Eetu Pajuoja

Analytiikkatyökalut Android-sovelluskehityksessä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

11.2.2016

Metropolia

Tekijä Otsikko Sivumäärä Aika	Eetu Pajuoja Analytiikkatyökalut Android-sovelluskehityksessä 49 sivua 11.2.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaaja	Yliopettaja Petri Vesikivi

Insinöörityön tavoitteena oli toteuttaa Android-sovellus, joka kertoo ajantasaista tietoa käyttäjän valitsemalla reitillä kulkevista junista hyödyntäen rata.digitraffic.fi-rajapintaa. Tämän lisäksi tavoitteena oli myös toteuttaa sovellukselle betatestaus ja kaatumisten raportointi sekä seurata sovelluksen käyttöä ja käyttäjien määrää analytiikkatyökalujen avulla. Työ tehtiin suomalaiselle IT-alan yritykselle, jolla ei ole aikaisempaa kokemusta mobiilisovelluksien ja niiden analytiikan toteuttamisesta. Työn tarkoituksena olikin tutkia ja kokeilla mobiilisovelluksen analytiikan toteuttamista mahdollisia tulevia projekteja varten.

Sovellus toteutettiin Android Studiossa. Analytiikkaa varten työssä otettiin käyttöön työkalu, jota yritys on aikaisemmin käyttänyt vain web-analytiikkaan. Tämän työkalun avulla toteutettiin sovelluksen käytön ja käyttäjien määrän seuranta. Tämän lisäksi käytettiin toista työkalua, joka valittiin sen ominaisuuksien, ilmaisuuden ja helppokäyttöisyyden vuoksi. Tällä työkalulla toteutettiin sovelluksen betatestaus ja kaatumisten raportointi sekä vertailun vuoksi myös käytön seuranta. Työn aikana analytiikkatyökaluihin tutustuttiin tarkemmin ja niiden yhtäläisyyksiä vertailtiin. Tämän lisäksi työssä perehdyttiin mobiilisovellusanalytiikan perusteisiin ja käytäntöihin.

Sovelluksen jakelu testausta varten onnistui vaivatta ja kaatumisten raportoinnin avulla löydettiin monta kriittistä virhettä ennen sovelluksen julkaisua ja julkaisun jälkeen. Käyttäjien määriä seuraamalla huomattiin odotusten mukaisesti, ettei sovellus saavuttanut suurta suosiota. Käytön seurannan avulla ymmärrettiin, että sovellusta käytetään enimmäkseen tärkeimmän käyttötapauksen mukaisesti ja että sovelluksen tiettyjä lisäominaisuuksia käytetään melko vähän.

Tulosten perusteella sovelluksessa käytetty analytiikka todettiin onnistuneeksi ja käytetyt työkalut hyödyllisiksi. Mobiilisovelluksen analytiikka on kuitenkin mahdollista toteuttaa monella eri tavalla, joten työssä käytetty tapa ei ole ainoa oikea. Kokeilujen myötä yrityksen mobiiliprojekteissa käytetään jatkossa todennäköisesti samoja työkaluja.

Avainsanat	analytiikka, mobiilisovellusanalytiikka, Android, mobiilisovellukset

Author Title Number of Pages Date	Eetu Pajuoja Analytics tools in Android software development 49 pages 11 February 2016
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructor	Petri Vesikivi, Principal Lecturer

The objective of the final year project was to develop an Android application, which displays up-to-date information about trains on a user-selected route using the rata.digitraffic.fi interface. In addition the objective was also to conduct a beta test and implement crash reporting for the application and to follow the usage and the amount of users of the application using mobile analytics tools. The purpose of the project was to explore and experiment ways of implementing analytics for a mobile application for a Finnish IT company which does not have much experience in developing mobile applications.

The application was developed in Android Studio. For observing the usage and the amount of users an analytics tool previously known for the company from web developing was used. Additionally another tool was utilized for beta testing and crash reporting. The tool was chosen for its features and ease of use and because it was free of charge. This thesis focuses on these analytics tools as well as the practices and basics of mobile analytics.

Distributing the application for the beta test was effortless and many critical bugs were found before and after the release of the application with crash reporting. By tracking the amount of new users it was noticed that the application did not gain much popularity, which was expected. Tracking the usage of the application revealed that the application was used as expected and that some smaller additional features of the application were not used that actively.

Based on the results the analytics used in the application was considered a success and the tools used useful. It is possible to implement mobile analytics in many ways; the method used in this project is not the only right way. The result of the experimentation is that the company will likely use the same tools used in the project in their future projects.

Keywords	analytics, mobile application analytics, Android, mobile applications

Sisällys

Lyhenteet

SDK	Software Development Kit. Kokoelma työkaluja, joilla voi luoda tiettyjä ohjelmistoja.
APK	Android Application Package. Asennettavien ja jakelussa olevien Android-sovellusten tiedostomuoto.
NDK	Native Development Kit. Kokoelma Android-työkaluja, joiden avulla voi kehittää Android-sovelluksia C:llä ja C++:lla.
HTTP	Hypertext Transfer Protocol. Protokolla, jota käytetään tiedonsiirtoon verkossa.

1 Johdanto

Insinöörityön tavoitteena on toteuttaa Android-käyttöjärjestelmälle mobiilisovellus, joka kertoo ajantasaista tietoa junista käyttäjän määrittelemille reiteille hyödyntäen Liikennevirasto n omistamaa rata.digitraffic.fi-rajapintaa. Samalla tavoitteena on myös tutustua mobiilisovellusten analytiikkaan ja toteuttaa eri työkalujen avulla sovelluksen kaatumisten raportointi ja betatestaus sekä seurata sovelluksen käyttöä ja hankintaa sen julkaisun jälkeen.

Työn tilaajayritys on suomalainen IT-alan yritys, jolla on parin vuosikymmenen kokemus alalta. Yrityksellä ei kuitenkaan ole vielä kokemusta mobiilisovellusten toteuttamisesta. Työn tarkoituksena onkin siis tutkia kahta eri työkalua, joiden avulla voi toteuttaa mobiilisovelluksen analytiikan.

Insinöörityöraportti keskittyy mobiilisovellusanalytiikkaan. Raportissa verrataan mobiilisovellusten analytiikkaa web-analytiikkaan, selvitetään mobiilianalytiikalle ominaisia suorituskykymittareita ja tutkitaan käytännön strategioita ja käytäntöjä, jotka on mobiilisovelluksen analytiikkaa suunnitellessa hyvä ottaa huomioon. Tämän lisäksi raportissa tutkitaan kahta eri analytiikkatyökalua, joiden yhtäläisyyksiä myös vertaillaan toisiinsa.

Työ toteutetaan Android Studiolla, joka on Googlen ainoa virallisesti tukema Android-sovelluskehitysalusta. Työn analytiikkapuoli toteutetaan Google Analytics- ja Fabric-työkaluilla, joista Google Analyticsia yritys on käyttänyt myös ennen web-analytiikkaan. Fabric puolestaan on uusi työkalu yritykselle. Tämän lisäksi työn aikana sovellusta testataan Testdroid Cloud -pilvipalvelussa, jonka avulla sovelluksesta etsitään virheitä.

2 Mobiilisovellusanalytiikka

2.1 Yleistä

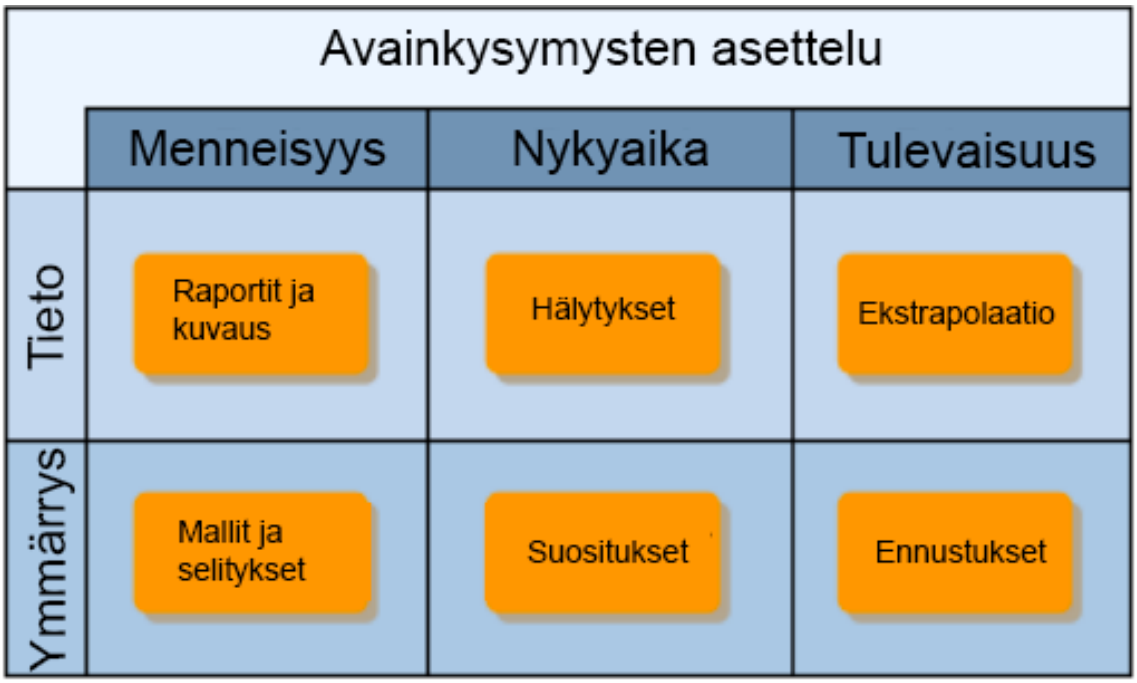
Vuonna 2008 internetiä käyttävien mobiililaitteiden määrän ennakoitiin ohittavan tietokoneiden ja kannettavien tietokoneiden määrän. Vuonna 2014 ennustus kävi toteen Yhdysvalloissa. Yhdysvaltalaisen internetanalytiikkayrityksen ComScoren raportin mukaan yhdysvaltalaisten digitaalisen median käytöstä vuonna 2014 40 prosenttia tapah-

tui tietokoneilla, 8 prosenttia mobiililaitteiden selaimilla ja 52 prosenttia mobiililaitteiden sovelluksissa. [1, s. 4] Mobiilisovellusten käytön kasvaessa luonnollisesti myös sovellusten ja niiden latauksien määrä kasvaa. Vuonna 2009 sovelluksia ladattiin maailmanlaajuisesti yhteensä 2,516 miljardia, ja vuonna 2017 latauksia ennustetaan tulevan 268,692 miljardia. Kesäkuussa 2015 pelkästään Googlen Play -sovelluskaupassa oli ladattavissa 1,6 miljoonaa eri sovellusta. [2] Toisin sanoen sovelluksille on paljon kysyntää ja tarjontaa, joten kilpailu niiden välillä on kovaa.

Nykyään markkinoiden lisäksi myös käyttäjät ovat vaativia. He odottavat sovelluksien olevan älykkäitä ja toimivan nopeasti missä tahansa ja millä laitteella tahansa. [3, s. 1] Jos sovellus ei onnistu vastaamaan käyttäjän odotuksiin, se poistetaan laitteesta ja tilalle ladataan jokin toinen vaihtoehtoinen sovellus. Koska Google Play -sovelluskauppa mittaa sovellusten laatua käyttäjien arvosanojen perusteella asteikolla 0–5, sovelluskehittäjille ei selvene, miksi heidän sovelluksensa ei menesty. Arvosanan perusteella on mahdoton tietää, johtuuko huono menestys sovelluksen huonosta suorituskyvystä, epävakaudesta, ominaisuuksien puutteesta, sekavasta käyttöliittymästä tai jostain muusta. Sovelluskehittäjät eivät siis tiedä, mitä korjata ja miten kehittää sovellusta jatkossa. [3, s. 2] Ratkaisu ongelmaan on analytiikka.

2.2 Analytiikka yleisesti

Analytiikalla tarkoitetaan tiedon keräämistä ja analysointia, jonka perusteella tehdään päätöksiä. Analytiikan avulla voidaan tarkkailla menneitä tapahtumia ja ennustaa uusia sekä tehdä tietoon perustuvia johtopäätöksiä arvailun sijaan. [4, s. 3] Mobiilisovellusanalytiikan avulla voidaan seurata, miten sovellus käyttäytyy ja miten käyttäjät käyttävät sovellusta. Analytiikka vastaa moniin sovelluksen parissa työskentelevien kysymyksiin, jotka Adam Cooper tiivistää avainkysymyksiin verkkojulkaisussaan CETIS analytics series volume 1, no 9: A brief history of analytics kuvan 1 mukaisesti.



Kuva 1. Analytiikan avainkysymysten taulukko [muokattu lähteestä 4, s. 4].

Kuvassa 1 on eroteltu vaakariveillä tosiasioihin ja tietoon liittyvät kysymykset ja analytiikan antamien tulosten syvempään ymmärrykseen liittyvät kysymykset. Sen lisäksi kysymykset on jaettu ajan mukaan pystyriveillä. Tieto-rivin määrittelemät kysymykset aikajärjestyksessä ovat ”Mitä tapahtui?”, ”Mitä tapahtuu tällä hetkellä?” ja ”Mihin suuntaan trendit ovat menossa?” Analytiikan tuottamat raportit ja yhteenvedot kerätystä tiedosta vastaavat kysymykseen ”Mitä tapahtui?” Analytiikan tarjoamat hälytykset puolestaan vastaavat kysymykseen ”Mitä tapahtuu tällä hetkellä?” ja kerätyn tiedon ekstrapolointi kysymykseen ”Mihin suuntaan trendit ovat menossa?” Ymmärrys-rivin kysymykset ovat ”Miksi ja miten jotakin tapahtui?”, ”Mitä kannattaa tehdä seuraavaksi?” ja ”Mitä todennäköisesti tapahtuu?” Analytiikkatyökalut voivat tuottaa selityksiä, suosituksia ja ennustuksia, jotka vastaavat näihin kysymyksiin, mutta myös analyytikot voivat antaa vastauksia analytiikan keräämän tiedon perusteella. [4, s. 4]

2.3 Web-analytiikka verrattuna mobiilisovellusanalytiikkaan

Viimeisen vuosikymmenen aikana web-analytiikka on kasvanut ja kehittynyt paljon, minkä myötä siitä on tullut välttämätön työkalu yrityksille. Ennen analytiikkaa verkkosivujen markkinointikampanjoiden tuloksia oli vaikea huomata välittömästi, koska palautetta saatiin lähinnä asiakaskyselyiden avulla. Googlen julkaistua Google Analy-

ticsin vuonna 2005 web-analytiikka tuli kuitenkin kaikkien saataville ja sen käyttö lisääntyi huomattavasti. [5, s. 15]

Web-analytiikan avulla voidaan seurata sivuston kävijöitä ja auttaa yritystä saavuttamaan sen markkinointitavoitteet luomalla kohdistettuja markkinointikampanjoita. Web-analytiikassa tärkeimmät seurattavat mittarit ovat sivuston kävijät, käynnit, sivun katselut ja tapahtumat. Verkkosivujen käyntien mittaaminen ei kuitenkaan ole suoraviivaista, sillä käyttäjät jättävät verkkosivuja auki taustalle, käyvät toistuvasti samoilla sivuilla unohtaessaan asioita ja saattavat estää evästeiden käytön. Nykyään kokemuksen kartuttua yritykset osaavat kuitenkin keskittyä mittareihin, jotka kertovat sivuston olennaisesta käytöstä. [6, s. 5]

