015). Progressive Enchancement menetelmässä luodaan ensin toteutukselle vakaa pohja, joka toimii kaikilla laitteilla, minkä jälkeen lisätään kerros kerrokselta ominaisuuksia niille selaimille, jotka tukevat näitä edistyneempiä ominaisuuksia (Selovuo, 2013).

Chrome Dev Summitissa (2017) PWA:n määritelmä muodostettiin FIRE-lyhenteen avulla. FIRE tulee sanoista:

- nopea (engl. fast)
- integroituva (engl. integrated)
- luotettava (engl. reliable)
- sitouttava (engl. engaging)

Tämä määritelmä kuvailee modernin, toimivan ja tehokkaan verkkosovelluksen, mutta se ei anna kovin tarkkaa kuvaa siitä, mikä tekee verkkosovelluksesta PWA:n.

Googlen kehittäjien, Sam Richardin ja Pete LaPagen (2020) mukaan progressiivinen web-sovellus on verkkosovellus, joka on suunniteltu kolmen periaatteen mukaan. Nämä periaatteet ovat:

- kykenevä
- luotettava
- asennettava

Nämä kolme suunnitteluperiaatetta saavat web-sovelluksen tuntumaan natiivilta sovellukselta.

Yhden määritelmän mukaan PWA on websivu, joka on tietoturvallinen, latautuu lähes välittömästi ja toimii myös offline-tilassa tai huonojen verkkoyhteyksien yli (Hume & Osmani, 2018). PWA-käsitteen kehittäjä Frances Berriman (2017) on itse kuvaillut käsitettä vain markkinoinnilliseksi välineeksi, jonka tarkoitus on herättää kiinnostusta selainten uusimpien ominaisuuksien tarjoamiin mahdollisuuksiin.

Puhtaasti teknisen määritelmän mukaan progressiivisen web-sovelluksen tulee sisältää kolme teknistä ominaisuutta: HTTPS-yhteyden, palvelutyöläisen ja web-manifestin (Keith, 2017). Tämän määritelmän mukaan progressiivisella web-sovelluksella voi olla paljon muitakin ominaisuuksia ja piirteitä, mutta ilman näitä kolmea verkkosovellusta ei voida kutsua progressiiviseksi.

Määritelmiä on siis useita, mutta näitä yhdistää se, että progressiivisen web-sovelluksen tulee tuntua samalta, näyttää samalta ja toimia samalla tavalla kuin natiivi sovellus. Tämä saavutetaan siten, että web-sovelluksen pitää olla asennettavissa päätelaitteelle, jolloin applikaatiota voidaan käyttää selaimen välilehden ulkopuolella. Applikaation tulee myös toimia offline-tilassa samaan tapaan kuin natiivi sovellus. Seuraavissa alaluvuissa kuvataan edellä mainitut progressiivisen web-sovelluksen tekniset vähimmäisvaatimukset.

3.1.1 HTTPS

HTTPS (lyh. Hypertext Transfer Protocol Secure) on verkkoliikenteen protokolla, jota käytetään verkkosivujen ja palvelimen välisen liikenteen salaamiseen. Salaamaton yhteys altistaa verkkosivun monille hyökkäyksille. Salattu yhteys on nykyään normaali käytäntö, jota lähes kaikki verkkosivut noudattavat, mutta tämä on progressiiviselle websovellukselle erityisen tärkeää, sillä jotkin PWA:n ominaisuudet, kuten palvelutyöläinen, vaativat turvallisuussyistä toimiakseen salatun yhteyden. (Basques, 2020)

3.1.2 Palvelutyöläinen

Palvelutyöläinen (engl. service worker) on tärkein PWA:n osa, sillä se mahdollistaa offline-toiminnan, mikä on aikaisemmin ollut mahdotonta. Vain muutaman rivin mittainen palvelutyöläinen mahdollistaa HTTP-pyyntöjen katkaisemisen, resurssien varastoinnin, taustasynkronoinnin ja push-notifikaatioiden hallinnan. (Hume & Osmani, 2018)

Palvelutyöläinen mahdollistaa offline-ominaisuuksia toimimalla välityspalvelimena. Tämä toimintamalli voidaan kuvata lennonjohtoanalogian avulla. Jos HTTP-pyynnöt ovat lähteviä lentokoneita, palvelutyöläinen on lennonjohtaja, joka ohjaa pyyntöjä. Pyynnöt voidaan täyttää välimuistista tai palvelimelta, riippuen välimuistin ja yhteyden tilasta. Jos selain ei tue palvelutyöläistä, verkkosovellus toimii ”normaalisti” ja pyynnöt ohjataan suoraan palvelimelle. (Hume & Osmani, 2018) Tämä toimintamalli on havainnollistettu kuvassa 2.

