LAMK Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

Mobiilinuokkukortti ja tiedonvälittäjäsovellus

Case: Nobiili

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2017
Joonas Mustonen

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

MUSTONEN, JOONAS:

Mobiilinuokkukortti & tiedonvälittäjä-sovellus
Case: Nobiili

Ohjelmistotekniikan opinnäytetyö, 45 sivua

Kevät 2017

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli tehdä mobiili nuorisokortti Lahden nuorisotoimelle. Sovelluksen tarkoituksena on toimia nuorisokortin lisäksi nuorille tarkoitettujen tapahtumien sekä uutisten tiedonvälittäjänä. Sovellus on suunnattu Lahden ja Lahden lähialueen nuorisolle. Toimeksiantajana oli Mediatalo Esa, joka siirtyi Keskisuomalaisen omistukseen vuonna 2016.

Tutkimusongelmana oli tehdä sovellus käyttäen sellaista teknologiaa, joka toimii mahdollisimman monella puhelimella käyttöjärjestelmästä riippumatta. Käytetyksi teknologiaksi valittiin Ionic-sovelluskehys sen laajan liitännäiskirjaston ja dokumentaation vuoksi.

Tuloksena valmistui toimiva sovellus, joka on julkaistu Google play-, Apple- sekä Windows-kaupoissa. Sovellus ei ole vielä korvannut aikaisemmin käytettyjä fyysisiä nuoriskortteja, mutta se on käytössä jo usealla Lahden seudun nuorella. Sovellus on ladattu pelkästään Google play -kaupassa yli 500 kertaa.

Asiasanat: Apache Cordova, Ionic Framework, AngularJS, JavaScript

Lahti University of Applied Sciences
Degree Programme in …

MUSTONEN, JOONAS:

Mobile youth centre card & communicator
Case: Nobiili

Bachelor's Thesis in Software Engineering ,45 pages

Spring 2017

ABSTRACT

The goal of this thesis was to develop a mobile authentication card for the youth to City of Lahti. Another purpose of this application is to channel information about local events and news aimed at children and teens. The thesis was commissioned by Mediatalo Esa.

The research problem was to create an application using technology which is usable on most mobile phones despite the operation system. The technology that was chosen is Ionic Framework because of Ionic's large plugin library and documentation.

The result of this thesis was a functional application, which is downloadble from app-stores. The application has not yet fully replaced the old card system. It has over 500 downloads from the Google play-store alone.

Key words: Apache Cordova, Ionic Framework, AngularJS, JavaScript

SISÄLLYS

1 JOHDANTO

Erilaisten mobiililaitteiden yleistyttyä palveluita on ryhdytty sähköistämään. Esimerkiksi ennen paperille tehdyt jäsenkortit halutaan siirtää digitaaliseen muotoon. Mobiilisovelluksia voidaan rakentaa natiivisovelluksina sekä hybridsovelluksina.

Mediatalo Esa on pitkään toiminut mediayhtiö, joka on erikoistunut painettujen ja digitaalisten sisältöjen sekä mediapalvelujen tuottamiseen. Mediatalo Esan liikevaihto vuonna 2015 oli 26,3 miljoonaa euroa. Vuonna 2016 Mediatalo Esa siirtyi Keskisuomalaisen omistukseen.

Lahden nuorisopalvelut tarjoavat tekemistä ja toimintaa nuorille vapaa-ajalle. Yksi näistä toiminnoista on nuorisotalojen ylläpito ja valvonta. Nuorisotaloissa käytetään jäsenkortteja tiloihin pääsemiseen.

Opinnäytetyön tavoitteena on tehdä mobiili nuorisokortti-sovellus Lahden seudun nuorille. Lisäksi sovelluksen tarkoituksena on toimia nuorille suunnattujen tapahtumien ja uutisten välittäjänä sekä järjestää pienimuotoisia kilpailuja nuorten kesken. Lisäksi tavoitteena on selvittää hybridisovellusten toimintaa.

Aikataulu projektille on rajallinen, joten sovelluskehyksenä toimii hybrid-sovelluskehys Ionic Framework. Tutkimusongelmana on rakentaa sovellus käyttäen kyseistä sovelluskehystä ja selvittää, miten tämä käytännössä tapahtuu. Ionic Framework mahdollistaa mobiilisovellusten kehittämisen Android-, iOS- sekä rajallisesti myös Windows-alustoille käyttäen samaa lähdekoodia. Natiivia sovellusta kehitettäessä sovellus täytyy tehdä erikseen jokaiselle halutulle alustalle.

Opinnäytetyö on jaettu kahteen osioon: Käytettyjen teknologioiden kuvaamiseen sekä sovelluksen rakenteeseen ja toimintaan. Ensimmäisessä osassa pääpainona on hybridisovellusten toiminta ja Ionic sekä Ionicissa käytettävät muut teknologiat, kuten AngularJS ja Apache Cordova. Toisessa osiossa käydään läpi sovelluksen kehittämiseen

käytetty kehitysympäristö, sovelluksen ominaisuudet ja toiminta ja sovelluksessa käytetty arkkitehtuuri.

2 HYBRIDISOVELLUSKEHITYS

Mobiilisovelluskehityksessä käytetään pääasiassa kolmea eri tapaa toteuttaa sovellus: Natiivi, HTML5 sekä Hybridi. Käytettävä tapa valitaan sovelluksen ominaisuuksien sekä projektin aikataulun perusteella (W3 2017).

Natiivisovelluksella tarkoitetaan sovellusta, joka on ohjelmoitu käyttäen suunnatun käyttöjärjestelmän tukevaa ohjelmointikieltä. Android-sovellukset käyttävät Java-kieltä, Apple-sovellukset Objective C:tä (Informationweek 2017). Natiisovellukset ovat hyvin skaalautuvia ja suorituskyvyltään tehokkaita, mutta mikäli sovellus halutaan suunnata useammalle alustalle, koko sovellus täytyy tehdä jokaiselle alustalle erikseen (Lifewire 2017). Poikkeustapauksena on React Native -sovelluskehys, joka kääntää JavaScriptillä tehdyn sovelluksen täysin natiiviksi pitäen natiivinomaisen suorituskyvyn ja ulkoasun. React Native ei kuitenkaan tue Androidin ja IOS:n lisäksi muita alustoja.

HTML5-sovellukset ovat mobiililaitteen Internet-selaimessa toimivia sovelluksia. Niiden kehittämiseen käytetään samoja tekniikoita (HTML, CSS, JavaScript), kuin perinteisissä web-sivuissa. Tämä mahdollistaa sovelluksen toiminnan jokaisella alustalla yhdellä lähdekoodilla. HTML5-sovellusten rajoituksena on suorituskyky raskasta sovellusta tehdessä. Myöskään kaikkia puhelimen toimintoja ei voida käyttää. (Developer Salesforce 2017)

Hybridisovellukset ovat yhdistelmä edellä mainittuja tekniikoita. Sovellukset toimivat niin sanotussa WebView-näkymässä, joka on sovelluksen sisäänrakennettu Internet-selain kokoruututilassa ilman osoitepalkkia tai muita selaimen toimintoja. Hybridisovellukset pääsevät natiivisovellusten tapaan käsiksi kaikkiin puhelimen toimintoihin samalla säilyttäen toiminnallisuuden eri alustoilla käyttäen yhtä lähdekoodia. Suorituskyvyssä hybridisovellukset kuitenkin jäävät natiivisovelluksille jälkeen. (W3 2017.)

Hybridi-sovelluskehyksiä on useita, ja monesti niistä hyödyntävät Apache Cordovan toimintoja. Osa niistä on rakennettu joko Apache Cordovan päälle, kuten Ionic Frameowrk, osa taas sisältää tuen Cordovaan, kuten jQuery mobile.

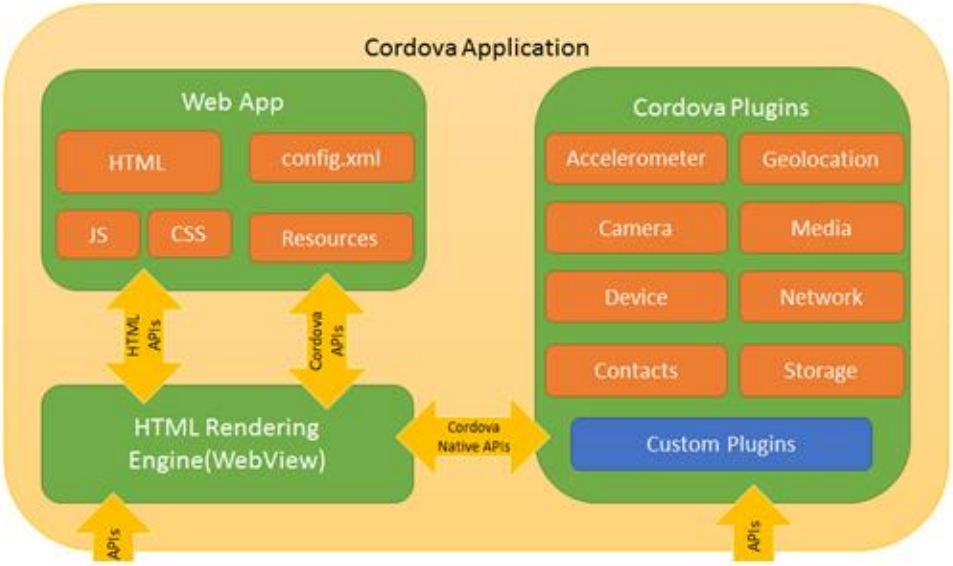
2.1 Apache Cordova

Apache Cordova on avoimeen lähdekoodiin perustuva hybridikehitysalusta. Aikaisemmin Cordova tunnettiin nimellä Phonegap, jonka luojana oli Nitobi vuonna 2009. Vuonna 2011 Adobe Systems osti Nitobin, jonka seurauksena Phonegapista tehtiin Open source-versio ja se uudelleennimettiin Apache Cordovaksi (Apache Cordova vs Adobe Phonegap 2017).

Apache Cordova mahdollistaa hybridisovellusten kehittämiseen eri alustoille. Tuettuja alustoja ovat Android, Applen iPhone, Bada, BlackBerry, Firefox OS, Symbian, Tizen, webOS, Ubuntu Touch sekä Windows Phone (Apache Cordova 2017).

Vaikka Apache Cordova itsessään tukee useita alustoja, kaikille alustoille sovellusten rakentaminen ei onnistu suoraan. Esimerkiksi Windows-laitteelle sovellusta kehitettäessä täytyy kehitysympäristönä olla Windows. Samoin IOS:lle kehittäessä täytyy olla Mac. Suunnattaessa sovellusta Androidille ei käyttöjärjestelmällä ole väliä. Adobe Phonegap tarjoaa mahdollisuuden kääntää projektin käyttäen pilvipalvelu Phonegap Buildia. Se tukee myös Windows- ja IOS-alustoja.

Phonegap Build on Adoben kehittämä pilvityökalu, joka mahdollistaa sovellusten kääntämisen HTML5-kielestä toimivaksi applikaatioksi. Pilvipalvelu sisältää myös jakomahdollisuuden sovellukselle. Sovelluksen voi asentaa suoraan pilvestä ilman yhdistämistä tietokoneeseen (Phonegap Build 2017). Phonegap Build ei ole osa Apache Cordovaa.