Vuosien myötä opitut käytännöt web-analytiikassa eivät kuitenkaan täysin päde mobiilisovellusanalytiikkaan. Tämä tarkoittaa sitä, että sovellusten analysointi joudutaan opettelemaan uudelleen. Pääsyy tähän on, että mobiilisovelluksia käytetään eri tavalla ja ne ovat rakenteeltaan erilaisia kuin verkkosivut. Esimerkiksi sovelluksissa vietetään yleensä vähemmän aikaa kerralla kuin verkkosivuilla, mutta vietetty aika käytetään tehokkaammin. [6, s. 4]

Mobiilisovellusanalytiikassa keskitytään näkymiin ja käyttäjän istuntoihin sivujen katselukertojen ja vierailujen sijaan. Yksi näkymä sovelluksessa ei vastaa verkkosivuston yhtä sivua, sillä etenkin Android-sovelluksissa yhden näkymän määrittely ei ole suoraviivaista. Sovelluksissa istuntojen aikakatkaisu on huomattavasti pienempi – jopa 15 sekuntia – kuin verkkosivustojen vierailujen – 30 minuuttia. [6, s. 5] Toisin sanoen sovellusten istunnot ovat lyhempiä ja niitä on enemmän kuin verkkosivujen vierailuja. Mobiilisovelluksissa yksittäiset käyttäjät tunnistetaan tunnisteiden avulla, kun verkkosivuilla käytetään evästeitä. Tunnisteet ovat yleensä tarkempia yksittäisten käyttäjien tunnistamiseen, sillä mobiililaite on usein henkilökohtainen, toisin kuin tietokone, jota saattaa käyttää monta henkilöä. Evästeet on myös mahdollista poistaa käytöstä, jolloin käyttäjän tunnistamista ei verkkosivuilla tapahdu. [7]

Kuva 2 kerää yhteen eroavaisuuksia PC- ja verkkosovellusten ja mobiilisovellusten analytiikan välillä. Mobiilisovellusten analytiikkaan on tarjolla huomattavasti enemmän työkaluja, mikä tekee mobiilianalytiikasta haastavampaa, mutta luo samalla enemmän mahdollisuuksia.

ANALYTIIKAN KEHITYS

■ Aina □ Joskus ▨ Harvoin / ei koskaan

MITTARIT	PC / VERKKO	MOBIILI
POIKKEUSTEN HALLINTA	■	■
SOVELLUKSEN HANKINTA	□	■
SITOUTUMINEN	□	■
KÄYTTÖTAPAHTUMAT	□	■
KÄYTTÄJIEN SÄILYVYYS	□	■
SUPPILOANALYYSI	▨	■
JOUKKOANALYYSI	▨	■
SOVELLUKSEN AVAAMINEN JA SULKEMINEN	▨	■
SOVELLUKSEN VERSIO	▨	■
SOVELLUSTA KÄYTTÄVÄ LAITE	▨	■
KÄYTTÄJÄN SIJAINTI	▨	■
KÄYTTÄJÄN LIIKE	▨	■
ONLINE / OFFLINE SYNKRONISAATIO	▨	■
LAITTEEN ORIENTAATIO	▨	■

Kuva 2. Analytiikan kehitys PC- ja verkkosovellusten ja mobiilisovellusten välillä [muokattu lähteestä 3, s. 3].

Erityisen tehokkaita mobiilisovellusanalytiikan työkaluja ovat suppilo- ja joukkoanalyysi. Suppiloanalyysillä tarkoitetaan käyttäjien seuraamista tietyssä ennalta määritellyssä tapahtumaketjussa. Esimerkiksi jos käyttäjät haluttaisiin saada rekisteröitymään maksulliseen uutissovellukseen, tapahtumaketju voisi olla seuraavanlainen: päänäkymän katselu, valitse ilmainen uutinen, valitse rekisteröinti, rekisteröidy. Suppiloanalyysin avulla voidaan seurata, kuinka moni käyttäjä suorittaa koko tapahtumaketjun ja missä vaiheessa käyttäjiä putoaa pois ketjusta, mikä auttaa ymmärtämään sovelluksen pullonkauloja ja suunnitteluvirheitä. [3, s. 4] Joukkoanalyysin avulla puolestaan voidaan tutkia tiettyjen joukkojen käyttäytymistä sovelluksessa. Joukkoja voi määritellä monella perusteella, kuten maantieteellisen sijainnin perusteella, ostotapahtumien perusteella tai sovelluksen hankinta-ajankohdan perusteella. Joukkoanalyysi kertoo, miten nämä ryhmät eroavat muista käyttäjistä, minkä perusteella voidaan pohtia sovelluksen jatkokehitystä. Jos esimerkiksi tietty sijainnin perusteella luotu joukko käyttää vähemmän rahaa sovelluksessa kuin keskivertokäyttäjät, voidaan kehittää keskitetty markkinointikampanja, jonka tavoitteena on saada tämä joukko kuluttamaan enemmän. Suppilo- ja joukkoanalyysi ovat Google Analyticsin keskeisiä työkaluja, joten niiden käyttö on yleistä myös verkkosivuilla, toisin kuin kuvasta 2 ilmenee. [8]

Mobiilisovelluksissa voi myös kerätä tietoa muusta kuin pelkästään sovelluksen tapahtumista, kuten laitteen GPS-sijainnista, kiihtyvyydestä tai tallennustilasta. Tietoa on myös mahdollista kerätä silloin, kun laitteella ei ole internetyhteyttä. Tämä kerätty tieto lähetetään, kun verkkoyhteys on seuraavan kerran saatavilla. [7]

Eroista huolimatta web-analytiikan ja mobiilisovellusanalytiikan tavoite on sama: ymmärtää, miten palvelua käytetään ja miten palvelu käyttäytyy sekä saada tietoa markkinointikampanjoiden vaikutuksesta palvelun käyttäjämääriin, minkä pohjalta tehdä muutoksia palveluun. [7]

2.4 Mittarit

Mittareilla tarkoitetaan suorituskykymittareita, joiden avulla voidaan seurata sovelluksen menestystä ajan kuluessa verrattuna sovellukselle määriteltyihin tavoitteisiin. Mittareiden kehitystä seurataan raporteista, jotka esittävät analytiikkapalveluiden keräämää tietoa. Yleisin mittari on sovelluksen latausmäärä. Latausmäärät eivät kuitenkaan kerro suoraan sovelluksen menestyksestä, sillä latauksen jälkeen sovellus saatetaan käynnistää kerran, minkä jälkeen se unohdetaan tai poistetaan kokonaan. Mittareita onkin olemassa hyvin paljon erilaisia. Niitä kaikkia ei kuitenkaan kannata seurata yhtä aikaa, vaan seurattavat mittarit tulee valita riippuen sovellukselle asetetuista tavoitteista. [9]

Appcelerator kuitenkin määrittelee white paperissaan Stop guessing, start measuring: 5 mobile metrics for great apps viisi erittäin keskeistä mittaria, joiden tulisi olla jokaisessa sovelluksessa: hankinta, sitoutuminen, säilyvyys, muuntaminen ja laatu. Hankinta mittaa sovellusten latausmäärää eri sovelluskaupoista. Latausmäärä ei itsessään kerro sovelluksen käytöstä mitään, joten sovelluksen on myös mitattava käyttäjien sitoutumista ja sovelluksen säilyvyyttä. Sitoutumiseen sisältyy sovelluksen käytön yleisyys ja istuntojen kesto. Käytön yleisyydellä tarkoitetaan sitä, kuinka usein käyttäjät keskimäärin avaavat sovelluksen. Istunnon kesto puolestaan kertoo, kuinka kauan käyttäjät viihtyvät sovelluksessa sen avattuaan. Säilyvyys vertailee aktiivisten käyttäjien määrää latausmäärään, mikä kertoo, kuinka moni sovelluksen ladanneista on lopettanut sovelluksen käytön. Jos sovelluksen säilyvyys on pieni, on selvitettävä, miksi käyttäjät hylkäävät sovelluksen vain yhden tai kahden käyttökerran jälkeen.

Muuntaminen seuraa käyttäjien kulkua sovelluksen liiketoimintatapahtumissa, kuten osto- tai rekisteröintitapahtumissa. Jos sovelluksen tarkoitus on myydä tuotteita, sovelluksen liiketoimintatapahtuma voisi olla seuraavanlainen: tuotteen valinta, tuotteen siirtäminen ostoskoriin, maksutavan valitseminen ja tilauksen vahvistaminen. Muuntaminen kertoo, kuinka moni sovelluksen avanneista käyttäjistä kävi läpi koko tapahtuman sekä missä vaiheessa käyttäjät peruuttivat tapahtuman. Tämä auttaa löytämään suunnitteluvirheitä tai muita kohtia sovelluksesta, jotka rohkaisevat käyttäjiä lopettamaan sovelluksen käytön.

Laadun mittaamisella tarkoitetaan niiden istuntojen määrää, joissa sovellus kaatuu poikkeuksen tapahtuessa. Jos kaatumisia tapahtuu enemmän kuin yhdessä sessiossa kahdestakymmenestä, on Appceleratorin mukaan kyseessä vakava ongelma. [3, s. 4] Kaatumisten vähentäminen on kuitenkin kriittistä käyttökokemuksen ja käyttäjien säilyvyyden kannalta, joten kaikista kaatumisista tulisi päästä eroon. Sovelluksen laatuun liittyvät myös sovelluksen käynnistys- ja latausajat. Jos käynnistys- ja latausajat ovat pitkiä, on syytä tehdä parannuksia sovellukseen. Käyttäjät eivät jaksa odotella pitkiä aikoja, vaan he olettavat, että sovellus toimii sujuvasti ilman katkoja sen käytössä. Pitkät odotusajat ja katkot saattavat saada käyttäjät hylkäämään sovelluksen. [10]

Muita hyviä mittareita ovat elinikäinen arvo ja hankinnan hinta. Elinikäisellä arvolla mitataan yhden käyttäjän tuottamaa arvoa yritykselle sovelluksessa koko sovelluksen käytön ajan. Se, mitä arvolla tarkoitetaan, riippuu sovelluksesta: esimerkiksi kauppasovelluksen arvo on käyttäjän tekemät ostokset ja uutissovelluksen arvo on käyttäjän katsomat mainokset. Elinikäisen arvon perusteella voidaan vertailla sovelluksen käyttäjien tuottamaa arvoa sovellusta vastaavien verkkosivujen käyttäjien tuottamaan arvoon ja arvioida, onko mobiilimarkkinointistrategia toiminut ja onko mobiilisovellus ylipäätään kannattava. Pohtia voi myös, miksi joidenkin käyttäjien elinikäinen arvo on suurempi kuin toisten ja miten käyttäjien elinikäistä arvoa voisi nostaa. [9] Hankinnan hinnalla puolestaan mitataan sitä, kuinka paljon yhden käyttäjän hankkiminen on maksanut. Käytännössä siis vertaillaan markkinointiin käytettyjä varoja uusien käyttäjien määrään. Hankinnan hinnan avulla voidaan löytää sovelluksen kohdeyleisö ja tavat, joilla kohdeyleisö tavoitetaan parhaiten. Hankinnan hintaa kannattaa seurata etenkin silloin, kun käynnissä on mainoskampanja. [9]

2.5 Analytiikkastrategia

Sovelluskehityksessä mobiilimarkkinointistrategian määrittely ja suunnittelu saatetaan jättää vähälle huomiolle ja sen sijaan keskitytään pelkästään itse sovelluksen kehitykseen. Sovelluksia julkaistaan yhdelle tai useammalle alustalle toivoen, että yksi tai useampi niistä saa tarpeeksi huomiota ja nousee latauslistojen yläpäähän. Pelkkä sovelluksen julkaisu sovelluskaupassa ei kuitenkaan riitä menestykseen, vaan sovelluksen latausmäärien ja käyttäjien värväämisen eteen on nähtävä vaivaa. Pelkästään sovellusten latausmäärien seuraaminen ei myöskään riitä, sillä moni käyttäjä kokeilee sovellusta vain kerran. Jotta sovelluksen mahdollisuudet menestyä sovelluskaupassa nousisivat, on sovelluksen käytöstä kerättävä paljon erilaista tietoa, mikä on mahdollista analytiikan avulla. [11, s. 4]

Jotta analytiikasta saadaan mahdollisimman paljon hyötyä sovelluskehityksessä, on sovellukselle määriteltävä erillinen analytiikkastrategia. Strategian tulisi määritellä, miten analytiikkaa käytetään saavuttamaan sovellukselle annetut tavoitteet, mitä työkaluja analytiikkaan käytetään, kuka asentaa analytiikan sovellukseen ja miten, mihin kysymyksiin analytiikan avulla halutaan vastata, mitä asioita analytiikalla pitäisi mitata ja miten kerätty tieto esitetään. Erin Copper jakaa analytiikkastrategian kolmeen osaan white paperissaan Mobile whitepaper: overcoming primary obstacles to effective mobile analysis: strateginen linjaus, teknologia ja infrastruktuuri ja ymmärrys ja parannukset. [12, s. 4]

Strateginen linjaus määrittelee tärkeimmät tavoitteet sovellukselle, eli mitä sovelluksella oikeastaan yritetään tehdä. Riippuen sovelluksesta tavoitteita voivat olla esimerkiksi myydä tietty määrä tuotteita tai kehittää yrityksen tunnettavuutta ja luoda uusi kanava, jonka kautta kuluttajat ja asiakkaat voivat vuorovaikuttaa yrityksen kanssa. Linjauksesta ilmenee myös, milloin yritys tietää onnistuneensa tavoitteissaan. Onnistumisen voi määritellä esimerkiksi liikevaihdolla, aktiivisten käyttäjien määrällä tai istuntojen keskimääräisellä kestolla. Linjaukseen kuuluu myös selvitys siitä, miten analytiikkaa käytetään, jotta voidaan onnistua tavoitteissa. [12, s. 6]

Teknologia ja infrastruktuuri määrittelee sovelluksessa käytettävät analytiikkatyökalut, tiedonkeräysstrategian, joilla sovelluksesta saadaan kerättyä jatkokehityksen kannalta oleellinen tieto, ja sen, miten tiedonkeräysstrategiat toteutetaan. Työkaluja mobiilisovellusten analysointiin on monia erilaisia, jolloin jokaiseen tarpeeseen löytynee analytiik-

katyökalu. Eri työkalut tarjoavat eritasoisia palveluita eri hinnoitteluilla, mikä tekee työkalujen päättämisestä haastavaa. Ei ole myöskään harvinaista, että sovellus hyödyntäisi enemmän kuin yhtä työkalua. Monet johtavista webanalytiikkatyökaluista tarjoavat keinoja mobiilisovellusten analysointiin, mutta nykyään on myös olemassa pelkästään mobiilisovelluksille suunnattuja analytiikkapalveluja. Nämä palvelut voivat antaa kattavampaa tietoa laitteista, käyttäjistä ja sovelluksen suorituskyvystä. [12, s. 7]