Kuva 2. *Palvelutyöläisen toiminta välityspalvelimena (Geddes, 2019).*

Palvelutyöläinen eroaa normaalista selaimessa ajettavasta JavaScript-ohjelmasta. Palvelutyöläinen asennetaan, kun käyttäjä vierailee verkkosivulla ensimmäisen kerran. Palvelutyöläinen ajetaan eri säikeessä ja globaalissa kontekstissa kuin itse verkkosovellus, eikä se ole sidottu yksittäiseen välilehteen, mikä mahdollistaa palvelutyöläisen ajamisen myös silloin, kun käyttäjä sulkee verkkosivun. Palvelutyöläinen ei voi suoraan muokata verkkosovelluksen elementtejä, koska sillä ei ole pääsyä verkkosovelluksen DOM:iin

(Document Object Model, suom. Dokumenttioliomalli). Nämä tekijät tulee ottaa huomioon palvelutyöläisen kehityksessä.

3.1.3 Web-manifesti

Web-manifesti on JSON-tiedosto, jossa määritellään miten PWA:n tulisi käyttäytyä, kun se asennetaan käyttäjän laitteelle. Tyypillisessä manifestissa määritellään applikaation nimi, kuvaus, ikonit ja ulkoasu eri laitteilla (LePage et al., 2022). Web-manifesti on se PWA:n osa, joka mahdollistaa verkkosovelluksen asentamisen käyttäjän laitteelle (Hume & Osmani, 2018). Ohjelma 1 alla on esimerkki yksinkertaisesta web-manifestista.

```
{
  "name": "Edistynyt web-sovellus",
  "short_name": "EWA",
  "description": "Edistynyt verkkosivusto, joka voidaan asentaa.",
  "lang": "fi",
  "icons": [
    {
      "src": "/android-chrome-192x192.png"
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "/android-chrome-512x512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "theme_color": "#215CCA",
  "background_color": "#215CCA",
  "display": "standalone",
  "start_url": "/",
  "categories": ["technology", "web"]
}
```

Ohjelma 1. *Web manifesti -esimerkki (Grigsby, 2018).*

Kun käyttäjä asentaa esimerkin mukaisen PWA:n mobiililaitteelleen, kotinäytölle lisätään applikaation ikoni, sekä ikonin alle applikaation lyhytnimi ”EWA”. Kun käyttäjä painaa ikonia kotinäytöllä, applikaatio avautuu omaan ikkunaansa ilman selaimen käyttöliittymää. Tämä toiminnallisuus määritellään manifestin display -kentän arvolla standalone.

Web manifesti lisätään verkkosivun otsakkeeseen ohjelman 2 mukaisena linkkielementtinä.

```
<link rel="manifest" href="/manifest.json">
```
(2)

Tämä linkki kertoo selaimille ja hakukoneille, mistä PWA:n web manifesti löytyy. (Grigsby, 2018)

4. TUNNISTETUT HYÖDYT

Edellä kuvattiin progressiivisten web-sovelluksien tunnuspiirteitä ja teknisiä ominaisuuksia. Kehittäjän kannalta kiinnostavaa on se, että mitä hyötyjä progressiivinen web-sovellus tarjoaa natiiviin sovellukseen tai perinteiseen verkkosovellukseen verrattuna. Khan et al. (2019) mukaan PWA oli paras vaihtoehto sovellustyypiksi, kun painotettuina mittareina olivat asennuksen koko, offline-ominaisuudet ja tuki useammalle laitealustalle. Tässä luvussa verrataan progressiivisia web sovelluksia perinteisiin verkkosovelluksiin sekä natiiveihin sovelluksiin. Tässä luvussa esitetään tekijöitä, joissa PWA suoriutuu paremmin kuin muut vertailun kohteet. Osassa kohdista progressiivisen web-sovelluksen ja perinteisen verkkosovelluksen välillä ei ole mitään eroa, joten vertailua näiden välillä ei suoriteta.

