Progressiivisen verkkosovelluksen soveltuvuus alustariippumattomana sovellustyyppinä

Oulun yliopisto
Tietojenkäsittelytiede
LuK-tutkielma
Konsta Lahtela
2023

Tiivistelmä

Alustariippumattomuus viittaa ohjelmiston kykyyn toimia eri alustoilla ilman tarvetta kehittää sitä alustoille erikseen. Tämä on hyvin tehokas tapa tehdä sovelluksesta mahdollisimman saatavilla oleva. Perinteisesti alustariippumaton ohjelmistotuote on toteutettu joko verkkosovelluksena tai hybridisovelluksena, mutta näiden lisäksi on kehitetty uusi alustariippumaton sovellustyyppi nimeltä progressiiviset verkkosovellukset (PWA).

Tässä tutkielmassa tutkittiin PWA:n soveltuvuutta alustariippumattomana sovellustyyppinä vertailemalla sitä muihin alustariippumattomiin sovellustyyppeihin. Tutkielmassa käytettiin tutkimusmenetelmänä kirjallisuuskatsausta, ja se tehtiin Scopus-tietokannan avulla. Kirjallisuuskatsauksen avulla tutkielmaan löydettiin 22 tieteellistä artikkelia. Tutkielman kannalta merkittävimmiksi lähdeteoksiksi ilmenivät Biørn-Hansenin, Majchrzakin & Grønlin (2017), sekä Malavoltan (2016) artikkelit, joissa käytiin sovellustyyppejä yksityiskohtaisesti läpi sekä vertailtiin PWA:n ja muiden sovellustyyppien eroavaisuuksia.

Tutkielmassa pyrittiin löytämään vastaus ongelmaan, jossa alustariippumatonta ohjelmistotuotetta harvoin tuotetaan PWA:na, vaikka sillä on tiedettävästi huomattavia etuja verrattuna muihin alustariippumattomiin sovellustyyppeihin. Tämä toteutettiin tutkimalla PWA:n vahvuuksia ja heikkouksia ja vertaamalla niitä muihin alustariippumattomiin sovellustyyppeihin. Aiempien tutkimusten perusteella vahvuuksiksi ilmenivät hyvä tehokkuuden-, saatavuuden-, ja tietoturvan taso, laaja potentiaali löydettävyydelle, ja pieni asennuskoko. Heikkouksiksi ilmeni tiettyjen toimintojen puute, riippuvuus selaimista, ja PWA teknologioiden kypsyyden matala taso. Tutkielman tuloksista selvisi, että PWA:n soveltuvuus riippuu pitkälti sen käyttötapauksista, ja on yhä melko riskialtista johtuen sen tuoreudesta.

Koska aihepiiri on melko laaja, ja PWA sekä muut alustariippumattomat sovellustyypit kehittyvät jatkuvasti, on aiheeseen liittyvälle jatkotutkimukselle paljon tarvetta. Erityisesti jatkotutkimus liittyen PWA:n ainutlaatuisiin ominaisuuksiin, sekä yksityiskohtainen ja vertaileva tutkimus PWA:n sekä hybridisovellusten sisäisten eroavaisuuksien välillä on hyvin tarpeellista.

Avainsanat
Progressiivinen verkkosovellus, PWA, verkkosovellus, hybridisovellus, alustariippumattomuus, alustariippumaton sovellustyyppi

Ohjaaja
Yliopistonlehtori Karin Väyrynen

Sisällysluettelo

1. Johdanto

Alustariippumattomien teknologioiden käyttö on hyvin tehokas tapa tuoda ohjelmistotuote mahdollisimman laajalle käyttäjäkannalle. Perinteisesti alustariippumattomiin sovelluksiin kuuluvat hybridi- ja verkkopohjaisia ohjelmistoarkkitehtuureita noudattavat sovellukset, eli hybridi- ja verkkosovellukset (Khan, Al-Badi & Al-Kindi, 2019). He esittävät, että näiden sovellusten lisäksi vuonna 2015 kehitettiin uusi sovellustyyppi nimeltä progressiiviset verkkosovellukset (eng. Progressive Web Application, PWA). PWA:t ovat verkkopohjaisia sovelluksia, jotka pyrkivät tuomaan lisäarvoa perinteisille verkkosovelluksille mukauttamalla ulkoasuaan, ja toimimaan natiivien sovellusten kaltaisesti (Diekmann & Eggbert, 2021). PWA:ta voidaankin kuvailla verkkosovelluksen päälle rakennetuksi lisäosaksi, jonka avulla verkkosovellukselle voidaan mahdollistaa uusia, natiiveille sovelluksille tyypillisiä toiminnallisuuksia (Huber, Demetz & Felderer, 2021).

PWA on kuitenkin vielä tuore ja kehitysvaiheessa oleva sovellustyyppi, ja käsite itsessään on vielä melko tuntematon niin kehittäjille kuin käyttäjillekin, joka luo PWA:n kehitykseen ja käyttöönottoon liittyen paljon epävarmuutta. Tämän tutkimuksen tarkoituksena on selvittää PWA:n soveltuvuus alustariippumattomana ohjelmistotuotteena vertailemalla sen arkkitehtuuria ja ominaisuuksia muihin alustariippumattomiin sovellustyyppeihin, joihin kuuluu hybridi- ja verkkosovellukset. Tutkielmassa hybridisovelluksiin sisällytetään verkkopohjaiset hybridisovellukset, sekä alustariippumattomat hybridisovellukset. Tämä laajan tason rajaus johtuu tutkielman rajatusta pituudesta, verkko- ja alustariippumattomien hybridisovellusten avonaisesta ja täten vaikeasti tehdystä määrittelystä aikaisemmassa kirjallisuudessa, sekä siitä, että tämä sisällytys on tehty useassa kirjallisuuskatsauksen avulla löydetyssä artikkelissa samalla tavalla. Verkkosovelluksiin sisällytetään tutkielmassa tavalliset verkkosovellukset ja mobiiliverkkosovellukset. Tämän lisäksi tutkielmassa rajataan täysin pois natiivit sovellukset, koska ne eivät ole alustariippumattomia.

Aihetta on tärkeä tutkia, koska PWA:lla on paljon potentiaalia, ja se voi tulevaisuudessa olla hyvinkin käytetty ja tehokas sovellustyyppi alustariippumattomassa ohjelmistokehityksessä. Tämän takia aiheesta tietoisuus on erityisen tärkeää ohjelmistokehittäjille ja -yrityksille, jotka ovat kiinnostuneita alustariippumattomasta ohjelmistokehityksestä. Aiheesta saattaa myös olla hyötyä PWA:n käyttäjille, jotka haluavat lisätietoa PWA:n käytöstä, sekä sen hyödyistä ja haasteista verrattuna muihin sovellustyyppeihin. Tutkielma voi hyödyttää myös aiheesta kiinnostuneita tutkijoita, ja koska aihealue on vielä melko tuore, sekä jatkuvan kehityksen alla, on aiheen jatkotutkimukselle paljon tarvetta liittyen muun muassa PWA:n yksityiskohtaisiin ominaisuuksiin, käyttäjien suhtautumiseen PWA:n käyttöön, PWA:n vertailuun muihin sovellustyyppeihin, sekä PWA teknologioiden kehityksen edistymiseen.

Tutkielman tutkimusongelma on se, että vaikka PWA arkkitehtuurin sovelluksilla on paljon etuja ja potentiaalia, on se silti vielä melko tuntematon ja vähän hyödynnetty sovellustyyppi. Tutkielmassa pyritään vastaamaan tutkimuskysymykseen: millaisia etuja ja haasteita ilmenee PWA ohjelmistotuotteen kehittämisessä ja käyttöönotossa verrattuna muihin alustariippumattomiin ohjelmistoratkaisuihin?

Tämä tutkielma koostuu viidestä kappaleesta. Nykyinen kappale, eli kappale yksi, toimii johdantona tutkielman aiheeseen. Kappaleessa kaksi käydään läpi tutkielmassa käytetty tutkimusmenetelmä. Kappaleessa kolme käydään aikaisemman tutkimuksen perusteella läpi aiheen terminologia, alustariippumattomuuden määritelmä, PWA ja muut

alustariippumattomat sovellustyypit yleisellä tasolla, sekä sovellustyyppien yksityiskohtaiset eroavaisuudet. Kappaleessa neljä pohditaan aiempien tutkimuksien tuloksia. Lopulta viidennessä kappaleessa käydään tutkielma yhteenvetoisesti läpi.

