jamk

React Native ja alustariippumaton mobiilikehitys

Olli Mikkonen

Opinnäytetyö, AMK
Joulukuu 2023
Tietojenkäsittelyn tutkinto-ohjelma

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Mikkonen, Olli

React Native ja alustariippumaton mobiilikehitys.

Jyväskylä: Jyväskylän ammattikorkeakoulu. Joulukuu 2023, 28 sivua.

Tietojenkäsittelyn tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: kyllä

Tiivistelmä

Opinnäytetyön tarkoituksena oli tutkia alustariippumattomaan mobiilisovelluskehitykseen käytettävää React Native -kehystä korostaen sen rakennetta, toiminnallisuutta ja parhaita käytäntöjä. Tutkimuksessa selvitettiin React Nativen potentiaalia toimivana kehyksen mobiilisovelluksia kehitettäessä sekä millä käytänteillä päästään parhaaseen lopputulokseen.

Tutkimuksen totetustapa oli tutkimuksellinen kehittämistyö. Tutkimuksen aluksi tietoperustaa kerättiin kattavasti ja monipuolisesti erinäisistä lähteistä ja kasvatettiin ymmärrystä React Native -kehyksen toiminnasta. Käytännössä kerättyä tietoa käytettiin demosovelluksen kehittämiseen ja muodostettiin tämän kokemuksen sekä syntyneen demosovelluksen perusteella tutkimustulokset.

Tutkimustuloksina saatiin kattavasti tietoa siitä, miten React Native -kehys soveltuu mobiilisovelluskehitykseen. Mobiilisovelluksen kehittämisprojektia aloittava sovelluskehittäjä voi tehdä tutkimustulosten perusteella johtopäätöksiä siitä kannattaako projektissa käyttää React Native -kehystä ja mitä ottaa huomioon näin tehdessä.

Yhteenvetona tutkimuksessa todettiin mitä haasteita ja hyötyjä React Nativella työskentelyssä voi olla, miten toimia havaittujen haasteiden kanssa, miten toimia kyseistä kehystä käyttäessä sekä onko kehys varteenotettava vaihtoehto mobiilisovellusta kehittäessä.

Avainsanat (asiasanat)

React Native, React, mobiilikehitys, Redux, alustariippumattomuus

jamk

Description

Mikkonen, Olli

React Native and cross platform mobile development

Jyväskylä: JAMK University of Applied Sciences, December 2023, 28 pages.

Degree Programme in Business Information Technology. Bachelor´s thesis.

Permission for open access publication: Yes

Language of publication: Finnish

Abstract

The purpose of the thesis was to investigate the React Native framework for cross platform mobile app development, emphasizing its structure, functionality, and best practices. The study aimed to explore the potential of React Native as a functional framework for developing mobile applications and identify the practices that lead to the best results.

The research was conducted as a developmental study with a research-oriented approach. The initial phase involved collecting comprehensive and diverse theoretical knowledge from various sources to gain an understanding of React Native's operation. The practical part utilized the gathered knowledge to develop a demo application, and the research results were derived from the experience gained and the resulting demo application.

The research provided extensive information on how the React Native framework is suitable for mobile app development. Based on the research results, a developer initiating a mobile app development project can draw conclusions about whether to use the React Native framework and what considerations to consider.

In summary, the research identified the challenges and benefits of working with React native, suggested ways to address observed challenges, provided guidance on working with the framework, and assessed whether the framework is a viable option for mobile app development.

Keywords/tags (subjects)

React Native, React, mobile development, Redux, cross platform

Sisältö

Kuviot

1 Johdanto

Opinnäytetyössä tutkitaan Facebookin (nykyisen Metan) mobiilisovelluskehitykseen julkaisemaa React Native -ohjelmointikehystä. Pääpaino tutkimuksessa on React Nativella luodun mobiilisovelluksen rakenteessa ja toimivuudessa sekä näiden hyvän ja oikeanlaisen toiminnan kannalta tärkeissä käytänteissä. React Nativen ollessa suosittu ja monien mobiilisovellusten kehittämiseen käytetty ohjelmointikehys on sen oikeaoppisen käyttämisen ja järkevien käytänteiden selvitys merkityksellistä kaikille mobiilisovelluskehityksestä kiinnostuneille kehittäjillä ja muille alan toimijoille.

React Native valikoitui tutkimuksen kohteeksi sen suosion vuoksi. Suosion perusta koostuu useammasta eri syystä:

- React Nativen alustariippumattomuudesta eli samasta koodipohjasta voidaan renderöidä niin Android, iOS kuin Windows -sovellus.
- Käyttää samoja käyttöliittymän rakennuspalikoita kuin natiivit iOS ja Android-sovellukset.
- Mahdollistaa nopean ja kustannustehokkaan sovelluskehityksen.
- On käytetty suosittujen sovellusten kehittämiseen, kuten Facebook, Instagram, Skype, Airbnb ja paljon muita.

Opinnäytetyön käytännön osuudessa syntyy yksinkertainen demosovellus, josta käy ilmi React Nativella tehtävän sovelluksen rakennetta ja toimintaa. Kehitystyössä pyritään keskittymään hyviin käytänteisiin, ja työn koodiin tutustumalla voidaan tuoda ilmi näitä käytänteitä. Sovellukseen käyttäjä pystyy tallentamaan muistiin esineitä ja luomaan näiden pohjalta pakkauslistoja käytettäväksi haluamiinsa tarkoituksiin, kuten matkoja tai vaelluksia varten.

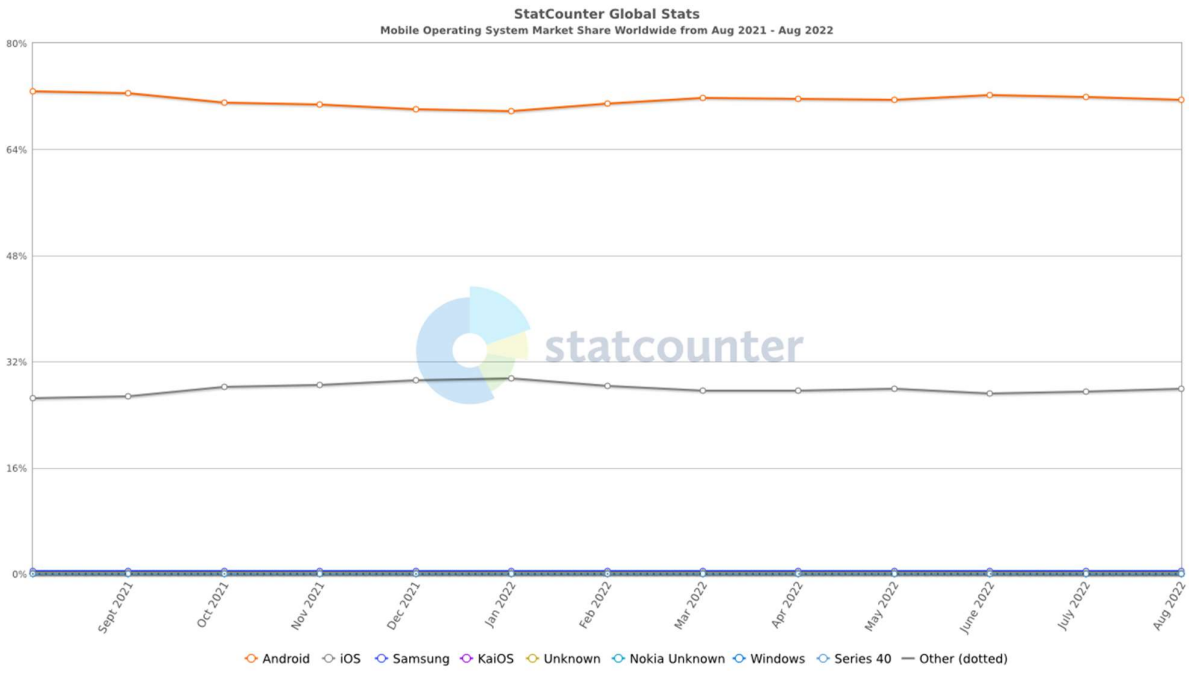
2 Mobiilisovelluskehitys

Mobiilisovelluskehitys on prosessi, jossa luodaan ja toteutetaan mobiililaitteilla kuten matkapuhelimilla toimivia ohjelmistosovelluksia (What is Mobile Application Development? n.d.).