Tiedonkeräysstrategian laatiminen alkaa vaatimusten ja mittareiden määrittelyllä. Kaikille mobiilialustoille voi määrittää yhteiset mittarit, joilla saa yleiskuvan koko projektin tilasta. Alustakohtaisilla mittareilla, kuten liikevaihto istuntoa kohti, voidaan tarkastella eri alustojen menestystä verrattuna toisiinsa ja tavoitteiden saavuttamista. Näiden lisäksi voi myös määritellä muita mittareita, kuten käyttökokemusta mittaavia mittareita, jotta saadaan kerättyä tarpeeksi tietoa perusteellista analyysiä varten. Tämän jälkeen sovitaan käytännöt, joiden mukaan analytiikka otetaan sovelluksessa käyttöön. Käytäntöjä sopiessa on hyvä muistaa, että sovellus todennäköisesti muuttuu ja kasvaa tulevaisuudessa, mikä vaikuttaa analytiikan käyttöönottoon. [12, s. 8]

Tiedonkeräysstrategiaa laatiessa on hyvä ottaa huomioon, että mobiilisovellusanalytiikan voi jakaa kahteen osaan: toimintaan ja käyttöön perustuvaan analytiikkaan. Toiminnallinen analytiikka kerää tietoa sovelluksen käyttäytymisestä, kuten latausajoista ja kaatumisista, sekä laitteista, joilla sovellusta käytetään. Toiminnallisen analytiikan tarkoituksena on havaita virheitä ja ongelmia sovelluksen suorituskyvyssä. Käyttöön perustuvan analytiikan avulla saadaan tietoa siitä, miten sovellusta käytännössä käytetään. Käyttöanalytiikan avulla siis voidaan tarkkailla, mitä sovelluksen osia käyttäjät käyttävät ja millaisia polkuja he sovelluksessa kulkevat. Tämän perusteella puolestaan voidaan tehdä päätelmiä sovelluksen eri osien käytettävyydestä ja hyödyllisyydestä, mikä saattaa vaikuttaa sovelluksen jatkokehitykseen. [13]

Analytiikasta saadun tiedon ymmärtäminen ja parannusehdotusten laatiminen ymmärryksen pohjalta on analytiikkastrategian haastavin osuus. Vaikeus johtuu lähinnä siitä, että analytiikassa käytetään monia eri työkaluja ja tietoa kerätään monella eri tavalla. Tämän takia kerätty tieto on itseään toistavaa, hajanaista ja sekavaa, eikä siihen kaikkeen pääse käsiksi yhdestä paikasta. [14] Jotta kerättyä tietoa voitaisiin verrata keskenään, se on kerättävä yhteen paikkaan. Tämä onnistuu kolmannen osapuolen työkaluilla, kuten Spotfire ja Klipfolio, mutta sen voi toteuttaa yksinkertaisesti myös Microsoft Excelillä. Spotfire ja Klipfolio keräävät tietoa monista eri lähteistä, kuten sovelluskau-

poista ja analytiikkapalveluista, ja näyttävät tiedot muokattavissa raporteissa. [12, s. 12] Raportit on hyvä rakentaa niin, että niistä saa nopeasti selville halutut tiedot. Raporteista tulisi ainakin käydä ilmi mittareiden arvot tällä hetkellä ja menneisyydessä sekä verrattuna yrityksen tavoitteisiin. Pelkät mittarit eivät välttämättä anna syvällistä tietoa sovelluksen tilasta, joten mittareiden lisäksi raporteissa kannattaa esittää tietoa muista sovellukselle olennaisista ominaisuuksista, kuten haku- tai ostostapahtumista. [12. s. 11]

3 Analytiikkatyökalut

3.1 Google Analytics

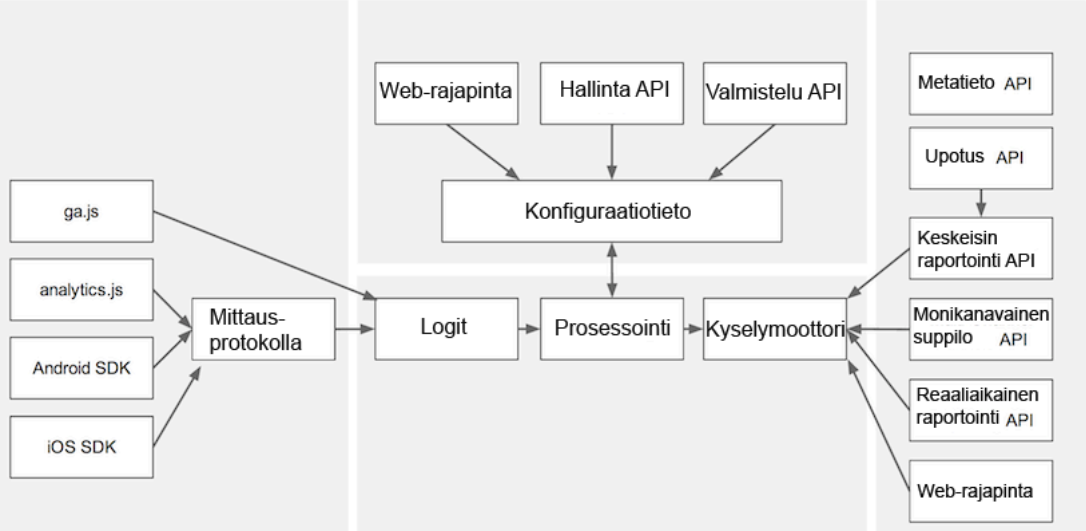
3.1.1 Yleistä

Vuonna 2005 Google osti pienen yrityksen nimeltä Urchin, joka tarjosi analytiikkatyökaluja pienille yrityksille. Urchinin myymä tuote Urchin Analytics maksoi noin 200 dollaria kuukaudessa. Ostettuaan yrityksen Google muutti Urchin Analyticsin nimen Google Analyticsiksi ja julkaisi sen ilmaiseksi ensin muutamalle suurelle verkkojulkaisulle ja sen jälkeen kaikille halukkaille. Tämän jälkeen Google Analyticsin suosio kasvoi niin voimakkaasti, että Google joutui rajoittamaan palvelun saatavuutta. [5, s. 15.] Nykyään Google Analytics on maailman suosituin analytiikkatyökalu [15]. Se on suunnattu pienille ja keskisuurille verkkosivuille ja mobiilisovelluksille, ja sen saa käyttöönsä avaamalla Google-tilin [16]. Tässä luvussa tarkastellaan Google Analyticsin yleistä toimintaperiaatetta, sekä insinöörityönä tehdyn sovelluksen kannalta olennaisia Google Analyticsin osia.

Google Analytics sisältää monia työkaluja, joiden avulla voi seurata sovelluksen hankintaa, käyttöä ja toimivuutta [16]. Vuonna 2011 Google Analyticsistä julkaistiin myös edistyneempi maksullinen versio Google Analytics Premium. Premium-versio on suunnattu suurille yrityksille, joiden sivustoilla ja sovelluksissa vierailee paljon käyttäjiä. Premium tarjoaa suuremman tiedonkäsittelykapasiteetin lisäksi muun muassa kehittyneempiä analysointityökaluja sekä koulutusta ja palveluja Googlen henkilökunnalta. Premium-versio maksaa 150 000 dollaria vuodessa. Vuonna 2012 Google laajensi Analyticsia siten, että sen voi ottaa käyttöön myös mobiilisovelluksissa. [17]

3.1.2 Toimintaperiaate

Google Analytics -järjestelmä jakautuu neljään pääkomponenttiin: tiedonkeruu, asetukset, tiedonkäsittely ja raportointi [19]. Komponentit ovat kuvattuna kuvassa 3.



Kuva 3. Google Analytics -alustan rakenne [18].

Tiedonkeruussa kerätään tietoja muun muassa siitä, millä laitteella sovellusta käytetään, mikä on laitteen käyttöjärjestelmä, miten sovelluksessa navigoidaan ja kuinka usein sovellusta käytetään. Android-sovelluksissa tietoja kerätään Google Analytics SDK:n (Software Development Kit) avulla. SDK:n keräämistä tiedoista muodostetaan paketteja, joita Google kutsuu osumiksi. Tämän jälkeen osumat lähetetään Google Analyticsille. Osumia ei kuitenkaan lähetetä heti niiden muodostuttua, vaan Google Analytics SDK varastoi osumia laitteelle sitä mukaa, kun sovellusta käytetään, ja lähettää kaikki varastoidut osumat myöhemmin yhtä aikaa. Osumia ei lähetetä reaaliajassa, koska mobiililaitteiden internetyhteys on epäluotettava. Toisin sanoen osumia voi jäädä vastaanottamatta, jos niitä yritetään lähettää ilman internetyhteyttä. Osumien lähetys reaaliajassa myös kuluttaa paljon enemmän mobiililaitteiden akkua kuin niiden varastointi ja lähetys yhdessä. Oletuksena osumat lähetetään 30 minuutin välein, mutta lähetysintervallia voi myös muuttaa. [20]

Google Analytics SDK myös erottelee sovelluksen asentaneet laitteet. Kun sovellus avataan ensi kertaa, laitteelle generoidaan oma tunniste, jonka Google Analytics tulkitsee käyttäjänä. Kun sovellus poistetaan laitteesta, myös tunniste poistuu Google Analyticsistä. [20]

Tiedonkäsittelyvaiheessa Google Analytics SDK:n keräämät tiedot muunnetaan sellaiseen muotoon, että ne voi esittää raporteissa. Tietoja käsitellessä otetaan käyttöön Google Analytics -hallintapaneelissa määritetyt asetukset. Asetuksista voi esimerkiksi lisätä suodattimia, jotka määrittelevät, millainen tieto jätetään huomioimatta tiedonkäsittelyssä. Esimerkiksi tietystä maasta tai tietystä laitteesta tulevat osumat voidaan suodattaa pois. Tiedonkäsittely osaa myös käsitellä muista Googlen palveluista, kuten AdWordsista ja AdSensestä, tullutta tietoa. Tiedonkäsittelyn jälkeen tietoa ei pysty enää muuttamaan, joten esimerkiksi suodattimilla sivuutettuja tietoja ei saa enää suodatuksen jälkeen takaisin. Tiedonkäsittelyn jälkeen tieto on nähtävissä Google Analyticsin raporteista. Google Analyticsin omien raporttien lisäksi tietoon voi päästä käsiksi Analytics Core Report -rajapinnan avulla, jolla voi luoda omia raportteja tai siirtää tietoa kolmannen osapuolen tarjoamiin raportteihin. [21]

3.1.3 Käyttöönotto

Google Analytics on yksi Google Play Servicesin sisältämistä palveluista. Google Play services APK sisältää Analyticsin lisäksi monia muita palveluita, kuten Google Mapsin ja Google+:n. Google Play Services on Android-käyttöjärjestelmässä taustalla käynnissä oleva palvelu, jonka kanssa Googlen tarjoamia palveluita käyttävät sovellukset keskustelevat. Esimerkiksi sovellus, joka haluaa lähettää tapahtuman Google Analyticsille, kutsuu Google Play Servicesin analytiikkarajapintaa, jolloin Googlen analytiikkapalvelu muodostaa tapahtumasta osuman ja lähettää sen palvelimelle sovelluksen puolesta. Google Play Services päivittyy Android-laitteissa automaattisesti Google Play -sovelluskaupan kautta, joten sovelluskehittäjän ei tarvitse huolehtia sovelluksen käyttämien Googlen palvelujen toimivuudesta eri laitteissa. [22]

Google Play Servicesin voi lisätä sovellukseen Android Studiossa joko lataamalla Google Play Services -kirjaston itse tai lisäämällä rippuvuuksia koontitiedostoihin. Google Services -lisäosan saa käyttöön lisäämällä riippuvuudet kirjastoon päätason ja sovellusmoduulin koontitiedostoihin. Kuvissa 4, 5 ja 6 on esitettynä riippuvuuden lisäys.

```
classpath 'com.google.gms:google-services:1.3.0'
```

Kuva 4. Päätason koontitiedoston Google Services -riippuvuus.

```
apply plugin: 'com.google.gms.google-services'
```

Kuva 5. Google Services -liitännäisen lisääminen sovellusmoduuliin koontitiedostossa.

```
compile 'com.google.android.gms:play-services-analytics:7.8.0'
```

Kuva 6. Google Servicesin riippuvuuden lisääminen sovellusmoduuliin koontitiedostossa.

Kuvan 4 riippuvuus määritellään sovelluksen päätason koontitiedostossa, mikä tarkoittaa, että riippuvuus vaikuttaa kaikkiin alempiin moduuleihin. Kuvat 5 ja 6 puolestaan lisäävät vielä Google Play Servicesin Android Studio -projektin sovellusmoduuliin, joka sisältää sovelluksen kehityksen kannalta olennaisimmat tiedostot, kuten Java-luokat, XML-pohjapiirrustukset ja AndroidManifest.xml-tiedoston. [23]

Kuvassa 7 on AndroidManifest.xml-tiedostossa sijaitsevien lupien määritelmät verkon ja verkon tilan käytölle. Google Play Servicesin käyttö Android-sovelluksessa vaatii lupien lisäämistä AndroidManifest.xml-tiedostoon.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Kuva 7. AndroidManifestin luvat internetin käytölle ja sen tilan kyselylle.

Google Services hakee palvelukohtaiset asetukset sovellukselle google-services.json-nimisestä tiedostosta. Tiedosto generoidaan developer.google.com-sivustolla kirjautumalla sisään Google-tunnuksilla ja lisäämällä sovellus tiliin. [23] Tämän jälkeen Google Analytics kerää automaattisesti tietoa käyttäjien määrästä, istuntojen kestoista, sovellusta käyttävistä laitteista ja niiden sijainneista. Jotta voi lähettää tietoja tapahtumista, näkymistä ja latausnopeuksista, on sovellukseen tehtävä lisää muutoksia.

Tracker-luokka on Google Play Servicesin luokka, joka koostaa ja lähettää osumat Google Analyticsille. Ennen kuin Tracker-olio voi lähettää osumia, sille on annettava Google Analyticsin sovellukselle luoma tunniste, jotta sovelluksen osumat osataan yhdistää oikeaan sovellukseen Google Analyticsin hallintapaneelissa. [24]

Kuvassa 8 on Tracker-olion alustus. Tracker-olion voi myös alustaa XML-konfiguraatiotiedoston avulla, jolloin sovelluksen tunnisteen tilalle annetaan viittaus konfiguraatiotiedostoon. Tiedostoon voi määritellä etukäteen muun muassa sovelluksen tunnisteen, automaattisen näkymien lähetyksen ja näkymien nimet. [24]

```
GoogleAnalytics analytics = GoogleAnalytics.getInstance(context);
Tracker tracker = analytics.newTracker("GA-ID");
```

Kuva 8. Tracker-olion alustus.