KUVIO 1. Cordovan arkkitehtuuri (Apache Cordova 2016)

Kuviossa 1 Cordovan käyttämä arkkitehtuuri sovellusten kehittämisessä. Sovelluksen kaikki lähdekoodi on Web Appin (Web-sovellus) sisällä, minkä Cordova muuttaa laitesovellukseksi liittämällä sen laitteen WebViewiin (Web-näkymä). Jokainen Cordova-sovellus sisältää config.xml-tiedoston, joka sisältää tietoja sovelluksesta, kuten sovelluksen nimi ja sen kuvaus (Apache Cordova 2016b).

Cordova-pohjaiset sovellukset toteutetaan käyttäen Javascript- sekä HTML5-kieltä. Koska Cordova-hybridisovellus on periaatteeltaan web-sivu, laitemallikohtaista natiivia koodia ei tarvita, vaan sama lähdekoodi toimii samalla tavalla laitteesta riippumatta.

Pluginit eli liitännäiset tarjoavat rajapinnan Cordovan ja laitteen natiivien komponenttien välillä. Tämä mahdollistaa natiivien ominaisuuksien, kuten yhteystietojen, käytön käyttäen JavaScriptiä. Kuviossa 1 näkyvät liitännäiset ovat osa Cordovan itse ylläpitämiä plugineita. Tämän lisäksi Cordovaan on saatavilla kolmannen osapuolen kehittämiä liitännäisiä, joka tarjoavat ominaisuuksia, joita ei välttämättä ole tarjolla jokaiselle

puhelinalustalle. Tällaisiä ovat esimerkiksi Google Analytics sekä SQLite-tietokanta (Apache Cordova 2017).

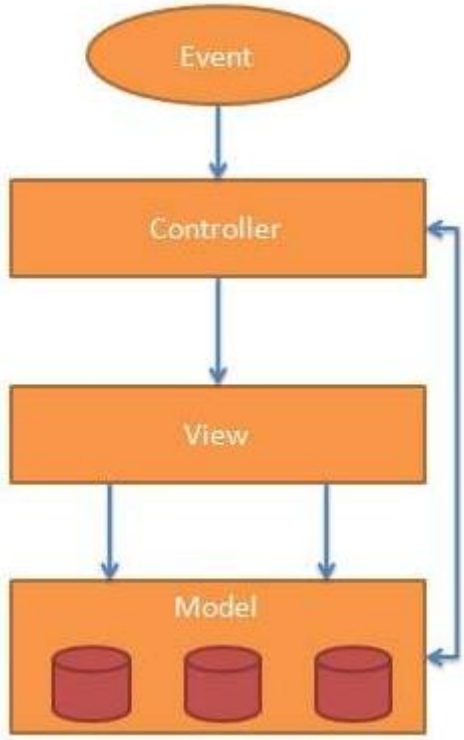
Cordova käyttää komentorivirajapintaa (CLI) uusien projektien aloittamiseen ja tuotantoon laittamiseen. Komentorivirajapinnan avulla projektiin myös lisätään tarvittavat alustat ja liitännäiset.

2.2 AngularJS

AngularJS on JavaScript-pohjainen avoimeen lähdekoodiin perustuva front-end web-sovelluskehys, jota ylläpitää pääasiassa Google ja yhteisö yksityishenkilöitä sekä yrityksiä. AngularJS:ä käytetään etenkin yhden sivun sovelluksiin (SPA) sekä mobiilisovellusten kehittämiseen. Muun muassa Ionic Framework käyttää AngularJS:ä (AngularJS 2016). AngularJS:n julkaisi alun perin Miško Hevery vuonna 2009. Siitä kaavailtiin maksullista sovelluskehystä yrityskäyttöön, mutta ideasta luovuttiin ja se julkaistiin avoimen lähdekoodin kirjastoksi (Wikipedia 2016). Angularista julkaistiin versio 2.0 syyskuussa 2016.

2.2.1 Arkkitehtuuri

AngularJS käyttää sovellusrakenteenaan MVC-mallia. MVC-mallia käytetään, koska se jakaa sovelluksen logiikan ja käyttöliittymän toisistaan täten selkeyttäen sovelluksen rakennetta. MVC-malli sisältää kolme osaa: Model, eli malli, View eli näkymä sekä Controller eli ohjain. Yleisen MVC-mallin lisäksi AngularJS käyttää Servicejä eli palveluita sekä Routeria eli reititintä (Tutorialspoint 2017a).



KUVIO 2. MVC-malli ja AngularJS (Tutorialspoint 2017b)

Kuviossa 2 on esitetty AngularJS:n tapa toteuttaa MVC-malli. Ohjain kuuntelee käyttäjän toimintoja ja hakee tarpeellisen tiedon mallilta, minkä jälkeen ohjain tuo tiedon näkymään. Yleensä itse ohjaimet eivät sisällä logiikkaa, vaan ne kutsuvat palveluita, jotka hakevat tiedot. AngularJS tarjoaa valmiita palveluita, kuten AJAX-kutsuihin käytettävä $http-palvelu. Yleensä kuitenkin palveluita joudutaan tekemään itse hyödyntämällä AngularJS:n valmiita palveluita koodin uudelleenkäytön ja selkeyden vuoksi (Tutorialspoint 2017a).

Näkymä on se osa, jonka käyttäjä näkee sovelluksesta. Näkymä sisältää kaiken HTML-koodin ja näyttää ohjaimen määrittämän tiedon käyttäjän syötteen perusteella. Näkymässä määritetään, mitä ohjainta käytetään ng-controller-direktiivin avulla (Angular Docs 2017.)

TAULUKKO 1. Direktiivejä

Yleisesti käytettyjä AngularJS-direktiivejä	
Direktiivi	Toiminta
ng-app	Määrittää Angular-sovelluksen juurielementin, minkä alla olevia direktiivejä voidaan käyttää
ng-bind	Määrittää DOM-elementin tekstiksi kyseisen muuttujan arvon
ng-model	Toimii, kuten ng-bind, mutta arvon muuttuessa myös ohjaimelta saadun mallin arvo muuttuu
ng-class	Mahdollistaa CSS-luokkien dynaamisen latauksen
ng-controller	Määrittää näkymässä käytettävän ohjaimen

Direktiivit ovat AngularJS:ssä käytettyjä HTML-attribuuttien kaltaisia uudelleenkäytettäviä elementtejä, jotka ennaltamäärittävät niiden sitomista komponenteissa. Datan sitominen tarkoittaa tiedontarjoajan ja saajan synkronointia. AngularJS:n tapauksessa tarjoaja on ohjain ja saaja näkymä. Direktiivejä voidaan myös luoda itse, jolloin niitä voidaan käyttää HTML-tagien tavoin näkymissä. AngularJS:n omissa direktiiveissä on etuliite ng. Taulukossa 1 on lista yleisesti käytettävistä direktiiveistä ja niiden toiminnasta.

Malli on yleensä tietoa, joka haetaan palvelimelta. Mallit eivät sisällä logiikkaa. Ohjain hakee mallin yleensä palvelun avulla ja sitoo sen näkymää.

Reitittimen tehtävänä on käyttäjän mennessä sivulta toiselle määrittää, mitä ohjainta ja näkymää milläkin sivulla käytetään. AngularJS:n reititin toimii omana moduulinaan ja se pitää liittää sovellukseen erikseen.

Moduulit ovat kokonaisuuksia, joista sovellus muodostuu. Yleisimmät tavat jakaa sovellus eri kokonaisuuksiin on joko liittää ohjaimet omaksi moduuliksi, näkymät omaksi moduuliksi sekä palvelut omaksi moduuliksi, tai liittää sovelluksen jokainen toiminto, kuten esimerkiksi kirjautuminen omaksi kokonaisuudeksi. Syitä moduulien tekoon on lähdekoodin ymmärrettävyys, moduulien uudelleenkäyttö sekä moduulien riippumattomuus muista. Testauksen näkökulmasta moduulirakenne vähentää testien määrää, mikä taas nopeuttaa varsinaista testaamista. (AngularJs 2017).

2.2.2 Datan sitominen

Datan sitomisessa ohjain välittää mallilta saadun tiedon näkymille käyttäen Scopea tai this-itseliitettä viitaten näkymään. Scope on objekti, mikä viittaa sovelluksen malleihin (AngularJs 2016).

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
<body>

<div ng-app="myApp" ng-controller="myCtrl">
<p>The url of this page is:</p>
<h3>{{myUrl}}</h3>
</div>

<p>This example uses the built-in $location service to get the absolute url of the
page.</p>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $location) {
    $scope.myUrl = $location.absUrl();
});
</script>