2.1 Mobiilisovellusten alustat

Mobiilisovellusten alustoista puhuttaessa markkinoilla on nykyään kaksi selvästi hallitsevaa alustaa. Ensimmäinen näistä alustoista on Applen iOS-käyttöjärjestelmä, joka toimii alustana kyseisen yrityksen iPhone-älypuhelimissa. Toinen samankaltaisessa asemassa oleva alusta on Googlen Android-käyttöjärjestelmä, joka ei toimi alustana vain Googlen omissa laitteissa, vaan myös useiden muiden yritysten mobiililaitteiden käyttöjärjestelmänä. (What is Mobile Application Development? n.d.)

Vuoden 2022 elokuussa mobiililaitteiden käyttöjärjestelmien markkinaosuuksia vertailtaessa Googlen Android-käyttöjärjestelmällä oli selvästi suurin markkinaosuus (71.52 %). Samalla ajanjaksolla toiselle sijalle markkinaosuuden suuruudessa ylsi Applen iOS-käyttöjärjestelmä (27.83 %). Muiden käyttöjärjestelmien markkinaosuudet jäivät mitattuna ajankohtana kauas näistä kahdesta (alle 1 %). Nämä luvut havainnollistavat kahden suurimman alustan merkitystä mobiilisovelluskehityksessä (ks. kuvio 1).



Kuvio 1. Mobiililaitteiden käyttöjärjestelmien markkinaosuudet 08.2021 – 08.2022. (Mobile Operating System Market Share Worldwide Aug 2021 – Aug 2022 2022.)

2.2 Kehitystavat

Kehitysmenetelmät jaetaan neljään eri pääkehitystapaan, kun puhutaan mobiilisovellusten kehittämisestä. Nämä ovat natiivisovelluskehitys, alustariippumaton sovelluskehitys, hybridisovelluskehitys ja progressiivinen web-sovellus (PWA). Kaikilla neljällä kehitystavalla on omat hyötynsä ja heikkoutensa. (What is Mobile Application Development? n.d.)

2.2.1 Natiivisovelluksen kehittäminen

Natiivisovellukset kehitetään kehitettävän alustan tarjoamilla ohjelmointikielillä ja kehyksillä ja ne toimivat suoraan mobiililaitteen käyttöjärjestelmällä. Esimerkiksi Applen iOS-käyttöjärjestelmälle kehitettävä natiivisovellus kehitetään käyttäen joko Swift tai Objective-C ohjelmointikieltä (Getting Started with iOS App Development. n.d.) ja Googlen Android-käyttöjärjestelmälle kehitettävä natiivisovellus kirjoitetaan käyttäen joko Java tai Kotlin ohjelmointikieltä (Getting Started with Android App Development. n.d.).

Natiivikehityksen hyötyjä ovat verrattuna muihin kehitystapoihin ovat paras ajonaikainen suorituskyky sekä suora pääsy mobiililaitteen ohjelmointirajapintaan. Natiivikehityksen haittapuoliksi voidaan mainita esimerkiksi kehityksen korkeammat kustannukset ja eri koodikannat jokaiselle alustalle. (What is Mobile Application Development? n.d.)

2.2.2 Alustariippumattoman sovelluksen kehittäminen

Alustariippumattoman mobiilisovelluksen kehittämiseen on vaihtoehtona useita eri ohjelmointikieliä sekä kehyksiä, mutta kehitetty sovellus käännetään suoraan laitteen käyttöjärjestelmällä toimivaksi natiivisovellukseksi. Esimerkkeinä käytettävistä ohjelmointikielistä ja kehyksistä Facebookin React Native-kehys ja siinä käytettävä JavaScript-ohjelmointikieli sekä Googlen Flutter-kehys ja siinä käytettävä Dart-ohjelmointikieli. (What is cross-platform mobile development? 2022)

Alustariippumattoman kehityksen hyödyiksi voidaan katsoa sama koodikanta kaikille alustoille sekä kehityksen nopeus ja kustannustehokkuus. Haittapuolina ovat riippuvuus ulkopuolisista kirjastoista ja niiden käytön vaikutus suoritustehoon. (What is Mobile Application Development? n.d.)

2.2.3 Hybridisovelluksen kehittäminen

Hybridisovellukset kehitetään käyttäen perinteisiä web-teknologioita kuten JavaScript, CSS ja HTML. Hybridisovellus pakataan asennuspaketiksi, mistä se asennetaan halutulle laitteelle. (What is Mobile Application Development? n.d.)

Hybridisovelluksen hyötyjä ovat yhtenäinen koodikanta web- ja mobiilisovellusten välillä ja web-teknologioiden hyödyntäminen mobiilikehityksessä. Haittapuolina natiivisovelluksia matalampi suoritusteho sekä rajallinen pääsy laitteen ominaisuuksiin. (What is Mobile Application Development? n.d.)

2.2.4 Progressiivisen web-sovelluksen kehittäminen

Progressiivinen web-sovellus (PWA) eroaa muista kehitystavoista siinä, että sitä ei asenneta laitteelle. PWA-sovellus toimii mobiililaitteen selaimessa ja käyttää hyödykseen selaimen ominaisuuksia luodakseen mobiilisovelluksen tapaisen käyttäjäkokemuksen. (What is cross-platform mobile development? 2022)

Hyötyjä ovat saman sovelluksen käyttäminen sekä web- että mobiilialustoilla ja sovelluksen käyttäminen ilman sen asentamista laitteeseen. Haitat vuorostaan ovat käytetyn selaimen sanelemat ominaisuudet sekä laitteen omien ominaisuuksien rajallinen käyttö. (What is Mobile Application Development? n.d.)

