Tiia Nyholm

HTML5-pohjaisen mobiilisovelluksen kehittäminen hybridisovelluskehyksen avulla

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

8.5.2014

Metropolia

Tekijä Otsikko Sivumäärä Aika	Tiia Nyholm HTML5–pohjaisen mobiilisovelluksen kehittäminen hybridisovelluskehyksen avulla. 38 sivua 8.5.2014
Tutkinto	insinööri (AMK)
Koulutusohjelma	mediatekniikka
Suuntautumisvaihtoehto	digitaalinen media
Ohjaajat	Projecr Owner Laura Avonius yliopettaja Kari Salo

Insinöörityön tarkoituksena oli luoda uusi mobiilisovellus hyödyntäen nykyaikaisia webteknologioita ja oppia hybridisovelluskehityksestä. Lopputuotteena oli tarkoitus tuottaa sovellus kahdelle eri alustalle. Työssä vertailtiin eri sovelluskehyksiä ja niiden soveltuvuutta mobiilisovelluksen kehitykseen.

Sovelluskehysten vertailussa tarkasteltiin sovelluskehyksen opittavuutta, soveltuvuutta tiettyyn sovellukseen ja sovelluskehyksen vakautta. Eri sovelluskehysten opittavuudessa ja laajuudessa on suuria eroja.

Insinöörityössä tehtyjen vertailujen pohjalta päädyttiin kehittämään sovellusta tuoreella sovelluskehyksellä, joka on rakennettu PhoneGap-sovelluskehyksen pohjalta.

Sovellus toteutettiin kahden hengen kehitystiimissä lyhyellä aikataululla. Sovelluksen kehityksessä kohdattiin ongelmia käyttöjärjestelmien ja sovelluskehysten päivittyessä jatkuvasti. Kehitystyössä huomattiin, että uudet ja innovatiiviset sovelluskehykset eivät välttämättä ole toimivin tapa tuottaa nopealla aikataululla sovellusta.

Työn lopputuotteena syntyi iOS-alustalle mobiilisovellus, joka toteutettiin käyttäen Steroidsia, AngularJs:ää ja Hammer.js:ää. Sovellus julkaistiin huhtikuussa 2014.

Avainsanat	AngularJS, PhoneGap, mobiilisovellus

Author Title Number of Pages Date	Tiia Nyholm HTML5 based mobile application development with hybrid framework. 38 pages 8 May 2014
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructors	Laura Avonius, Project Owner Kari Salo, Principal Lecturer

In this thesis the main target was to develop a mobile application with modern web technologies and learn how to do a hybrid application. The end product should be mobile application for two different platforms. In this thesis different frameworks and their applicability to mobile application development were compraed.

In the framework comparison frameworks were studied from the point of learnability and suitability for this particular application and the maturity of the framework.

In this thesis, the comparisons were made on the basis of the development of the application with a fresh application framework, which is built on a PhotoGap application framework.

The final product was a mobile application for iOS platform, which was carried out using steroids, AngularJs and Hammer.js.

Keywords	PhoneGap, AngularJs, mobile application

Sisällys

Lyhenteet

API	Application programming interface. Ohjelmointirajapinta, jolla eri ohjelmat voivat tehdä pyyntöjä ja vaihtaa tietoja eli keskustella keskenään.
CSS	Cascading Style Sheets. HTML-dokumenttien tyylittelykieli.
HTML5	HyperText Markup Language. Uusin versio verkkodokumenttien merkintäkielestä.
LESS	Dynaaminen tyylittelykieli, joka perustuu CSS:ään.
MVC	Model-view-controller. Ohjelmistoarkkitehtuurityyli, jonka tarkoituksena on käyttöliittymän erottaminen sovellusaluetiedosta.
SDK	Software development kit. Ohjelmistokehityspaketti, joka mahdollistaa kehittämisen tietylle alustalle.

1 Johdanto

Mobiililaitteiden käytön lisääntyessä yrityksen on hyvä miettiä omaa markkinointiaan ja asemaansa mobiilikäyttäjien keskuudessa. Yhä useampi käyttäjä käyttää internetiä mobiililaitteella. Yli 60 % suomalaisista omistaa älypuhelimen tai muun mobiililaitteen, kuten tabletin. 47 % suomalaisista 16–89-vuotiasta käyttää internetiä langattomasti mobiililaitteella. Mobiilin verkkosivuston tai natiivin sovelluksen kehittäminen on kannattavaa yritykselle, jotta se voi palvellla asiakkaitaan mahdollisimman monipuolisesti. [1.]

Natiivin mobiilisovelluksen avulla yritys pystyy kasvattamaan tunnettuuttaan ja palvelun kannattavuutta. Natiivisovelluksella voidaan luoda käyttäjälle uniikki käyttökokemus, jota pelkällä mobiilioptimoidulla verkkosivulla ei pystytä tarjoamaan. Natiivisovelluksella on myös helppo laajentaa palvelun markkinointistrategiaa ja lisätä palvelun myyntiä. Natiivisovelluksen kehittäminen eri alustoille on kuitenkin kallista, mikäli jokaiselle mobiilialustalle kehitetään oma sovelluksensa. Nykyään on mahdollista kehittää sovellus hyödyntäen perinteisiä web-teknologioita, kuten HTML5, JavaScript ja CSS.

Insinöörityössä tutkitaan natiivisovelluksen kehittämistä perinteisillä web-teknologioilla: Miten eri tekniikoiden valmiita kirjastoja voidaan käyttää hyväksi ja mitä eroja kirjastoilla on? Mitä kaikkea mobiilisovelluksen kehittämisessä tulee ottaa huomioon ja mitä eri mahdollisuuksia kehittämiseen on? Tarkoituksena on myös julkaista mobiilisovellus kahdelle eri alustalle ja oppia hybridisovelluksen kehittämisestä. Samalla pyritään kasvattamaan maksavien käyttäjien konversiota treffipalvelussa. Insinöörityön asiakkaana on Suomi24, joka on Suomen suurin verkkoyhteisö. Keskustelut ovat Suomi24:n tunnetuin ja käytetyin palvelu, mutta lisäksi sillä on myös muun muassa sähköposti-, tori- ja treffi-palvelut. Suomi24:ää käyttää noin 1,3 miljoonaa käyttäjää viikossa. Pelkästään keskusteluihin lähetetään 20 000 viestiä päivässä. [1.]

Suomi24 on perustettu vuonna 1998, jolloin Telia julkaisi sen nimellä Sirkus.com. Vuonna 2000 nimi vaihdettiin Suomi24:ään ja omistajuus vaihtui Scandinavia Online Ab:lle. Eniro osti vuonna 2001 Scandinavia Online Ab:n, jolloin Suomi24:stä tuli osa Eniro-konsernia. Aller Media Oy osti puolet Suomi24:stä Enirolta vuonna 2008 ja vuonna 2010 Aller osti loputkin Suomi24:stä, mistä lähtien se on ollut kokonaan Aller Media Oy:n omistuksessa. [2.]

Suomi24 Treffit on seuranhakupalvelu, jossa käyttäjä voi luoda profiilin, etsiä seuraa ja olla yhteydessä muihin käyttäjiin. Kirjautumaton käyttäjä voi hakea profiileita ja selata tiivistettyjä profiileita. Kirjautunut käyttäjä näkee profiilit kokonaisuudessaan. Palvelu on kuulunut Suomi24:n palveluihin alkuajoista lähtien. Alkuvuodesta 2013 Suomi24 Treffit uudistettiin kokonaan. Palvelu sai uuden käyttöliittymän, ja sen käyttämisestä tehtiin helpompaa kaikilla päätelaitteilla. Palvelulla on yli 150 000 aktiivista käyttäjää, joista enemmistö (63 %) on miehiä. Palvelun peruskäyttö on maksutonta kaikille, mutta käyttäjän on mahdollista ostaa itselleen Treffit Plus -jäsenyys, jolla saa käyttöönsä muun muassa megafonihuudot ja rajattoman viestittelyn. Käyttäjä voi hakea toisia käyttäjiä perustietojen (ikä, sukupuoli, paikkakunta) lisäksi esimerkiksi harrastusten, lemmikkien tai uskonnon perusteella. Joka kuukausi 2 500–3 000 käyttäjää poistaa profiilinsa palvelusta, koska ovat löytäneet rakkauden Suomi24 Treffeistä. [3.]

2 Mobiilisovelluskehitys

Mobiililaitteiden yleistyessä laitteen selaimen käyttäminen oli vielä vuonna 2010 yleisempää kuin mobiilisovelluksen. Kuitenkin jo vuonna 2011 sovellusten käyttäminen kiri selaimen käyttämisen ohi, ja joulukuussa 2011 mobiililaitteen sovellusta käytettiin keskimäärin 72 minuuttia päivässä, kun taas sovelluksia yhteensä noin 94 minuuttia päivässä. Vuonna 2013 selaimen käyttö oli enää 20 % mobiililaitteen käytöstä, loppu 80 % jäi erilaisille mobiilisovelluksille. [4; 5.]

Mobiilisovellukset voidaan jakaa kolmeen pääryhmään: natiivisovellus, HTML5-sovellus ja hybridisovellus. Jokaisen kehittäminen on hieman erilaista ja tarjoaa eri hyötyjä. [6; 7; 8; 9, s. 2.]

Natiivisovellus

Natiivisovellus on tietylle alustalle suunniteltu ja rakennettu sovellus. Sovellus tulee ladata laitteen muistiin laitekohtaisesta palvelusta. Sovellus kehitetään jokaiselle mobiilialustalle alustan tukemalla ohjelmointikielellä. Natiivin sovelluksen kehittäminen on hidasta ja vaatii perehtymistä eri alustojen ohjelmointikieliin. Lisäksi sovellus julkaistaan ja jaellaan jokaisella alustalla eri portaalin kautta, ja eri alustoilla on myös erilaiset laatuvaatimukset sovelluksen käyttöliittymän ja hyödyllisyyden suhteen. [8; 10.]

Apple iOS -laitteille kehittäminen tehdään Objective-C-ohjelmointikielellä, joka on laajennus C-ohjelmointikieleen. Kehittäminen tehdään Applen omalla Xcode-ohjelmistolla, eikä sovellusta voi julkaista ilman Xcodea. iOS-käyttöjärjestelmän sovelluksella on tarkka yhtenäinen graafinen ohjeistus, jota sovelluksen kehityksessä tulee noudattaa. Mikäli ohjeistusta ei noudateta, voi olla, että sovellusta ei julkaista Applen App Storessa. [11; 12.]

Android-laitteiden sovelluskehityksessä käytetään Javaa. Sovelluskehityksen aluksi on ladattava Android SDK, ja Androidin kehittäjäohjeistukset suosittelevat Eclipseä kehitysympäristöksi, mutta kehittäminen onnistuu muissakin ympäristöissä. Androidille sovellusta kehittäessä ohjeistus ei ole niin tiukka: Google Play Kauppa puuttuu lähinnä sovelluksen sisäisiin mainoksiin ja Google Playhin ilmoitettavaan nimeen ja lyhenteeseen. [16.]

Selainsovellukset

Selainsovellus eli mobiilisivusto toimii laitteen selaimessa. Sivusto on optimoitu tukemaan mobiililaitteita, jolloin käyttöliittymä eroaa työpöytäversiosta ja on usein kevyempi ja helpompi käyttää. Selainsovelluksen kehittäminen on nopeaa ja helppoa, koska sen voi tehdä yleisillä web-teknologioilla ja sama koodi toimii kaikissa laitteissa. Sivuston kehittämiseen ei tarvitse erikseen ympäristöä tai ohjelmistoja, vaan kehitys tehdään työpöytäversion tavoin. Sovellus voidaan julkaista omassa verkko-osoitteessaan, mutta useimmiten työpöytäversiosta tehdään mobiiliin skaalautuva. [7.]