</body>
</html>
```

KUVIO 3. Scope (W3bschools 2016)

Kuviossa 3 on yksinkertainen esimerkki AngularJS:n toiminnasta. Skripti-tagien sisällä on määritelty ohjain, minkä scopella on ominaisuus nimeltään myUrl. Kyseinen scope käyttää AngularJS:n valmista palvelua $location, minkä metodi absUrl palauttaa tämänhetkisen sivun osoitteen. Näkymässä on määritelty ng-controller-direktiivin kohdalla mitä ohjainta kuunnellaan. Varsinainen scopesta saatu tieto näytetään näkymässä {{}} tagien sisällä. Toinen vaihtoehto on näyttää tieto käyttäen ng-model- tai ng-bind-direktiiviä.

Mikäli scopeen tulee muutoksia, $scopen ominaisuus päivittyy ja muutokset tulevat automaattisesti näkymään. Asia toimii myös toiseen suuntaan: Jos näkymässä oleva scopen arvo on sidottu ng-model-direktiivillä näkymään, näkymän arvon muuttuessa myös ohjaimessa oleva

kyseisen scopen arvo päivittyy (W3bschools 2017). Kuvan tapauksessa näin ei kuitenkaan tapahdu, koska kyseistä direktiiviä ei käytetä.

2.3 Ionic Framework

Ionic-sovelluskehys on valmis avoimen lähdekoodin SDK (Sovelluskehitys paketti) mobiilisovellusten kehitykseen. Ionic on rakennettu AngularJS:ä ja Apache Cordovaa käyttäen ja tarjoaa työkaluja ja palveluita hybridisovellusten kehittämiseen käyttäen Web-teknologioita, kuten CSS, HTML5 ja Sass. Sovellukset voidaan rakentaa käyttäen web-teknologioita ja sen jälkeen levittää natiiveissa sovelluskaupoissa tekemällä sovellustiedosto käyttäen Cordovaa (Wikipedia 2017).

Ionicin loivat Max Lynch, Ben Sperry ja Adam Bradley vuonna 2013. Toukokuussa 2015 julkaistiin Ionic 1.0 lopullinen versio ja 2.0 versiota aloitettiin julkaisemaan vuonna 2016. Vuonna 2015 luotiin yli 1.3 miljoonaa Ionic-sovellusta (Ionic Framework 2017a).

Koska Ionic on rakennettu Apache Cordovan pohjalta, sen toiminta muistuttaa Cordovaa. Cordovan sisältämät liitännäiset ja CLI-komennot toimivat myös Ionicissa. Sovelluskehykseen on lisätty oma CLI-rajapinta, mikä sisältää useita kehitystä nopeuttavia toimintoja. Esimerkiksi sovelluksen splash-screenin sekä ikonikuvan muuttaminen eri näyttökoon omaaville laitteelle optimoiduksi toteutuu yhdellä CLI-rajapinnan komennolla (Ionic Framework 2017a).

Ionic tukee Android-versioita 4.1 eteenpäin, IOS:a versiosta 7 eteenpäin. Ionicin 1.0 versiolla ei ole Windows-phone tukea, mutta sovellusten kehittäminen Windows-alustoille on silti mahdollista. Kaikki Ionicin ominaisuudet eivät kuitenkaan toimi, eikä niitä ole aikomusta lisätä (Mike Hartington 2016). Ionic 2.0 tukee Androidin ja IOSsin lisäksi Windows 10-alustaa.

Ionic toimii rakenteeltaan ja toiminnoiltaan samalla tavalla, kuin AngularJS-sovellus. Suuri osa AngularJS:ssä olevista palveluista ja direktiiveistä on läsnä ja näiden lisäksi Ionic sisältää kehykselle kustomoituja palveluita ja direktiivejä.

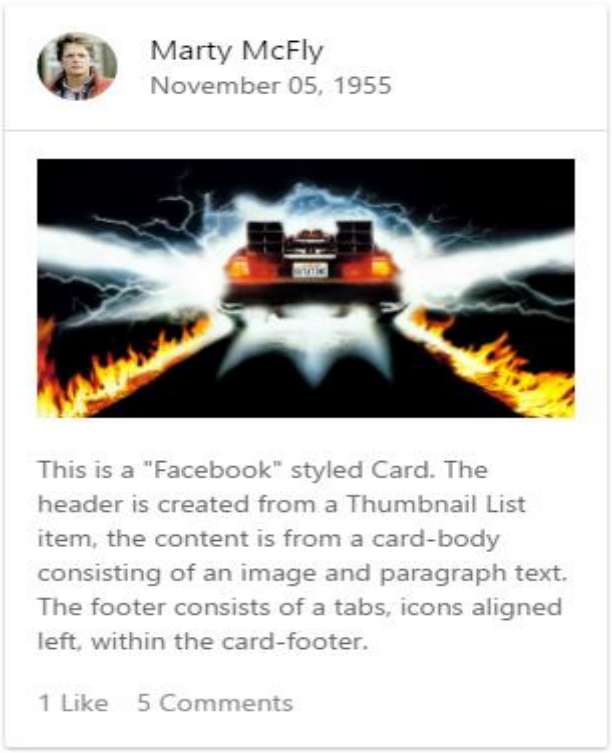
KUVIO 4. Esimerkkinäkymä

Kuviossa 4 on pelkistetty kuvakaappaus Ionicin näkymästä. Kaikki Näkymän sisältö on ion-view-direktiivin sisällä. Tämä on oleellista, koska kyseinen direktiivi mahdollistaa näkymien tallentumisen välimuistiin näkymien vaihtamisen yhteydessä. Se parantaa suorituskykyä, sillä kun näkymään navigoidaan uudestaan, ohjaimen ei tarvitse hakea tietoa uudestaan. Kyseisen ominaisuuden saa haluttaessaan pois päältä (Ionic Framework Docs 2017).

2.3.1 Käyttöliittymäkomponentit

Ionic tarjoaa valmiita ja muokattavia responsiivisia luokkia ja direktiivejä. Näitä direktiivejä ja luokkia kutsutaan CSS-komponenteiksi. Tämän johdosta Ionicin lisäksi ei tarvita muita CSS-sovelluskehyksiä, kuten Bootstrappia vai Googlen Material Design Liteä, mutta niitä kuitenkin voidaan käyttää. Ionicille on myös saatavilla erilaisia ulkoasuteemoja, joista osa on maksullisia (Ionic Framework 2017b).

CSS-komponentteihin kuuluu esimerkiksi lomakekentät, painikkeet, navigointipainikkeet sekä kortit (KUVIO 5).



KUVIO 5. Käyttöliittymäelementtejä (Ionic Framework docs 2017b)

CSS-komponenttien lisäksi Ionic sisältää yli 700 erilaista ikonia, joita kutsutaan Ioniconeiksi. Eri käyttojärjestelmille on myös omia Ioniconeita. Niitä voidaan myös muokata ja lisätä (Tutorialspoint Ionicons 2017).

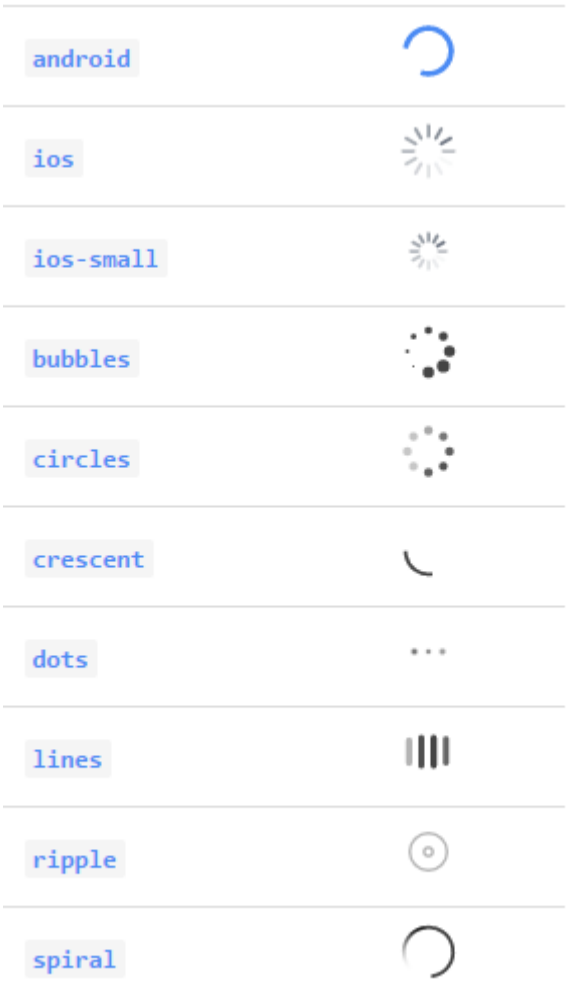
Ioniconit sisältävät web-kehityksessä usein käytettyjä ikoneita (KUVIO 6).

KUVIO 6. Ionicons (Ionic Framework docs 2017b)

Kyseiset lisäosat eivät rajoita käyttäjää tekemästä omia ikoneitaan tai käyttöliittymäkomponentteja. Komponentteja voidaan myös muokata tarvittaessa käyttäen CSS-kieltä, kuten tavallisessa web-kehityksesä.

2.3.2 AngularJS-liitännäiset

CSS-komponenttien lisäksi Ionic Framework sisältää joukon AngularJS:stä tuttuja palveluita ja direktiivejä. Ionicin omat palvelut sisältävät muun muassa reitittimen eli navigaattorin sovelluksen näkymien välillä, popup-ikkunoiden luomisen sekä käytettävän alustan havaitsemispalvelun, minkä avulla tunnistetaan laitteen tila sekä mahdollistetaan esimerkiksi Android-laitteiden takaisin-painikkeen toiminnallisuuden muokkaaminen (AngularJS-liitännäiset 2017). Ionicin reititin on nimeltään State provider. Perusperiaatteiltaan se toimii samalla tavalla, kuin AngularJS:n reititin. Mainittavia eroja ovat reittien kutsuminen tiloiksi (state) sekä automaattinen tilan tallentaminen välimuistiin.



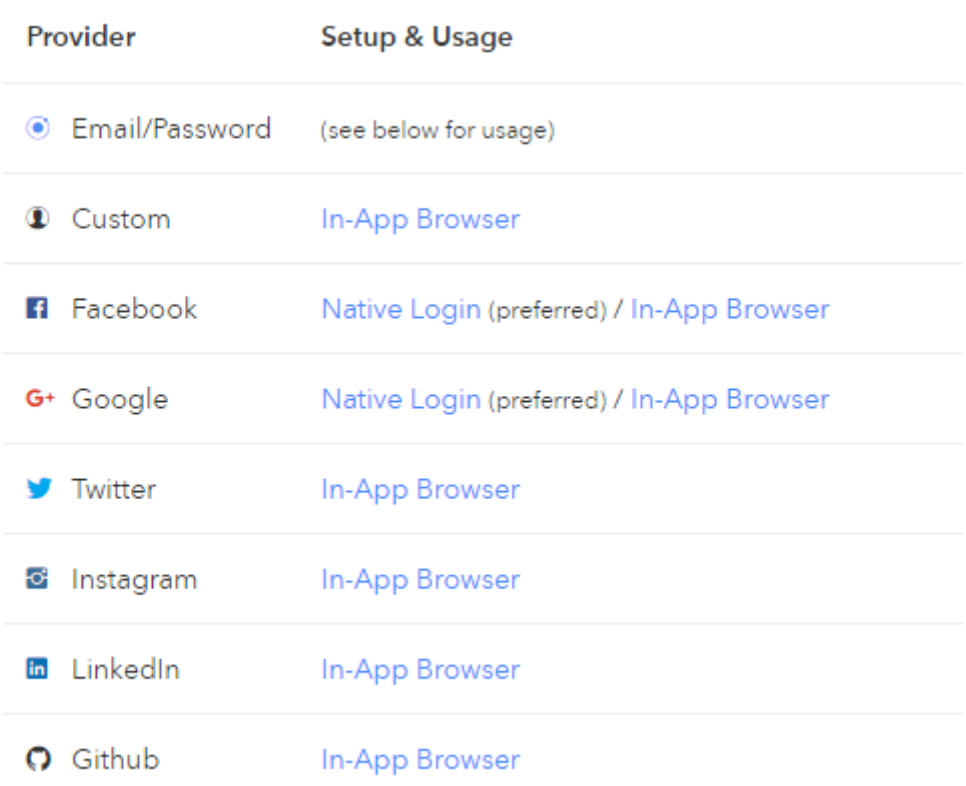
KUVIO 7. Spinner-direktiivejä

Direktiivit sisältävät pääasiassa ulkoasua elävöittäviä toimintoja. Esimerkkidirektiivejä ovat erilaiset spinnerit (KUVIO 7) sekä mobiililaitteisiin optimoitu lista.

2.3.3 Työkalut

Ionic tarjoaa monia työkaluja mobiiliapplikaatioiden kehittämiseen. Työkalut ovat ilmaisia, mutta lisäominaisuudet, kuten useamman sovelluksen hallinta samanaikaisesti, ovat maksullista (Ionic Cloud 2017).

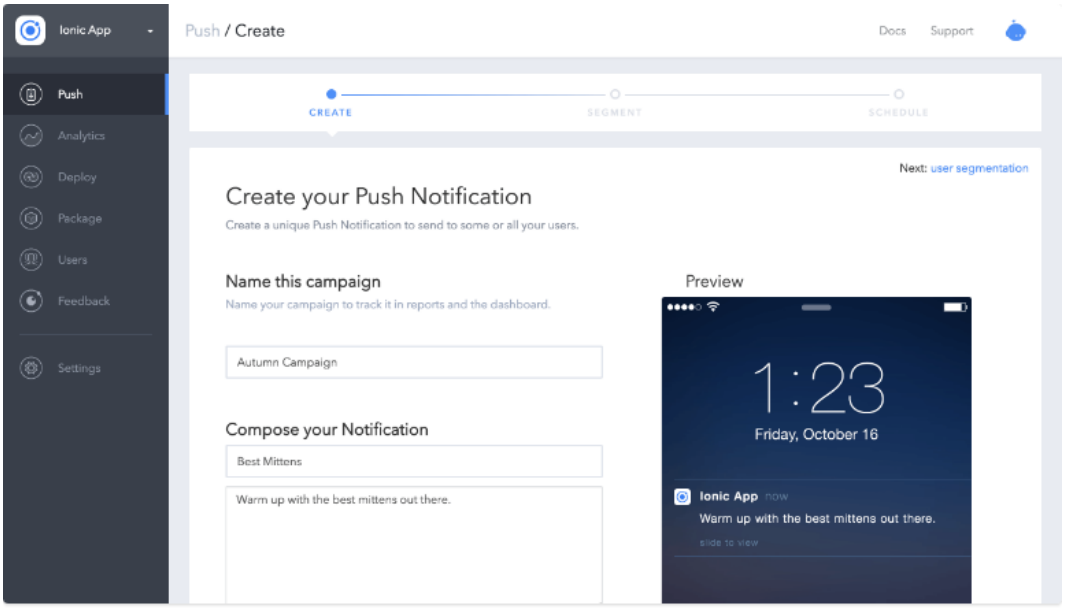
Ionic Cloud on pilvipalvelualusta Ionic-sovellusten hallintaan ja ylläpitoon. Ionic Cloud tarjoaa rajapinnan autentikoinnin eri sosiaalisten medioiden kautta, pushdown-notifikaatioiden lähettämiseen, käyttäjänhallinnan, natiiveja liitännäisiä, tietokannan sekä sovelluksien kääntömahdollisuuden pilvessä ilman Cordovassa tavallisesti vaadittavia työkaluja (Ionic Io 2017).



KUVIO 8. Varmennus-tarjoajat (Ionic Cloud 2017)

Ionic Auth on palvelu, mikä hallinnoi käyttäjän autentikoinnin Ionic-sovelluksissa. Autentikointia ei ole pakko käyttää. Se huolehtii

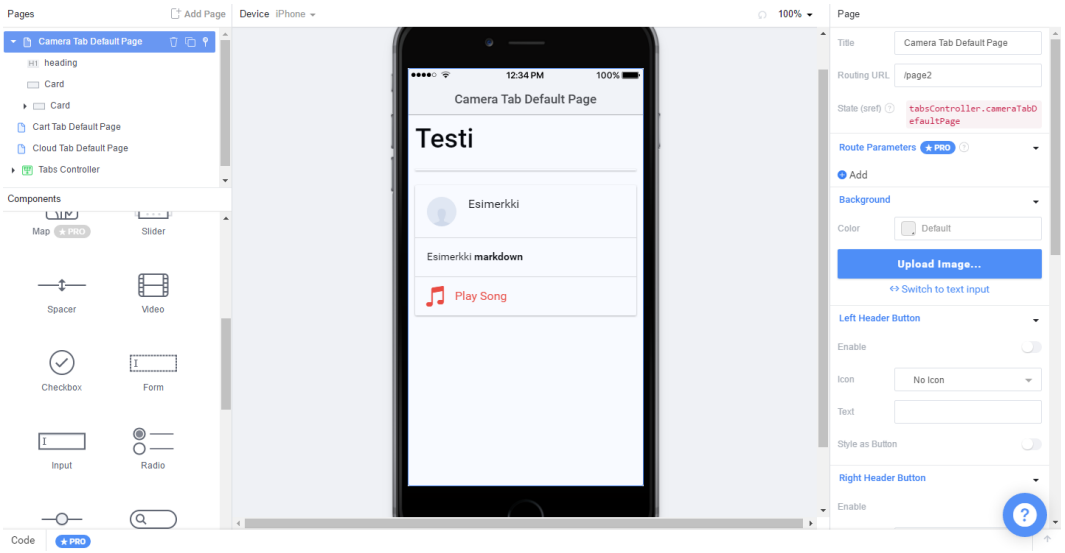
autentikoinnin eri lähteiden kautta käytäen Oauth-varmennusta, säilyttää käyttäjän salasanat, hallitsee sessionit ja tallettaa käyttäjän tiedot ja mahdolliset käyttäjäkohtaiset tiedot. Eri autentikointitarjoajien lisäksi (KUVIO 8), myös perinteinen sähköposti/salasana autentikointi on mahdollista. Ionic Auth sisältää valmiit lomakepohjat ja toiminnallisuuden kirjautumiseen ja salasanan unohtamiseen (IonicAuth 2017).



KUVIO 9. Push-notifikaation lähettäminen (Ionic Cloud 2017)

Ionic Push mahdollistaa viestien ja lähettämisen laitteelle, vaikka sovellus olisi suljettuna. Viestejä voidaan lähettää käyttäjäkohtaisesti tiettyinä aikoina, tai silloin kun käyttäjä laukaisee jonkin tapahtuman sovelluksessa. Ionic Pushin käyttöönotto on ilmaista, mutta viestien määrä on rajoitettu. Lisämaksusta viestejä voidaan lähettää niin paljon kuin halutaan. Notifikaatioita voidaan lähettää Ionic Cloudin kautta (KUVIO 9), tai rajapinnan kautta, mikäli notifikaatioita halutaan lähettää omalta palvelimelta (IonicPush 2017). Ionic Package on Phonegap Buildin kaltainen palvelu, minkä avulla sovellus saadaan käännettyä natiiviksi tiedostoksi pilvessä kehitysympäristön käyttöjärjestelmästä riippumatta. Pilvestä saatavat tiedostot voidaan lähettää sellaisenaan sovelluskauppoihin (IonicPackage 2017).

Ionic DB on Ionic Auth-palvelun kanssa toimiva NoSQL-tietokantapalvelu. Tietokanta on JSON-rakenteinen ja mahdollistaa reaaliaikaisen pääsyn tietoon. Se on yhteydessä sovellukseen socketin välityksellä. Tietokannan saa myös pois päältä Ionic Cloudista tarvittaessa (IonicDB 2017).



KUVIO 10. Ionic Creator (Ionic Creator 2017)

Ionic Creator (KUVIO 10) on niin sanottu raahaa ja pudota prototyyppityökalu Ionic-sovellusten suunnitteluun. Maksullisessa versiossa prototyypistä voi tehdä kokonaan toimivan sovelluksen ilman komentorivityökaluja. Ohjelmoiminen on myös mahdollista Ionic Creatorin kautta. Se käyttää Ionicin CSS-komponentteja ja Ioniconeja ja sisältää valmiita teemoja. Prototyyppiin voidaan myös liittää kuvia ja videoita. Sen ulkoasua ja navigointia voi tarkastella puhelin sekä tablet-näkymässä. Ionic Creator toimii selaimessa, mutta sen saa myös ladattua omana sovelluksenaan. (Ionic Creator 2017). Creator pakottaa käyttäjän tekemään koodia tietyn rakenteen mukaisesti, joten koodista tulee teoriassa Ionicin arkkitehtuurin mukaista (Ionic Blog 2014).

Ionic View on mobiilisovellus, minkä avulla vielä kehitysvaiheessa olevia sovelluksia voidaan jakaa esimerkiksi testaajille ilman sovelluksen

laittamista sovelluskauppaan. Tämä mahdollistaa sovelluksen testaamisen myös IOS-laitteella. Ionic View ei kuitenkaan tue kaikkia liitännäisiä, joten täydellistä testaamista sillä ei voi tehdä. Sovellus on saatavilla Androidille ja IOS:lle (Rapid Development with Ionic View 2016).

3 SOVELLUKSEN TOIMINTA

Nobiili toimii Lahden seudun nuorten nuorisokorttina sekä erilaisten nuorille suunnattujen tapahtumien välittäjänä. Tapahtumat on jaettu seuraaviin kategorioihin: tiedotteet, uutiset sekä tapahtumat. Jokaisen kategorian sisältö tulee alun perin eri lähteestä, minkä sovellukseen yhteydessä oleva palvelin hakee ja välittää Nobiiliin.

Tiedotteet ovat Lahden nuorisopalveluiden tekemiä tapahtumia, tapahtumat ovat Etelä suomen sanomien Menoinfo-rajapinnasta tulevia nuorille suunnattuja tapahtumia, ja uutiset ovat Etelä Suomen Sanomien nuorille suunnattuja uutisia.

3.1 Kehitysympäristö

Mobiilisovellus haluttiin saada beta-versioon mahdollisimman nopeasti, joten paras vaihtoehto oli tehdä siitä hybridisovellus. Valittu Framework oli Ionic 1.0 sen valmiiden komponenttien ja työkalujen takia. Versio 2.0 oli vielä alkuvaiheessa sovellusta kehittäessä, joten sen käyttäminen ei olisi ollut järkevää.

Sovelluksen kehittäminen tapahtui pääasiassa Windows 10 -käyttöjärjestelmällä. Tekstieditorina toimi suurimman osan ajasta Sublime Text, mutta loppuvaiheessa käyttöön otettiin Microsoftin Visual Studio sen Windows Phone -tuen johdosta. Sovellusta myös testattiin ja ja käännettiin ajoittain Mac-laitteella käyttäen xCodea, koska Apple-laitteelle kääntäminen vaatii IOS-ympäristön.

Versionhallintana toimi Git. Alun perin sovellusversiot sijaitsivat Bitbucketissa, mutta puolessavälissä projektia Mediatalo Esa tahtoi sovelluksen siirrettävän heidän omaan Github-säiliöön. Bitbucket ja Github toimivat lähestulkoon samalla tavalla, joten siirtäminen ei tuottanut suurempia ongelmia.