Mobiilisovelluksissa Google Analyticsin näkymä vastaa periaatteessa verkkosivuston yhden sivun katselukertaa. Koska Android-sovelluksissa ei kuitenkaan ole varsinaisia sivuja, jää sovelluskehittäjän vastuulle miettiä, mikä sovelluksen osa määritellään näkymäksi. Useimmiten on järkevää määritellä yksi Activity yhdeksi näkymäksi, jolloin näkymä lähetetään koko näytön siirtymän jälkeen, eli silloin, kun yksi sovelluksen Activity-komponentti vaihtuu toiseksi. Toinen vaihtoehto on määritellä yhden Activityn sisältämät Fragmentit omiksi näkymiksi, jos yhden Fragmentin näyttäminen vie koko näytön verran tilaa. Myös sovelluksen näyttämät Dialog-komponentit voi käsittää näkyminä. [25] Näkymien määrittelyn ja lähettämisen avulla selviää, mikä sovelluksen osa on suosituin käyttäjien keskuudessa ja miten käyttäjät navigoivat sovelluksessa.

```
GoogleAnalytics analytics = GoogleAnalytics.getInstance(context);
Tracker tracker = analytics.newTracker("GA-ID");
tracker.setScreenName(screenName);
tracker.send(new HitBuilders.ScreenViewBuilder().build());
```

Kuva 9. Näkymän lähetys sovelluksesta Google Analyticsille.

Kuvassa 9 lähetetään näkymä sovelluksesta. Näkymää lähetettäessä kutsutaan ensin Google Analyticsin Tracker-olion setScreenName-metodia, jolla annetaan parametriksi lähetettävän näkymän nimi. Tämän jälkeen näkymä lähetetään kutsumalla Tracker-olion send-metodia, jolle annetaan parametriksi uusi näkymäosuma. Google Analyticissa on olemassa myös automaattinen näkymänlähetys, joka lähettää näkymän aina, kun Activity-komponentti näytetään käyttäjälle. Automaattisen näkymänlähetyksen voi kytkeä päälle konfiguraatiotiedostosta. [26]

Tapahtumat ovat erikseen määriteltyjä sovelluksen sisällä tapahtuvia toimintoja, jotka eivät vaadi näkymän vaihtumista ja jotka syntyvät käyttäjän vuorovaikutuksesta sovel-

luksen kanssa. Tapahtumia lähetetään esimerkiksi silloin, kun käyttäjä valitsee videon toistettavaksi, jakaa jotain sosiaalisessa mediassa tai ostaa jotain sovelluksen kaupasta. Tämä vaatii, että sovelluksessa on erikseen määritetty, mistä toiminnoista tapahtuma lähetetään ja mitä tietoa tapahtuma sisältää. [5, s. 230]

Tapahtumat jaetaan neljään osaan, jotka määrittelevät tapahtumien rakenteen raporteissa: kategoria, toiminto, tunniste ja arvo. Kategoria ja toiminto ovat pakollisia parametreja ja tunniste ja arvo ovat vapaaehtoisia. Tapahtuman kategoria on tapahtumien ylin yhdistävä tekijä. Kategoria voi esimerkiksi olla ”Videot”. Toiminnolla tarkoitetaan kategoriaan liittyvää toimintoa, joka tapahtumassa suoritetaan. Se voisi olla esimerkiksi ”Toista” tai ”Pysäytä” tai jokin muu videon katseluun liittyvä toiminto. Tunniste antaa lisätietoa tapahtumasta kertomalla esimerkiksi toistetun videon nimen. Arvolle voi muiden tapahtuman osien sijaan antaa pelkän numeroarvon. Arvo voisi esimerkiksi olla videon lataukseen käytetty aika. Näin Tapahtuman rakenne voisi olla siis ”Videot”, ”Toista”, ”Yrityksen promootiovideo”, ”60”. On erityisen tärkeää suunnitella tapahtumien rakenteet etukäteen ennen tapahtumien lähettämistä ja nimetä osiot selkeästi, sillä mitään tietoa ei voi poistaa Google Analyticsin raporteista. Hyvin suunnitellut tapahtumat myös selkeyttävät raporttien tulkitsemista. [27]

Kuvassa 10 tapahtuma lähetetään määrittelemällä Tracker-oliolle näkymä ja sen jälkeen tapahtumaosuma HitBuilders-luokan EventBuilderin avulla, joka annetaan Tracker-olion send-metodille parametriksi.

```
Tracker tracker = analytics.newTracker("GA-ID");
tracker.setScreenName(screenName);
tracker.send(new HitBuilders.EventBuilder()
        .setCategory(action)
        .setAction(category)
        .setLabel(label)
        .build());
```

Kuva 10. Tapahtuman lähetys sovelluksesta Google Analyticsille.

Latausaikojen avulla voidaan mitata sovelluksen eri toimintoihin käytettyä aikaa. Myös latausajat jaetaan neljään osaan, joiden perusteella luodaan sovelluksen nopeus – raportin rakenne. Osat ovat kategoria, arvo, nimi ja tunniste, joista kategoria ja arvo ovat pakollisia osia. Mitatut ajat jaetaan raporteissa erikseen kategorian mukaan. Arvo määrittelee latausajan sekunteina, nimi kertoo tarkempaa tietoa siitä, mitä on ladattu ja

tunniste sisältää lisätietoa latauksesta. Kategoria voisi olla vaikkapa ”JSON-lataukset”, arvo ”10”, nimi ”kommenttien lataus” ja tunniste ladattujen kommenttien määrä. [28]

```
tracker.send(new HitBuilders.TimingBuilder()
        .setCategory(category)
        .setVariable(name)
        .setLabel(label)
        .setValue(time)
        .build());
```

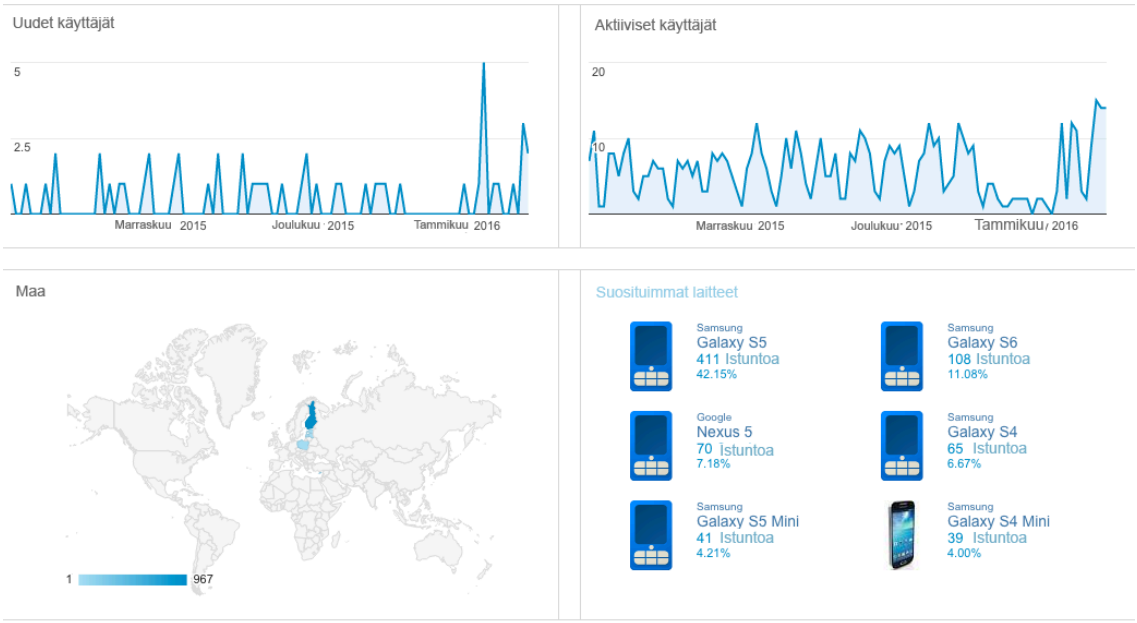
Kuva 11. Latausajan lähetys sovelluksesta Google Analyticsille.

Kuvassa 11 latausaika lähetetään kutsumalla Tracker-olion send-metodia, jolle annetaan parametrina aikaisemmin mainitut osat.

3.1.4 Raportit

Google Analyticsin keräämä tieto sovelluksesta esitetään raporteissa, joita tutkimalla voi muun muassa saada selville käyttäjien määrän, mistä käyttäjät ovat ladanneet sovelluksen, miten he navigoivat sovelluksessa ja minkä näkymän kohdalla he sulkevat sovelluksen. Raporttien avulla siis analysoidaan käyttäjien sovelluksenkäyttöä, minkä pohjalta tehdään muutoksia sovellukseen. [5, s. 243] Raportit on jaettu niiden esittämän tiedon perusteella eri ryhmiin: sovelluksen yleiskatsaus, reaaliaikainen, yleisö, hankinta, käyttäytyminen ja konversiot. [29]

Sovelluksen yleiskatsaus -raportti on koottu kaikkein tärkeimmistä Google Analyticsin raporttien tiedoista. Raportista voi nähdä sovelluksen yleisen tilanteen ja tarkastella muutoksia keskeisimmillä raportointialueilla. Kuvassa 12 on yleiskatsauksen esittämiä tietoja sovelluksen tilasta.



Kuva 12. Sovelluksen yleiskatsaus -raportin osa [29].

Yleiskatsausta voi muokata muuttamalla tarkastelun aikaväliä ja lisäämällä segmenttejä. Yleiskatsaukseen tehdyt muutokset vaikuttavat myös muihin raportteihin, kunnes muutokset poistetaan tai Google Analyticsistä kirjaudutaan ulos. Raportin voi myös jakaa sähköpostina. Sovelluksen yleiskatsaus on oletusraportti Google Analyticsin ohjauspaneelissa. [30]

Reaaliaikainen raportti antaa reaaliaikaista tietoa sovelluksen käytöstä. Raporteista näkee, kuinka monta aktiivista käyttäjää sovelluksessa on tällä hetkellä, heidän sijaintinsa, katsomansa näkymät, aiheuttamat tapahtumat ja konversiot. [29]

Yleisöraportissa on tietoja sovelluksen käyttäjistä, kuten heidän sijaintinsa, käyttämänsä laite, kuinka usein he käyttävät sovellusta ja kuinka pitkään he viihtyvät sovelluksessa. Yleisöraportin yleiskatsauksesta selviää, kuinka kauan keskimääräinen istunto sovelluksessa kestää, näkymien katselukerrat, katselukertojen määrä istunnossa, uusien käyttäjien lukumäärä ja suosituimmat kielet ja sijainnit. [31] Hallintapaneelissa yleiskatsauksen alla on tarkempia tietoja sovelluksen käyttäjistä ja käytöstä: sovellusversiot, yleisötiedot, aihepiirit, geo, käyttäytyminen ja laitteet ja verkko. [29]

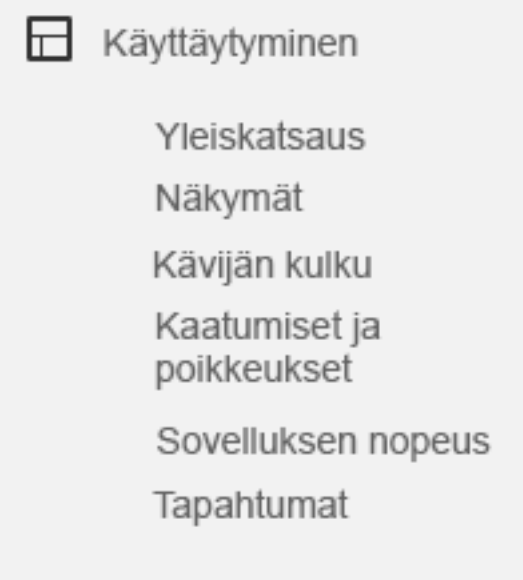
Sovellusversioista näkee sovelluksen kaikki eri versiot, joita käyttäjät käyttävät. Versioiden erottelun avulla voi nähdä, kuinka suuri osa käyttäjistä on päivittänyt sovelluksen

uuteen version ja kuinka moni käyttää vielä vanhaa versiota. Yleisötiedot ja aihepiirit liittyvät mainontaan eri kohderyhmille, kuten tietyn ikäisille, tietylle sukupuolelle ja tietyistä asioista kiinnostuneille. Geosta näkee, missä sovellusta käytetään ja kuinka paljon. Jos sovellus esimerkiksi menestyy jossain ennalta arvaamattomassa paikassa, saattaa se vaikuttaa sovelluksen jatkokehitykseen. [31] Käyttäytymisen alla on uusien ja palaavien käyttäjien määrä ja istuntoihin liittyvää tietoa, kuten istuntojen kesto ja niiden välissä kulunut aika. Laitteista ja verkosta selviää, millä laitteilla ja millä käyttöjärjestelmällä sovellusta käytetään ja minkä operaattorin asiakkaita käyttäjät ovat. Myös käyttäjien laitteiden resoluutiot ja istuntojen määrä resoluutiota kohti ovat näkyvissä. [29]

Hankintaraporteista näkee, miten käyttäjät ovat löytäneet sovelluksen, uusien käyttäjien kokonaismäärän ja heidän käyttämänsä laitteet ja käyttöjärjestelmät. Raporteista näkee myös sovelluksen latausten ja asennusten lukumäärän, joiden avulla voi päätellä eri mainoskampanjoiden vaikutukset sovelluksen latausmääriin. Hankintaraportin eri osiot ovat uudet käyttäjät, sovellusmarkkinapaikka, liikenteen lähteet, Google Playsta tulevat viittaukset ja Adwords. [29]

Uudet käyttäjät -osio antaa tietoa istunnoista, joissa sovellus avataan ensimmäisen kerran. Raportti kertoo näiden istuntojen määrän, käyttäjien käyttöjärjestelmän, sovelluksen version ja käyttäjien sijainnin. [32] Sovellusmarkkinapaikasta selviää, mistä käyttäjät ovat ladanneet sovelluksen. Tämän avulla voi arvioida mainoskampanjoiden toimivuutta eri paikoissa. Sovellusmarkkinapaikat tukevat Google Playta ja kolmannen osapuolen markkinapaikkoja. Liikenteen lähteet puolestaan kertoo, miten hyvin sovellukset pärjäävät eri markkinapaikoissa. Raportin avulla selviää, miten potentiaaliset käyttäjät ajautuvat sovelluksen sivulle ja kuinka moni heistä latasi sovelluksen. Google Playsta tulevista viittauksista käy ilmi, miten käyttäjät siirtyvät eteenpäin hankintaprosessissa, eli sovelluksen löytämisestä aina sen ensikäynnistämiseen asti. Liikenteen lähteen täydellinen hyödyntäminen ja Google Playsta tulevien viittausten käyttöönotto vaatii Google Analyticsin ja Google Playn kehittäjäkonsolin linkittämistä. AdWords-raportit antavat tietoa käyttäjistä, jotka klikkasivat sovelluksen AdWord-mainosta ja asensivat sovelluksen. Raporttien avulla voi selvittää, mistä käyttäjät ovat löytäneet mainoksia ja millaisia käyttäjiä mainokset johdattavat Google Play Storeen ja miten sovelluksen näyttö- ja hakukampajat onnistuvat. [32]

Käyttäytyminen-raportti kertoo, miten käyttäjät käyttävät sovellusta. Raportista ilmenevät muun muassa kaikki istunnossa katsotut näkymät, niiden katselujärjestys, sovelluksen kaatumiset ja latausajat. Käyttäytyminen-raportin yleiskatsaus on yhteenveto raportin muista osista. [33] Kuvassa 13 on raportin rakenne: yleiskatsauksen alla hallintapaneelissa ovat näkymät, kävijän kulku, kaatumiset ja poikkeukset, sovelluksen nopeus ja tapahtumat. [29]



Kuva 13. Käyttäytyminen-raportin rakenne [29].