Hybridisovellus

Hybridisovelluksessa hyödynnetään natiivisovelluksen ja selainsovelluksen kehitystyötapoja. Sovelluksessa käytetään natiivisovelluksen koodin lisäksi yleisiä web-teknologioita. Varsinainen kehitys tehdään HTML5-, CSS- ja JavaScript-kielillä, jolloin kehittäminen on nopeaa ja helpompaa oppia. Ohjelma käännetään käyttäen siihen soveltuvaa kehitysalustaa. [7; 14; 15.] Hybridisovelluksen etuja on se, että saadaan käyttöön natiivin mobiililaitteen käyttöjärjestelmän tarjoamat ominaisuudet. Nämä ominaisuudet mahdollistavat muun muassa laitteen kameran ja geolokaation käyttämisen. Hybridisovellus toimii laitteen sisäänrakennetussa selaimessa, mutta käyttäytyy loppukäyttäjälle natiivisovelluksen tapaisesti. Käyttöliittymä on siis vain selainnäkymä, joka

ottaa laitteen ruudun täyden leveyden ja korkeuden, jolloin selainikkunaa ei erota. [14; 15; 16.]

Jotta sovelluksen kehittäminen olisi ketterää ja nopeaa, on hyvä käyttää erilaisia kirjastoja helpottamaan kehittämistä. Eri sovelluskehysten avulla helpotetaan sovelluksen näkymien vaihtamista ja API-kutsuja. Lisäksi eri sovelluskehyksiä voidaan käyttää esimerkiksi käyttöliittymän rakentamiseen ja kosketuseleiden tunnistamiseen. [16.]

3 Hybridisovelluskehykset

Kun hybridisovellusta tehdään, ensimmäisenä päätetään, miten sovellusta lähdetään kehittämään. Kun tiedetään sovelluksen tarve ja se, mille alustoille sovellus halutaan julkaista, päätetään, kannattaako tehdä natiivi-, selain- vai hybridisovellus. Jos sovellus on suppea eikä se tarvitse mobiililaitteen natiiviominaisuuksia (kuten kameraa, lokaatiota tai reaaliaikaisia ilmoituksia), on hyvä miettiä, riittäisikö selainsovellus, joka on nopein kehittää. Päätökseen vaikuttavat myös kehittäjätiimin koko ja osaaminen eri ohjelmointikielissä. Mikäli tiimi on pieni ja erikoistuminen on vähäistä, on todennäköisesti viisainta lähteä kehittämään hybridisovellusta. Natiivisovelluksen kohdentaminen ja eri alustojen eroavaisuuksien huomioiminen on helpointa, mutta natiivisovellus vaatii eniten osaamista ja erikoistumista eri ohjelmointikieliin ja todennäköisesti eniten aikaa kehitystyöltä. [7; 9.]

3.1 PhoneGap-sovelluskehys

PhoneGap on vuonna 2008 julkaistu Adobe Systemsin omistuksessa oleva avoimen lähdekoodin HTML5-sovelluskehys. PhoneGap perustuu Apache Cordova -projektiin. Sovelluskehys mahdollistaa hybridisovelluksen kehittämisen usealle eri laitealustalle vain pienillä koodimuutoksilla. Tuetut laitealustat ovat iOS, Android ja Windows Phone 8. [16; 17.]

PhoneGap tarjoaa Javascript API-rajapinnan, jolla saadaan käyttöön laitteen natiiviominaisuudet, kuten kamera ja sijaintitiedot. Tämä rajapinta mahdollistaa sen, että sovellusta kehitetään JavaScriptillä, jolloin PhoneGap huolehtii kehittäjän koodin ja natiivien ominaisuuksien välisestä keskustelusta. [17; 18.]

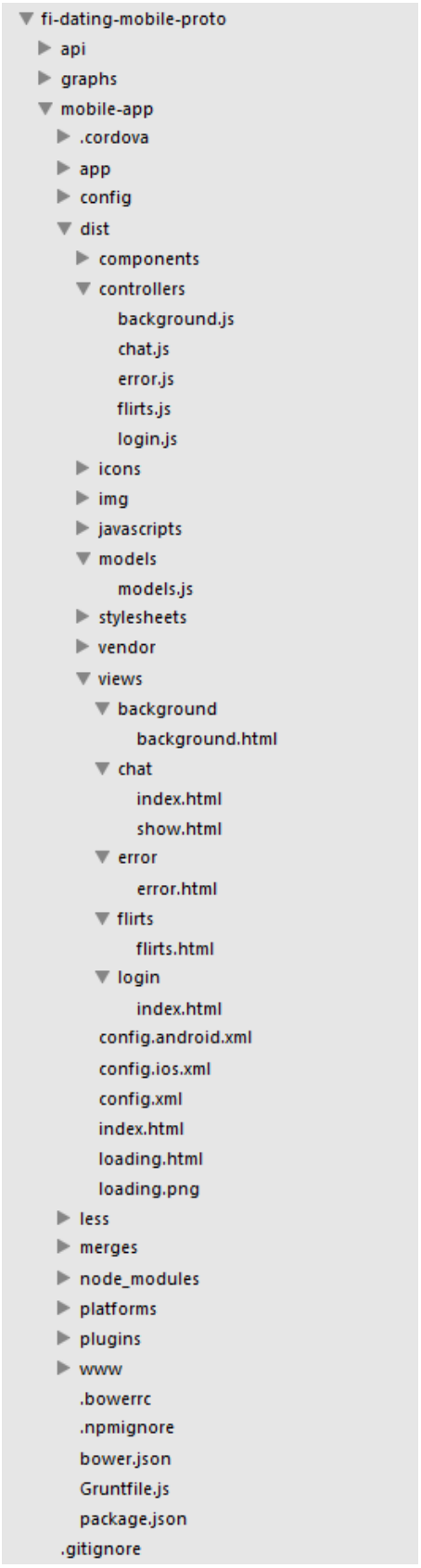
Kehitys tehdään tyypillisillä web-teknologioilla, lähestulkoon samoin kuin minkä tahansa web-sovelluksen kehitys. Sovellus toimii laitteen sisäänrakennetussa internetselaimessa, josta jätetään piirtämättä selaimen omat käyttöliittymäosat, kuten kehykset ja osoitepalkki. Näytettävä grafiikka on sama WebView-tekniikka kuin natiivisovelluksissakin (iOS-käyttöjärjestelmän Objective-C UIWebView class, Android-käyttöjärjestelmän android.webkit.WebView). Käyttöjärjestelmien WebView-näkymillä kuitenkin on pieniä eroja, jotka on hyvä ottaa huomioon käyttöliittymää suunniteltaessa. [16; 17.]

Avoimen lähdekoodin ansiosta PhoneGapilla kehittäminen on ilmaista. PhoneGap Build -palvelu tarjoaa yhden sovelluksen kehittämisen ilmaiseksi. Mikäli sovelluksia halutaan kehittää enemmän, on PhoneGap Build -palvelun hinta 9,99 dollaria kuukaudessa, jolloin sovelluksia voidaan kehittää 25. [19.]

3.2 Steroids-sovelluskehys

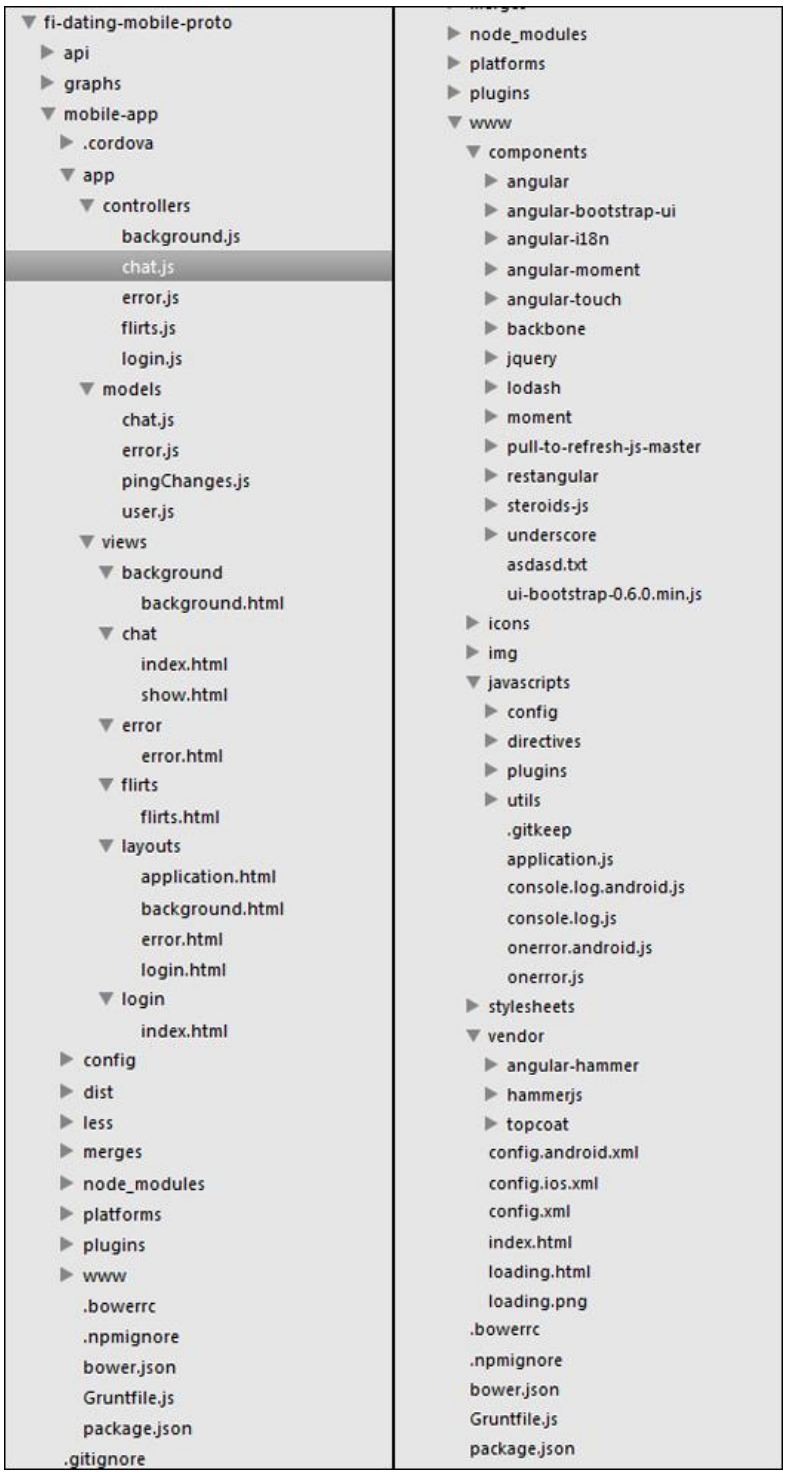
Steroids on suomalaisen AppGyverin kehittämä mobiilikehitysalusta, joka perustuu PhoneGapiin. AppGyver koki, että PhoneGapin suorituskykyä ja sillä sovelluksen kehittämistä voisi kehittää vieläkin paremmaksi. Steroidsin pääajatuksena oli luoda mobiilikehityksestä nopeaa ja helppoa. Steroids tukee vielä toistaiseksi vain Android- ja iOS-käyttöjärjestelmiä, joten laitetuessa se häviää PhoneGap-sovelluksille. Kuitenkin Steroidsin tuomat lisäominaisuudet ja varsinkin sovelluksen natiivituntuma ovat tärkeitä käyttökokemuksen kannalta. [20.]