```
npm install -g cordova ionic
```

KUVIO 11. Ionicin asennus

Ionicin asennus tapahtuu NodeJS:n paketinhallintatyökalu npm:n avulla. Kuviossa 11 on komentorivikomento, joka asentaa Cordovan sekä Ionicin. Globaali lippu -g tarkoittaa, että kyseiset työkalut asennetaan käyttöjärjestelmässä sellaiseen paikkaan, että niitä voidaan käyttää missä tahansa käyttöjärjestelmän hakemistossa.

```
ionic start Nobiili tabs
ionic platform add android
ionic serve
ionic build android
ionic run android
```

KUVIO 12. Ionic CLI -komentoja

Kuviossa 12 on Ionic -komentorivityökalun keskeisiä komentoja. Jokainen komentorivikomento sisältää etuliitteen ionic. Ensimmäisessä komennossa argumentti start määrittää, että tehdään uusi projekti, seuraavana tulee sovelluksen nimi ja viimeisenä vapaaehtoinen pohja. Ionic sisältää kolme pohjavaihtoehtoa: blank sisältää vain pakolliset tiedostot ja kansiorakenteen, sidemenu tekee projektiin valmiina toimivan sivuvalikon, ja tabs tekee valmiin välilehtivalikon. Nobiilin tapauksessa valittiin tabs-vaihtoehto.

Platform-komento lisää projektiin halutun alustan. Alustavaihtoehtoja ovat Android, iOS sekä Windows. Kaikki alustat eivät kuitenkaan toimi kaikilla käyttöjärjestelmillä.

Serve-komento käynnistää sovelluksen nettiselaimessa. Sivu päivittyy automaattisesti, kun lähdekoodiin tehdään muutoksia. Selain ei tue Cordovan tai Ionicin kaikkia liitännäisiä, joten kaikkia sovelluksen

ominaisuuksia ei voi testata. Kyseistä toimintoa käytetäänkin lähinnä ulkoasun tarkasteluun.

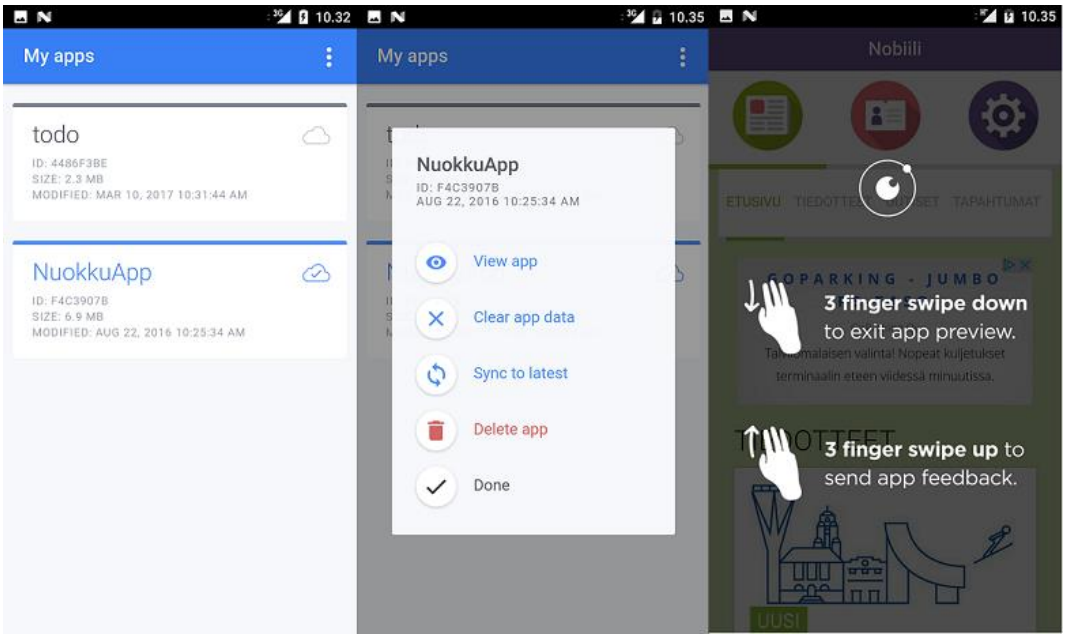
Build-komento kääntää sovelluksen halutulle alustalle ja luo siitä sovellustiedoston, minkä voi siirtää puhelimeen tai emulaattoriin, joko siirtämällä sen manuaalisesti tai käyttäen run-komentoa. Jotta sovelluksen voi kääntää halutulle alustalle ja testata laitteella tai emulaattorilla, täytyy jokaisen halutun alustan työkalut ottaa käyttöön. Android Studio sisältää tarvittavat kehitystyökalut Androidille, Visual Studio Windows Phonelle ja xCode IOS:lle.

Sovelluksen eri ulkoasumuutosten esittelyyn asiakkaille käytettiin Ionic View -sovellusta. Tämä mahdollisti kommenttien ja palautteen saannin nopeasti ja vähensi turhan työn määrää.

