TAMPEREEN
AMMATTIKORKEAKOULU

Open311-rajapinta React Native-mobiilisovelluksessa

Juho Rautio

Opinnäytetyö
Heinäkuu 2017
Tietojenkäsittelyn koulutusohjelma
Ohjelmistotuotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely
Ohjelmistotuotanto

RAUTIO JUHO:
Open311-rajapinta React Native-mobiilisovelluksessa

Opinnäytetyö 32 sivua
Heinäkuu 2017

Tässä opinnäytetyössä oli tavoitteena selvittää, kuinka Open311-palauterajapintaa käytetään React Native-mobiilisovelluksessa ja kuinka rajapintaa voidaan hyödyntää muissa projekteissa. Tarkoitus oli tuottaa Open311-rajapintaa hyödyntävä mobiilisovellus toimeksiantajalle, Haltu Oy:lle. Sovelluksen tilaajana toimi Helsingin kaupunki.

Työssä selvitettiin konstruktiivisilla tutkimusmenetelmillä, miten Open311-rajapintaa käytetään React Native-mobiilisovelluksessa ja millaisia erilaisia käyttömahdollisuuksia rajapinnalla voisi olla. Rajapinnan käyttöä havainnollistettiin, käyttötapausten, koodiesimerkkien ja ruutukaappausten avulla. Sovelluksen toteutuksen aikana kerättiin ylös myös hyviä ja huonoja puolia rajapinnasta ja sen käytöstä React Nativella, joita tässä työssä käsiteltiin.

Projektin toteutuksen ja tulosten tarkastelun myötä selvisi, että Open311-rajapinnassa on potentiaalia moniin erilaisiin palveluihin ja sovelluksiin. Rajapinnan käyttö React Native-sovelluksessa osoittautui suoraviivaiseksi ja suhteellisen ongelmattomaksi, vaikkakaan ohjelmistokehys itsessään ei tuonut erityistä hyötyä rajapinnan käyttöön, muiden järjestelmäriippumattoman sovelluskehityksen hyötyjen lisäksi. Helsingin kaupungin tilaama mobiilisovellus saatiin onnistuneesti valmiiksi ja sovellus on ladattavissa Google Play ja App Store-mobiilisovelluskaupoista sekä Andorid- että iOS-käyttöjärjestelmille.

Asiasanat: open311-rajapinta, react native, mobiilisovelluskehitys, rajapinta

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Bachelor of Business Administration
Software Development

RAUTIO JUHO:
Open311 API in a React Native mobile application

Bachelor's thesis 32 pages
July 2017

The aim of this thesis was to examine how Open311 API (Application programming interface) could be used in a React Native mobile application and how the API could be utilized to different services and applications. The purpose was to develop a React Native mobile application which utilizes the Open311 API, bought by the city of Helsinki, for my employer, Haltu Oy.

Constructive and innovative research methods were used in order to find out what kind of different purposes there are for the API. The use of the API was demonstrated with code samples, use cases and screenshots. Also, the pros and cons of using the API with React Native were examined.

After working on the project and reviewing the results of this thesis, it became clear that Open311 has huge potential, especially for feedback -related applications. Using the API in a React Native application was very straightforward and relatively effortless, however React Native framework itself didn't bring any specific value to using the interface. React Native is still a great choice for developing a cross-platform mobile application utilizing the Open311 interface, due to its great qualities as a cross-platform framework.

Key words: open311, react native, mobile application development, api

SISÄLLYS

ERITYISSANASTO tai LYHENTEET JA TERMIT (valitse jompikumpi)

CitySDK	Sovelluskehityspaketti joka hyödyntää kolmea eri rajapintaa (Open311 API, Linked Data Api ja Tourism API).
FAB	Floating Action Button, Android Material Designin mukainen käyttöliittymäkomponentti.
GeoReport v2	Open311-rajapinnan tietomallin määrittelevä rajapinta.
Java	Ohjelmointikieli (Esimerkiksi Android-sovellukset).
ListView	React Nativessa käytettävä listakomponentti.
Material Design	Android-sovelluksissa käytettävä tyylimäärittely.
SDK	Software Development Kit, sovelluskehityspaketti.
Objective-C	Ohjelmointikieli iOS-sovelluksia varten.
Swift	Ohjelmointikieli iOS-sovelluksia varten.
Ohjelmistokehys	Ohjelmistotuote, jonka avulla rakennetaan uusia ohjelmistoja.
React	JavaScript-pohjainen ohjelmistokehys web-kehitykseen.
React Native	JavaScript-pohjainen ohjelmistokehys mobiilikehitykseen.

1 JOHDANTO

Suurimmassa osassa maailman kaupunkeja on siirrytty käyttämään entistä enemmän digitaalisia palveluita. Lähes kaikki kunnalliset palvelut on tuotu verkkoon, mahdollistaen palvelujen helpon ja nopean käytön. Useimmat kaupungit kuitenkin suunnittelevat ja toteuttavat verkkoinfrastruktuurin sekä palvelurajapinnan itse, mikä on työlästä ja kallista. Yhdysvalloissa on otettu käyttöön viime vuosina useissa suurissa kaupungeissa avoin palvelurajapinta nimeltä Open311, jonka on tarkoitus toimia standardiprotokollana kaupunkien palveluinfrastruktuurille. Rajapinta on helppo ottaa käyttöön ja sitä on mahdollista räätälöidä kunkin kaupungin tai muun järjestelmän tarpeisiin.

Tämän opinnäytetyön tavoitteena on selvittää, kuinka Open311-rajapintaa käyttävä mobiilisovellus toteutetaan React Native-ohjelmistokehyksen avulla, onko rajapinnan käyttö projektissa kyseisellä tekniikalla kannattavaa ja pohtia millaisia erilaisia potentiaalisia käyttömahdollisuuksia Open311-rajapinnalla on sekä kuinka niitä hyödynnetään muissa projekteissa. Tarkoituksena on kehittää Helsingin kaupungin tilaama, Open311-rajapintaa hyödyntävä, mobiilisovellus iOS- ja Android-käyttöjärjestelmille, käyttäen alustariippumattomaan mobiilisovelluskehitykseen soveltuvaa React Native-ohjelmistokehystä.

React Native valittiin projektiin käytettäväksi tekniikaksi, koska Haltu Oy:llä on paljon kokemusta React Nativen käytöstä mobiilisovellusprojekteissa. Sovellusta on kehittänyt aiemmin myös toinen alihankkija, joka kehitti sovellusta React Nativella, joten myös tämän takia tekniikan valinta oli järkevä, sillä näin ollen aiempaa ohjelmistokoodia oli helpompi käyttää suoraan hyväksi projektin toteutuksessa. Tässä työssä keskitytään pohtimaan tutkielman tavoitteita etenkin sovelluskehittäjän näkökulmasta, mutta kokonaisuutta ajatellen asioita on lähestyttävä myös käyttäjän näkökulmasta.

