KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutus

Tero Kerkkänen

MOBIILISOVELLUS URHEILUSEUROJEN SEURANTAA VARTEN

Opinnäytetyö
Helmikuu 2021

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Helmikuu 2021
Tietojenkäsittelyn koulutusohjelma

Tikkarinne 9
80200 JOENSUU
+358 13 260 600 (vaihde)

Tekijä(t)
Tero Kerkkänen

Nimeke
Mobiilisovellus urheiluseurojen seurantaa varten

Toimeksiantaja
Dataline Group Oy

Tiivistelmä

Opinnäytetyön tarkoituksena oli tutustua mobiilisovelluksien kehitykseen sekä kehittää mobiilisovellus Dataline Group Oy:n urheiluseura-asiakkaille. Sovelluksen tavoitteena oli luoda verkkosivuille vaihtoehtoinen ja aiempaa nopeampi ratkaisu saada urheiluseuroista tietoa. Tiedot sovellukseen haetaan JSON-syötteestä, joka sisältää kuuden urheiluseuran tiedot. Syötteen ylläpito oli Dataline Group Oy:n vastuulla. Sovelluksen yhtenä vaatimuksena oli, että sen pitää toimia Android- ja iOS-käyttöjärjestelmillä.

Sovelluksen kehitykseen valittiin React Native- ja Expo-sovelluskehykset. React Native valittiin toteutukseen, koska se mahdollistaa sovelluksen rakentamisen hybridisovelluksena Android- ja iOS-käyttöjärjestelmille käyttäen samaa koodipohjaa. Expo-sovelluskehystä hyödynnettiin sovelluksen hallintaan ja suorittamiseen.

Sovellus kehitettiin määräajassa toimeksiantajan vaatimuksien mukaisesti. Tällä hetkellä syöte sisältää melkein kokonaan yhden joukkueen tiedot. Muiden joukkueiden tiedot päivitetään syötteeseen myöhemmin. Suunnitelmana on jatkokehittää sovellusta lisäämällä uusia ominaisuuksia ja parantamalla käyttöliittymän ulkoasua sekä syötteen tietoturvaa ja tiedonhakua.

Kieli	Sivuja 33
suomi	Liitteet 0
	Liitesivumäärä 0

Asiasanat

mobiilikehitys, hybridisovellus, react native

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
February 2021
Degree in Business information technology

Tikkarinne 9
80200 JOENSUU
FINLAND
+ 358 13 260 600 (switchboard)

Author (s)
Tero Kerkkänen

Title
Mobile Application to Follow Sports Clubs

Commissioned by
Dataline Group Oy

Abstract

The purpose of the thesis was to gather knowledge about mobile software development and develop a mobile application for Dataline Group Oy's sports club customers. The goal of the mobile application was to create an alternative and a faster solution for the websites to receive information about sports clubs. The data for the application is gathered from JSON feed which contains the data of six sports teams. Maintenance of the feed was Dataline Group Oy's responsibility. One of the application requirements was that it must run on Android and iOS operating systems.

React Native and Expo frameworks were selected for application development. React Native was chosen because it allows the application to be built as a hybrid application for Android and iOS operating system using the same codebase. Expo framework was used to manage and execute the application.

The application was developed on time, according to the commissioner's requirements. Currently, the feed contains nearly all the data of one team. The rest of the team's data will be updated to the feed later. The plan is to develop the application further by adding new features, improve the layout of the user interface as well as feed security and data gathering.

Language	Pages 33
Finnish	Appendices 0
	Pages of Appendices 0

Keywords

mobile development, hybrid application, react native

Sisältö

Käsitteet

AOT	Ahead-of-Time. Ohjelmointikieli Dartin kääntämiseen liittyvä vaihe, jossa saadaan optimoitua koodi halutulle laitteelle. (Flutter 2020a.)
APK	Android Application Package. Android-sovelluksen asennukseen käytetty pakettitiedosto. (PCMag 2021.)
CSS	Cascading Style Sheets. Tyylikieli, jota käytetään esimerkiksi verkkosivujen ulkoasun muokkaamiseen.
Dart	Googlen kehittämä ohjelmointikieli esimerkiksi mobiilisovelluksien toteutukseen (Alessandria 2020, 9).
Flexbox	CSS-tyylikielen tarjoama ominaisuus verkkosivun elementtien sijoittamiseen.
HTML	Hypertext Markup Language. Kuvauskieli verkkosivujen rakennukseen.
JavaScript	Ohjelmointikieli, jota voidaan käyttää esimerkiksi verkkosivuilla olevien toimintojen toteutuksessa.
JavaScript Core	JavaScript-koodin suorittamisen mahdollistava moottori. Moottoria käyttää esimerkiksi Safari-selain. (React Native 2021a.)
JSON	JavaScript Object Notation. Tiedostomuoto, jonka avulla voidaan säilyttää ja lähettää dataa. Käytetään usein palvelimen ja verkkosivun välisessä tiedonsiirrossa. (W3Schools 2021.)

Node.js JavaScript-koodin suorittamisen mahdollistava ympäristö, joka hyödyntää V8 JavaScript-moottoria (Patel 2018).

Silta Ominaisuus, jonka avulla saadaan eri ympäristöjä toimimaan yhdessä (Novick 2017, 17).

Skia 2D-grafiikkakirjasto, jota esimerkiksi Google Chrome -selain käyttää grafiikkamoottorina (Bellinaso 2018).

WebView Selaimen tapainen puhelimessa oleva komponentti, jonka avulla voidaan näyttää dataa (Biessek 2019, 97).

1 Johdanto

Mobiililaitteiden käyttö on nykypäivänä hyvin yleistä. Käytämme niissä olevia sovelluksia esimerkiksi uutisten lukemiseen, kommunikointiin ja yleisen ajanvietteen apukeinona. Statistan syyskuussa vuonna 2019 tekemän tutkimuksen mukaan älypuhelimen käyttäjämäärä on kasvanut vuodesta 2012 alkaen. Kasvun ennustetaan myös jatkuvan vuosina 2020 ja 2021. Vuodelle 2021 ennustetaan, että älypuhelimien käyttäjämäärä nousisi 3,8 miljardiin. (O'Dea 2019.) Tästä voidaan päätellä, että sovelluksille tulee olemaan kysyntää mobiililaitteiden käyttäjiltä myös tulevaisuudessa. Tästä syystä sovelluksien kehittäminen mobiililaitteille on suositeltava taito opetella, jos haluaa mahdollisuuden työllistyä ohjelmointialalla.

Opinnäytetyöni tarkoituksena oli tutustua mobiilisovelluksien kehitysmahdollisuuksiin sekä toteuttaa mobiilisovellus. Mobiilisovelluksen kehitystavoitteena oli luoda ohjelmointiyritys Dataline Group Oy:n urheiluseura-asiakkaiden verkkosivuille vaihtoehtoinen ratkaisu urheiluseurojen seurantaa varten. Sovellus on yksinkertaistettu versio joukkueiden omista kotisivuista. Tavoitteenani oli toteuttaa toimeksianto viimeistään joulukuun loppuun mennessä.

Valitsin mobiilisovelluksen kehittämisen opinnäytetyön aiheeksi, koska minulla ei ollut aiempaa kokemusta kyseisestä aiheesta. Ajattelin, että tässä on oiva mahdollisuus oppia jotain täysin uutta. Yleinen kiinnostus urheilua kohtaan auttoi myös päätöksessä ja toive siitä, että pääsen sovelluksen avulla parantamaan urheiluseurojen näkyvyyttä.

Raportin toisessa luvussa käydään läpi eri mobiilisovellustyyppejä ottaen huomioon niiden hyvät ja huonot puolet. Kolmannessa luvussa käydään läpi kolmen eri hybridisovelluskehyksen esittely. Neljännessä luvussa kerrotaan tarkemmin toimeksiannosta ja sen vaatimuksista. Viidennessä luvussa keskitytään ohjelman toteutuksessa käytettyihin menetelmiin. Kuudennessa luvussa kerrotaan, kuinka

ohjelma on toteutettu tärkeimpien osa-alueiden osalta. Seitsemännessä luvussa keskitytään tuloksiin esittelemällä ohjelman käyttöliittymää. Kahdeksannessa luvussa pohditaan, kuinka opinnäytetyöprosessi meni ja kuinka valittu sovelluskehys sopi ohjelman rakennukseen. Pohdinnassa tarkastellaan myös ohjelman jatkokehitystä.

2 Mobiilisovellustyypit

2.1 Web-sovellus

Web-sovelluksesta puhuessa tarkoitetaan sovellusta, joka käynnistetään puhelimessa olevan web-selaimen kautta esimerkiksi käyttämällä Mozilla Firefoxia tai Google Chromea. Tällöin käyttäjän ei tarvitse ladata ja asentaa sovellusta omaan puhelimeensa esimerkiksi Google Play -kaupan tai iOS App Storen kautta. Tästä sovellustyypistä puhuessa mobiiliverkkosivun ja web-sovelluksen ero on varsin vähäinen ja on paljon henkilön omasta mielipiteestä kiinni, mihin rajan laittaa näiden kahden välillä. (Griffith 2017, 4.)

Ratkaisun hyvänä puolena on se, että sovellus alkaa toimimaan muilla mobiilikäyttöjärjestelmillä Androidin ja iOS:n lisäksi suoraan, jolloin koodiin ei välttämättä tarvitse tehdä mitään muutoksia. Ohjelman päivitykset on myös helppo saada käyttäjien saataville siirtämällä päivitetty koodi palvelimelle, jolloin käyttäjän ei tarvitse huolehtia omatoimisesti sovelluksen päivittämisestä. (Griffith 2017, 4.)

