TAMPEREEN
AMMATTIKORKEAKOULU

BEEVELO

Mobiilisovelluksen kehittäminen

Rami Kallio

Opinnäytetyö
Joulukuu 2018
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

KALLIO RAMI:
Beevelo
Mobiilisovelluksen kehittäminen

Opinnäytetyö 20 sivua
Joulukuu 2018

Beevelo on kuntoilusovelluksen tyylinen ohjelma älypuhelimille, joka seuraa kuljettua matkaa ja palkitsee liikuntaa pisteillä. Sovelluksen pääkohdeyleisö ei kuitenkaan ole kuntoilijat, vaan työntekijät ja opiskelijat jotka matkustavat työpaikalle ja kouluun joka päivä. Projektin taustalla on idea innostaa heitä pyöräilemään ja kävelemään, autolla liikkumisen sijaan, tarjoamalla kerättyjä pisteitä vastaan alennuskuponkeja läheisiin liikkeisiin.

Tässä työssä käsitellään vain projektin mobiilisovelluksen suunnittelua ja toteutusta. Aluksi käsitellään sovellukselle asetettavat vaatimukset, niiden tuottamat ongelmat ja niihin valitut ratkaisut.

Lopuksi käsitellään käytännön ratkaisuja esimerkein, projektin kulkua ja sen aikana ilmenneitä haasteita.

Koska opinnäytetyö sisältää salassa pidettävää materiaalia, osa sisällöstä on poistettu julkisesta versiosta.

Asiasanat: ohjelmistokehitys, mobiili, älypuhelinsovellus

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Computer Systems Engineering
Software Engineering

KALLIO RAMI:
Beevelo
Mobile application development

Bachelor's thesis 20 pages
December 2018

Beevelo is fitness app like application for mobile devices. It records covered distances and awards points for outdoor activities such as walking and cycling. The intended audience for the app are workers and students who travel to work or school every day. The idea behind the project is to encourage and reward walking and cycling with prizes, such as discounts in stores.

Only the mobile application designs are covered in this thesis.
First part consists of application requirements, problems these requirements can cause and chosen solutions to these problems.
In the last part I will go through some example solutions, project flow and challenges that had to be overcome.

This thesis contains confidential information, which has been removed from the public version.

Key words: software development, mobile, smartphone application

SISÄLLYS

LYHENTEET JA TERMIT

AngularJS	Angular JavaScript, JavaScriptiin pohjautuva ohjelmistokehys
App / appi	Application, sovellus/ohjelma
Cross-platform	Alusta-/järjestelmäriippumaton, eli pystytään käyttämään usealla eri alustalla
CSS	Cascading Style Sheets, muotoilukieli merkintäkielille kts. HTML
GPS	Global positioning system, satelliittipaikannus
HTML	Hyper Text Markup Language, merkintäkieli websivujen rakentamiseen
JS	JavaScript, skriptikieli
Multiplatform	kts. Cross-platform
MVC	Model – view – controller, arkkitehtuurimalli
TAMK	Tampereen ammattikorkeakoulu
Wrapper	Käärin, luokka jonka tarkoitus on esim. tulkata toimintoja toisesta kirjastosta tai kielestä

1 JOHDANTO

Beevelo on kuntoilusovelluksen tyylinen ohjelma älypuhelimille, joka kerää paikallistamisdataa käyttäjästä. Kerätyistä matkoista lasketaan pisteitä käyttäjälle, joita voidaan käyttää sovelluksessa. Pääkohdeyleisönä ei kuitenkaan ole kuntoilijat, vaan työntekijät ja opiskelijat jotka matkustavat työpaikalle/kouluun joka päivä. Projektin taustalla on idea innostaa heitä pyöräilemään ja kävelemään, autolla liikkumisen sijaan, tarjoamalla kerättyjä pisteitä vastaan palkintoja. Palkintoina voi olla esimerkiksi alennuskuponkeja läheisiin liikkeisiin, joiden tuotteista ko. henkilö olisi kiinnostunut, jolloin tämä hyödyttäisi myös yrityksiä.