```
$ ionic login
$ ionic upload
$ ionic share EMAIL
```

KUVIO 13 Komentoja Ionic Cloudin käyttöönottoon

Sovelluksen siirtäminen Ionic Viewiin tapahtuu siirtämällä sovellus Ionic Cloud -järjestelmään käyttäen Ionic CLI -rajapinnan ylläolevassa kuvassa olevia komentoja. Kirjautumisen jälkeen sovellus lähetetään pilvipalveluun ja tämän jälkeen sovelluksen voi jakaa muille Ionic Viewin käyttäjille. Uuden version päivitys tapahtuu samalla tavalla, kuin lähetyksen.



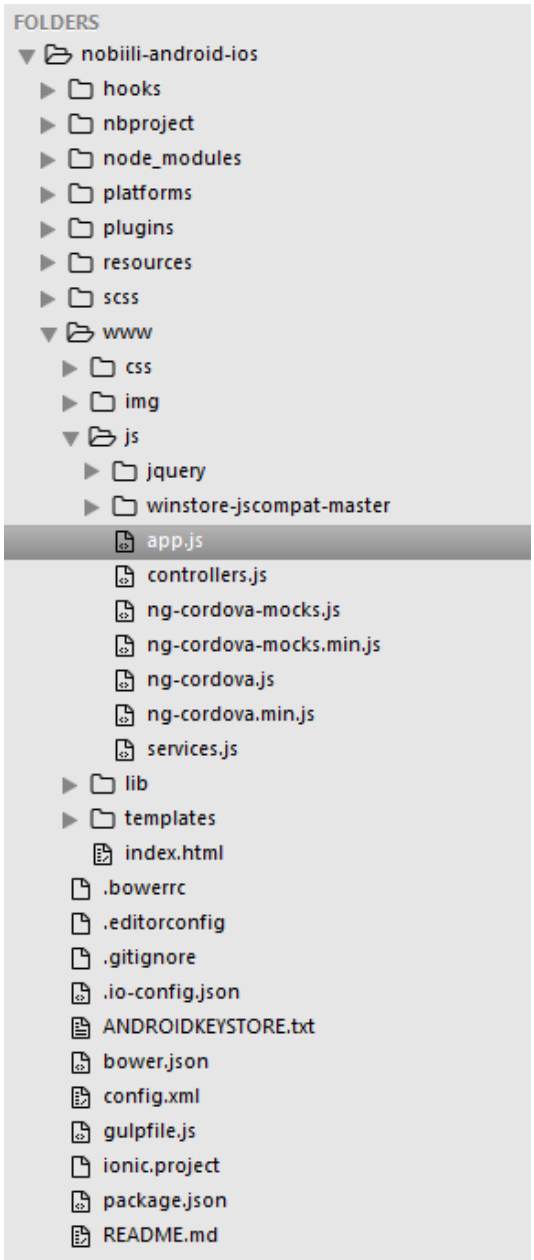
KUVIO 14. Ionic View-sovellus

Ionic View sisältää listan käyttäjälle jaetuista sovelluksista (KUVIO 14). Sovelluksen nimeä painaessa ruutuun ilmestyy vaihtoehtovalikko. Sovellukset eivät päivity automaattisesti, vaan käyttäjän pitää manuaalisesti synkronisoida sovellus painamalla Sync to latest-painiketta. Tarkasteltavan sovelluksen saa näkymiin View app-nappulaa painamalla. Tarkasteltava sovellus toimii samalla tavalla, kuin oikea sovellus ilman, että sitä pitää erikseen asentaa. Ionic View ei kuitenkaan tukenut kaikkia sovellukseen liitettyjä liitännäisiä, joten esittelyversioon joitain toimintoja oli pakko karsia.

Palautteen antaminen sovelluksesta onnistuu suoraan Ionic Viewin kautta. Sovelluksen tarkastelunäkymässä ohjeita noudattamalla (KUVIO 15) avautuu lomake, mitä kautta palautteen antaminen onnistuu. Palautteeseen voi myös halutessaan liittää kuvan sovellusnäkymästä. Palaute ilmestyy Ionic Cloudiin, missä palautetta voi tarkastella. Siinä näkyy myös laitteen käyttöjärjestelmä sekä malli.

3.2 Sovelluksen ominaisuudet

Kuviossa 15 on sovelluksen teossa käytetty tiedostorakenne, mikä vastaa AngularJS:n/Ionicin perusrakennetta. Config.xml-tiedosto sisältää sovelluksen tietoja, kuten nimi, versionumero, eri alustat sekä muita määrityksiä. Config.xml-tiedoston lisäksi lähdekoodi löytyy suurimmaksi osaksi projektin www-kansiosta.



KUVIO 15. Sovelluksen tiedostorakenne

Sovelluksen CSS-määritykset löytyvät css-kansiosta, kuvat sekä ikonit img-kansiosta ja näkymät ja direktiivit löytyvät templates-kansiosta. Varsinainen sovelluslogiikka eli ohjaimet, palvelut sekä app.js-tiedosto sijaitsevat js-kansiossa. Sovelluksen suhteellisen pienen koon vuoksi päätettiin jakaa ohjaimet omaan moduuliin sekä palvelut omaan moduuliin, sen sijaan, että moduulit olisi pilkottu toiminnoittain.

```
.config(function($httpProvider,$stateProvider,$ionicConfigProvider, $urlRouterProvider) {
  $ionicConfigProvider.navBar.alignTitle('center');


  $ionicConfigProvider.tabs.position('top'); //bottom
  $stateProvider
    .state('tab', {
    url: '/tab',
    abstract: true,
    templateUrl: 'templates/tabs.html'
  })
  .state('news', {
      url: '/news',
      cache:false,
      parent: 'tab',
      views: {
        'tab-news': {
          templateUrl: 'templates/news.html',
          controller: 'NewsCtrl'
        }
      }
    })
  .state('news.news', {
      cache: false,
      url: '/uutiset',
      views: {
        'newsContent': {
          templateUrl: 'templates/notifications.html',
          controller: 'ArticlesCtrl'
        }
      }
    })
```

KUVIO 16. App.js Reititin

Sovellus toimii Ionicissa käytettävän MVC-mallin mukaisesti. Jokaisella statella eli tilalla on tietty näkymä, jolla on yksi tai useampi ohjain. Ohjain käyttää puolestaan poikkeuksetta palveluja, jotka hakevat, tai prosessoivat tietoa. Kuviossa 16 on esimerkkejä App.js-tiedostossa olevien tilojen määrittämisestä. Tiloja voi myös tallentaa välimuistiin, jolloin jo haettua tietoa ei haeta uudestaan, mutta koska sovelluksen tarkoituksena on

hakea uutisia, kyseinen ominaisuus otettiin pois asettamalla cache-lippu epätodeksi tilan määrittelyssä. Tilojen hallinnan lisäksi App.js-tiedosto sisältää sovelluksen konfiguraatioasetukset, kuten http-kutsujen aikakatkaisu ja SQLite-tietokannan alustus.

```
.factory('$localstorage', ['$window', function($window) {
  return {
    set: function(key, value) {
      $window.localStorage[key] = value;
    },
    get: function(key, defaultValue) {
      return $window.localStorage[key] || defaultValue;
    },
    setObject: function(key, value) {
      $window.localStorage[key] = JSON.stringify(value);
    },
    getObject: function(key) {
      return JSON.parse($window.localStorage[key] || '{}');
    }
  }
}])
```

KUVIO 17. Local Storage-factory

Kaikki sovelluksen näyttämä muuttuva tieto, kuten uutiset ja jäsenkunnat ja niiden tiedotteet haetaan Adapteri-nimisestä rajapinnasta. Adapterin teko oli osa varsinaista projektia, mutta tässä opinnäytetyössä keskitytään vain mobiilisovellukseen.

Tietojen tallentamiseen käytetään laitteen omaa Local storage-tallennustilaa sekä joissain tapauksissa laitteen SQLite-tietokantaa. Aluksi suunnitelmana oli käyttää pelkkää Local storagea sen helppokäyttöisyyden vuoksi, mutta projektin loppupuolella otettiin käyttöön myös SQLite.

Syynä molempien käyttöön oli IOS-laitteiden ominaisuus, mikä tyhjentää Local storagen, mikäli laitteen muisti on vähissä. Osa tallennettavasta tiedosta, kuten uutiset haetaan palvelimelta vähintään kerran sovelluksen käynnistäessä, joten niiden siirtämistä SQLite-tietokantaan ei nähty

tarpeelliseksi. Molempien tallennusmuotojen käyttöön tehtiin erillinen service tai factory. Kuviossa 17 on Local Storage-factory ja sen sisältämät funktiot. Objektien tallentaminen toteutettiin muuttamalla perinteinen Javascript-objekti ensin merkkijonoksi ja suorittamalla tallennus sen jälkeen. Objekteja haettaessa merkkijono muutetaan objektiksi ja tallennetaan sitten muuttujaan. SQLiten käyttöön myös asennettiin erillinen Cordova-liitännäinen.