Steroids toimii MVC-arkkitehtuurilla. Steroids luo automaattisesti tietyn kansiorakenteen. Sovellusta käännettäessä luodaan `dist/`-kansio (kuva 1), joka kopioidaan puhelimeen sovellusta suoritettaessa. `dist/` on sovelluksen HTML5-osuus, ja se luodaan automaattisesti, kun Steroids yhdistetään laitteeseen. [20.]



Kuva 1. Steroids-sovelluksen dist/-kansion rakenne.

Steroids mahdollistaa `www/-` ja `app`/-kansioiden käytön (kuva 2). `www/`-kansio yliajaa `dist/`-kansion suoraan. `app/` antaa mahdollisuuden organisoida sovelluksen rakenteen paremmin.



Kuva 2. app/- ja www/-kansioden rakenne MVC-arkkitehtuurin mukaisesti sovelluksessa.

AppGyver suosittelee rakentamaan sovelluksen MVC-arkkitehtuurin `app/`-kansion sisään. `app/`–kansion alle luodaan omat kansionsa näkymille, käsittelijöille ja malleille. [20.]

Näkymät

Steroidsin `app/views/layouts`-kansio sisältää sovelluksen eri näkymien käyttöliittymät. Peruskäyttöliittymä on aina `layouts/application.html`, jota käytetään, mikäli tarkemmin määriteltyä ulkoasua näkymälle ei löydy. Steroidsissa on sisäänrakennettu `<%= yield.view %>` -koodi, joka osaa hakea kohtaan kuuluvan näkymän, jolloin `<html>` ja `<head>` -tageja ei tarvitse ladata joka kerta erikseen (kuva 3). Steroids osaa nimien perusteella hakea oikean ulkoasun: mikäli halutaan käyttää esimerkiksi perusulkoasua muualla paitsi etusivulla, luodaan `app/views/layouts/etusivu.html` ja sijoitetaan etusivun alla olevat tiedostot `app/views/etusivu`-kansion sisälle. [20.]

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="utf8">
5    <title>Esimerkki</title>
6
7    <script src="http://localhost/cordova.js"></script>
8    <script src="/components/steroids-js/steroids.js"></script>
9
10 </head>
11 <body>
12
13
14   <%= yield.view %>
15
16 </body>
17 </html>
```

Kuva 3. Steroidsin perusulkoasu, johon näkymä haetaan.

Käsittelijät

Steroidsin `app/controllers/`–kansio sisältää käsittelijät eli näkymäkohtaiset JavaScript-tiedostot. `app/controllers/`–kansio rakentuu samoin kuin `app/views/`, jolloin käsittelijöiden täytyy olla samannimisiä niihin kuuluvien näkymien kanssa, jotta sovellus osaa hakea ne oikein. Steroids ymmärtää `<%= yield.controller %>` -

koodin, jolla oikea käsittelijä voidaan hakea `application.html`-tiedoston `<head>`-osiossa (kuva 4). Nyt etusivun käsittelijä olisi siis polussa `app/controllers/etusivu.js` ja etusivun näkymä `app/views/etusivu/index.html`. Sovellus osaa hakea ne oikein eikä sekoita muiden näkymien käsittelijöihin tai näkymiin. Suoritettaessa `$ steroids make` -komento Steroids kääntää `app/`-kansion sisällön `dist/`-kansioon, joka kopioidaan puhelimeen. `dist/views/layouts/etusivu.html`-tiedoston `<head>`-tagin sisällä olevat koodi ei sisälläkään enää `<%= yield.controller %>`-koodia, vaan koodin `<script src="/controllers/etusivu.js"></script>`.[20.]

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf8">
  <title>Esimerkki</title>

  <script src="http://localhost/cordova.js"></script>
  <script src="/components/steroids-js/steroids.js"></script>
  <script src="/controllers/<%= yield.controller %>.js"></script>

</head>
<body>


  <%= yield.view %>

</body>
</html>
```

Kuva 4. Steroidsin <head>-osio, johon oikea käsittelijä ladataan.

Mallit

Kaikki Steroids-sovelluksen mallit ovat `app/models/`−kansion sisällä. Steroids kuitenkin yhdistää `app/models/`-kansion sisällä olevat mallit yhdeksi `dist/models/models.js`-tiedostoksi. Kehittämisen kannalta mallit on hyvä pitää erillään, jolloin rakenne pysyy selkeämpänä, mutta sovelluksen kannalta mallien on hyvä olla kaikkien näkymien saatavilla. Näin ollen `app/views/layouts/etusivu.html`-tiedostossa voidaan ladata vain `models.js`-tiedosto, eikä tarvita erillistä koodia oikean tiedoston löytämiseksi. [20.] Lopullinen yksinkertainen ulkoasun hakeminen olisi kuvan 5 mukainen.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf8">
  <title>Esimerkki</title>

  <script src="http://localhost/cordova.js"></script>
  <script src="/components/steroids-js/steroids.js"></script>
  <script src="/controllers/<%= yield.controller %>.js"></script>
  <script src="/models/models.js"></script>

</head>
<body>


  <%= yield.view %>

</body>
</html>
```

Kuva 5. Yksinkertainen sovelluksen ulkoasu.

3.3 Steroids-natiivisovelluksen käyttöliittymä

PhoneGapilla toteutetut sovellukset paljastuvat helposti hybridisovelluksiksi, varsinkin eri näkymien välisissä siirtymissä. Steroids tuo sovellukseen paljon natiivisovelluksen tuntuisia ominaisuuksia. [20; 22.]

Natiivinavigaatio

Suurimmaksi natiivisovelluksen tuntua tuovaksi ominaisuudeksi AppGyver ilmoittaa näkymien väliset siirtymät. Siinä missä PhoneGap-sovellus vaihtaa webviewissä näkyvää sivua ilman mitään animaatiota, tuo Steroids mahdollisuuden vaihtaa näkymää natiivisovelluksen tavoin animaatiolla. Steroids mahdollistaa natiivin navigaatiopalkin luomisen sovellukseen, ja sen avulla sovellus tietää, mikä webview-näkymä on sillä hetkellä aktiivinen ja edelliseen sivuun siirtyminen on mahdollista navigaatiopalkissa olevan takaisin-painikkeen kautta. Todellisuudessa Steroidsin tarjoama JavaScript vaihtaa kahta eri verkkosivua, jolloin voidaan luoda esimerkiksi kaksi eri HTML-sivua ja JavaScript-tiedosto, joka hoitaa näymien vaihtamisen (kuva 6). [20; 22.]

```
ensimmainenNakyma.html

<!DOCTYPE html>
<html>
<head>
  <script src="components/steroids-js/steroids.js"></script>
  <script src="javascripts/application.js"></script>
</head>
<body>
  <h1>Hello World</h1>
  <button onClick="steroids.layers.push(toinenNakyma);">Siirry toiseen näkymään</button>
</body>
</html>


toinenNakyma.html

<!DOCTYPE html>
<html>
<head>
  <script src="components/steroids-js/steroids.js"></script>
  <script src="javascripts/application.js"></script>
</head>
<body>
  <h1>Tämä on toinen näkymä.</h1>
</body>
</html>