4.1 Hyödyt verrattuna natiiviin sovellukseen

Natiivit sovellukset kirjoitetaan yhdelle laitealustalle ja ne pystyvät hyödyntämään kyseisen laitteen koko suorituskykyä ja kaikkia ominaisuuksia. Natiiveilla sovelluksilla päästään käsiksi mm. laitteen tiedostojärjestelmään, laitteen portteihin liitettyihin laitteisiin ja laitteen kontakteihin sekä kalenteriin. Natiivit sovellukset tuntuvat osalta laitetta. Mobiililaitteille rakennettuja natiiveja sovelluksia jaetaan pääasiassa alustan oman kauppapaikan kautta. (LePage & Richard, 2020)

4.1.1 Asennettavuus

Googlen käyttäjätutkimuksen (2019) mukaan 50 prosenttia mobiililaitteiden käyttäjistä valitsevat yrityksen nettisivun natiivin sovelluksen sijaan, sillä he eivät halua asentaa sovellusta laitteeseensa. Headyn markkinatutkimuksen (Khosla, 2021) mukaan yli 90 prosenttia käyttäjistä turhautuvat, jos heidän on asennettava sovellus käyttääkseen yrityksen palveluita ja yli 54 prosenttia käyttäjistä jättää sovelluksen asentamatta. Toisaalta mobiililaitteelle asennetun sovelluksen käyttäjät tuottavat yrityksille enemmän liikevaihtoa, kuin verkkosivulla vierailevat asiakkaat (Fourault, 2022).

Progressiiviset web-sovellukset yhdistävät näiden molempien tapojen hyviä puolia. Käyttäjälle voidaan ehdottaa PWA:n asentamista laitteelle, mutta asentaminen tai asentamatta jättäminen ei vaikuta sovelluksen ominaisuuksiin. Progressiivisen web-sovelluksen asentaminen on myös helpompaa, kuin natiivin sovelluksen. PWA:n asentaminen

vaatii käyttäjältä vain yhden painalluksen, kun natiivin sovelluksen asentamiseen vaaditaan vähintään kolme toimintoa (Fourault, 2022). Web-sivulle lisätty banneri, joka ehdottaa asentamaan PWA:n mobiililaitteeseen, tuottaa asennuksen 5–6 kertaa useammin, kuin banneri, jossa ehdotetaan asentamaan natiivi sovellus sovelluskaupasta (Russell, 2017).

Joillekin yrityksille voi olla arvokasta saada sovellus tarjolle myös vakiintuneisiin kauppapaikkoihin näkyvyyden, saatavuuden tai sovelluskaupan luoman luottamuksen takia (Siavosh, 2019). Microsoftin avoimen lähdekoodin työkalulla, PWABuilderilla on mahdollista paketoida PWA natiiviksi sovellukseksi mm. Androidille, iOS:lle, macOS:lle ja Windowsille (Wargo, 2020). Tällä tavoin paketoitu PWA voidaan tuoda tarjolle myös laitekohtaisille kauppapaikoille. Näin ei voi kuitenkaan toimia ei-julkisten, kuten yrityksen sisäisten verkkosovelluksien kanssa, koska kauppapaikka ei pysty tarkistamaan sisäisiä sovelluksia ennen julkaisua (Firtman, 2020). PWA:n muuntaminen natiiviksi paketiksi onnistuu alle tunnissa (Naukkarinen, 2022).

4.1.2 Tallennustila

Googlen markkinatutkimuksessa (2016) selvisi, että liian suuri tallennustilan käyttö oli suurin syy sovelluksen poistamiselle mobiililaitteesta. Etenkin kehittyvillä markkinoilla mobiililaitteiden tallennuskapasitetti ja tiedonsiirto on rajallista, minkä takia sovelluksen tulisi käyttää mahdollisimman vähän tallennustilaa. Tyypillisen 20 MB kokoisen natiivin sovelluksen lataaminen 2G-tasoisen yhteyden yli voi kestää jopa 30 minuuttia ja lataaminen epäonnistuu usein. (Roy, 2016)

Progressiivisen web sovelluksen rakentaminen vaihtoehdoksi natiiville sovellukselle on tuonut merkittäviä säästöjä sovelluksen käyttämään tallennustilaan. Säästöt ovat peräisin siitä, että web sovelluksen ajoympäristöä ei tarvitse sisällyttää mukaan sovellukseen, toisin kuin monissa natiiveissa sovelluksissa, sillä PWA:n ajoympäristö on laitteeseen asennettu selain.