2 TAUSTA

2.1 Toimeksiantaja

2.1.1 Toimeksiantajan esittely

Opinnäytetyön toimeksiantajana toimii Haltu Oy, joka on Tamperelainen ohjelmistotalo, joka on erikoistunut web-palveluiden ja mobiilisovellusten toteutukseen sekä ylläpitoon. Toimeksiantaja on hiljattain alkanut toteuttaa alustariippumattomia mobiilisovelluksia React Native-ohjelmistokehyksen avulla, jota on käytetty myös tätä opinnäytetyötä käsittelevän sovelluksen toteutukseen. Haltu Oy toteuttaa kaikenlaisia projekteja ja osa projekteista saattaa olla hyvinkin aikataulu- tai budjettikriittisiä jolloin alustariippumattomien ohjelmistokehysten käyttäminen on hyvin kustannustehokasta, sillä se vähentää tarvetta kirjoittaa samaa ohjelmakoodia useaan kertaan. Toimeksiantaja on toivonut myös saavansa lisää tietoa Open311-rajapinnan käytöstä mobiilisovellusprojekteissa, jotta voisi tarjota vastaavanlaisia projekteja myös muille asiakkaille.

2.1.2 Projektin esittely

Projektin tilaajana toimii Helsingin kaupunki, joka on tilannut mobiilisovelluksen iOS- ja Android-käyttöjärjestelmille, joka hyödyntää Helsingin kaupungin käyttämää Open311-palauterajapintaa. Palauterajapinnan kautta voi antaa Helsingin kaupungille kaupunki-infrastruktuurin korjaamiseen liittyvää palautetta, kuten esimerkiksi ilmoittaa rikkoutuneista liikennemerkeistä sekä antaa palautetta liittyen muihin kaupungin osa-alueisiin kuten kulttuuriin ja vapaa-aikaan sekä sosiaali- ja terveyspalveluihin. Rajapinnan kautta on myös mahdollista hakea muiden kaupunkilaisten antamaa palautetta. Helsingin kaupungilla on jo ennestään web-pohjainen käyttöliittymä olemassa palautteen antamista varten, mutta haluaa tehdä palautteen antamisesta entistä helpompaa tuomalla palvelun mobiilialustoille.

Projekti päätettiin toteuttaa React Native-ohjelmistokehyksellä, sillä tilaaja oli jo aiemmin toisen alihankkijan kanssa aloittanut vastaavaa projektia React Native-ohjelmistokehyksellä, joka oli kuitenkin jäänyt kesken. Näin ollen projektin toteutuksessa voisi hyödyntää myös toisen alihankkijan kirjoittamaa ohjelmistokoodia. React Native sopi myös

hyvin toimeksiantajalle, sillä Haltu Oy on toteuttanut useita alustariippumattomia mobiilisovellusprojekteja React Native-ohjelmistokehyksellä.

2.2 Tekniikoiden esittely

2.2.1 Web-rajapinnat

Web-rajapinnat ovat verkon yli toimiva rajapintoja, jotka tarjoavat sovelluskehittäjälle pääsyn lukea ja muokata järjestelmässä olevaa dataa. Tietoa käsitellään rajapinnan tarjoamien funktioiden avulla. Näin sovelluskehittäjän ei tarvitse tehdä kutsuja suoraan järjestelmän tietokantaan, vaan palvelin hoitaa itse tietokannan kanssa keskustelun optimoiduilla tietokantakutsuilla ja tarjoaa rajapinnan, jonka avulla kehittäjä pääsee tietokannan dataan käsiksi. Web-rajapinnat ottavat vastaan HTTP-protokollan mukaisia GET- tai POST-pyyntöjä ja palauttavat vastauksen yleensä JSON tai XML muodossa.

2.2.2 Open311-rajapinta

Open311-rajapinta on niin kutsuttujen “311-palveluiden” eli kaupunki-infrastruktuuriin liittyvien palvelupyyntöjen lähettämiseen ja käsittelemiseen tarkoitettu web-rajapinta. 311-palvelut saavat nimensä Pohjois-Amerikassa käytetystä 3-1-1 puhelinnumerosta, joka on hätänumeron (9-1-1) tapainen palvelunumero joka toimii väylänä kunnallispalveluiden tarjoamiseen kansalaisille.

Open311-rajapinta tuo palvelun verkkoon ja on aiemman yksisuuntaisen puhelinpalvelun sijaan kahdensuuntainen palvelu kaupunki-infrastruktuurin ongelmiin liittyvien palvelupyyntöjen lähettämiseen ja seurantaan, joka tarjoaa avoimen näkyvyyden ja paremman kanssakäymisen kansalaisten ja kaupungin välille. Open311-rajapintaa käytetään useissa suurissa kaupungeissa Euroopassa ja Pohjois-Amerikassa.

2.2.3 React Native-ohjelmistokehys

React Native on Facebookin kehittämä ohjelmistokehys, jonka avulla voidaan kehittää alustariippumattomia mobiilisovelluksia Android- sekä iOS-käyttöjärjestelmille. React

Nativella tehdyt sovellukset ovat suorituskyvyltään lähes natiivisovellusten veroisia, sillä React Native hyödyntää natiiveja käyttöliittymäkomponentteja.

React Native toimii abstraktiokerroksena joka mahdollistaa JavaScript-ohjelmistokoodin kääntämisen kummallekin mobiilialustalle, ilman että kehittäjien tarvitsee erikseen kirjoittaa Java ja Objective-C tai Swift koodia molempia käyttöjärjestelmiä varten. Monista muista alustariippumattomista ohjelmistokehyksistä poiketen React Native ei “renderöi” eli hahmonna käyttöliittymää HTML- ja JavaScript-elementteinä, vaan käyttää natiiveja Android- ja iOS-käyttöliittymäkomponentteja mahdollistaen React Native-sovelluksille lähes natiivisovelluksia vastaavan suorituskyvyn.

3 OPEN311-RAJAPINTA

3.1 Open311-rajapinnan taustoja

3.1.1 3-1-1-puhelinpalvelu

3-1-1-puhelinpalvelu otettiin kokeilukäyttöön ensimmäisen kerran vuonna 1996 Baltimoressa, New Yorkissa. Sen tarkoituksena oli tuoda hätänumeron kaltainen, kolminumeroinen puhelinpalvelu kiireettömiä palvelupyyntöjä varten sellaisiin tapauksiin, joihin tarvittiin poliisin osallistumista, mutta eivät olleet kiireellisiä tai vaatineet välitöntä toimintaa. Käyttäjiä rohkaistiin soittamaan puhelinpalveluun tehdäkseen valituksia esimerkiksi laittomasta roskaamisesta ja väärinpysäköinnistä. Ajatuksena oli että 3-1-1 numeroon soitetut puhelut ohjataan samaan puhelinpalvelukeskukseen kuin hätänumeroon soitetut puhelut, mutta ne otettiin käsittelyyn, vain jos hätäkeskuspäivystäjiä oli vapaina oikeilta hätätapauksilta.

Vuonna 2001 Baltimoressa laajennettiin 3-1-1-puhelinpalvelu sisältämään kaikki kaupungin palvelut graffitien poistosta puistojen hoitoon, jonka jälkeen 3-1-1-puhelinpalvelusta tuli pian uusi standardi koko Yhdysvaltoihin. 3-1-1-soittokeskuksia on mainostettu sloganilla “Palava talo? Soita 9-1-1. Polttava kysymys? Soita 3-1-1.” Nykypäivänä 3-1-1-soittokeskukset vastaanottavat jopa 50 000 soittoa päivässä 180:llä eri kielellä ympäri vuorokauden. (Goodyear, 2015)

Suomessakin on käytössä vastaavanlainen palvelunumero 116 115, johon soittaminen on maksutonta.