Huonona puolena on se, että ohjelmalle on hankala saada näkyvyyttä, koska sitä ei löydy puhelimen tukeman sovelluskaupan kautta (Griffith 2017, 4). Ohjelman testaus on myös aikaa vievää, koska sovellus pitää testata Android ja iOS ympäristössä usealla eri selaimella.

2.2 Natiivisovellus

Natiivisovelluksella tarkoitetaan mobiiliohjelmaa, joka on toteutettu käyttämällä käyttöjärjestelmän oletuksena tukemaa ohjelmointikieltä. Androidilla tällainen ohjelmointikieli on esimerkiksi Java ja iOS:lle Swift. (Griffith 2017, 3.)

Natiivisovelluksen hyvänä puolena on se, että kehittäjä pääsee toteuttamaan ohjelmaa kehitysympäristössä ohjelmointikielellä, joka on suunniteltu mobiilisovelluksen käyttöjärjestelmää varten. Natiiveja kehitysympäristöjä ovat esimerkiksi Android studio Androidille ja Xcode iOS:lle. Natiivisovelluksen käyttö on myös sulavaa ja nopeaa, koska se pyörii omassa natiivissa ympäristössään ilman ylimääräisiä muunnoksia. (Griffith 2017, 3–4.)

Huonona puolena on se, että ohjelma ei toimi suoraan kuin yhdessä käyttöjärjestelmässä. Ohjelman toteutus pitäisi tehdä kahdella eri ohjelmointikielellä, jotta ohjelma saadaan toimimaan Android- ja iOS-ympäristössä. (Griffith 2017, 4.) Yritysten täytyy palkata mahdollisesti useampi työntekijä, jotka osaavat tarvittavat koodauskielet ja tämä aiheuttaa ylimääräisiä kuluja. Sovelluksen kehityssykli kokonaisuudessaan on pitempi natiivisovellusta tehdessä kuin web-sovelluksessa tai hybridisovelluksessa.

2.3 Hybridisovellus

Hybridisovellus-termistä puhuttaessa tarkoitetaan mobiiliohjelmaa, jonka kehitykseen käytetään samaa koodipohjaa eri käyttöjärjestelmillä. Hybridisovellukset voidaan jakaa kahteen eri kategoriaan, jotka toimivat hieman eri tavalla. (Khanna, Yusuf & Phan 2017, 5.)

Ensimmäisessä kategoriassa ovat WebView-komponenttia hyödyntävät ratkaisut, jotka mahdollistavat esimerkiksi HTML-, JavaScript- ja CSS-koodilla luodun ohjelman. Toisessa kategoriassa oleva cross-compiled-ratkaisu käyttää

ainoastaan yhtä ohjelmointikieltä. Se muuntaa koodin esimerkiksi siltaratkaisua hyödyntäen käyttöjärjestelmän tukemiksi natiiveiksi komponenteiksi tai sovelluskehys hyödyntää omia kustomoituja komponentteja. (Khanna ym. 2017, 5.)

Hybridisovelluksen hyvänä puolena on se, että sovellus toteutetaan käyttämällä samaa koodipohjaa esimerkiksi Android- ja iOS-ympäristöön, mikä säästää kehitykseen tarvittavaa aikaa (Griffith 2017, 5). Käyttöjärjestelmät voivat kuitenkin tarvita käyttöjärjestelmäkohtaisia koodimuunnoksia, jotta kaikki toimii (Khanna ym. 2017, 5). Tällainen tilanne voi syntyä esimerkiksi, jos jokin komponentti ei tue kumpaakin käyttöjärjestelmää tai laitteen natiivi ominaisuus ei toimi oikein.

Yhtenä huonona puolena on hybridisovelluksen ylläpitokustannukset. Vaikka päivitykset yleensä korjaavat asioita, ne voi myös rikkoa sovelluksessa olevia ominaisuuksia. Ongelmia voi aiheuttaa sovelluskehyksen tai käyttöjärjestelmän päivitykset. Korjauksia täytyy ongelmatilanteissa tehdä, mikä aiheuttaa ylimääräisiä kustannuksia. (Avendano 2019.)

3 Hybridisovelluskehyksiä

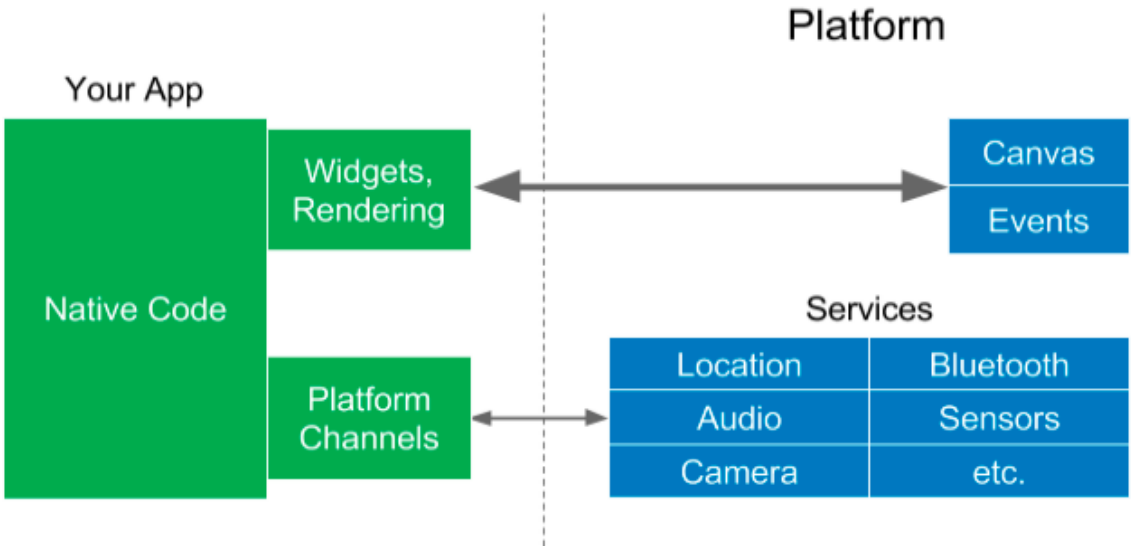
3.1 Flutter

Flutter on Googlen vuonna 2018 julkaisema avoimen lähdekoodin sovelluskehys mobiilisovelluksien kehitykseen (Biessek 2019, 95). Flutter käyttää Dart-ohjelmointikieltä, jonka ensimmäinen vakaa versio julkaistiin vuonna 2013 ja kehityksen myötä Dart 2.0 julkaistiin vuonna 2018. Aiempi kokemus JavaScript- tai Java-ohjelmointikielistä helpottaa Dart-ohjelmointikielen oppimista. (Alessandria 2020, 9.) Flutterilla toteutettuja sovelluksia ovat esimerkiksi eBay motors app, Google Ads ja Google Stadia -sovellukset (Flutter 2020b).

Flutter hyödyntää toiminnassaan Dart-ohjelmointikielen AOT-ominaisuutta. Esimerkiksi Android-käyttöjärjestelmään käännetty ohjelma toimisi seuraavasti: AOT-ominaisuuden avulla ohjelman koodi käännetään laitteen tukemaksi

kirjastoksi ja se siirretään osaksi APK-asennuspakettia. Ohjelman asennuksen jälkeen laite käyttää luotua kirjastoa ohjelman suorittamiseen. (Flutter 2020a.)

Flutter huolehtii käyttöliittymän rakennuksesta kokonaan itse hyödyntämällä omia komponentteja tietojen näyttämiseen. Käyttöjärjestelmän pitää tukea Skia-grafiikkakirjastoa, jota hyödyntämällä käyttöliittymä toteutetaan. (Bellinaso 2018.) Laitteen palveluiden käyttöön Flutter hyödyntää laitekohtaisia metodeja. Ohjelmaan pitää laitekohtaisesti kirjoittaa omat koodit, joilla voidaan haluttuja palveluita hyödyntää. (Flutter 2021a.) Nämä asiat mahdollistavat sen, että Flutter toimii nopeasti, koska se ei tarvitse siltaratkaisua tai kommunikointia WebView-komponentin kanssa (kuvio 1).



Kuvio 1. Flutter-sovelluskehyksen rakenne (Leler 2017).

Vaikka Flutter-sovelluskehys pyrkii matkimaan natiivia lopputulosta, se ei siihen aina kykene. Ongelma tulee esille, jos käyttöjärjestelmän natiivi komponentti päivittyy. Tässä tapauksessa pitää odottaa, että sovelluskehys päivittyy myös, jotta saadaan uusi versio komponentista saataville. Kehittäjän pitää tehdä myös tarvittavat muutokset koodiin, jotta päivitetty versio komponentista saadaan lisättyä ohjelmaan. (Bellinaso 2018.) Tämä ei tietenkään ole ongelma, jos käyttäjä hyväksyy, että kaikki komponentit ei ole aina natiiveja ulkoasultaan.

Kuvassa 1 on esimerkki Dart-ohjelmointikielen käytöstä. Flutterin sisältämä koodi koostuu käytännössä kokonaan widgeteistä, joilla tarkoitetaan sovelluksen eri

komponentteja (Alessandria 2020, 28). Widgetit merkitään koodiin syntaksilla widget(). Sulkujen sisään laitetaan haluttu tieto esimerkiksi toinen widget, kuten kuvassa 1 on tehty.