3 React Native

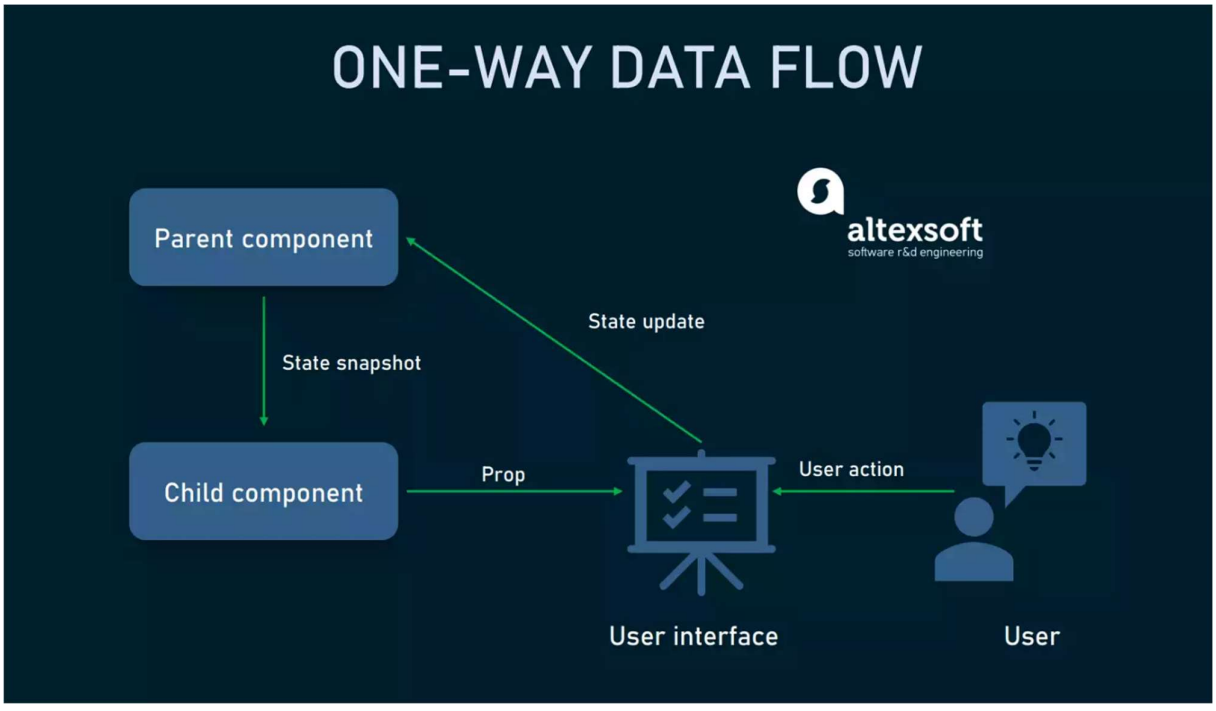
React Native on Facebookin vuonna 2015 julkaisema ja edelleen vuonna 2022 ylläpitämä mobiilisovelluskehitykseen suunniteltu ohjelmointikehys. React Native on alustariippumaton eli sitä käyttämällä voidaan kehittää sovellus useammalle eri alustalle käyttäen samaa koodipohjaa. Metan mukaan React Native yhdistää natiivin mobiilikehityksen parhaat puolet käyttäen Reactia ja sitä on käytetty tuhansien eri sovellusten kehittämisessä. (React Native n.d.)

3.1 Toiminta

3.1.1 React

Dabitin (2019) mukaan yksi React Nativen hyvistä ominaisuuksista on, että se käyttää Reactia. React on avoimen lähdekoodin JavaScript-kirjasto, joka on alunperin kehitetty käyttäjärajapintojen luomiseen web-sovelluksiin (React n.d; Dabit 2019). Nykyään Reactia voidaan kuitenkin käyttää myös palvelinpuolen kehityksessä sekä mobiilikehityksessä (React Native).

Reactilla rakennetuissa sovelluksissa data kulkee vain yhteen suuntaan (ks. kuvio 2). Kuten havainnollistavasta kuviosta huomataan, data kulkee vain kantakomponentista lapsikomponentille, eikä toisin päin. Munirin (Munir n.d.) mukaan yhdensuuntaisesta tiedonvälityksestä on hyötyä esimerkiksi virheiden paikallistamisessa, tehokkuuden parantamisessa sekä ohjelmistokoodin hallittavuudessa (Munir n.d.).



Kuvio 2. Tiedon kulku Reactissa (The Good and the Bad of React Development 2021)

3.1.2 JSX

JSX on lyhenne sanoista JavaScript XML (React JSX n.d.). JSX on siis JavaScriptin syntaksin laajennus. JSX:n käyttö ei ole pakollista, mutta se helpottaa React-sovellusten kehittämistä (React JSX n.d.). Myös Reactin dokumentaatiosivuilla kehotetaan käyttämään tätä syntaksin laajennusta Reactilla sovelluksia kehittäessä (Introducing JSX n.d.).

3.1.3 Komponentit

Komponentit muodostavat React Nativessa perustan käyttöliittymän rakentamiselle. Komponentit hallitsevat sekä ulkoasua että käyttöliittymän takana tapahtuvaa logiikkaa. React Nativen kehittäjät sekä kolmannet osapuolet ovat luoneet erittäin suuren määrän valmiita komponentteja käytettäväksi erilaisten vaatimusten sekä vaikeuksien ylitsepääsemiseksi, mutta Holmesin (Holmes 2022) mukaan on kuitenkin React Nativea käyttäessä ensiarvoisen tärkeää osata tehdä kustomoituja komponentteja vastaamaan omiin yksilöllisiin tarpeisiin.

3.2 Hyödyt

React Nativella kehitetty mobiilisovellus käyttää sitä pyörittävän alustan natiiveja ohjelmointirajapintoja, joka on suuri etu sovelluksen käyttökokemuksen ja toiminnan kannalta. Natiivit komponentit saavat sovelluksen tuntumaan ja näyttämään natiivilta sovellukselta sekä sovelluksen toiminta on sulavaa. (Chapter 1. What Is React Native? n.d.)

Natiiveja mobiilikehitys tekniikoita käyttäen täytyy sovellus pakata aina uudestaan muutosten jälkeen, johon voi aikaa mennä yllättävänkin paljon. React Native tarjoaa tähän ongelmaan helpotusta. React Nativella ohjelmistokoodiin tehdyt muutokset näkyvät käytännössä heti päivittämällä sovelluksen, mikä säästää huomattavasti aikaa ja helpottaa kehitystyötä (Dabit 2019; Chapter 1. What Is React Native? n.d.).

3.3 Haitat

Ensimmäisenä React nativella kehittämisen haittapuolista tulee esiin sen ikä. Vuonna 2015 julkaistu teknologia on natiiveja kehitystyökaluja huomattavasti uudempi (Dabit 2019). Tämä vaikuttaa tarjolla olevan tiedon sekä taidon määrään. On kuitenkin otettava huomioon, että teoksen

(Dabit 2019) julkaisemisesta on kulunut jo kolme vuotta tämän tutkimuksen teko hetkellä ja React Nativen elinkaari on siis lähes tuplaantunut.

React Native on kehitetty käyttämään sovellusta pyörittävän alustan ohjelmointirajapintoja (API). Näiden alustojen kuten iOS tai Android julkaistaessa uusia versioita käyttöjärjestelmistä saattaa React Native jäädä kehityksessä jälkeen, ennen kuin käyttöjärjestelmien päivityksiin ehditään reagoimaan. (Dabit 2019)

3.4 Hyvät käytänteet

Tässä luvussa tarkastellaan millaisia käytänteitä React Nativella tehtävässä sovelluskehityksessä suositellaan käytettäväksi sekä mitä käytänteitä tulisi vältellä.

3.4.1 Puhtaat komponentit

Käyttöliittymää luodessa käyttäen Reactia tai mobiilisovelluskehityksessä React Nativea, hajotetaan käyttöliittymä osiin ja niitä osia kutsutaan komponenteiksi. Komponenteille määritellään eriävät visuaaliset tilat (state) ja ne yhdistetään toisiinsa, jotta tieto kulkee niiden välillä. Sitä miten ja kuinka pieniksi komponentit jaetaan, voidaan lähestyä eri näkökulmien mukaan esimerkiksi kehittäjän taustan tai kehitettävän käyttöliittymän monimutkaisuuden perusteella. Yhtenä perusteena sille miten komponentit jaetaan, voidaan pitää single responsibility principle -periaatetta. (Thinking in React. N.d.)

Single responsibility principle (SRP) on periaate, jonka mukaan jokaisella ohjelman moduulilla saisi olla vain yksi syy muuttua. Toisin sanoen tämän periaatteen mukaan täytyisi kerätä yhteen asiat, jotka muuttuvat samoista syistä ja eritellä muista syistä tapahtuvat asiat (Martin. R. 2014). Tätä periaatetta React -komponentteihin soveltamalla voidaan pitää hyvänä käytänteenä jakaa komponentit niin, että jokainen komponentti suorittaa vain yhtä sille tarkoitettua tehtävää.