application.js

steroids.view.navigationBar.show("Hello World");
var secondView = new steroids.views.WebView("toinenNakyma.html");

```

Kuva 6. Steroidsin kahden näkymän välillä vaihtaminen.

Muut natiivit käyttöliittymäominaisuudet

Navigaation lisäksi Steroids tarjoaa muitakin natiivisovelluksen tuntua tuovia ominaisuuksia. Niitä ovat esimerkiksi

- välilehdet, joiden avulla kahta tai useampaa eri näkymää voidaan vaihdella
- drawer, jolla mahdollistetaan sivusta tuotava navigaatio tai muu elementti. Android 4.4 julkisti navigation drawerin, jota käytetään useassa sovelluksessa
- modaali, joka mahdollistaa uuden sivun avaamisen natiivianimaation avulla toisen sivun päälle
- animaatiot, erilaiset natiivinomaiset animaatiot, joilla siirrytään näkymästä toiseen. Nämä animaatiot ovat yhteensopivia vain iOS-käyttöjärjestelmien kanssa eivätkä vaadi CSS:ää ollenkaan.

3.4 Steroids-kehitystyömalli

Steroids mahdollistaa projektien luomisen helposti oman käyttöliittymänsä kautta. Kun Steroids on kerran asennettu, voi komennolla `$ steroids create` luoda uuden projektin, joka automaattisesti lataa käyttäjälle eri SDK:t. [20; 22.]

AppGyver tarjoaa myös oman AppGyver Scanner -sovelluksensa, jonka avulla luotuja sovelluksia voi helposti testata omassa laitteessaan. AppGyver Scanner on ladattavissa Android- ja iOS-käyttöjärjestelmille. Suorittamalla komennon `$ steroids connect` Steroids luo sovellukselle QR-koodin, jonka voi lukea AppGyver Scannerin avulla. Laitteen ja tietokoneen, jolla kehitystä tehdään, tulee olla samassa langattomassa verkossa, jotta QR-koodi toimii. Sovellus pyörii tällöin AppGyverin Scannerissa, ja se käyttäytyy kuin lopullinen sovellus. Sovelluksen koodiin voi tehdä muutoksia, ja AppGyver Scanner tunnistaa ne ja päivittää laitteessa suoritettavan testisovelluksen automaattisesti. [20; 22.]

Steroids tarjoaa valmiita koodipohjia, joiden avulla erilaisten projektien aloittaminen on helppoa. Tällaisia koodipohjia ovat AppGyverin tekemät eri sovellusesimerkit, joilla havainnollistetaan esimerkiksi, miten laitteen kameraan päästään käsiksi tai miten natiivi navigaatiopalkki toimii. Lisäksi Steroidsiin on liitetty valmiita pohjia esimerkiksi AngularJS-sovelluskehyksellä tehtäville sovelluksille. [20;22.]

3.5 PhoneGap ja Steroids

Steroids on täysin PhoneGap-yhteensopiva, koska se on rakennettu PhoneGapin (tai Apache Cordovan) päälle. Steroids käyttää samoja API-rajapintoja kuin PhoneGap ja on laajentanut nämä API:t omalla natiivikäyttöliittymä-API:llaan. Tämä tarkoittaa, että kaikki, mikä on mahdollista toteuttaa PhoneGapilla, on mahdollista myös Steroidsilla (kuva 7). Kaikki PhoneGapin lisäosat toimivat myös Steroidsissa, ja näin ollen PhoneGapilla tehtyjen sovelluksien uudelleen rakentamisen Steroidsilla pitäisi olla helppoa. [20; 22.]

	PhoneGap	Steroids
Accelerometer	✓	✓
Camera	✓	✓
Capture	✓	✓
Compass	✓	✓
Connection	✓	✓
Contacts	✓	✓
Device	✓	✓
Events	✓	✓
File	✓	✓
Geolocation	✓	✓
Globalization	✓	✓
InAppBrowser	✓	✓
Media	✓	✓
Notification	✓	✓
Splashscreen	✓	✓
Storage	✓	✓
Calendar		✓
Native UI: Multiple WebViews		✓
Native UI: Tabs		✓
Native UI: Modal windows		✓
Native UI: Navigation bar		✓
Native UI: Native animations		✓
Native UI: Drawer		✓
DB: Built-in SQLite		✓
DB: Built-in TouchDB on iOS		✓

Kuva 7. PhoneGapin ja Steroidsin eroavaisuudet [21].

4 Sovellusprojekti

4.1 Lähtökohdat

Insinöörityön osana tehtiin Suomi24 Treffit -palvelun mobiilisovellus. Sovellus päädyttiin tekemään hybridi-sovelluksena, koska kehitystiimissä ei ollut tarpeeksi osaamista natiivisovelluksen kehittämiseen. Projektin aikataulu oli tiukka, ja sovelluksen tuli olla mahdollisimman nopeasti valmiina, joten päädyttiin kehittämiseen perinteisillä webteknologioilla, jonka ajateltiin olevan nopeampaa. Tiimin sisältä löytyi valmiiksi JavaScript-, HTML5- ja CSS-osaajia, joten aikaa ei tuhlaantunut turhan paljon uuden opetteluun. PhoneGap oli kaikille tiimin jäsenille entuudestaan tuttu ainakin nimeltä, ja ennakkotiedot siitä olivat pelkästään hyvät. Lisäksi osalla tiimin jäsenistä oli pientä kokemusta PhoneGapilla kehittämisestä, joten päädyttiin valitsemaan PhoneGap alustaksi.

AppGyverilta tarjottiin mahdollisuutta kehittää sovellusta Steroidsin avulla. Kehitystiimi sai vuorokauden ajaksi AppGyverilta henkilön esittelemään Steroidsia. Varsinkin natiivianimaatiot ja siirtymät vakuuttivat. Lisäksi kehittäminen vaikutti helpolta ja nopealta. AppGyver itse suosi AngularJS-sovelluskehystä kehittämiseen, ja osa esittelystä pohjautuikin vahvasti Steroidsin ja Angularin yhteensopivuuteen. Päätettiin ottaa Steroids kehitykseen mukaan, jotta sovelluksesta tulisi natiivimman oloinen.

4.2 Vaatimukset

Projektin lähtökohtana oli parantaa käyttäjän käyttökokemusta ja helpottaa Suomi24 Treffit -palvelun käyttöä entisestään. Mobiilisovellus mahdollistaa reaaliaikaiset ilmoitukset palvelusta suoraan käyttäjälle ja palvelee käyttäjää näin ollen paremmin kuin mobiili verkkosivusto. Samalla sovelluksesta päätettiin tehdä riisutumpi ja kevyempi kuin mobiilisivusto, jolloin kehittäminen on nopeampaa ja käyttäjän on helppo alkaa käyttää sovellusta. Sovelluksen tavoite oli myös olla lisäpalvelu nykyiseen mobiilisivustoon. Lisäpalveluna sovelluksen avulla voidaan lisätä Treffit Plus -käyttäjien määrää ja mahdollistetaan myöhemmässä vaiheessa mikro-ostojen tekeminen.

Lopputuote on sovellus, jolla käyttäjä voi lähettää viestejä jo olemassa oleville keskustelukumppaneille ja vastaanottaa viestejä niin uusilta kuin olemassa oleviltakin keskus-

telukumppaneilta. Sovelluksella voi lukea olemassa olevia viestiketjuja, mutta ei aloittaa uusia. Uusien viestiketjujen aloittaminen päätettiin rajata pois ensimmäisestä versiosta, koska uuden viestiketjun aloittamiseksi sovellukseen tulisi integroida myös haku. Haun kehittäminen sovelluksen ensimmäiseen versioon veisi liian kauan aikaa, ja sovelluksen ensimmäinen versio oli tarkoitus toteuttaa lyhyenä projektina. Lisäpalveluna sovellukseen haluttiin tuoda push-ilmoitukset, joilla käyttäjä saa reaaliajassa uudet viestit mobiililaitteeseensa. Push-ilmoitukset tulevat olemaan sovelluksen suurin hyöty ja eroavaisuus verrattuna mobiilisivustoon.

Päätettiin, että sovellus on ensimmäisessä versiossaan vain lisäpalvelu Treffit Plus -asiakkaille eikä palvelua voi käyttää ilman Treffit Plus -pakettia. Näin ollen sovelluksen ei tarvitse olla maksullinen sovelluskaupassa ja kynnys sen kokeilemiseen on matalampi.

Käyttäjätilastojen mukaan Suomi24 Treffeillä on iOS-käyttäjiä eniten, mutta Android- ja Windows Phone -käyttäjiäkin on. Pelkän iOS-sovelluksen kehittämistä harkittiin, mutta kehittäjätiimissä ei kuitenkaan ollut ketään, jolla olisi ollut tarpeeksi osaamista natiivisovelluksen kehittämisestä iOS-alustalle. Päätettiin tehdä hybridisovellus, joka mahdollistaa sovelluksen kääntämisen kaikille alustoille ja kehittäminen tulisi olemaan nopeampaa ja matalakynnyksisempää oppia.

4.3 Työkalujen valinta

Päädyttyämme kehittämään sovellusta Steroidsin avulla, oli päätettävä, mitä sovelluskehyksiä tarvittiin kehitystyöhön. Oli selvää, että tarvittiin jokin MVC-sovelluskehys sovelluksen taustalle hoitamaan API-kutsuja ja näkymien vaihtumista. Taustasovelluskehyksen tuli olla nopeasti opittava ja helppokäyttöinen eikä liian raskas sovelluksen tarkoitukseen.

Mahdollisia taustalla toimivia JavaScript-sovelluskehyksiä on kehitetty todella paljon, joten aloitin selvittämällä, mitä eroja niillä on, mihin tarkoitukseen kukin sovelluskehys soveltuu parhaiten ja kuinka nopeasti se on opittavissa. Selvityksen jälkeen päädyin kolmeen mahdolliseen sovelluskehykseen: AngularJS, Backbone.js ja Ember.js. Nämä kolme sovelluskehystä ovat ne, joita useimmiten käytetään vähänkin isompien sovel-

lusten kehittämiseen ja joista on paljon kiistelty. Lisäksi CanJS:ää tunnutaan käyttävän jonkin verran. Jätin CanJS:n pois vertailusta, koska se ei vaikuttanut lainkaan soveltuvan sovelluksen tarpeisiin. [23; 24; 25.]

MVC-sovelluskehystä valittaessa on tärkeää miettiä, mitä kaikkia ominaisuuksia se mahdollistaa ja mitä niistä tarvitaan juuri kyseisessä projektissa. Muutamia yleisiä tarvittavia asioita ovat näkymien ja käsittelijöiden väliset sidokset ja keskustelu (binding, esimerkiksi lomakkeiden käsittely ”lennosta”), reitittäminen (selaimen osoitteeseen liitettävät osat ja niiden kuuntelu sekä sen mukaan näkymän muuttaminen) ja osittaiset näkymät (näkymät näkymien sisällä). [22.]