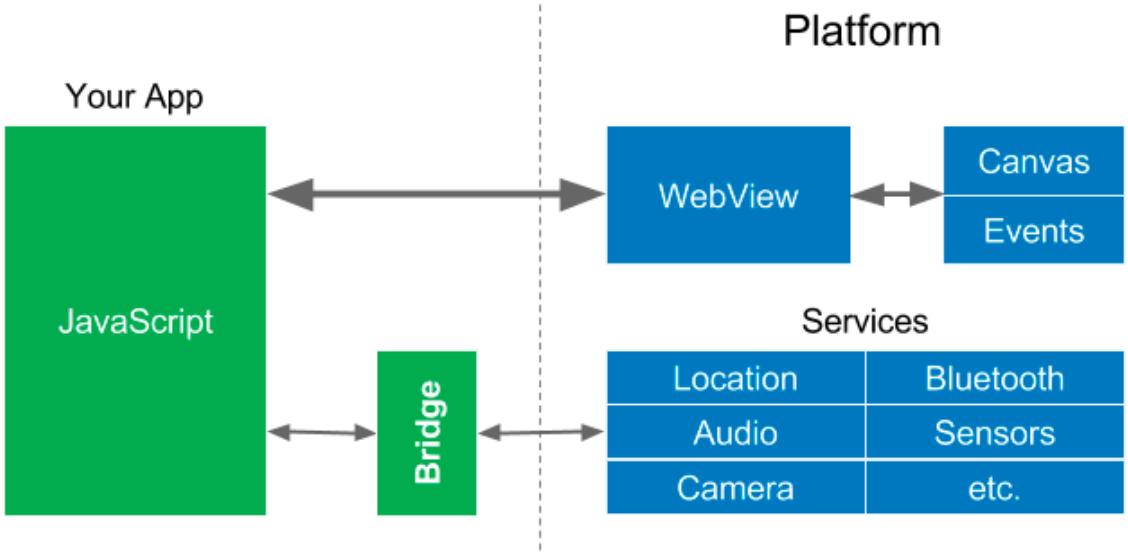
```
import 'package:flutter/material.dart';

void main() {
  runApp(
    Center(
      child: Text(
        'Tämä teksti tulostuu keskelle ruutua.',
        textDirection: TextDirection.ltr,
      ),
    ),
  );
}
```

Kuva 1. Esimerkki Flutter-sovelluskehyksen käytöstä (mukaillen Alessandria 2020, 32–34).

Koodiin tuodaan ensimmäisenä import-komentoa hyödyntäen paketti, josta voidaan hakea koodiin tarvittavia widgettejä. Main metodissa määritellään runApp-funktion suorittaminen. Seuraavalla rivillä oleva Center on flutterin oma widget, joka mahdollistaa elementtien sijainnin määrittelyn keskelle ruutua automaattisesti. Child-ominaisuuden avulla saadaan kytkettyä widgettejä toisten widgetin lapsiksi. Eli tässä tapauksessa Text widget on Center widgetin lapsi, joka laittaa tekstin keskelle ruutua. Text widgetin sisälle tulee teksti, joka halutaan tulostaa. TextDirectionia käytetään nimensä mukaan ilmoittamaan tekstin suunta, joten ltr tarkoittaa vasemmalta oikealle. (Alessandria 2020, 33–34.)

3.2 Ionic

Ionic on vuonna 2013 julkaistu avoimen lähdekoodin sovelluskehys, joka tukee HTML-, CSS- ja JavaScript-kieliä. Ionic sovellukset renderöidään WebView-komponenttia hyödyntämällä. (Griffith 2017, 4–5.) Se käyttää toiminnassaan myös Apache Cordova -sovelluskehystä tai Capacitor-sovelluskehystä, joiden tehtävä on luoda silta koodin ja puhelimessa olevien natiivien ominaisuuksien välille (kuvio 2) (AltexSoft 2019).



Kuvio 2. Ionic-sovelluskehyksen rakenne (Leler 2017).

Kehityksen myötä Ionic on alkanut tukemaan useita muita sovelluskehyksiä, joista uusimpana on tuki Vue.js-sovelluskehykselle (DeBeasi 2020). Ionic- ympäristöä on helpompi lähteä käyttämään, jos jokin sen tukema sovelluskehys on kehittäjälle ennestään tuttu. Ionicin avulla on tehty sovelluksia esimerkiksi Nasa:lle, NHS:lle ja BMW:lle (Ionic 2020a).

Ionic-sovelluksen yhtenä heikkoutena voidaan pitää sen käyttämää WebViewta. WebViewta käyttävän sovelluksen toimintanopeus voi vaihdella riippuen siitä, kuinka paljon grafiikkaa ohjelma sisältää. Hitautta voi myös aiheuttaa se, kuinka paljon ohjelma joutuu kommunikoimaan siltaa hyödyntämällä puhelimen natiivien ominaisuuksien kanssa. (AltexSoft 2019.)

Ionic tarjoaa laajan web-komponenttikirjaston käyttöliittymien rakennukseen sekä niille useita eri muokkausvaihtoehtoja. Se käyttää Flutter-sovelluskehyksen tavoin omia komponentteja, jotka matkivat natiivia lopputulosta. (Netkow 2021.) Komponentit alkavat aina <ion-elementti> alkuisesti ja loppuvat </ion-elementti> (Ionic 2020b). Kuva 2 sisältää esimerkin, kuinka ohjelmaa voidaan rakentaa ion-komponentteja hyödyntämällä.

```
<ion-app>
  <ion-content>
    <ion-row style="display:flex;align-items:center;justify-content:center;height:100%">
      <ion-text>Tämä teksti tulostuu keskelle ruutua.</ion-text>
    </ion-row>
  </ion-content>
</ion-app>
```

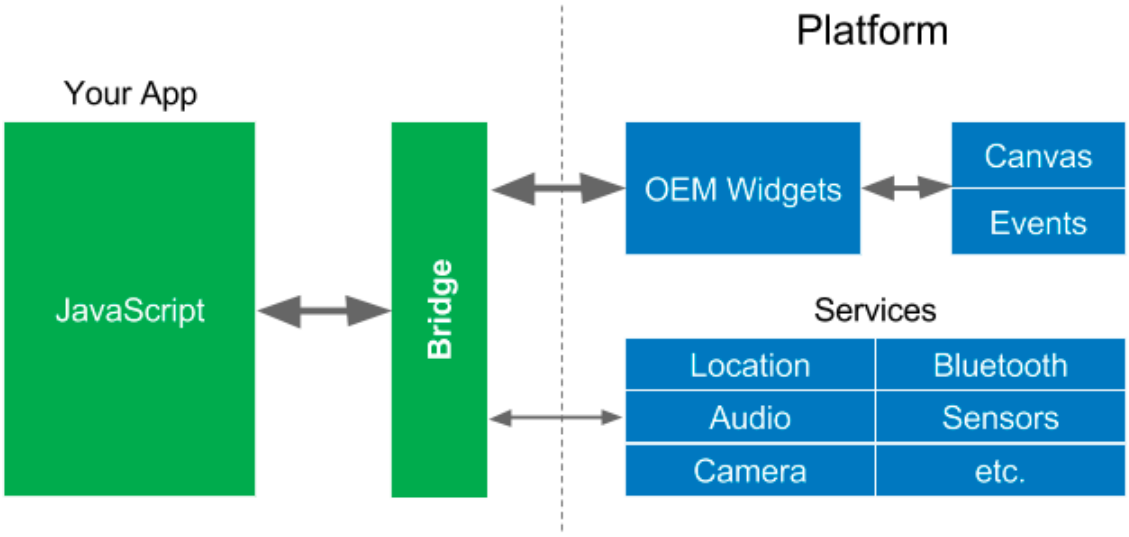
Kuva 2. Esimerkki Ionic-sovelluskehyksen käytöstä.

Ohjelmaan luodaan ion-row-elementti ja määritellään sille tarvittavat tyylit, jotta sen sisällä oleva teksti saadaan siirrettyä keskelle näyttöä. Tässä esimerkissä käytetään apuna flexboxia, jotta teksti saadaan sijoitettua halutusti.

3.3 React Native

React Native on vuonna 2015 Facebookin julkaisema ja JavaScriptiä hyödyntävä avoimen lähdekoodin kirjasto mobiilisovelluksien kehitykseen. React Native pohjautuu web-kehityksessä käytetyn React-sovelluskehyksen ominaisuuksiin. (Masiello & Friedmann 2017, 45.) React Nativella toteutettuja sovelluksia ovat esimerkiksi Facebook, Instagram, Airbnb, Tesla ja Skype (React Native 2020a).

React Native toteuttaa natiivia ratkaisua hyödyntämällä JavaScriptCorea. JavasScriptCore ajaa ohjelman koodin ja React Native hyödyntää omia laitekohtaisia siltaratkaisuja. Siltaratkaisun avulla kommunikointi onnistuu laitteen natiivin osion ja koodatun tiedon välillä, jolloin koodi muutetaan laitteelle natiiveiksi komponenteiksi. React Native hyödyntää myös siltaratkaisua koodin ja laitteen natiivien ominaisuuksien kanssa toimimiseen (kuvio 3). (Masiello & Friedmann 2017, 45.)



Kuvio 3. React Native -sovelluskehyksen rakenne (Leler 2017).

React Native -sovelluskehyksen yhtenä heikkoutena on sen hyödyntämä siltaratkaisu. Useasti tapahtuva kommunikointi voi aiheuttaa hitautta ohjelmassa varsinkin, jos tietoa joudutaan näyttämään kerralla paljon ja sitä selataan nopeasti. (Cook 2020.)