Aluksi tässä työssä käsitellään mobiilisovelluksen suunnittelua ja tutkitaan mahdollisia teknisiä ratkaisuja esitettyyn ideaan. Koska projektin tarkoituksena on saada mahdollisimman moni innostumaan liikkumisesta, ensimmäisenä tulisi valita mille alustalle kehittäminen aloitetaan. Valtaosalla ihmisistä on nykypäivänä Android tai iOS-älypuhelin ja tavoitteena on saada sovellus molemmille alustoille. Kääntäminen toiselta alustalta toiselle ei toimi, koska molemmat käyttävät omia ohjelmointikieliään ja -rajapintojaan. Samaan aikaan molemmille alustoille kehittäminen natiivisti ei ole ideaalista eikä taloudellista. Tähän ongelmaan on kehitetty useita teknisiä ratkaisuja, jotka hyödyntävät molemmilla alustoilla toimivaa webtekniikkaa. Tähän projektiin valittiin käyttöön kokeneemman ohjelmistokehittäjän suosituksesta Cordova ja Ionic, jotka olivat kehittämisen aloittamisen aikaan parhaat vaihtoehdot.

Lopuksi käsitellään käytännön ratkaisuja esimerkein, projektin kulkua, sen aikana ilmenneitä ongelmia ja mahdollisia ratkaisuja niihin.

2 MÄÄRITTELY JA SUUNNITTELU

2.1 Tarve

Työn tilaaja halusi kuntoilusovelluksen tyylisen ohjelman, joka pystyisi keräämään paikallistamisdataa (GPS) käyttäjästä ja lähettämään sen palvelimelle käsiteltäväksi. Tästä datasta laskettaisiin pisteitä käyttäjälle, joita hän pystyisi käyttämään sovelluksen kaupassa palkintoihin.

2.2 Vaatimukset

Mobiilisovellukselle asetetaan seuraavat vaatimukset:

- Toimii yleisimmillä mobiilikäyttöjärjestelmillä
    - Android
    - iOS
- Käyttää laitteen GPS ja verkkoyhteyttä
    - Kykenee toimimaan myös katve alueella
- Käyttää laitteen sisäistä muistia ja muistikorttia
- Lähettää datan palvelimelle

2.3 Tekniset ratkaisut

Ensimmäisenä ja tärkeimpänä on valita alusta, jolle kehitys tapahtuu. Tämä määrittelee käytettävät kielet ja rajapinnat, joita myöhemmin tullaan käyttämään sekä projektin kustannukset. Projektin vaatimuksessa on määritelty toiminta kolmella eri alustalla, jotka eivät ole ristiin yhteensopivia vaan jokainen pitäisi kehittää erikseen käyttämällä niiden omia työkaluja, kieliä ja rajapintoja.

2.3.1 Mobiilisovellustyypit

Natiivilla (Native App) tarkoitetaan tietylle alustalle toteutettua sovellusta käyttämällä sen omia ohjelmointikieliä ja -rajapintoja. Tämä on paras vaihtoehto, jos tarkoituksena on kehittää sovellus vain yhdelle tietylle alustalle. Kehitettäessä usealle alustalle samaa sovellusta, tulee jokaisen alustan kielet ja rajapinnat opetella erikseen. Pahimmassa tapauksessa tiettyjä ominaisuuksia ei välttämättä voida toteuttaa toisella alustalla. (Imaginovation 2018.)