Twitterin PWA käyttää 97% vähemmän tallennustilaa, kuin Twitterin natiivi mobiiliapplikaatio (Grzybowska, 2020). Khan et al. (2019) mukaan keskimääräisen Android-applikaation koko on 9.6 MB, iOS-applikaation koko on 56 MB ja PWA:n koko on 150 KB. Bjørn-Hansen et al. tutkimuksissa (2017, 2018) PWA oli 42-43 kertaa pienempi, kuin natiivi sovellus. Tutkimusten perusteella hyödyt tallennustilan käytössä ovat merkittäviä.

4.2 Hyödyt verrattuna perinteiseen web-sovellukseen

Perinteisiä verkkosovelluksia ajetaan selaimessa, joten verkkosovellusta voidaan käyttää lähes millä tahansa laitteella, jossa on verkkoyhteys, näyttö ja selain. Verkkosovelluksia ei hallitse mikään yksittäinen yritys, joten pääsy verkkosovelluksiin on vapaata. Verkkosovelluksien etuna on myös laaja ekosysteemi, jonka hedelmiä kehittäjät voivat hyödyntää sovelluksissaan. (LePage & Richard, 2020)

4.2.1 Nopeus

Kaupallisille verkkosivuille olennainen metriikka, välitön poistumisprosentti (engl. bounce rate) kertoo prosenttiosuuden käyttäjistä, jotka saapuvat sivustolle ja poistuvat siirtymättä toiseen alasivuun (Palomäki, 2022). Verkkosivun latautumisen nopeus on merkittävä välittömään poistumisprosenttiin vaikuttava tekijä sillä välitön poistumisprosentti lähtee jyrkkään nousuun heti, kun verkkosivun lataaminen kestää yli 3 sekuntia (Pingdom, 2018). Ericssonin tutkimuksen (2016) mukaan verkkosovelluksen hidastelu aiheuttaa myös merkittävää nousua mobiililaitteiden käyttäjien sykkeessä ja stressitasossa.

Palvelutyöläisen ominaisuudet, kuten välimuistin hyödyntäminen välityspalvelimena tekevät progressiivisista web-sovelluksien latautumisesta merkittävästi nopeampaa (Malavolta et al., 2020). PWA:n kyvykkyyksien menestyksekäs hyödyntäminen on olennaista verkkosivun nopeuttamisessa. Boudreau (2019) totesi, että PWA:t ovat keskimäärin 43 prosenttia nopeampia perinteisiin verkkosivuihin verrattuna, mutta toisessa tutkimuksessa (Pande et al., 2018) heikosti totutettu PWA:n ominaisuuksien hyödyntäminen toi vain 2-10%:n nopeutuksen perinteiseen verrattuna. Natiivin sovelluksen latautumisnopeus riippuu sen toteutustavasta, joten sitä ei ole järkevää verrata tässä mielessä.

4.2.2 Datan käyttö

Perinteiset web-sovellukset eivät pysty säilömään dataa käyttäjän laitteelle, mutta joitakin web-sovellukseen liittyviä tiedostoja selain voi tallentaa välimuistiin. Progressiivisen web-sovelluksen palvelutyöläinen mahdollistaa laajan välimuistin käytön, mikä pienentää sovelluksen käyttöön vaadittavan datansiirron määrää (Khan et al., 2019). Yhdessä tutkimuksessa mitattiin 25:n verkkosivun dataliikennettä kuukauden ajan ja huomattiin, että PWA:n implementointi toi 25 %:n säästön datankulutukseen (Pande et al., 2018).

