# React Nativen soveltuvuus HAMK tietojenkäsittely koulutuk-sessa opinnäytetyön tekijän näkökulmasta

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Kevät 2022

Teemu Viljanen

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittely

**Tekijä** Teemu Viljanen **Vuosi** 2022

**Työn nimi** React Nativen soveltuvuus HAMK Tietojenkäsittely koulutuksessa opinnäytetyön tekijän näkökulmasta

**Työn ohjaaja/t** Tommi Lahti

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli selvittää miten voisi toteuttaa React Nativella HAMK tietojenkäsittely koulutuksen Mobile Programming Cross-platform-kurssin keskeisimmät sisällöt React Nativella, mahdolliset ongelmat käytön aikana ja miten se soveltuu muun koulutuksen kanssa. Opinnäytetyön toimeksiantaja on HAMK tietojenkäsittely koulutus.

Opinnäytetyön teoriaosuudessa käydään läpi yleisesti mikä on React Native, sen arkkitehtuuri ja siihen liittyviä tekniikoita. Teoriaosuuden lopuksi arvioidaan React Nativen opittavuutta ja soveltuvuutta HAMK tietojenkäsittely koulutukseen. Käytännönosuudessa käydään läpi opinnäytetyön aikana tehdyt sovellukset ja kehitysympäristön asennus.

Opinnäytetyössä aikana saatiin selville, että React Native on web-kehittäjille suunnattu ohjelmistokehys mobiilisovellusten rakentamiseen. React Native on nopeasti opittavissa oleva vaikkei erityisesti ole kokemusta web-kehittämisestä. React Native soveltuu ja kykenee hyvin HAMK tietojenkäsittely koulutukseen ja samalla ajantasaistaa koulutuksen cross-platform-kurssin tekniikoita nykyaikaisemmaksi.

**Avainsanat** React Native, mobiilisovellukset, JavaScript, Android, iOS

**Sivut** 41 sivua ja 1 liite

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology

**Author** Teemu Viljanen **Year** 2022

**Subject** Suitability of React Native, HAMK

**Supervisors** Tommi Lahti

ABSTRACT

The objective of the thesis was to find out how to implement the key contents of the HAMK Business Information Technology's Mobile Programming Cross-Platform course with React Native, possible on-the-go problems during use and how it is suitable for training program. The thesis is commissioned by HAMK.

In the theory section of thesis, we go through in general what is React Native, its architecture and related techniques. At The end of the theory section is assessed learnability of React Native's and suitability for HAMK. The practical part includes the applications made during the thesis and the installation of the development environment.

During the thesis, it was discovered that React Native is a software framework aimed for web developers. Even though one is not particularly experienced in web development. React Native is still an easy to learn framework. React Native is well suited and capable for HAMK and at the same time good upgrade for the cross-platform course to make it a more modern.

**Keywords** React Native, mobile applications, JavaScript, Android, iOS

**Pages** 41 pages and 1 appendix

SISÄLLYS

**SANASTO**

| | |
|---|---|
| JS | JavaScript ohjelmointikieli |
| NPM | Node Package Manager, JavaScript-paketinhallintajärjestelmä |
| API | Application Programming Interface, Ohjelmointirajapinta |
| IDE | Integrated Development Environment, Ohjelmointiympäristö |
| REST | Representational State Transfer, Http-protokollalla toimiva rajapinta. |

# 1 JOHDANTO

Hämeen ammattikorkeakoulu tietojenkäsittely koulutuksen Mobile Programming moduuli on toteutettu aikaisemmin Xamarin-sovelluskehyksellä. Tämän kanssa on ollut paljon erilaisia vaikeuksia erityisesti kehitysympäristön asennuksen kanssa.

Opinnäytetyön tavoitteena on selvittää miten voisi toteuttaa Hämeen ammattikorkeakoulu tietojenkäsittelyn Mobile Programming moduulin Xamarin-sisällöt React Nativella ja miten se soveltuu HAMK tietojenkäsittelyn muun opetussisällön kanssa. Tavoitteena on myös selvittää minkälaisia ongelmia mahdollisesti kehitysympäristön asennuksen ja käytön aikana ilmenee. Opinnäytetyön tilaaja on Hämeen ammattikorkeakoulu tietojenkäsittely koulutus.

Opinnäytetyön teoriaosuudessa käydään läpi yleisesti mikä on React Native, sen käyttämiseen liittyvät kirjastot ja ohjelmat, joita käytetään käytännönosuuden toteuttamisessa. Teoriaosuudessa myös vertaillaan lyhyesti React Native- ja Xamarin-ohjelmistokehyksiä. Lopuksi arvioidaan React Nativen soveltuvuutta HAMK tietojenkäsittelyn muun opetussisällön kanssa.

Käytännönosuudessa asennetaan React Native ja sen käyttöön tarvittavat ohjelmat Windows 10 -käyttöjärjestelmälle. Tämän jälkeen toteutetaan muutama esimerkki sovellus mobile Programin-moduulin keskeisimmistä asioista esimerkiksi käyttöliittymän luonti, tiedon tallennus ja tiedonsiirto(pilvipalvelut).

Opinnäytetyössä pyritään vastaamaan seuraaviin kysymyksiin:

1. Miten Mobile Programming moduulin Xamarin-sisällöt voisi toteuttaa React Nativella (muutamat keskeisimmät asiakokonaisuudet)?
2. Xamarinin kanssa on ollut ongelmia etenkin kehitysympäristön asennuksen kanssa. Miten tämä React Nativella?
3. Miten React Native istuu muuhun opetussisältöömme (React on oma kielensä jne.)?

# 2 REACT NATIVE

## 2.1 ReactJS

React on Facebookin kehittämä JavaScript-kirjasto, joka on tarkoitettu verkkosivujen käyttöliittymien rakentamiseen ja datan näyttämisen loppukäyttäjälle. React on suunniteltu erityisesti mielessä pitäen suuret käyttöliittymät, jossa data vaihtuu dynaamisesti. MVC-arkkitehtuurissa React sijoittuu näkymä tasoon (View). (Paul Krill, 2014)

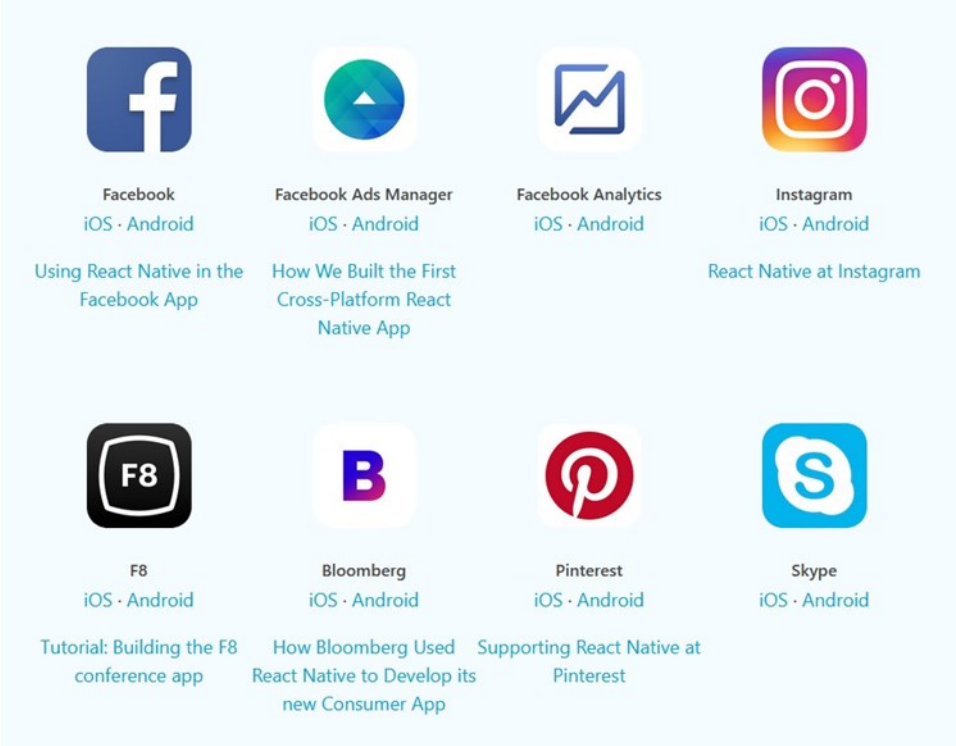
## 2.2 React Native

React Native on Facebookin kehittämä JavaScript-sovelluskehys, jota käytetään mobiilisovellusten kehittämiseen usealla alustalle samanaikaisesti (Facebook inc, Github.io). React Native perustuu Facebookin aikaisemmin kehitettyyn ReactJS-kirjastoon, mutta verkkosivujen sijasta sovellusta kehitetään mobiilialustalle ja käyttöliittymän elementit renderöidään natiivi kirjastoilla (Bonnie Eisenman, 2019, s. 17-21).

React Native tukee tällä hetkellä mobiilisovellusten kehittämisen virallisesti Android- ja iOS-käyttöjärjestelmille (Facebook inc, Github). React Nativeen on yhteisön puolesta lisätty myös tuki kehittää Microsoftin UWP-applikaatioita (Windows Apps Team, 2016). React Native on alun perin julkaistu 2015 vain iOS ja myöhemmin samana vuonna Androidille (Daniel Witte & Philipp von Weitershausen,2015). Lisäksi 2016 lisättiin tuki Microsoftin UWP-alustalle (Windows Apps Team, 2016).

React Nativella on tehty monentyyppisiä ja kokoisia sovelluksia, sosiaalisen media palveluita, kauppa- sovelluksia, suoratoisto sovelluksia ja jopa pelejä. Se on myös tullut suosituksi monessa isossa yrityksessä mm. Facebook, Microsoft ja Tesla.

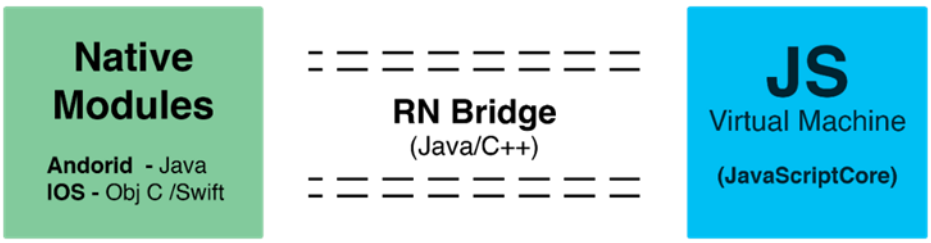
Kuva 1. Yrityksiä, jotka käyttävät React Native-sovelluskehystä. Kokolistaa voi tarkastella osoitteesta: http://facebook.github.io/react-native/showcase.html (Facebook, 2019)




## 2.3 Arkkitehtuuri

React Native sovellusarkkitehtuuri muodostuu kolmesta eri pääpiirteestä, natiivi moduulit, React Native silta(bridge) ja JavaScript Virtuaalikoneesta. Alustojen välillä arkkitehtuuri on sama, mutta luonnollisesti natiivi moduulit ovat alusta kohtaisia. (Rahul Gaba & Atul R)

Kuva 2. Rahul Gaba & Atul R tekemä luonnos React Nativen arkkitehtuurista.