3.4.2 Tyylit

Mobiilisovellusten frontend-kehitystä tehtäessä on hyvin keskeisessä osassa sovelluksen ulkonäön rakentaminen vastaamaan haluttua lopputulosta. Voidaan puhua sovelluksen tyylittelystä ja tyy-

lien implementoinnista. Sovelluskehittäjän näkökulmasta on esimerkiksi tärkeää ajatella miten sovelluksen tyylittely olisi järkevää toteuttaa niin että eri komponentit voivat hyödyntää samoja tyylejä, sovelluksen ohjelmakoodi on selkeää sekä tyylejä koskeva koodi löytyy helposti ja on helppolukuista.

React Nativessa tyylittely tehdään käyttäen JavaScriptiä. React Nativen kaikki juurikomponentit hyväksyvät properties-objektin nimeltä style. Tyyli voidaan määrittää suoraan inline-tyylimäärittelynä halutulle komponentille tai voidaan käyttää muusta koodista erillään olevaa StyleSheet-komponenttia. Tyylien nimet ja arvot yleensä muistuttavat tunnetun ja paljon web-puolella tyylittelyyn käytetyn CSS-ohjelmointikielen vastaavia, mutta sanat on kirjoitettu yhteen isoin alkukirjaimin (camelCase). (Style n.d.) Dabitin (2019) mukaan yhtäläisyydet CSS ja React Native tyylittelyn välillä loppuvat kuitenkin tähän. Dabit (2019) kehottaakin React Native-sovelluksen tyylittelyssä päästämään irti mahdollisesti jo opituista CSS-käytänteistä ja opettelemaan käyttämään hyväksi JavaScriptillä tehtävän tyylittelyn edut. (Dabit 2019.)

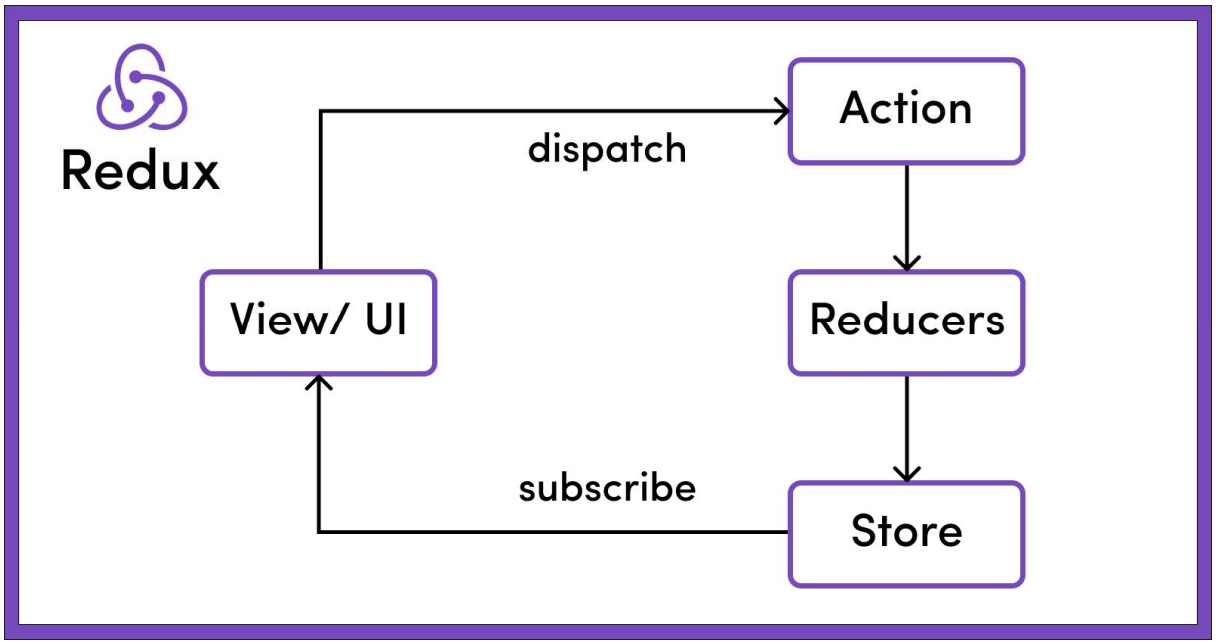
3.4.3 Redux

Dabitin (2019) mukaan React Native sovelluksia kehittäessä voi datan käsittelystä helposti tulla monimutkainen ja sekava prosessi ilman tarkkaa sekä harkittua lähestymistapaa. Datan käsittelyn voi suorittaa React Native sovelluksessa paitsi käyttämällä komponentin tilaa(state) ja jakamalla sen properties-objektina(props) tarvitsemaansa paikkaan sovelluksessa myös käyttämällä avuksi erillistä kirjastoa datan käsittelyyn. (Dabit 2019.)

Redux tarjoaa ennustettavaa tilanhallintaa JavaScript-sovelluksia varten (Getting Started with Redux 2022). Dabitin (2019) mukaan Redux on Reactilla kehitetyissä sovelluksissa laajimmin käytetty tapa datan käsittelyyn. Reduxin avulla on mahdollista yksinkertaistaa sovelluksen tilanhallintaa siirtämällä se kaikki samaan paikkaan(store). Tällöin tiedetään aina tarvittavan datan tarkka sijainti sovelluksessa ja voidaan olettaa tämän datan olevan ajan tasalla sekä käytettävissä myös muualla sovelluksessa. (Dabit 2019.)

Ensimmäisenä Reduxin toiminnan kolmessa pääperiaatteesta on koko sovelluksen yleisen tilan sijainti objektipuussa yhdessä tilasäilössä(store), mikä helpottaa esimerkiksi sovelluksen tarkastelua

sekä testausta. Toinen pääperiaate on, että tilaa voidaan vain lukea eli muuttaakseen tilasäilössä(store) olevaa tilaa täytyy lähettää(dispatch) toiminto-objekti(action), joka kertoo mitä muutoksia tilaan halutaan tehdä. Viimeinen pääperiaatteista on, että halutut tilan muutokset toteutetaan puhtailla funktioilla(reducers) toimintojen perusteella (ks. kuvio 3). (Three Principles 2021.)



Kuvio 3. Reduxin toiminta (What is Redux? Store, Actions, and Reducers Explained for Beginners 2022)

4 Tutkimusasetelma

4.1 Ongelma

React Native on aloittelijaystävällinen ja helppokäyttöinen ohjelmointikehys, jota väärin käyttämällä tai huonojen tapojen mukaan ohjelmoitaessa koodipohjasta voi tulla sekava ja luotu sovellus ei välttämättä toimi parhaalla mahdollisella tavalla. Tätä välttääkseen on syytä tutkia React Nativella kehittämisen hyviä käytänteitä ja toimintatapoja. Näin saadaan lisää tutkimuspohjaa asiasta ja voidaan välttää jatkossa turhien virheiden tekemistä ja parantaa React Nativella tapahtuvan mobiilisovelluskehityksen tehokkuutta.

4.2 Tavoite

Tutkimuksen tavoitteena on saada alan toimijoita hyödyttävää tietoa, jota voidaan jatkossa käyttää apuna päätöksen teossa koskien mobiilisovelluskehitystä. Tuloksista tavoitellaan olemaan apua niin oikean teknologian valinnassa mobiilisovellusta kehitettäessä, kuin React Nativella kehitettävän sovelluksen kehitysmenetelmien valinnassa.

Tutkimuksessa pyritään vastamaan kahteen tutkimuskysymykseen. Mitkä ovat React Nativella suoritettavan sovelluskehityksen hyviä käytänteitä sekä mitä hyötyjä ja haittoja löytyy React Nativen käytössä sovelluskehityksessä.