4.2.3 Ominaisuudet

Perinteistä web-sovellusta ei voi käynnistää ilman verkkoyhteyttä, mutta PWA:ssa tämä on mahdollista palvelutyöläisen varastoimien resurssien avulla. Palvelutyöläinen mahdollistaa myös taustasynkronoinnin, missä data siirtyy sovelluksen ja palvelimen välillä, kun sovellus ei ole aktiivisena (Biørn-Hansen et al., 2018). Esimerkiksi sosiaalisen median sovellukseen ladattava kuva säilötään palvelutyöläisen avulla, jos verkkoyhteyttä ei ole saatavilla. Palvelutyöläinen lataa kuvan palvelimelle myöhemmin verkkoyhteyden palattua. Palvelutyöläinen voi myös tasaisin väliajoin hakea uusia postauksia palvelimelta, jolloin käyttäjä saa avatessaan välittömästi nähtävilleen sisältöä, vaikka verkkoyhteyttä ei sillä hetkellä olisi saatavilla.

Googlen markkinatutkimuksen mukaan 85 % mobiililaitteidenkäyttäjistä pitää push-notifikaatioita hyödyllisinä. Push-notifikaatiot eivät ole mahdollisia perinteisessä verkkosovelluksessa, mutta palvelutyöläinen mahdollistaa notifikaatiot progressiivisissa websovelluksissa (Fourault, 2022). iOS-ympäristössä tätä tukea ei vielä ole, mutta se on tulossa vuonna 2023 (Apple, 2022).

5. TUNNISTETUT RAJOITTEET

Progressiivisiin web-sovelluksiin liittyy myös merkittäviä haasteita ja rajoitteita, mitkä esitetään tässä luvussa.

5.1 Ominaisuudet ja laskentateho

Natiiveilla sovelluksilla on suora pääsy sellaisiin laitteen ominaisuuksiin, joihin web-sovellukset eivät pääse käsiksi. Tällaisia ovat esimerkiksi laitteen tiedostojärjestelmä ja monet laitteen portteihin kytketyt lisälaitteet (Grigsby, 2018). Jos sovelluksen on päästävä käsiksi tällaisiin ominaisuuksiin, ei sitä voi toteuttaa progressiivisena web sovelluksena.

Natiivit sovellukset pystyvät myös hyödyntämään paremmin laitteen koko laskentatehon, kun suuri osa verkkosovelluksista toteutetaan JavaScriptillä, joka ajetaan yhdessä säikeessä. WebAssembly teoriassa mahdollistaa käännetyn ohjelmointikielen tuoman tehokkuuden myös verkkosovelluksille, mutta empiirisessä tutkimuksessa tulokset vaihtelevat 30 % tehokkuusparannuksista (De Macedo et al., 2022) jopa heikompaan suorituskykyyn mobiililaitteilla (Asegehegn, 2022). PWA:t eivät voi myöskään kommunikoida suoraan muiden laitteeseen asennettujen sovellusten kanssa.

5.2 Tuki

Applen iOS ja Googlen Android ovat merkittävimmät mobiililaitealustat. Näistä Android tukee suurta osaa moderneista web-teknologioista, mutta iOS:n tuki joillekin keskeisille ominaisuuksille on puutteellista. iOS ei tue push-notifikaatioita, eikä taustasynkronointia.

Apple on tunnettu siitä, että se suodattaa tarkasti kaikki sen sovelluskauppaan App Storeen tulevat sovellukset. Apple myös ottaa App storen myynnistä merkittävän 30 %:n osuuden (Ambrasaitė & Smagurauskaitė, 2021). App Storen liikevaihto oli 85.1 miljardia dollaria vuonna 2021. Googlen play storen liikevaihto oli 47.9 miljardia dollaria (Iqbal, 2022). Progressiiviset web-sovellukset tarjoavat kehittäjille mahdollisuuden huolehtia itse sovelluksen levittämisestä ja siten kiertää nämä kauppapaikat.

Apple ei ole ollut innokas parantamaan tukea kaikille PWA:n ominaisuuksille, koska tämä kehitys voisi rajoittaa App Storen tuomaa liikevaihtoa. Applen merkittävään markkina-asemaan ja vallankäyttöön App Storen kautta liittyy myös useita oikeudenkäyntejä, joissa Applea syytetään monopoliaseman väärinkäytöstä ja kilpailun rajoittamisesta (Ambrasaitė & Smagurauskaitė, 2021; Liptak & Nicas, 2019; Satariano & Nicas, 2019).