3.1.2 Open311-rajapinnan synty

Open311-rajapinta otettiin ensimmäisenä käyttöön Washington D.C.:ssä vuonna 2010. Sen oli tarkoitus toimia verkkorajapintana 3-1-1-palvelupyynnöille Washington D.C.:ssä, mutta San Franciscon kaupungin sekä monien yritysten yhteistyö rajapinnan kehittämiseksi tekivät siitä työkalun jolla hallitukset ja yhteisöt pystyvät paremmin kommunikoimaan kansalaisten kanssa ja tekemään yhteistyötä, kanssakäymisen ollessa avointa ja läpinäkyvää kaikille osapuolille. Open311-rajapinnan ja tätä hyödyntävien palveluiden

käyttö tuo myös sosiaalisia hyötyjä, sillä se aktivoi ihmisiä kansalaistoimintaan ja mahdollistaa kansalaisille väylän vaikuttaa suoraan oman yhteisönsä asioihin.

3.1.3 Rajapinnan käyttö Helsingissä

Helsingissä rajapinta julkaistiin avoimeksi kehittäjien käyttöön toukokuussa 2013 osana Euroopan komission rahoittamaa CitySDK-hanketta. Ensimmäinen julkinen rajapintaa hyödyntävä palvelu oli Metro Fiksaa -niminen verkkopalvelu, joka toimii samankaltaisena väylänä lähettää ja seurata palautetta kaupungin rakennusvirastolle, kuin tämän opinnäytetyön aikana toteutettava mobiilisovellus. Metro Fiksaa palvelun kautta oli lyhyessä ajassa lähetetty jo satoja palvelupyyntöjä, mikä toimi osoituksena siitä, että Open311-rajapinnan käyttö todellakin tuo palvelut lähemmäksi kansalaisia. (City of Helsinki's issue reporting API open for developers, 2013)

3.2 Rajapinnan tarkoitus

Open311-rajapinnan on tarkoitus toimia avoimena standardiprotokollana jonka avulla käyttäjät voivat lähettää eri kategorioihin, kuten kaupunki-infrastruktuuriin, liittyviä palvelupyyntöjä, jotka rajapinta automaattisesti ohjaa oikeille tahoille sekä tarjoaa käyttäjille mahdollisuuden seurata omia sekä muiden lähettämiä palvelupyyntöjä, niiden käsittelyn etenemistä ja niihin liittyviä kommentteja. Rajapinnasta on haluttu tehdä avoin standardi, jotta kuka tahansa voi osallistua tekniikan kehitykseen ja kehitys olisi läpinäkyvää, mutta myös siksi, että sen käyttöönotto eri kaupunkeihin ja palveluihin olisi mahdollisimman helppoa. Samaa standardia toteuttavien palvelujen ja sovellusten etu on myös se, että tällaiset sovellukset voivat myös keskustella ja vaihtaa tietoa keskenään.

Vaikka 3-1-1-palvelupyyntöjen ja Open311-rajapinnasta puhuttaessa keskitytäänkin paljon kaupunki-infrastruktuuriin liittyviin asioihin, kuten teiden kunnossapitoon ja töhryjen poistamiseen, rajapinta tuo pohjan kaikenlaiselle viestinnälle hallitusten ja yhteisöjen välille. Tulevaisuudessa rajapinnan avulla voidaan mahdollisesti hoitaa esimerkiksi verojen maksua ja monia muita kunnallisiin palveluihin liittyviä asioita. (Steinberg, 2013)

3.3 Rajapintamäärittely

Rajapinta vastaanottaa HTTP pyyntöjä jotka sisältävät XML- tai JSON-muodossa olevaa dataa. Rajapinta palauttaa HTTP vastauksen joka sisältää tilakoodin sekä pyydetyn datan GeoReport v2-rajapinnan määrittelemässä muodossa.

3.3.1 Palveluiden hakeminen

Lista palveluista ja palvelutunnuksista voidaan hakea rajapinnasta HTTP GET-pyynnöllä. Palvelutunnus määrittää mihin palveluun lähetettävä palvelupyyntö liittyy.

```
[
  {
    "service_code":001,
    "service_name":"KAUPUNKI",
    "description":"Kaupunki-infrastruktuuriin liittyvät palvelut",
    "metadata":true,
    "type":"realtime",
    "keywords":"kadut, kaupunki, siivous",
  },
  {
    "service_code":002,
    "metadata":true,
    "type":"realtime",
    "keywords":"lorem, ipsum, dolor",
    "group":"street",
    "service_name":"PUISTOT",
    "description":"Puistojen ja viheralueiden hoito.",
    "keywords":"puistot, siivous, viheralueet, metsä"

  },
  {
    "service_code":003,
    "metadata":true,
    "type":"realtime",
    "keywords":"lorem, ipsum, dolor",
    "group":"street",
    "service_name":"VANDALISMI",
    "description":"Graffitien poisto, aitojen korjaus, siivous"
    "keywords":"melu, häiriö, graffiti, siivous, korjaus, kaupunki",

  }
]
```


KUVA 1. Esimerkki palveluista ja palveluihin liittyvistä tiedoista GeoReport v2-rajapinnan määrittelemässä muodossa

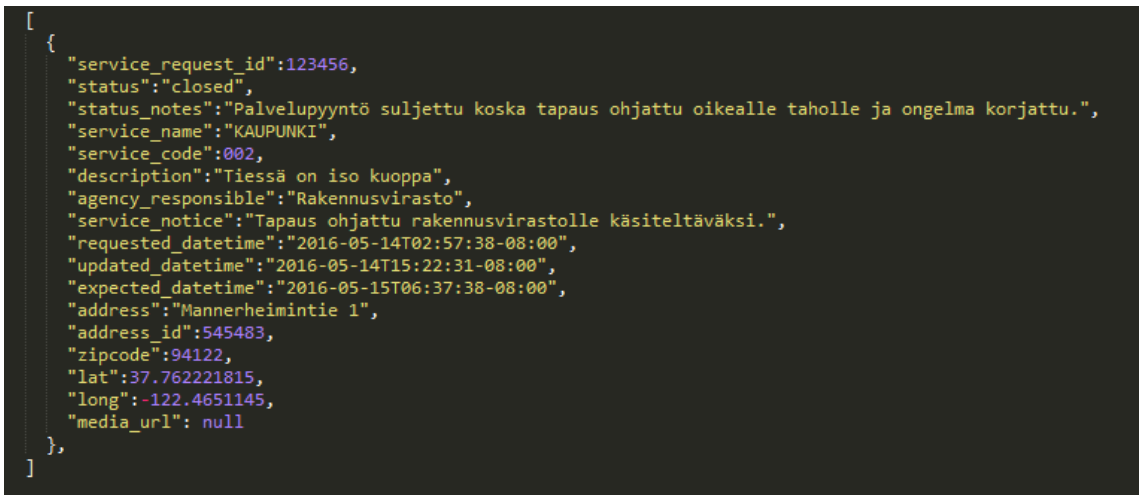
3.3.2 Palvelupyynnön lähetys

Palvelupyyntö lähetys vaatii pakollisena parametrina rajapinnasta löytyvää palvelua vastaavan palvelutunnuksen. Palvelupyyntöön on mahdollista liittää myös erilaista dataa, kuten sijainti- ja osoitetiedot, palvelupyynnön lähettäjän yhteystiedot ja mediatiedostoja, kuten ääntä, kuvia ja videoita. Pyyntöön voidaan myös liittää sen laitteen tunniste, josta pyyntö on lähetetty. Laitteen tunniste liitetään yleensä vain käytettäessä rajapintaa mobiililaitteesta. Palvelupyyntö lähetetään rajapintaan HTTP POST-pyynnöllä.