4.3 Esiolettamukset työn lopputuloksista

Työn lopputuloksena syntyy selvitys siitä, mitä React Nativella suoritettavan sovelluskehityksen menetelmiä ja käytänteitä suosien pääsee selkeään ja hyvin toimivaan lopputulokseen. Tutkimuksesta tulee saamaan kattavan tietopaketin koskien mobiilisovelluskehitystä, etenkin React Nativella kehittämisen näkökulmasta. Lukija pystyy soveltamaan tutkimuksesta saamaansa tietoa omissa projekteissaan ja mahdollisesti tämän tiedon avulla välttämään epäkäytännöllisiä menetelmiä.

4.4 Tutkimusmenetelmä

Kehittämistyö on merkittävässä roolissa useissa tietojenkäsittelyn ammatillisissa rooleissa. Ohjelmistokehityksen parissa työskenneltäessä kehitystyö tehdään aina tarkasti suunnitelluissa projekteissa, jolloin haluttuun tavoitteeseen päästään käytössä olevilla resursseilla sekä ennalta määritellyssä aikataulussa. Laajojen ja monimutkaisten kokonaisuuksien jakaminen pienempiin osiin sekä sykleissä tapahtuva iteratiivinen tapa toimia on keskeisessä osassa ohjelmistokehitystä. Näitä toimintamalleja voidaan hyvin soveltaa myös alan tutkimuksellisen kehittämistyön tekemisessä (Bister 2019).

Tutkimukselliselle kehittämistyölle on olennaista ennen varsinaiseen kehittämiseen liittyvän työn aloittamista kerätä monipuolisesti tietoa eri lähteistä ja näin luoda mahdollisimman syvä ymmär-

rys tutkittavasta kohteesta sekä kehittämisen mahdollisuuksista. Varsinaista kehittämistyötä suorittaessa havainnoidaan omaa tekemistä sekä kerätään tietoa koskien tutkittavaa kohdetta. Näin voidaan verrata ja pohtia kehittämistyössä syntyneitä havaintoja kerättyyn tietoperustaan, jolloin saadaan muodostettua tuloksia (Bister 2019).

Työlle asetetut tavoitteet ovat ratkaistavissa hyvällä tiedonhaulla sekä löydetyn tiedon kokeilemisella käytännössä esimerkkisovelluksen kehitystyössä. Käytännön testauksen myötä saadaan parempaa tietoa siitä miten oletetut parannukset toimivat ja voidaan pohtia niiden merkitystä sovelluksen kehityksessä.

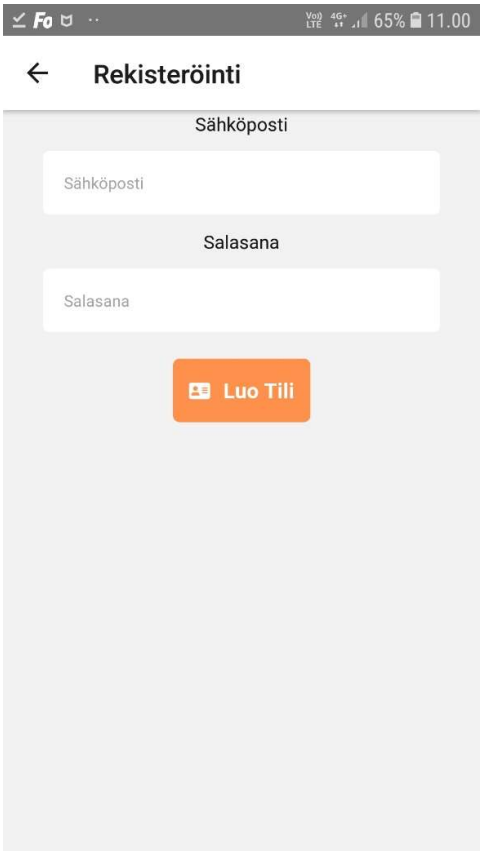
Kehittämistyön kohteena on React Native -sovelluskehys ja sen järkevä käyttäminen mobiilisovelluskehityksessä. Aineistona toimii erinäisistä artikkeleista ja julkaisuista kerätty tietoperusta ja esimerkkisovellukseksi kehitetyn sovelluksen koodipohja. Ennen tutkimuksen aikana toteutettavan sovelluksen kehittämistä kerätään tietopohjaa hyvistä käytänteistä ja siitä, miten sovellus tulisi oikeaoppisesti rakentaa. Toteutunutta sovellusta tarkastelemalla voidaan analysoida käytettyjen toimintatapojen merkitystä ja niistä johtuvien haittojen tai hyötyjen suuruutta kehitystyössä sekä sovelluksen toiminnassa.

5 Sovellus

Tutkimuksen tuloksena syntynyt mobiilisovellus toimii käyttäjälleen apuna pakkaamisessa. Rekisteröidyttyään ja kirjauduttuaan sisään sovellukseen käyttäjä pääsee luomaan, hallitsemaan ja käyttämään pakkauslistoja. Käyttäjä luo sovellukseen tavaroita, joita haluaa pakata mukaansa matkoilleen. Sovellukseen luoduista tavaroista käyttäjä koostaa pakkauslistoja sovellukseen.
Pakkauslistoja käyttäjä voi käyttää apunaan pakatessa sekä merkata sovellukseen tavarat, jotka on jo pakannut matkatavaroihin.

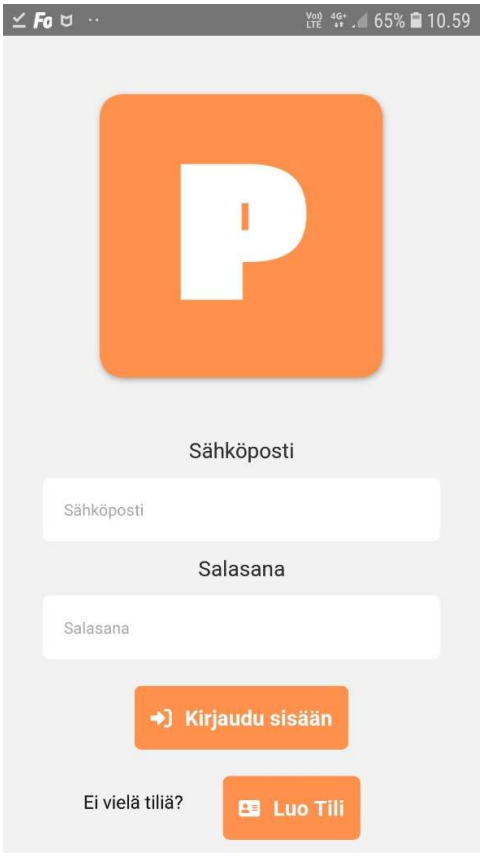
5.1 Rekisteröinti ja kirjautuminen

Autentikaatio on keskeisessä osassa kehittämääni demosovellusta, ja se sisältää sekä rekisteröitymisen että kirjautumisen. Nämä toiminnot on toteutettu erillisissä komponenteissa, jotka hyödyntävät Firebase Authentication -palvelua varmistaakseen käyttäjien turvallisen pääsyn sovellukseen.



Kuvio 4. Rekisteröitymisnäkymä

Rekisteröinti ja kirjautuminen ovat kaksi erillistä toimintoa, mutta niiden tekninen toteutus jakaa samankaltaisia piirteitä. Käyttäjätilin luominen (rekisteröinti) on toteutettu ”RegisterForm”-komponentissa (ks. kuvio 5), kun taas kirjautuminen ”Loginform”-komponentissa (ks. kuvio 6). Komponenttien näkymät ovat pelkistettyjä ja tarjoavat käyttäjälle mahdollisuuden navigoida rekisteröitymisen ja kirjautumisen välillä. Molemmat komponentit edellyttävät käyttäjältä sähköpostin ja salasanan syöttämistä omiin tekstikenttiinsä. Syötteet tallennetaan tiloihin (state), mistä niitä voidaan käyttää ja validoida.