Näkymät sisältää tietoa sovelluksen näkymistä; kuinka monta kertaa tiettyä näkymää on katseltu, näkymässä vietetty keskivertoaika, näkymän poistumisprosentti ja näkymien yksittäiset katselut. Yksittäisillä katseluilla tarkoitetaan istuntoa, jonka aikana näkymää katsottiin vähintään kerran. [33]

Kävijän kulku esittää graafisesti käyttäjien kulkeman polun näkymien ja tapahtumien välillä. Suosituimmat näkymät ja siirtymät näkymästä toiseen on kuvattu suurempina kuin harvemmin katsellut näkymät tai tehdyt siirtymät. Raportista selviää myös, minkä näkymän kohdalla käyttäjät ovat sulkeneet sovelluksen. [29]

Kaatumiset ja poikkeukset sisältää sovelluksessa tapahtuneet virheet ja niiden kuvaukset. Raportissa muut poikkeukset erotellaan kaatumisista. Jotta Google Analytics saa tiedot poikkeuksista, tulee ne määrittää sovelluksessa erikseen. [33]

Sovelluksen nopeus -raportista näkee ennalta määriteltyjen tapahtumien latausajat. Latausaikoja voi tarkastella kolmella eri välilehdellä: tutki, jakelu ja kartan peittokuva. Tutki-välilehti näyttää määritetyn latausajan nimen ja sen lataamiseen käytetyn keskimääräisen ajan. Jakelu-välilehti näyttää kaikkien latausaikojen keston yhteensä ja latausaikojen prosentuaaliset osuudet kaikista latauksista. Kartan peittokuva -välilehti näyttää latausajat kartalla. Mitä pidempi latausaika, sitä tummemmin kartan alue on väritetty. [33]

Tapahtumat-raportti sisältää kaikki sovelluksen tapahtumat. Tapahtumat on esitelty raportissa sovelluksessa määriteltyjen osioiden perusteella. [29]

Konversioraportit seuraavat sovelluksen tuloja ja tehokkuutta. Raporteista ilmenee, miten Google Analyticsissä erikseen määriteltyjä tavoitteita saavutetaan. Tavoitteita voivat olla esimerkiksi istunnon minimikesto, näkymien määrä istunnossa tai tapahtumien määrä. Raporteista käyvät ilmi näiden tavoitteiden toteutumisprosentit. Myös sovelluksessa tehtyjä ostoksia on mahdollista seurata. Konversioraportit jakautuukin kahteen osaan: tavoitteet ja verkkokauppa. [33]

Tavoitteet-raportti näyttää tiedot kaikista tavoitteista. Yleiskatsaus kokoaa yhteen tiedot tavoitteista ja niiden toteutumisesta. Tavoitenäkymät kertovat, miten tavoitteet ovat toteutuneet eri näkymissä. Tavoitereitti kertoo visuaalisesti, miten käyttäjät ovat kulkeneet tavoitteisiin ja kuinka iso osa käyttäjistä päätyi tavoitteiseen. Verkkokauppa-raportin yleiskatsauksessa on yhteenveto sovelluksen tapahtumista ja tuotosta. Tuotteen kehitys sisältää Google Analyticsin avulla kerättyjä tietoja tuotteista, kuten ostoista tulleet tulot ja ostojen määrä. Myynnin tehokkuus seuraa tuloja ja ostojen ajankohtia. Tapahtumat antavat tietoa tuotteiden toimituksista ja muista lisämaksuista, kuten veroista. [33]

3.1.5 Segmentit ja omat raportit

Raporttien näyttämää tietoa on mahdollista muokata segmenteillä. Segmentit ovat istunto- tai käyttäjätiedon alijoukkoja. Esimerkiksi kaikista istunnoista voidaan segmentoida tietyt istunnot niiden alkamisajankohdan perusteella. Käyttäjiä voisi segmentoida maan tai kaupungin perusteella, tai sen perusteella, ovatko he tehneet ostoksia sovelluksessa. Segmenttien avulla voidaan siis erotella ja tutkia Google Analyticsin tiedon

alijoukkoja, jotka eivät näy valmiissa raporteissa, mikä auttaa sovelluksen käytön trendien löytämisessä. [34]

Segmenttejä voi vertailla raporteissa yhtäaikaisesti. Google Analyticsissä on ennalta määritettyjä segmenttejä, joita voi muokata tai käyttää sellaisenaan. Segmenttejä voi luoda myös kokonaan itse ja tuoda Analyticsin Ratkaisugalleriasta, joka on Analyticsin käyttäjien segmenttien ja ratkaisujen jakamiseen tarkoitettu kauppapaikka. [34]

Google Analyticsin valmiit raportit ja niihin lisätyt omat segmentit eivät välttämättä anna juuri haluttua tietoa sovelluksesta, joten myös omien raporttien luonti on mahdollista. Omiin raportteihin voi itse valita haluamansa tietotyypit ja ulottuvuudet. Tietotyypeillä tarkoitetaan mittoja, joilla on määrällinen arvo, kuten istuntojen määrä. Ulottuvuuksilla tarkoitetaan erilaisia piirteitä, kuten kaupungin nimeä tai laitetta. Omissa raporteissa myös tiedon esitystavan voi itse määrittää. [35]

3.2 Fabric-analytiikkatyökalu

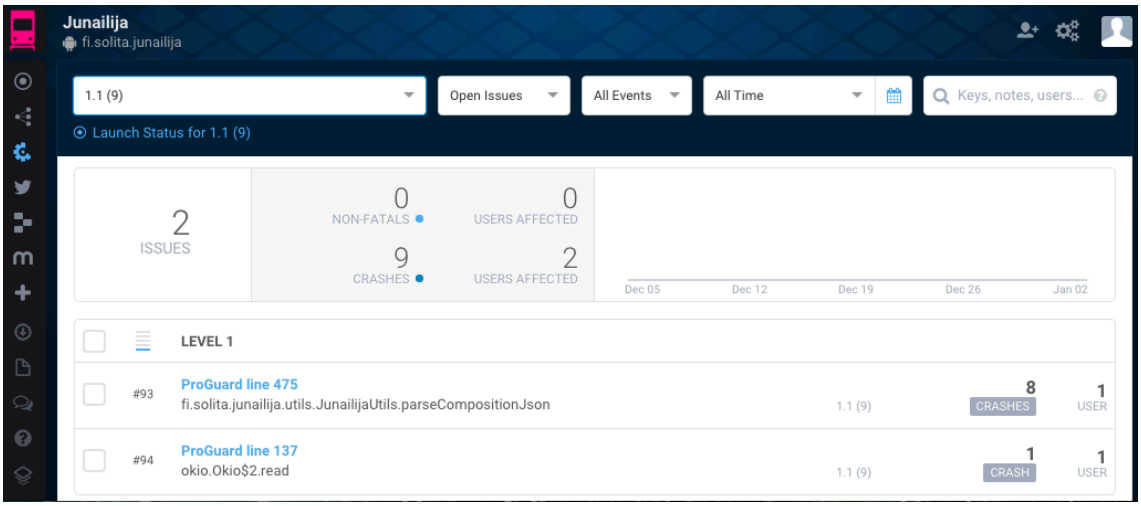
3.2.1 Yleistä

Fabric on Crashlyticsin, Twitterin ja MoPubin kehittämä ja Twitterin lokakuussa 2014 julkaisema moduuleista koostuva mobiilialusta, jonka tarkoituksena on helpottaa sovellusten kehittämistä. Fabric on integroitu muun muassa Android Studion ja Eclipsen kanssa, mikä helpottaa moduulien pitämistä ajan tasalla, niiden integrointia ja sovellusten testiversioiden jakelua. [36] Julkaisun aikaan Fabric sisälsi kolme eri moduulia: Crashlytics, MoPub ja Twitter. Moduulit yrittävät auttaa mobiilisovelluskehityksen yleisimpien ongelmien ratkaisussa. Näitä ongelmia ovat erimerkiksi sovelluksen jakelu testauskäyttöön, kaatumisten raportointi ja sovelluksen tuottamien tulojen seuranta. Nykyään Fabric pyrkii ratkaisemaan nämä ja monta muutakin ongelmaa mahdollisimman pienellä vaivalla. Fabricin tavoitteena on, että sen käyttöönottoon ei kuluisi muutamia minuutteja enempää aikaa ja että suurimman osan sen moduuleista voisi ottaa käyttöön lisäämällä sovellukseen vain muutaman rivin koodia. Moduuleista voi valita käyttöönsä vain ne, jotka itse haluaa. [36] Fabric on ilmainen palvelu. Twitter saa tulonsa Fabricin avulla asennettavista mainoksista, joiden tuottamista tuloista osa menee Twitterille [37]. Tässä luvussa tarkastellaan insinöörityössä käytettyjä Fabricin moduuleja.

3.2.2 Crashlytics-moduuli

Crashlytics on yksi kolmesta moduulista, jotka Fabric on sisältänyt sen julkaisusta asti. Crashlyticsin kehitys alkoi vuonna 2011 itsenäisenä palveluna, mutta vuonna 2013 Twitter osti Crashlyticsin, minkä jälkeen sen kehitys jatkui osana isompaa kokonaisuutta, eli Fabricia. [38]

Crashlyticsin tarkoituksena on havaita sovelluksessa tapahtuvat kaatumiset ja raportoida niistä reaaliajassa. Poikkeukset kootaan raportteihin, joihin pääsee käsiksi Fabricin verkkosivuilla olevasta kehittäjille tarkoitetusta hallintapaneelista. Kuvassa 12 on sovelluksen Fabric-tilin Crashlytics-raportti, josta näkee sovelluksessa tapahtuneet poikkeukset. Crashlytics näyttää, millä rivillä poikkeus on tapahtunut ohjelmassa, milloin poikkeus on tapahtunut, kuinka monta kertaa se on tapahtunut ja kuinka moneen käyttäjään se on vaikuttanut. Korjattuaan poikkeuksen aiheuttaneen koodin, voi poikkeuksen merkitä suljetuksi, jolloin se poistuu kuvan 12 oletusnäkymästä. Näytettäviä poikkeuksia voi suodattaa sovelluksen version, avoimuuden, poikkeuksen tyypin ja ajan mukaan. Poikkeuksen yksityiskohtaisista tiedoista selviää, millä laitteilla poikkeus on tapahtunut ja tarkempi kuvaus poikkeuksesta. [39]

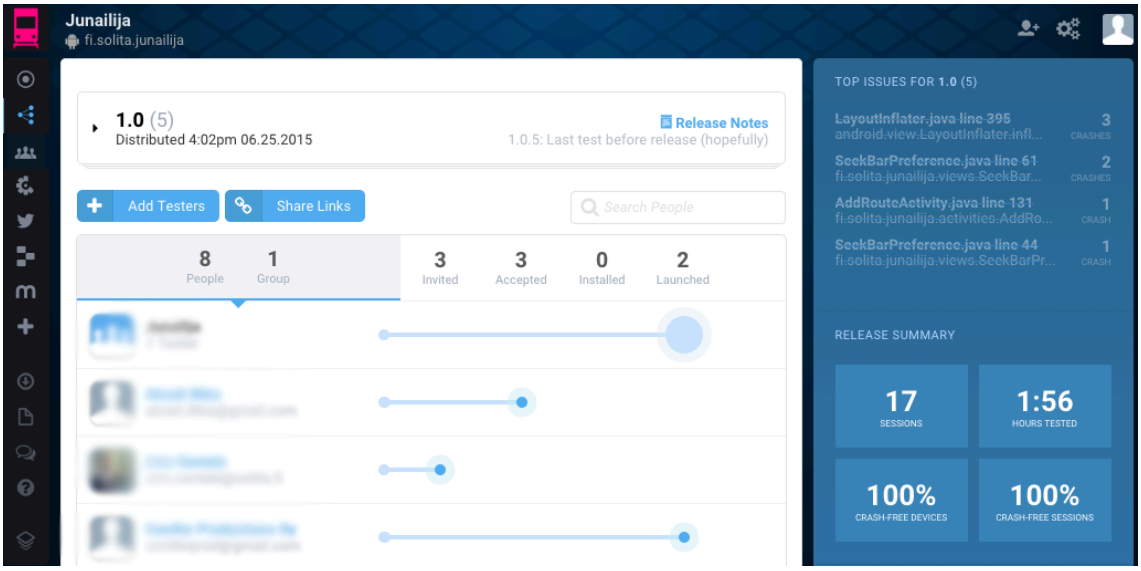


Kuva 12. Junailija-sovelluksen Fabric-tilin Crashlytics-hallintapaneeli [39].

Crashlytics tukee Android SDK:n lisäksi myös Android NDK:ta, eli poikkeusten raportointi toimii myös sovelluksissa, jotka on ohjelmoitu C:llä ja C++:lla [36].

3.2.3 Beta-moduuli

Fabricin Beta-moduuli on Crashlyticsin kehittäjien tekemä sovelluksen testausjakeluun tarkoitettu palvelu. Sen tarkoituksena on yksinkertaistaa sovelluksen jakelu testikäyttäjille ja helpottaa testikäyttäjien hallitsemista. Sovelluksen jako testaajille onkin yksinkertaista: valmis APK-tiedosto raahataan Android Studion liitännäiseen, minkä jälkeen julkaisulle voi lisätä kommentteja. Tämän jälkeen sovellus on valmis testijakoon. Testaajia lisätään syöttämällä testaajan sähköpostiosoite Fabricille hallintapaneelin kautta tai Android Studion liitännäisessä. Tämän jälkeen testaaja saa sähköpostiinsa kutsun, jonka hyväksyttyään hänen tulee ladata Crashlyticsin Beta-sovellus, johon käyttäjä kirjautuu samalla sähköpostiosoitteella. Sovelluksesta käyttäjä voi ladata kaikki sovellukset, joihin hänet on testaajaksi kutsuttu. Beta-sovellukseen tulee myös sovellusten päivitykset, joista sovellus ilmoittaa käyttäjälle. [40]



Kuva 13. Junailija-sovelluksen Fabric-tilin Beta-näkymä [39].