3.3.3 Palvelupyynnön seuraaminen

Rajapinnasta on mahdollista hakea useita palvelupyyntöjä kerralla sekä yksittäisiä palvelupyyntöjä. Useita palvelupyyntöjä haettaessa käytetään GET-pyyntöä, jolle annetaan erilaisia suodattimia parametreiksi. Palvelupyyntöjä voi suodattaa tunnisteiden, palvelutunnuksen, aikavälin ja palvelupyynnön tilan mukaan. Rajapinta palauttaa halutut palvelupyynnöt XML- tai JSON-muodossa. Niiden tietojen lisäksi, jotka palvelupyynnön lähettäjä on pyyntöön alun perin liittänyt, rajapinnan vastauksesta löytyy myös tietoa palvelupyynnön tilasta, kuten onko pyyntö jo käsitelty, mille tahoille pyyntö on ohjattu käsiteltäväksi ja palvelupyyntöön liittyviä kommentteja.

Rajapinnan käyttöön ottanut taho voi lisätä myös ylimääräisiä attribuutteja palvelupyyntöihin, vaikka GeoReport v2-rajapintamäärittely ohjeistaa lisäämään mahdollisimman vähän uusia attribuutteja, jotta tietomalli pysyisi standardin mukaisena ja eri kaupunkien rajapintojen hyödyntäminen olisi mahdollista samalla sovelluksella. (Open311 Wiki, 2010) Esimerkiksi Helsingin käytössä olevaa Open311-rajapinnassa palvelupyynnön tietomallia on jatkettu CitySDK-lisäosalla, jolla palvelupyyntöön ja käsittelyn etenemisprosessiin saadaan enemmän informaatiota mukaan, mutta tietomalli ei pysy GeoReport v2-rajapinnan standardin mukaisena, eikä rajapintaa voida geneerisesti hyödyntää muissa Open311-rajapintaa käyttävissä järjestelmissä, elleivät nekin toteuta CitySDK-rajapintaa.



KUVA 2. Esimerkkivastaus Open311-rajapinnalta palvelupyynnön kyselyyn.

KUVA 3. CitySDK-rajapinnan lisäattribuutteja.

3.3.4 Esimerkkikäyttötapaus

Esimerkkikäyttötapauksessa Hilkalla on käytössään älypuhelin, johon on asennettu Open311-rajapintaa hyödyntävä asiakassovellus. Palvelimena toimii kaupungilla käytössä oleva Open311-rajapinta johon sovellus ottaa yhteyden.

Hilkka käy hakemassa lähikaupasta maitoa ja huomaa, että kaupan seinä on töhritty graffiteilla. Hilkka ottaa esiin älypuhelimensa ja käyttää Open311-rajapintaa hyödyntävää mobiilisovellusta.

Ensimmäisenä Hilkka ottaa palvelimelle yhteyden ja kysyy palvelimelta mitä palveluita rajapinta tarjoaa.

HILKKA

```
Mitä palveluita tarjoatte?
```

GET Service List

PALVELIN

```
Tarjoamme seuraavia palveluita:

001 KAUPUNKI: Kaupunki-infrastruktuuriin liittyvät palvelut
002 PUISTOT: Puistojen ja viheralueiden hoito
003 VANDALISMI: Graffitien poisto, aitojen korjaus, siivous
```

200 OK

KUVA 4. Esimerkkikäyttötapaus palveluiden listaamisesta.

Rajapinta palauttaa listan palvelutunnuksista. Jokaiseen palvelupyyntöön on liitettävä jokin palvelutunnus.

HILKKA

```
Näin kaupan seinässä töhryjä. Tarvitsisin VANDALISMI -palvelua
```

POST Service Request

```
{
        service_code: "003",
        adderss_string: null,
}
```

PALVELIN

```
Virhe, unohdit kertoa missä kauppa sijaitsee.
```

400 Error

KUVA 5. Esimerkkikäyttötapaus virheellisestä palvelupyynnön lähetyksestä.

Jotkin pyynnöt vaativat tiettyjä parametreja palvelua pyytäessä, kuten esimerkiksi sijaintitietoa tai valokuvaa. Palvelimelta on mahdollista myös kysyä lisätietoa tietystä palvelutunnuksesta.

HILKKA

```
Kerrotko lisää VANDALISMI -palvelusta?
```

```
GET Service Definition
{
        service_code: "003",
}
```

PALVELIN

```
Autamme siivoamaan graffiteja ja muita töhryjä, korjaamaan rikottuja paikkoja ja
siivoamaan lasinsiruja, ynnä muuta sellaista. VANDALISMI -palvelu vaatii
toimiakseen sijaintitiedon.
```

```
200 OK
```

KUVA 6. Esimerkkikäyttötapaus palvelun lisätietojen kysymisestä

Varmistettuaan että Hilkalla ei puutu vaadittuja tietoja palvelupyynnöstä, hän lähettää uuden palvelupyynnön.

HILKKA

```
Tarvitsen VANDALISMI -palvelua. Näin kaupan seinässä töhryjä, osoitteessa
Esimerkkikuja 123, 12345 Helsinki.
```

```
POST Service Request
{
        service_code: "003",
        adderss_string: "Esimerkkikuja 123, 12345 Helsinki",
}
```

PALVELIN

```
OK, palvelupyyntö vastaanotettu. Palvelupyyntösi uniikki tunniste on 642056.
```

```
200 OK
```

KUVA 7. Esimerkkikäyttötapaus palvelupyynnön lähetyksestä.

Palvelin palauttaa Hilkalle tunnisteen jonka avulla palvelupyynnön tilaa voi myöhemmin

4 KÄYTTÖJÄRJESTELMÄRIIPPUMATON SOVELLUSKEHITYS

4.1 Natiivisovellukset

Kolme yleisintä mobiilikäyttöjärjestelmää ovat Android, iOS ja Windows Phone. Maailmalla eniten käytetty edellä mainituista on Android, kun taas Windows Phonea käytetään vähiten, tosin Suomessa Windows Phonen käyttö on vielä suhteellisen yleistä.

Ohjelmoitaessa natiiveja mobiilisovelluksia, on jokaiselle alustalle ohjelmoitava eri ohjelmointikielellä, eivätkä ohjelmakoodit siis ole keskenään yhteensopivia. Esimerkiksi iOS-sovelluksia ohjelmoidaan Objective-C tai Swift-ohjelmointikielillä, kun taas Android-sovelluksia ohjelmoidaan Javalla. Natiivin mobiilikehityksen etu on se, että ohjelmoijan on helppo hyödyntää suoraan käyttöjärjestelmän rajapintaa ohjelmoidessaan. Natiivit sovellukset ovat myös suorituskyvyltään järjestelmäriippumattomia mobiilisovelluksia parempia. (Abrosimova, 2014)

4.2 Web-sovellukset

Järjestelmäriippumattomia mobiilisovelluksia voi kehittää myös normaaleilla web-kehittäjän työkaluilla ja taidoilla. Ohjelmoija luo responsiivisen, mobiililaitteille sopivan, verkkosivun jota näytetään mobiililaitteessa käyttäen käyttöjärjestelmän natiivia WebView-komponenttia. Tällöin natiivia ohjelmakoodia ei tarvita kovin paljoa ja kehitys on edullista. Web-sovellukset ovat kuitenkin käytännössä vain tavallisia verkkosivuja, joten ne eivät yllä lähimainkaan natiivisovellusten suorituskykyyn ja käytettävyyteen.