### 2.3.1 Natiivi moduulit

Natiivi moduulit sisältävät alustakohtaista koodia kuten Java Androidille, Objective-c tai Swift iOS:lle. Natiivi koodin käyttäminen ei ole pakollista React Nativessa, mutta se saattaa lisätä suorituskykyä raskaiden toimintojen suorittamiseen, jotka vaativat usean säikeen käytön kuten kuvien prosessointi. (Facebook)

### 2.3.2 Silta (React Native Bridge)

React Native Bridge eli silta hoitaa kommunikoinnin molempiin suuntiin JavaScript virtuaalikoneen ja natiivikoodin välillä. Silta ottaa vastaan asynkronisesti JavaScriptillä tehtyjä pyyntöjä JSON muodossa ja välittää ne eteenpäin natiivi moduuliin prosessoitavaksi tai toisinpäin. (Marvin Frachet)

### 2.3.3 JavaScript Virtuaalikone

JavaScript virtuaalikoneessa suoritetaan kaikki koodit, jotka on kirjoitettu JavaScriptillä. JSVirtuualikone käyttää Applen kehittämää JavaScriptCore-moottoria, joka on alun perin kehitetty WebKittiä varten. Sama JavaScript-moottoria on käytössä Safari selaimessa, jonka takia sitä ei tarvitse erikseen pakata iOS sovellusten mukaan. Android sovelluksissa JavaScriptCore pakataan sovelluksen mukaan ja täten vie hiukan enemmän tilaa kuin iOS. (Rahul Gaba and Atul R)

## 2.4 Ohjelmointikielet

React Native perustuu Facebookin aikaisemmin kehitettyyn ReactJS JavaScript-kirjastoon, joka näkyy samanlaisuuksina syntaksissa ja käyttöliittymän toteutus tavassa. Näin ollen web-kehittäjien on helpompi aloittaa mobiilisovellusten kehittäminen pienemmällä kynnyksellä varsinkin, jos React on ennestään tuttu web-ohjelmoinnista. (Facebook, React Native, Github 2019)

React Native -sovellukset kirjoitetaan JavaScriptillä ja JSX-merkintäkielellä, joka muistuttaa XML ja JavaScriptin sekoitukselta. Suurin osa koodista, joka kirjoitetaan JSX tai JavaScriptillä voidaan jakaa eri alustojen välillä yhtenäisenä koodikantana. Varsinainen Business-logiikka on myös mahdollista jakaa web-sovellusten kanssa. JSX lisäksi on

mahdollistaa kirjoittaa sovelluksen osia, joita ei pysty toteuttamaan JavaScriptillä alustan natiivi ohjelmointikielellä esimerkiksi Java, Objective-c tai Swift, mutta tämä estää koodin uudelleenkäytön toisella alustalla. React Native käyttää hyväksi kohde alustan rajapintaa, joka mahdollistaa ohjelmalle natiivin ulkoasun ja paremman suorituskyvyn, kuin webview-hybridi sovellukset. React Nativen kanssa on myös mahdollista käyttää kolmannen osapuolien JavaScript kirjastoja, jotka eivät ole DOM riippuvaisia. (Bonnie Eisenman, 2019, s. 17-21)

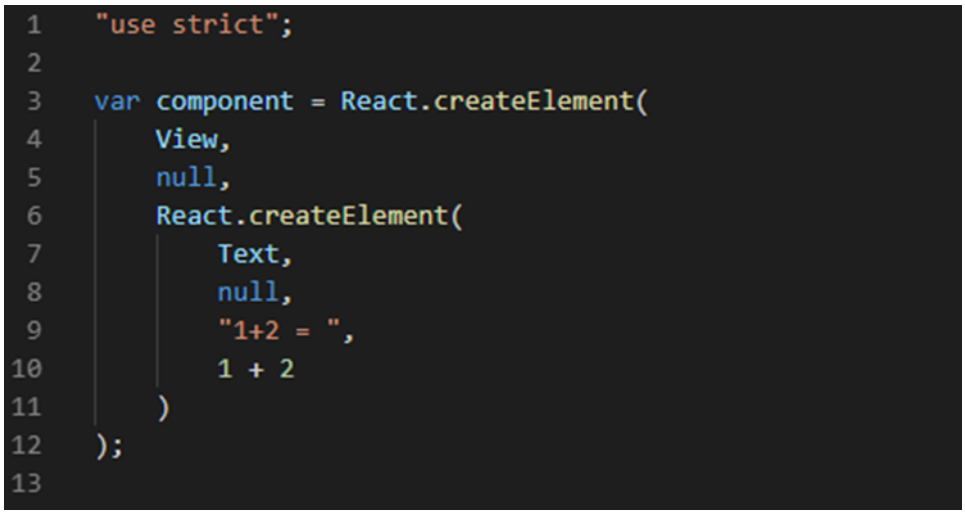
### 2.4.1 JSX

JSX on merkintäkieli, jota käytetään React ja React Native kanssa. Se muistuttaa syntaksilta XML-merkintäkieltä. JSX on tarkoitettu esiprosessorien käytettäväksi, joka muuntaa JSXn ECMAScript-standardiin eli JavaScriptiksi. React Native sovelluksen React elementit kuten teksti kirjoitetaan JSX syntaksilla, joka muistuttaa html ja JavaScriptin sekoitusta. JSX tarkoitus on yksinkertaistaa koodi rakennetta ja näin tehdä siitä helpommin tulkittava. React JSX, jsx-transform, Babel ja Bublé kykenevät muuntamaan JSX syntaksia ja näistä Babel tulee valmiiksi React Nativen mukana. (Facebook JSX, 2019)

Kuva 3. JSX-merkintäkielellä tehty komponentti, joka renderöi näytölle 1+2 = 3.

```
1   const component = <View><Text>1+2 = {1+2}</Text></View>
2
3
4
```

Kuva 4. Kuvan 1 JSX koodi JavaScript muodossa, joka on huomattavasti pidempi.



```
"use strict";

var component = React.createElement(
    View,
    null,
    React.createElement(
        Text,
        null,
        "1+2 = ",
        1 + 2
    )
);

```

### 2.4.2 Yhteenveto

Pähkinänkuoressa React Native sovelluksien kehittäminen toimii suurin piirtein samalla tavalla kuin kehitettäisiin web-sovellusta, mutta kohde alusta on älypuhelin eikä verkkosivu. Erona web-ohjelmointiin on, että React kirjastossa JSX käyttö on pakollista, joka määrää vain syntaksin toteuttaa graafisen käyttöliittymän ja DOM sijasta käyttöliittymä renderöi natiivi komponenteilla.

## 2.5 Komponentit (Component)

Yksi oleellisimmista React ja React Nativen ominaisuuksista on, että käyttöliittymä rakennetaan komponenteista. Komponentit muodostuvat joko luokista (class/function) tai funktioista (function), jotka ottavat vastaan attribuutteja(props) ja palauttavat React-elementtejä, jotka kuvaavat miltä käyttöliittymän pitäisi näyttää sovelluksessa. Jokaisella komponentilla on aina vähintään render() -funktio, joka palauttaa JSX syntaksilla kirjoitettua koodia. (Facebook)

Yksinkertaisimmillaan komponentti voi olla teksti, kuva tai painike. Suurimmillaan komponentti voi olla vaikka koko sovellus. Komponenttien tarkoitus on kumminkin auttaa

koodin uudelleen käytettävyyttä ja selkeyttää käyttöliittymän rakennetta pilkkomalla ne pienemmiksi osiksi.

Kuva 5. Esimerkki komponentti, joka tulostaa näytölle ”Hello React!”.

```
61
62  class Component_example extends Component{
63    render(){
64      return(
65        <View>
66          <Text>Hello React!</Text>
67        </View>
68      );
69    }
70  }
71
72  export default class App extends Component {
73    render() {
74      return (
75        <View style={styles.container}>
76          <Component_example/>
77        </View>
78      );
79    }
80  }
81
```

### 2.5.1 Attribuutit (Props)

Attribuutit ovat muuttujien tyylisiä asetuksia, johon voi tallentaa satunnaista tietoa esimerkiksi tyyli tiedosto, tekstiä, arvoja jne. Propsin tarkoitus on periyttää parametrillä annettava tieto seuraavalle komponentille. Propsia ei voi muokata periytymisen jälkeen. Se ikään kuin toimii komponentin alustus asetuksena. (Facebook)

Kuva 6. Kuvan koodissa annetaan Props_example komponentille parametriksi ”React!” ja se tulostaa näytölle ”Hello React!”.

### 2.5.2 Tila (State)

Komponentin sisällä oleva tila(state), joka toimii muuttujan lailla. Toisin kuin attribuutit(props) sitä voi muokata, kun tilan arvo muuttuu React komponentti päivitetään käyttöliittymässä. Esimerkiksi hakukenttään kirjoitetaan jotain ja haku ehdotuksia ilmestyy joka kirjaimen muutoksella. Komponentin tilaa voi muokata suoraan vain constructor() -metodin sisällä. Muutoin tilan vaihtaminen onnistuu vain setState() -funktiolla. (Facebook Inc, State)

Kuva 7. Esimerkki koodi, jossa tila(state) on alun perin ”React!”, mutta kun renderöityä tekstiä kosketaan, vaihtuu sen tila ”World!” ja täten tulostuu ruudulle Hello World!

```
class State_example extends Component{
  constructor(props){
    super(props);

    this.state = {
      s: 'React!'
    }
  }

  _changeState(){
    this.setState({
      s: 'World!'
    });
  }

  render(){
    return (
      <View>
        <Text onPress={() => this._changeState()}>Hello {this.state.s}</Text>
      </View>
    );
  }
}

export default class App extends Component {
  render() {
    return (
      <View style={styles.container}>
        <State_example/>
      </View>
    );
  }
}
```

# 3 REACT NATIVEN SOVELTUVUUS TIETOJENKÄSITTELYN MUUHUN OPETUSSISÄLTÖÖN

## 3.1 Mitä on opiskeltu aikaisemmin?

Hämeen Ammattikorkeakoulussa ohjelmointi on ollut pääasiassa olio-ohjelmointia C# tai Javalla. Ensimmäisenä vuonna tutustuttiin ohjelmoinnin perusteissa C# ja samalla kielellä olio-ohjelmoinnin perusteet. Myöhemmin käytiin läpi web-tekniikoita, johon sisältyi html, CSS, JavaScript ja PHP. JavaScriptin opiskelu rajoittui lähinnä vain JQuery-kirjastoon ja Ajaxin käyttöön. (Hämeen Ammattikorkeakoulu)

## 3.2 Opittavuus

React Native ohjelmointi painottuu suurimmalta osin JavaScriptiin ja JSX-merkintäkieleen ja se on selvästi suunniteltu web-kehittäjille laskemaan kynnystä web-sovelluksien ja mobiilisovelluksien välillä. Ensimmäiseksi React Native saattaa olla askarruttava näky, jos kokemusta ei ole hirveästi JavaScriptistä, JSX, ReactJS tai yleisesti web-ohjelmoinnista.

React Nativen rakenne on hyvin erilainen verrattuna perinteisiin C#- ja Java-ohjelmiin, mutta React Native ei ole kovin vaikeasti opittava ohjelmistokehys. React Native kumminkin vaatii jonkunlaista perusosaamista ohjelmoinnista esim. ohjelmoinnin- ja olio-ohjelmoinnin perusteet auttavat huomattavasti. Tärkeintä on ymmärtää miten komponentit toimivat ja niiden attribuutit(props) ja tilat(state), jotka liittyvät ReactJS-kirjastoon. Nämä asiat oppimalla voi jo alkaa rakentamaan ensimmäisiä sovelluksia ja oppimaan vähän vaativimpia asioita.

JSX on melko nopeasti opittavissa oleva merkintäkieli. Se muistuttaa hiukan JSP-tekniikkaa, jota on käytetty aikaisemmin Hämeen ammattikorkeakoulun tietojenkäsittely Ohjelmistokehityksen Www-sovelluspalvelut kurssilla. React Nativen aloitus kynnys on hiukan tavallista korkeampi, jos aikaisempaa kokemusta ei ole ReactJS tai

JavaScriptistä, mutta helpottuu huomattavasti, kun on ymmärtänyt perusasiat. Kun Reactin komponenttimalli ja JSX on opittu, tapahtuu ohjelmointi lähes samalla tavalla kuin missä tahansa muussa ohjelmointikehyksessä. JavaScriptin ennestään osaaminen tai jopa ReactJS helpottaa huomattavasti.