Kuvassa 13 on Fabricin Beta-näkymä, josta näkee testaajien tilan. Näkymästä ilmenee, kuka on kutsuttu testaajaksi sovellukseen ja missä vaiheessa testausta hän on. Testauksen vaiheet on jaettu neljään osaan: kutsuttu, hyväksynyt, asentanut ja käynnistänyt. Näkymä siis kertoo, onko käyttäjä vielä hyväksynyt kutsua, asentanut sovellusta ja käynnistänyt sitä. Testijulkaisusta näkyy myös yhteenveto, joka kertoo testiversion kaatumisista ja testausajoista sekä suurimmista kaatumisten aiheuttajista. Testiversioita on myös mahdollista selata versionumeron perusteella. Testaajista voi luoda erilaisia ryh-

miä, jolloin testiversioita voi julkaista vain tietyille henkilöille. Testaajia voi myös poistaa. [39]

3.2.4 Answers-moduuli

Crashlyticsin kehittäjien Answers antaa lähes reaaliaikaista analytiikkatietoa sovelluksesta. Answers luotiin antamaan hyödyllistä tietoa sovelluksen suorituskyvystä ja käytöstä välittömästi sen sijaan, että kehittäjät joutuisivat itse suunnittelemaan ja luomaan omia raportteja, joista selviäisi haluttu tieto. [37]

Answers on mahdollista ottaa käyttöön, jos sovelluksessa on jo valmiiksi Crashlytics, mutta se on mahdollista lisätä sovellukseen myös itsenäisenä osana. Answers kerää automaattisesti tietoa sovelluksen käytöstä, mutta jotta saa yksityiskohtaisempaa tietoa ja paremman ymmärryksen sovelluksen käytöstä, on sovellukseen lisättävä tapahtumia. Tapahtumien avulla voi mitata tiettyjä haluttuja toimintoja sovelluksessa, kuten ostotapahtumia, sisäänkirjautumisia ja hakuja. Tapahtumia mittaamalla voidaan tarkemmin seurata käyttäjien käyttäytymistä sovelluksessa. Tapahtumat lähetetään puhelimesta mahdollisimman nopeasti, mutta samalla laitteen akun varausta säästäen. Tapahtumat lähetetään erissä, kun sovellus käynnistetään tai kun se joutuu taustalle. [41]

Answers seuraa automaattisesti sovelluksen suorituskyvyn kannalta olennaisimpia mittareita, joista muutama näkyy kuvassa 14. Mittarit jaetaan päätason mittareihin ja lisämittareihin. Päätason mittareita ovat aktiiviset käyttäjät, päivittäiset aktiiviset käyttäjät, kuukausittaiset aktiiviset käyttäjät, päivittäiset uudet käyttäjät, kaatumisista vapaat käyttäjät ja istunnot. Aktiivisilla käyttäjillä tarkoitetaan käyttäjiä, jotka ovat aloittaneet session viiden minuutin sisällä. Lisämittareita ovat suosituimmat versiot, päivittäiset käyttäjät version mukaan, päivittäiset aktiiviset käyttäjät laitteen mukaan, päivittäiset aktiiviset käyttäjät käyttöjärjestelmän mukaan, aktiiviset käyttäjät asennuspäivän mukaan, päivittäisten aktiivisten käyttäjien suhde kuukausittaisiin aktiivisiin käyttäjiin, käyttäjien säilyminen, kaatumisista vapaat istunnot, istunnot käyttäjää kohti, käytetty aika sovelluksessa käyttäjää kohti ja istunnon keston mediaani. [42] Kun mittari valitaan kuvan 14 näkymässä näkyviin tulee erilaisia diagrammeja, jotka kuvaavat mittarin arvon kehitystä ajan funktiona. Tämän lisäksi esiin tulee myös muita valittuun mittariin liittyviä mittareita. [39]

Kuva 14. Junailija-sovelluksen Fabric-tilin Answers-näkymä [39].

Mittareiden ja tapahtumien lisäksi Answers yhdessä Twitterin mainosten kanssa mahdollistaa mainoskampanjoiden tulosten seuraamisen. Answers siis seuraa, kuinka moni klikkaa sovelluksen mainosta ja päätyy sitä kautta asentamaan sovelluksen. Answers kertoo myös, missä klikattu mainos on sijainnut. Tämän lisäksi se osaa kertoa väestötieteellistä tietoa sovelluksen käyttäjistä ja heidän kiinnostuksenkohteistaan. Tämä onnistuu hyödyntämällä Twitterin keräämää tietoa Twitterin käyttäjistä. [41]

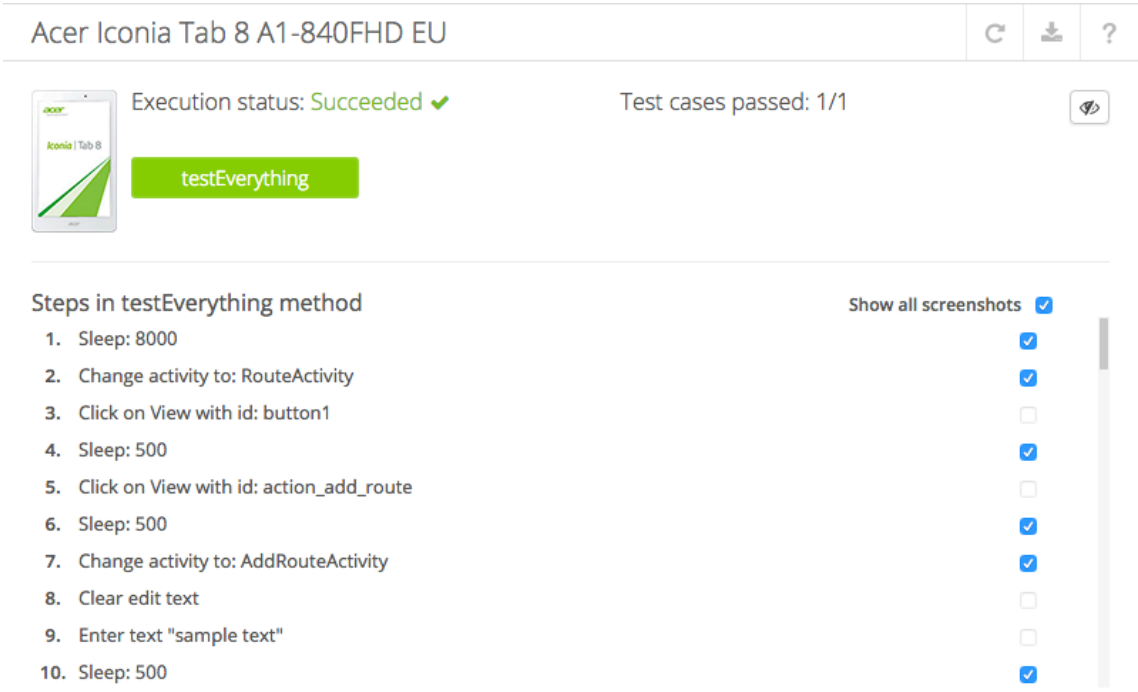
3.3 Testdroid Cloud -palvelu

Testdroid Cloud on Bitbarin kehittämä internetissä sijaitseva palvelu, jonka avulla sovelluskehittäjät voivat testata sovelluksiaan monella erilaisella oikealla laitteella ostamatta laitteita itse. Palvelu testaa sovelluksen asennusta, käynnistystä ja käyttöä simuloiden ihmismäistä sovellusten käyttöä. Testdroid Cloudissa testaaminen jakaantuu kehittäjän näkökulmasta kolmeen osaan: testien määrittely, testien suoritus ja testituloksien arviointi. [43]

Testien määrittelyä varten Bitbar on kehittänyt ohjelman, joka tallentaa testejä Android-sovelluksille. Ohjelma seuraa, mitä käyttöliittymän komponentteja testin laatija painaa sovelluksessa, ja tallentaa painetut kohdat. Painalluksia tallennetaan painetun kohdan x- ja y-koordinaattien perusteella tai painetun komponentin tekstin tai tunnisteen perusteella. Kun painallukset tallennetaan peräkkäin, syntyy painalluksien ketju, joka toistetaan palvelussa, kun testi suoritetaan. Painalluksien lisäksi etukäteen voidaan määritellä hetkiä ajassa, jolloin palvelu ottaa ohjelman näytöstä kuvakaappauksen. [43] Testin voi jättää myös määrittelemättä, jolloin testaus tapahtuu robotin avulla. Robotti testaa sovellusta painelemalla eri kohtia sovelluksesta ja ottamalla siitä kuvakaappauksia. Robotti osaa myös kirjautua sisään sovellukseen, jos sille annetaan käyttäjätunnus ja salasana. [44] Ongelma robotin avulla testaamisessa on, ettei se välttämättä osaa navigoida sovelluksen kaikkiin osiin, jolloin sovelluksen osia jää testaamatta. Tämän takia ennalta määritetty testi on todennäköisesti kattavampi kuin automaattinen.

Testitavan valinnan jälkeen määritellään laitteet, joilla sovellusta testataan. Testdroid Cloudin ilmaisessa versiossa testilaitteita on rajallinen määrä, mutta maksullisessa versiossa laitteita on yli 100 eri laitevalmistajilta. Laitteita voi valikoida niiden Android-version, näyttöjen koon tai sisäisten komponenttien perusteella. [43]

Testiä suoritettaessa palvelu tarkkailee monia eri asioita sovelluksessa ja laitteessa. Ensimmäisenä testituloksista selviää, menivätkö laitteen testit läpi. Jos testit eivät menneet läpi, ongelmakohdat selviävät laitteen lokitiedostosta. Tuloksissa esitetään kaikki sovellukselle tehdyt painallukset ja painalluksien välillä pidetyt tauot. Kuvassa 15 on testissä yhdellä laitteella suoritetut painallukset sekä ilmoitus, menivätkö testit läpi. Tuloksissa olevat kuvakaappaukset ovat numeroitu ja esitetty niiden ottojärjestyksessä. Kuvakaappausten avulla voi selvittää, missä kaikkialla robotti on käynyt sovelluksessa ja miltä sovellus näyttää eri laitteilla. Palvelu mittaa myös sovelluksen muistin ja suorittimen käyttöä, minkä se esittää diagrammina ajan funktiona. [45]



Kuva 15. Junailija-sovelluksen Testdroid Cloud -testitulosten painallukset [45].

Testitulosten valmistuttua on palvelusta mahdollista ladata testien perusteella automaattisesti generoitu raportti. Raportista ilmenee yleisiä tietoja testistä, kuten testauksen ajankohta ja testaustapa, sekä kaikki laitteet, joilla sovellusta testattiin ja se, menivätkö testit läpi laitteilla. [45]

3.4 Työkalujen vertailua

Google Analyticsin mobiilianalytiikan ja Fabricin tarjoamissa palveluissa on joitakin päällekkäisyyksiä. Esimerkiksi eri mittareiden seuraaminen, tapahtumien lähettäminen ja kaatumisten tarkasteleminen on mahdollista toteuttaa molemmilla työkaluilla. Näiden ominaisuuksien käyttöönotossa ja käytössä on kuitenkin suuria eroja työkalujen välillä. Tässä luvussa vertaillaan insinöörityön kannalta olennaisia päällekkäisyyksiä työkalujen välillä.

Molempien työkalujen SDK mittaa tärkeimpiä mittareita automaattisesti ilman lisäyksiä sovellukseen. Nämä mittarit liittyvät käyttäjien laitteisiin, sovelluksen käyttäjämääriin ja istuntoihin. Google Analyticsissä on automaattisesti mitattujen mittareiden lisäksi jonkin verran mittareita, joita voi lisätä sovellukseen. Tämän lisäksi on myös mahdollista luoda

omia mittareita, jotka lähetetään laitteesta näkymien mukana. Fabricin Answers-moduulissa puolestaan mittareita on pienempi määrä, ja ne kaikki otetaan käyttöön automaattisesti. [42]

Varmasti osittain tämän vuoksi Fabric esittää kerätyt mittarit huomattavasti selkeämmin hallintapaneelissaan. Fabricin Answers on myös suunniteltu tarkoituksellisesti niin, että oleellisin tieto sovelluksesta on nähtävissä heti sen etusivulta, ilman erillisiä raportteja. [37] Tämän ansiosta Fabricin mittareiden arvojen seuraaminen on yksinkertaista ja vaivatonta. Google Analyticsin raportit sen sijaan ovat hieman sekavampia. Google Analyticsin mittaama tieto on jaettu seitsemään eri raporttiin, jotka puolestaan on jaettu alakategorioihin. Alakategorioita selaamalla pääsee käsiksi yleiskatsauksiin ja mittareihin ja niiden perusteella luotuihin diagrammeihin, joita voi muokata esimerkiksi muuttamalla tarkasteltavaa ajanjaksoa. Tämän lisäksi valmiita raportteja on mahdollista muokata segmenttien avulla. Myös omien raporttien luonti on mahdollista, mikä tekee Google Analyticsin raporteista kattavampia kuin Fabricin. [30]

Google Analyticsissa tapahtumia on mahdollista lähettää Google Analytics SDK:n Tracker-olion avulla. Tapahtumille on mahdollista asettaa neljä eri parametria, jotka määrittelevät tapahtuman sisällön hierarkian. [27] Fabricin Answers-moduulissa puolestaan tapahtumien lähetys on monipuolisempaa. Answersissa on 12 ennalta määritettyä tapahtumaa, joiden parametrit vastaavat tapahtuman tarkoitusta. Esimerkiksi ostotapahtuman parametrit ovat tuotteen hinta, valuutta, tuotteen nimi, tyyppi, tunniste ja se, onnistuiko osto vai ei. Tämän lisäksi on mahdollista luoda omia tapahtumia, joille voi itse määritellä parametrejä. [41] Google Analyticsissa tapahtumat ovat käyttäytyminen-raportin alla tapahtumat-alakategoriassa, jossa tapahtumia ja niiden parametreja voi selata. Tapahtumat on esitetty kattavasti ja tapahtuman sisältöä on helppo vertailla, sillä sisällön voi järjestellä muun muassa määrän ja yksittäisten tapahtumien määrän mukaan. Fabricissa puolestaan tapahtumat näkyvät mittareiden kanssa samalla sivulla. Yksityiskohtaista tietoa tapahtumasta saa valitsemalla tapahtuman. Yksityiskohtainen tieto sisältää annettujen parametrien lisäksi tapahtumien lukumäärän, lukumäärän käyttäjää kohti ja käyttäjien määrän, jotka ovat lähettäneet kyseisen tapahtuman. [39]

Poikkeusten ja kaatumisten seuraaminen on erittäin yksinkertaista Fabricin Crashlytics-moduulilla. Sovellukseen ei tarvitse lisätä kuin kolme riviä koodia Fabricin liitännäisen havaittua sovelluksen, minkä jälkeen Crashlytics huomaa ja lähettää kaikki sovelluksessa tapahtuvat käsittelemättömät poikkeukset Fabricille. Crashlyticsin raporttiin on

koottu kaikki poikkeukset yhdelle sivulle, josta valitsemalla voi nähdä tietyn poikkeuksen pidemmän kuvauksen. Google Analyticsissa poikkeusten seuraaminen vaatii vain pienen määrittelyn. Määrittelyn avulla poikkeuksista saa näkyviin kuitenkin vain poikkeuksen viestin, eikä edes riviä, jolla virhe on tapahtunut. Jotta koodirivin ja lisätiedot poikkeuksesta saa näkyviin, on sovellukseen lisättävä huomattavasti lisää koodia, mikä tekee käyttöönotosta paljon työläämpää kuin Crashlyticsissa. [46]

Kaiken kaikkiaan tapahtumia lukuun ottamatta Google Analyticsilla voi tehdä enemmän kuin Fabricilla: raportteja ja poikkeuksien viestejä voi muokata ja mittareita ja raportteja voi luoda itse. Tämä vaatii kuitenkin paljon lisää työtä. Fabricin vahvuus piileekin sen äärimmäisen helpossa käyttöönotossa ja hallintapaneelin yksinkertaisuudessa. Fabricin avulla on helppo asentaa analytiikkaa sovellukseen ja seurata tärkeimpiä mittareita nopeasti. Google Analytics on huomattavasti hitaampaa ja hankalampaa ottaa käyttöön, mikä johtunee siitä, että alun perin Google Analytics on suunnattu verkkosivuille. Google Analyticsia ei siis ole suunniteltu mobiilisovellukset ensimmäisenä mielessä, mikä näkyy kehittäjän näkökulmasta kankeana käyttöönottona.