4.3 Ohjelmistokehykset

Ohjelmoitaessa järjestelmäriippumatonta mobiilisovellusta, voidaan käyttää jonkin kolmannen osapuolen luomaa ohjelmistokehystä. Mobiilisovelluskehityksessä yleisesti käytettyjä ohjelmistokehyksiä ovat esimerkiksi Ionic, React Native ja MeteorJS.

Ohjelmistokehykset tuovat kehitykseen ylimääräisen abstraktiotason, joka mahdollistaa sen, että kehittäjä voi esimerkiksi pelkästään JavaScript-ohjelmakoodia kirjoittamalla luoda sovelluksen kolmelle eri mobiilikäyttöjärjestelmälle. Ohjelmistokehysten etu web-

sovelluksiin nähden on siinä, että ohjelmistokehys mahdollistaa osittain käyttöjärjestelmän rajapinnan hyödyntämisen, jolloin esimerkiksi käyttöliittymäkomponentteina voi käyttää suoraan natiiveja käyttöliittymäkomponentteja. Monet ohjelmistokehysten avulla kehitetyt mobiilisovellukset yltävät yksinkertaisimpien sovellusten kohdalla niin hyvään suoritustasoon, ettei niitä erota natiivisovelluksesta.

Kehitys on myös huomattavasti nopeampaa ja halvempaa kuin natiivisovelluksia kehittäessä, sillä koko sovellusta ei tarvitse ohjelmoida uudelleen kahteen tai kolmeen kertaan. Yksikkötestien kirjoittaminen on myös helpompaa, kun ne voidaan kirjoittaa yhtä ohjelmistokoodia varten.

5 REACT JA REACT NATIVE

5.1 React

React tai React JS on Facebookin kehittämä JavaScript-kirjasto web-käyttöliittymän rakentamiseen. Reactin ajatuksena on rakentaa itsenäisiä käyttöliittymäkomponentteja, jotka hallitsevat omaa tilaansa, ja rakentaa näistä komponenteista suurempia kokonaisuuksia. React käyttöliittymäkomponentteja kirjoitetaan JSX-syntaksilla, joka on samankaltainen kuin XML-syntaksi. Lopulta JSX-koodi kompiloidaan tavalliseksi HTML-koodiksi, joka näytetään käyttäjälle.

5.2 React Native

5.2.1 React Nativen taustoja

React Native on melko uusi React JS:ään pohjautuva ohjelmistokehys. Sen kehitys alkoi vuonna 2013 Facebookin työntekijöiden sisäisenä Hackathon projektina. Ensimmäinen versio tuli julkiseen jakeluun avoimena lähdekoodina Githubissa maaliskuussa 2015. Sen jälkeen React Nativen käyttö mobiilisovellusten ohjelmointiin on kasvattanut suosiotaan huimasti. React Nativella on takanaan suuria sijoittaja, kuten Microsoft ja Samsung, jotka haluavat tuoda React Native -tuen myös Windows- ja Tizen-alustoille. React Nativea käyttävät mobiilisovelluskehitykseen monet yritykset, kuten Instagram, Facebook ja Discord.

5.2.2 Miksi valita React Native

React Nativella on mahdollista luoda natiivisovelluksia vastaavia sovelluksia käyttäen pelkästään JavaScript-ohjelmointikieltä. React Native on malliltaan samanlainen kuin React JS, joka on niin ikään Facebookin kehittämä suosittu web-käyttöliittymäkomponenttikirjasto. React Nativella tehdyt sovellukset eivät ole web-sovelluksia tai hybridisovelluksia, vaan lähes natiivisovellusta vastaavia. React Native käyttää iOS:n ja Androidin natiiveja käyttöliittymän rakennuspalikoita, joita kuitenkin käytetään JavaScript-ohjelmakoodilla, Objective-C- tai Java-koodin sijaan. React Nativen rinnalla on mahdollista kirjoittaa myös natiivia ohjelmakoodia, joten ohjelmistokehys ei rajoitu pelkästään JavaScriptillä ohjelmointiin. React Nativella on myös aktiivinen kehittäjäyhteisö,

joka julkaisee avoimena lähdekoodina uudelleen hyödynnettäviä komponentteja, joita kehittäjät voivat käyttää projekteissaan. Kolmansien osapuolten kehittämien komponenttien lisääminen projektiin ja käyttäminen on todella suoraviivaista React Native-sovelluksessa.

Vaikka React Native helpottaakin järjestelmäriippumatonta kehittämistä huomattavasti, täytyy kuitenkin muistaa, että React Native on yksi melko suuri abstrahointitaso lisää mobiilisovelluksen kehitykseen. Abstrahoinnilla tässä yhteydessä tarkoitetaan sitä, että kirjoitettavan JavaScript-koodin ja natiivin käyttöjärjestelmän välissä on iso nippu ohjelmakoodia, johon kehittäjä ei voi juuri vaikuttaa. Jos ohjelmistokehyksestä löytyy kriittinen ongelma, kolmannen osapuolen kehittäjälle sen ohittaminen voi olla mahdotonta. (Eberhardt, 2015)

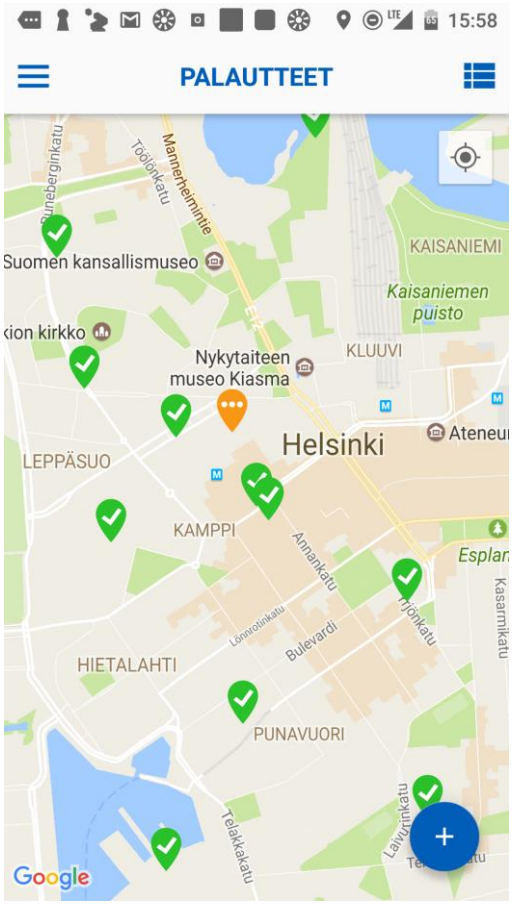
6 PROJEKTIN TOTEUTUS

6.1 Suunnittelu ja kehitys

Projektia lähdettiin suunnittelemaan kahden ohjelmoijan, käytettävyys- ja käyttöliittymäsuunnittelijan, projektin johtajan sekä projektin tilaajan kanssa. Projektin budjetti oli suhteellisen pieni ja tilaajan toiveena olikin, että keskitytään tekemään vain muutama ominaisuus sovellukseen, mutta tehdään ne hyvin. Tilaaja painotti, että sovelluksen tulee toimia sulavasti ja siirtymien eri näkymien välillä tulee olla animoituja. Mahdollisia lisäominaisuuksia toteutettaisiin erillisenä projektina myöhemmin.