5.3 Uudet laitteet

Mobiililaitteiden jatkuva kehitys on tuonut markkinoille uusia laitetyyppejä (Rieger & Majchrzak, 2018). Uudet laitteet tuovat mukanaan uusia mahdollisuuksia sovelluskehitykselle, mutta PWA:t ovat riippuvaisia selaimesta, jolla on tuki PWA:n ominaisuuksille. Puettavista laitteista, kuten älykelloista selain puuttuu, jolloin PWA:n hyödyntäminen niissä on mahdotonta (Rieger & Majchrzak, 2019).

5.4 Turvallisuus

Progressiiviset web-sovellukset käyttävät selainta ajoalustana, mikä tekee PWA:n alttiiksi selaimen haavoittuvuuksille. Progressiivisten web-sovellusten palvelutyöläisestä ja suositusta push-notifikaatioiden toteutustavasta on löytynyt heikkouksia, joiden avulla on onnistuttu käyttämään hyväksi käyttäjän laitetta (Lee et al., 2018). Mukhopadhyay ja Ghosh totesivat tutkimuksessaan (Mukhopadhyay & Ghosh, 2021), että PWA:n turvallisuusominaisuudet eivät ole riittävän kehittyneitä pankkisovelluksen kehittämiseen.

6. YHTEENVETO

Tämän kirjallisuuskatsauksen tavoitteena oli selvittää, miten progressiivinen web-sovellus määritellään kirjallisuudessa ja miten progressiiviset web-sovellukset vertautuvat perinteisiin verkkosovelluksiin ja natiiveihin sovelluksiin. Progressiivisen web-sovelluksen tarkka määritteleminen osoittautui hankalaksi vuosien saatossa muuttuneiden ja epämääräisten määritelmien vuoksi. Tämän tutkielman kontekstissa PWA:n määritelmäksi valittiin salatun yhteyden yli tarjoiltu verkkosovellus, joka hyödyntää palvelutyöläistä ja jolla on web manifesti.

Tutkielman perusteella voidaan sanoa, että progressiivinen web sovellus on monissa yksinkertaisissa sovelluksissa jopa paras vaihtoehto sovelluksen toteuttamiseen sen pienen koon, kohtalaisen suorituskyvyn ja kehittyneiden kyvykkyyksien perusteella. PWA:n hyödynnettävyyteen vaikuttaa olennaisesti sovelluksen nykytila: jos sovellus on valmiiksi toteutettu natiivina, sen muuttaminen PWA:ksi ei tuo merkittäviä hyötyjä kustannuksiin verrattuna. Perinteisen verkkosivun muuttaminen PWA:ksi onnistuu usein maltillisilla kustannuksilla ja muutoksen tuomat hyödyt voivat olla merkittäviä.

Sovelluskehityksen alkuvaiheessa PWA:n sopiminen sovellustyypiksi riippuu sovelluksen vaatimuksista. Natiivi sovellus on usein ainoa vaihtoehto niissä tapauksissa, joissa vaaditaan korkeaa laskentatehoa, tai jotakin ominaisuutta, jota progressiiviset web-sovellukset eivät tue. Tutkielman perusteella progressiivisen web-sovelluksen hyödyt tulevat parhaiten esille kehittyvillä markkinoilla, missä tilankäyttö ja datankäyttö ovat ratkaisevassa osassa sovelluksen menestyksessä. Useat yritykset ovat saavuttanee taloudellista hyötyä toteuttamalla kevyemmän progressiivisen web-sovelluksen vaihtoehdoksi natiivin sovelluksen rinnalle.

Progressiivisten web sovellusten heikkouksiksi paljastui vaihteleva tuki eri alustojen välillä. Applen iOS-käyttöjärjestelmän puutteellinen tuki keskeisille ominaisuuksille haittaa PWA:n hyödyntämistä toisella merkittävimmistä mobiilialustoista, eikä ole varmuutta siitä tuleeko iOS tukemaan näitä ominaisuuksia tulevaisuudessakaan.