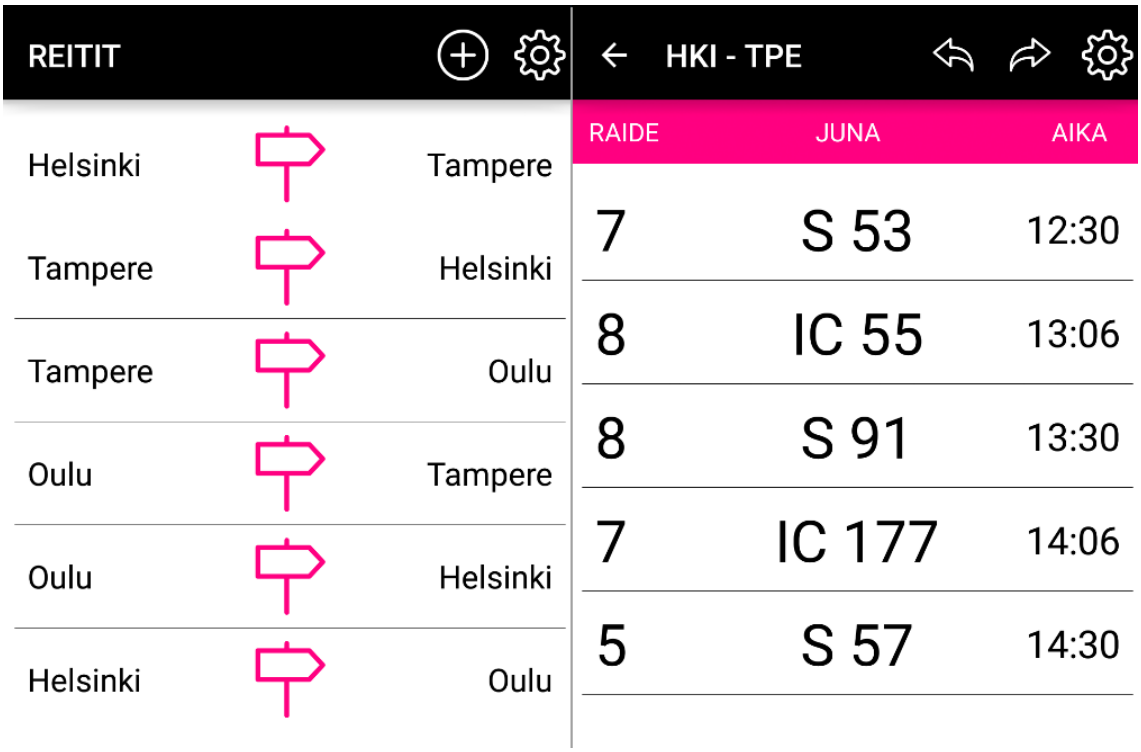
4 Junailija-sovelluksen kehitys

4.1 Sovellus

4.1.1 Yleistä

Insinöörityössä tehty Junailija on Android-käyttöjärjestelmälle kehitetty sovellus, joka on tarkoitettu Suomen sisäistä junaliikennettä usein käyttäville henkilöille. Sen tarkoituksena on näyttää ajankohtaista tietoa käyttäjän valitsemalla reitillä kulkevista junista. Sovelluksessa voi tallentaa muistiin useita reittejä, joista käyttäjä on kiinnostunut. Reittejä lisätessä asemat voi valita kirjoittamalla tai painamalla hae lähin asema -nappia, joka hakee laitteen sijaintia lähinnä olevan juna-aseman GPS- tai verkkopaikkatietojen avulla. Kun reitti valitaan reittinäkymässä (kuva 16), Junailija hakee rata.digitraffic.fi-rajapinnasta reitille seuraavaksi lähtevät junat ja näyttää ne käyttäjälle listana. Listassa on junan numero, lähtöraide ja lähtöaika valitulta lähtöasemalta. Junia voi myös selata taaksepäin ja eteenpäin ajassa yläpalkin nuolista. Valitsemalla yhden junan näkee lisätietoja valitulle junalle. Lisätiedoissa on lueteltuna eri vaunuissa sijaitsevat palvelut, jos niitä junassa on, sekä lähtö- ja saapumistietojen lisäksi junan senhetkinen sijainti. Kos-

ka rajapinnasta saadut tiedot ovat ajan tasalla, ovat myös Junailijan näyttämät ajat viimeisintä tietoa. Junailija myös päivittää automaattisesti näyttämänsä ajat minuutin välein. Myös käyttäjän on mahdollista päivittää junan yksityiskohtaiset tai listan tiedot. Tämän lisäksi sovellus sisältää asetukset, joista voi säätää listassa näytettävien junien määrää, lähettää palautetta ja selata sovelluksen käyttöehtoja ja ohjeita.



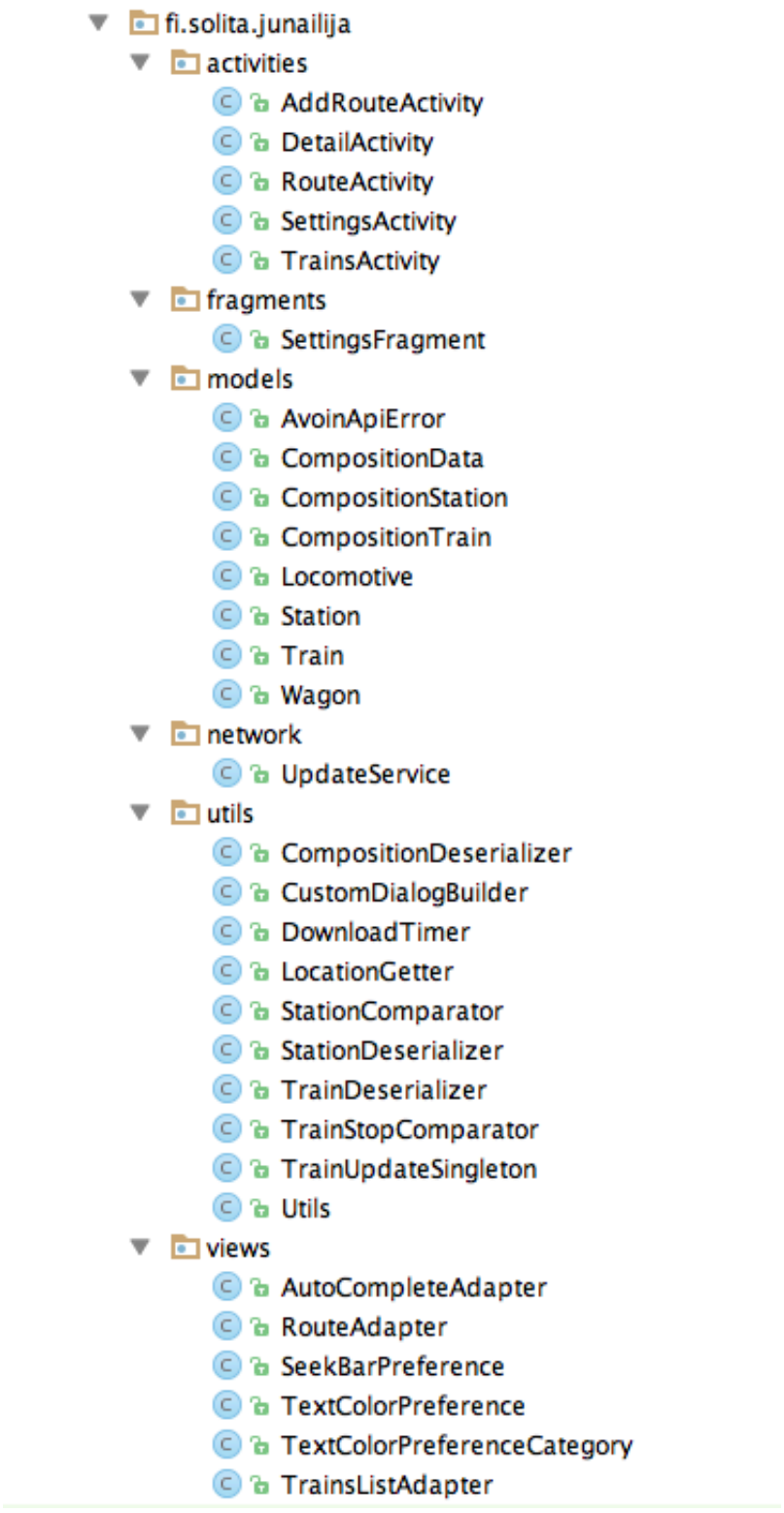
Kuva 16. Junailijan reitti- ja junalistanäkymät.

4.1.2 Sovelluksen rakenne ja toteutus

Junailija kehitettiin Android Studiossa, johon asennettiin Fabric-palvelun liitännäinen. Sovelluksen minimikohteena ovat Android-laitteet, jotka käyttävät Android-versiota 4.0 tai uudempaa. Androidin tukikirjastojen AppCompat v7 ja Support Library v4 lisäksi sovelluksessa käytettiin Google Play Servicesiä, OkHttp:ta, Okiota, Gsonia, Android inApp Feedbackia ja Crashlyticsia. Tukikirjastojen avulla saatiin uusimpia Android-ominaisuuksia toimimaan vanhemmilla Android-versioilla. Google Play Services tarvittiin sijaintitietoja ja Google Analyticsia varten. OkHttp:ta ja Okiota käytettiin HTTP-yhteyksien luomiseen ja Gsonia puolestaan HTTP-kyselyiden JSON-muotoisten vastausten käsittelyyn. Android inApp Feedbackin avulla toteutettiin asetuksissa oleva pa-

lautteen lähetys. Crashlyticsin avulla sovellus jaettiin testikäyttöön ja seurattiin sovelluksessa tapahtuvia käsittelemättömiä poikkeuksia sekä muutamia mittareita.

Kuvassa 17 on sovelluksen Java-luokkien rakenne. Activities-paketti sisältää sovelluksen Activity-komponentit eli päänäkymät, jotka näkyvät käyttäjälle. Fragments-paketissa on listamuotoisten asetuksien kannalta välttämätön SettingsFragment-komponentti. Models-paketissa on rata.digitraffic.fi-rajapinnan vastauksien tietorakennetta heijastavat Java-luokat. Network-paketti sisältää sovelluksen verkkoyhteyksiin liittyvät toiminnot käsittelevän UpdateService-komponentin. Utils-paketti sisältää sovelluksen toimintoja tukevia luokkia, kuten JSON-vastauksien käsittelyssä auttavat Deserializer-luokat ja Utils-luokan, joka sisältää sovelluksen kannalta yleishyödyllisiä metodeja. Views-paketin luokat liittyvät tiedon näyttämiseen näkymissä.



Kuva 17. Junailijan Java-luokkien rakenne.

Sovelluksen käyttöliittymä päätettiin toteuttaa pelkästään Activity-komponenteilla Activity- ja Fragment-komponenttien yhdistelmän sijaan. Sovelluksen käyttöliittymä haluttiin pitää mahdollisimman yksinkertaisena, sillä itse sovelluksen ideakin oli yksinkertainen. Toisin sanoen ajatuksena oli, ettei sovelluksen käyttöliittymä sisällä monimutkaisia rakenteita, jotka vaativat useita komponentteja toimiakseen. Tämän takia Fragment-komponentit eivät olleet välttämättömiä muualla kuin asetuksissa. Vaikka Fragment-komponentteja voidaan käyttää myös Activity-komponenttien tukena erottelemaan käyttöliittymän koodi siihen liittymättömästä koodista, työn aikana koettiin, että käyttöliittymän ja näkymien välisten siirtojen hallinta ohjelmallisesti helpottuu ja yksinkertaistuu, kun Fragment-komponentit jättää pois käyttöliittymästä kokonaan. Sen sijaan kaikki käyttöliittymän osat sijaitsevat Activity-komponenttien ja valikoiden XML-pohjapiirustuksissa. Poikkeuksena on SettingsActivity ja SettingsFragment, joiden avulla toteutettiin Android-sovelluksille tyypillinen listamainen asetusnäkymä.

Sovelluksen junia kuvaava tietorakenne luotiin lähes identtiseksi rajapinnan rakenteen kanssa ladatun tiedon jäsentelyn nopeuttamiseksi ja yksinkertaistamiseksi. Sovelluksen aikaisessa kehitysvaiheessa kokeiltiin streamausjäsentelijää, jonka käyttö osoittautui rajapinnan vastausten suuren koon vuoksi liian raskaaksi ja monimutkaiseksi. Tämän jälkeen rajapinnan vastauksia on muokattu kevyemmiksi, joten streamausjäsentelijän käyttö saattaisi olla ajankohtaista.

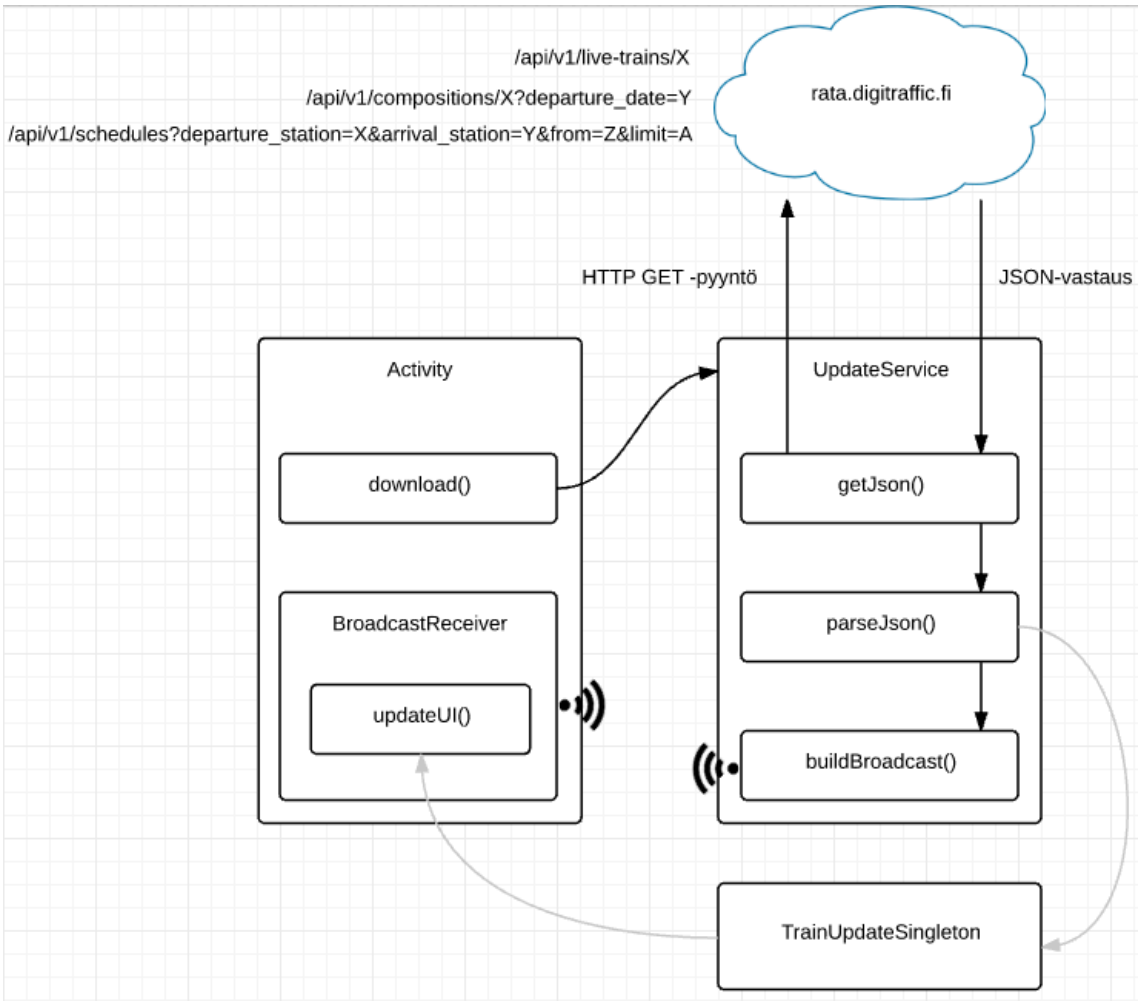
4.1.3 Rata.digitraffic.fi-rajapinnan käyttö