Tilaajan vaatimukset sovellukselle olivat karttanäkymä, joka näyttää rajapintaan lähetettyjen palautepyyntöjen sijainnit, toiminnallisuus palautepyynnön lähettämistä varten, palautepyyntöjen näyttäminen listanäkymässä sekä näkymä yksittäisen palautepyynnön detaljeja varten. Tilaaja toimitti myös väriteeman ja käynnistyskuvan sovellusta varten, mutta jätti muuten suhteellisen vapaat kädet sovelluksen ulkonäön suhteen. Projektia varten sovittiin viikoittainen puhelinpalaveri, jossa tarkastellaan projektin etenemistä ja jaetaan ajatuksia.

Sovelluksen käyttöliittymän suunnittelua päätettiin lähestyä Androidin Material Design -suositusten mukaisesti. Karttanäkymään alanurkkaan päätettiin lisätä FAB (Floating Action Button), jota painamalla käyttäjä pääsee palautepyynnön lähetysnäkymään. FAB:n tarkoitus on ohjata käyttäjän huomio tärkeimpään toimintoon, jonka palautepyynnön lähettämisen todettiin olevan.



KUVA 15. FAB (Floating Action Button) karttanäkymän oikeassa alanurkassa.

Palautepyyntöjen listausnäkymästä haluttiin yksinkertainen. Yksittäisen palautepyynnön tiedot päätettiin esittää listassa supistettuina, niin että pikaisella vilkaisulla selviää, koska palautepyyntö on jätetty sekä pieni ote palautepyynnön viestistä ja tämän otsikko. Myöhemmässä vaiheessa projektia todettiin, että listanäkymään olisi hyvä saada vielä tieto siitä, onko palautepyyntöön vastattu ja mikä taho tai virasto vastauksen on antanut.

Yksittäisen palautepyynnön detaljinäkymään haluttiin saada näkyviin palautepyynnön alkuperäiset tiedot, sekä kyseiseen palautepyyntöön liittyvät tapahtumat ja kommentit. Malli palautepyynnön detaljinäkymälle muuttui iteraatioiden myötä moneen kertaan projektin aikana, käyttäjäpalautteen myötä.

Alkuperäisissä suunnitelmissa varsinainen navigointivalikko päätettiin jättää pois, sillä käytännössä sovellukseen oltiin tekemässä ainoastaan kaksi näkymää, karttanäkymä ja listanäkymä. Kummastakin näkymästä on pääsy FAB:n kautta palautteen lähetysnäkymään, sekä yksittäistä palautepyyntöä klikkaamalla palautepyynnön detaljinäkymään. Myöhemmässä vaiheessa projektia päätettiin kuitenkin lisätä niin kutsuttu “hamburger-

menu”, eli ruudun sivusta aukeava vedettävä valikko, mahdollisia tulevia lisäominaisuuksia varten.

Tilaaja toivoi, että käyttäjällä olisi myös jokin keino antaa palautetta. Yhteisen puhelinpalaverin aikana saatiin ilmoille ajatus, että Helsingin kaupungin Open311-rajapintaan lisätään oma palvelutunnus sovelluspalautetta varten, jotta pystyisimme hyödyntämään olemassa olevaa rajapintaa sovelluspalautteen lähettämiseen, mikä toimii hyvänä esimerkkinä rajapinnan monikäyttöisyydestä ja säästi paljon aikaa ja vaivaa sovelluksen kehittämiseltä, kun aikaisempaa palautepyynnön lähetykseen liittyvää ohjelmakoodia pystyi uusiokäyttämään.

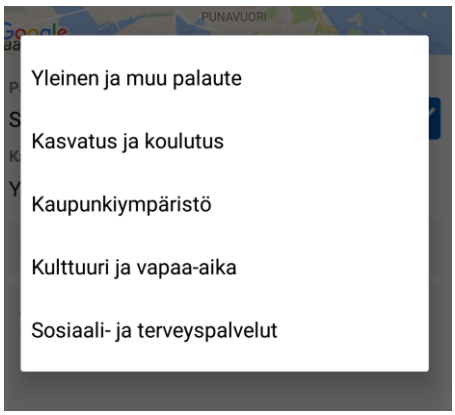
6.2 Rajapinnan käyttö sovelluksessa

6.2.1 Palveluiden hakeminen

Ensimmäisenä käyttöliittymää varten on haettava lista palveluista, jotta käyttäjä voi valita mihin palveluun lähetettävä pyyntö liittyy. Koodiesimerkissä lähetetään Open311-rajapinnalle HTTP GET-pyyntö, joka palauttaa listan palveluista ja palvelutunnuksista.

```
fetchServices() {
  var url     = Config.OPEN311_SERVICE_LIST_URL + Config.OPEN311_SERVICE_LIST_LOCALE + 'fi';
  var headers = {'Accept': 'application/json', 'Content-Type': 'application/json'};

  makeRequest(url, 'GET', headers, null, null)
  .then(result => {
    this.parseServiceList(result);
  }, err => {
    showAlert(transError.networkErrorTitle, transError.networkErrorMessage, transError.networkErrorButton);
  });
}
```

KUVA 8. http-pyyntö React Nativessa, jolla haetaan Open311-rajapinnasta lista palveluista



KUVA 9. Käyttäjälle esitettävä lista palveluista

6.2.2 Palvelupyynnön lähetys

Sovelluksessa käyttäjän on mahdollista liittää palvelupyyntöön tiedoiksi valitun palvelutunnuksen lisäksi otsikko, viesti, sijaintitieto sekä liitetiedostona yksi kuva. Rajapintaan on mahdollista lähettää myös useita kuvia kerralla. Tiedot syötetään listaan avain–arvo-pareina, jotka lähetetään rajapintaan POST-pyynnöllä. Pyyntö palauttaa vastauksena tunnisteen, jolla palautetta on mahdollista seurata.

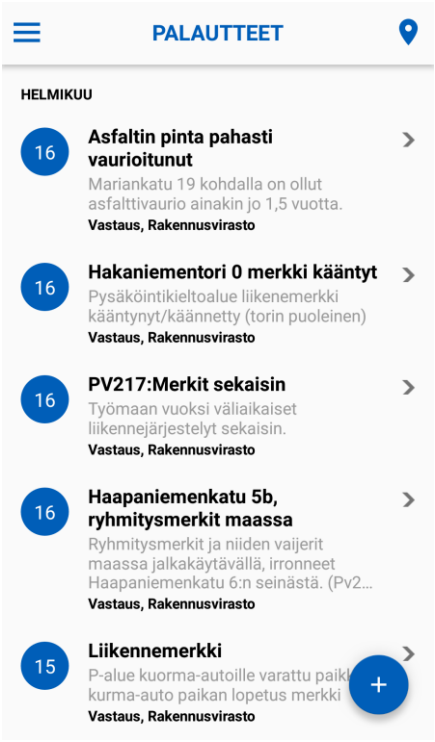
```
sendServiceRequest() {
  var url     = Config.OPEN311_SEND_SERVICE_URL;
  var method  = 'POST';
  var headers = {'Content-Type': 'multipart/form-data', 'Accept': 'application/json'};
  var data    = new FormData();

  data.append('api_key', Config.OPEN311_SEND_SERVICE_API_KEY);
  data.append('service_code', this.state.selectedServiceCode);
  data.append('description', this.state.descriptionText);
  data.append('title', this.state.titleText !== null ? this.state.titleText : '');
  data.append('lat', this.state.markerPosition.latitude);
  data.append('long', this.state.markerPosition.longitude);

  data.append('media', {
    ...file, name: this.state.imageData.fileName,
    type: 'image/jpeg',
  });

  makeRequest(url, method, headers, data)
  .then(result => {

    if ('service_request_id' in result[0]) {
      serviceRequestModels.insert(result[0]['service_request_id'])
    }
  }, error => {
    showAlert(transError.networkErrorTitle, transError.networkErrorMessage, transError.networkErrorButton);
  });
}
```
KUVA 10. Palvelupyynnön lähetys Open311-rajapintaan