2. Tutkimusmenetelmä

Tutkielmassa käytetty tutkimusmenetelmä on kirjallisuuskatsaus, joka tarkoittaa, että tutkielmassa ei tehdä ollenkaan omaa empiiristä tutkimusta, vaan tutkimus perustuu täysin olemassa olevaan aikaisempaan tutkimukseen. Tutkielmassa käytetty kirjallisuus haettiin Scopus tietokannasta. Haku tehtiin 13.3.2023, ja haun tulokset tallennettiin Covidence nimiseen verkkopalveluun. Scopuksessa aineiston hakemiseen käytettiin hakulauseketta "Progressive web application" OR "Progressive web app". Tällä hakulausekkeella ei rajattu aihetta PWA:ta enempää, koska alustariippumattomuus on jo itsessään PWA:n ominaisuus, ei hakutermiin tarvitse sisällyttää alustariippumattomuuden näkökulmaa erikseen. Tällä hakulausekkeella löytyi 168 artikkelia. Ne järjesteltiin Covidence-palvelussa osuvuuden mukaan, jonka jälkeen artikkeleihin tehtiin pikainen katsaus käymällä läpi niiden tiivistelmä, sekä tarvittaessa tarkasteltiin myös artikkelin sisältö laajemmin. Artikkeleista valittiin sellaiset, joissa ilmeni vertailua sovellustyyppien välillä, tai yksityiskohtaista tietoa sovellustyypeistä. Artikkeleita käytiin läpi siihen asti, kunnes tutkielmaan saatiin tarpeeksi kirjallisuutta, sekä havaittiin, että kirjallisuudessa esiintyvät alustariippumattomat sovellustyypit, sekä niissä esiintyvät eroavat ominaisuudet tulivat selkeästi ilmi, eikä uusista läpikäydyistä artikkeleista enää löytynyt tutkielman kannalta oleellista tietoa. Yhteensä käytiin läpi 52 eri artikkelia, jonka pohjalta tutkielmaan valiintui 22 artikkelia. Tämän lisäksi tutkielman ohjaaja ehdotti kahta artikkelia liittyen tutkimusmenetelmiin, jotka otettiin tutkielmaan mukaan. Tällöin tutkielmaan sisältyi yhteensä 24 artikkelia.

Kirjallisuuskatsaus toteutettiin käyttäen scoping review menetelmää. Munn, Peters, Stern, Tufanaru, McArthur & Aromataris (2018) määrittelevät scoping review menetelmän olevan erityisen hyödyllinen tilanteissa, joissa kirjallisuuskatsauksen tavoite on tutkia kirjallisuutta ja niissä käytettyjä tutkimusmenetelmiä yleisellä tasolla, kartoittaa terminologiaa ja tärkeimpiä käsitteitä, ja tunnistaa kirjallisuuden aihepiirin puutteellisuutta. Se tyypillisesti keskittyy laajempaan aihekokonaisuuteen (Arksey & O'Malley, 2005). Toinen kirjallisuuskatsauksen toteuttamismenetelmä on systematic review (Munn ym., 2018). Sille tyypillisiä piirteitä on se, että siinä keskitytään tarkasti rajatun tutkimuskysymyksen vastaamiseen, jossa tutkimus ja sen suunnittelu voidaan pitkälti määritellä etukäteen. (Arksey & O'Malley, 2005). Scoping review valiintui tämän tutkielman kirjallisuuskatsauksen menetelmäksi, sillä tutkielman aihepiiri oli tutkielman alussa itselleni vielä melko vieras, jolloin tutkimuksen tarkkaa suunnittelua etukäteen oli vaikea tehdä. Tämän lisäksi koska tutkielman aihepiiri on yhä melko tuore, soveltui scoping review menetelmä paremmin tutkielman toteuttamiseen. Scoping review menetelmän avulla pystyttiin tekemään yleiskatsaus aiheeseen, selvittämään tärkeimmät käsitteet ja aihepiirin kokonaisuudet, sekä löytämään puutteita aihepiirin aiemmassa kirjallisuudessa.

Kirjallisuuskatsauksen aikana esiin tuli paljon sellaisia artikkeleita, joissa yksinkertaisesti dokumentoitiin PWA:n kehitysprosessia, jonka takia ne pitkälti jätettiin pois tutkielmasta, koska niissä ei tyypillisesti ilmennyt tutkielman kannalta oleellista tietoa. Kirjallisuuden etsiminen oli myös osittain haastavaa johtuen aiheen terminologian vaihtelevuudesta ja avoimuudesta. Lisäksi haasteita aiheutti se, että hybridisovellusten ominaisuuksista löytyi melko vähän tietoa verrattuna muihin sovellustyyppeihin, joka hidasti kirjallisuuskatsausta. Tämä johtui tutkielmassa tehdystä laajasta hybridisovellusten kategorisoinnista, jolloin kategoria sisälsi paljon sisäisiä eroja, jolloin yleisen määritelmän mukaista vertailua oli vähän. Esimerkiksi hybridisovellukset eivät rajoitu pelkästään sovelluskauppoihin kuten natiivit sovellukset, tai pelkästään selaimeen kuten

verkkosovellukset ja PWA:t, vaan ne voidaan toteuttaa toimimaan kummalla tahansa näistä, tai jopa molemmilla. Kaikki tämän tyyppiset kategorian sisäiset eroavaisuudet tekivät kirjallisuuden avulla hybridisovellusten yleisen tason ominaisuuksien määrittelystä melko haastavaa verrattuna muihin sovellustyyppeihin. Erityisesti hybridisovellusten yleisestä tietoturvan tasosta ja sen vertailusta PWA:n tietoturvan tasoon ei löytynyt aikaisemmasta kirjallisuudesta lainkaan.

3. Aiempi tutkimus

Tässä luvussa tarkastellaan aiempia tutkimuksia liittyen PWA:n ja muihin alustariippumattomiin sovellustyyppeihin, sekä erityisesti niiden vertailuun. Kappaleessa käydään läpi ensin aiheen terminologia ja siihen liittyvät haasteet. Tämän jälkeen käydään läpi yleiskatsaus alustariippumattomuudesta, sekä alustariippumattomista sovellustyypeistä, jonka jälkeen vertaillaan alustariippumattomien sovellustyyppien tekijöiden eroavaisuuksia. Lopuksi käydään tekijöiden eroavaisuudet yhteenvetona läpi.

3.1 Terminologia

Aiheeseen liittyvä terminologia on jossain määrin haasteellista, sillä suuri osa termeistä on hyvin avonaisia, ja määritelty eri tavoin eri tutkimuksissa. Tämän lisäksi sovellustyyppien erottelu ja kategorisointi oli eroavaa.

Tutkielmassa ilmeni myös haasteita liittyen terminologian tuoreuteen, sillä monelle aihepiirin termille, kuten muun muassa PWA:n komponenteille, ei ole olemassa suomenkielisiä käännöksiä. Tämä pakottaa välillä käyttämään englanninkielistä terminologiaa, joka hankaloittaa tekstin luettavuutta.

Toinen haaste oli termi ”cross-platform app”, joka esiintyi paljon kirjallisuudessa. Tällä termillä oli eroavaisuuksia eri tutkimuksissa. Diekmann & Eggbert (2021) viittaavat termillä kaikkiin alustariippumattomiin sovellustyyppeihin, mukaan lukien PWA:t, hybridisovellukset ja verkkosovellukset. Majchrzak, Biørn-Hansen & Grønli (2018) puolestaan viittaavat samalla termillä pelkästään alustariippumattomia frameworkkeja käyttäviin hybridisovelluksiin, eli alustariippumattomiin hybridisovelluksiin. Biørn-Hansen ym. (2017) puolestaan eivät luokittele verkkosovelluksia tai PWA:ta ollenkaan mukaan tähän ”cross-platform apps” sovelluskategoriaan. Tässä tutkielmassa termi ”cross-platform app”, eli alustariippumaton sovellus, viittaa ylempään kategoriaan, johon mukaan luetaan kaikki alustariippumattomat sovellukset. Majchrzak ym. (2018) käyttämän termin määrittelyyn viitataan tutkielmassa nimellä alustariippumaton hybridisovellus.

Koska aiheen terminologia oli melko vaihtelevaa, ongelmia ilmeni myös sovellustyyppi kategorioiden määrittelyssä. Jotkin tutkimukset määrittelivät ne vielä alemmalla tasolla. Muun muassa tutkimuksessaan Biørn-Hansen ym. (2017) vertailevat keskenään PWA:ta, hybridisovellusta, sekä tulkattua (interpreted) sovellusta. Tulkattu sovellus ei kuitenkaan ole tyypillisesti määritelty omaksi sovellustyypiksi samalla tasolla kuin PWA, verkkosovellukset ja hybridisovellukset, vaan se on alemman tason kategorisointi, joka ei rajoitu tiettyyn sovellustyyppiin. Tutkielmassa sovellustyyppien kategorisointi tehtiin yleisellä tasolla, jossa pyrittiin tutkimaan sovellustyyppien arkkitehtuurista johtuvia eroja ja niiden vaikutuksia käyttäjien ja ohjelmistotuotannon näkökulmasta. Tämä tarkoittaa, että alemman tason eroavaisuuksia, jotka eivät johdu suoranaisesti sovellustyyppien arkkitehtuurista, ei otettu tutkielmassa huomioon.

3.2 Alustariippumattomat sovellustyypit

Alustariippumattomuus on ohjelmiston ominaisuus, joka määrittelee sen pystyvän toimimaan usealla eri käyttöjärjestelmällä ilman että sitä tarvitsee erikseen kehittää eri alustoille. Alustariippumattomalla ohjelmistolla selkeä tekijä on se, että sillä on aina vain

yksi koodikanta (Majchrzak ym., 2018). Alustariippumattomien sovellusten vastakohta on natiivit sovellukset (Majchrzak ym., 2018). He määrittelevät, että natiivit sovellukset toimivat aina vain yhdellä alustalla, ja jos sovellus halutaan tuoda usealle alustalle, on se kehitettävä sille erikseen. Tämä tarkoittaa, että natiivissa kehityksessä jokainen ohjelmistokehityksen aktiviteetti on tehtävä jokaiselle käytetylle alustalle erikseen, sekä ohjelmistoa kehittävien organisaatioiden täytyy myös palkata alustakohtaisia osaajia (Majchrzak ym., 2018). Tämä tekee kehityksestä raskasta, sekä luo korkeita ylläpitokustannuksia, jotka voidaan alustariippumattomuudella täysin välttää. (Loreto, Braga, Peixoto, Machado & Abelha, 2018).