React Native sisältää myös mahdollisuuden kirjoittaa natiivikoodia mikä ei välttämättä ole pakollista kaikentyyppisille sovelluksille. Mikä saattaa lisätä lisä vaikeutta. Opinnäytetyön käytännönosuudessa pilvipalvelu sovelluksessa kirjoitettiin 2 riviä Java-koodia, joka oli käytännössä vain Firebase kirjaston tuominen natiivi tasolle.

Oppimateriaalin löytäminen React Nativelle ei ole ongelmallista. Suurin osa dokumentaatiosta on Facebookin itse tuottamaa, mutta yhteisönpuolesta löytyy paljon artikkeleiden, videoiden ja kirjojen muodossa. Koska React Native on tällä hetkellä vielä kovan kehityksen alla, näkyy se myös valitettavasti etenkin kirjoissa, joiden materiaali on ollut aika vanhentunutta vaikkei kirja ole ollut edes kovin vanha.

### 3.3 Kehitysympäristön asennus

React Native kehitysympäristön asentaminen on suhteellisen helppo, jos seuraa React Nativen virallisen asennus dokumentaation ohjeita. Asennus vaatii muutaman erillisen ohjelman asennuksen ennen, kun voi asentaa itse React Nativen ja muutaman ympäristömuuttujan asettamisen Windows 10 tietokoneelle.

Android Studion Emulaattori vaatii muutaman erillisen säädön Windows-käyttöjärjestelmälle kuten Intel HAXM asennuksen tai Hyper-V:n käyttöönoton. Tämä prosessi on hiukan erilainen AMD ja Intel prosessoreille. Android studio tarjoaa hiukan helpomman tavan luoda ja hallita virtuaalilaitetta, mutta ehdottomasti hitain ohjelma on Android Studio ja etenkin sen käynnistäminen. AVD on myös mahdollista asentaa komentoriville, joka saattaa nopeuttaa projektien aloitusta. Suurin osa opinnäytetyön käytännönosuuden ongelmista oli emulaattorin kanssa, mutta on helposti korjattavissa Android Studion SDK-asetuksista. Esimerkiksi emulaattori jäätyy käynnistyksessä tai Javan versio on väärä, jotka ovat korjattavissa emulaattorin asetuksista.

Komentorivin käyttö React Native -kehitysympäristössä on pakollista. React Native vaatii jokaisen projektin aloitukseen yksinkertaisen komennon ”react-native init [projektin nimi]” ja ”react-native run-android” tai iOS vastaavan. React Nativen kirjastoja ja paketteja voi asentaa komentoriviltä käyttämällä NPM- tai Yarn-paketinhallintatyökaluja. Jos komentorivi on ennestään tuttu, toimii projektinhallinta helposti komentoriviltä, mutta vaikkei olisikaan ei React Nativessa ole montaa komentoa mitä tarvitsee muistaa. Kontrastina tähän Xamarin tarjoaa graafisenkäyttöliittymän NuGet-pakettien ja projektinhallintaan.

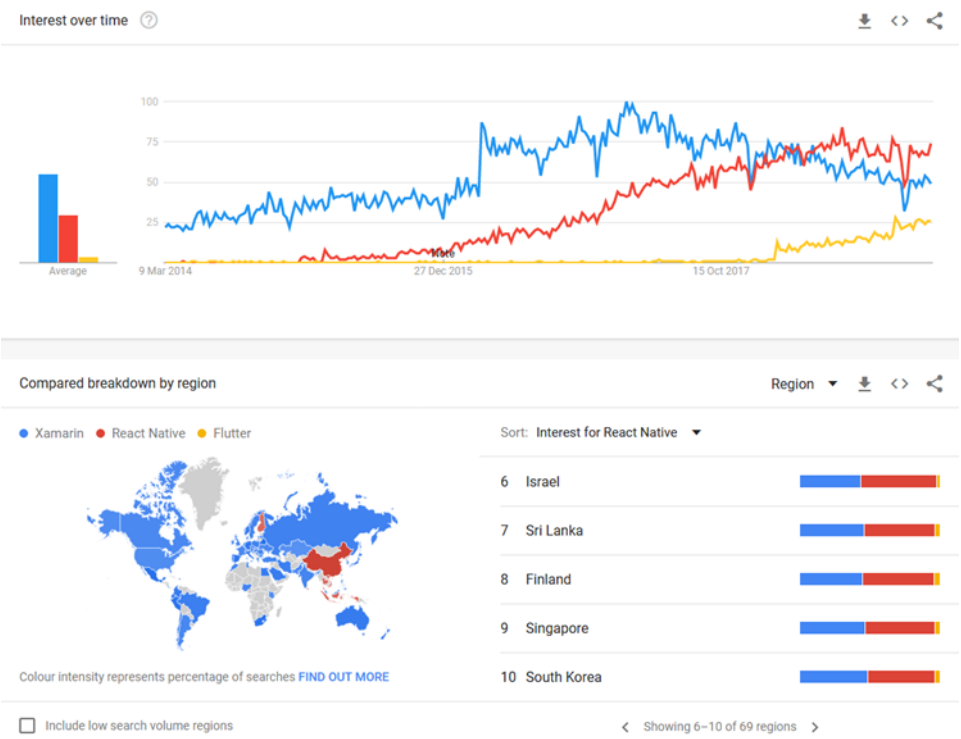
React Nativessa IDE ei ole lukittu ohjelmistokehykseen eli on mahdollista valita itselle sopiva tekstieditori tai IDE. Tässä opinnäytetyössä käytettiin Microsoft Visual Studio Code koodin kirjoittamiseen, joka soveltui siihen oikein mainiosti. React Native -kehitysympäristö ei ole myös lukittu Windows- ja MacOS-käyttöjärjestelmiin, joka mahdollistaa sen käytön muilla alustoilla kuten Linux.

### 3.4 Soveltuvuus muun opetuksen kanssa

React Nativen -kehitysympäristö on hyvin erilainen verrattuna mitä Hämeen Ammattikorkeakoulun Tietojenkäsittely koulutuksen muut ohjelmointiin liittyvät kurssit tarjoavat lukuun ottamatta toisen lukukauden Web-ohjelmointi moduuli. Vaikka React Nativen tapa toteuttaa käyttöliittymä ja ohjelmointi kieli on täysin erilainen kuin Java- ja C# -kielet ja tekniikat on tällä hetkellä huomattavasti ajankohtaisempi ohjelmointikehys kuin Xamarin.

Kuvan 8 Google Trends taulukon perusteella voimme päätellä, että React Native on huomattavasti suositumpi tai ainakin haetumpi kuin Xamarin Googlen hakupalvelussa. Xamariniin liittyvien hakujen suosi on selvästi laskenut vuoden 2017 jälkeen. (Google Trends)

Kuva 8. Xamarin, React Native ja Flutter hakusanojen suosio 5 vuoden sisällä Google Trends palvelussa. React Native on kerännyt suosiota hakujen määrässä toisin kuin Xamarin aiheiset hakusanat vähentynyt. Suomi sijoittuu tällä hetkellä 8. sijalle React Native liittyvien hakujen suosiossa.




Opittavuudelta aloituskynnys on hiukan suurempi React Nativessa verrattuna Xamariniin etenkin, jos siirtyy C# tai Java-ohjelmoinnista, mutta React Native ei vaadi JavaScriptin täydellistä osaamista, joten sen syntaksin voi oppia helposti samalla kun rakentelee ensimmäisiä React Native sovelluksia. Siten ohjelmointikielen vaihtaminen ei pitäisi aiheuttaa ongelmia. C# ja Javan opiskelusta ei myös ole ajanhukkaa, koska React Native sovelluksissa saattaa joutua kirjoittamaan natiivi koodia UWP tai Android alustalle.

React Nativella ei ole estettä toimia Hämeen Ammattikorkeakoulun Mobile Programming moduulin Cross-platform Development kurssin opetuksessa. Se soveltuu samankaltaisiin tehtäviin, kun aikaisemmin käytetty Xamarin-ohjelmistokehys. Sillä voi toteuttaa mobiilisovelluksia usealle alustalle samanaikaisesti ja sen voi integroida pilvipalvelun kanssa. React Nativen opiskelu ei ole sen haastavampi kuin Xamarinin oppiminen ja tämän lisäksi on mahdollista asentaa useammalle käyttöjärjestelmälle.

# 4 KÄYTÄNNÖNOSUUDEN TAVOITTEET

## 4.1 Alustavaa tietoa

Hämeen Ammattikorkeakoulun Mobile Programin moduulin on alun perin toteutettiin Xamarin-ohjelmistokehyksellä. Moduulissa on käyty läpi ohjelmistoympäristön asennus omalle tietokoneelle, graafisenkäyttöliittymän toteutus, usean sivun sovellus, usean sivun sovellukset, sivujen välinen tiedonsiirto, tiedon paikallinen tallennus ja tiedon siirto verkkopalvelujen välillä.

## 4.2 Käytännönosuuden tavoitteet

Opinnäytetyön käytännönosuuden tavoitteena on saada vastaus opinnäytetyön ensimmäiseen tutkimuskysymykseen: Miten voisi toteuttaa Hämeen Ammattikorkeakoulun Mobile Programming Cross-platform Development osuuden Xamarin sisällöt React Nativella. Ensimmäisessä osiossa käydään läpi kehitysympäristön asennus Windows 10 -käyttöjärjestelmälle, sen tarvittavat ohjelmat ja projektin aloitus.

Käytännönosuuden ensimmäisessä sovelluksessa on tarkoitus tutustua React Nativen tapaan luoda käyttöliittymä ja miten se onnistuu React Nativella. Ensimmäinen sovellus on yksinkertainen laskin, jossa käytetään apuna komponentteja (component), ominaisuuksia (props), tiloja (state) ja Flexbox-tyyliä luoda käyttöliittymä.

Toisen osuuden tarkoitus on tutustua rajapintaan ja miten tehdä pyynnön Rest-rajapintaan. Tavoitteena on rakentaa yksinkertainen sääsovellus, joka hakee päivän sään REST-rajapinnasta ja muistaa edellisen haetun kohteen. Käyttäjä syöttää halutun kaupungin nimen kenttään ja sovellus näyttää tämänhetkisen lämpötilan ja sään. Sovellus käyttää opeanweathermap.org tarjoamaa ilmaista rajapintaa.

Kolmannen ja viimeisen sovelluksen tarkoituksena on tutustua miten voi liittää pilvipalvelun React Native-sovellukseen. Tavoitteena on selvittää miten hakea ja lähettää dataa pilvipalveluun. Pilvipalveluna toimii Google Firebase.

# 5 REACT NATIVE KEHITYSYMPÄRISTÖNÄ

## 5.1 Kehitysympäristön asennus

Koska React Native on vielä nuori ja ripeässä kehityksessä on suositeltavaa vilkaista kumminkin ensin React Nativen virallinen asennus dokumentaatio (https://facebook.github.io/react-native/docs/getting-started). Tässä kohdassa käydään kumminkin läpi tämänhetkinen React Native -kehitysympäristön (version 58, 2019 Q1) asennus Windows Pro 10 N -käyttöjärjestelmälle ja ensimmäisen projektin aloitus.