6.2.3 Palvelupyyntöjen listaaminen

Rajapinnalta voi kysyä GET-pyynnöllä lista palvelupyynnöistä, antamalla pyynnölle parametreiksi aikavälin jolta palvelupyynnöt halutaan. Rajapinta palauttaa JSON-muodossa listan vastauksista, jotka jäsennellään haluttuun muotoon. Tämän jälkeen lista palvelupyynnöistä voi esittää esimerkiksi ListView-elementillä.

```
fetchServiceRequests() {
  var timeSpan = Util.getTimeSpan();
  var parameters = '?start_date=' + timeSpan.startDate + '&end_date=' + timeSpan.endDate
                 + Config.OPEN311_SERVICE_REQUESTS_EXTENSIONS_POSTFIX;
  var url = Config.OPEN311_SERVICE_REQUESTS_URL + parameters;
  var headers = {'Accept': 'application/json', 'Content-Type': 'application/json'};

  makeRequest(url, 'GET', headers, null, null)
  .then(result => {

    var serviceRequestList = Util.parseServiceRequestList(result);

    this.setState({
      serviceRequestList: serviceRequestList,
    });

  }, error => {
      showAlert(transError.networkErrorTitle, transError.networkErrorMessage,
        transError.networkErrorButton);
    }
  });
}
```

KUVA 11. Palvelupyyntöjen hakeminen Open311-rajapinnasta halutulta aikaväliltä.



KUVA 12. Palvelupyynnöt listattuna sovelluksessa ListView-elementissä.

6.2.4 Yksittäisen palvelupyynnön hakeminen

Yksittäisen palvelupyynnön tiedot voi pyytää rajapinnasta HTTP GET-pyynnöllä, antamalla parametrina haettavan palvelupyynnön tunnisteen.

```
fetchServiceRequestDetails(id) {
  var url = Config.OPEN311_SERVICE_REQUEST_BASE_URL + id + Config.OPEN311_SERVICE_REQUEST_PARAMETERS_URL;
  var headers = {'Accept': 'application/json', 'Content-Type': 'application/json'};

  makeRequest(url, 'GET', headers, null, null)
  .then(result => {
    var data = Util.parseServiceRequestDetails(result);

    this.setState({
      data: data,
    });
  }, error => {
    this.setState({ isLoading: false });
      showAlert(transError.serviceNotAvailableErrorTitle,
        transError.serviceNotAvailableErrorMessage, transError.serviceNotAvailableErrorButton);
  });
}
```

KUVA 13. Yksittäisen palvelupyynnön hakeminen Open311-rajapinnasta.

Rajapinta palauttaa alkuperäisen palvelupyynnön lisäksi tietoa palvelupyynnön käsittelyn etenemisestä, kuten mille taholle palvelupyyntö on ohjattu ja mikä sen tämänhetkinen tila on.

KUVA 14. Yksittäinen palvelupyyntö esitettynä sovelluksessa.

7 TULOKSET

7.1 Open311-rajapinnan käyttö React Native-mobiilisovelluksessa

Omalta osaltani tämä oli ensimmäinen projekti, joka toteutettiin käyttäen React Native-ohjelmistokehystä, joten lähtötaso projektiin alkaessa oli melko alhainen. React Native oli kuitenkin hyvin helposti omaksuttava ohjelmistokehys ja olin häkeltynyt sovelluksen suorituskyvystä, sillä aikaisempien kokemusteni perusteella järjestelmäriippumattomista ohjelmistokehyksistä, odotukseni eivät varsinaisesti olleet erityisen korkealla.

Open311-rajapinnan käyttö React Native-sovelluksesta oli hyvin suoraviivaista. Rajapintaan lähetettiin ja sieltä otettiin vastaan tietoa HTTP-pyynnöillä, joten ohjelmistokehys, käyttöjärjestelmä tai ohjelmointikieli ei sinällään vaikuttanut itse rajapinnan ja sovelluksen välillä kulkevan tiedonvälityksen toteutukseen. React Nativen hyvät puolet pääsivät kuitenkin oikeuksiinsa, kun rajapinnasta saatua dataa haluttiin esittää käyttöliittymässä. React Nativen uudelleen käytettäviin käyttöliittymäkomponentteihin perustuva lähestymistapa toimi hyvin tämän tapaiseen sovellukseen, sillä tarvittavat käyttöliittymäkomponentit esimerkiksi palautepyyntöjen detaljien esittämiseen oli helppo suunnitella valmiiksi ja niihin oli myös vaivatonta tehdä muutoksia myöhemmin. Käyttöliittymän ohjelmoiminen oli myös huomattavasti helpompaa, etenkin web-kehittäjälle, kuin esimerkiksi Android-projekteissa.

React Native on myös hyvä valinta Open311-rajapintaa käyttäviin sovelluksiin siksi, että rajapintaa käyttävät sovellukset perustuvat hyvin pitkälle tiedon esittämiseen käyttäjälle ja varsinaisia kovempaa suoritustehoa vaativia toimintoja ei tällöin tarvita, jolloin eroa natiivisovelluksen ja React Native-sovelluksen välillä on käytännössä mahdoton huomata käyttäjän näkökulmasta. Saattaisin kehittäjänä päätyä valitsemaan tällaisessa sovelluksessa React Nativen ennen natiivitoteutusta, vaikka sovellus olisi tarkoitus julkaista vain yhdelle käyttöjärjestelmälle.

7.2 Open311-rajapinnan potentiaalisia käyttömahdollisuuksia

Vaikka Open311-rajapinta on melko uusi rajapinta ja sen kehitys on vielä vaiheessa, on se onnistuttu ottamaan käyttöön menestyksekkäästi useissa suurissa kaupungeissa. Open311-rajapinna kaltaisessa avoimen standardin rajapinnan käytössä on monia merkittäviä hyötyjä, sekä palvelua käyttävien yhteisöjen, että rajapintaa hyödyntävien sovelluskehittäjien kannalta. Ensinnäkin, rajapinnan käyttöönotto on hyvin yksinkertaista, eikä sen käyttäminen maksa mitään. Useat kaupungit ovat päätyneet toteuttamaan vastaavia rajapintoja omien tarpeidensa mukaan, mutta tällaisten rajapintojen suunnittelu ja toteutus on kallista ja ne joudutaan yleensä tilaamaan joltakin kolmannelta osapuolelta, eikä tällaisia rajapintoja voida kuitenkaan sellaisenaan hyödyntää esimerkiksi muiden kaupunkien käyttöön. Ulkoiselta yritykseltä tilatun rajapinnan huonoina puolina on myös se, että järjestelmien ylläpito voi olla hyvinkin kallista. Etenkin valtion ja kaupunkien rahoittamissa, julkisten palvelujen ulkoisilta tahoilta tilatuissa tietojärjestelmissä, ohjelmistotalojen hinnat järjestelmien ylläpidolle ja uudistamiselle saattavat paisua jättimäisiin summiin, kuten on käynyt esimerkiksi potilastietojärjestelmien uusimisessa. (Ovaskainen, 2012)