Web-sovellus (Web App) on verkkosivun kaltainen sovellus, jota käytetään alustan selaimesta. Mikä sitten erottaa verkkosivun ja -sovelluksen? Verkkosivut sisältävät puhtaasti tietoa, eivätkä tarjoa kuin minimaalisen vuorovaikutuksen käyttäjille esimerkiksi kommentointi mahdollisuuden uutisartikkeliin. Verkkosovellukset taas vaativat käyttäjän toimia tehdäkseen jotain. Hyvänä esimerkkinä verkkosivusta mainittakoon Aamulehti ja verkkosovelluksesta Facebook. (Imaginovation 2018.)

Hybridi (Hybrid App) yhdistää natiivin ja websovelluksen toimintoja toteutuksessaan. Luotu ”verkkosivu” paketoidaan alustan natiiviksi sovellukseksi ja tuodaan esille natiiviin verkkonäkymään (esim. Android webview). Tämä mahdollistaa yhden sovelluksen kehittämisen usealle alustalle, koska paketointi tapahtuu aina alustan omilla kehitystyökaluilla (Software Development Kit). Suurin heikkous hybridissä on sen hitaus ja käyttöliittymän ulkoasu verrattuna natiivisesti kehitettyyn sovellukseen. Lisäksi jokaiselle alustalle kääntäminen pitää tehdä kehitystyökalujen asettamien rajoitteiden mukaisesti. Esimerkiksi Windows-ympäristössä voidaan kääntää vain Windows Phonelle ja Androidille, mutta ei iPhonelle joka vaatii Mac-tietokoneen. Tästä johtuen internettiin on alkanut ilmestyä palveluita, jotka kääntävät koodin tarvittavalle alustalle pientä kustannusta vastaan. (Imaginovation 2018.)

2.3.2 Apache Cordova

Apache Cordova on mobiilialustoille suunnattu avoimen lähdekoodin ohjelmistokehys, jolla voidaan kehittää alustariippumattomia sovelluksia (Hybrid App) hyödyntämällä webkieliä, kuten HTML5, CSS3 ja JavaScript. (Apache Cordova 2018.)

Cordovan yksi suurimmista vahvuuksista on sen laajennettavuus lisäosilla (plugin). Jokainen lisäosa toteutetaan halutun alustan tai alustojen natiivilla kielellä. Tämä mahdollistaa pääsyn normaalin webympäristön asettamista rajoitteista laitteen syvempiin toimintoihin, kuten GPS, kamera tai kiihdytysanturi.

2.3.3 Ionic framework v1

Ionic 1 on avoimen lähdekoodin kehitystyökalupaketti hybridi ohjelmistojen kehittämiseen mobiliilaitteille, joka on rakennettu Cordovan ja AngularJS päälle. Se keskittyy pääasiassa UI-rajapinnan toimintojen parantamiseen, mutta sisältää työkalut nopeaan testaukseen. (Drifty co. 2018.)

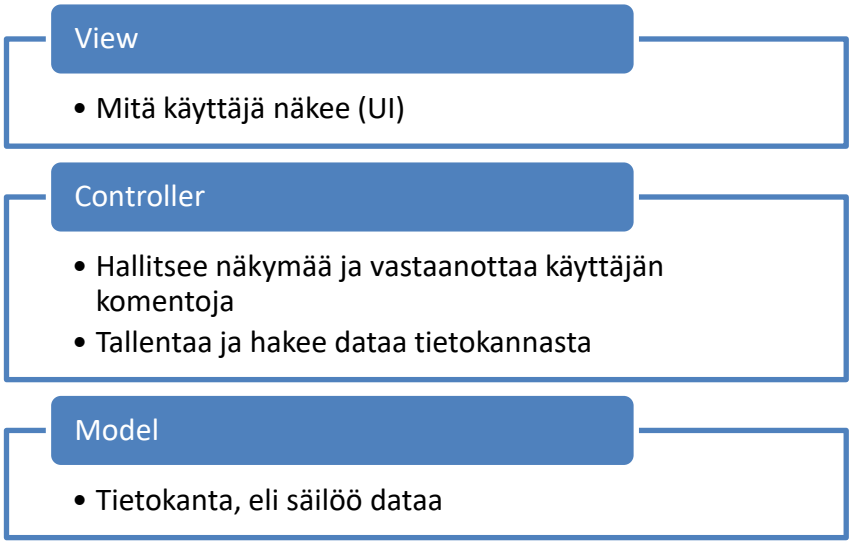
Koska kyseessä olevat ohjelmistot ovat hybridejä, pohjalta löytyy aina webtekniikoita (kts. Mobiilisovellustyypit). Tästä syystä testausta voidaan helposti tehdä esim. kehitystietokoneella ajamalla lähdekoodia webpalvelimella. Tälle löytyy valmis toiminto Ionic:n työkaluista, jolla se luo lokaalin webpalvelin kehitystietokoneelle. Tämä on paljon nopeampaa verrattuna koodin kääntämiseen ja ajamiseen esim. älypuhelimelle. Valitettavasti on mahdollista, että osa sovelluksen ominaisuuksista eivät toimi verkkoselaimessa ja ne täytyy testata kohdelaitteilla.