Kehitettäessä web-teknologioilla ei ole itsestään selvää, että mobiililaite tunnistaa käyttäjän erilaiset kosketuseleet ja ruudun koskettamisen. Tarvitaan sovelluskehyksen tunnistamaan ja käsittelemään käyttäjän kosketuseleet. Kosketuksen tunnistamiseen ei ollut saatavilla kovinkaan monta sovelluskehystä, joten päädyin vertailemaan Hammer.js:ää ja Zepto.js:ää. [22.]

Sovelluskehysten vertailu

Pohdimme kehitystiimin kanssa, millaista osaamista tiimistä jo löytyi ja mitä sovelluksen kannalta tarvitsimme. Tärkeimpinä asioina pidettiin uuden sovelluskehyksen opittavuutta ja joustavuutta juuri kehitteillä olevan sovelluksen tarpeisiin. Tiimistä löytyi valmiiksi Backbone-osaaja, jonka kanssa kävin keskustelua pääosin Angularin ja Backbonen väliltä. Angularin, Backbonen ja Emberin vertailu tehtiin lähinnä erilaisten luotettavien blogikirjoituksien ja artikkeleiden pohjalta sekä itse vertailemalla sovelluskehysten ominaisuuksia. Kaikki kolme ovat avoimen lähdekoodin sovelluskehyksiä, joten siihen ei tarvinnut puuttua vertailussa. [23; 24; 25.]

Ember.js

Ember on verrattain uusi sovelluskehys. Se on julkaistu vuonna 2011 nimellä Amber.js ja vaihtunut sen jälkeen Ember.js:ksi. Amber.js sekoitettiin Amber Smalltalk -nimiseen JavaScript-kirjastoon, joten Amber.js:n kehittäjät päättivät vaihtaa nimeä heti alussa. Huolimatta nuoresta iästään Ember.js on noussut nopeasti kohtuulliseen suosioon. Ember oli vielä alkutekijöissään lokakuussa 2013, kun aloitimme oman projektimme.

Dokumentaatio ei ollut vielä kovin kattavaa, versio 1.0 oli juuri julkaistu elokuussa 2013. [22; 26; 27.]

Ember on laajennettavissa eri lisäosilla, ja siihen on helppo liittää muita JavaScript-kirjastoja tukemaan tehtäviä, eikä kaikkea tarvitse koodata itse. Kuitenkin Emberillä on tarkka tapa, miten asiat pitää laajentaa, ja tämän oppiminen ja ymmärtäminen voi olla hankalaa. Mikäli asioita ei tee juuri niin kuin Emberin kehittäjät ovat tarkoittaneet, voi olla, että sovellus ei toimi ollenkaan. [23; 24; 25; 27.]

Emberin oppimiskynnystä kuvataan vaikeaksi. Siinä on paljon asioita, jotka pitää tehdä juuri eikä melkein, niin kuin ohjeistetaan. Oppimisen kannalta tämä tuottaa vaikeuksia alussa, ja useat kehittäjät ovatkin sitä mieltä, että Emberistä puuttuu alusta kokonaan helppouden luoma ”wow-efekti”. Emberin uudistettua dokumentaatiota kuitenkin kehutaan paljon, ja sen avulla Emberin rakenteen oppimisen ei pitäisi olla kovin vaikeaa. Emberin opittavuutta alussa kehutaan myös, koska kun kaikkien kehittäjien on pakko tehdä asiat samalla tavalla, myöhemmin se helpottaa sovelluksen hallittavuutta. [23; 24; 27.]

Alun oppimisvaiheen jälkeen Emberillä kehittäminen on lähteiden mukaan helppoa. Siinä on paljon sisäänrakennettuja ominaisuuksia, jotka tekevät asioita valmiiksi ja helposti lisättäväksi. [23.]

JavaScript-kirjastona Ember on todella suuri, mutta tämä myös mahdollistaa kaikkien hienojen ominaisuuksien kattamisen ilman lisäosia. Useimmiten sovelluskehyksiä on laajennettava kuitenkin muillakin JavaScript-kirjastoilla, mutta Emberin väitetään sisältävän jo valmiiksi suurimman osan ominaisuuksista. Emberissä on pakko olla mukana jQuery, mikä selittää paljon ladattavan kirjaston kokoa. [23; 24; 27.]

Emberin ympärillä olevaa yhteisöä kehutaan myös hyväksi ja laajaksi. Yhteisön tuoma tieto ja nopea vastaaminen on tärkeää, varsinkin silloin, kun kehittäessä ilmenee ongelmia. Emberillä on valmiiksi kattava dokumentaatio ja esimerkkejä, minkä lisäksi Ember-kehittäjät ovat StackOverflow-palvelussa vastanneet moneen ongelmaan. [23; 24; 25.]

Backbone.js

Backbone on hieman vanhempi kuin Ember, se julkaistiin vuonna 2010. Backbonella on oma laaja ja uskollinen käyttäjäkuntansa. Backbone on listannut paljon sovelluksia, jotka käyttävät Backbone.js:ää, muun muassa Sony Entertainment Network ja Nokia Profiles by Nokia. [23; 24; 25; 27.]

Projektin alkaessa lokakuussa 2013 Backbonen uusin versio oli 0.9.10, joka oli julkaistu tammikuussa. Backbonen kehitystiimi ei ollut päivittänyt dokumentaatioita, ja tuki vaikutti vähäiseltä. Versio 1.0.0 julkaistiin kuitenkin marraskuussa 2013, jolloin projektin kehitystyö oli jo ehtinyt alkaa. [28.]

Backbonen käyttäjät kehuvat paljon sen laajennettavuutta. Backbone.js ei juurikaan vaikuta siihen, miten tai mitä muita JavaScript-kirjastoja sen kanssa käytetään. Backbone onkin todella sopeutuvainen eri tyyleille kirjoittaa ja kehittää sovellusta. Backbonea pidetään myös helposti opittavana, koska se tosiaan antaa käyttäjän itse valita sovelluksen rakenteen. Kuitenkin seinä saattaa nousta nopeastikin pystyyn, kun Backbonen päälle on pakko ottaa jokin muukin sovelluskehys hoitamaan asioita. Helpon opittavuuden takia Backbonen kanssa joutuu kirjoittamaan eniten itse niin sanottua pohjakoodia (engl. boilerplate code), jolloin kehittäessä sovellusta täytyy itse tietää ja päättää sovelluksen rakenteesta. Jossain tapauksissa sovelluksen rakenteen voi helposti sotkea heti alussa, mikäli pohjakoodin tekemisestä ei ole tarpeeksi osaamista. Mikäli pohjakoodin kirjoittamisessa tai sovelluksen muussa kehittämisessä tulee ongelmia, on Backbonen yhteisöä kehuttu kaikista eniten. Internetistä löytyy Backbonelle tutoriaaleja ja foorumeita kaikista eniten. [23; 24; 25.]

Backbone on verrattavista sovelluskehyksistä kaikkein kevyin. Se tarjoaa kaiken tarvittavan, mutta ei mitään ylimääräistä. Tämä vaikuttaa siihen, että Backbonen kanssa on pakko ottaa käyttöön muitakin JavaScript-kirjastoja, joiden valitsemiseen saa taas käytettyä aikaa. Kuitenkin suuren yhteisön ansiosta melko varmasti joku muu on ollut kehittämässä samankaltaista sovellusta ja miettinyt valmiiksi, mitkä kirjastot ovat parhaita. [23; 24; 25.]

AngularJS

AngularJS on Googlen kehittämä sovelluskehys, joka julkaistiin vuonna 2009. Näin ollen Angular on kolmesta verrattavasta sovelluskehyksestä vanhin, ja sillä onkin tämän takia ehkäpä suurin käyttäjäkuntakin. Angularia käyttävät suuretkin yritykset, ja esimerkiksi PlayStation3:n Youtube-sovellus on rakennettu Angularilla. [23; 24; 25; 29.]

Koska ylläpitäjä on Google, ei ole yllätys, että Angularin päivitys on tasaista ja luotettavaa. Lokakuussa 2013 projektin alkaessa Angularista oli juuri ilmestynyt versio 1.2.0 [30].

Angular toimii laajennusten kanssa pitkälti samoin kuin Emberkin: asiat pitää tehdä niin kuin sovelluskehyksellä on tarkoitettu tehtävän tai ne eivät välttämättä toimi ollenkaan. Kuitenkin Angular on laajennettavissa helposti, kunhan noudattaa ohjeistusta tarkasti. Useimmat kehittäjät kehuvat Angularin oppimiskynnystä aluksi ja sitä, kuinka helpolta kehittäminen sillä tuntuu. Kuitenkin mitä syvemmälle kehittämiseen menee, sitä monimutkaisemmaksi se muuttuu. Angularilla alkuun pääseminen onkin käyttäjien mielestä helpointa, mutta jatkokehitys vaativampaa. Angular on kehitetty niin, että käyttäjien tulee seurata tarkkoja ohjeita, mikä saattaa aiheuttaa myöhemmin ongelmia. [23; 24; 25.]

Google on dokumentoinut Angularin todella hyvin, ja pelkästään Angularin omalta sivustolta löytyy kattavia tutoriaaleja, puhumattakaan kaikista muista Angularille tehdyistä tutoriaaleista. Angularin erikoisuutena ovat niin sanotut kaksisuuntaiset sidokset (two way bindings), jotka mahdollistavat näkymän ja käsittelijän välisen keskustelun kumminkinpäin. Nämä sidokset ovat yksi haastavimmista asioista ymmärtää, mutta kun niitä oppii käyttämään on, kehittäminen helppoa ja upeiden toimintojen tekeminen mielekästä. [23; 24; 25; 29.]

Tiukan koodiohjeistuksen takia Angularia pidetään myös hyvänä sovelluskehyksenä aloittaa, mikäli aikaa oppimiseen ei ole paljoa. Koodista tulee automaattisesti puhdasta ja hyvin järjestettyä, kun muita vaihtoehtoja ei ole. Angular on myös tiukasti sitonut käyttäjänsä MVVM-arkkitehtuuriin. [23; 24; 25.]

Yhteisö Angularin ympärillä on suuri, onhan Angular ollut kauemmin olemassa kuin Backbone tai Ember. Valmiit ja selkeät dokumentaatiot helpottavat kehittämisen aloittamista, ja Angular-sovelluksia on olemassa niin paljon, että jokaiseen ongelmatilanteeseen on varmasti jo valmiiksi vastaus. [23; 24; 25.]

Kooltaan Angular on pienempi kuin Ember, mutta ei yllä Backbonen keveyteen kuitenkaan. Angular on näistä kolmesta ainut sovelluskehys, joka ei tarvitse rinnalleen mitään muita kirjastoja toimiakseen, vaikka niitä voikin käyttää. [23; 24; 25.]