Kuitenkin erityisesti mobiilialustoilla on ohjelmistokehityksessä ollut pitkään haasteena menettely siitä, miten alustariippumattomuus on tehokkainta toteuttaa, eli mikä on tehokkain tapa tehdä sovellus saatavilla olevaksi usealle alustalle. (Huber ym., 2021). Tämä johtuu laitteiden välisistä konfiguraatioeroista sekä verkkokommunikointiin liittyvistä ongelmista natiivi-, hybridi, ja verkkosovelluksissa. (Khan ym., 2019). He esittävät tämän olevankin oleellisin ongelma, joka alun perin pyrittiin PWA sovellustyypillä korjaamaan. Majchrzak ym. (2018) määrittelevät, että alustariippumattomuus voidaan tehdä aikaiseksi kahdella eri menetelmätyypillä. Ensimmäinen menetelmä on generatiiviset menetelmät, jossa sovelluksen koodikanta generoidaan erikseen yhteensopivaksi eri alustoille, ja tämä generoitu koodi sitten ajetaan suoraan alustalla natiivien sovelluksien kaltaisesti (Majchrzak ym., 2018). He tuovat esille, että toinen tapa toteuttaa alustariippumattomuus on käyttää ajonaikaista ympäristöä, jossa ohjelmistoa ei ajeta suoraan alustan päällä, vaan sen sijaan se ajetaan sovelluksen ja alustan välissä, käyttäen jotain muuta ympäristöä, kuten esimerkiksi verkkoselainta.

Tapa miten alustariippumattomuus toteutetaan, määrittelee myös ohjelmiston sovellustyypin. Sovellustyypit on jaettu tutkielmassa kolmeen osioon: progressiiviset verkkosovellukset (PWA:t), verkkosovellukset, ja hybridisovellukset.

3.2.1 Progressiiviset verkkosovellukset

Progressiiviset verkkosovellukset ovat verkkosovelluksia, jotka käyttävät eri verkkoselainten sovellusohjelmointirajapintoja sekä progressiivisia parannusstrategioita parantaakseen sovelluksen käyttäjäkokemusta (Hasanuddin, Arham, Prakoso & Hermanto, 2022). Ne ilmestyivät ensimmäisen kerran vuonna 2015 (Khan ym., 2019), ja toimivat verkkoselaimen avulla tavallisen verkkosovelluksen tavoin, mutta ne eroavat verkkosovelluksista siten, että ne on mahdollista asentaa selaimen kautta, jolloin niitä voidaan käyttää työpöydälle, tai kotiruudulle luotujen pikakuvakkeiden kautta (Malavolta, Procaccianti, Noorland, & Vukmirovic, 2017). PWA:t kehitettiin alun perin ratkaisuksi mobiilialustoilla laitteistoihin ja verkkoyhteyksiin kohdistuviin rajoitteisiin, joita esiintyi natiivi-, hybridi- ja verkkosovelluksissa. (Khan ym., 2019).

PWA:t käyttävät standardeja verkkoteknologioita kuten HTML, CSS & JavaScript, ja ne vaativat toimiakseen pelkästään verkkoselaimen, joka tukee PWA:n arkkitehtuuria (Samah, Azam, Hamzah, Chew & Riza, 2021). PWA:t ovat erityisen hyödyllisiä mobiilialustoilla, mutta artikkelissaan Samah ym. (2021) myös toteavat, että niitä voi myös hyödyntää työpöytäalustoilla, joissa ne toimivat asennettuina pitkälti samalla tavoin kuin perinteiset natiivit työpöytäsovellukset. Koska PWA:t ovat selainpohjaisia, ne vaativat toimiakseen verkkoselaimen tarjoaman alustariippumattoman ajonaikaisen ympäristön. (Biørn-Hansen ym., 2017). He esittävät, että PWA:n voidaan ajatella olevan mukautettu versio perinteisistä verkkosovelluksista, joiden tavoite on toimia natiivien

verkkosovellusten tavoin. He toteavat, että arkkitehtuuriltaan PWA:t sisältävät kolme merkittävää komponenttia. Ensimmäinen komponentti on app manifest tiedosto, joka mahdollistaa PWA:n asennuksen käyttäjän laitteelle. Toinen esitelty komponentti service worker, joka puolestaan mahdollistaa suurimman osan PWA:n toiminnallisuudesta, kuten koko toiminnallisuuden selaimessa. Kolmas komponentti on application shell, joka saa aikaan nopean latausajan, sovelluksen dynaamisen sisällön esittämisen, sekä sovelluksen tietojen tallentamisen käyttäjän laitteen välimuistiin.

3.2.2 Verkkosovellukset

Myös tavallisten verkkosovelluksien voidaan ajatella olevan alustariippumattomia, sillä ne eivät ole riippuvaisia itse käyttöjärjestelmästä, vaan vaativat toimiakseen vain verkkoselaimen (Malavolta ym., 2017). Tämä tarkoittaa, että ne toteuttavat alustariippumattomuuden käyttämällä verkkoselainta ajonaikaisena ympäristönä (Majzhrak ym., 2018). Gambhir & Raj (2018) määrittelevät verkkosovellusten olevan verkossa hostattuja sovelluksia, jotka käyttävät toimiakseen standardeja verkkoprotokollia. Ne ovat saatavilla URL-osoitteen kautta, ja niiden voidaan ajatella olevan tavallisia verkkosivustoja, jotka ovat hyvin interaktiivisia, sekä toiminnoiltaan ja ulkoasultaan muistuttavat natiiveja sovelluksia. (Gambhir & Raj, 2018). Koska verkkosovellukset ovat pohjimmiltaan verkkosivustoja, käytetään niiden kehitykseen tyypillisiä verkkokehitykseen käytettyjä teknologioita. (Behl & Raj, 2018). Verkkosovellukset tyypillisesti kehitetään toimiviksi kaikilla alustoilla käyttämällä tiettyjä alustasopeutuvia mekanismeja, jonka avulla verkkosovellukset mukautuvat eri ruutujen kokoihin, joka on yksi merkittävä tekijä alustariippumattomuuden kannalta (Aguirre, Delía, Thomas, Corbalán, Cáseres, & Sosa, 2020).

3.2.3 Hybridisovellukset

Myös hybridiarkkitehtuuria noudattavat sovellukset ovat alustariippumatomia. Tutkielmassa hybridisovelluksiin sisällytetään verkkopohjaiset hybridisovellukset sekä alustariippumattomat hybridisovellukset.

Hybridisovellukset ilmestyivät ensimmäisen kerran vuonna 2009, ja niiden kehitykseen käytetään suurimmaksi osaksi verkkoteknologioita, kuten HTML5, CSS, JavaScript, jQuery, ym, (Khan ym., 2019). Tyypillisesti hybridisovellukset sisältävät käyttöjärjestelmille oman natiivin paketoinnin, jonka avulla sovellus pystyy käyttämään eri käyttöjärjestelmien spesifejä toimintoja (Majchrzak ym., 2018). Tämä paketointi tapahtuu hybridisovelluksissa olevan native wrapper komponentin avulla (Malavolta, 2016). Hybridisovellukset voivat toteuttaa alustariippumattomuuden eri tavoilla, riippuen niiden toteutukseen käytettävistä menetelmistä ja teknologioista, joka tarkoittaa, että ne voivat käyttää joko generatiivista menetelmää tai ajonaikaista ympäristöä (Huber ym., 2022).

Hybridisovelluksia käytetään suurimmaksi osaksi mobiilialustoilla, ja ne voivat hyödyntää mobiililaitteiden järjestelmällisiä toiminnallisuuksia erilaisten rajapintojen avulla (Khan. ym., 2019). Niiden tavoitteena on yhdistää natiivien ja verkkosovellusten parhaat puolet. (Gambhir & Raj, 2018). Ne asennetaan natiivien sovellusten tavoin yleensä sovelluskaupoista, kuten mobiilialustoilla App Storesta tai Google Playsta. (Malavolta, 2016).

3.3 Keskeisten tekijöiden eroavaisuudet

Nämä alustariippumattomat sovellustyypit, eli PWA, verkkosovellukset, ja hybridisovellukset, eroavat toisistaan merkittävästi. Kirjallisuuskatsauksen avulla valituista tutkimuksista ilmeni, että keskeisimmät vertailtavat tekijät PWA:n ja muiden alustariippumattomien sovellustyyppien välillä ovat suorituskyky, toiminnallisuus, saatavuus, löydettävyys, koko, tietoturva, ja energiankulutus. Näiden eroavien ominaisuuksien lisäksi vertaillaan tutkimuksessa myös PWA:n ja muiden alustariippumattomien sovellustyyppien välisiä eroja ohjelmistotuotannollisesta näkökulmasta, eli vertaillaan niiden kehittämiseen liittyviä näkökulmia, kuten kehittäjiltä vaadittuja resursseja. Tämä näkökulma on tutkielman tutkimuskysymyksen kannalta hyvin oleellinen, ja tällöin sillä on suuri vaikutus tutkielman tutkimuskysymykseen vastaamisessa.

Seuraavaksi käydään jokainen merkittävän tekijän läpi, syvennytään tekijään, ja vertaillaan sen eroja eri alustariippumattomien sovellustyyppien välillä. Tekijöiden eroavaisuudet käydään yhteenvetona läpi kappaleen 3.4 taulukoissa Taulukko 1. ja Taulukko 2.