Kuva 3 sisältää perusesimerkin React Nativen koodin syntaksista. React nativessa voidaan luoda komponentteja joko funktioina tai luokkina (React Native 2020b). Tarvittavat komponentit otetaan ohjelmassa käyttöön import-komennolla koodin alussa (Novick 2017, 46). Esimerkissä on käytetty komponentteja React, Component, Text ja View. Aaltosulkeita ei tarvita komponenttien tuomiseen, jos komponentti on tuotu käyttämällä export default -komentoa pelkän export-komennon sijaan.

```
import React from 'react';
import { Text, View } from 'react-native';

class Test extends React.Component {
    render() {
        return (
            <View style={{ flex: 1, justifyContent: "center", alignItems: "center" }}>
                <Text>Tämä teksti tulostuu keskelle ruutua!</Text>
            </View>
        );
    }
}
export default Test;
```

Kuva 3. Esimerkki React Native -sovelluskehyksen käytöstä luokkakomponenttina (mukaillen React Native 2020b).

Luokka Test perii Component-komponentin react-kirjastosta (React 2020). Luokan sisällä olevaan render-funktioon laitetaan koodi, joka halutaan näyttää. View-komponentin sisälle luodaan halutut tyylit ja tässä esimerkissä hyödynnetään myös Flexboxin mahdollistamia ominaisuuksia. Text-komponentin sisälle lisätään tulostettava teksti. Koodin lopussa olevalla export default -komennolla määritellään, mitä luokkaa halutaan käyttää. Luvun 3.2 ja tämän luvun esimerkeissä tyylit on laitettu suoraan koodiin, mutta normaalisti ne sijoitetaan omaan tyylitiedostoon koodin lukemisen selkeyttämiseksi.

4 Toimeksianto

Opinnäytetyöni toiminnallisen osuuden aiheena oli toteuttaa ohjelmointiyritys Dataline Group Oy:n urheiluseura-asiakkaille mobiilisovellus. Sovelluksen tavoitteena oli luoda verkkosivuille vaihtoehtoinen ja aiempaa nopeampi ratkaisu saada urheiluseuroista tietoa. Sovelluksen vaatimuksena oli, että sen pitää toimia Android- ja iOS-käyttöjärjestelmillä. Tästä syystä oli selkeää alusta alkaen, että ohjelmaa on järkevä alkaa toteuttamaan hybridisovelluksena ottaen huomioon opinnäytetyöhön käytössä olleen ajan.

Tarvittavat tiedot mobiilisovellukseen haetaan Dataline Group Oy:n hallitsemasta syötteestä JSON-muodossa. Syöte sisältää alkuun kuuden eri urheiluseuran tiedot. Näistä minun tehtäväni oli varmistaa, että kolmen joukkueen tiedot toimivat oikein. Tulevaisuudessa uudet urheiluseurat, jotka tulevat Dataline Group Oy:n asiakkaaksi, pääsevät mobiilisovellukseen mukaan, niin halutessaan. Sovellus on tarkoitus lisätä mukana olevien urheiluseurojen kotisivuilta ladattavaksi.

Sovelluksen ensimmäisellä käynnistyskerralla käyttäjä siirretään valitse joukkue -sivulle, jossa näkyy lista joukkueista, joiden tiedot löytyvät ohjelmasta. Käyttäjä klikkaa haluamaansa joukkuetta ja siirtyy valitun joukkueen aloitussivulle. Tiedon valitusta joukkueesta pitää myös jäädä sovelluksen muistiin. Seuraavalla

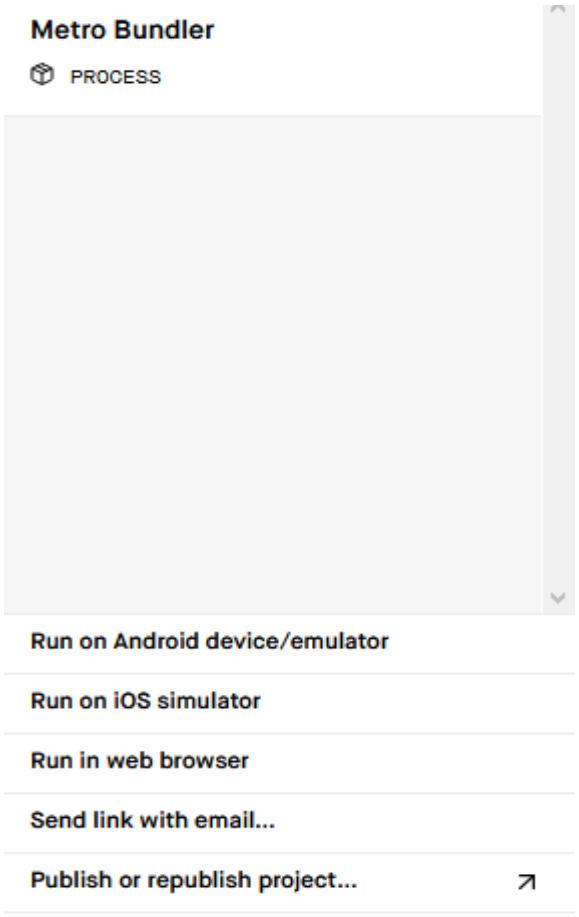
käynnistyskerralla käyttäjä pitää ohjata suoraan valitsemansa joukkueen aloitussivulle eikä valitse joukkue -sivulle.

5 Menetelmät

5.1 Kehitysympäristö

Ohjelman kehitykseen valitsin React Native -sovelluskehyksen, koska sen käyttämä JavaScript-ohjelmointikieli oli itselle ennestään jonkin verran tuttu. Tiesin ennakkoon, että React-sovelluskehystä hyödynnetään esimerkiksi web-sovelluksien rakennuksessa ja tulevaisuutta varten uskoin, että sitä on helpompi lähteä käyttämään React Nativen perusteella. React Native oli myös toimeksiantajan suosittelema sovelluskehys ohjelman toteutukseen, kuten myös Expo-sovelluskehyksen hyödyntäminen.

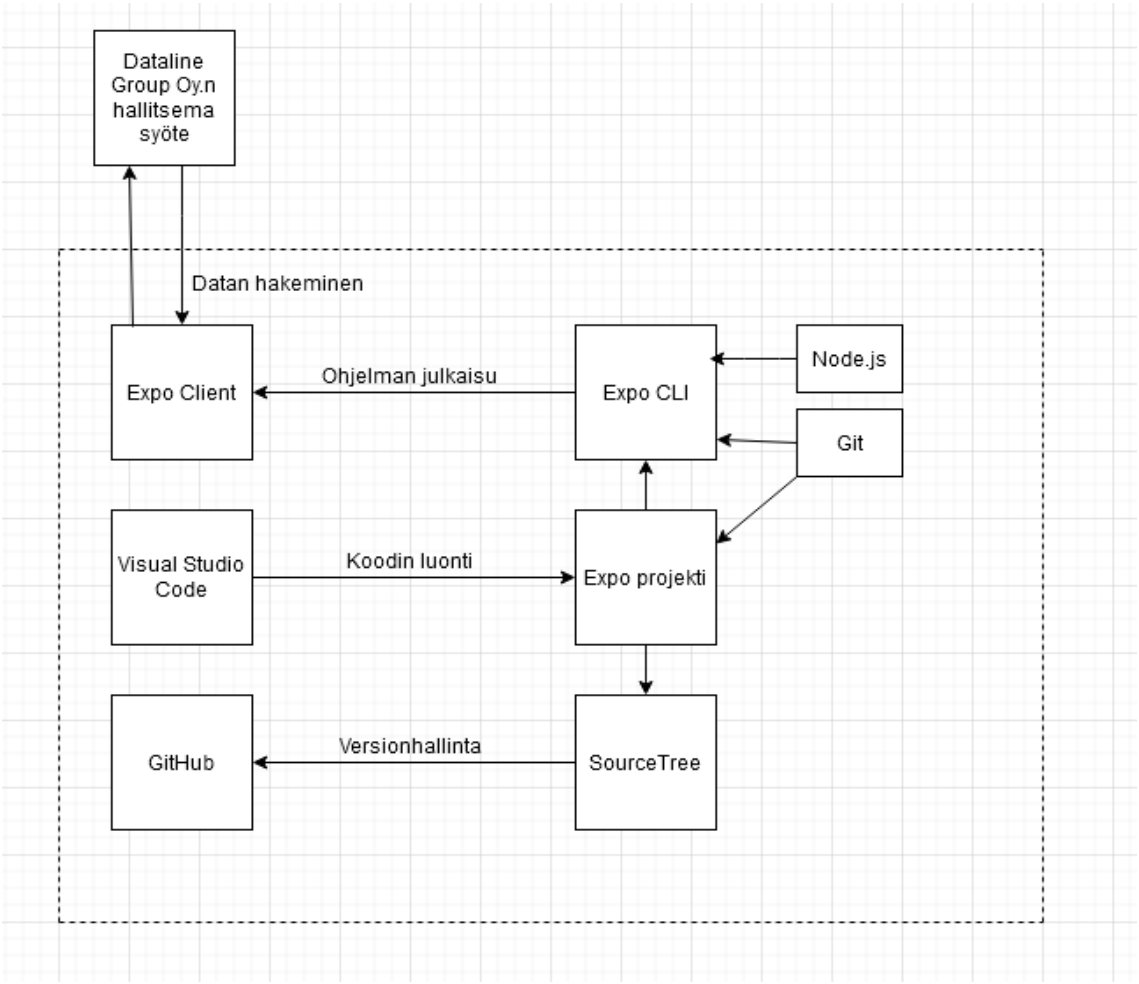
Expo mahdollistaa React Native -sovelluksen hallinnan ja suorittamisen. Expo vaatii Windows-käyttöjärjestelmälle Node.js-ympäristön ja Git-versionhallintajärjestelmän toimiakseen. Expo sovelluskehyksen työkaluista käytin toteutuksen aikana Expo CLI- ja Expo Client -työkaluja. Projektia voidaan suorittaa joko komentokehotteen tai graafisen käyttöliittymän kautta. Komentokehote ja graafinen käyttöliittymä tarjoavat samat ominaisuudet, joten on käyttäjästä kiinni kumpaa haluaa käyttää. (Expo 2020.) Kuvassa 4 näkyy kuva Expo CLI:n käyttämästä graafisesta käyttöliittymästä.