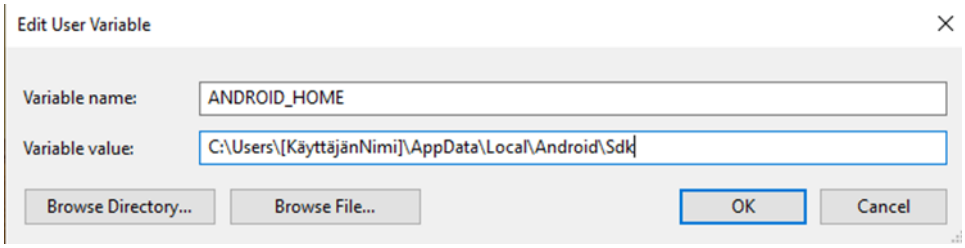
Taulukko 1. Laitteiston speksit, johon asennettiin React Native -kehitysympäristö.

| | |
|---|---|
| Käyttöjärjestelmä | Windows 10 Pro N |
| Prosessori | AMD Ryzen 1700 |
| Näytönohjain | Radeon RX 580 8GB |
| Keskusmuisti | 16GB |

### 5.1.1 Riippuvuudet

Windows 10 käyttöympäristössä React Native vaatii NodeJS, vähintään Python2, Java Development Kit 8 ja Android Studion tai Android SDK komentorivi sovelluksen, mutta Android studio on huomattavasti helpompi asentaa Windows-käyttöjärjestelmälle. Android studion Android emulaattorin asennus saattaa osittain epäonnistua ensimmäisellä kerralla, jos Hyper-V tai Windows Hypervisor Platform ei ole kytketty päälle. Android studiota tarvitaan lähinnä vain emulaattorin takia. Mikäli haluaa vain testata ohjelmia Android älypuhelimella voi jättää emulaattorin asennuksen kokonaan pois. React Native vaatii myös Windows 10 järjestelmällä asettaa muutaman ympäristömuuttujan. Ympäristömuuttujiin pääsee käsiksi painamalla win+Pause tai avaamalla ohjauspaneeli (**Control panel**)>Järjestelmä(**system**))> Järjestelmän lisäasetukset (**Advanced system settings**)>Ympäristömuuttujat (**Enviorement Variables**).

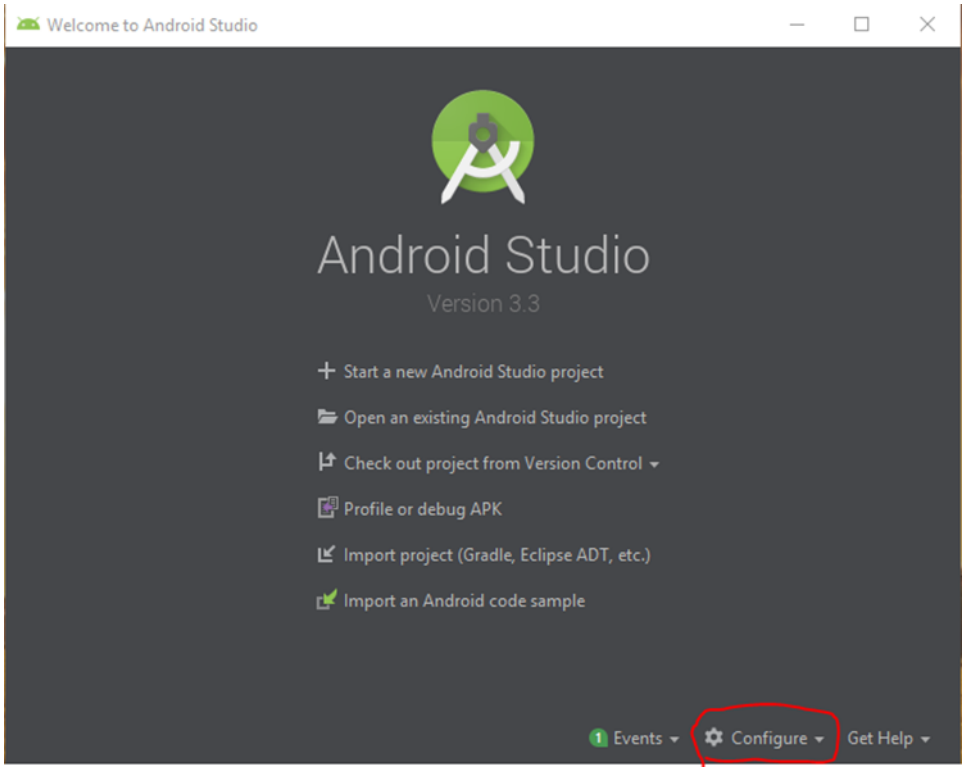
Kuva 9. Windows 10 ympäristömuuttujan (Enviorement Variable) lisäys lomake.




### 5.1.2 Emulaattori

Emulaattori asennettiin tässä opinnäyteyössä Android Studion kautta. Käyttämällä AVD hallintatyökalua (AVD manager), jonka löytää Android Studion käynnistysvalikon määrityksien alta. Tässä opinnäytetyössä käytettiin virtuaalilaitteen profiilina Pixel puhelinta ja käyttöjärjestelmänä Android Pie 9.0, jos 9.0-käyttöjärjestelmä ei ole käytettävissä. Sen voi ladata SDK hallintatyökalulla, jonka löytää samasta valikosta AVD hallintatyökalun kanssa. Ohjelmistokehityspaketti (SDK) asennukset löytyvät: Appearance & Behavior>System Settings>Android SDK välilehdeltä.

Kuva 10. Android studion aloitusvalikko(launcher), jossa näkyy AVD- ja SDK-hallinnan asetusvalikko oikeassa alakulmassa.




### 5.1.3 React Native

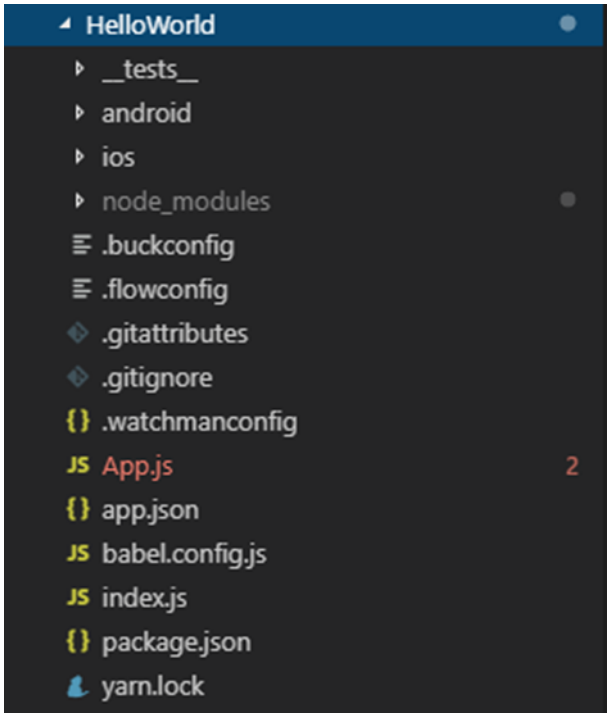
Kun tarvittavat riippuvuudet on asennettu. Voidaan asentaa itse React Native. Facebookin virallisten ohjeiden mukaan on suositeltavaa asentaa react-native-cli käyttämällä npm paketinhallintaa, joka tuli NodeJS asennuksen mukana. React Nativen asennus npm paketinhallinnalla onnistuu avaamalla CMD tai PowerShell komentorivin ja käyttämällä komentoa: **npm install -g react-native-cli**. Komento asentaa React Native komentorivi sovelluksen globaalisti, joka mahdollistaa sen käytön useammassa projektissa ja komentorivillä. React Nativen asennuksen jälkeen on myös mahdollista asentaa Yarn paketinhallintajärjestelmä, joka nopeuttaa React Native projektin luontia, mutta ei ole pakollinen.

### 5.1.4 Projektin aloitus

Kehitysympäristön testaaminen hoidetaan perinteisellä Hello World! sovelluksella, jonka tarkoitus on vain näyttää ohjelmassa teksti Hello World! Kehitys ympäristön tai tekstieditorin voi valita vapaasti oman mieltymyksen mukaan, mutta tässä opinnäytetyössä käytettiin Microsoft VScode ohjelmaa ja sen React Native Tools lisäosaa. Uuden projektin voidaan aloittaa avaamalla komentorivi haluttuun paikkaan/kansioon ja kirjoittamalla **react-native init [projektin nimi],** joka tuottaa kyseiseen kansioon uuden kansion projektin nimellä.

Ensimmäisellä kerralla saattaa kestää hiukan tavallista kauemmin projektin kasaamisessa, mutta lopputuloksena pitäisi olla toimiva malliprojekti.
Jos projektin rakennuksen aikana tulee varoituksia, että Yarn paketinhallinta ei ole käytössä kestää projektin rakentamisessa hiukan kauemmin. Yarn paketinhallinnan voi asentaa osoitteesta https://yarnpkg.com/en/ tai **npm install -g yarn** komennolla.

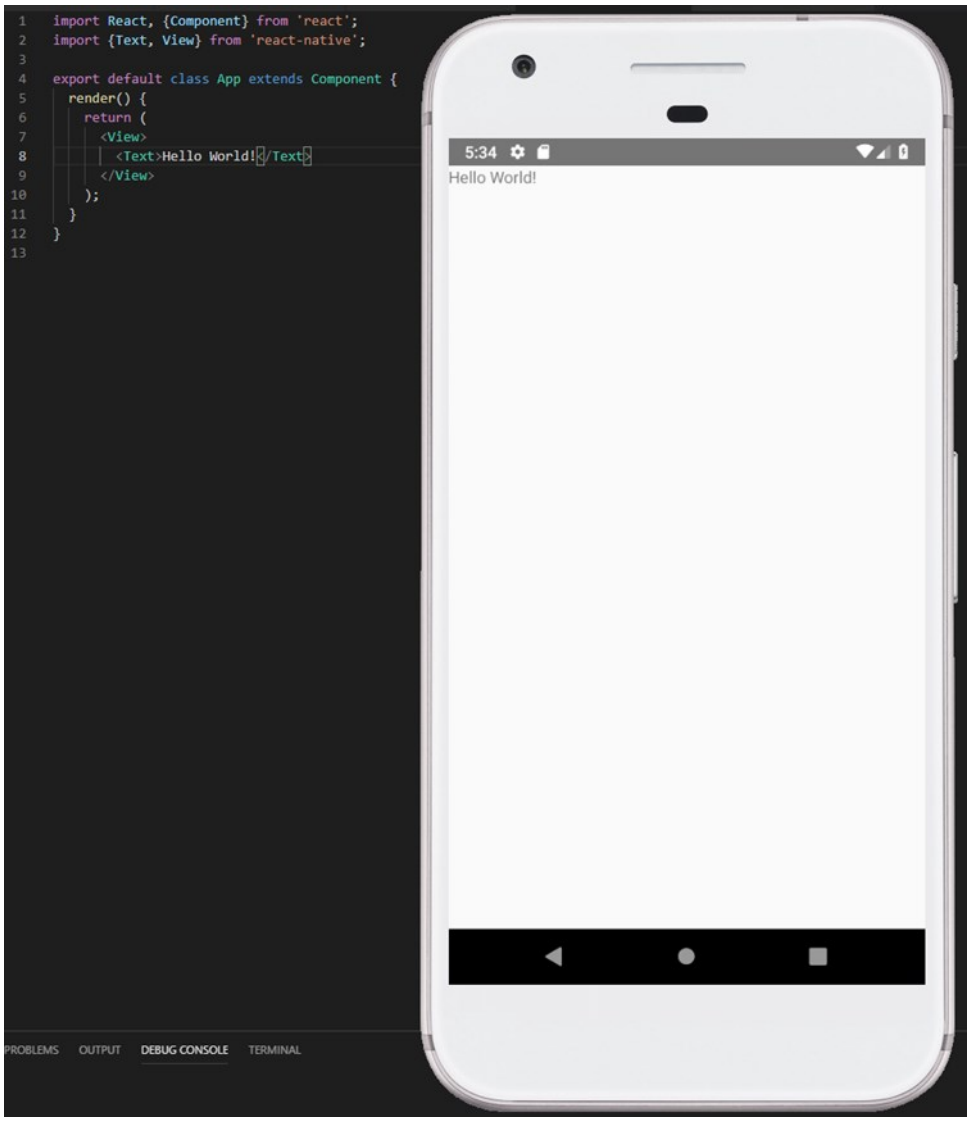
Kuva 11. **react-native init HelloWorld** komennosta syntyvä projektikansion oletus rakenne.