3.3.1 Suorituskyky

Suorituskyvyllä viitataan tutkielmassa sovelluksen nopeuden ja reagointikyvyn tasoon, sekä käytetyn muistin määrään. Romao, Neuenschwander, Denecke & Nüssli (2021) toteavat, että PWA on suorituskyvyltään tehokkaampi kuin perinteinen verkkosovellus, jolloin myös sovelluksen latausajat ovat nopeampia niin nopeilla kuin hitaillakin verkkoyhteyksillä. PWA:n hyvä suorituskyky johtuu service worker komponentin mahdollistamasta käyttäjän laitteen välimuistiin sovelluksen tietojen tallentamisesta. (Gambhir & Raj, 2018). Magomadov (2020) puolestaan toteaa, että PWA:t eivät vaadi toimiakseen yhtä paljon dataa verrattuna perinteisiin verkkosovelluksiin, joka on myös yksi merkittävä syy parempaan suorituskykyyn. Rêgo, Portela, & Santos (2019) toteavat PWA:n vahvuudeksi sen, että se käyttää toimiakseen vähemmän verkkoliikennettä. Parempi suorituskyky johtaa siihen, että PWA toimii verkkosovellusta paremmin varsinkin hitailla verkkoyhteyksillä (Romao ym., 2021). PWA pystyy myös service worker komponenttinsa avulla taustaprosessointiin, joka parantaa suorituskykyä (Malavolta ym., 2017).

Hybridisovelluksissa puolestaan on natiivien sovellusten kaltoin huono ja rajoitettu käytettävyys huonolla verkkoyhteydellä. (Khan ym., 2019). Biørn-Hansen ym. (2017) esittävät tutkimustuloksessaan sen, että kevyt PWA on huomattavasti kevyttä hybridisovellusta nopeampi käynnistysnopeudeltaan ja sovelluksen työkalupalkin renderöimisessä. Khan ym. (2019) myös esittävät, että hybridi arkkitehtuuria noudattavat sovellukset ovat raskaampia ja tällöin myös käyttävät enemmän laitteen muistia verrattuna perinteisiin verkkosovelluksiin ja PWA:han

Artikkelissaan Behl & Raj (2018) esittävät, että suurin osa käyttäjistä harvoin käyttää enempää kuin muutamaa sovellusta päivittäin, joka tarkoittaa, että on yleistä, että käyttäjien laitteista tuhlautuu paljon muistia ylimääräisistä ja harvoin käytetyistä sovelluksista. He esittävät, että PWA ratkaisee tämän ongelman, sillä se vähentää sovelluksen käyttöön vaaditun muistin määrää huomattavasti.

3.3.2 Toiminnallisuus

Toiminnallisuus tarkoittaa sovelluksen toimintojen laajuutta. Tutkielmassa pyritään selvittämään, millaisia mahdollisuuksia ja rajoitteita eri sovellustyypeillä ilmenee sovelluksen toiminnallisuuteen. Malavolta (2016) toteaa että PWA:n toiminnallisuus on hyvin riippuvainen selainten tarjoamasta tuesta ja tavoista, miten ne tukevat verkkostandardeja. PWA:n merkittävimpiä toiminnallisuuksia ovat verkkoyhteydetön käyttö, kotiruudulle pikakuvakkeen asennus, taustasynkronointi, sekä notifikaatioiden lähettäminen. (Majchrzak ym., 2018). He kuitenkin myös toteavat, että ainakin verkkoyhteydetön toiminnallisuus, notifikaatiot ja taustasynkronointi ovat myös mahdollisia hybridisovelluksissa. Magomadov (2020) esittää, että toiminnallisuudeltaan PWA:t ovat huomattavasti perinteisiä verkkosovelluksia funktionaalisempia, sillä niissä on saatavilla enemmän toimintoja. PWA:sta puuttuu kuitenkin vielä toiminnallisuuksia, kuten pääsy laitteen kalenteriin, laitetason hälytyksiin, puhelinsoittotietoihin, ja muihin laitteen sensoreihin ja matalan tason toimintoihin (Malavolta, 2016).

PWA:t ovat toiminnallisuudeltaan laajempia kuin perinteiset verkkosovellukset (Wijaya, Crisgar, Pakpahan, Syamsuddin & Hasanuddin, 2021). Ne mahdollistavat muun muassa notifikaatioiden lähetyksen käyttäjän laitteelle, joka mahdollistaa laajemman interaktiivisuuden asteen ja kontaktin käyttäjään, joka puolestaan tyypillisesti parantaa konversion astetta (Wijaya ym., 2021). PWA:n web app manifest komponentin avulla PWA:t pystyvät myös tallentamaan käyttäjän laitteen välimuistiin sovelluksen dataa, joka mahdollistaa PWA:n käytön myös ilman verkkoyhteyttä, riippuen itse sovelluksen toiminnallisuudesta. (Ivanova & Georgiev, 2019). He esittävät, että tämä eroaa perinteisestä verkkosovelluksesta, koska niissä verkkoyhteydetön toiminnallisuus ei ole mahdollista lainkaan.

Diekmann & Eggbert, (2021) toteavat, että PWA:n toiminnallisuus on kuitenkin vielä epäjohdonmukaista, ja ne eivät välttämättä tue kaikkia toimintoja, jotka kehittäjät saattavat niiden olettaa tukevan. Tämän takia PWA:n kehityksessä on oltava varovainen, ja ennen kehitystä tehtävä tarkka vaatimusmäärittely ja tutkittava millaisia rajoitteita PWA arkkitehtuurilla on. (Diekmann & Eggbert, 2021). He toteavat myös, että itse PWA:n arkkitehtuurin rajoitteiden lisäksi on aina otettava huomioon myös verkkoteknologioiden rajoitteet toiminnallisuuteen. Romao ym. (2021) toteaa olevan tärkeää, että kehittäjät tutkivat myös itse verkkoselainten tarjoamia toiminnallisuuksia, ja varmistavat, että PWA:ssa tarvittavat toiminnallisuudet ovat tarjolla vähintäänkin kaikista suosituimissa selaimissa. Romao ym. (2021) tuovat myös esille, että toiminnallisuudessa voi myös esiintyä rajoittavia tekijöitä. He esittävät, että PWA:han upotetut videot ovat tyypillisesti huonosti yhteensopivia ja huonokäyttöisiä verkkoyhteyttömässä tilassa johtuen siitä, että videoissa esiintyvä suuri datamäärä voi helposti ylikuormittaa sovelluksen, ja täten hidastaa sitä.

3.3.3 Saatavuus

Saatavuudella tarkoitetaan tasoa, jolla sovellus on käytettävissä tilanteissa, joissa käyttäjät voivat sitä tarvita. Aguirre ym. (2020) esittävät, että työpöydälle tai kotiruudulle tallentamisen mahdollisuus tekee PWA:sta saatavuudeltaan huomattavasti paremman verrattuna perinteiseen verkkosovellukseen, koska käyttäjän ei tarvitse sovelluksen käyttöön aina avata selainta, joka voi hidastaa käyttöprosessia ja tehdä siitä työläämmän. Lisäksi koska PWA on verkkopohjainen, ei sen päivittämiseen tarvitse tehdä erillistä päivityksen lataamista esimerkiksi sovelluskaupasta, vaan se tapahtuu automaattisesti, ja siihen ei liity erillistä viivettä (Malavolta, 2016). Biørn-Hansen ym. (2017) esittää, että

mahdollisuus PWA:n käyttöön asentamatta sitä myös parantaa saatavuutta, sillä se on aina saatavilla selaimen kautta, eikä vaadi toimiakseen erillistä asennusta. Tämä on usein myös etu verrattuna hybridisovellukseen, koska käyttäjät voivat tällöin myös testata PWA:n toimintaa asentamatta sitä (Biørn-Hansen ym., 2017).

Majchrzak ym. (2018) esittävät PWA:n ja hybridisovelluksen pystyvän toimimaan ilman verkkoyhteyttä, jolloin sovelluksen käyttöä ei rajaa toimiva verkkoyhteys, joka parantaa PWA:n ja hybridisovellusten saatavuuden tasoa verrattuna verkkosovelluksiin. Aguirre ym. (2020) tuovat esille, että PWA:n työpöytätilan ansiosta niiden käytettävyys on selkeästi parempi kuin perinteisten verkkosovellusten. He toteavat tämän johtuvan siitä, että työpöytätila mahdollistaa kokoruututilan, jossa käyttäjän ei tarvitse turhaan nähdä ja olla vuorovaikutuksessa selainten eri elementtien kanssa, kuten tavallisten verkkosovellusten käytön tapauksessa.

PWA:lla vaikuttaa myös olevan epäjohdonmukaisuutta ja rajoitteita iOS alustalla, johtuen siitä, että iOS:n ainoana selaimena toimii Safari, jolla on rajoitteita PWA:n toiminnallisuuksien suhteen verrattuna muiden alustojen laajempaan selainvalikoimaan (Aguirre ym., 2020). Myös Majchrzak ym. (2018) toteavat iOS käyttöjärjestelmän rajoitteiden olevan yksi merkittävimmistä tekijöistä, miksi PWA teknologian käyttöönotto ja kehitys on melko epävarmaa ja etenee hitaasti, ja esittävät, että jos iOS alustalle ei saada PWA:lle täyttä tukea Applelta, ei PWA voi toimia korvaajana muille alustariippumattomille vaihtoehdoille.