Kuvio 5. Kirjautumisnäkymä

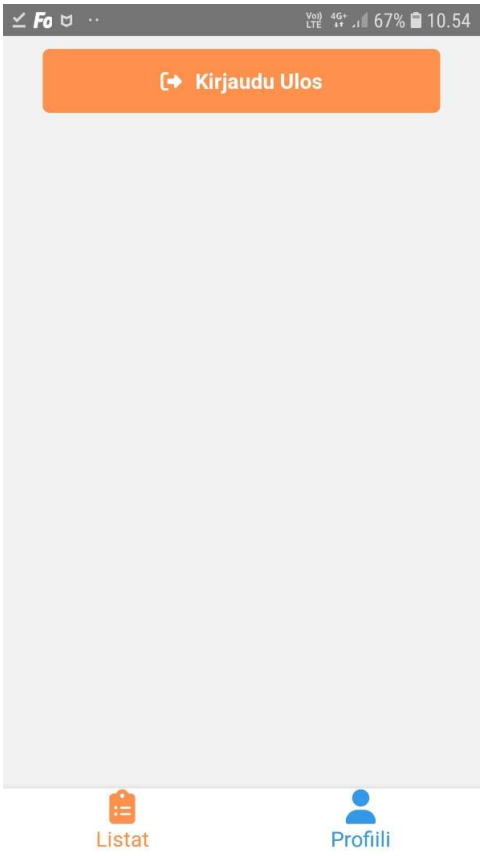
Sekä rekisteröitymis- että kirjautumisprosessit tarjoavat käyttäjälle virheilmoituksia, jos syötetyissä tiedoissa on puutteita tai virheitä. Firebase Authentication -palvelun virhetilanteissa näytetään käyttäjälle virheeseen sopiva virheilmoitus, joka auttaa käyttäjää ymmärtään, mikä on vikana (ks. kuvio 6).

```
const onLoginPress = () => {
  let emailValid = validateEmail(email);
  let passwordValid = validatePassword(password);
  if (!emailValid || !passwordValid) return;
  setProcessing(true);
  auth()
    .signInWithEmailAndPassword(email, password)
    .then(() => {
      console.log('Signed in with email & pw');
      setProcessing(false);
    })
    .catch(error => {
      setProcessing(false);
      if (error.code === 'auth/email-already-in-use') {
        console.log('Sähköposti on jo käytössä');
        setLoginError('Sähköposti on jo käytössä');
      } else if (error.code === 'auth/invalid-email') {
        setEmailError('Sähköposti on väärässä muodossa');
      } else if (
        error.code === 'auth/wrong-password' ||
        error.code === 'auth/invalid-email'
      ) {
        setLoginError(
          'Kirjautuminen epäonnistui. Tarkista sähköposti ja salasana',
        );
        console.log('Wrong email or password');
      } else {
        console.log(error.code);
        setLoginError(
          'Kirjautuminen epäonnistui. Tarkista sähköposti ja salasana',
        );
      }
      console.log(error);
    });
};
```

Kuvio 6. Kirjautumisen mekaniikka

5.2 Profiili

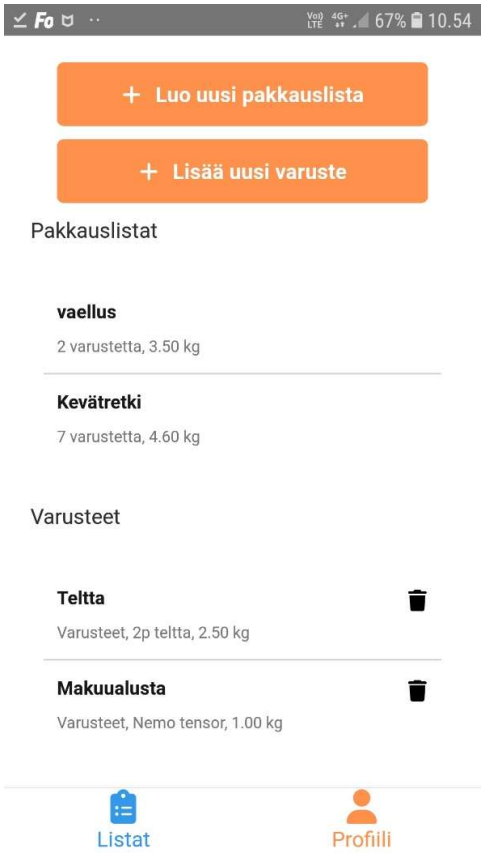
Sovelluksen profiilinäkymä on pelkistetty näkymä, mistä löytyy painike käyttäjän sovelluksesta ulos kirjautumista varten (ks. kuvio 7). Käyttäjän painaessa ”Kirjaudu Ulos”-painiketta käyttäjä kirjataan ulos sovelluksesta Firebase Autentication palvelun kautta.



Kuvio 7. Sovelluksen profiilinäkymä

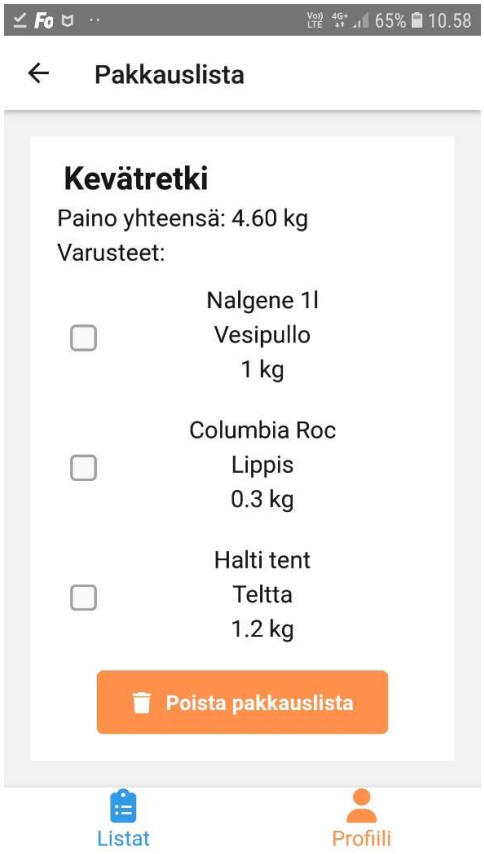
5.3 Listat

Käyttäjän kirjautuessa sisään sovellukseen aukeaa sovelluksen päänäkymä, josta käyttäjä pääsee navigoimaan luomaan uuden varusteen tai pakkauslistan sovellukseen ja selaamaan sekä hallitsemaan sovellukseen jo luotuja pakkauslistoja ja varusteita (ks. kuvio 8).



Kuvio 8. Sovelluksen päänäkymä

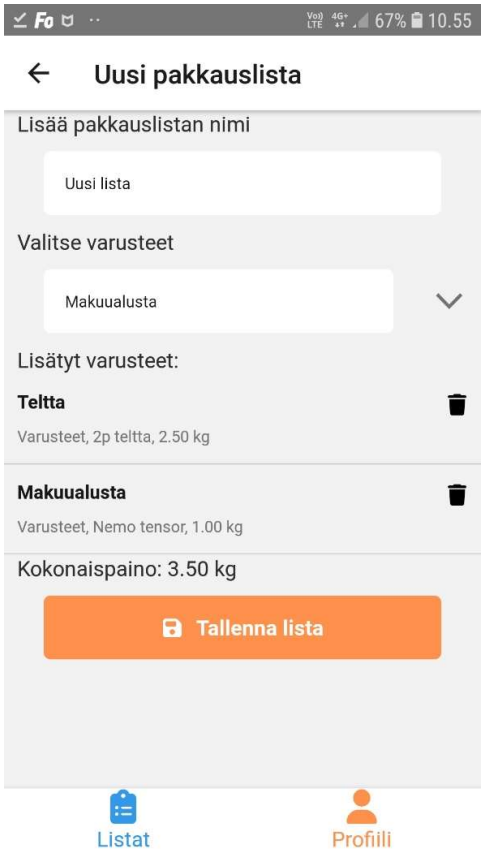
Käyttäjä pystyy painamaan sovelluksen päänäkymästä haluamaansa pakkauslistaa, jolloin pakkauslista aukeaa sovellukseen. Avautuvasta näkymästä käyttäjä näkee valitsemansa pakkauslistan nimen, pakkauslistan kokonaispainon sekä kaikki pakkauslistalle valitut tavarat (ks. kuvio 9). Halutessaan käyttäjä voi käyttää sovelluksessa avattua pakkauslistaa apuna pakatessaan merkitsemällä jo pakkaamansa tavarat niiden vieressä olevaa laatikkoa painamalla. Painamalla ”Poista pakkauslista”-painiketta käyttäjä pystyy poistamaan sovelluksesta pakkauslista, jota ei enää tarvitse.