Kuva 4. Expo CLI -graafinen käyttöliittymä.

Publish or republish project -nappia painamalla projekti voidaan siirtää Expo clientille käynnistettäväksi. Expo client on iOS- ja Android-käyttöjärjestelmille tarjolla oleva ohjelma, jonka avulla projektia voidaan suorittaa fyysisellä laitteella ilman, että ohjelmaa tarvitsee asentaa laitteeseen. (Expo 2020.)

Expolla luotu projekti tarjoaa myös hyödyllisiä kirjastoja esimerkiksi ohjelman ulkoasun parantamiseen. Näistä käytin projektissa esimerkiksi ikonikirjastoa, jota hyödynsin mobiilivalikon rakennuksessa. Kuvio 4 sisältää kokoisuudessaan kehitysympäristön kaaviona kuvattuna.



Kuvio 4. Kehitysympäristö kaaviona.

Sovelluksen versionhallinnan toteutin Git, GitHub sekä SourceTreen yhteistyönä. SourceTree tarjoaa graafisen käyttöliittymän Git-versionhallintajärjestelmän käyttämiseen. GitHub on projektien versionhallintaa mahdollistava palvelu verkossa. Koodin kirjoittamisen käytin Visual Studio Code -tekstieditoria. Katkoviivalla merkitty osuus oli opinnäytetyön toteutuksessa minun vastuullani. Dataline Group Oy oli vastuussa kokonaan syötteen toiminnasta.

5.2 Työprosessi

Työn aloitin tutustumalla React Nativen ja Expon perusteisiin kirjojen, videoiden ja dokumenttien avulla. Perusteita aloin testaamaan luomalla oman testiprojektin. Tätä projektia käytin kehityksen aikana omiin testailuihin. Kun omasta mielestäni perusteet oli tarpeeksi hallussa, aloitin varsinaisen ohjelman toteuttamisen. Loin projektin yrityksen Expo-tunnukselle ja aloitin tarkemman tutustumisen, kuinka

toimeksiantajan vaatimat ominaisuudet saan ohjelmaan toteutettua. Muutoksia näihin valintoihin tuli työprosessin aikana, kun löytyi esimerkiksi parempi kirjasto HTML-kuvauskieltä sisältävän tiedon renderöimiseen.

Ohjelman testausta kehityksen aikana tein joko web-selaimen tai Android-emulaattorin kautta. Toteutuksen aikana siirsin projektin Expo Client -ohjelmaan myös useita kertoja päivässä, jotta pystyin varmistamaan, että ohjelma toimii odotetusti fyysisellä laitteella. Tämä mahdollisti myös toimeksiantajalle nähdä päivittäin, kuinka ohjelma on edistynyt. Android-toimivuuden testasin Samsung Galaxy A20e -puhelimella ja iOS-laitteen testauksen iPhone 7 -puhelimella. Versionhallintaan pyrin viemään päivitetyn koodin noin 1–3 kertaa päivässä riippuen siitä, kuinka paljon ominaisuuksia olin ehtinyt saamaan aikaiseksi.

Projektin alussa pidin toimeksiantajan ajan tasalla ohjelman edistymisestä viikoittain lähettämällä laajemman tekstikokonaisuuden, mitä ohjelmaan olen saanut tehtyä. Myöhemmin projektin edistyessä aloimme pitämään viikoittain palaveria, jossa käytiin läpi tarkemmin projektintilannetta, syötteeseen tarvittavia muutoksia ja mitä asioita ohjelmasta puuttui. Palavereissa sain toimeksiantajalta myös vinkkejä koodin parantamiseen.

5.3 Props ja state

React Nativen komponenttien tiedonhallintaan voidaan hyödyntää kahta erilaista tietotyyppiä, ja nämä ovat nimeltään props ja state. Props tulee sanasta properties, jotka lähetetään komponentilta toiselle (Masiello & Friedmann 2017,15). Propsien avulla toteutin esimerkiksi komponenttikohtaisesti sovelluksen hakuosoitteen muuttamiseen.

State eroaa propseista niin, että sitä tietoa ei lähetetä komponentille. State on komponentin sisäistä tietoa ja sen tietoa voidaan päivittää komponentin suorittamisen aikana. (Masiello & Friedmann 2017, 27–28.) Ohjelman toteutuksessa käytin state-tietotyyppiä esimerkiksi syötteestä haetun tiedon tallentamiseen.

5.4 React Navigation

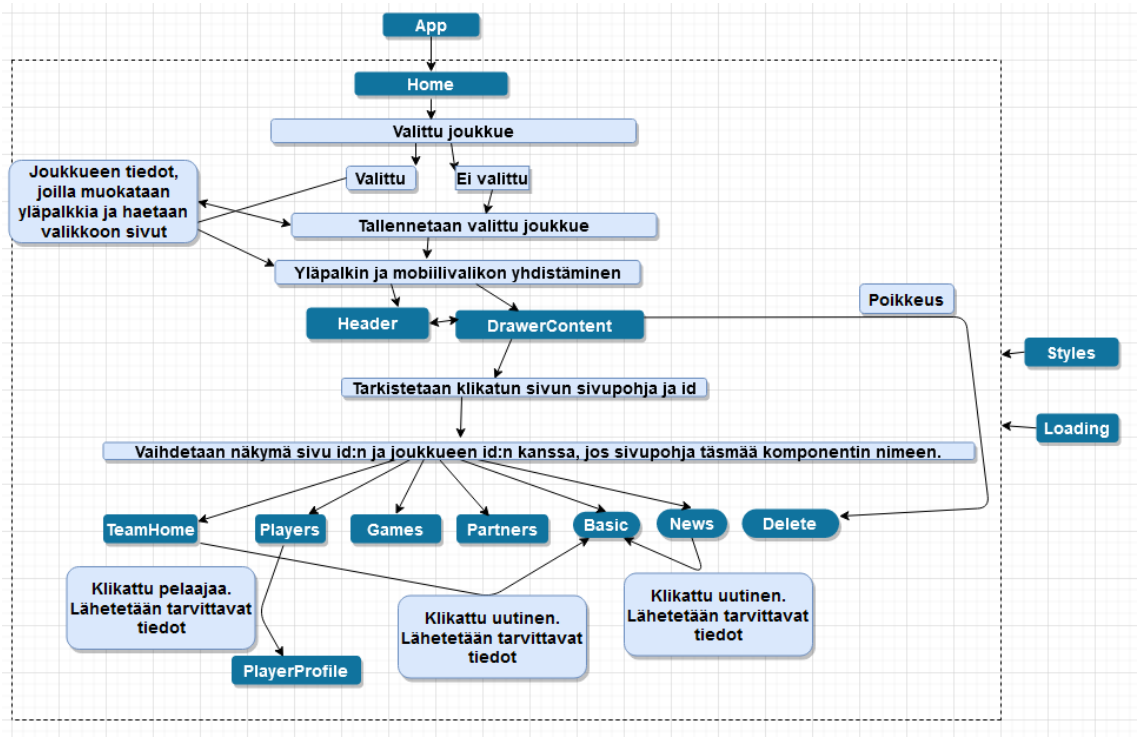
React Navigation 5.0 tarjoaa kolme erilaista ratkaisua suorittaa navigointi ja ne ovat stack navigator, drawer navigator ja tab navigator. Stack navigatorin avulla saadaan lisättyä haluttuja komponentteja pinoon, jolloin liikkuminen tapahtuu pinossa olevien komponenttien välillä. Stack navigator luo automaattisesti myös yläpalkkiin napin, jonka kautta voidaan palata edelliseen komponenttiin. Tab navigator mahdollistaa menun luomisen välilehtinä, jotka ilmestyvät esimerkiksi puhelimen alareunaan. Drawer navigator mahdollistaa perinteisen mobiilivalikon luonnin, jossa mobiilivalikon avaaminen tapahtuu esimerkiksi pyyhkäisemällä puhelimen näytön reunasta. (Mclaren 2020.)

Ohjelmassa päädyin käyttämään stack navigatorin ja drawer navigatorin yhdistelmää. Navigoinnin rakennuksesta on tarkemmin tietoa luvussa 6.4.

6 Ohjelman toteutus

6.1 Toimintakaavio

Kuvio 5 sisältää ohjelman toimintakaavion. Tummansinisellä taustalla merkityt ovat ohjelman komponentteja. Vaaleammalla sinisellä merkityt laatikot ovat tapahtumia, joita komponenteista toisiin siirtyessä tapahtuu.



Kuvio 5. Ohjelman toimintakaavio.

Styles-komponentti sisältää ohjelman tyylit, joilla muokataan katkoviivan sisäpuolella olevien komponenttien ulkoasua. Loading-komponentti näytetään katkoviivan sisällä olevien komponenttien kanssa yhdistelmänä tietojen haussa, josta seuraavassa luvussa lisää. Kuvassa näkyvästä poikkeustapahtumasta kerrotaan lisää luvun 6.4 lopussa.

6.2 Ohjelman käyttämän datan hakeminen ja esittäminen

Ohjelman käyttämät hakuosoitteet ovat muodoltaan yleensä api.sportti.org/sites/joukkueid/kysely. Ilman kyselyä hoidetaan kaikkien joukkueiden hakeminen valitse joukkue -sivulle osoitteesta api.sportti.org/sites. Kyselyyn voidaan käyttää joko nimeä tai sivu id:tä. Nimeä voidaan käyttää seuraaviin kyselyihin: menu, home, games, news, players ja partners. Esimerkiksi api.sportti.org/sites/1/players hakee joukkueen id 1 kaikki pelaajat.