3.3.4 Löydettävyys

Löydettävyys on ominaisuus, jolla viitataan tasoon, kuinka helposti uudet käyttäjät voivat löytää sovelluksen. Mobiilialustoilla PWA:t eivät ole ollenkaan riippuvaisia eri sovelluskaupoista, joka voi mahdollisesti helpottaa löydettävyyttä, sillä niitä on helpompi mainostaa esimerkiksi sosiaalisen median alustoilla (Magomadov, 2020). Hän myös toteaa, että PWA:t käyttävät sovelluksen sisäiseen navigaatioon URL:ää verkkosovellusten kaltoin, joka mahdollistaa helpon sovelluksen sisällön jaon muille käyttäjille. Koska PWA:t julkaistaan verkossa, niiden sisältöön kohdistuu hakukoneiden hakukoneoptimointi, joka tarkoittaa, että PWA ja sen sisältö on mukana hakukoneilla tehdyissä hauissa, joka voi tuoda PWA:lle mainostusta ja uusia käyttäjiä hakukoneilla tehtyjen hakujen myötä (Magomadov, 2020). Poissaolo sovelluskaupoissa on kuitenkin myös samalla heikkous, sillä sovelluskauppojen sovellukset koetaan yleisesti luotettavimmiksi muun muassa käyttäjäarvosteluiden takia (Magomadov, 2020). Myös Rêgo ym. (2019) määrittelevät PWA:n selaimen kautta asennuksen heikkoudeksi, johtuen siitä, että se on monille käyttäjille vielä täysin tuntematon prosessi, jolloin käyttäjät saattavat tuntea PWA:n asennuksen epäturvallisena. Biørn-Hansen ym. (2017) esittävät, että PWA:n selainpohjaisuus voi luoda käyttäjille turvattomuuden tunnetta verrattuna sovelluskaupoista ladattaviin natiivi- ja hybridisovelluksiin, sillä sovelluskaupat tarjoavat mahdollisuuden käyttäjäarvioinnille, sekä sovellusten tilastojen, kuten asennusmäärien tutkimiselle. Tämän lisäksi sovelluskauppoihin julkaisuilla on tiettyjä minimivaatimuksia, jotka edesauttavat siellä olevien sovellusten turvallisuuden tunnetta (Biørn-Hansen ym., 2017).

Aguirre ym. (2020) tuo esille strategian nimeltä TWA (Trusted Web Activities), jonka avulla PWA voidaan integroida Android applikaatioon. Tämä mahdollistaa PWA:n tuomisen Android alustan sovelluskauppaan. Aguirre ym. (2020) esittää, että tämä sovellus sitten toimii samalla tavoin kuin tavallinen PWA, jonka mahdollistaa Digital Assets Links niminen strategia, jonka avulla luodaan yhteys TWA:n APK tiedostoon ja

PWA:n verkkosivuun, jolloin TWA:n avattaessa avataan automaattisesti PWA. Tämä eroaa verkkosovelluksista, sillä ne eivät ole saatavilla sovelluskaupoista, vaan niitä on aina käytettävä verkkoselaimen kautta (Malavolta, 2016).

3.3.5 Asennuskoko

Asennuskoolla viitataan tässä tutkielmassa sovelluksen asentamiseen vaadittu tila käyttäjän laitteessa. Tutkimuksessaan Biørn-Hansen ym. (2017) esittävät että PWA:t ovat asennuskooltaan huomattavasti hybridisovelluksia pienempiä. He tuovat esille, tutkimustuloksissaan, että heidän tutkimustansa varten kehittämässään sovelluksista PWA on asennuskooltaan 43 kertaa pienempi kuin hybridiarkkitehtuurilla kehitetty sama sovellus. Jos PWA tuodaan sovelluskauppaan TWA:n avulla, kasvaa sovelluksen vaatima asennuskoko suuremmaksi (Aguirre ym., 2020). Perinteisissä verkkosovelluksissa asentaminen ei ole ollenkaan mahdollista, jolloin vertailua niihin ei voida tehdä.

3.3.6 Tietoturva

Tietoturvalla viitataan tutkielmassa sovellusten tietoturvaan liittyviin seikkoihin ja erityisesti sovelluksiin kohdistuviin uhkiin. Malavolta (2016) esittää, että PWA:n on pakko käyttää HTTPS protokollaa, jotta se, joka tarkoittaa, että se on vaatimus PWA:lle, ja toteutuu tällöin jokaisessa PWA:ssa. Artikkelissaan hän jatkaa esittämällä, että tämä vaatimus on olemassa man-in-the-middle hyökkäyksiä vastaan. Tämä on asia, jossa PWA:t eroavat tavallisista verkkosovelluksista, sillä tavalliset verkkosovellukset eivät pakosta käytä HTTPS protokollaa, joka tarkoittaa, että tietoturvaltaan PWA:t ovat keskimääräisesti parempia (Luntovskyy, 2018). Artikkelissaan myös Ivanova & Georgiev (2019) painottavat tietoturvaa yhdeksi PWA:n vahvuusalueiksi, johtuen PWA:n service worker scriptistä, jonka lävitse jokainen verkkopyyntö lähetetään. Tämä estää ulkopuolisia nuuskimasta sovelluksen tietoliikennettä (Ivanova & Georgiev, 2019). He toteavat, että tämä myös luo sovelluksen käyttäjille turvallisen olon ja rakentaa luottamusta sovellusta kohtaan. Toisaalta Lee, Kim, Park, Shin & Son (2018) toteavat että PWA:lla on arkkitehtuurinsa ja mahdollisten toimintojensa vuoksi monia ainutlaatuisia tietoturvariskejä. He tuovat esille riskejä kuten muun muassa tietojenkalastelun notifikaatioiden avulla, kolmannen osapuolen palveluiden käyttämisen push-notifikaatioiden sallimiseen (joka on yleinen tapa PWA:ssa), käyttäjän selaamishistoriaan liittyvät riskit, sekä service worker komponentin väärinkäyttöön liittyvät riskit. Tämän lisäksi PWA:n laajempi toiminnallisuus verrattuna verkkosovelluksiin luovat myös uusia haavoittuvuuksia, jotka eivät ilmene tavallisissa verkkosovelluksissa (Rêgo ym., 2019).

3.3.7 Energiankulutus

Energiankulutus on sovelluksen ominaisuus, joka mittaa sovelluksen käytön viemää energian määrää. Huber, Demetz & Felderer (2022) määrittelevät sen olevan erityisen tärkeä varsinkin mobiililaitteissa niiden määrällisen akun vuoksi. Artikkelissaan Malavolta ym. (2017) esittävät, että PWA:n arkkitehtuurin service worker komponentti ei huononna PWA:n energiatehokkuutta. Huber ym. (2022) puolestaan toteavat, että PWA on energiatehokkuudeltaan kelvollinen, ja että PWA:lla ei ilmene heikkouksia energiatehokkuuden suhteen. He toteavat, että energiatehokkuus PWA:ssa oli tiettyjä hybridisovelluksia parempi, mutta tuovat ilmi, että tämä pitkälti riippui itse sovelluksen toteutuksesta. Huber ym. (2022) toteavat, että PWA:ssa ja verkkosovelluksissa voidaan

käyttää tiettyjä optimisaatiomenetelmiä, jotka vähentävät niiden energiankulutusta, ja toteavat, että tämän takia sovellustyypit ovat energiankulutukseltaan pitkälti samoja.

3.3.8 Kehitysprosessi

Kehitysprosessilla viitataan tutkielmassa ohjelmistokehityksellisiin seikkoihin, jotka vaikuttavat sovelluksen kehitykseen ja julkaisemiseen liittyviin seikkoihin. Diekmann & Eggbert (2021) toteavat että PWA:n kehityksessä on paljon toiminnallisia rajoitteita, jolloin PWA:n kehityksessä on tehtävä paljon tutkimusta liittyen sen rajoitteisiin ja verrattava rajoitteita itse sovelluksessa vaadittuihin toiminnallisuuksiin, joka tarkoittaa, että PWA:ta kehittävä taho joutuu käyttämään resursseja vaatimusten ja rajoitteiden tutkimiseen. Magomadov (2020) tuo myös ilmi sen, että koska PWA:n kehitysprosessiin kuuluu osaksi ensin perinteisen verkkosovelluksen kehittäminen, joka myöhemmässä kehitysvaiheessa muunnetaan PWA:ksi, on PWA:n kehitysprosessi on aina aikaa vievämpi prosessi kuin perinteisen verkkosovellusten kehittämisprosessi. Toisaalta Fonseca, Peixoto, Braga, Machado & Abelha (2019) toteavat, että PWA:n kehittämisprosessi ei käytännössä vie juurikaan enempää aikaa verrattuna tavallisen verkkosovelluksen kehittämisprosessiin.

Toinen heikkous kehitysprosessissa on se, että PWA on käsitteenä melko tuore, joka tarkoittaa, että sillä on matalampi ohjelmistokypsyyden taso, eli PWA:n kehitysprosessi ja teknologiat eivät ole niin pitkälle kehitettyjä, dokumentoituja ja määriteltyjä verrattuna perinteisiin verkkosovelluksiin (Aguirre ym., 2020). He kuitenkin toteavat, että verkkosovelluksiin käytetyt teknologiat kehittyvät koko ajan suuntaan, jossa PWA:n kehittäminen helpottuu ja nopeutuu.

Hyviä puolia PWA:n kehityksessä on muun muassa se, että PWA:ta ei tarvitse päivittää jatkuvasti, joka pitkässä juoksussa voi säästää kehittäjiltä resursseja (Magomadov, 2020). Behl & Raj (2018) myös tuovat esille sen, että koska PWA:t käyttävät pitkälti samaa teknologiastakkia kuin yleisimmät verkkosovellukset, ja erityisesti koska PWA:n kehitys perustuu pitkälti JavaScript ohjelmointikielen ympärille, ohjelmistoyritykset eivät vaadi erillistä osaamista ja asiantuntemusta PWA:n kehitystä varten, vaan voivat hyödyntää samoja verkkokehittäjiä myös PWA:n kehittämisessä.