Yksi Ionic:n testipalvelimen ominaisuuksista on LiveReload-tila. Tämä toiminto tarkkailee projektin lähdekoodissa tapahtuvia muutoksia ja päivittää testipalvelimelle automaattisesti viimeisimmät muutokset lähdekoodiin. Tämä mahdollistaa nopean testauksen ja kehittämisen, koska muutokset tulevat näkyviin lähes heti.

2.3.4 Angular JavaScript

Angular JavaScript on Googlen kehittämä avoimen lähdekoodin JavaScript-ohjelmistokehys. Se on kehitetty helpottamaan yhden sivun sovelluksien (single-page application) kehittämistä. Yhden sivun sovellus on juuri sellainen miltä se kuulostaa. Kaikki toiminta tapahtuu yhdellä sivulla muuttamalla vain näkymää käyttäjän interaktion mukaan. Käytännössä kuitenkin ”sivuja”, tai tarkemmin sanottuna sivumalleja, on useampi ja niitä näytetään pääsivulla ohjelmistokehyksen omassa HTML-elementissä. (Guru99.)

AngularJS toteuttaa MVC-arkkitehtuurimallia, jonka tavoitteena on helpottaa sovelluksen osien samanaikaista kehittämistä ja lähdekoodin uudelleenkäyttöä hajauttamalla toimintaa eri osiin. Valitettavasti tämä myös vaikeuttaa lähdekoodien lukemista, koska niistä tulee hajautetumpia. MVC tulee sanoista model (malli), view (näkymä) ja controller (ohjain). Kuvassa 1 esitetty MVC-malli sisältää yksinkertaiset selitykset mitä kukin osa käytännössä tekee. (Guru99.)



Kuva 1. MVC-malli

2.3.5 ngCordova

ngCordova on Ionic tiimin luoma AngularJS wrapper -kokoelma yleisimmin käytetyille Cordovan lisäosille. Sen tarkoituksena on helpottaa lisäosien liittämistä Ionic projekteihin. Lisäosan kutsumisen sijaan lähdekoodissa kutsutaan AngularJS palvelua, joka vastaa lisäosan funktioiden käskyttämisestä asynkronisesti. Tämä mahdollistaa asynkronisten funktioiden kutsumisen synkronisessa järjestyksessä. (Morony J 2017.)

2.3.6 Sass

Syntatically awesome stylesheets (Sass) on esikääntäjälaajennus CSS-kieleen. Sass:n suurin vahvuus on sen kyky hyödyntää mm. muuttujia, funktioita ja matemaattisia operaatioita, joita normaalissa CSS-kielessä ei voida käyttää. (Sass 2018.)