Oletuksena sovellus alkaa index.js tiedostosta, jossa vain määritellään nimi ja koodin aloituspiste, joka on oletuksena App.js. Varsinainen koodi siis alkaa App.js tiedostosta, jossa on valmiina yksinkertainen esimerkki koodi. Alustakohtaiset asetukset ja koodit löytyvät luonnollisesti Android ja iOS kansioista. Sovelluksen nimi löytyy oletuksena app.json -tiedostosta ja sovelluksen riippuvuudet löytyvät package.json -tiedostosta.

React Native sovelluksen voi asentaa Android emulaattorille tai laitteelle komentorivillä siirtymällä projekti kansioon ja syöttämällä komento ”**react-native run-android”.** Ennen tätä emulaattori on oltava päällä tai haluttu Android puhelin kytketty USB:llä. On myös hyvä huomioida, että tämän lisäksi Android älypuhelimesta on aktivoitava kehittäjätila ja USB-vianetsintä (USB debugging), jotta sovelluksen testaaminen onnistuisi.

Kuva 12. Yksinkertainen esimerkki koodi, joka tuottaa vain ”Hello World!” tekstin näytölle ilman minkäänlaisia tyylejä selkeyden vuoksi.




## 5.2 Laskinsovellus

Koska opinnäytetyön tekijällä ei ollut aikaisempaa kokemusta React Nativen käytöstä, oli luonnollista valita ensimmäiseksi sovellukseksi jokin yksinkertainen. Tähän hommaan soveltui hyvin laskin. Laskin sovelluksessa tulee helposti näkyviin React Nativen komponentti pohjainen rakenne ja yksinkertaisen käyttöliittymän luominen toimintoineen.

Jokainen React Native-sovellus alkaa juurikomponentista. Jokainen komponentti on eritelty erillisiin JavaScript tiedostoisiin komponentin omalla nimellä, joka helpottaa koodin luettavuutta, kun jokaisen komponentin omakohtainen koodi on erillisissä tiedostoissa.

Tässä sovelluksessa ylin komponentti on App, joka on jokaisen projektin oletus aloitus komponentti. App-komponentin alle tuli Calculator- ja CalButton-komponentit. Calculator-komponentti sisältää laskimen näytön ja painikkeet, jotka muodostuvat CalButton-komponentista.

### 5.2.1 Käyttöliittymä

Käyttöliittymän rakentaminen React Native sovelluksissa on helppoa, kun muutoksen näkee heti ilman, että koodia joutuu kääntämään erikseen pienien päivityksien jälkeen ja tämä säästää huomattavasti kehitysaikaa. Suurin osa käyttöliittymään liittyvästä koodista, joka on JavaScriptiä, joka ei vaadi erillistä koodin kääntämistä vaan riittää, että painaa Android emulaattorista oletuksena ’RR’ tai ctrl+M ja ponnahdusikkunasta refresh niin muutos näkyy jo näytöllä.

React Nativen käyttöliittymän muodostuu JSX:llä tehdystä koodista ja StyleSheet-objekteista. JSX:llä tehty käyttöliittymä muodostaa HTML:llä muistuttavan rakenteen, jonka sisälle rakennetaan sisäkkäin komponentti elementtejä ja JavaScript-koodia.

Kuva 13. Laskinsovelluksen render() -funktion sisältämä JSX-koodi, jossa käytettään styles nimistä Stylesheet objektia määrittelemään komponentin ulkonäkö.

```
22    render(){
23      return(
24        <View style={styles.container}>
25
26          <View elevation={1} style={styles.display}>
27            <Text style={styles.displayText}>{this.state.inputValue} </Text>
28          </View>
29
30          <View style={styles.numpad}>
31            {this._rederButtons()}
32          </View>
33
34        </View>
35      );
36    }
```

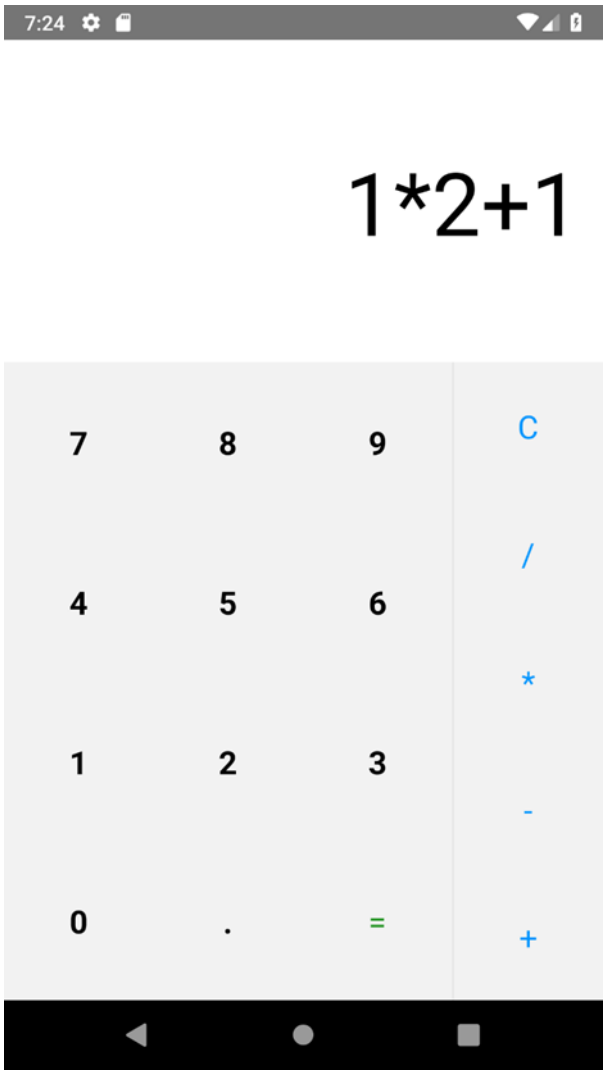
StyleSheet-objektin syntaksi muistuttaa CSS:n tyyliä määrittää käyttöliittymän ulkoasu. On myös mahdollista kirjoittaa komponenttien ulkonäköä kuvaavat tyylit suoraan JSX-koodiin, mutta tyylien erottelu Stylesheet-objekteihin mahdollistaa niiden uudelleen käytettävyyden ja helpottaa koodin lukemista.

Kuva 14. Laskinsovelluksessa käytettävä StyleSheet objekti.

```
import { StyleSheet } from "react-native";

export default StyleSheet.create({
    container:{
      flex: 1,
      backgroundColor: 'white',
    },
    display: {
      flex: 1,
      alignItems: 'flex-end',
      justifyContent: 'center',
      margin: 1,

    },
    displayText: {
      fontSize: 60,
      color: 'black',
      fontWeight: 'bold',

    },
    numpad: {
      flex: 2,
      flexDirection: 'row',
      backgroundColor: 'rgba(0,0,0,0.05)',
    },

```

Laskimen, painikkeiden ja näytön paikka sovelluksessa määriteltiin flex nimisellä attribuutilla, joka toimii lähes samalla tavalla, kun CSS:n Flexbox layout. Flexbox antaa sovellukselle laatikkomaisen rakenteen. Flexboxin hyöty on se, että käyttöliittymä muokkaantuu laitteen näytön koon mukaan. Tämä mahdollistaa sen, että käyttöliittymä näyttää mahdollisimman samalta eri laitteella näytön koosta riippumatta. Esimerkkinä laskinsovelluksessa juurikomponentin arvo on flex: 1, joka peittää koko sovelluksen näytön. Juurikomponentin alla on kaksi komponenttia display(flex:1) ja numpad(flex:2). Tässä tapauksessa display vie ruudusta 1/3, kun taas numpad vie 2/3.

Kuva 15. Laskinsovelluksen lopullinen ulkoasu.




### 5.2.2 Logiikka

Laskimen logiikka oli hyvin yksinkertainen. Sovelluksen käynnistyttyä luotiin ensin laskin ja näyttö. Tämän jälkeen renderöitiin painikkeet.

Painikkeiden arvot muodostuivat valmiiksi määritellystä taulukosta. Painikkeet ottivat vastaan ominaisuudeksi(props) arvon, joka tuli painikkeen tekstiksi ja funktion, joka määritteli mitä tapahtuu, kun painiketta kosketaan. Painikkeiden luontihetkellä tarkistettiin minkä tyyppinen arvo painikkeelle annetaan ja sen perusteella päätetään ulkonäkö.

Kuva 16. Yksinkertainen funktio, joka rakentaa laskinsovelluksen painikkeet.

```
38    _renderButtons(){
39
40      let columns = [];
41      for(let c = 0; c < numpadButtons.length; c++){
42
43        let column = numpadButtons[c];
44        let ButtonColumn = [];
45        for(let i = 0; i < column.length; i ++){
46
47          let input = column[i];
48          ButtonColumn.push(<CalButton value={input}
49            onPress={ () => {this._buttonPressed(input)}} key={i} />);
50        }
51        columns.push(<View style={this._IsLastColumn(c)}
52          key={c}>{ButtonColumn}</View>)
53      }
54      return columns;
55    }
```

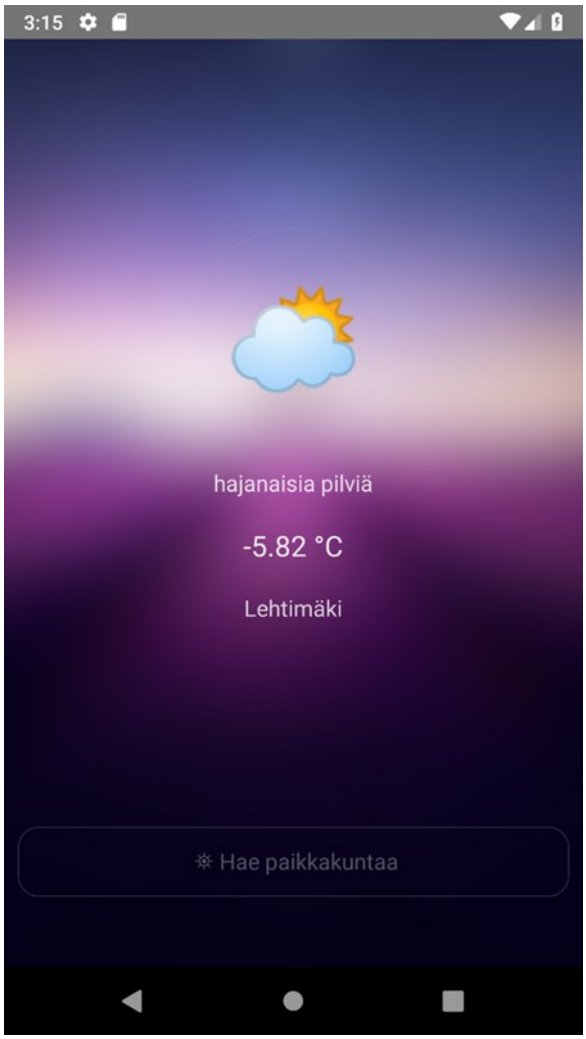
Jokaisen napin onPress tapahtumassa eli kosketus kutsutaan funktiota _buttonPressed, joka tarkistaa painikkeen arvon ja mitä siitä tapahtuu. Esimerkiksi painetaan nappuloita 1*2+1 niin Calculator-komponentin inputValue tila(state) päivitetään, niin päivittyy laskimen näytön teksti vastaavaksi.