Myös hybridikehityksessä voi esiintyä tämä ilmiö, sillä siinä hyödynnetään myös yleisesti samaa kehitysstäkkiä kuin verkkokehityksessä (Malavolta, 2016). Toisaalta hybridisovellusten kehityksessä täytyy sovellus tyypillisesti manuaalisesti integroida jokaiselle käytetävälle alustalle toisin kuin PWA:ssa ja verkkosovelluksessa (Gambhir & Raj, 2018).

3.4 Keskeisten tekijöiden eroavaisuuksien yhteenveto

Tässä kappaleessa käydään yhteenvetona läpi aikaisemmista tutkimuksista ilmi tulleet alustariippumattomien sovellustyyppien tekijöiden eroavaisuudet. Taulukko 1. kuvaa yhteenvetona PWA:n ja verkkosovellusten tekijöiden eroavaisuudet, ja Taulukko 2. kuvaa yhteenvetona PWA:n ja hybridisovellusten tekijöiden eroavaisuudet.

Taulukko 1. Yhteenveto PWA:n ja verkkosovellusten tekijöiden eroavaisuuksista.

	PWA:n ja verkkosovellusten tekijöiden eroavaisuudet
Suorituskyky	- PWA:lla on parempi tehokkuus kuin verkkosovelluksella (Romao ym., 2021). - PWA:t vaativat vähemmän dataa toimiakseen (Magomadov, 2020). - PWA:t toimivat myös paremmin hitailla verkkoyhteyksillä (Romao ym., 2021).
Toiminnallisuus	- PWA:lla on huomattavasti enemmän saatavilla olevia toimintoja (Magomadov, 2020). - PWA:n laajempi toiminnallisuus parantaa interaktiivisuuden astetta, joka lisää konversion määrää (Wijaya ym., 2021). - PWA:n työpöytätila mahdollistaa paremman käyttökokemuksen (Aguirre ym., 2020). - PWA:lla on matalampi kypsyyden taso, joka tarkoittaa, että enemmän mahdollisia riskejä (Diekmann & Eggbert, 2021). - PWA:lla on mahdollisuus verkkoyhteydettömään käyttöön (Ivanova & Georgiev, 2019).
Saatavuus	- PWA:n mahdollinen asentaminen työpöydälle asentaminen parantaa saatavuutta (Aguirre ym., 2020). - PWA:n verkkoyhteydetön käyttö parantaa saatavuutta (Majchrzak ym., 2018). - PWA:lla on rajoitteita iOS alustalla (Aguirre ym., 2020).
Löydettävyys	- PWA voidaan lisätä Android sovelluskauppaan TWA:n avulla, joka parantaa sen löydettävyyttä (Aguirre ym., 2020), toisin kuin verkkosovellusta, jonka löytämiseen on aina käytettävä selainta (Malavolta, 2016) - PWA:n asennusprosessin ainutlaatuisuus voi saada aikaan turvattoomuuden tunnetta käyttäjissä, jolloin käyttäjät voivat tuntea PWA:n asennuksen epäturvallisena (Rêgo ym., 2019).
Asennuskoko	- Verkkosovellusta ei voi asentaa, ei vertailtavissa.
Tietoturva	- PWA:lla yleisesti parempi tietoturvan taso, sillä ne käyttävät aina HTTPS verkkoprotokollaa, toisin kuin verkkosovellukset (Luntovskyy, 2018). - PWA:ssa jokainen verkkopyyntö lähetetään service worker scriptin kautta, joka lisää tietoturvan tasoa (Ivanova & Georgiev, 2019). - PWA:n arkkitehtuuri mahdollistaa ainutlaatuisia tietoturvauhkia, jotka on otettava huomioon (Lee ym., 2018).
Energiankulutus	- Ei huomattavia eroja energiankulutuksessa (Huber ym., 2022).
Kehitysprosessi	- PWA rakennetaan verkkosovelluksen päälle, joka tarkoittaa, että se on kehitysprosessiltaan aina pidempi (Magomadov, 2020). - PWA teknologioilla on matalampi kypsyyden taso (Aguirre ym., 2020). - PWA:ssa ja verkkosovelluksessa voidaan käyttää samaa teknologiastäkkiä (Behl & Raj, 2018).

Taulukko 2. Yhteenveto PWA:n ja hybridisovellusten tekijöiden eroavaisuuksista.

	PWA:n ja hybridisovellusten tekijöiden eroavaisuudet
Suorituskyky	- PWA:lla nopeampi käynnistymisnopeus (Biørn-Hansen ym., 2017). - PWA:t vaativat vähemmän muistia laitteelta (Behl & Raj, 2018). - Kevyet PWA:t ovat käynnistymisnopeudeltaan ja ulkoasun renderöimisessä nopeampia (Khan ym., 2019). - PWA:t ovat kevyempiä, jonka takia ne vaativat toimiakseen vähemmän muistia (Khan ym., 2019).
Toiminnallisuus	- PWA:t ovat riippuvaisia verkkoselaimista (Malavolta, 2016).
Saatavuus	- PWA:n mahdollinen käyttö ilman asennusta parantaa sen saatavuutta (Biørn-Hansen ym., 2017). - PWA:ssa käyttäjät voivat testata sovellusta ennen sen lataamista, joka on suuri etu uusien käyttäjien kannalta (Biørn-Hansen ym. 2017). - PWA:lla on rajoitteita iOS alustalla (Aguirre ym., 2020).
Löydettävyys	- PWA:t ei riippuvaisia sovelluskaupoista, joka voi helpottaa ja lisätä mahdollisuuksia parantaa löydettävyyttä (Magomadov, 2020). - PWA:n URL pohjainen navigaatio mahdollistaa sovelluksen sisällön helpon jaon käyttäjien kesken (Magomadov, 2020). - PWA:ssa voidaan hyödyntää hakukoneoptimointia (Magomadov, 2020). - PWA:lla ei ole sovelluskauppojen luomaa luotettavuutta (Magomadov, 2020). - PWA:n asennusprosessin ainutlaatuisuus voi saada aikaan turvattomuuden tunnetta käyttäjissä, jolloin käyttäjät voivat tuntea PWA:n asennuksen epäturvallisena (Rêgo ym., 2019). - Käyttäjät tuntevat PWA:t asennuksen turvattomammaksi johtuen siitä, että PWA:ssa ei välttämättä toteutu sovelluskauppojen vaatimukset, eivätkä käyttäjät voi nähdä suoraan PWA.sta lisätietoja kuten sovelluskaupoissa on tyypillistä (Biørn-Hansen ym., 2017).
Asennuskoko	- Asennuskooltaan PWA:t ovat huomattavasti pienempiä verrattuna hybridisovelluksiin (Biørn-Hansen ym., 2017).
Tietoturva	- PWA:n arkkitehtuuri mahdollistaa ainutlaatuisia tietoturvauhkia, jotka on otettava huomioon (Lee ym., 2018).
Energiankulutus	- Ei huomattavia eroja energiankulutuksessa (Huber ym., 2022).
Kehitysprosessi	- Molemmissa voidaan pitkälti käyttää samaa teknologiastäkkiä (Malavolta, 2016). - Hybridisovellukset tyypillisesti on integroitava erikseen eri alustoille, joka voi lisätä vaadittuja resursseja (Gambhir & Raj, 2018).

4. Pohdinta

Aikaisempien tutkimusten perusteella ilmeni, että PWA:lla on paljon vahvuuksia ja heikkouksia verrattuna muihin alustariippumattomiin sovellustyyppeihin. Näiden vahvuuksien ja heikkouksien vaikutukset ovat pitkälti riippuvaisia sovelluksen käyttötarkoituksista ja käyttäjäkannasta.

PWA:n kehitys verrattuna muihin sovellustyyppeihin ilmeni melko riskialttiiksi. Yksi selkeimmistä riskeistä ilmeni PWA:n riippuvuus selaimesta, ja varsinkin Applen tuki iOS ja macOS alustoilla käytettävälle Safari-selaimelle (Aguirre ym., 2020). Jos PWA:t eivät Safarin kautta saa täyttä tukea näille alustoille, laskee PWA:n saatavuus huomattavasti, koska se voi rajoittaa, tai jopa täysin poissulkea suuren osan sovelluksen Applen tuotteita käyttävästä käyttäjäkannasta. Tällöin PWA ei soveltuisi hyvin alustariippumattomaksi ratkaisuksi, sillä alustariippumattomuuden päätavoite, joka on sovelluksen saatavuuden parantaminen, ei toteutuisi tehokkaalla tasolla.

Toinen selkeä heikkous on tiettyjen toiminnallisuuksien puuttuminen. Malavolta (2016) toteaa, että muun muassa pääsy laitteen kalenteriin, hälytyksiin, soittotietoihin, sensoreihin ja muihin matalan tason toimintoihin puuttuu PWA:sta täysin. Tämä rajoittaa PWA:n käytön sellaisiin käyttötarkoituksiin, joissa näitä toimintoja ei tarvita. Tämä saa aikaan sen, että PWA:n käyttö on hyvin käyttötapauksesta riippuvaista, ja voi toimia merkittävänä riskinä PWA:na toteutetun sovelluksen laajentamiselle tulevaisuudessa. Lisäksi koska PWA:lla ei ole täyttä pääsyä matalan tason toimintoihin, ei se sovellu kovin hyvin raskasta prosessointia vaativiin sovelluksiin, kuten peleihin.