Sass:lla voidaan valita kahdesta eri syntaksista, jotka voidaan kääntää laajennuksen työkalulla ristiin keskenään:

- SCSS, joka noudattaa CSS-tyyliä
- SASS käyttää sulkujen ja puolipisteiden sijaan sisennyksiä

Mitä tapahtuu käytännössä? Käännettäessä sovellusta esikääntäjä muuttaa Sass-tiedoston CSS muotoon, jolloin muuttujat sijoitetaan paikoilleen ja matemaattiset operaatiot lasketaan. Lähdekoodissa 1 on esitetty miltä SCSS näyttää ennen kääntämistä ja sen jälkeen, kun se on käännetty CSS muotoon. (Sass 2018.)

```
/* SCSS */
$text-color: #222222;
$padding: 10px;
$font: Arial;

p {
  font: 100% $font;
  color: $text-color;
}

body {
  padding: $padding /2;
}

h1 {
  font: 100% $font;
}


/* CSS */
p {
    font: 100% Arial;
    color: #222222;
}

body {
    padding: 5px;
}

h1 {
    font: 100% Arial;
}
```

Lähdekoodi 1. SCSS esimerkki

3 TOTEUTUS

Ensimmäisenä vaiheena toteutuksessa loin tabs-esimerkkiprojektin Ionic:llä, jolla opettelin ohjelmistokehysten toimintaa. Kaikki ohjelmointi tapahtui käyttämällä JavaScript ja HTML -kieliä, koska ohjelmistokehys laajensi AngularJS:iä. Seuraavaksi siirryin luomaan yksinkertaisen GPS-hakutoiminnon, jolla haettiin mobiililaitteen sijainti nappia painamalla ja tulostettiin saadut koordinaatit konsoliin. Tämän jälkeen aloin etsiä sopivaa lisäosaa mahdollistamaan GPS:n toiminnan, vaikka ohjelma olisi taustalla. Viimeistelynä tutkin teemojen käyttämisen mahdollisuutta.

Testaamiseen käytin apuna Chrome-selaimen kehittäjätyökaluja (Inspector). Varsinkin Androidilla testaaminen oli helppoa em. selaimen Android live debuggerin avulla, joka mahdollisti kehitettävän sovelluksen käytön laitteella selaimesta käsin.

3.1 Näkymien navigointi ja reititys

Navigointi tapahtuu käyttämällä Angularin UI Router:ia, joka hoitaa navigoinnin eri sivumallien välillä ja niiden sitomisen ohjainkomponentteihin (Lähdekoodi 2). Näiden näkymien välillä voidaan liikkua esimerkiksi käyttämällä $state.go() -komentoa ja antamalla sille tilan (state) nimi String muodossa.

```
.config(function($stateProvider, $urlRouterProvider) {
    $stateProvider
        .state('tabs', {
            url: "/tab",
            abstract: true,
            templateUrl: "templates/tabs.html"
        })
        .state('tabs.home', {
            url: "/home",
            views: {
                'home-tab': {
                    templateUrl: "templates/home.html",
                    controller: 'HomeTabCtrl'
                }
            }
        })
        .state('tabs.facts', {
            url: "/facts",
            views: {
                'home-tab': {
                    templateUrl: "templates/facts.html"
                }
            }
        });
        /* ... */
  $urlRouterProvider.otherwise("/tab/home");
});
```

Lähdekoodi 2. Angular UI router esimerkki (Ionic.)

3.2 Cordova Geolocation Plugin

Cordovan Geolocation-lisäosa käyttää laitteen GPS:iä paikantamisessa, mutta tarvittaessa kykenee hyödyntämään muita paikannusmenetelmiä apunaan. Lähdekoodissa 3 esitetyllä koodinpätkällä haetaan laitteen koordinaatit ja sijoitetaan ne muuttujiin.

```
module.controller('GeoCtrl', function($cordovaGeolocation) {
    var posOptions = {timeout: 10000, enableHighAccuracy: false};
    $cordovaGeolocation
        .getCurrentPosition(posOptions)
        .then(function (position) {
            var lat  = position.coords.latitude
            var long = position.coords.longitude
        }, function(err) {
            // error
        });
});
```

Lähdekoodi 3. Geolocation esimerkki (ngCordova 2018)

3.3 Cordova Plugin Background Geolocation

Geolocationin suurimpina ongelmina olivat sen akun kulutus ja toiminta ainoastaan, jos ko. sovellus on aktiivisena eikä taustalla. Sijainnin jatkuva kyseleminen GPS:ltä kuluttaa nopeasti puhelimen akkua, joka on todella suuri heikkous tämän kaltaisille sovelluksille.