Standardoidun avoimen rajapinnan käyttö myös takaa sen, ettei rajapinta ole riippuvainen mistään tietystä teknologiasta, vaan sitä voidaan käyttää ristiin eri valmistajien järjestelmissä, sovelluksissa ja palveluissa. Esimerkiksi web-pohjainen palvelu, jolla seurataan vaikkapa kaupungissa käynnissä olevia tietöitä ja rakennushankkeita, sekä mobiilisovellus, joka on tarkoitettu esimerkiksi liikenneruuhkien seurantaan, pystyisivät hyödyntämään samaa rajapintaa sekä tietokantaa, ilman erillistä integraatiota palveluiden välillä. Eikä rajapinnan hyödyntäminen ristiin eri palveluiden ja sovellusten välillä rajoitu pelkästään saman järjestelmän käyttöön, vaan potentiaalisesti yhden kaupungin tarpeisiin tarkoitettua palvelua voidaan sellaisenaan suoraan käyttää myös muiden standardoitua rajapintaa toteuttavien kaupunkien välillä. Juuri tässä järjestelmien ristiin yhteensopivuudessa tuntuukin piilevän rajapinnan suurin potentiaali, jota rajapinnan kehittäjät myös tuovat kovasti esiin (Open311.org, 2017). Valitettavasti kuitenkin käytännössä kaikki tällä hetkellä Open311-rajapintaa käyttävät sovellukset ja palvelut on suunniteltu ja kehitetty jonkin tietyn kaupungin tai yhteisön tarpeisiin, eivätkä tällaiset sovellukset pysty hyödyntämään muiden kaupunkien vastaavia järjestelmiä, huolimatta siitä, että niissä käytetään samaa standardoitua rajapintaa. (Rakesh, 2015, 57-58) Rajapinta on kuitenkin vielä suhteellisen uusi ja rajapinnan kehitys on yhä vaiheessa, joten nämäkin sovellukset

auttavat kehittämään rajapintaa vastaamaan paremmin erilaisiin tarpeisiin. Rajapinnan kehittyessä, toivottavasti aletaan nähdä yhä laajemmin tätä hyödyntäviä sovelluksia.

Open311-rajapinnan tämänhetkinen toteutus rajoittaa rajapinnan käyttöä pitkälti nimenomaan palautteen lähettämiseen ja seurantaan, mutta tämä on vasta jäävuoren huippu. Rajapinnan kehittyessä sillä on potentiaalia toimia työkaluna kaikenlaisessa hallituksen ja kansalaisten välisessä toiminnassa, kuten esimerkiksi apuna Kelan hakemuskäsittelyssä tai vaikkapa taloyhtiöiden pyykki- ja saunavuorojen varaamiseen.

Kaupallisessa mielessä sovelluskehittäjien ja ohjelmistotalojen näkökulmasta, Open311-rajapintaa käyttävien sovellusten monetisointi sotii avoimen rajapinnan periaatteita vastaan, mutta ohjelmistotalojen kannattaa tutustua rajapintaan ja miettiä potentiaalisia palveluita joihin tätä voisi hyödyntää, sillä varsinkin Suomessa Open311-rajapintaa hyödyntäviä palveluita ja sovelluksia on toistaiseksi hyvin vähän. Helsingissä rajapintaa kuitenkin kehitetään jatkuvasti ja sillä on jo nyt tuhansia aktiivisia käyttäjiä. Rajapinnan käyttöönoton tuomat taloudelliset ja sosiaaliset hyödyt saattavat tulevaisuudessa innostaa myös muita kaupunkeja ottamaan käyttöönsä vastaavan järjestelmän, joten tarve Open311-rajapintaa hyödyntäville sovelluksille ja palveluille voi kasvaa äkillisesti, ja tällöin aiempi osaaminen rajapinnan parissa voi tuoda merkittävän edun yritykselle valittaessa toteuttajaa tällaisille projekteille.

8 POHDINTA

Tämän opinnäytetyön tavoitteena oli selvittää Open311-palvelurajapinnan käytön kannattavuutta React Native-ohjelmistokehyksellä, sekä miettiä rajapinnan potentiaalia ja hyödyntämistä muissa projekteissa. Työn aikana Helsingin kaupungille toteutettiin Open311-rajapintaa käyttävä React Native-mobiilisovellus, joka toi paljon käytännön kokemusta rajapinnan hyödyntämisestä ohjelmistoprojektissa sekä vastauksia opinnäytetyön tavoitteisiin.

Rajapinnan käyttö React Native-ohjelmistokehyksellä osoittautui hyvin suoraviivaiseksi ja suhteellisen ongelmattomaksi, vaikkakaan ohjelmistokehys itsessään ei tuonut erityisiä hyötyjä Open311-rajapinnan käyttöön, muiden järjestelmäriippumattoman sovelluskehityksen hyötyjen lisäksi. React Native on kuitenkin vahva valinta toteuttaessa Open311-rajapintaa hyödyntävää sovellusta, sillä React Native-mobiilisovellusten suorituskyky on lähes natiivisovellusten veroinen tämän tyyppisissä sovelluksissa, jotka eivät vaadi niin suurta suoritustehoa laitteelta.

Tämän työn tulosten perusteella Open311-rajapinnan suurin potentiaali vaikuttaisi olevan rajapinnan standardoidussa mallissa, joka mahdollistaa useiden eri rajapintaa käyttävien järjestelmien käyttämisen ristiin, jopa globaalissa mittakaavassa. Rajapinnan suurinta potentiaalia hyödyntäviä sovelluksia on vielä toistaiseksi suhteellisen vähän, joten siinä on osa-alue johon ohjelmistotalojen, kuten projektin toimeksiantajan, Haltu Oy:n, kannattaa ehdottomasti keskittyä, sillä rajapintaa kehitetään jatkuvasti ja se on jo nyt osoittautunut joustavuutensa erilaisiin käyttötarkoituksiin.

LÄHTEET

Goodyear, S. 2015. 3-1-1: A City Services Revolution. Luettu 26.2.2017. http://www.citylab.com/city-makers-connections/311/

Steinberg, T. 2013. Open311 - What is it, and why is it good news for both governments and citizens? Luettu 11.3.2017 https://www.mysociety.org/2013/01/10/open311-introduced/

Open311 API. Luettu 16.2.2017. www.open311.org

Abrosimova, K. 2014. Native VS Cross-Platform. Luettu 26.2.2017. https://yalantis.com/blog/native-vs-cross-platform-app-development-shouldnt-work-cross-platform/

Ovaskainen, T. 2012. 1 800 000 000 €:n hanke: Tämä on Suomen ja Viron ero. Uusi Suomi. Luettu 22.3.2017. https://www.uusisuomi.fi/kotimaa/53415-1-800-000-000-eun-hanke-tama-suomen-ja-viron-ero-sanoo-sitran-antti-kivela

Patel, R. 2015. A Guide to Open311. Partridge Publishing. E-kirja.

Eberhardt, C. 2015. Retrospective on developing an application with React Native. Luettu 23.2.2017. http://blog.scottlogic.com/2015/03/26/react-native-retrospective.html

GeoReport v2 -dokumentaatio. Open311 Wiki. Luettu 22.2.2017 http://wiki.open311.org/GeoReport_v2/

Open311 Helsingin kaupungin dokumentaatio. Luettu 22.2.2017. https://dev.hel.fi/apis/open311

City of Helsinki's issue reporting API open for developers. 2013. Forum Virium. Luettu 20.3.2017. https://forumvirium.fi/en/city-of-helsinkis-issue-reporting-api-open-for-developers/