## 5.3 Sääsovellus

Opinnäytetyön toiseksi sovellukseksi päätettiin tehdä sääsovellus, jonka tarkoituksena oli tutustua miten React Nativella otetaan yhteyttä REST tyyppiseen rajapintaan ja tallentamiseen. Tämän projektin rajapinta palveluksi valittiin OpenWeatherMapin. Palvelu tarjoaa ilmaisena tämänhetkisen sään ja 5 päivän sää ennustuksen. Ennen käyttöä palvelu vaatii kirjautumisen ja API-avaimen, jotta sen käyttö onnistuisi.

Sovelluksesta tehtiin erittäin yksinkertainen. Käytännössä sovelluksen käynnistettyä sovellus lataa edellisen haetun kaupungin nimen muistista ja tekee nimellä pyynnön OpenWeatherMapin rajapintaan, jos pyyntö on onnistunut, tehdään Forecast komponentti. Forecast komponentti näyttää tämänhetkisen sään lämpötilan ja säähän liittyvän kuvan. Kuva vaihtuu rajapinnasta tulevan datan merkityksen mukaan. Lopuksi lisättiin lyhyt animaatio, kun Forecast komponentti päivittyy.

Kuva 17. Sääsovelluksen lopputulos.




### 5.3.1 Rajapinta

React Nativessa pyynnön lähettäminen REST-rajapintaan ei onneksi ole vaikeata. Pyynnön voi tehdä fetch- tai XMLHtppRequest -rajapintaa käyttäen. Tässä sovelluksessa käytettiin fetch API. Fetch pyynnön voi tehdä lupausketjulla. Lupausketju on käytännössä sarja funktioita toimivat annetussa järjestyksessä. Kuvan 18 esimerkki koodista voi tarkastella lupausketjua, joka alkaa fetch() pyynnöstä, kun pyyntö on valmis, suoritetaan seuraava .then. Vain virhetilanteessa suoritetaan .catch pyyntö. Vaihtoehtoisesti Fetch pyynnön voi tehdä myös ES2017 standardissa lisätyllä async/await syntaksilla, jota React Native myös tukee.

Kuva 18. Fetch pyyntö rajapintaan. Jos päivitys on onnistunut, päivitetään forecast tila(state).

```
30
31      _fetchDataFromApi = city =>{
32        fetch( uri + city + apiKey + param)
33       .then((response) => response.json())
34       .then((responseJson) => {
35         this.setState({
36           forecast: {
37               main: responseJson.weather[0].main,
38               description: responseJson.weather[0].description,
39               temp: responseJson.main.temp,
40               city: city
41           }
42         });
43       })
44       .catch(() => {
45         this.setState({
46             forecast: null
47         });
48       });
49      }
50
```

### 5.3.2 Tallentaminen ja tallennuksen lataaminen

Projektissa päätettiin käyttää React Nativen mukana tulevaa AsyncStorage-tekniikkaa. AsyncStorage tallentaa laitteelle ilman salausta, joten salasanojen tallentamiseen se ei ole sopiva. Tallennukset poistuvat myös, jos sovelluksen poistaa laitteelta. Sääsovelluksessa tallennettiin vain yksi merkkijono, joka sisälsi kaupungin nimen. Merkkijono ladataan uudelleen sovelluksen käynnistyksen yhteydessä laitteelta. Tarkemmin AsyncStorage-latausfunktiota kutsuttiin componenDidMount() -metodin sisällä. ComponentDidMount() -metodi on React-komponenttien oletus funktio, jota kutsutaan, kun komponentti on renderöity kerran. Se soveltuu datan lataamiseen käynnistyksen ohella.

Kuva 19. Koodi esimerkki miten voi yksinkertaisimmillaan tallentaa AsyncStorage:lla. Koodi tallentaa kaupungin nimen ’city’ avainsanalla.

```
52    _saveDataLocal = async () => {
53      try {
54        await AsyncStorage.setItem('city', JSON.stringify(this.state.city));
55      }catch (error){
56        alert(error);
57      }
58    }
```

Kuva 20. Koodi esimerkki miten palautetaan muistista AsyncStorage:lla tallennettu kaupungin nimi.

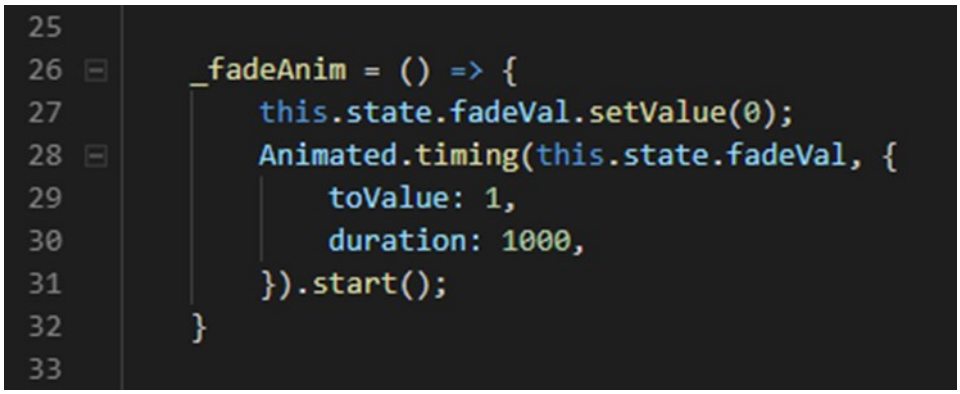
```
59
60      _retriveDataLocalAndFetch = async () => {
61          AsyncStorage.getItem('city')
62          .then((json) => JSON.parse(json))
63          .then((city) => {
64            this.setState({
65              city: city
66            });
67          })
68          .then(()=> this._fetchDataFromApi(this.state.city))
69          .catch((error) => alert(error))
70      };
71
```

### 5.3.3 Animaatiot

React Nativen on mukana tuleva animaatio kirjasto nimeltä Animated. Kirjasto sisältää perustyökalut animaatioiden tekemiseen. Sillä on mahdollista tehdä nopeasti yksinkertaisia animaatioita ja isommalla ajan panostuksella vähän näyttävimpiäkin. Animated-kirjastolla voi vaikuttaa komponenttien liikkeeseen ja ulkonäköön. Animaatiot oletus komponenteissa voi ottaa käyttöön lisäämällä Animated-komponentin eteen esimerkiksi <View> komponentista tulee <Animated.View>. Animated-kirjasto ei välttämättä toimi kaikissa oletus komponenteissa, mutta on mahdollista tehdä oma animoitu komponentti, joka kiertää tämän. Animated-kirjaston lisäksi React Nativella on useita kolmannen osapuolen animaatiokirjastoja helposti saatavilla ja vapaasti käytettävissä kuten react-native-pose, react-native-animatable jne.

Sääsovelluksessa käytettiin yksinkertaista häivytys animaatiota sää tietojen näyttämisessä. Animaatio käynnistyy, kun komponentti luodaan tai sen tietoja päivitetään.

Kuva 21. Esimerkki koodi yksinkertaisista animaatioista. Kuvan koodi muuttaa läpinäkyvän komponentin näkyväksi 1 sekunnin sisällä.

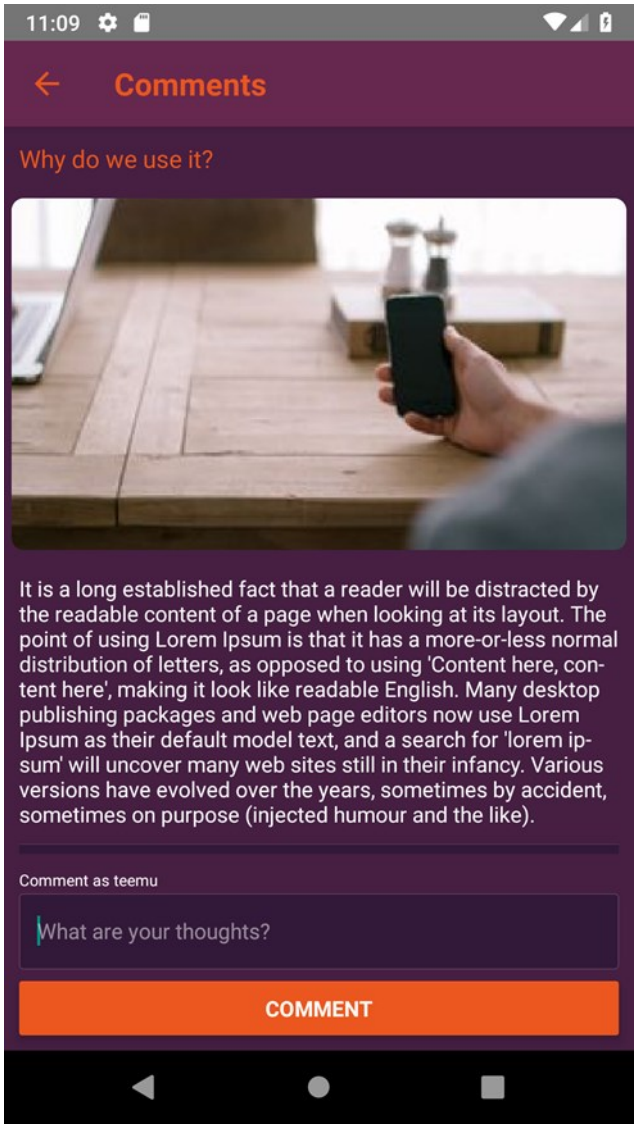


## 5.4 Pilvisovellus

Opinnäytetyön käytännönosuuden viimeisen sovelluksen tarkoitus on selvittää miten voi liittää pilvipalvelun React Native -mobiilisovellukseen. Tässä sovelluksessa käytettiin hyväksi Google Firebasen tarjoamaa autentikointipalvelua ja Firestore-tietokantaa. Firebase pilvipalvelu ratkaisuun päädyttiin, koska se oli helposti liitettävissä React Native -projekteihin, hyvin dokumentoitu ja soveltui hyvin pieniin mobiilisovelluksiin tai isoihin sovelluksiin.

Sovellukseksi tehtiin yksinkertainen sovellus, johon käyttäjä lähettää viestin ja muut käyttäjät voivat kommentoida viestiä. Sovellukseen täytyy ensin kirjautua sähköpostilla ja salasanalla ennen, kun pääsee kirjoittamaan tai lukemaan viestejä.

Kuva 22. Kuvankaappaus sovelluksen kommentti näkymästä. Yläosassa näkyy artikkeli ja tämän alla kommentit. Käyttäjä on kirjautunut Teemu nimisellä käyttäjällä.




### 5.4.1 Firebasen

Firebase ei aluksi tue React Native-sovelluksia, mutta tähän ongelmaan on valmis ratkaisu React Native Firebase-kirjasto. Kirjasto sisältää tuen tässä sovelluksessa käytettävälle Cloud Firestore -tietokannalle, autentikointipalvelulle ja monelle muulle Firebase-moduulille. Kirjaston muita tuettuja moduuleja voi tarkastella virallisesta dokumentaatiosta (https://rnfirebase.io/).

Kirjaston voi asentaa manuaalisesti projektiin komennolla: **npm install --save react-native-firebase** tai **yarn add react-native-firebase.** Asennuksen jälkeen linkitetään asennettu Firebase-moduuli projektiin komennolla: **react-native link reac-native-firebase.** Tämän jälkeen on suositeltavaa tarkistaa, että moduulin generoitu osoite ei ole virheellinen. Linkkaus komento saattaa välillä generoida moduulin osoitteen kenoviivoilla vinoviivojen sijasta. Tämä aiheuttaa virheen sovelluksen kääntö vaiheessa. Generoidut osoitteet löytyvät tiedostosta **/android/settings.gradle**.