Kosketuseleiden hallinta

Kosketuseleiden tunnistamiseen ja niiden kanssa työskentelemiseen vertailuun valittiin Hammer.js ja Zepto.js. Tiimin toinen jäsen oli käyttänyt Zepto.js:ää edellisissä projekteissaan Backbone.js:n kanssa. AppGyver taas tarjoaa esimerkkejä ja suosittelee käyttämään Hammer.js:ää. [22.]

Zepto.js on laajempi ja korvaa jQueryn kokonaan, Hammer.js taas on todella kevyt, eikä tarvitse rinnalleen mitään muita komponentteja. Zeptolla on mahdollista tehdä paljon muutakin kuin hallita kosketuseleitä ja niiden tuottamia toimintoja, koska se on enemmän jQueryn kaltainen laaja kirjasto. Zepton kosketuseleiden tunnistaminen jää suurimmaksi osaksi pelkästään käyttäjän zoomauseleen tunnistamiseen. [31; 32; 33.]

Zepto.js:n dokumentaatio ja alustatuki on laaja, mutta se on keskittynyt paljon muuhun kuin kosketuseleisiin. Zepto.js on kehitetty palvelemaan web-sovelluksia, joita on tarkoitus käyttää niin mobiililaitteilla kuin pöytäkoneillakin. Kooltaan Zepto.js ei ole kovin suuri, mutta ei niin kevytkään kuin Hammer.js: Zepto.js on ladattaessa noin kahdeksan kilobitin kokoinen. [32.]

Hammer.js vaikutti heti paremmalta vaihtoehdolta projektiimme, koska emme tarvinneet kirjastolta muuta kuin kosketuseleiden tunnistamisen tuen. Hammer.js keskittyy pelkästään kosketuseleisiin ja tarjoaa laajan tuen erilaisille kosketustapahtumille ja eri selaimille (taulukko 1) [34.]

Taulukko 1. Hammer.js:n tukemat kosketuseleet eri alustoilla [34].

	iOS	Android	Windows Phone
Napautus	x	x	x
Kaksoisnapautus	x	x	x
Pitkä napautus	x	x	x
Pyyhkäisy	x	x	x
Monipistetunnistus	x	x	x

Mobiilialustojen lisäksi Hammer.js toimii useimmissa selainten työpöytäversioissa. Sen käyttöä kehutaan helpoksi, eikä sen liittäminen projektiin vaadi paljoa. Lisäksi se on ladattaessa vain kolmen kilobitin kokoinen. [34; 35.]

Hammer.js on EightMedian kehittämä ja yrityksen GitHub-tilillä on laaja dokumentaatio, kuinka Hammer.js:ää käytetään. Kehittämisen aloittaminen on tehty helpoksi aloitusdokumentaatiolla. [34.]

4.4 Sovelluskehysten valinta

Näkymien hallinta

Näkymien ja reitittämisen hallintaan käytettävän sovelluskehyksen päättäminen oli tärkein ja kehittämiseen eniten vaikuttava tekijä. Sovelluskehysten vertailemisen jälkeen

Ember vaikutti Backbonen ja Angularin rinnalla tuntemattomalta ja vähän käytetyltä sovelluskehykseltä, minkä takia pudotin sen pois pohdinnoista. Backbonesta tiimistä löytyi jo valmiiksi osaamista, mutta sen elinkaari ei ollut tarpeeksi luotettavan oloinen, koska melkein vuoteen kehittäjätiimi ei ollut päivittänyt sovelluskehystä ollenkaan.

Keskusteltuamme tiimin kesken päädyimme valitsemaan Angular.js:n. Angular on tarpeeksi hyvin laajennettavissa ja soveltuu monipuoliseen kehittämiseen. Tiimin jäsenet eivät ennestään osanneet Angularia, mutta vaikka uuden opetteleminen vie aikaa, luotimme sen olevan pidemmän päälle kannattavampaa. Angularin alun oppimiskynnystä pidetään niin matalana, että luotimme sen helpottavan kehitystyön aloittamista.

Steroids mahdollistaa RestAngularin käytön, joten päätimme ottaa sen käyttöön RestFul APIn kanssa työskentelyyn.

Kosketuseleiden hallinta

Kosketuseleiden hallitsemiseen valittiin Hammer.js, koska Zepto.js oli aivan liian laaja ja monipuolinen tarpeisiimme. Hammer.js:llä eleiden tunnistaminen ja käyttö on tehty yksinkertaiseksi.

Angular.js:n ja Hammer.js:n valintaan vaikutti se, että AppGyver oli valmiiksi tehnyt omia esimerkkejään Steroidsin käytöstä Angularin ja Hammerin kanssa, jolloin saatavilla oli valmiita koodiesimerkkejä. Lisäksi AppGyverin kehittäjät osaisivat tarvittaessa auttaa Steroidsin ja näiden kahden sovelluskehyksen kanssa kehittämisessä.

Käyttöliittymä

Käyttöliittymän puolella päädyttiin käyttämään Twitter Bootstrapin LESS-versiota. Twitter Bootstrap on käytössä myös palvelun selainversiossa, joten sen avulla käyttöliittymästä saataisiin helposti yhtenäinen. Esimerkiksi lomakkeiden ja nappuloiden tyylit olivat valmiiksi määritelty palvelun selainversiota varten.

Bootstrapin LESS-versio nopeutti ja kevensi kehitystyötä. Twitter Bootstrap on valmiiksi rakennettu niin, että yhden päätiedoston avulla voi kääntää ja koota valitut LESS-tiedostot yhdeksi tyylitiedostoksi.

Push-ilmoitukset

Sovelluksen pääominaisuutena tulisivat olemaan Push-ilmoitukset, joiden avulla käyttäjä saa reaaliaikaisesti mobiililaitteeseensa tiedon, milloin uusi viesti on saapunut, ilman että sovellusta varsinaisesti tarvitsee avata. Push-ilmoituksen voi lisätä laitteen lukitusnäytölle tai ylänavigaatioon, jolloin se näkyy samoin kuin esimerkiksi vastaamatta jäänyt puhelu tai saapunut tekstiviesti. Käyttäjä voi itse valita, miten ilmoitus näkyy laitteeseen ja pitääkö laite ääntä viestin saapuessa. Push-ilmoitusten hallitsemiseen tarvitaan aina ulkopuolinen palvelu, joka hallitsee ilmoitusten lähettämisen käyttäjän mobiililaitteeseen. [36; 37.]

Urban Airship on perustettu 2009, ja nykyään se on yksi suurimmista push-ilmoituspalveluiden tarjoajista. Palvelu on hyvin dokumentoitu, ja sen käyttöönotto on tehty helpoksi. [38.]

Tiimillä ei ollut mitään kokemusta entuudestaan push-ilmoitusten kehittämisestä ja hallitsemisesta. Steroids tarjosi mahdollisuuden helposti lisätä Urban Airship -lisäosan sovellukseen sen julkaisemisvaiheessa. [22.]

Urban Airshipillä on ilmainen versio, jolla voi lähettää ja vastaanottaa 1 000 000 ilmaista ilmoitusta. Muusta hinnoittelusta on aina oltava yhteydessä Urban Airshipin myyntiin [37.]

Sovelluksen kehitys voitiin aloittaa seuraavilla sovelluskehyksillä:

- Steroids, pohja koko sovellukselle
- Angular.js, näkymien, API-kutsujen ja reitityksien hallinta
- Hammer.js, kosketuseleiden hallinta
- Twitter Bootstrap, käyttöliittymäkomponentit, kuten nappuloiden tyylit
- Urban Airship, push-ilmoitusten hallinta.

4.5 Toteutus

AppGyver mahdollistaa sovelluksen eri koontiversioiden kääntämisen oman pilvipalvelunsa AppGyver Cloudin kautta. Cloudin käyttöön on selkeät ohjeet, joita noudattaa. Sovelluksesta voidaan koota kolme eri versiota AppGyverin Cloudin kautta: räätälöity Steroidsskanneri, Ad Hoc ja lopullinen tuote. [39.]

- Räätälöity Steroidsskanneri

Tavallisesti sovellusta kehitettäessä ladataan AppGyverin oma skannerisovellus, jolla voidaan testata sovellusta mobiililaitteessa. Perusskannerisovelluksessa ei kuitenkaan voi testata sovelluksen lisäosia, kuten push-ilmoituksia.

- Ad Hoc

Ad Hoc -koontiversio on käytännössä sama kuin tuotantoversio, mutta sen pystyy asentamaan laitteeseen ilman Applen omaa sovelluskauppaa. Ad Hoc -versio vaatii, että Applen developer-keskuksen sertifikaatteihin on merkitty, mihin laitteisiin asentaminen on sallittua.

- Lopullinen tuote

Tuotekoontiversio on sovelluksen lopullinen versio, joka ladataan Applen AppStoreen hyväksyttäväksi ja loppukäyttäjien ladattavaksi. [39.]

Push-ilmoitusten käyttöönottaminen

Urban Airshipin ja sovelluksen välinen yhteys täytyy konfiguroida, jotta push-ilmoitukset tulevat oikein perille sovellukseen. Urban Airshipilta saadaan eri koontiversioille tarkoitetut avaimet, jotka tarvitaan sovelluksen konfiguroimiseen oikein. Nämä avaimet laitetaan niin palvelimen päähän, joka lähettää ilmoituksia, kuin sovelluksen `.config`-tiedostoon. Lisäksi Applen provision-profiilitiedostoon täytyy sallia push-ilmoitukset. [40.]

Eri koontiversioiden luominen

AppGyver on tehnyt koontiversioiden luomisen melko helpoksi, mutta eri vaiheiden toteuttamisessa saa olla tarkkana. [39.]

Applen sertifikaattien kanssa saa tehdä töitä hieman enemmän. Applen sertifikaatteja tai provision-profiileita ei voi luoda millään muulla kuin Applen OS X -käyttöjärjestelmillä. Sovelluksen kokoamiseen tarvitaan Applelta .p12-sertifikaattitiedosto ja .mobileprovision-tiedosto. Näiden tiedostojen luomiseen tarvitaan Applelta ensin distribution-sertifikaatti, jonka avulla muut tiedostot voidaan luoda. Kun Applen kehittäjäkeskuksessa on saatu luotua sertifikaatti, ladataan .cer-tiedosto koneelle. Tämä tiedosto avataan Applen Keychain Access -ohjelmassa, jolla voidaan luoda ja viedä ulos .p12-tiedosto. [39.]

Tämän jälkeen Applen kehittäjäkeskuksesta voidaan hakea .mobileprovision-tiedosto, joka määrittää, mihin laitteisiin sovelluksen voi asentaa. Mikäli ollaan kokoamassa Ad Hoc -versiota, tarvitaan .mobileprovision, johon on määritelty sallittujen iOS-laitteiden UDID-koodit. Mikäli kootaan lopullisen tuotteen versiota, voidaan luoda yleinen provision-profiili. [39.]