```
.service('OAuth',function($state,avaimet,$localstorage,$http,$q){
    var keys = avaimet;
    // AccessToken
    var getToken = function(){
        console.log("haeaan access tokenia....");
        var url = [redacted];
        // Haetaan token palvelimelta käyttäen avainta ja secrettiä
        var promise = $http({
            method: 'get',
            url: url,
            params: {
            client_id: avaimet.client_id,
            client_secret: avaimet.client_secret,
            grant_type: "client_credentials"
            }
        //Tallennetaan Access_token localstorageen
        }).then(function(response){
                $localstorage.set('access_token',response.data.access_token);
        },function(err){

        },function(response){
            return $q.reject(response);
        });
        return promise;
    }
    return {
        getToken: getToken,
    }
})
```

KUVIO 18. Autentikointi-service

Palvelimen ja mobiilisovelluksen kommunikointiin käytetään Oauth 2-autentikointia, mikä mahdollistaa turvallisen tiedonvälityksen palvelimen ja puhelinsovelluksen välillä. Sovelluksella on tiedossa avain ja arvo- pari, joiden avulla sovellus saa palvelimelta väliaikaisen tokenin, mikä lähetetään jokaisen tiedonhaku, tai lähetys -funktion mukana.

Tokenit ovat voimassa yhden tunnin, jonka jälkeen uutta tiedonhakukutsua tehtäessä palvelin palauttaa virhekoodin 401, jolloin mobiilisovellus hakee ensin uuden tokenin, jonka jälkeen suorittaa halutun toimenpiteen uudestaan. Joissain tapauksissa palvelin ei anna suoranaista virhekoodia, vaan sisällyttää sen vastaukseen. Tiedonhakupalveluissa on otettu molemmat tilanteet huomioon.

Tokenin hakemiseen tehtiin erillinen palvelu (KUVIO 18), minkä funktiota getTokenia muut servicet käyttävät tiedonhaun yhteydessä. Avaimien lisäksi funktio vaatii parametriksi halutun oikeustyypin. Nobiilin tapauksessa ainoa haluttu tyyppi on Client Credentials, mikä tarkoittaa suoraa pääsyä tietoon ilman käyttäjätunnuksia. Muita tyyppejä ovat Resource Owner Password Credentials, mikä on yksilöyty tokeni käyttäjän henkilökohtaiseen tietoon, Refresh Token, mikä päivittää tokenin automaattisesti sen vanhennuttua sekä Implicit, mikä toimii samalla periaatteella, kuin Resource Owner Password Credentials. Nobiilin tapauksessa valittiin Client Credentials, sillä Nobiili ei vaadi, tai tarjoa suoranaista rekisteröitymistä sovelluksen käytössä. Kun token saadaan palvelimelta, se tallennetaan local storageen.

```
if(typeof analytics !== "undefined") {
    analytics.startTrackerWithId([redacted]);
    analytics.trackView('Sovellus käynnistettiin');

} else {
    console.log("Desktop mode");
}
```

KUVIO 19. GA-liitännäisen käyttöesimerkki

Sovelluksen käyttäjiltä haluttiin kerätä anonyymiä tietoa siitä, millä nuorisotalolla käyttäjät käyvät eniten, mikä antaa osviittaa siitä, mitä uudistuksia sovellukseen kannattee tehdä. Tämä toteutettiin käyttäen

Google Analytics-palvelua. Kyseiseen palvelun käyttöönottoon projektiin liitettiin Cordova-liitännäinen, mikä helpotti integrointia merkittävästi. Kuviossa 19 on esimerkki liitännäisen käytöstä.

Sovellus toimii rajoitetusti myös ilman verkkoyhteyttä. Tällöin sovellus uudelleenohjaa käyttäjän Uutisvirta-sivun sijaan Jäsenkortti-sivulle, missä näkyy käyttäjän nuorisokortti. Laitteen verkkoyhteyden tilan tarkasteluun käytettiin Network Information-liitännäistä. Sovelluksen käynnistyessä verkkoyhteyden tilan perusteella verkkoyhteyttä tarvittavat ominaisuudet suodatetaan sovelluksesta pois. Mikäli verkkoyhteys ei ole käytössä sovellusta käynnistäessä ensimmäistä kertaa, sovellus antaa ilmoituksen verkkoyhteyden puuttumisesta ja sulkee itsensä, sillä ensimmäisellä käynnistyskerralla sovellus hakee tarvittavat tiedot palvelimelta toimiakseen.

Käytettävään laitteeseen tunnistamiseen käytettiin Device-liitännäistä. Ionic käyttää oletuksena laitteen natiivia skrollia, mutta vanhemmissa Android-versioissa kyseinen ominaisuus ei toiminut tarpeeksi sulavasti, joten näillä laitteilla sovellusta käyttäessä sovellus vaihtaa natiivin scrollin JavaScript-scrolliksi. Laitteen uniikki tunnistenumero saadaan myös liitännäisestä, mitä käytetään jäsenkortin aktivoimisessa.

3.2.1 Uutisvirta

Sovelluksen uutisvirta on jaettu kolmeen eri kategoriaan. Uutiset sisältävät Mediatalo Esan tuottamia nuorille suunnattuja uutisia. Tiedotteet sisältävät nuorisotalojen tiedotteita. Tapahtumat ovat Mediatalo Esan tapahtumakalenterista haettuja erilaisia tapahtumia, kuten elokuvia, tai urheilutapahtumia.

KUVIO 20. Uutiset-navigaatio

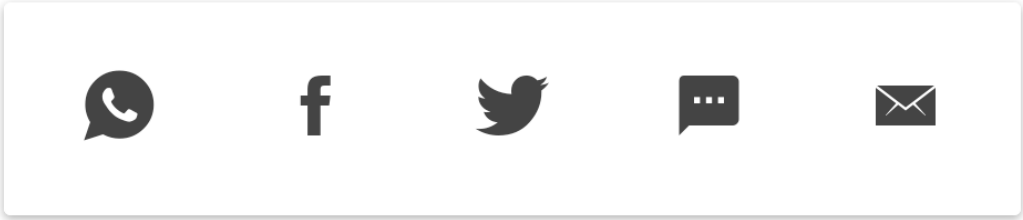
```
getCompare: function(newArray,current){
  if(newArray.constructor != Array || current.constructor != Array){
    return;
  }
    var timestamp  =  new Date().getTime() - (3 * 24*60*60*1000);
    newArray.forEach(function(result){
        var found = false;
        for(var i=0;i<current.length;i++){

            var publish = new Date(result.publishTime);
            result.publishTime = publish;
            if(publish.getTime()- timestamp <0){
                found = true;
                break;
                return;
            }
            var newString = JSON.stringify(result);
            var currentString = JSON.stringify(current[i]);
                if(newString == currentString){
                    found = true;
                    break;
                }
        }
        //LÖYTYY UUSI
        if(found == false){
            result.new = 'UUSI';
        }
    })
},

compareNew : function(array, id,storagename){
    var found = false;
    for(var i=0;i<array.length;i++){
        if(array[i].EventId == id){
            var temp =  array;
            delete temp[i].new;
            $localstorage.setObject(storagename,temp);
            array = $localstorage.getObject(storagename);
            found = true;
            return true;
            break;
        }
    }
    return false;
```

KUVIO 21. compareNews-factoryn funktiot

Uutisten, jotka ovat alle kolme päivää vanhoja eikä käyttäjä ole niitä vielä lukenut, sisältävät Uusi-tagin. Joka kerta, kun käyttäjä siirtyy Uutisvirta-sivulle, sovellus hakee uutiset ja vertaa niitä jo haettuihin uutisiin käyttäen compareNews-factorya. Aikavertailun lisäksi factoryn funktio getCompare varmistaa, että mikäli uutinen on jo kertaalleen haettu ja luettu, siihen ei lisätä tagia uudelleen. Uutista klikatessa edellisen factoryn compareNew-funktio poistaa Uusi-tagin uutisesta.



KUVIO 22. Uutisen jakaminen

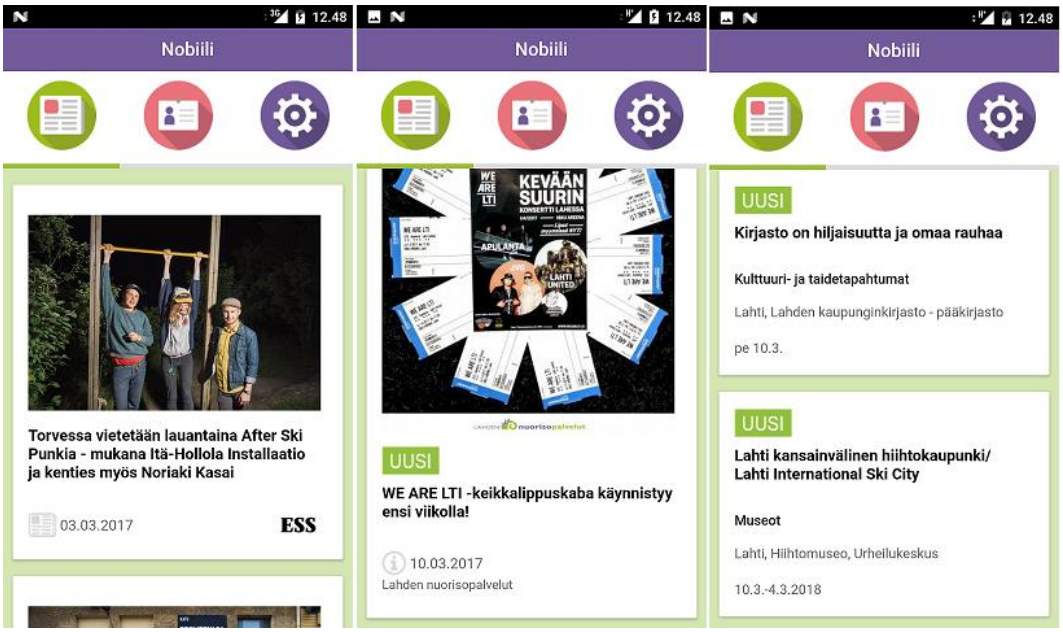
Uutiset sisältävät myös jakomahdollisuuden sosiaalisessa mediassa. Tuettuja jakomuotoja ovat WhatsApp, Facebook, Twitter, sms sekä sähköposti. Uutisten jakoon käytettiin Cordovan Social sharing-liitännäistä. Kyseiset painikkeet jakavat Mediatalo Esan virallisen version uutisesta, eikä puhelinsovelluksessa näkyvää uutista. Tämän takia jakomahdollisuutta ei lisätty tiedotteisiin tai tapahtumiin.

```
$scope.filters  = [

        {
            filterID:1,
            name:"UUTISET",
            url: "#tab/news/uutiset",
            data:'news'
        },
        {
            filterID:2,
            name:"TIEDOTTEET",
            url: "#tab/news/tiedotteet",
            data:'tiedotteet',
             new: null
        },
        {
            filterID:3,
            name:"TAPAHTUMAT",
            url: "#tab/news/events",
            data:'events'
        },
        ];
```

KUVIO 23. Uutisvirran navigaatio

Uusi-tagin poisto ja muut jokaiselle kategorialle yhteiset ominaisuudet toteuttaa News-tila, minkä alla kaikki kolme kategoriaa ovat. Yhteisiä ominaisuuksia ovat navigoinnin näyttäminen sekä itse navigointi uutissivujen välillä, varsinaisen uutisen avaaminen sekä Google Analytiikasta huolehtiminen. Kuviossa 23 on NewsControllerin uutisvirtasivun navigaatio, mikä on sidottu näkymään kiinni käyttäen $scopea.



KUVIO 24. Uutisvirta-näkymän versiot

Uutisvirran kategoriat käyttävät samaa näkymää uutisten näyttämiseen. Kategorioiden uutisvirrat eivät sisällä täysin samoja elementtejä (KUVIO 24), esimerkiksi tapahtumat eivät sisällä ollenkaan kuvia. Tällaisissa tapauksissa käytetään AngularJS:n sisältämää ng-if-direktiiviä, mikä näyttää halutun elementin vain silloin, jos kategorian uutinen sisältää sen. Uutisvirran kaikki uutiset näytetään käyttäen ng-repeat-direktiiviä.