Kuva 23. **/android/settings.gradle**:n asetukset. Ylempänä oikein merkitty moduulin osoite. Alempana komennon **react-native link** generoitu moduulin osoite, joka on väärin.

```
//Oikein merkitty osoite
include ':react-native-firebase'
project(':react-native-firebase').projectDir =
new File(rootProject.projectDir, '../node_modules/react-native-firebase/android')

//Väärin merkitty osoite
include ':react-native-firebase'
project(':react-native-firebase').projectDir =
new File(rootProject.projectDir, '..\node_modules\react-native-firebase\android')
```

Manuaalinen asennus vaatii Firebase moduulien kuten tässä sovelluksessa käytettyjen Firestore ja autentikointipalvelu lisäämisen erikseen tiedostoisiin **/android/build.gradle**, **/android/app/build.gradle** ja **MainApplication.java**. Ajantasaiset ohjeet jokaisen käytettävissä olevien Firebase moduulien lisäämiseen projektiin löytyy virallisesta dokumentaatiosta (https://rnfirebase.io/).

Toinen vaihtoehto ja huomattavasti nopeampi tapa asentaa **react-native-firebase** -kirjasto on kloonata Firebase aloitusprojekti Githubista. Aloitus projekti on sama kuin **react-native init** komennolla muodostuva aloitusprojekti. Lisänä on vain, että react-native-firebase on valmiiksi asennettu ja kaikki moduulien riippuvuudet määritetty valmiiksi. Aloitusprojektin voi lataa osoitteesta: https://github.com/invertase/react-native-firebase-starter.

Projektin luominen Firebaseen on helppoa ja sen käyttö on ilmaista. Projektin luominen vaatii vain nimen ja vaihtoehtoisesti palvelimen sijainnin. Tämän jälkeen voi liittää React Native-sovelluksen Firebaseen. Jotta Firebase toimisi sovelluksessa se pitää rekisteröidä sovelluksen oikealla nimellä. React Native -projektin nimen löytää juurikansiosta **package.json** nimisestä tiedostosta. Android-sovellus kohtaisen nimen löytää **android/app/src/AndroidManifest.xml.** Tämän jälkeen ladataan **google-services.json**-tiedosto**,** joka siirretään projektin juuressa olevaan Android-kansioon. Tiedosto pitää sisällään Firebase -projektin tiedot. Kun edellä mainitut asiat on tehty, yhteys pitäisi toimia React Native -sovelluksen ja Firebasen välillä. Yhteyden toimivuuden voi kumminkin vielä varmistaa sovelluksen rekisteröinnin ja react-native-firebase -kirjaston asennuksen jälkeen käynnistämällä projektin esimerkiksi emulaattorissa. Tämän jälkeen sovelluksen pitäisi näkyä todennettuna Firebasen hallintapaneelissa.

### 5.4.2 Tiedon lähettäminen ja vastaanottaminen

Firebasessa on tällä hetkellä kaksi erilaista tietokantaa Realtime Database ja Firestore. Molemmat näistä toimii reaaliajassa eli muutos näkyy heti sovelluksessa ja tietokannassa. Erona on vain, että data tallennetaan hiukan erilaisesti ja Firestore on suunniteltu skaalaamaan paremmin isoisiin sovelluksiin. Tässä projektissa käytettiin Firestorea.

Tiedonhaku toteutettiin sovelluksessa siten, että kun sovellus käynnistyi, otettiin yhteys Firestore -tietokantaan. Tietokannasta haettiin kaikki viestit, jotka olivat kokoelmassa nimeltä Posts. Posts-kokoelma sisälsi kaikki käyttäjien lähettämät viestit ja Comments nimisen kokoelman. Viestit muodostuivat otsikosta, leipätekstistä, kuvasta ja aikaleimasta. Kommentit sisälsivät käyttäjän nimen ja kommentin. Aikaleimaa käytettiin viestien ja kommenttien kronologiseen järjestämiseen.

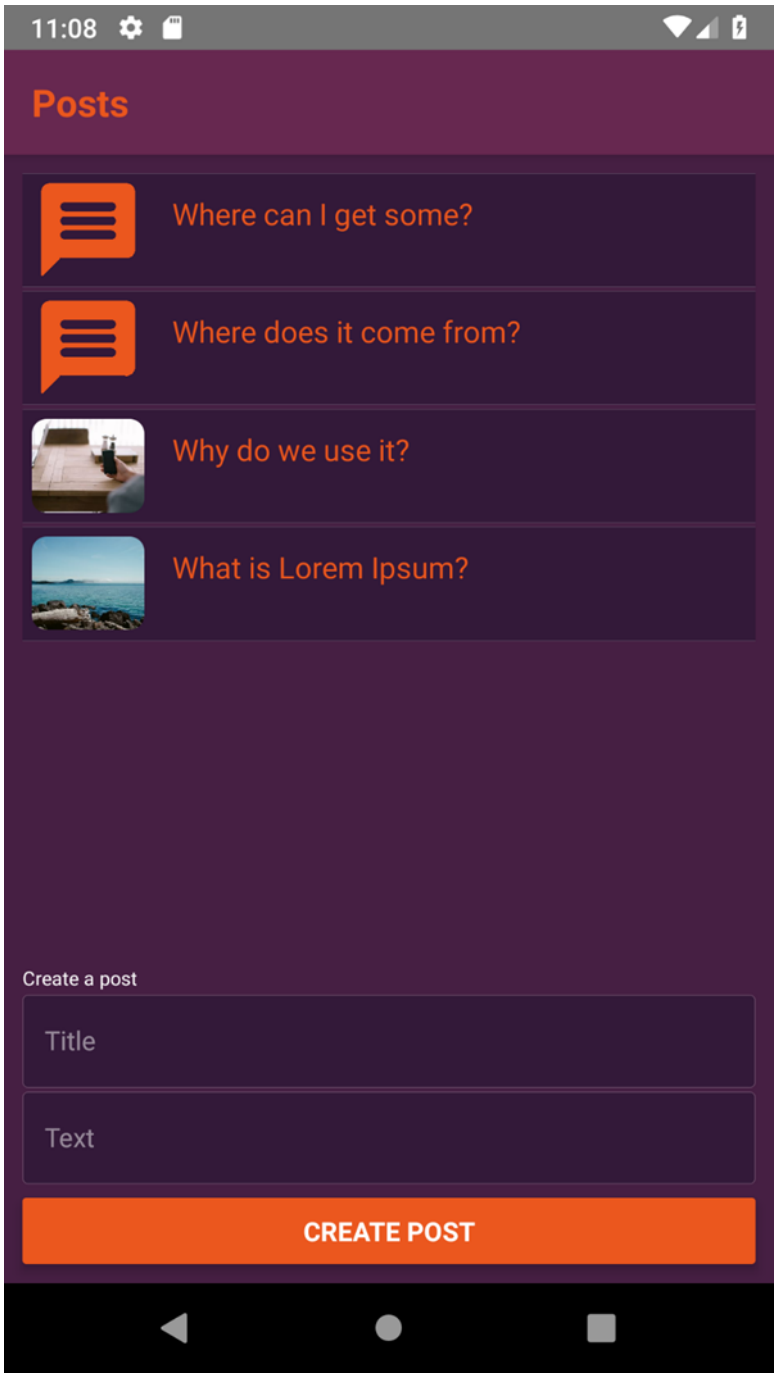
Kuva 24. Käyttäjien kommenttien haku tietokannasta yksinkertaisimmillaan. Kuvan koodissa haetaan tietokannasta comments kokoelma ja siirretään se listaan aika järjestyksessä.

```
48
49      snap = this.postsRef.doc(id).collection('comments').onSnapshot(this.onCollectionUpdate)
50
51      onCollectionUpdate = (querySnapshot) => {
52        const comments = [];
53        querySnapshot.forEach((doc) => {
54          const { user, main, time } = doc.data();
55          comments.push({
56            key: doc.id,
57            doc,
58            user,
59            main,
60            time,
61          });
62        });
63        this.setState({
64            comments: comments.sort((a,b) => a.time - b.time),
65            loading: false,
66        });
67      }
```

Käyttäjien tekemät viestit näytettiin sovelluksen päänäkymässä listana. Tässä vaiheessa päätettiin näyttää vain viesteistä esikatselukuva ja viestin otsikko, jotta sivusta ei tulisi sekava. Viestien esikatselun lisäksi päänäkymässä pystyy tekemään uusia viestejä.

Kommentti näkymään voi siirtyä yksinkertaisesti painamalla halutun viestin kuvaa tai otsikkoa. Näin aukeaa viestin kommentti näkymä, josta voi tarkastella kommentteja, viestin sisältöä ja kirjoittaa uusia kommentteja.

Kuva 25. Sovelluksen päänäkymä, jossa näkyy viestit. Puhekuplalla varustetut viestit sisältävät vain tekstiä.




### 5.4.3 Kirjautuminen

Google Firebasella on monta tapaa toteuttaa kirjautuminen esimerkiksi sähköpostilla, google tilillä, Facebookilla ja muilla sosiaalisen median alustalla. Firebase tarjoaa myös valmiin kirjautumissivun kaikilla edellä mainituilla ratkaisuilla. Tässä sovelluksessa päätettiin käyttää sähköposti ja salasana yhdistelmää kirjautumisessa.

Firebase autentikointi tekee käyttäjän kirjautumisen todennuksen toteuttamisesta hyvin yksinkertaista. Yksinkertaisimmillaan se vaatii muutaman metodin kutsun kuten käyttäjän tietojen lähetys tai kirjautumisen tilan tarkistus.

Kuva 26. Koodinpätkä, jossa tehdään sähköposti ja salasana todennus pyyntö Firebaseen. Kun kirjautuminen onnistuu, siirretään käyttäjä päänäkymään. Jos kirjautuminen epäonnistuu, näytetään käyttäjälle virheilmoitus.

```
10
11  LoginEmailPassword = () => {
12    const { email, password } = this.state
13    firebase
14      .auth()
15      .signInWithEmailAndPassword(email, password)
16      .then(() => this.props.navigation.navigate('Main'))
17      .catch(error => this.setState({ errorMessage: error.message }))
18  }
19
```

Kun käyttäjä kirjautuu palveluun, tallennetaan palvelimelle käyttäjä objekti, joka sisältää käyttäjän tiedot. Tällä objektilla myös määritellään käyttäjän tila eli onko esimerkiksi käyttäjä kirjautunut.