PWA:n pieni asennuskoko on yksi sen vahvuuksista, ja on erityisen vaikuttava vanhemmilla sekä vähemmän tehokkailla laitteilla, joissa tallennustilaa ja muistia on hyvin rajallinen määrä. Biørn-Hansen ym. (2017) totesivat että PWA:n asennuskoko voi olla jopa kymmeniä kertoja pienempi verrattuna muihin asennettaviin sovellustyyppeihin. Tällä on hyvin mittavia etuja, sillä PWA:n käyttö muiden sovellustyyppien sijaan voi parantaa koko laitteiden suorituskykyä ja tällöin myös käyttökokemusta, ja mahdollistaa käyttäjän laitteen laajemman toiminnallisuuden johtuen mahdollisuudesta asentaa ja ladata enemmän sovelluksia. Tämä tekee siitä selkeästi hybridi- ja jopa natiivisovellusta paremman vaihtoehdon kevyissä sovelluksissa. Koska PWA on suorituskyvyltään huomattavasti verkkosovellusta parempi (Romao ym., 2021), se ikään kuin yhdistää ladattavien (hybridisovellukset) ja ei-ladattavien (verkkosovellukset & hybridisovellukset) parhaat puolet, jossa käyttäjä saa tehokkaan, ja hyvin vähän resursseja vievän sovelluksen, jota voi myös mahdollisesti käyttää ilman verkkoyhteyttä.

Tietoturvaltaan PWA:t ovat arkkitehtuuriltaan hyviä, sillä niissä on minimivaatimuksena käyttää turvallista HTTPS tietoliikenneprotokollaa sovelluksen verkkoliikenteen salaamiseen (Luntovskyy, 2018). Tämän lisäksi PWA:t käyttävät kommunikointiin service worker komponenttia, joka välityspalvelimen kaltoin salaa PWA:n ja käyttäjän väliset verkkopyynnöt (Ivanova & Georgiev, 2019). Vaikka PWA:n arkkitehtuurissa ilmeni yksityiskohtaisia tietoturvariskejä (Lee ym., 2018), on ne huomioon otettaessa mahdollista minimisoida, ja on mahdollista, että tulevaisuudessa PWA teknologioiden kypsyessä ne voidaan jopa mitätöidä täysin.

Koska PWA:t ovat riippumattomia sovelluskaupoista (Magomadov, 2020), ja koska ne on teknisesti mahdollista myös lisätä sovelluskauppoihin (Aguirre ym., 2020), voivat PWA:t tulevaisuudessa olla myös löydettävyydeltään huomattavasti tehokkaampia verrattuna pelkästään selaimen kautta käytettyihin verkkosovelluksiin. Kuitenkin

nykyhetkellä tämä on vasta osittain mahdollista, sillä TWA:n avulla PWA:t eivät ole julkaistavissa iOS alustojen App Storessa, vaan pelkästään Android alustoilla (Aguirre ym., 2020). Tämä tuo jälleen ilmi PWA:n yhden keskeisimmistä riskeistä, joka on Applen rajattu tuki PWA:n kehittämistä varten.

Kehitysprosessiltaan PWA:ssa ei ilmennyt muita vakavia haasteita kuin Aguirre ym. (2020) esille tuoma tämänhetkinen PWA teknologioiden kypsyyden taso, joka ilmeni huomattavasti matalemmaksi kuin verkko- ja hybridisovelluksissa. Tämä ilmiö kuitenkin katoaa ajan kuluessa, kun PWA:n teknologioita kehitetään eteenpäin, jonka takia se ei ole pitkällä tähtäimellä kovinkaan suuri ongelma. PWA:n kehitykseen liittyen ilmeni paljon positiivisia seikkoja, jotka tekevät siitä kehityksen kannalta hyvin kelvollisen sovellustyypin. Koska kehitykseen käytetään samaa teknologiastäkkiä kuin verkkokehityksessä (Malavolta, 2016), ja koska ne kehitetään verkkosovelluksen päälle (Magomadov, 2020), on yritysten hyvin helppo mukautua verkkosovellusten kehityksestä PWA:n kehitykseen, ja muuntaa jo olemassa oleva verkkosovellus PWA:ksi. Tämän ei pitäisi vaikuttaa aikaisempaan käyttäjäkantaan, sillä PWA toimii selaimen kautta tavallisen verkkosovellusten kaltaisesti, jolloin käyttäjät voivat itse valita haluavatko jatkaa sovelluksen käyttöä selaimen avulla, vai haluavatko asentaa PWA:n, ja hyötyä sen tuomista eduista. Koska myös hybridisovelluksissa tyypillisesti käytetään samoja verkkoteknologioita, pitäisi myös niiden kehittämisestä olla melko vaivatonta mukautua kehittämään PWA:ta.

PWA sovellustyypin tuoreus ja tuntemattomuus tuo mukanaan myös muita riskejä. Koska PWA eroaa asennusprosessiltaan muista sovellustyypeistä, ja PWA on itsessään vielä muita sovellustyyppejä vähemmän tunnettu, voi moni käyttäjä olla täysin tietämätön PWA:n asennusominaisuuden olemassaolosta, ja tällöin eivät hyödy kaikista PWA:n tuomista eduista. Tämän takia PWA:n kehittävien yritysten kannattaa pyrkiä myös informoimaan PWA:n olemassaolosta käyttäjille, sekä tarjota ohjeita ja opastusta sen asentamiseen ja käyttöön. Tällöin myös yleinen tietämys sovellustyypin olemassaolosta paranisi käyttäjien keskuudessa, suurempi osa käyttäjistä ymmärtäisi sen hyödyt, ja koko sovellustyypin suosio saattaisi kasvaa. Tämä myös saattaisi painostaa selaimia kehittäviä yrityksiä panostamaan enemmän PWA:n tukemiseen. Tällä hetkellä PWA:n tuntemattomuuden takia alustariippumattoman sovelluksen toteutus voi olla käyttäjien kannalta helpompaa toisella, perinteisemmällä, sovellustyypillä, sillä niiden käyttö on suurimmalle osalle käyttäjäkuntaa tutumpi, oli kyseessä sovellus mikä tahansa. Yritysten näkökulmasta PWA:n käyttö voi siis tällä hetkellä tuntua turhan monimutkaiselta ratkaisulta, sillä suurin osa käyttäjistä ei edes tiedä sovellustyypin olemassaolosta

Ominaisuuksiltaan PWA siis sopii hyvin tavaksi toteuttaa alustariippumaton sovellus, jos sovelluksen käyttötarkoitus ei vaadi tiettyjä PWA:sta puuttuvia laitetason toiminnallisuuksia, joihin kuuluu Malavolta (2016) mukaan pääsy laitteen kalenteriin, hälytyksiin, soittotietoihin, sensoreihin ja muihin matalan tason toimintoihin. PWA:lla on kuitenkin haasteita käyttäjien PWA:n käytön matalan tuntemuksen tason vuoksi, joka voi tehdä siitä riskialttiimman tavan toteuttaa alustariippumaton sovellus. Tämän takia PWA:ta kehittävän yrityksen kannattaa aina pyrkiä opastamaan käyttäjiä PWA:n asentamisessa ja käytössä. Jos PWA:n heikkoudet, kuten laitetason toimivuuksien puute, verkkoselainten tuki, ja sovellustyypin vähäinen tuntemuksen taso käyttäjien keskuudessa ratkaistaan, voi PWA tulevaisuudessa jopa syrjäyttää kevyet natiivi- ja hybridisovellukset, joissa ei vaadita matalan tason mahdollistavaa tehokasta raskasta prosessointia.

5. Yhteenveto

Tämä tutkielma tutki kirjallisuuskatsauksen avulla PWA sovellustyypin soveltuvuutta alustariippumattomana sovellustyyppinä vertailemalla sitä muihin alustariippumattomiin sovellustyyppeihin, joihin kuuluu verkko- ja hybridisovellukset. Aiemman tutkimuksen perusteella tutkielmassa käsiteltiin kahdeksaa merkittävää tekijää, joihin sovellustyyppien vertailu kohdistettiin. PWA:n vahvuuksiksi ilmeni hyvä tehokkuuden, saatavuuden, ja tietoturvan taso, laajempi toiminnallisuus verrattuna verkkosovelluksiin, suuri potentiaali erityisen hyvälle löydettävyyden tasolle, ja pieni asennuskoko. Heikkouksiksi puolestaan ilmeni joidenkin toimintojen puute, riippuvuus selainten & niitä kehittävien yritysten tuesta, sovellustyypin matala tuntemuksen taso käyttäjien keskuudessa, ja PWA teknologioiden matala kypsyyden taso.

Tutkielman tuloksiin vaikutti se, että tutkielmassa alustariippumattomat sovellustyypit rajattiin melko yleisellä tasolla, varsinkin hybridisovellusten kannalta. Hybridisovelluksissa on todella laajasti sisäisiä eroja, riippuen niiden toteutusmenetelmistä. Ne voivat olla joko verkkopohjaisia tai alustariippumattomia, ja tämän lisäksi alustariippumattomissa hybridisovelluksissa on myös paljon sisäisiä eroja. Koska tutkielmassa kaikki hybridisovellukset sisällytettiin hybridisovellus-kategoriaan, voi vertailu niihin olla epäjohdonmukaista, sillä hybridisovellukset vaihtelevat keskenään huomattavasti yksityiskohdiltaan, joka tarkoittaa, että vertailu hybridisovelluksiin saattoi olla tehty liian yleisellä tasolla. Kuitenkin tutkielman kannalta ne oli järkevintä jakaa yleisellä tasolla hybridisovelluksiin, koska niiden erillinen käsittely olisi ollut liian laaja tutkielman laajuuden kannalta, sekä tämä sama luokittelu oli myös tehty suuressa osassa kirjallisuuskatsauksessa ilmi tulleissa tutkimuksissa, joka viittaa siihen, että hybridisovellusten yleisen tason kategorisointi on sopiva rajaus sovellustyyppien väliseen vertailuun.