AppGyverin Cloud-palvelussa on erikseen iOS-versioon kokoamiseen tarkoitettu sovellus, johon äsken luodut tiedostot ladataan. Lisäksi määritellään sovelluksen nimi, versio ja Applen pyytämä *Bundle ID*. Jokaiselle koontiversiolle ladataan oma .mobileprovision-tiedostonsa. [39.]

Koontiversioon laitetaan lisäosat mukaan liittämällä lisäosan github-projekti. Lisäosat ovat usein PhoneGap-lisäosia (vaikkakin AppGyver tarjoaa myös oman lisäosapalvelunsa). Urban Airship tarjoaa PhoneGap-lisäosan, joten tarvitsee vain lisätä sen GitHub-osoite AppGyver-pilvipalveluun ja konfiguroida sovellus käyttämään oikeaa sovellusavainta. [39.]

Cloudissa määritetään, missä orientaatiossa sovellusta on mahdollista käyttää milläkin laitteella.

Sovellus ladataan AppGyverin palvelimelle suorittamalla komentorivillä komento `steroids deploy`, minkä jälkeen voidaan luoda eri koontiversioita. Cloud-palvelu automaattisesti yhdistää palvelimelle sijoitetun uusimman version sovelluksesta ja määritel-

lyt ominaisuudet ja kääntää ne joko halutusti Ad Hoc -versioksi tai lopulliseksi tuotteeksi. AppGyverin palvelin käsittelee kääntämistä hetken, minkä jälkeen valmis paketti on ladattavissa. [39.]

4.6 Testaus

Sovelluksen kiireellisen aikataulun vuoksi ei pystytty järjestämään erillisiä käyttäjätestauksia, joten sovellus testattiin yrityksen sisäisesti. Halukkaat testaajat ilmoittivat omien laitteidensa UDID-koodin, jolla mobiililaitteen pystyi lisäämään Applen mobileprovision-tiedostoon. Tämän jälkeen laitteisiin oli mahdollista asentaa sovelluksen Ad Hoc -versio.

Testeissä oli tarkoitus lähinnä varmistaa, että päätoiminnallisuudet toimivat, niin kuin ne oli suunniteltu. Push-ilmoitusten lähettäminen ja vastaanottaminen laitteisiin huolestutti eniten.

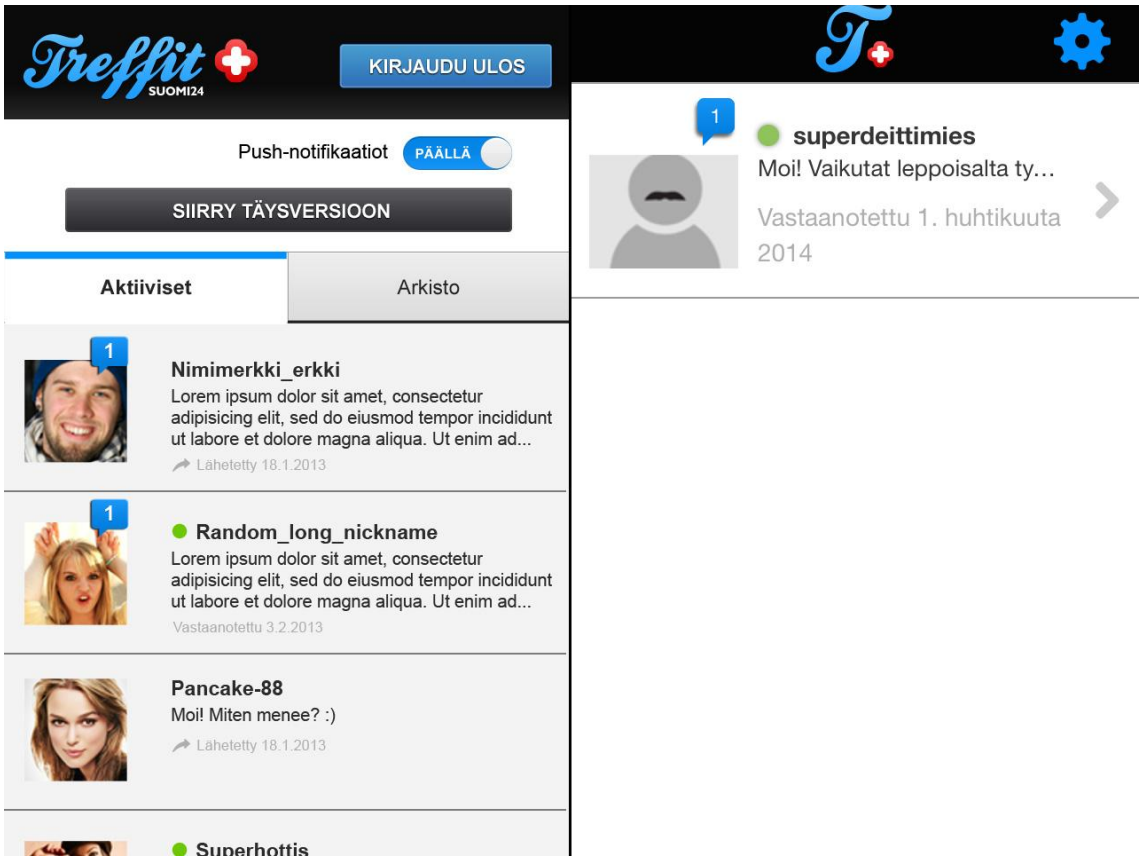
Jokaisella testaajalla tuli olla oma Suomi24 Treffit -tunnus, jolla sovellusta pystyy käyttämään. Testaajien tunnuksille annettiin Treffit Plus -ominaisuus, jota ilman sovellus on hyödytön. Testaus järjestettiin vapaamuotoisesti kehittäjätiimin kesken, ilman erillistä testaajan seuraamista. Yksikään testaajista ei ollut nähnyt tai käyttänyt sovellusta ennestään. Testaajille kerrottiin vain, että sovelluksella pystyy viestittelemään jo olemassa olevissa keskusteluketjuissa eikä uusia keskusteluja voi aloittaa.

Testausvaiheessa jokainen testaaja sai vapaasti käyttää sovellusta ilman minkäänlaista alkuohjaamista. Vuorokauden testaamisen jälkeen testaajat totesivat käyttöliittymän olevan yksinkertainen, mutta sitä voisi vielä yksinkertaistaakin. Sovellus aiheutti hämmennystä, mikäli käyttäjälle ei valmiiksi ollut yhtään aktiivista keskustelua palvelussa, vaan päänäkymä oli vain valkoinen. Käyttäjälle lisättiin kehotus siirtyä sovelluksen täysversioon, mikäli hänellä ei valmiiksi ole keskusteluketjuja. Päädyttiin myös poistamaan sovelluksen yläreunasta erillinen ”aktiiviset keskustelut” -otsikko, koska se oli turha. Lisäksi sovelluksen täysiversioon siirtyminen ja ulos kirjautuminen siirrettiin erilliseen näkymään, jotta päänäkymästä saatiin helpommin luettava.

Muutosten jälkeen sovelluksen käyttäminen oli kaikkien testaajien mielestä miellyttävämpää ja selkeämpää.

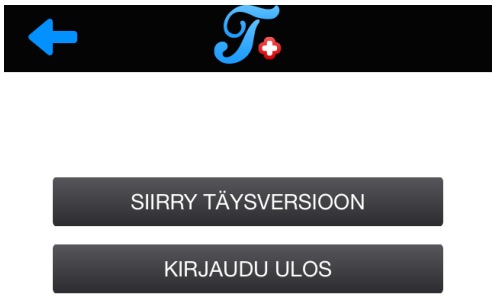
4.7 Lopputuote

Kehitystyön tuloksena sovelluksen käyttöliittymä muuttui paljon yksinkertaisemmaksi siitä, millaiseksi se oli alun perin suunniteltu. Päänäkymässä oleva keskustelulistaus yksinkertaistettiin näyttämään pelkästään keskustelut, ilman ylimääräisiä painikkeita tai välilehtiä (kuva 8).



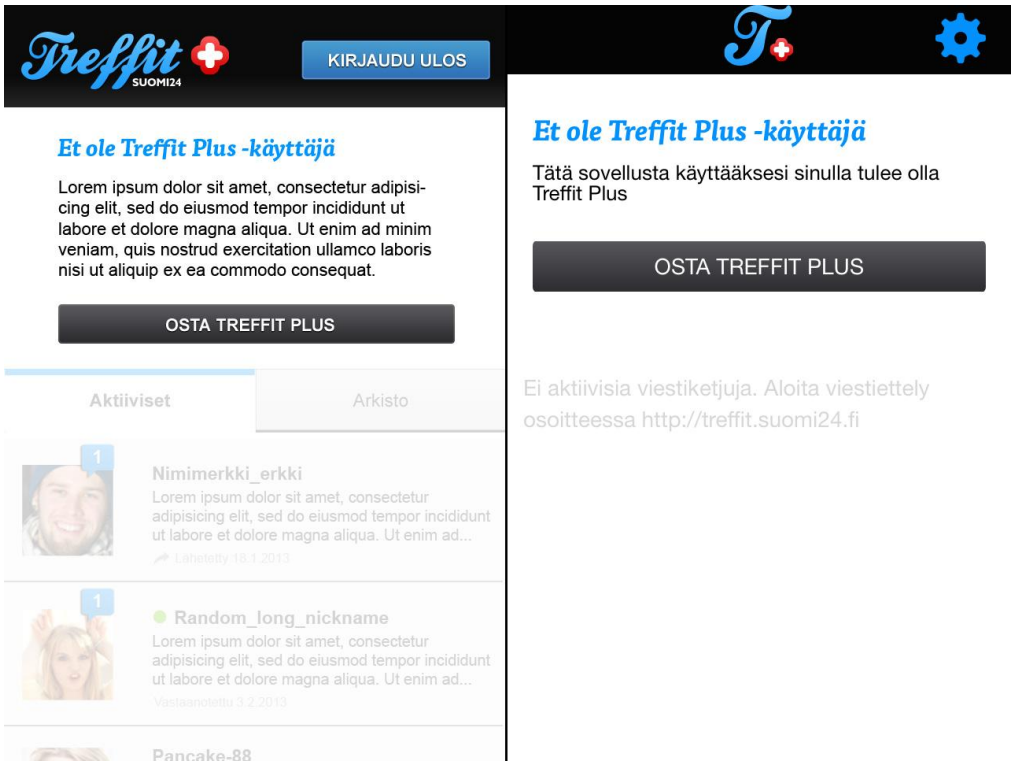
Kuva 8. Sovelluksen alkuperäinen keskustelulistaus ja oikealla toteutunut listausnäkymä.

Navigaation uloskirjautuminen ja päänäkymän ”Siirry täysversioon” -painike siirrettiin erilliseen näkymään (kuva 9).



Kuva 9. Erillinen näkymä sovelluksesta ulos kirjautumiseen ja palvelun täysversioon siirtymiseen.

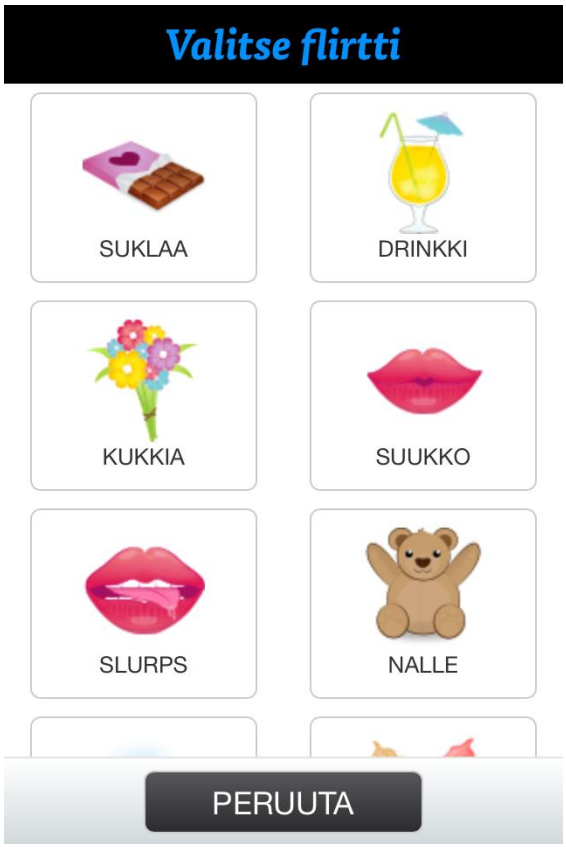
Sovellusta ei voi käyttää, mikäli ei ole Treffit Plus -jäsen, joten käyttäjille tulee näyttää tieto Plus-jäsenyyden ostamisesta (kuva 10). Alkuperäisen ulkoasun ja toteutetun välille ei syntynyt paljon eroa, lähinnä taustalla näkyvä muu käyttöliittymä.



Kuva 10. Ei-Plus-käyttäjälle näkyvä käyttöliittymä sovelluksen käynnistyessä.

Varsinaiseen keskustelunäkymään ei tehty suuria muutoksia, koska se oli alun perinkin tarpeeksi yksinkertainen käyttää (kuva 11). Keskustelunäkymään tehtiin mahdollisuus liittää viestiin flirttejä (kuva 12), jotka ovat myös palvelun täysversiossa.

Kuva 11. Alkuperäinen suunnitelma viestittelynäkymästä ja toteutunut versio.



Kuva 12. Sovelluksen kautta on mahdollista liittää viestiin flirtti.

4.8 Projektin yhteenveto

Tarkoituksena oli julkaista kahdelle eri alustalle mobiilisovellus ja oppia hybridisovelluksen kehittämisestä. Lopputuotteena saatiin kuitenkin vain iOS-versio sovelluksesta, ja aikataulusta jäätiin jälkeen muutama kuukausi. Sovellus kuitenkin saatiin valmiiksi vastoinkäymisistä huolimatta.

Taustatyöstä huolimatta en ottanut huomioon kaikkia mahdollisia skenaarioita. Alustat päivittyvät jatkuvasti, ja varsinkin hybridisovelluksen kehittämisessä alustojen uudet käyttöjärjestelmäpäivitykset saattavat olla kriittisiä.

Noin puolessavälissä projektia Android julkaisi uuden Android 4.4 KitKat -käyttöjärjestelmänsä, joka muutti webview'n käyttäytymistä. Käyttöjärjestelmän päivityt-

työ piti PhoneGapin koodin päivittyä, ja koska Steroids käyttää PhoneGapin pohjakoodia, ei Steroidsin koodia voinut päivittää, ennen kuin PhoneGap oli päivittynyt. Steroidsin päivityksen hitauden vuoksi jouduttiin luopumaan sovelluksen Android-versiosta kokonaan. Android-version kehittäminen olisi vaatinut paljon pelkästään Androidille räätälöityä koodia, jolloin hybridikehittämisen idea ja helppous olisi jäänyt.

Lisäksi en ottanut huomioon Steroidsin tuoreutta ja vielä epävakaata koodia. Ideana Steroids on mielenkiintoinen ja toimiva. Se poistaisi kaiken, mikä PhoneGap-sovelluksissa paljastaa, ettei sovellus oikeasti ole natiivia koodia. Todellisuudessa virheitä on vielä paljon. Oma kehitys takkusi ja hidastui muun muassa sen takia, että navigaatiopalkki ei toiminut, niin kuin dokumentaatio antoi ymmärtää. Steroidsin päivitettyä navigaation tekotapaa eivät ongelmat silti poistuneet, ja sovelluksessa vilahtaa ”back”-näppäin, vaikka sen ei kuuluisi siellä olla.

Projektissa olisi myös pitänyt olla yksi henkilö, joka olisi joskus julkaissut Applen AppStoressa sovelluksen. Ensikertalaiselle Applen vaatimat sertifikaatit olivat todella sekavia, eikä niiden luominen ollut yksinkertaista edes hyvän dokumentaation avulla.

Sovellus oli myös aivan liian suuritöinen kahdelle henkilölle: alusta asti tiimiin olisi tarvittu yksi pelkästään taustajärjestelmän kehittäjä, nyt taustajärjestelmän kehittäminen tehtiin kunnolla vasta projektin loppuvaiheessa.

Projektin alussa luotu aikataulu ei todennäköisesti ole pitävä. Kehitettäessä uutta sovellusta kahden hengen tiimillä täysin alusta asti ja tuntemattomalla tekniikalla voidaan olla varmoja, että aikataulu pettää. Uusia asioita ja kehittäjistä riippumattomia ongelmia tulee eteen aivan varmasti. Mikäli alottaisin projektin nyt, koettaisin löytää tiimiin jonkun, joka on jo tehnyt samankaltaisen sovelluksen. Myös haastattelisin eri tekniikoilla sovelluksia tehneitä henkilöitä sen sijaan, että tekisin taustatyön vain internetin ja lehtiartikkeleiden perusteella.

Lisäksi valitsisin projektiin mahdollisimman vähän eri sovelluskehyksiä, ja nekin päättäisin sen perusteella, minkä elinkaari on luotettavin. En lähtisi enää mukaan uuden startup-yrityksen koodiin, koska projektien kehittämisessä menee aina aikaa eikä stabiilia palvelua voi odottaa alle vuoden vanhalta projektilta.

Harkitsisin jopa vaihtoehtoa, että sovellus kehitettäisiin natiivina, mikäli olisi mahdollista saada yritykseen konsultti, jolla olisi tarpeeksi tietotaitoa.

5 Yhteenveto

Nykyaikana mobiililaitteiden käyttö on lisääntynyt ja osittain syrjäyttänyt tietokoneiden käytön, jolloin mobiilisovelluksen kehittäminen on yritykselle hyödyllistä. Mobiilisovelluksia voi kehittää yleensä kolmella eri tavalla: natiivisovelluksena, selainsovelluksena tai hybridisovelluksena.

Mobiilisovelluksen kehittäminen on haastavaa ja aikaavievää, vaikka nykypäivänä sitä onkin helpotettu mahdollistimalla hybridisovellusten luominen. Hybridisovellusten käyttämistä web-teknologioista tulee olla vankka kokemus, jotta sovelluksen kehittäminen on ketterää.