Tähän löytyi ratkaisu Mauron85 kehittämästä Cordova Background Geolocation -lisäosasta, joka oli kuin tehty tälle projektille. Se sisälsi monia ominaisuuksia, joita projektissa kipeästä tarvittiin:

- Taustalla ajo
- Akun keston parantaminen
- Paikallistamistietojen lähettämisen suoraan palvelimelle
- Tietojen tallennus paikallisesti, jos lähettäminen epäonnistui

Varsinkin akun kulutuksen ongelma oli ratkaistu ovelasti, tekemällä GPS päivitykset vain silloin, kuin käyttäjän havaittiin liikkuvan. (Mauron85 2018.)

3.4 Listan luominen dynaamisesti

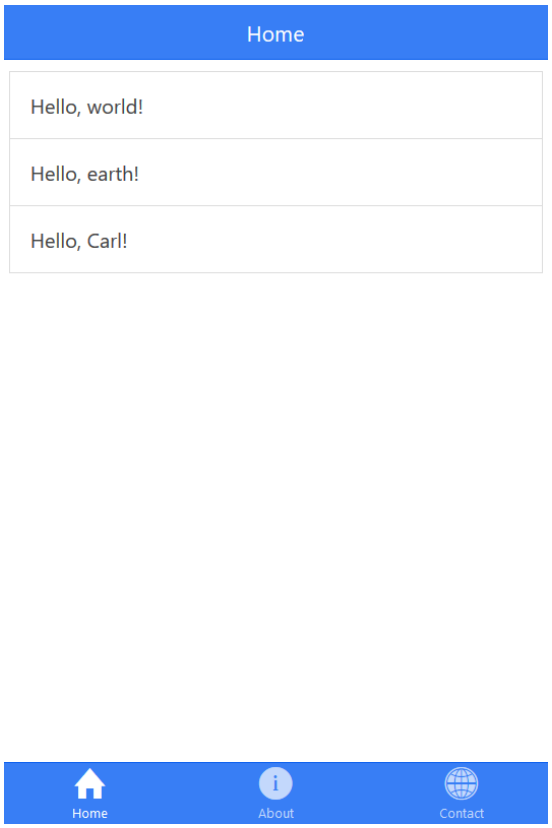
Jossain vaiheessa tulee tarve luoda sisältöä dynaamisesti, esimerkiksi palvelimelta saadun datan esittämiseksi. Tämä on yksinkertainen toteuttaa hyödyntämällä AngularJS:n ng-repeat -direktiiviä, joka luo silmukan avulla HTML-elementtejä saadusta taulukosta. Lähdekoodeissa 4 ja 5 on esitetty esimerkit dynaamisen listan luomiseen palvelimelta saadusta taulukosta. Kuvassa a on esitetty aikaan saatu tulos ohjelmassa.

```
<ion-list>
    <ion-item ng-repeat="item in items">
        Hello, {{item}}!
    </ion-item>
</ion-list>
```

Lähdekoodi 4. Dynaaminen lista esimerkki HTML

```
$scope.$on('$ionicView.beforeEnter', function() {
    $http.get(serverAddress, {timeout: 30000})
        .then(function (data) {
            console.log(data);
            $scope.items = data;
        }, function (error) {
            console.error("Error occurred: ", error);
        });
});
```