LÄHTEET

Ambrasaitė, P., & Smagurauskaitė, A. (2021). Epic Games v. Apple: Fortnite battle that can change the industry. *Vilnius University Open Series*, 6–25. https://doi.org/10.15388/TMP.2021.1

Apple. (2022). *iOS 16—Uudet ominaisuudet*. Apple (Suomi). https://www.apple.com/fi/ios/ios-16/features/

Asegehegn, N. T. (2022). *Evaluation of Rust and WebAssembly when building a Progressive Web Application: An analysis of performance and memory usage*. https://www.diva-portal.org/smash/get/diva2:1668630/FULLTEXT01.pdf

Basques, K. (2020, April 7). *Why HTTPS matters*. Web.Dev. https://web.dev/why-https-matters/

Berriman, F. (2017, June 26). *Naming Progressive Web Apps*. Fberriman.Com. https://fberriman.com/2017/06/26/naming-progressive-web-apps/

Berriman, F., & Russel, A. (2015, June 15). *Progressive Web Apps: Escaping Tabs Without Losing Our Soul*. Infrequently Noted. https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/

Biørn-Hansen, A., Majchrzak, T. A., & Grønli, T.-M. (2017). Progressive web apps: The possible web-native unifier for mobile development. *International Conference on Web Information Systems and Technologies*, *2*, 344–351.

Biørn-Hansen, A., Majchrzak, T. A., & Grønli, T.-M. (2018). *Progressive Web Apps for the Unified Development of Mobile Applications*. 64–86. https://doi.org/10.1007/978-3-319-93527-0_4

Boudreau, I. (2019, December 23). *Progressive Web Apps: A Way To Speed Page Load Time?* https://www.directivegroup.com/digital-marketing/progressive-web-apps/

Chrome Dev Summit. (2017, October 24). *The New Bar for Web Experiences*. https://www.youtube.com/watch?v=PsgW-0M67TQ

De Macedo, J., Abreu, R., Pereira, R., & Saraiva, J. (2022). WebAssembly versus JavaScript: Energy and Runtime Performance. *2022 International Conference on ICT for Sustainability (ICT4S)*, 24–34. https://doi.org/10.1109/ICT4S55073.2022.00014

Ericcson. (2016). *Ericsson Mobility Report* (p. 12). https://www.iot.gen.tr/wp-content/uploads/2016/08/160307-Ericsson-mobile-report-MWC-Update-edition.pdf

Fauzan, R., Krisnahati, I., Nurwibowo, B. D., & Wibowo, D. A. (2022). A Systematic Literature Review on Progressive Web Application Practice and Challenges. *IPTEK The Journal for Technology and Science*, *33*(1), Article 1. https://doi.org/10.12962/j20882033.v33i1.13904

Firtman, M. (2020, August 18). Google Play Store now open for Progressive Web Apps 😱. *Medium*. https://medium.com/@firt/google-play-store-now-open-for-progressive-web-apps-ec6f3c6ff3cc

Fourault, S. (2022, August 9). *How Progressive Web Apps can drive business success*. Web.Dev. https://web.dev/drive-business-success/

Geddes, D. (2019, June 4). *Service worker mindset*. Web.Dev. https://web.dev/service-worker-mindset/

Google. (2016). *How people use their phones for travel*. Google. https://www.thinkwithgoogle.com/_qs/documents/79/app-marketing-travel-consumer-journey.pdf

Google. (2019, January 1). *Smartphone user mobile shopping preferences*. Think with Google. https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/smartphone-user-mobile-shopping-preferences/

Grigsby, J. (2018). *Progressive Web App*. A Book Apart.

Grzybowska, K. (2020, November 1). *20+ companies that use PWA and how it works for them*. Divante. https://www.divante.com/blog/companies-that-use-pwa

Hume, D. A., & Osmani, A. (2018). *Progressive web apps*. Manning Publications.

Iqbal, M. (2022). *App Revenue Data (2022)*. https://www.businessofapps.com/data/app-revenues/

Keith, J. (2017, November 15). *What is a Progressive Web App?* https://adactio.com/journal/13098

Khan, A. I., Al-Badi, A., & Al-Kindi, M. (2019). Progressive Web Application Assessment Using AHP. *Procedia Computer Science*, *155*, 289–294. https://doi.org/10.1016/j.procs.2019.08.041

Khosla, R. (2021, February 1). *Market Study: Mobile Customer Experience Issues Highlight Use Cases for iOS App Clips*. https://www.heady.io/blog/market-study-mobile-customer-experience-issues-highlight-use-cases-for-ios-app-clips