Ohjelma poimii tarvitsemansa datan syötteestä (kuva 5), yleensä joko componentDidMount tai componentDidUpdate-metodia käyttäen. ComponentDidMount-metodia käytetään, kun sen sisältävää ohjelmakomponenttia käytetään

ensimmäisen kerran. ComponentDidUpdate-metodia käytetään, kun ohjelmakomponenttia käytetään useammin kuin kerran ja komponentissa olevia tietoja halutaan päivittää. (React 2020.) Komponenttiin propseina lähetetyn sivun id:n avulla tarkistetaan, onko id muuttunut. Jos id on muuttunut, suoritetaan componenDidUpdate-metodi.

ComponenDidUpdate-metodi on melkein identtinen koodin osalta componenDidMount-metodin kanssa. Ainoa ero näissä on hakuosoitteen muuttunut sivun id sekä if-lause, jossa propsien vertailu tehdään. Eniten componentDidUpdate-metodia käyttävä komponentti on Basic-komponentti. Kuvassa 5 ja 6 olevaa basic-komponenttia käytin ohjelman toteutuksessa sivuille, joiden tietoihin kuuluvat otsikko, sisältö ja kuvat.

Kuvassa 5 on näytetty, kuinka ohjelmassa käytetyn basic-komponentin tiedonhaku toteutetaan. Komponenttiin tulee ensimmäisenä konstruktori propsien kanssa. Sen sisälle määritellään muuttujat tyhjillä tiedoilla ja isLoading boolean-arvolla, ettei sitä ole suoritettu. Sen jälkeen luodaan componentDidMount-metodi, jossa ensimmäisenä laitetaan fetch-komennon sisälle osoite, mistä tietoa haetaan. Osoitteessa on propseina valitun joukkueen id ja sivu id. Sen jälkeen haetaan sisältö JSON-muodossa. Sitten määritellään mitä haetulle datalle tehdään ja this.setState kohdassa määritellään muuttujiin haettu tieto sekä muutetaan, että latausta ei enää suoriteta. Catch tulostaa virheilmoituksen konsoliin, jos jokin tiedonhaussa epäonnistuu. Samaa logiikkaa käytin useille eri komponenteille vaihtamalla hakuosoitetta sekä this.setState-tallennuskohdassa mitä tietoja tarvitaan tallentaa muuttujiin.

```
class Basic extends React.Component {
    constructor(props) {
        super(props);
        this.state = {
            isLoading: true,
            title: null,
            body: null,
            img: null,
        };
    }

    componentDidMount() {
        fetch('https://api.sportti.org/sites/' + this.props.route.params.team_id + '/' + this.props.route.params.page_id)
            .then((response) => response.json())
            .then((data) => {
                this.setState({
                    isLoading: false,
                    title: data.title,
                    body: data.body,
                    img: data.img,
                })
            })
            .catch((error) => {
                console.log(error)
            });
    }
```

Kuva 5. Datan haun esimerkki.

Kuva 6 sisältää esimerkin haetun tiedon esittämisestä. Tietojen esittäminen hoidetaan render-funktiossa. Aluksi tarkistetaan, onko tietojen hakeminen valmis. Jos ei ole, ruudulla pyörii Loading-komponentti.

```
    render() {
        if (this.state.isLoading) {
            return (<Loading />);
        } else {
            let title = this.state.title;
            let images = this.state.img != null ? this.state.img.map((val, key) => {
                return <View key={key}>
                    <Image style={styles.news_img} source={{ uri: val[0][0] }} />
                </View>
            }) : null;
            let content = this.state.body != null ? this.state.body : "Tietoja ei ole saatavilla.";

            return (
                <View style={styles.container}>
                    <ScrollView>
                        <Text style={[styles.toptitle, { backgroundColor: color }]}>{title}</Text>
                        <View style={[styles.main]}>
                            {images}
                            <HTML source={{ html: content }} />
                        </View>
                    </ScrollView>
                </View>
            );
        }
    }
}
```

Kuva 6. Tiedon esittäminen.

Kun tiedonhaku on suoritettu, tallennetaan halutut tiedot omiin muuttujiin. Kuvissa ja sisältömuuttujassa tarkastetaan, onko niillä saatavilla tietoja. Jos kuvia löytyy, käytetään map-metodia, jonka avulla käydään läpi kaikki taulukossa olevat kuvat ja tulostetaan ne. Sisältömuuttuja hyödyntää tulostuksessa react-native-render-

html-kirjastoa, joka mahdollistaa HTML-tageja sisältävän tekstin tulostuksen ottamaan huomioon käytetyt tagit.

6.3 Valitun joukkueen tallentaminen

Valitun joukkueen tietojen tallennukseen käytin kirjastoa nimeltä Async storage. Se mahdollistaa tiedon tallentamisen React Nativella luotuihin sovelluksiin ja tiedot pysyvät tallessa, vaikka ohjelma sammutettaisiin. Tietoja voidaan Async storagen avulla esimerkiksi tallentaa, lukea ja poistaa. (Async Storage 2020.)

Valitusta joukkueesta tallennetaan joukkueen id, nimi, pääväri ja logon osoite (kuva 7). Kun tiedot on tallennettu, suoritetaan käyttäjän siirtäminen Header-komponenttiin (kuva 8). Vaikka nimi ei sitä suoraan kerro, tämä tarkoittaa joukkueen etusivua, jossa on yhdistetty yläpalkki ja joukkueen oma etusivu.

```
saveData = async (id, name, logo, color) => {
    try {
        const teamid = JSON.stringify(id)
        await AsyncStorage.setItem('id', teamid)
        await AsyncStorage.setItem('name', name)
        await AsyncStorage.setItem('logo', logo)
        await AsyncStorage.setItem('color', color)
    } catch (error) {
        console.log(error)
    }
}
```

Kuva 7. Valitun joukkueen tietojen tallennus.

```
this.props.navigation.navigate('Header',
{ id: val.id, name: val.name, logo: url + val.img + "." + format, color: val.colors[0] })
```

Kuva 8. Käyttäjän siirtäminen tallennuksen jälkeen joukkueen etusivulle mukanaan joukkueen tiedot.

Tallennetun joukkueen id tarkistetaan aina, kun käyttäjä avaa ohjelman. Jos id löytyy, haetaan tallennetut tiedot AsyncStorage-kirjastosta ja käyttäjä siirtyy valitun joukkueen -etusivulle näiden tietojen kanssa. Jos id:tä ei ole, siirretään käyttäjä valitse joukkue -sivulle. Ohjelmassa on myös erillinen komponentti, jonka avulla poistetaan valittu joukkue ohjelman muistista.

6.4 Ohjelmassa navigointi

Ohjelman navigointi on toteutettu yhdistämällä stack ja drawer navigatorin ominaisuudet. Stack navigator sisältää kaksi komponenttia, jotka ovat Home ja Header (kuva 9). Home-komponenttia käytetään ainoastaan valitse joukkue -sivulla. Header-komponentti on käytössä kaikilla muilla sivuilla paitsi valitse joukkue -sivulla. Stack navigatorin luomaa yläpalkkia on hyödynnetty lisäämällä Header-komponenttiin oikealle nappi, jolla hallitaan mobiilivalikon aukaisemista. Napin ikoni haetaan hyödyntämällä Expon tarjoamaa vector-icons-kirjastoa.

```
createHomeStack = () => {
    return <Stack.Navigator>
        <Stack.Screen name="Valitse joukkue:" component={Home} options={{ headerTitleAlign: 'center', }} />
        <Stack.Screen name='Header' children={this.createDrawer} options={({ navigation }) => ({
            headerRight: () => (<TouchableOpacity style={{ paddingRight: 10 }}>
                <Feather name='menu' size={30} color='white' onPress={() => navigation.dispatch(DrawerActions.toggleDrawer())} /></TouchableOpacity>)
        })} />
    </Stack.Navigator>
}
```

Kuva 9. Stack navigator kokonaisuudessaan.

Yläpalkin keskelle tulee valitun joukkueen nimi sekä vasemmalle valitun joukkueen logo. Yläpalkin ja mobiilivalikon taustavärit muuttuvat tallennetun joukkueen päävärin mukaan. Nämä ulkoasuominaisuudet on määritelty DrawerContent-komponentissa, ja se hakee tiedot TeamHome-komponenttiin määriteltyjen propsien kautta. Propsit ovat Header-komponentin lähettämät, jonka lapsi drawer navigator on kokonaisuudessaan (kuva 10). Mobiilivalikkoon haetaan sivujen nimet DrawerContent-komponentissa menu-hakunimeä hyödyntämällä.

```
createDrawer = (props) => {
    return <Drawer.Navigator drawerStyle={{ backgroundColor: null, width: '100%' }}
        drawerPosition="right" drawerContent={props => <DrawerContent{...props} />}>
        <Drawer.Screen name='Etusivu' component={TeamHome}
            initialParams={{ id: props.route.params.id, name: props.route.params.name, logo: props.route.params.logo, color: props.route.params.color }} />
        <Drawer.Screen name='Pelaajat' component={Players} />
        <Drawer.Screen name='PelaajaProfiili' component={PlayerProfile} initialParams={{ profile_img: props.route.params.profile_img }} />
        <Drawer.Screen name='Ottelut' component={Games} />
        <Drawer.Screen name='Kumppanit' component={Partners} />
        <Drawer.Screen name='Sivu' component={Basic} />
        <Drawer.Screen name='Uutiset' component={News} />
        <Drawer.Screen name='Poista tallennettu joukkue' component={Delete} />
    </Drawer.Navigator>
}
```

Kuva 10. Drawer navigator kokonaisuudessaan.