Kuvio 9. Pakkauslista auki painettuna

5.4 Uuden pakkauslistan lisääminen

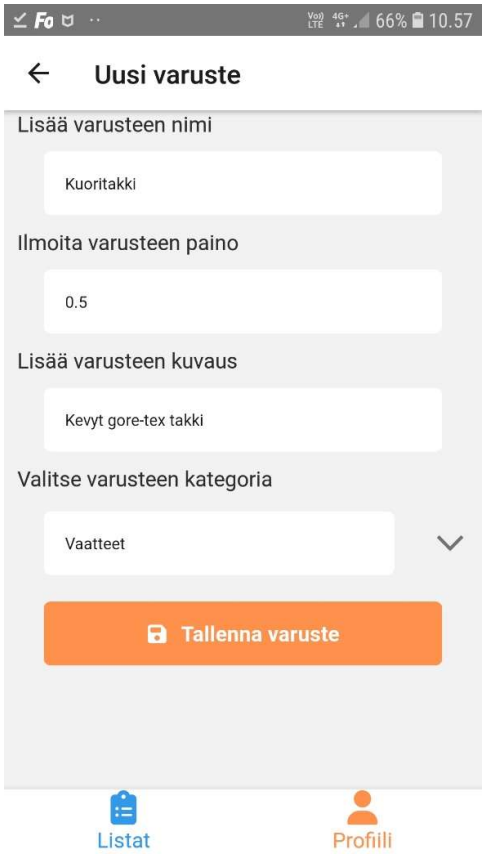
Sovellukseen tallentamistaan tavaroista käyttäjä pystyy muodostamaan pakkauslistoja sovellukseen. Luodessaan uutta pakkauslistaa sovellukseen käyttäjä pystyy asettaman listalle kuvaavan nimen, jolla käyttäjä erottaa pakkauslista muiden luomiensa listojen seasta. Käyttäjä pystyy koostamaan pakkauslistan sovellukseen tallentamistaan tavaroista. Uutta pakkauslistaa luodessaan käyttäjä näkee koko ajan tavaroita pakkauslistalle valitessaan kokonaispainon, mikä helpottaa matkatavaroiden painon hallinnassa (ks. kuvio 10).



Kuvio 10. Uusi pakkauslista sovellukseen

5.5 Uuden varusteen lisääminen

Käyttäjä pystyy tallentamaan sovellukseen matkoilla tarvitsemiaan tavaroita. Nimen lisäksi käyttäjä pystyy asettamaan tavaroita luodessaan niille lyhyen kuvauksen sekä valita tavaralle kategorian helpottamaan tavaran tunnistamista. Tavaroille käyttäjä pystyy asettamaan painon, mikä auttaa pitämään silmällä matkatavaroiden kokonaispainoa ja pitämään sen haluamallaan tasolla (ks. kuvio 11).



Kuvio 11. Uusi varuste sovellukseen

6 Pohdinta

Käytännön tutkimus Facebookin (nykyisen Metan) kehittämän React Native-kehyksen käyttöön mobiilisovelluksen kehittämisessä on tuonut ilmi niin työskentelyä helpottavia kuin haasteita herättäviä puolia. Päätös tutkia juuri React Nativea syvemmin oli sen laaja käyttö mobiilisovelluksia kehittäessä, mikä puhui jo puolestaan tämän kehyksen monipuolisista eduista.

Yhdeksi React Nativen suurimmista eduista opinnäytetyön tietoperustassa nousi sen mahdollistama usean alustan mobiilisovelluksen kehittäminen yhtä aikaa ja yhdestä koodipohjasta. Ominaisuuden voidaan ajatella säästävän paljon aikaa ja kuluja kehitysprosessissa sekä kasvattavan valmiin sovelluksen käyttäjäkuntaa huomattavasti, jos sen vaihtoehtona olisi tehdä mobiilisovellus aluksi vain yhdelle alustalle. Tutkimuksessa syntyneen sovelluksen käytännön testaaminen eri alustoilla ei kuitenkaan ollut mahdollista, koska käytettävissä ei ollut siihen tarvittavaa laitteistoa. Tämän vuoksi tutkimuksessa keskityttiin React Nativen kehittämiseen Android-alustalle ja näin ollen

usean alustan yhtäaikaisen kehityksen käytännön toimivuus jäi demosovellusta kehitettäessä toteamatta.

Demosovelluksen koodia tarkasteltaessa voidaan huomata tiettyjä osa-alueita, jotka olisi voinut ratkaista eri lailla ja näin ollen päästä parempaan lopputulokseen. Sovelluksen rakenteen jakaminen pienempiin komponentteihin, joita olisi pystynyt hyödyntämään useissa sovelluksen osissa olisi tehnyt sovelluksen koodirakenteesta helpommin ymmärrettävää sekä vähentänyt työmäärää. Opinnäytetyön tietoperustassa esiin tuotu sovelluksen tyylien keskittäminen suurimmaksi osaksi yhteen paikkaan olisi osaltaan myös auttanut tekemään sovelluksen koodipohjasta selkeämmän sekä helpottanut myöhemmin mahdollisesti tapahtuvaa sovelluksen ylläpitoa. Näiden havaintojen perusteella voidaan alleviivata sovelluksen koodipohjan rakenteen sekä hyvien käytänteiden tärkeyttä React Native -kehyksellä suoritetun mobiilisovelluksen kehitystyössä, jotta kehyksen mahdollistamaa monipuolisuutta sekä tehokkuutta päästään hyödyntämään mahdollisimman tehokkaasti.

Tutkimuksen aikana esiin nousseiden hyötyjen lisäksi todettiin myös mahdollisia haasteita React Nativen käytössä mobiilisovelluksen kehittämisessä. Monimutkainen sovelluksen tilanhallinta pystyttiin havaitsemaan ongelmaksi jo tutkimusta varten kehitetyssä pienessä demosovelluksessa, mistä voidaan tehdä johtopäätös ongelman vain kasvavan entisestään sovelluksen laajuuden kasvaessa. Tähän ongelmaan löytyi kuitenkin ratkaisuksi opinnäytetyön tietoperustassa esitelty tilanhallintatyökalu Redux, jonka käyttöön ottaminen demosovellusta kehittäessä teki tilanhallinnasta huomattavasti selkeämpää ja helpompaa.

Kokonaisuudessaan tutkimuksen perusteella voidaan todeta React Nativen olevan potentiaalinen valinta kehykseksi mobiilisovelluksien kehittämistä suunnitellessa. React Nativen voidaan ajatella tarjoavan mobiilisovelluksen kehittämiseen monipuolisuutta ja tehokkuutta sen usealle alustalle yhtäaikaisen kehityksen sallivan luonteen perusteella sekä esimerkiksi laajan ja ylläpidettyjen ulkoisten kirjastojen valikoiman vuoksi. React Nativella on myös edelleen vuonna 2023 elinvoimainen ja runsas tukiyhteisö, jonka lasken suureksi eduksi kehystä käytettäessä. React Nativen alati kasvava ja jo lähes vuosikymmenen kestänyt elinkaari kertoo myös jotain siitä, miten kehys tarjoaa edelleen tutkimuksen teon hetkellä oivan ratkaisun mobiilisovellusta kehittäessä.