Lee, J., Kim, H., Park, J., Shin, I., & Son, S. (2018). Pride and Prejudice in Progressive Web Apps: Abusing Native App-like Features in Web Applications. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 1731–1746. https://doi.org/10.1145/3243734.3243867

LePage, P., Beaufort, F., & Steiner, T. (2022, September 14). *Add a web app manifest*. Web.Dev. https://web.dev/add-manifest/

LePage, P., & Richard, S. (2020, June 9). *What are Progressive Web Apps?* Web.Dev. https://web.dev/what-are-pwas/

Liptak, A., & Nicas, J. (2019, May 13). Supreme Court Allows Antitrust Lawsuit Against Apple to Proceed. *The New York Times*. https://www.nytimes.com/2019/05/13/us/politics/supreme-court-antitrust-apple.html

Malavolta, I., Chinnappan, K., Jasmontas, L., Gupta, S., & Soltany, K. A. K. (2020). Evaluating the impact of caching on the energy consumption and performance of progressive web apps. *Proceedings of the IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems*, 109–119. https://doi.org/10.1145/3387905.3388593

Mukhopadhyay, I., & Ghosh, A. (2021). Banking Transaction Considerations on Standardized Browser Architecture. *Proceedings of International Conference on Computational Intelligence, Data Science and Cloud Computing*, 635–644. https://doi.org/10.1007/978-981-33-4968-1_49

Naukkarinen, T. (2022). *Selvitys erilaisista vaihtoehdoista muuntaa PWA-sovellus APK-paketiksi* [Jyväskylän ammattikorkeakoulu]. https://www.theseus.fi/bitstream/handle/10024/754029/Opinnaytetyo_Naukkarinen_Tomi.pdf

Palomäki, J. (2022). *Hakukoneoptimointi: Verkkosivujen hakukonenäkyvyyden kehittäminen ohjelmistoyritykselle* [Fi=AMK-opinnäytetyö|sv=YH-examensarbete|en=Bachelor's thesis|]. http://www.theseus.fi/handle/10024/725590

Pande, N., Somani, A., Prasad Samal, S., & Kakkirala, V. (2018). Enhanced Web Application and Browsing Performance through Service-Worker Infusion Framework. *2018 IEEE International Conference on Web Services (ICWS)*, 195–202. https://doi.org/10.1109/ICWS.2018.00032

Pingdom. (2018, January 18). *Does Page Load Time Really Affect Bounce Rate?* Pingdom.Com. https://www.pingdom.com/blog/page-load-time-really-affect-bounce-rate/

Rieger, C., & Majchrzak, T. A. (2018). A Taxonomy for App-Enabled Devices: Mastering the Mobile Device Jungle. In T. A. Majchrzak, P. Traverso, K.-H. Krempels, & V. Monfort (Eds.), *Web Information Systems and Technologies* (pp. 202–220). Springer International Publishing. https://doi.org/10.1007/978-3-319-93527-0_10

Rieger, C., & Majchrzak, T. A. (2019). Towards the definitive evaluation framework for cross-platform app development approaches. *Journal of Systems and Software*, *153*, 175–199. https://doi.org/10.1016/j.jss.2019.04.001

Roy, G. (2016, March 9). How we built Facebook Lite for every Android phone and network. *Engineering at Meta*. https://engineering.fb.com/2016/03/09/android/how-we-built-facebook-lite-for-every-android-phone-and-network/

Russell, A. (2017, May 8). Why Are App Install Banners Still A Thing? *Dev Channel*. https://medium.com/dev-channel/why-are-app-install-banners-still-a-thing-18f3952d349a

Satariano, A., & Nicas, J. (2019, March 13). Spotify Accuses Apple of Anticompetitive Practices in Europe. *The New York Times*. https://www.nytimes.com/2019/03/13/business/spotify-apple-complaint.html

Selovuo, K. (2013, November 6). Mobile First—Simple First. *Corellia Helsinki Oy*. https://corellia.fi/mobile-first-simple-first/

Siavosh, K. (2019, September 4). why use The PWA? What are Advantages and Disadvantages of PWA? part 2. *Avenger IT Next Generation*. https://avengering.com/en/why-use-the-pwa-what-are-advantages-and-disad-vantagesof-pwa-part-2/

Wargo, J. M. (2020). *Learning Progressive Web Apps: Building Modern Web Apps Using Service Workers*. Addison-Wesley Professional.