Lähdekoodi 5. Dynaaminen lista esimerkki JavaScriptillä



Kuva 2. Dynaamisesti luotu lista

3.5 Teemat

Yhtenä ominaisuutena tutkin mahdollisuutta muuttaa käyttäjärajapinnan teemoitusta. Ainoa toimiva ratkaisu oli luoda uusi CSS-tyyliviittaus JavaScriptillä sovelluksen käynnistyksen yhteydessä (Lähdekoodi 6).

```
.run(function($rootScope, $ionicPlatform, localStorageService) {
    // ...

    var theme = localStorageService.get('theme') || "";
    var pathToCss = "css/theme2.css";
    if(theme) {
        var fileref=document.createElement("link");
        fileref.setAttribute("rel", "stylesheet");
        fileref.setAttribute("type", "text/css");
        fileref.setAttribute("href", pathToCss);
        document.getElementsByTagName("head")[0].appendChild(fileref);
    }

    // ...
}
```

Lähdekoodi 6. Teeman muuttaminen sovelluksen käynnistyessä.

Ensimmäisenä haetaan sovellukseen tallennettu tieto teemasta (esim. nimi tai vastaava tunniste). Jos sitä ei ole olemassa, sijoitetaan muuttujaan tyhjä teksti jolloin if-ehto ei toteudu ja sovellus käynnistyy normaalisti vakio teemalla. Jos taas poikkeava teema on määritelty, tarvitaan relatiivinen polku (relative path) lisättävälle CSS-tiedostolle. Lopuksi elementti lisätään HTML headeriin, jossa kaikki CSS-tiedostojen määrittelyt sijaitsevat. Lopuksi tarvitsee tallentaa aiemmin mainittu tieto teemasta ja ladata sovellus uudestaan (Lähdekoodi 7.).

```
$scope.applyTheme = function(theme) {
    localStorageService.set("theme", theme);
    location.reload();
};
```

Lähdekoodi 7. Teeman asettaminen ja sivun uudelleen lataus.

Tällä tekniikalla pystytään vaikuttamaan vain CSS-muotoiluun, eli rakenteellisia muutoksia sillä ei juurikaan voida tehdä.

3.6 Latausruutu

Välillä datan saaminen palvelimelta kestää ja käyttäjälle tarvitsee näyttää latauksen olevan kesken. Tähän on parasta käyttää Ionic:n latausruutu ominaisuutta, joka estää käyttäjän toimet latauksen ajaksi. Lähdekoodi 6:ssa haetaan esimerkkinä palvelimelta dataa, jonka aikana ruudulla näkyy ”Loading…” teksti. Jos palvelin ei vastaa 30 sekunnin kuluessa, latausruutu piilotetaan ja tilalle tuodaan virheikkuna käyttämällä $ionicPopup.alert() -komentoa.

```
$scope.$on('$ionicView.beforeEnter', function() {

    $ionicLoading.show({template: "Loading..."});
    $http.get(ServerAddress, {timeout: 30000})
        .then(function (data) {
            console.log(data);
            $ionicLoading.hide();
        }, function (error) {
            console.error("Error occurred: ", error);
            $ionicLoading.hide();
            $ionicPopup.alert({
                title: 'Error',
                template: error.toString()
            });
        });
});
```

Lähdekoodi 8. Latausruutu esimerkki

4 POHDINTA

Aloitusvaiheessa projekti tuntui kokemattomalle kehittäjälle pieneltä ja nopealta toteuttaa. Myöhemmin tajusin vasta, kuinka olin haukannut liian ison palan nieltäväksi. Suurimmaksi haasteeksi kuitenkin muodostui kehittää samaan aikaan toimiva backend-palvelin, jota tässä työssä käytettiin autentikointiin ja sijaintidatan tallentamiseen. Onneksi sain apua näihin ongelmiin kokeneemmalta opiskelijalta, joka suositteli projektissa käytettyjä ohjelmistokehyksiä. Hänen ansiostaan pääsin hyvin alkuun ja sain toimivan prototyypin valmiiksi.