DrawerContent sisältää funktion, joka suoritetaan melkein aina, kun jotain sivua on klikattu. Funktioon lähetetään sivun id, jonka avulla muutetaan osoitetta, jotta voidaan hakea mikä on klikatun sivun sivupohja. Sen jälkeen tarkistetaan if-

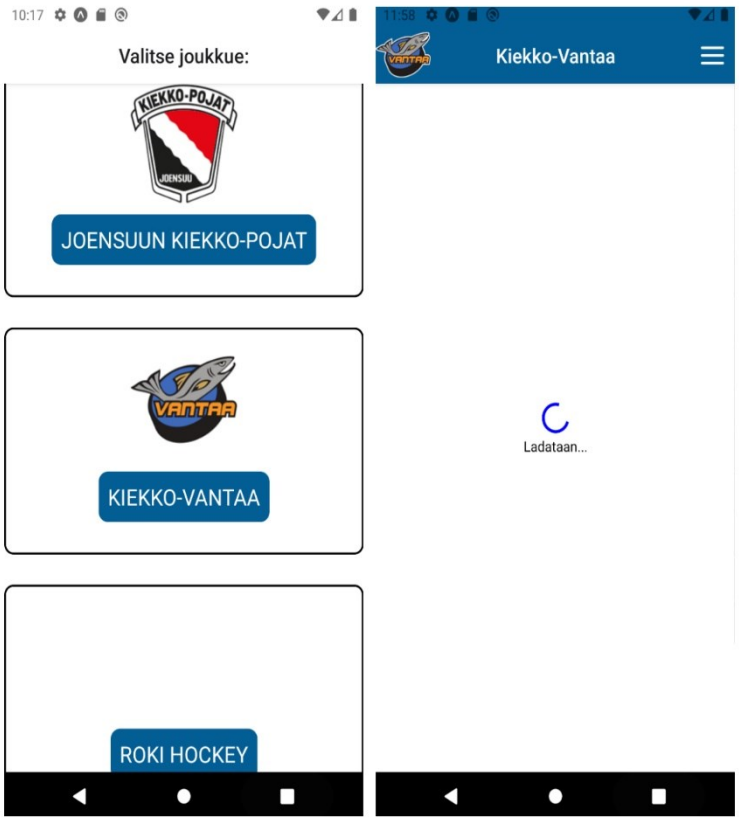
lauseessa, löytyykö kyseistä sivupohjaa listattuna drawer navigatorin näyttöjen joukosta nimen perusteella. Jos löytyy, siirretään käyttäjä näytön sisältämään komponenttiin mukanaan joukkueen id ja sivun oma id, joiden avulla muutetaan hakuosoitetta. Tästä tapahtumasta poikkeuksena on valittu joukkue -sivu, joka on lisätty oletuksena valikkoon, jonka klikkauksen jälkeen käyttäjä siirretään suoraan Delete-komponenttiin.

7 Tulokset

Opinnäytetyön toiminnallisen osuuden lopputuloksena sain luotua toimeksiantajan vaatimuksien mukaisen hybridisovelluksen, joka toimii Android- ja IOS-käyttöjärjestelmillä. Toimeksiannon toteutus onnistui ajallaan joulukuun loppuun mennessä.

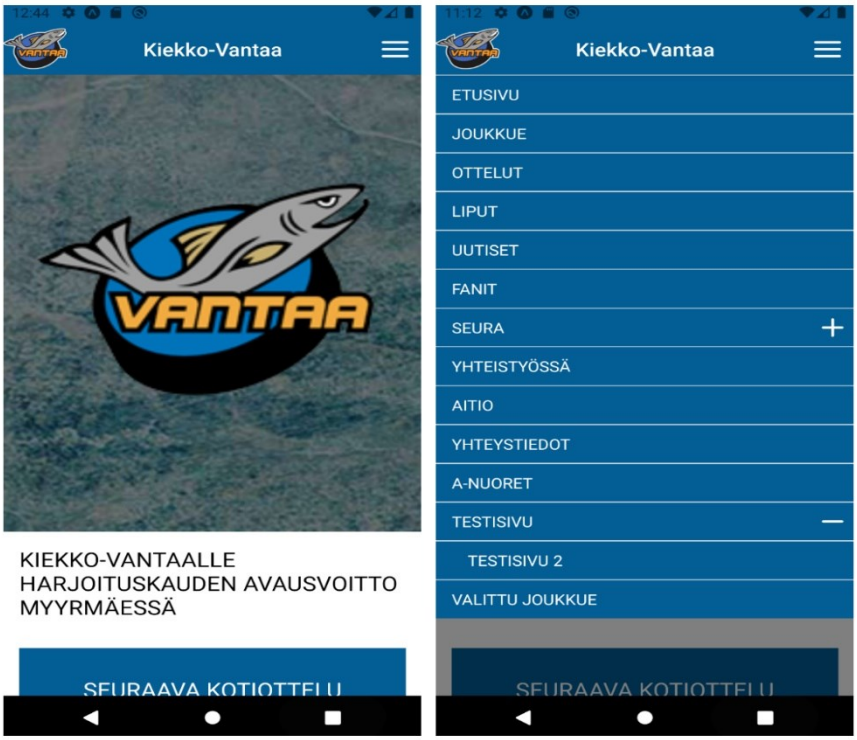
Ohjelma sisältää tällä hetkellä Kiekko-Vantaa-joukkueen tiedot melkein kokonaan. Muilla joukkueilla syötteen tiedot tämän kirjoitushetkellä ovat vielä keskeneräiset. Niiltä osin mitä tiedonhakua pystyin testaamaan, toimii hakeminen myös muiden joukkueiden osalta. Toimeksiantajan tarkoitus on päivittää syötteen tiedot muiden joukkueiden osalta myöhemmin.

Kuva 11 esittää ohjelman valitse joukkue -sivun, jossa näkyvät niiden kolmen Mestis-joukkueen tiedot, joiden tiedot olivat minun tehtävänäni laittaa toimintaan. Roki hockey -joukkueelta puuttuu logo, mutta se päivittyy syötteeseen myöhemmin. Samassa kuvassa on myös kuva ladataan-ilmoituksesta, joka näytetään aina, kun tietoja ollaan hakemassa syötteestä.



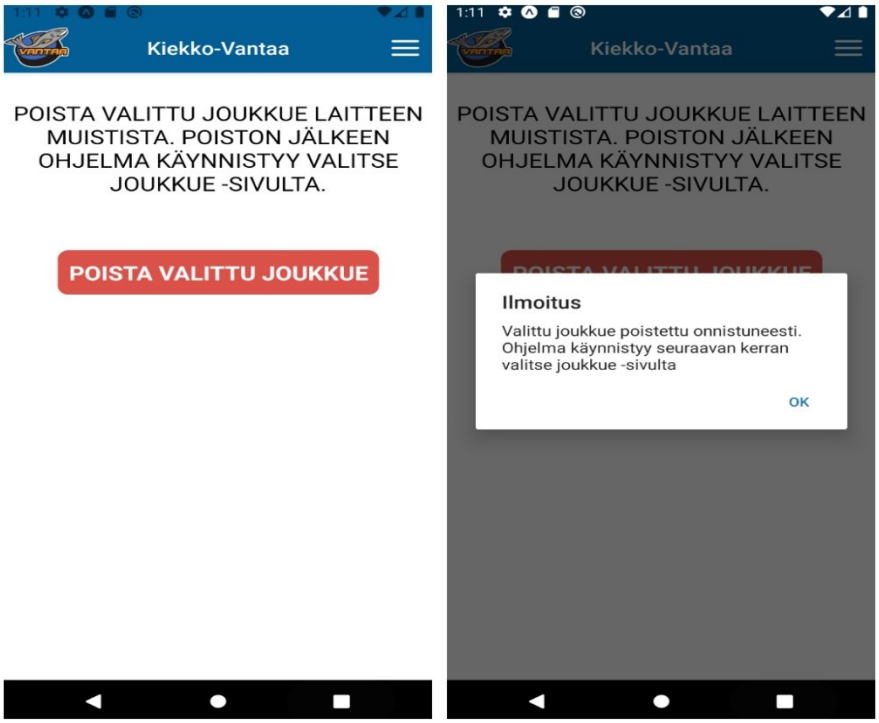
Kuva 11. Ohjelman aloitussivu ja tietojen hakemisen aikana näytettävä ladataan -ilmoitus.

Kun joukkue on valittu, siirrytään joukkueen omalle etusivulle. Kuva 12 sisältää esimerkin, kuinka Stack Navigatorin luomaa yläpalkkia hyödynnetään valitun joukkueen logon, nimen, valikkoikonin ja päävärin osalta. Joukkueen pääväri on myös laitettu kuvassa osittain näkyvän seuraava kotiottelu -laatikon sekä mobiilivalikon taustaväriksi. Mobiilivalikko avataan yläpalkin oikealla laidalla olevaa ikonia painamalla. Valikko tulee esiin animaationa oikealta vasemmalle. Valikossa näkyvät testisivut tulevat häviämään lopulliseen versioon. Toimeksiantaja lisäsi nämä pyynnöstäni syötteeseen, jotta pystyin testaamaan mobiilivalikon alavalikkojen toiminnan, jos niitä on useampia. Mobiilivalikossa voi olla auki yksi alavalikko kerrallaan.



Kuva 12. Valitun joukkueen etusivu sekä mobiilivalikko.

Kuva 13 sisältää valittu joukkue -sivun näkymän. Poista valittu joukkue -nappia painamalla poistetaan tallennetun joukkueen id muistista. Tämän jälkeen käyttäjälle tulee kuvassa näkyvä ilmoitus.



Kuva 13. Valitun joukkueen poistaminen.