Projektin tarkoituksena oli tuottaa mobiilisovellus Android- ja iOS-alustoille. Sovellusta kehitettiin aluksi kummallekin alustalle, ja kehittäminen oli nopeaa, koska se luonnistui samalla molemmille. Android-sovellus jouduttiin kuitenkin hylkäämään noin puolessa välissä projektia, koska Androidin käyttöjärjestelmä päivittyi uuteen versioon. Projektissa käytetty sovelluskehysympäristö Steroids ei pystynyt kehittymään yhtä nopeasti.

Lopputuotteena saatiin aikaan karsittu versio sovelluksesta iOS-alustalle. Projekti venyi suunnitellusta aikataulusta, kun yrityksen sisäisissä testauksissa sovelluksessa ilmeni pieniä vikoja. Lisäksi päädyttiin muokkaamaan sovelluksen käyttöliittymää vielä alkuperäisestä, koska testauksessa huomattiin sovelluksen olevan käytettävämpi pelkistetympänä.

Eri kehitystyökalujen päättäminen on tärkeä osa projektia, ja niitä päätettäessä tulee ottaa huomioon kehitystiimin osaaminen. Mikäli osaaminen on vähäistä, on projektiin varattava enemmän aikaa kuin verkkosivuston kehittämiseen, vaikkakin tekniikat periaatteessa ovat samat.

Nuorten kehitysympäristöjen kannattaminen ja kokeileminen on yritysmaailmassa hyvä asia, mutta mikäli projektilla on kiireinen aikataulu, ei ole välttämättä paras vaihtoehto

ottaa sovelluskehykseksi kovin uutta ja vielä epävakaata ympäristöä. Steroids on pääajatukseltaan hyvä, mutta koska se on vielä alkutekijöissään, se aiheutti kehityksessä takapakkia ja hidasti projektin etenemistä.

Lopputuotteena julkaistiin huhtikuussa 2014 S24 Treffit Notifikaatiot -sovellus Applen AppStoressa. Apple hyväksyi sovelluksen AppStoreen yhden korjauskierroksen jälkeen, jossa lähinnä puututtiin sovelluksen liiketoimintamalliin.

Lähteet

1 Mikä on Suomi24? Verkkodokumentti. Suomi24 Oy. <http://www.suomi24.fi/yritys>. Luettu 16.12.2013.

2 Tietoa Allerista. Verkkodokumentti. Aller Media Oy. <http://www.aller.fi/fi/aller/tietoa-allerista>. Luettu 16.12.2013.

3 Mikä on Suomi24 Treffit? Verkkodokumentti. Suomi24 Oy. <http://treffit.suomi24.fi/site/help>. Luettu 16.12.2013.

4 Newark-French, Charles. 2012. Mobile App Usage Further Dominates Web, Spurred by Facebook. Verkkodokumentti. Flurry. <http://blog.flurry.com/bid/80241/Mobile-App-Usage-Further-Dominates-Web-Spurred-by-Facebook>. Luettu 22.3.2014.

5 Khalaf, Simon. 2013. Flurry Five-Year Report: It's an App World. The Web Just Lives in It. Verkkodokumentti. Flurry. <http://blog.flurry.com/bid/95723/Flurry-Five-Year-Report-It-s-an-App-World-The-Web-Just-Lives-in-It>. Luettu 22.3.2014.

6 Häkkinen, Ville. 2012. Kannattaako tehdä mobiilisivusto vai mobiiliapplikaatio? Verkkodokumentti. Fonecta Enterprise Solutions. <http://www.fonectaenterprise.fi/akatemia/blogi/kannattaako-tehda-mobiilisivusto-vai-mobiiliapplikaatio/>. Luettu 14.12.2013.

7 Hybridi-applikaatioiden tekeminen - webkehittäjän kokemus. 2012. Verkkodokumentti. Iwa Labs Oy. <http://www.iwa.fi/blog/hybridi_applikaatioiden_tekeminen_-_webkehittajan_kokemus>. Luettu 14.12.2013.

8 Traeg, Peter. 2013. Best Of Both Worlds: Mixing HTML5 And Native Code. Verkkodokumentti. Smashing Magazine. < http://mobile.smashingmagazine.com/2013/10/17/best-of-both-worlds-mixing-html5-native-code/>. Luettu 15.12.2013.

9 Ghatol, Rohit & Patel, Yogesh. 2012. Beginning PhoneGap: Mobile Web Framework for JavaScript and HTML5. New York: Apress.

10 Rouse, Margaret. 2013. Native app. Verkkodokumentti. Search Software Quality. <http://searchsoftwarequality.techtarget.com/definition/native-application-native-app>. Luettu 15.12.2013

11 Traeg, Peter. 2013. Four Ways To Build A Mobile Application, Part 1: Native iOS. Verkkodokumentti. Smashing Magazine. < http://mobile.smashingmagazine.com/2013/11/22/four-ways-to-build-a-mobile-app-part1-native-ios/>. Luettu 15.12.2013

12 Fransis, Nick. 2010. Web Development For The iPhone And iPad: Getting Started. Verkkodokumentti. Mashing Magazine. < http://mobile.smashingmagazine.com/2010/05/28/web-development-for-the-iphone-and-ipad-getting-started/>. Luettu 20.12.2013.

13 Google Play Policies and Guidelines. Verkkodokumentti. Google. <http://developer.android.com/distribute/googleplay/policies/index.html>. Luettu 3.1.2014.

14 Mattsson, Sofia. 2012. HTML5:n hyödyntäminen mobiililaitteissa. Insinöörityö. Vaasan ammattikorkeakoulu.

15 Toivonen, Jesse. 2013. Phonegap-työvälineohjelmistojen kehitys älypuhelimiin. Insinöörityö. Turun ammattikorkeakoulu.

16 Karenius, Mika. 2012. PhoneGap-ohjelmistokehyksen hyödyntäminen alustariippumattomassa mobiilisovelluskehityksessa. Insinöörityö. Metropolia Ammattikorkeakoulu

17 FAQs. PhoneGap. 2014. Verkkodokumentti. Adobe Systems Inc. <http://phonegap.com/about/faq/>. Luettu 20.1.2014.

18 Heikka, Kai. 2012. Mobiilisovelluksen toteutus web-tekniikoilla PhoneGap-kehykselle. Insinöörityö. Oulun seudun ammattikorkeakoulu.

19 Adobe PhoneGap Build. 2014. Verkkodokumentti. Adobe Systems Inc. <http://html.adobe.com/edge/phonegap-build/faq.html>. Luettu 20.1.2014.

20 AppGyver Steroids. 2013. Verkkodokumentti. AppGyver Inc. <http://www.appgyver.com/steroids>. Luettu 20.1.2014

21 Steroids vs Vanilla PhoneGap™ Hardware access comparison. 2013. Verkkodokumentti. AppGyver Inc. <http://www.appgyver.com/steroids_game_change#comparison-in-detail>. Luettu 20.1.2014.

22 Sarsa, Harri. 2013. Steroids koulutustilaisuus. AppGyver Inc.

23 Porto, Sebastian. 2013. A Comparison of Angular, Backbone, CanJS and Ember. Verkkodokumentti. <http://sporto.github.io/blog/2013/04/12/comparison-angular-backbone-can-ember/>. Luettu 10.2.2014.

24 Genev, Martin. 2013. Backbone or Angular or Ember? Here is my choice and why. Verkkodokumentti. 100PercentJS. <http://www.100percentjs.com/backbone-or-angular-or-ember-here-is-my-choice-and-why/>. Luettu 10.2.2014.

25 Orsini, Lauren. 2014. Angular, Ember, And Backbone: Which JavaScript Framework Is Right For You? Verkkodokumentti. readwrite. <http://readwrite.com/2014/02/06/angular-backbone-ember-best-javascript-framework-for-you#awesm=~oypoP8JQIERNHu>. Luettu 10.2.2014.

26 Katz, Yehuda. 2011. Amber.js (formerly SproutCore 2.0) is now Ember.js. Verkkodokumentti. <http://yehudakatz.com/2011/12/12/amber-js-formerly-sproutcore-2-0-is-now-ember-js/>. Luettu 10.2.2014.

27 Ember.js. 2014. Verkkodokumentti. Tilde Inc. <http://emberjs.com/>. Luettu 10.2.2014.

28 Backbone.js. 2014. Verkkodokumentti. <http://backbonejs.org/>. Luettu 10.2.2014.

29 Angular.JS by Google. 2013. Verkkodokumentti. Google. <http://angularjs.org/>. Luettu 10.2.2014.

30 Angular Changelog. 2014. Verkkodokumentti. Google. <https://github.com/angular/angular.js/blob/master/CHANGELOG.md>. Luettu 14.2.2014.

31 Pointer, Ian. 2013. So, what is Zepto.js? Verkkodokumentti. Pack Publishing. <http://www.packtpub.com/article/so-what-is-zepto-js>. Luettu 16.2.2014.

32 The Essentials of Zepto.js. 2012. Verkkodokumentti. tuts+. <http://code.tutsplus.com/tutorials/the-essentials-of-zeptojs--net-24867>. Luettu 16.2.2014.

33 Fuchs, Thomas. 2014. Zepto.js. Verkkodokumentti. Freckle Online Time Tracking. <http://zeptojs.com/>. Luettu 16.2.2014.

34 Tangelder, Jorik. 2013. Hammer.js. Verkkodokumentti. Eight Media. <http://eightmedia.github.io/hammer.js/>. Luettu 18.2.2014.

35 Chaize, Michaël. 2012. Multitouch with Hammer.js. Verkkodokumentti. Applines. <http://www.appliness.com/multitouch-with-hammer-js/>. Luettu 18.2.2014.

36 Nations, Daniel. 2014. What is Push Notification? Verkkodokumentti. About.com. <http://ipad.about.com/od/iPad-Glossary/g/What-Is-Push-Notification.htm>. Luettu 20.2.2014.

37 iOS: Understanding notifications. 2014. Verkkodokumentti. Apple Inc. <http://support.apple.com/kb/ht3576>. Luettu 20.2.2014.

38 Urban Airship – About us. 2014. Verkkodokumentti. Urban Airship. <http://urbanairship.com/about>. Luettu 20.2.2014.

39 Urban Airship – how to buy. 2014. Verkkodokumentti. Urban airship. <http://urbanairship.com/products/how-to-buy>. Luettu 20.2.2014.

40 iOS Build Configuration. 2015. Verkkodokumentti. Steroids Guides. <http://guides.appgyver.com/steroids/guides/cloud_services/ios-build-config/>. Luettu 20.2.2014.

41 Using Urban Airship Push Notification Plugin. 2015. Verkkodokumentti. Steroids Guides. <http://guides.appgyver.com/steroids/guides/phonegap_on_steroids/ua-plugin//>. Luettu 20.2.2014.