7 Luotettavuus ja eettisyys

Opinnäytetyön tietoperustaa varten on tietoa kerätty tutkimuksessa käytettyjen teknologioiden virallisista dokumentaatioista sekä tunnistetusta aihetta käsittelevästä kirjallisuudesta. Suurimmaksi osaksi virallisiin ja vakiintuneisiin tietolähteisiin keskittymällä pyrittiin varmistamaan tutkimukseen kerätyn tiedon tarkkuus ja autenttisuus. Näiden tutkimuksessa mainittujen lähteiden käyttö edesauttoi vakaan tietoperustan luonnissa tutkimuksen pohjaksi.

Tutkimuksen osana syntynyt esimerkkisovellus toimi tietoperustaan kerätyn tiedon käytännöntoteutuksena. Esimerkkisovelluksen tarkoitus tutkimuksessa oli tutkia tietoperustaan kerätyn teorian toimivuutta oikeassa käytännön projektissa. Tutkijan omakohtaiset kokemukset esimerkkisovellusta kehittäessä toivat tutkimukseen tärkeän ulottuvuuden. Tutkimusta luettaessa on syytä pitää mielessä sovelluksen kehittämisestä saatujen tietojen olevan havainnollistavia, mutta ne tiedot ovat yksinomaan tutkijan omaan kokemukseen perustuvia havaintoja. Tutkijalla ei myöskään ollut ennestään vahvaa osaamista ja tietotaitoa tutkimuksen kohteena olevista teknologioista, mikä omalta osaltaan vaikutti tutkimustuloksiin.

Koko tutkimuksen ajan on painotettu eettistä harkintaa ja pyritty hyvään tieteelliseen käytäntöön sekä läpinäkyvyyteen kerätyn tiedon alkuperästä. Tietoperustan keräämiseen käytetyt julkaisut ovat kaikki merkattu opinnäytetyön lähdeluetteloon ja niihin löytyy viitteet tekstistä. Tietoperustaa on kerätty pääosin tutkimuksessa käytettyjen teknologioiden virallisilta sivustoilta ja tämän lisäksi muiden alalla tunnettujen toimijoiden sivustoilta, artikkeleista sekä opetusmateriaaleista.

Vaikka tutkimus nojaakin vahvasti ulkopuolisista lähteistä kerättyyn tietoperustaan sisältää tutkimus myös omakohtaisiin kokemuksiin perustuvaa tietoa. Tutkimuksessa on kiinnitetty huomiota tutkijan omien havaintojen ja kokemusten perusteella syntyneiden tulosten esittämiseen tutkijan omakohtaisena kokemuksena. Hyvällä tasapainolla näiden kahden elementin välillä tutkimuksessa pyrittiin muodostamaan kokonaisvaltainen, eettinen sekä luotettava katsaus alustariippumattomaan mobiilikehitykseen käyttäen React Native -ohjelmointikehystä.

Lähteet

Mobile Operating System Market Share Worldwide Aug 2021 – Aug 2022. 2022. StatCounter. Viitattu 2.10.2022. https://gs.statcounter.com/os-market-share/mobile/worldwide.

Getting Started with iOS App Development. N.d. Artikkeli AWS:n dokumentaatiosivuilla. Amazon Web Services, Inc. Viitattu 2.10.2022. https://aws.amazon.com/mobile/mobile-application-development/native/ios/.

Getting Started with Android App Development. N.d. Artikkeli AWS:n dokumentaatiosivuilla. Amazon Web Services, Inc. Viitattu 2.10.2022. https://aws.amazon.com/mobile/mobile-application-development/native/android/.

What is Mobile Application Development? N.d. Artikkeli AWS:n dokumentaatiosivuilla. Amazon Web Services, Inc. Viitattu 2.10.2022. https://aws.amazon.com/mobile/mobile-application-development/.

What is cross-platform mobile development? 2022. Artikkeli Kotlinin dokumentaatiosivuilla. Viitattu 2.10.2022. https://kotlinlang.org/docs/cross-platform-mobile-development.html.

Dabit, N. 2019. React Native in action: developing iOS and Android apps with JavaScript. Shelter Island: Manning Publications. Viitattu 25.10.2022. https://janet.finna.fi/, Skillsoft Books ITPro.

Chapter 1. What Is React Native?. N.d. Artikkeli O,Reilly:n sivuilla. O'Reilly Media, Inc. Viitattu 25.10.2022. https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html.

React Native. N.d. Artikkeli React Nativen dokumentaatiosivuilla. Meta Platforms, Inc. Viitattu 25.10.2022. https://reactnative.dev/.

React. N.d. Artikkeli Reactin dokumentaatiosivuilla. Meta Platforms, Inc. Viitattu 26.10.2022. https://reactjs.org/.

Introducing JSX. N.d. Artikkeli Reactin dokumentaatiosivuilla. Meta Platforms, Inc. Viitattu 26.10.2022. https://reactjs.org/docs/introducing-jsx.html.

React JSX. N.d. Opetusmateriaali W3Schools-sivustolla. Refsnes Data. Viitattu 26.10.2022. https://www.w3schools.com/react/react_jsx.asp.

The Good and the Bad of React Development. 2021. Artikkeli Altexsoftin sivustolla. AltexSoft. Viitattu 26.10.2022. https://www.altexsoft.com/blog/react-pros-and-cons/.

Munir, S. What is unidirectional data flow in React? N.d. Artikkeli Educativen sivustolla. Educative, Inc. Viitattu 26.10.2022. https://www.educative.io/answers/what-is-unidirectional-data-flow-in-react.

Getting Started with Redux. 2022. Artikkeli Reduxin dokumentaatiosivuilla. Dan Abramov and the Redux documentation authors. Viitattu 7.11.2022. https://redux.js.org/introduction/getting-started.

De Roy, S. 2022. What is Redux? Store, Actions, and Reducers Explained for Beginners. Artikkeli freeCodeCampin sivustolla. Viitattu 8.11.2022. https://www.freecodecamp.org/news/what-is-redux-store-actions-reducers-explained/.

Three Principles. 2021. Artikkeli Reduxin dokumentaatiosivuilla. Dan Abramov and the Redux documentation authors. Viitattu 8.11.2022. https://redux.js.org/understanding/thinking-in-redux/three-principles.

Style. N.d. Artikkeli React Nativen dokumentaatiosivuilla. Meta Platforms, Inc. Viitattu 02.12.2022. https://reactnative.dev/docs/style.

Holmes, M. 2022. Creating Apps with React Native: Deliver Cross-Platform 0 Crash, 5 Star Apps. Berkeley, California: Apress. Viitattu 11.11.2023. https://janet.finna.fi/, Skillsoft Books ITPro.

Bister, T. 2019. Tietojenkäsittelyn opinnäytetyö: Viittoja ja karttoja tutkimisen ja kehittämisen teille. Jyväskylä: Jyväskylän ammattikorkeakoulu. Viitattu 25.11.2023. https://janet.finna.fi/, Booky.

Thinking in React. N.d. Opetusmateriaali Reactin tutoriaalisivulla. Meta Platforms, Inc. Viitattu 03.12.2023. https://react.dev/learn/thinking-in-react.

Martin, R. 2014. The Single Responsibility Principle. Kirjoitus The Clean Code Blog -blogissa. Julkaistu 08.05.2014. Viitattu 03.12.2023. https://blog.cleancoder.com/uncle-bob/2014/05/08/SingleReponsibilityPrinciple.html.