```
.service('ess',function($window,compareNews,$localstorage,$http,$q,OAuth,avaimet){
    var getEss = function(){
        var promise = $http({
            method: 'post',
            url: 'http://52.19.106.153/plugin/tapkal/getess',
            params: {access_token: $localstorage.get('access_token')}
        }).then(function(response){
             if(response.data.error){
                OAuth.getToken().then(function(){
                  return  getEss();
                });
             }
             else{
                compareNews.getCompare(response.data,$localstorage.getObject('ess'));
                response.data.forEach(function(article){
                  article.publishTime = new Date(article.publishTime);
                });
                $localstorage.setObject('ess', response.data;);
                return $localstorage.getObject('ess');
             }
          },function(err){
          //401 = Unauthorized (access_token on mennyt vanhaksi!!)
          if(err.status == 401){OAuth.getToken().then(getEss());}
        },function(response){
            return $q.reject(response);
        });
        return promise;
    };
    return{
        getEss: getEss
```

KUVIO 25. ess-service

Uutisvirtojen hakeminen tapahtuu AJAX-kutsuilla. Jokaiselle kategorialle kutsun rakenne on hieman erilainen, mutta pääperiaate pysyy samana. AJAX-kutsut on tehty palveluiksi (KUVIO 25) joita kyseisen uutisvirran ohjain kutsuu. Pakollisen tokenin lisäksi tiedotteiden hakupalvelun AJAX-pyyntöön annetaan parametriksi halutut nuorisotalojen nimet, mikäli halutaan hakea myös tietyn nuorisotalon yksityisiä tiedotteita. Tapahtumapalvelu taas lisää käyttäjän kotikaupungin sekä asetuksista asetetut kaupungit yhdeksi parametriksi, mikäli niitä on asetettu.

```
.controller('ArticlesCtrl', function (ess, $scope) {
    ess.getEss().then(function (result) {
        $scope.news = result;
    })
})
```

KUVIO 26. ArticlesCtrl

Kuviossa 26 artikkeliohjain kutsuu ess-palvelun getEss-funktiota ja liittää uutiset näkymään. Ohjaimessa on myös käytetty funktioiden ketjutusta, mikä mahdollistaa http-kutsun suorituksen päättymisen odottamisen. Sen jälkeen loput koodista suoritetaan käyttäen then-toiminnallisuutta. Uutisvirran muut ohjaimet ovat lähes identtisiä kyseisen ohjaimen kanssa.

3.2.2 Mobiili nuokkukortti

Sovelluksen pääasiallinen tarkoitus on toimia sähköisenä nuokkukorttina Lahden seudun nuorille. Nuorisokortti-näkymä (KUVIO 27) sisältää myös listan niistä nuorisotaloista yhteystietoineen, millä mobiili nuokkukortti on käytössä. Molemmat näkymät ovat varsinaisen Kortti-tilan alitiloja. Kortti-tilan ainoana tehtävänä on näyttää navigaatio näkymien välillä.

KUVIO 27. Nuokkukortti

Kortti aktivoidaan antamalla koodi, minkä nuori on saanut nuorisotyöntekijältä. Se voi olla kerrallaan voimassa vain yhdellä laitteella, sillä kortin aktivoinnissa toisena parametrinä on laitteen uniikki tunniste, mikä tallennetaan taustajärjestelmään. Kortin aktivoinnin jälkeen sovellus saa palvelimelta nuoren nimen, iän, kuvan ja nuorisokortin voimassaoloajan. Tyypillisesti kortti on kerrallaan voimassa kaksi vuotta, jonka jälkeen kortti täytyy uusia. Sovellus myös tarkistaa nuokkukortin tiedot joka kerta, kun sovellus käynnistetään, koska kortti voidaan myös ottaa pois käytöstä jostain muusta syystä. Nuorisokortin tiedot tallennetaan laitteen SQLite-tietokantaan. Kortin aktivointi voidaan myös peruuttaa sovelluksesta, jos käyttäjä haluaa esimerkiksi liittää kortin toiseen laitteeseen. Aktivoinnin poisto löytyy nuorisokortin alta ja näkyy vain silloin, kun kortti on aktivoitu. Kuviossa 28 on jäsenkorttinäkymän lähdekoodi.

```
<ion-nav-view >
    <div  class="ion-content nuokkuCard">
        <div class="list "   >
            <div ng-if="status != 'active'" >
                <input ng-model="data.koodi" type="hidden" />
                <h4 class="aligned">{{greeting.title}}</h4>
                <ion-item class="item-text-wrap">
                    <p>{{greeting.content}}</p>
                    <button  ng-click="nuoliAuth()"  class="button-balanced">
                    Aktivoi jäsenkortti
                    </button>
                </ion-item>
            </div>
            <div ng-if="status == 'active'" class="activeCard" >
                <h1 class="text-center username">{{nuokkucard.name}}</h1>
                <h2 class="text-center" >Kotikunta: {{home.title}}</h2>
                <div align="center">
                    <img ng-if="nuokkuimage != undefined"  class="img-circle" src="{{nuokkuimage}}" alt="Nuokkukuva" >
                </div>
                <img ng-show="imgURI !== undefined" src="{{nuokkuImage}}">
                <h1 class="text-center userdate"><b>{{nuokkucard.syntymaaika | date : 'dd.MM.yyyy'  }}</b></h1>
                <div class="row greencolumn">
                    <div class="col text-center" >
                        <h2>Jäsenyys voimassa <b >{{nuokkucard.expirationDate  | date : 'dd.MM.yyyy' }} </b>asti.</h2>
                    </div>
                </div>
            </div>
            <div ng-if="status == 'active'">
                <h4 class="aligned">Nuokkukortin aktivoinnin poisto</h4>
                    <button ng-click="removeUser()" class="button button-small button-assertive">
                        Poista
                    </button>
            </div>
            <div ng-if="error" >
                <ion-item class="item-text-wrap">{{error}}</ion-item>
            </div>
        </div>
    </div>
</ion-nav-view>
```

KUVIO 28. Jäsenkortti-näkymä

Kuten myös uutistilan alitiloissa, tilojen näkymät sijoitetaan ion-nav-view-direktiivin sisälle. Suuri osa näkymän teksteistä saadaan ohjaimelta käyttäen AngularJS:n kaksoisaaltosulku-merkintätapaa, mikä sitoo ohjaimelta saadun scopen muuttujan näkymään. Kortin aktivoinnin tilasta riippuen näkymä näyttää joko ohjeet nuorisokortin hankintaan (KUVIO 28), tai varsinaisen nuokkukortin sisällön hyödyntäen ng-if-direktiiviä. Ohjaimen sisältämät funktiot suoritetaan käyttäen ng-click-direktiiviä.

Funktioiden toiminnallisuuteen tehtiin Nuoli-palvelu, mitä ohjain käyttää tietojen haussa ja funktioiden suorittamisessa. Nuoli-palvelu hakee käyttäjän nuorisokorttitiedot adapterin kautta. Tämä puolestaan hakee tiedot Lahden nuorisotoimen omasta järjestelmästä. SQLite-tietokantaan tiedon tallentaminen tapahtuu samalla periaatteella, kuin perinteisempään MySQL-tietokantaan tallentaessa.

```
var auth = function(uniqueDeviceID,authCode){
    $localstorage.setObject('uniqueDeviceID',uniqueDeviceID);
    var promise = $http({
        method: 'post',
        url: 'http://52.19.106.153/plugin/api/nuoli/v1/auth',
        params: {access_token: $localstorage.get('access_token'), client_id: avaimet.client_id,
                 uniqueDeviceID: uniqueDeviceID, authCode: authCode}
        }).then(function(response){
          if(response.data.error){
          OAuth.getToken().then(function(){
            auth();
          })
        }
        var d = response.data;
        var query = "INSERT INTO nuokkucard (expirationDate, image,kunta,msg,name,syntymaaika) VALUES (?,?,?,?,?,?)";
        var removeErrors = "DELETE FROM errors";
        db.transaction(function(tx) {
            tx.executeSql(query, [d.expirationDate,d.image, d.kunta,d.msg,d.name,d.syntymaaika]);
            tx.executeSql(removeErrors);
        }, function(error) {
        }, function()
        alert(response.data.msg);
        },function(err){
          if(err.status == 401){
                    OAuth.getToken().then(function(result){
                        auth();
                    })
                    return;
                }
            },function(response){return $q.reject(response); });
    return promise;
}
```

KUVIO 29. Nuoli-palvelun auth-funktio

Cordova-liitännäisestä saatava db-muuttuja toteuttaa transaktion, minkä sisälle syötetään yksi tai useampi lauseke ja lausekkeen sisältämät arvot. Kuviossa 29 on tapahtumaa havainnollistava kuva, missä kortin tietojen haun jälkeen tiedot tallennetaan SQLite-tietokantaan.

Nuorisotilat – välilehteen nuorisotilat haetaan palvelimelta ensimmäisen kerran sovellusta käynnistäessä. Nuorisotalojen päivitys päätettiin toteuttaa lisäämällä asetussivulle nappula, mikä päivittää nuorisotilojen lisäksi kunnat, mitkä ovat osana mobiilisovellusta. Nuorisotilojen tietojen näyttäminen näkymässä on toteutettu käyttäen ng-repeat-direktiiviä.

3.2.3 Asetukset

KUVIO 30. Asetukset-sivu

Sovelluksen Asetukset-sivulla (KUVIO 30) käyttäjä pääsee määrittelemään, minkä nuorisotalojen ja kaupunkien uutisvirtaa sovellukseen näytetään. Sivulla on myös tietoja sovelluksesta sekä käyttöehdot. Kaikki asetukset ovat samassa näkymässä ja asetuksia voi avata useamman kerrallaan.

Käyttäjätilin asetukset sisältävät ainoastaan käyttäjän kotikunnan valitsemisen. Kotikuntatietoa käytetään Google Analytiikassa tarkastellen, minkä kaupungin nuoret käyttävät sovellusta eniten. Tieto ei ole pakollinen ja se täyttyy automaattisesti nuokkukortin aktivoinnin yhteydessä.

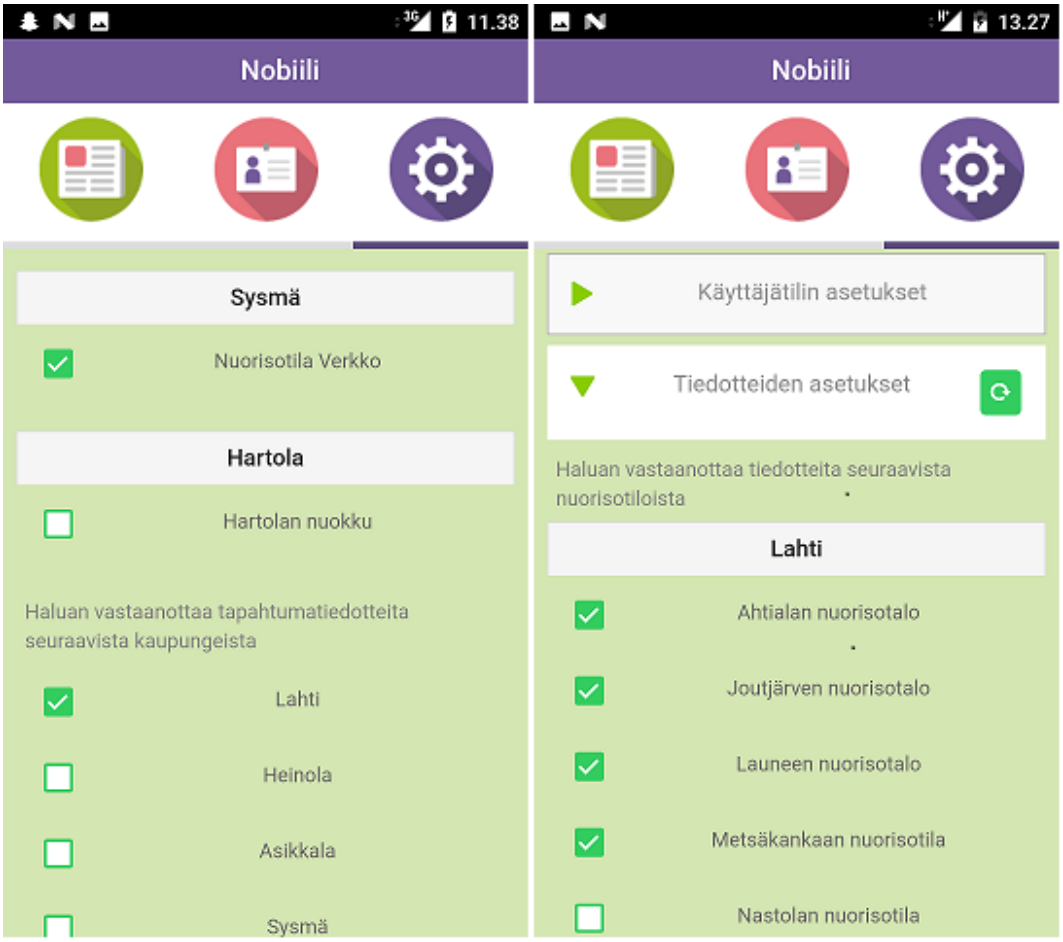
Tiedotteiden asetukset hallitsevat uutisvirran syötettä. Tiedotteiden lisäksi myös Menoinfosta saatavien tapahtumia voi suodattaa valitsemalla halutut kunnat listasta. Valikon oikealla puolella olevasta nappulasta sovellus päivittää kaupunki- ja nuokkulistan hakemalla uusimmat tiedot palvelimelta. Tietojen hakuun ja parsimiseen tehtiin palvelu, mikä tallentaa tiedot Local storageen.