Käyttäjä voi poistaa ilmoituksen painamalla OK. Tämän jälkeen ohjelma pitää käynnistää uudelleen, jotta käyttäjä pääsee valitse joukkue -sivulle. Tässä olisi yksi mahdollinen jatkokehitysominaisuus, jossa ohjelma käynnistyisi automaattisesti uudelleen OK-napin painamisen jälkeen.

8 Pohdinta

Opinnäytetyö onnistui suunnitelmien mukaisesti ja sain toteutettua ohjelman, joka täytti toimeksiantajan vaatimukset sovitussa aikataulussa. Kehitykseen valitut työkalut toimivat koko kehityksen ajan ilman ongelmia ja ne oli helppo laittaa toimimaan. React Native sopi mainiosti ohjelman sovelluskehykseksi, koska sillä sai melko vaivattomasti luotua ohjelmaan tarvittavat ominaisuudet. React Nativesta oli myös helppo löytää tietoa Googlea hyödyntäen kehityksessä ilmenneisiin ongelmiin sekä ominaisuuksien toteutukseen.

Opin opinnäytetyöprosessin myötä yleisesti paljon tietoa mobiilisovelluksien kehityksestä sekä kuinka luoda hybridisovellus React Nativea hyödyntäen. Tiedonhaku ja asiatekstin kirjoittamistaitoni parantuivat myös prosessin myötä. Ymmärsin myös, kuinka tärkeää on kommunikointi toimeksiantajan kanssa tällaisessa projektissa. Koin, että haastavinta opinnäytetyön teossa oli raportin kirjoittaminen sekä kokonaan uuden sovelluskehyksen opettelu.

Jos aloittaisin nyt opinnäytetyöprosessin uusiksi, muuttaisin muutamia asioita. Ensimmäisenä asiana järjestäisin toimeksiantajan kanssa palaverin kehitysprosessin alusta alkaen kerran viikossa. Alkuun luulin, että se ei ole tarpeellista, koska koin, että vaatimukset ovat sen verran selkeät, että pystyn ne toteuttamaan ilman useampaa palaveria. Huomasin kuitenkin, että palavereissa sain paremmin palautetta toimeksiantajalta projektin tilanteesta sekä paremman käsityksen myös itse projektista kokonaisuudessaan. Toimeksiantajalta tuli myös rakentavaa palautetta liittyen tekemääni koodin. Palautteen avulla sain tehtyä koodista siistimpää sekä muutettua ominaisuuksien toteutusta järkevämmäksi. Palavereissa tuli myös hyvin ilmi, jos syötteeseen piti tehdä muutoksia. Uskoisin, että

aikaisemmin järjestetyt viikoittaiset palaverit olisi nopeuttanut kehitysprosessin valmistumista.

Toisena muutoksena dokumentoisin asioita ylös enemmän jo kehityksen aikana. Nyt menetelmien ja toimintojen raportointi jäi paljolti kehityksen jälkeen toteutetuksi. Tässä olisin voinut hyödyntää esimerkiksi kehitysblogin ylläpitoa, jonka avulla olisin voinut kirjoittaa raporttiin kehitykseen liittyvistä ominaisuuksista nopeammin. Vähempi kirjoittaminen aiheutti lopulta sen, että en pystynyt palauttamaan opinnäytetyöraporttia aikataulussa, jota olin alkuun tavoitellut.

Ohjelman jatkokehityksestä on keskusteltu toimeksiantajan kanssa. Ensinnäkin, kun syötteet on saatu kuntoon muidenkin joukkueiden osalta pitää varmistaa, että kaikki asiat toimivat oikein. Syötteestä voisi laittaa videoita ohjelmaan uutena ominaisuutena. Ohjelmaan voisi tehdä myös muutoksen, että käyttäjä ei pääse valitse joukkue -sivulle painamalla puhelimen takaisin-nappia. Tällä hetkellä tämä onnistuu, jos joukkue on valittu. Muutoksia on myös tulossa syötteen osoitteeseen, jolloin se ei käyttäisi tulevaisuudessa joukkueen id:tä vaan joukkueen nimeä. Syötteen tietoturvaa on tarkoitus kehittää muuttamalla esimerkiksi, että kaikki haut vaativat lupakoodin tietojen hakemiseen. Ulkoasun lopullinen versio tulee myös muuttumaan mukaillen joukkueiden verkkosivujen uusia teemoja.

Lähteet

Alessandria, S. 2020. Flutter Projects: A practical, project-based guide to building real-world cross-platform mobile applications and games. Birmingham: Packt Publishing Ltd. Näyte painetusta kirjasta. https://books.google.fi/books?id=junbDwAAQBAJ&lpg=PA51&dq=flutter%20mobile%20development&hl=fi&pg=PP1#v=onepage&q&f=false. 2.11.2020.

AltexSoft. 2019. The Good and the Bad of Ionic Mobile Development. https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-ionic-mobile-development/. 12.1.2021.

Async Storage. 2020. Async Storage. https://react-native-async-storage.github.io/async-storage/. 25.11.2020.

Avendano, A. 2019. Why you should not build hybrid apps in 2020. Bluefletch. https://bluefletch.com/avoid-hybrid-apps-in-2020/. 12.1.2021.

Bellinaso, M. 2018. Flutter: the good, the bad and the ugly. Medium. https://medium.com/asos-techblog/flutter-vs-react-native-for-ios-android-app-development-c41b4e038db9. 6.11.2020.

Biessek, A. 2019. Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter and Dart 2. Birmingham: Packt Publishing Ltd. Näyte painetusta kirjasta. https://books.google.fi/books?id=pF6vDwAAQBAJ&lpg=PP1&dq=flutter%20for%20beginners&hl=fi&pg=PP1#v=onepage&q=flutter%20for%20beginners&f=false. 2.11.2020.

Cook, J. 2020. How the React Native bridge works and how it will change in the near future. Dev. https://dev.to/wjimmycook/how-the-react-native-bridge-works-and-how-it-will-change-in-the-near-future-4ekc. 16.11.2020.

DeBeasi, L. 2020. Announcing Ionic Vue. Ionic. https://blog.ionicframework.com/announcing-ionic-vue/. 2.11.2020.

Expo. 2020. Installation. https://docs.expo.io/get-started/installation/. 8.12.2020.

Flutter. 2020a. How does Flutter run my code on Android?. https://flutter.dev/docs/resources/faq#run-android. 16.11.2020.

Flutter. 2020b. Showcase. https://flutter.dev/showcase. 2.11.2020.

Flutter. 2021a. Writing custom platform-specific code. https://flutter.dev/docs/development/platform-integration/platform-channels?tab=android-channel-java-tab. 18.1.2021.

Griffith, C. 2017. Mobile App Development with Ionic, Revised Edition: Cross-Platform Apps with Ionic, Angular, and Cordova. Sebastopol: O'Reilly Media, Inc. Näyte painetusta kirjasta. https://books.google.fi/books?id=x9MxDwAAQBAJ&lpg=PR2&dq=Ionic%20mobile%20development&lr&pg=PR1#v=onepage&q&f=false. 30.10.2020.

Ionic. 2020a. Customers. https://ionicframework.com/customers.30.10.2020.

Ionic. 2020b. Components. https://ionicframework.com/docs/components. 30.10.2020.

Khanna, R. & Yusuf, S. & Phan, H. 2017. Ionic : Hybrid Mobile App Development. Birmingham: Packt Publishing Ltd. Näyte painetusta kirjasta. https://books.google.fi/books?id=Dng5DwAAQBAJ&lpg=PP1&dq=Ionic%20mobile%20development&hl=fi&pg=PP1#v=onepage&q&f=false. 30.10.2020.

Leler, W. 2017. What's Revolutionary about Flutter. Hackernoon. https://hackernoon.com/whats-revolutionary-about-flutter-946915b09514. 12.2.2021.

Masiello, E. & Friedmann J. 2017. Mastering React Native. Birmingham: Packt Publishing Ltd. Näyte painetusta kirjasta. https://books.google.fi/books?id=A1QoDwAAQBAJ&lpg=PP1&hl=fi&pg=PP1#v=onepage&q&f=false. 27.10.2020.

Mclaren, N. 2020. React Navigation 5: Stack, Tab, and Drawer All in One. Medium. https://medium.com/better-programming/react-navigation-5-stack-tab-drawer-all-in-one-ead723188056. 8.12.2020.

Netkow, M. 2021. Ionic. https://ionicframework.com/resources/articles/ionic-vs-flutter-comparison-guide. 20.1.2021.

Novick, V. 2017. React Native - Building Mobile Apps with JavaScript. Birmingham: Packt Publishing Ltd. Näyte painetusta kirjasta. https://books.google.fi/books?id=YJZGDwAAQBAJ&lpg=PP1&dq=react%20native&lr&hl=fi&pg=PA10#v=onepage&q=react%20native&f=false. 26.10.2020.

O'Dea, S. 2019. Number of smartphone users worldwide from 2016 to 2021. Statista. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/. 5.1.2021.

Patel, P. 2018. https://www.freecodecamp.org/news/what-exactly-is-node-js-ae36e97449f5/. 1.2.2021.

PCMag. 2021. APK. https://www.pcmag.com/encyclopedia/term/apk. 1.2.2021.

React. 2020. React.Component. https://reactjs.org/docs/react-component.html. 23.11.2020.

React Native. 2020a. Showcase. https://reactnative.dev/showcase. 26.10.2020.

React Native. 2020b. Introduction. https://reactnative.dev/docs/getting-started. 26.10.2020.

React Native. 2021a. JavaScript Environment. https://reactnative.dev/docs/javascript-environment. 22.1.2021.

W3Schools. 2021. What is JSON?. https://www.w3schools.com/whatis/whatis_json.asp. 22.1.2021.