Omat ongelmansa toi kääntäminen iOS-järjestelmään, jota varten sain onneksi TAMK:lta lainaksi Mac-tietokoneen. Suurimmat vaikeudet tässä johtuivat kehitystyökalujen ja OS X:n versioiden ristiriidoista. Esimerkiksi jos halusi kehittää uudelle iOS versiolle, olisi pitänyt olla uusin versio OS X:stä joka ei olisi toiminut ko. koneella rautarajoitusten takia. Lisäksi jos olisi halunnut kehittää vanhemmille versioille, siitä pitänyt maksaa kehittäjä lisenssi joka oli useita satoja euroja.

Suurin työ tuli kuitenkin kommunikaatio virheestä sovelluksen ulkoasun kehittämisvaiheessa. Suunnittelutyö tilattiin kolmannelta osapuolelta yrityksen toimesta, joka toimitti nätin ja toimivan paketin. Valitettavasti tieto siitä, ettei sovelluksessa käytetty tiettyä arkkitehtuuria, välittynyt suunnittelijalle. Tämän huomasin vasta sen jälkeen, kun aloin implementoimaan muutoksia koodiin. Tässä vaiheessa se oli jo liian myöhäistä muuttaa, joten jouduin käsin tekemään kaikki muutokset. Onneksi lopputulos vastasi lähes kokonaan suunniteltua ulkonäköä, mutta aikaa olisi säästynyt paljon, jos tämä virhe olisi huomattu ajoissa.

Kokonaisuudessaan projekti toi paljon uutta näkökulmaa ja kokemusta ohjelmistokehittämiseen. Suurin virhe oli ehdottomasti projektin koon ja työmäärän aliarvioiminen. Kehitystyön tein pääasiassa yksin, joka oli myöskin virhe.

Jatkokehitystä varten ensimmäinen ratkaistava ongelma olisi päivittää lähdekoodit uudempaan versioon Ionic:lle tai siirtyä React Nativeen. Toinen ongelma on datan turvallisuus ja yksityisyys. Paikallistamis- ja käyttäjätiedot ovat arvokkaita nykymaailmassa niin rikollisille kuin muille yrityksillekin. Tämän lisäksi EU alueella keväällä 2018 voimaan

tullut GDPR-laki tulee vaikuttamaan ratkaisevasti järjestelmään ja sen vaikutuksia tulisi tutkia.

LÄHTEET

Apache Software Foundation. Apache Cordova. 2018. Luettu 12.12.2018. https://cordova.apache.org/docs/en/latest/guide/overview/

Cordova geolocation. ngCordova. Luettu 12.12.2018. http://ngcordova.com/docs/plugins/geolocation/

Drifty co. Ionic v1 Guide. Luettu 10.12.2018. https://ionicframework.com/docs/v1/guide/preface.html

Guru99. What is AngularJS? Architecture & Features. Luettu 13.12.2018. https://www.guru99.com/angularjs-introduction.html

Imaginovation. Luettu 24.1.2018. https://medium.com/@Imaginovation/app-development-decisions-native-web-or-hybrid-31c103f9b4e1

Ionic. Tabs and Navigation. Luettu 13.12.2018. https://codepen.io/ionic/pen/odqCz

Mauron85. 2018. Cordova Background Geolocation. https://github.com/mauron85/cordova-plugin-background-geolocation/tree/2.x

Morony, J. 2017. Installing ngCordova in an ionic 1.x Application. Luettu 12.12.2018. https://www.joshmorony.com/installing-ngcordova-in-an-ionic-application/

Sandeep Panda. 2018. An overview of JavaScript promises. Kirjoitettu 17.4.2018. Luettu 12.12.2018. https://www.sitepoint.com/overview-javascript-promises/

Sass Basics. Sass. Luettu 13.9.2018. https://sass-lang.com/guide