Kuva 27. Kuvan koodissa tarkistetaan käyttäjä kirjautumisen tila. Tilan mukaan ohjataan käyttäjä sovelluksen päänäkymään tai kirjautumisnäkymään.

```
7
8     componentDidMount(){
9       firebase.auth().onAuthStateChanged(user => {
10        this.props.navigation.navigate(user ? 'Main' : 'Login')
11      })
12    }
13
```

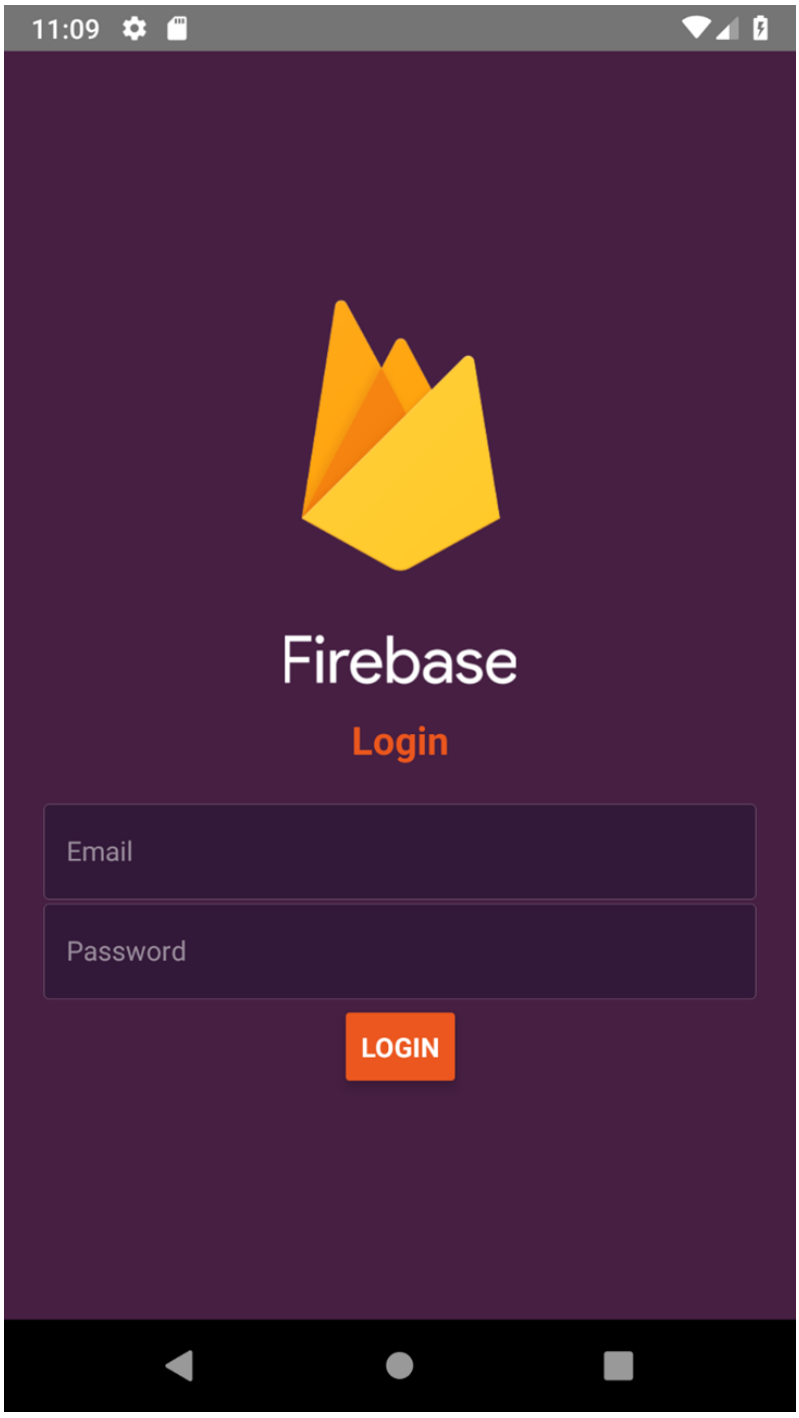
Kun käyttäjän kirjautumisen tila tallennetaan palvelimelle, voidaan tällä välttää niin sanotut ylimääräiset kirjautumiset, jos sovelluksen sammuttaa tai käynnistää uudelleen. Sovelluksen käynnistyksen yhteydessä riittää vain, että käyttäjän tila tarkistetaan.

Tässä sovelluksessa tarkistettiin sovelluksen ensimmäisellä välilehdellä käyttäjän kirjautumisen tila. Tämän mukaan ohjattiin käyttäjä joko sisäänkirjautumisnäkymään tai sovelluksen päänäkymään.

Kuva 28. Yksinkertainen toteutus käyttäjän uloskirjautumisesta. Jos uloskirjautuminen onnistuu, siirretään käyttäjä takaisin aloitusnäkymään.


Sovelluksen kirjautumisen toteutuksen jälkeen vaihdettiin tietokannan käyttäjä oikeudet niin, että vain rekisteröityneet käyttäjät voivat lukea ja kirjoittaa tietokantaan. Tämä estää rekisteröitymättömien käyttäjien lukemasta lähetettyjä viestejä ja niiden lähettämisen täysin. Sovellus kumminkin vaatii, että käyttäjä on kirjautunut ennen, kun pääsee tarkastelemaan viestejä. Sovelluksen olisi voinut myös toteuttaa siten, että käyttäjä voi lukea vistejä ilman kirjautumista, mutta ei kommentoida antamalla tietokantaan lukuoikeudet kirjautumattomille.

Kuva 29. Pilvisovelluksen lopullinen kirjautumisnäkymä.



# 6 YHTEENVETO

Opinnäytetyön tavoitteena oli saada vastaus 3 tutkimuskysymyksiin:

4. Miten Mobile Programming moduulin Xamarin-sisällöt voisi toteuttaa React Nativella (muutamat keskeisimmät asiakokonaisuudet)?
5. Xamarinin kanssa on ollut ongelmia etenkin kehitysympäristön asennuksen kanssa. Miten tämä React Nativella?
6. Miten React Native istuu muuhun opetussisältöömme (React on oma kielensä jne.)?

Opinnäytetyössä onnistuttiin vastaamaan kaikkiin tutkimuskysymyksiin joissain määrin. React Nativen soveltuvuutta HAMK tietojenkäsittely koulutuksen muuhun opetussisältöön arvioitiin pääosin opinnäytetyön tekijän näkökulmasta ja kokemuksesta.

Opinnäytetyön tekijällä ei ollut aikaisempaa kokemusta React Nativesta tai JavaScriptistä, mutta tämä ei estänyt sen perusteiden opiskelua. C# tai Java -kielistä siirtyminen JavaScript kieleen ja React Native -sovelluskehykseen vaatii hiukan aikaa oppia React komponentti rakenne ja tutustua JavaScript-kieleen, mutta ei pitäisi tuottaa ongelmia, jos ohjelmoinnin perusteet on opiskeltu. React Native on helposti opittavissa oleva JavaScript-sovelluskehys, vaikka ei aikaisempaa kokemusta ole JavaScriptistä. Näin ollen soveltuu hyvin HAMK tietojenkäsittelyn muun opetuksen kanssa.

Kehitysympäristön asennuksen ja käytön aikana ilmenneitä ongelmia dokumentoitiin opinnäytetyön käytännönosuuden aikana. Asennuksen ja käytön aikana ei ilmennyt käyttöä estäviä ongelmia tai erityisen lieviä. Suurin osa ongelmista liittyi Android emulaattorin käyttöön, mutta nämä ongelmat ovat korjattavissa Android emulaattorin asetuksista.

Käytännön ja teoriaosuuksissa selvitettiin myös, miten voisi toteuttaa Mobile Programming Xamarin-sisällöt React Nativella. Tässä onnistuttiin tekemällä 3 sovellusta, joissa käytiin läpi React Nativen perusteita ja pilvipalvelun liittäminen React Native -

sovellukseen. Teoriaosuudessa käytiin läpi React Nativen yleisiä käsitteitä, jota ilman käytännönosuus ei olisi mahdollista toteuttaa.

Vaikka React Nativen sopivuuden arviointi keskittyi pääsääntöisesti HAMK tietojenkäsittely koulutukseen. Opinnäytetyötä voi hyödyntää kaikki tahot, jotka ovat kiinnostuneita mobiilikehityksestä, harkitsevat React Nativen käyttöä tai haluavat perustietoa React Nativesta ja sen käytöstä.

React Nativen opittavuuden ja sopivuuden arviointi opinnäytetyössä perustui yhden henkilön näkökulmaan. Tässä tapauksessa ei välttämättä tule tarkasti ilmi, kuinka helposti React Native on opittavissa eri taitotasoille henkilöille. Tätä voisi jatkokehittää lisäämällä useamman henkilön näkökulman työhön.

## LÄHTEET

Daniel Witte & Philipp von Weitershausen, Facebook, React Native for Android: How we built the first cross-platform React Native app, haettu 2019 osoitteesta https://code.fb.com/developer-tools/react-native-for-android-how-we-built-the-first-cross-platform-react-native-app/

Facebook Inc, State, React Native docs, haettu 2019 osoitteesta https://facebook.github.io/react-native/docs/state

Facebook inc, JavaScript Environment, React Native docs, haettu 2019 osoitteesta https://facebook.github.io/react-native/docs/javascript-environment.html

Facebook inc, React Native, Github, haettu 2019 https://github.com/facebook/react-native

Facebook, JSX, haettu 2019 osoitteesta https://github.com/facebook/jsx

Google Trends, Google, haettu 2019 osoitteesta https://trends.google.com/trends/explore?cat=31&date=today%205-y&q=Xamarin,React%20Native,Flutter

Hämeen Ammattikorkeakoulu, haettu 2019 https://huoasl.outsystemsenterprise.com/opetussuunnitelmat/OpetussuunnitelmanTiedot.aspx?CurriculumCodeInput=TRTK16A

Jerry, Stack Overflow, haettu 2019 osoitteesta https://stackoverflow.com/questions/44446523/unable-to-load-script-from-assets-index-android-bundle-on-windows

John Kagga, The Andela Way, haettu 2019 osoitteesta https://medium.com/the-andela-way/understanding-react-components-37f841c1f3bb

Marvin Frachet, Hacker Noon, haettu 2019 osoitteesta https://hackernoon.com/understanding-react-native-bridge-concept-e9526066ddb8

Native Modules, Facebook Inc, haettu 2019 osoitteesta https://facebook.github.io/react-native/docs/native-modules-ios

Paul Krill, Making faster, smoother UIs for data-driven web apps, haettu 2019 React: Making faster, smoother UIs for data-driven Web apps, https://www.infoworld.com/article/2608181/javascript/react--making-faster--smoother-uis-for-data-driven-web-apps.html

Rahul Gaba and Atul R, React Made Native Easy, haettu 2019 osoitteesta https://www.reactnative.guide/3-react-native-internals/3.1-react-native-internals.html

Reactjs.Component, Facebook Inc, haettu 2019 osoitteesta https://reactjs.org/docs/react-component.html

Rohan Paul, Understanding React Component, haettu 2019 osoitteesta https://medium.com/@paulrohan/understanding-reactjs-component-741f3f0e4a8f

Tom Occhino, Facebook, React Native: Bringing modern web techniques to mobile, haettu 2019 osoitteesta https://code.fb.com/android/react-native-bringing-modern-web-techniques-to-mobile/

Windows Apps Team, Microsoft, React Native on the Universal Windows Platform, haettu 2019 osoitteesta https://blogs.windows.com/buildingapps/2016/04/13/react-native-on-the-universal-windows-platform/

**Liite 1: Aineenhallintasuunnitelma**

Opinnäytetyön teon aikana ainestoa säilytettiin paikallisesti tekijän tietokoneella, sekä aineisto varmuuskopioitiin erilliselle kiintolevylle, ja HAMK:n tarjoamaan Office 365™ -pilvipalveluun. Opinnäytetyössä ei käytetty arkaluontoisia tietoja, tai ei-julkisesti saatavilla olevia aineistoja. Aineisto sisältää kuvia, lähdeviitteitä, ja React Nativella tehtyjä esimerkkikoodinpätkiä. Aineistoa säilytetään 1 vuosi opinnäytetyön hyväksymispäivästä, jonka jälkeen se tuhotaan. Aineisto on saavutettavissa osoitteesta: https://hameenamk-my.sharepoint.com/personal/teemu1522_student_hamk_fi/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fteemu1522%5Fstudent%5Fhamk%5Ffi%2FDocuments%2FLiitteet%2Fsrc%2Ezip&parent=%2Fpersonal%2Fteemu1522%5Fstudent%5Fhamk%5Ffi%2FDocuments%2FLiitteet tai pyytämällä opinnäytetyöntekijältä.