Tutkielman tuloksiin vaikutti myös kirjallisuuskatsauksen käyttö tutkimusmenetelmänä. Joistakin ominaisuuksista, kuten hybridisovellusten tietoturvan tasosta, ei löytynyt vertailua PWA:n tietoturvan tasoon, jonka takia tietoturvatason vertailua oli haastavaa tehdä. Kirjallisuuskatsaus saattoi myös saada aikaan liian teoreettisen näkökulman PWA:sta ja muista hybridisovelluksista, sen kelvollisuuden tutkimisesta saattoi jäädä pois käytännön tekijöitä, muun muassa esimerkiksi käyttäjien suhtautumisesta PWA:han uutena sovellustyyppinä ja PWA:n asennusprosessin opittavuus käyttäjien näkökulmasta. Tämän lisäksi aikaisempi tutkimus keskittyi hyvin pitkälti mobiilialustoille, eikä tutkimuksissa otettu huomioon juurikaan muita alustoja.

Tämän tutkielman tuloksia voidaan hyödyntää ohjelmistokehityksessä alustariippumattoman sovellustyypin määrittelyssä ja valinnassa. Tutkielma on erityisen hyödyllinen, jos sovellusta kehittävä organisaatio harkitsee sovelluksensa toteutusta PWA:na, jolloin tutkielmaa voi käyttää hyväksi PWA:n etujen ja haasteiden vertailuun sovelluksen käyttöympäristössä, ja sitä kautta määritellä PWA:n soveltuvuuden taso suunnitellun sovelluksen käyttöympäristössä.

PWA, ja alustariippumattomat sovellustyypit yleisesti ovat hyvin laaja aihepiiri. Tämä luo tarvetta jatkotutkimukselle, joka keskittyy yhä tarkemmin tiettyihin sovellustyyppeihin, sekä sovellustyyppien eroavaisuuksiin. Muun muassa laajempi tutkimus liittyen hybridisovelluksiin kuuluviin alustariippumattomiin hybridisovelluksiin, ja erityisesti niiden eroja PWA:han on tarvetta tutkia yhä tarkemmin. PWA:ta on myös tarpeen tutkia tarkemmin vielä käytännön tasolla. PWA:lla on ainutlaatuinen asennusprosessi, jossa asennus tapahtuu selaimen kautta. Tämä on monelle

käyttäjälle täysin uusi ilmiö, joka tarkoittaa, että käyttäjät eivät välttämättä ole tottuneet siihen tai edes ole tietoisia siitä. Tämä luo tarvetta käyttäjätutkimukselle liittyen PWA:n asennus- ja käyttöprosessiin.

Tämän lisäksi aihetta voidaan tutkia laajemmin myös ottamalla huomioon natiivit sovellukset, ja vertailemalla niitä PWA:han muistakin kuin alustariippumattomuuden rajauksesta. Tarvetta jatkotutkimukselle tuo myös se, että PWA on vielä hyvin uusi ja jatkuvan kehityksen alla oleva teknologiailmiö, joka tarkoittaa, että sen ominaisuudet sekä niiden vaikutukset voivat muuttua hyvinkin äkillisesti. Tämä saa aikaan jatkuvan jatkotutkimuksen tarpeen PWA sovellustyypille.

Lähteet

Aguirre, V., Delía, L., Thomas, P., Corbalán, L., Cáseres, G., & Sosa, J. F. (2020). *PWA and TWA: recent development trends*. Computer Science–CACIC 2019: 25th Argentine Congress of Computer Science, (CACIC 2019), 25, 205-214.

Arksey, H., & O'Malley, L. (2005). *Scoping studies: towards a methodological framework*. International journal of social research methodology, 8(1), 19-32. https://doi.org/10.1080/1364557032000119616

Behl, K., & Raj, G. (2018). *Architectural pattern of progressive web and background synchronization.* 2018 International Conference on Advances in Computing and Communication Engineering (ICACCE), 366-371.

Biørn-Hansen, A., Majchrzak, T. A., & Grønli, T. M. (2017). *Progressive Web Apps: The Possible Web-native Unifier for Mobile Development*. Proceedings of the 13th International Conference on Web Information Systems and Technologies (WEBIST), 344-351. https://doi.org/10.5220/0006353703440351

Diekmann, J., & Eggert, M. (2021). *Is a Progressive Web App an Alternative for Native App Development?*. 3. Wissenschaftsforum: Digitale Transformation (WiFo21), 35-48.

Fonseca, F., Peixoto, H., Braga, J., Machado, J. M., & Abelha, A. (2019). *Smart mobile computing in pregnancy care*. Proceedings of 34th International Conference on Computers and Their Applications, 58, 219-224 https://doi.org/10.29007/SR6Q

Gambhir, A., & Raj, G. (2018). *Analysis of cache in service worker and performance scoring of progressive web application*. 2018 International Conference on Advances in Computing and Communication Engineering (ICACCE). 294-299. https://doi.org/10.1109/ICACCE.2018.8441715

Hasanuddin, M. O., Arham, A. P., Prakoso, G. A., & Hermanto, B. R. (2022). *Implementation of Progressive Web App-Based Applications for Vehicle Tracking Systems*. 2022 12th International Conference on System Engineering and Technology (ICSET). 6-12. https://doi.org/10.1109/ICSET57543.2022.10010767

Huber, S., Demetz, L., & Felderer, M. (2021). *Pwa vs the others*: *A comparative study on the ui energy-efficiency of progressive web apps*. Web Engineering: 21st International Conference, ICWE 2021, 464-479. https://doi.org/10.1007/978-3-030-74296-6_35

Huber, S., Demetz, L., & Felderer, M. (2022). *A comparative study on the energy consumption of Progressive Web Apps*. Information Systems, 108(1). https://doi.org/10.1016/j.is.2022.102017

Ivanova, S., & Georgiev, G. (2019). *Using modern web frameworks when developing an education application: a practical approach*. 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 1485-1491. https://doi.org/10.23919/MIPRO.2019.8756914

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

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*. 2018 CCS '18: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 1731-1746. https://doi.org/10.1145/3243734.3243867

Loreto, P., Braga, J., Peixoto, H., Machado, J., & Abelha, A. (2018). *Step towards progressive web development in obstetrics*. Procedia Computer Science, 141, 525-530. https://doi.org/10.1016/j.procs.2018.10.131

Luntovskyy, A. (2018). *Advanced software-technological approaches for mobile apps development*. 2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET), 113-118. https://doi.org/10.1109/TCSET.2018.8336168

Magomadov, V. S. (2020). *Exploring the role of progressive web applications in modern web development*. Journal of Physics: Conference Series, 1679(2). https://doi.org/10.1088/1742-6596/1679/2/022043

Majchrzak, T. A., Biørn-Hansen, A., & Grønli, T. M. (2018). *Progressive web apps: the definite approach to cross-platform development?*. Hawaii International Conference on System Sciences, 5735-5744. https://doi.org/10.24251/HICSS.2018.718

Malavolta, I. (2016). *Beyond native apps: web technologies to the rescue! (keynote)*. Proceedings of the 1st International Workshop on Mobile Development, 1-2. https://doi.org/10.1145/3001854.3001863

Malavolta, I., Procaccianti, G., Noorland, P., & Vukmirovic, P. (2017). *Assessing the impact of service workers on the energy efficiency of progressive web apps*. 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft). 35-45. https://doi.org/10.1109/MOBILESoft.2017.7

Munn, Z., Peters, M. D., Stern, C., Tufanaru, C., McArthur, A., & Aromataris, E. (2018). *Systematic review or scoping review? Guidance for authors when choosing between a systematic or scoping review approach*. BMC medical research methodology, 18(143), 1-7. https://doi.org/10.1186/s12874-018-0611-x

Romao, P., Neuenschwander, S., Denecke, K., & Nüssli, S. (2021). *Digital Health Intervention to Support Refugees in Switzerland*. Stud Health Technol Inform, 279, 95-102. https://doi.org/10.3233/SHTI210094

Rêgo, F., Portela, F., & Santos, M. F. (2019). *Towards PWA in healthcare.* Procedia Computer Science, 160, 678-683. https://doi.org/10.1016/j.procs.2019.11.028

Samah, K. A. F. A., Azam, N. A., Hamzah, R., Chew, C. S., & Riza, L. S. (2021). *Optimizing Smartphone Recommendation System through Adaptation of Genetic Algorithm and Progressive Web Application*. International Journal of Advanced Computer Science and Applications, 12(12), 266-273. https://doi.org/10.14569/IJACSA.2021.0121235

Wijaya, P. R., Crisgar, P. V., Pakpahan, M. D., Syamsuddin, E. Y., & Hasanuddin, M. O. (2021). *Implementation of Motor Vehicle Tracking Software-as-a-Service (SaaS) Application Based on Progressive Web App*. 2021 International Symposium on Electronics and Smart Devices (ISESD), 1-6. https://doi.org/10.1109/ISESD53023.2021.9501600