```
var info = response.data;
 // Jos inffo on haettu aikaisemmin
 if($localstorage.getObject('info').constructor === Array){
     var currentInfo = $localstorage.getObject('info')[0];
     //Päijät-Häme jne...
     info.forEach(function(finLand){
         console.log(currentInfo);
         finLand.__children.forEach(function(area){
             var result = currentInfo.__children.filter(function (chain) {
                     return chain.title === area.title; })[0];
             if(result == undefined){
             }
             else{
                 //Kaupungit....
                 area.__children.forEach(function(city){
                     var cityresult = result.__children.filter(function (chai) {
                             return chai.title === city.title; })[0];
                     if(cityresult == undefined){
                     }
                     else{
                         city.checked = cityresult.checked;
                         console.log(city);
                         // Nuokut.......
                         city.__children.forEach(function(nuokku){
                             var nuokkuresult = cityresult.__children.filter(function (chai) {
                                     return chai.title === nuokku.title; })[0];
                             if(nuokkuresult != undefined){
                                 nuokku.checked = nuokkuresult.checked;
                             }
                         })
                     }
                 })
             }
         })
         })
         $localstorage.setObject('info', response.data);
 }
 else{
     var content = [];
     findAndReplace(info,content);
     $localstorage.setObject('info', response.data);
 }
```

KUVIO 31. Ote kaupunkitietojen hakupalvelusta

Palvelimelta saatavat tiedot tulevat yhtenä puumaisena JSON-objektina. Puumaisuusuudella tässä tapauksessa tarkoitetaan, että ylimmällä tasolla, mikä on Suomi, on useita oksia, eli maakuntia, joilla on useampi oksa, eli kaupunkeja, jotka vielä sisältävät nuorisotaloja. Jo valittujen kuntien ja

nuorisotalojen haluttiin jäävän voimaan listaa päivittäessä, joten uutta haettua listaa verrataan jo olemassaolevaan, kuten kuviossa 31 näkyy. Mikäli listan haettu arvo on jo laitteen muistissa, vanhasta listasta lisätään uuteen listaan checked-lippu, mikä kertoo, onko kyseinen tietue valittuna. Kun objekti on käyty kokonaan läpi, se korvaa kokonaan laitteessa olevan vanhan objektin.



KUVIO 32. Tiedotteiden asetukset-välilehti

Näkymässä kaupunki –ja nuokkutiedot näytetään samalla puumaisella periaatteella. Kaikki tiedot ovat yhden objektin sisällä, mitä näytetään kerros kerrallaan käyttäen ng-repeat-direktiivejä. Tiedotteiden hallinnoinnin jälkeen on kaupunkilista. Uutisvirran syötteiden näyttämistä hallinnoidaan checkbox-periaatteella. Kun checkboxin tilaa vaihdetaan, scope mikä

näyttää listan, päivittyy ja tieto tallennetaan uudelleen Local storageen. Kaksisuuntaisen tiedon sitomisen vuoksi objektia ei tarvitse muokata ohjaimen päässä manuaalisesti ollenkaan.

```
.controller('SettingsCtrl', function($sce,getInfo,$ionicPlatform,$localstorage,
                                     $scope, $state) {

    $scope.get_pre = function(html_code) {return $sce.trustAsHtml(html_code);}
    $scope.sovellusinfo = $localstorage.getObject('sovellusinfo');
```

KUVIO 33. HTML-elementtien kääntäminen

Sovelluksen tiedot- ja käyttöehdot -välilehti sisältävät palvelimelta haetun tiedon, mikä tallennetaan local storageen. Palvelimelta saatu tieto saadaan tekstinä, mikä sisältää HTML-elementtejä. AngularJS ei suoraan näytä HTML-elementtejä vaan näyttää tekstin sisältäen HTML-elementit tekstimuodossa tietoturvasyistä. AngularJS:n palvelu Strict Contextual Escaping (sce) suorittaa kääntämisen. Ylläolevassa kuviossa sovellusinfo-muuttuja on sidottu näkymään käyttäen ng-bind-html-direktiiviä, mikä suorittaa funktion, mikä kääntää muuttujan tekstin HTML-kelpoiseksi.

4 YHTEENVETO

Tavoitteena oli tehdä mobiiliapplikaatio, mikä korvaa fyysisen nuorisokortin ja tarjoaa uutis- ja tapahtumatietoja Lahden seudun nuorille. Lisäksi tavoitteena oli ottaa selvää, kuinka Ionic-sovelluskehys toimii ja millaisia rajoitteita sen käyttäminen tuo mobiilikehitykseen.

Tuloksena oli toimiva mobiilisovellus. Ionic osoittautui toimivaksi sovelluskehykseksi rakentaa hybrid sovellus sen helppokäyttöisyyden ja hyvän liitännäiskokoelman ansiosta. Projektin perusteella todettiin, että Ionicilla on kuitenkin rajoitteita. Suorituskyky vanhemmilla laitteilla ja etenkin 1.0 version rajallinen Windows phone -tuki saattavat olla kompastuskiviä, mikäli Ionicilla haluaa tehdä graafisempia ja monikäyttöisempiä sovelluksia.

Aikataulussa pysyttiin periaatteessa. Ensimmäinen versio saatiin julkaistua ennen nuorisotalojen aukeamista, mutta se ei sisältänyt kaikkia ominaisuuksia Windows-alustalla sekä sisälsi jonkin verran ohjelmointivirheitä. Ne kuitenkin saatiin korjattua sovelluspäivityksellä.

Tällä hetkellä sovellus on suhteellisen aktiivisessa käytössä ja sitä on ladattu pelkästään Android-laitteisiin yli 500 kertaa. Käyttäjäpalaute on ollut pääosin positiivista, jos ensimmäisestä versiosta tulleita toiminnallisuuksien puuttumista ei oteta huomioon.

Juuri tähän projektiin Ionic Framework oli hyvä valinta, koska sovellus ei vaadi raskaita toimintoja, mutta sen sijaan Windows phone-tuki on ehdoton. Toisaalta toinen varteenotettava vaihtoehto hybridisovellusten tilalle on React Native, mikä kääntää JavaScriptillä kehitetyn sovelluksen oikeasti natiiviksi pitäen suorituskyvyn parempana.

Tällä hetkellä sovelluksesta odotellaan käyttökokemuksia ja käyttäjien toiveita siitä, millaisia muutoksia sovellukseen tarvitsee vielä tehdä. Versioon 2.0 lisätään mahdollisesti ominaisuuksia, joiden oli tarkoitus tulla jo ensimmäiseen versioon, mutta jätettiin aikataulullisista syistä käyttöönottamatta. Tälläisiä ominaisuuksia ovat QR-koodi skanneri, minkä

avulla nuorisotaloilla käyntiä saisi tilastoitua myös nuorisotalojen työntekijöille sekä mahdollinen sovelluksen pelillistäminen.

LÄHTEET

Angular docs 2017. Concepts [viitattu 12.3.2017]. Saatavissa: https://docs.angularjs.org/guide/concepts

Angularjs-liitännäiset 2017. Angularjs-liitännäiset [viitattu 12.3.2017]. Saatavissa: https://ionicframework.com/docs/api/

Apache Cordova 2017. Documentation [viitattu 12.3.2017]. Saatavissa: https://cordova.apache.org/docs/en/latest/

Informationweek 2015. Mobile Applications. [viitattu 12.3.2017] Saatavissa: http://www.informationweek.com/mobile/mobile-applications/6-top-programming-languages-for-mobile-development/d/d-id/1320687?image_number=1

Ionic Cloud 2017. Ionic Cloud [viitattu 12.3.2017]. Saatavissa: https://ionic.io/cloud

Ionic Creator. 2017. [viitattu 12.3.2017]. Saatavissa: https://creator.ionic.io

Ionic Framework 2017a. Ionic Framework [viitattu 12.3.2017]. Saatavissa: https://ionicframework.com/

Ionic Framework. 2017b. CSS-kompontit [viitattu 12.3.2017]. Saatavissa: https://ionicframework.com/docs/components/

Limewire 2016. Native vs Web. [viitattu 12.3.2017]. Saatavissa: https://www.lifewire.com/native-apps-vs-web-apps-2373133

Natiivi 2017. Native mobile app [viitattu 12.3.2017]. Saatavissa: https://www.techopedia.com/definition/27568/native-mobile-app

Tutorialspoint 2017a. Ionicons [viitattu 12.3.2017]. Saatavissa: https://www.tutorialspoint.com/ionic/ionic_icons.htm

Tutorialspoint 2017b. AngularJS – MVC Architecture [viitattu 12.3.2017]. Saatavissa: https://www.tutorialspoint.com/angularjs/angularjs_mvc_architecture.htm

Tutorialzine 2015. Top Frameworks [viitattu 12.3.2017]. Saatavissa: http://tutorialzine.com/2015/10/comparing-the-top-frameworks-for-building-hybrid-mobile-apps/

W3 2017. Kolme tapaa kehittää mobiilisovellus [viitattu 12.3.2017]. Saatavissa: https://w3.fi/kolme-tapaa-kehittaa-mobiilisovellus/

W3bschool. 2017. Scope [viitattu 12.3.2017]. Saatavissa: https://www.w3schools.com/angular/angular_scopes.asp

Wikipedia. 2017. Ionic [viitattu 12.3.2017]. Saatavissa: https://en.wikipedia.org/wiki/Ionic_(mobile_app_framework)