Junailija rakentuu avoimen rata.digitraffic.fi-rajapinnan päälle, jonka tarkoitus on tarjota ajankohtaista tietoa Suomen rautateillä kulkevista junista. Rajapinnan avulla on mahdollista tarkastella junien aikatauluja, lähtö- ja saapumisasemia sekä junien kokoonpanoja eli vaunuja ja vaunujen tietoja. Tämän lisäksi rajapinnan kautta voi tarkastella junien historiatietoja. Rajapinta on Liikenneviraston omistama. [47]

Sovellus hyödyntää rajapinnan tarjoamaa aikataulutietojen hakua, reaaliaikaista seurantaa ja kokoonpanotietoja. Aikataulutiedot palauttavat tiedot muun muassa tietylle reitille kulkeville junille. Aikataulutietojen hakua hyödynnetäänkin listassa, jossa näytetään lähtevät junat valitulle reitille. Aikataulutiedot eivät kuitenkaan sisällä ennusteita tai toteutuneita aikoja junille. [47] Jotta Junailija voi tarjota käyttäjälle lähes reaaliaikaista tietoa junien aikatauluista, sen on hyödynnettävä myös rajapinnan reaaliaikaista seurantaa. Kun lista junista tai junan yksityiskohtaiset tiedot päivitetään sovelluksessa,

sovellus ottaa yhteyttä rajapinnan reaaliaikaiseen seurantaan ja palauttaa yhden junan tämänhetkiset aikataulutiedot. Tämän jälkeen näkymässä olevan junan tiedot päivitetään. Kokoonpanotiedot ladataan rajapinnasta, kun käyttäjä valitsee listasta yhden junan, jonka yksityiskohtaisia tietoja hän haluaa selata. Kokoonpanotiedot sisältävät tietoa junan lähtö- ja pääteasemasta sekä veturista ja vaunuista ja vaunujen palveluista. [47]

Kuva 18 kuvaa sovelluksen toimintaa rata.digitraffic.fi-rajapintaa vasten yksinkertaistetusti. Tietojen lataus rajapinnasta aloitetaan Activity-komponentista, jossa käynnistetään UpdateService. UpdateService on IntentService-luokan laajennus. IntentService on luokka, joka suoritetaan sovelluksessa taustalla. Kun IntentService käynnistyy, sitä ei voi pysäyttää, vaan se suorittaa aina toimintonsa loppuun asti. [48] UpdateServicen lataama tieto määritellään käynnistystä ennen Intent-oliolla, jolle annetaan parametrina tarvittavat junaan tai reittiin liittyvät tiedot. Tämän jälkeen UpdateService tekee HTTP GET -pyynnön rajapintaan, joka antaa vastauksena JSON-muotoista tietoa. Tieto jäsennellään Gson-kirjaston avulla. Jos pyynnön palauttama tieto sisältää halutut asiat, jäsentelyn lopputuloksena on yksi tai useampia Train- tai CompositionTrain-olioita. Nämä oliot asetetaan TrainUpdateSingleton-olion muuttujiksi, minkä jälkeen lähetetään sovelluksen sisäinen lähetys, joka ilmoittaa tietojen haun olevan valmis. Alkuperäisen Activityn BroadcastReceiver-komponentti kuuntelee lähetystä. Vastaanotettuaan lähetyksen se hakee TrainUpdateSingleton-oliolta ladatut tiedot ja päivittää käyttöliittymän.



Kuva 18. Sovelluksen vuorovaikutus rata.digitraffic.fi-rajapinnan kanssa.

Usein Activity- ja Service-komponenttien välinen tiedonsiirto toteutetaan Intent-oliolle annettavalla Bundle-oliolla, joka sisältää siirrettävät tiedot. Tätä tapaa käytettiin sovelluksessa alkuvaiheessa. Bundlen käyttö ei kuitenkaan onnistunut, sillä rajapinnan palauttaman tiedon koko ylitti Bundlen sisältämän tiedon enimmäiskoon, jolloin sovellus kaatui. Tämä ongelma kierrettiin TrainUpdateSingleton-luokan avulla. Singleton-luokasta on mahdollista luoda vain yksi instanssi, joka on globaalisti saatavilla. [49] TrainUpdateSingletonista on siis olemassa vain yksi instanssi, jota on mahdollista käyttää missä tahansa sovelluksen osassa. Tämä instanssi sisältää UpdateServicen lataamat tiedot, joita Activity-komponentit hyödyntää.

4.1.4 Testdroid Cloud -testaus

Ennen sovelluksen julkaisua sitä testattiin Testdroid Cloud -palvelun ilmaisversiossa. Palvelussa sovellus asennetaan ja käynnistetään monella eri laitteella ja tutkitaan, miten se käyttäytyy. Testaus haluttiin suorittaa, jotta voitiin varmistaa, että sovellus toimii monilla eri Android-puhelimilla ilman ongelmia. Testauksessa päätettiin käyttää palvelun automaattitestausta eli robottia ennalta määritettyjen painallusten sijasta. Sen hetken viimeisin APK ladattiin palveluun, jossa testilaitteiksi valittiin kaikki ilmaiset Android-laitteet. Tämän jälkeen suoritettiin automaattinen testi. Sovellus toimi kaikilla paitsi yhdellä laitteella, joka käytti Android-versiota 4.0.3. Ongelma johtui kuvan asettamisesta, jonka yhteydessä kutsuttiin metodia, jonka Android-version minimivaatimus oli 4.1. Ongelma korjattiin korvaamalla metodi vanhemmalla metodilla.

Testituloksien kuvakaappauksista huomattiin, ettei robotti ollut edennyt sovelluksessa reittinäkymästä eteenpäin. Tämä johtui siitä, ettei robotti ollut osannut lisätä reittejä, joita valitsemalla sovelluksessa etenee seuraavaan näkymään. Testin jälkeen päätettiin suorittaa toinen testi, jossa testattavaan APK:hon oli ennalta lisätty reittejä. Toisessa testissä robotti eteni toiseen näkymään, mutta ei odottanut, että sovellus olisi ehtinyt ladata reitin junat rajapinnasta. Odottamisen sijasta robotti siirtyi takaisin edelliseen näkymään ja kävi kaikki loput reitit samalla tavalla läpi. Robotti ei siis koskaan edennyt junan yksityiskohtaisiin tietoihin. Ongelma olisi voitu selvittää lisäämällä junalistaan kovakoodattuja junia tai suunnittelemalla testi itse automaatin käytön sijasta. Itse suunnitellussa testissä on nimittäin mahdollista painalluksien lisäksi määritellä taukoja painallusten välille. Testejä ei kuitenkaan päätetty suorittaa, sillä kaikki paitsi yksi näkymä testattiin. Viimeinen näkymä ei myöskään sisältänyt vuorovaikutusta käyttäjän kanssa, joten sen testausta ei pidetty välttämättömänä.

Testdroid Cloudin avulla saatiin siis kiinni yksi poikkeus, jota ei aikaisemmin ollut huomattu. Vaikka poikkeus ei ollut sovelluksen kannalta kriittinen, ottaen huomioon kuinka vähän Android 4.0:lla varustettuja laitteita on käytössä, oltiin palvelun käyttöön ja lopputuloksiin tyytyväisiä. Palvelu oli helppokäyttöinen ja hyödyllinen. Erityisen hyödyllinen palvelu olisi, jos sovellukselle ei suoritettaisi erillistä testausta ennen julkaisua.

4.2 Sovelluksen analytiikka

4.2.1 Yleistä

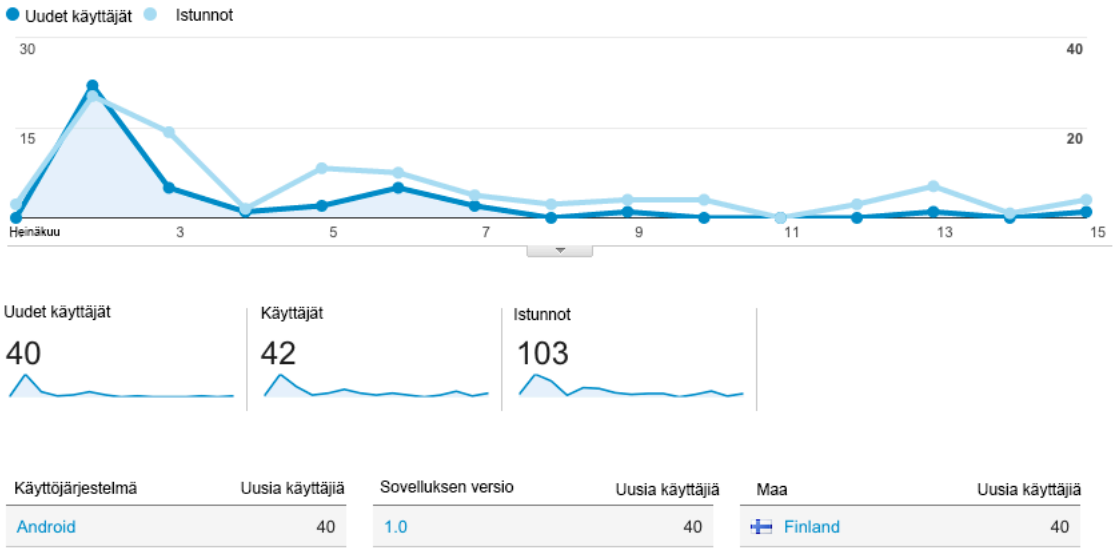
Sovelluksessa käytettäviksi analytiikkatyökaluiksi valittiin Google Analytics ja Fabricin Crashlytics. Alun perin Google Analyticsia suunniteltiin käytettävän enimmäkseen uusien käyttäjien määrän ja istuntojen seuraamiseen, mikä onnistuu Analyticsin automaattisesti keräämillä tiedoilla. Tämän lisäksi Analyticsin avulla sovellukseen haluttiin määritellä erilaisia tapahtumia ja näkymiä, jotka kertovat sovelluksen käytöstä. Myöhemmin sovelluksessa otettiin käyttöön myös latausaikoien seuraus. Fabricista suunniteltiin käytettävän Crashlyticsia, jonka avulla pystyy seuraamaan sovelluksissa tapahtuvia käsittelemättömiä poikkeuksia. Fabricista otettiin myöhemmin käyttöön Beta, jonka avulla sovellus jaettiin testikäyttöön, ja kokeilumielessä myös Answers, jolla seurataan samoja mittareita, joita Google Analyticsillakin voi seurata. Tämän lisäksi ennen sovelluksen julkaisua sitä testattiin Testdroid Cloud -palvelussa mahdollisten virheiden takia, joita ei kehityksessä ja testauksessa ollut huomattu.

4.2.2 Google Analytics sovelluksessa

Google Analytics valittiin yhdeksi työkaluksi insinöörityön sovellukseen siksi, että se oli kohtuullisen helppokäyttöinen, ilmainen ja yritykselle jo entuudestaan tuttu. Yritys ei kuitenkaan ollut käyttänyt työkalua mobiilisovelluksissa, vaan pelkästään verkkosivuilla. Yrityksellä oli jo valmiiksi Google Analytics -tili, mikä helpotti työkalun käyttöönottoa. Google Analytics otettiin käyttöön jo sovelluksen kehitysvaiheessa, kun suurin osa sovelluksen toiminnallisuudesta oli valmiina. Tämän ansiosta työkalua voitiin testata huolehtimatta siitä, että tiedon oikeellisuus kärsisi. Varsinaista mielenkiintoista dataa Analytics alkoi kerätä sovelluksen julkaisun jälkeen.

Google Analyticsin avulla seurattiin uusien käyttäjien määrää sovelluksen julkaisun aikana ja sen jälkeen. Kuvassa 19 on uusien käyttäjien määrä ajan funktiona. Sovellus julkaistiin heinäkuun toinen päivä, ja sitä mainostettiin vain yrityksen sisällä. Julkaisupäivänä uusia käyttäjiä tuli jonkin verran, mutta sen jälkeen uusien käyttäjien määrä väheni huomattavasti. Julkaisun ajankohta ei ollut hyvä, sillä heinäkuussa suurin osa ihmisistä on lomalla. Toisin sanoen moni ei välttämättä edes huomannut sovelluksen julkaisua. Tämän lisäksi Junailija on tarkoitettu ihmisille, jotka kulkevat usein junalla. Lomalla junalla usein kulkeminen ei välttämättä ole kovin yleistä, mikä näkyy sovelluk-

sen vähäisenä käyttönä. Julkaisun jälkeen uusia käyttäjiä on tullut tasaisesti 0–2 päivässä. Julkaisupäivänä käyttäjiä oli enemmän kuin istuntoja, mikä kertoo siitä, että moni latasi sovelluksen, muttei kertaakaan avannut sitä.



Kuva 19. Junailijan uudet käyttäjät ja istunnot julkaisun jälkeisenä kahtena viikkona hankintaraportissa [muokattu lähteestä 29].

Tapahtumia määriteltiin sovelluksessa monelle eri toiminnolle. Tapahtumien tarkoituksena oli seurata, miten sovellusta käytetään, ja tarkastella, jääkö jokin sovelluksen toiminto käyttäjiltä huomaamatta. Tapahtumille määriteltiin Google Analyticsin tapahtumien mukaisesti kategoria, toiminto ja tunniste. Arvoa ei tapahtumille määritelty, sillä kaikki tarvittava tieto saatiin välitettyä kolmen parametrin avulla.

Taulukko 1 kuvaa reittien ja junalistan tapahtumien rakennetta. Reitille määriteltiin neljä eri toimintoa eli neljä erilaista tapahtumaa. Kun käyttäjä haluaa nähdä junat tietylle reitille ja valitsee reitin, lähetetään reitti valittu -tapahtuma. Kun käyttäjä lisää sovelluksessa uuden reitin, lähetetään reitti lisätty -tapahtuma. Reitti poistettu -tapahtuma lähetetään, kun käyttäjä poistaa reitin ja paluureitti lisätty -tapahtuma, kun olemassa olevalle reitille lisätään paluureitti. Kaikki tapahtumat lähettävät toiminnon tunnisteena reitin, jolle toiminto suoritettiin. Junat kuvaa sovelluksen junalistassa tapahtuvia toimintoja. Junat päivitetty -tapahtuma lähetetään, kun käyttäjä päivittää listan. Seuraavat ja edelliset junat painettu -tapahtumat lähetetään, kun käyttäjä selaa tulevia tai menneitä junia reitille. Huono reittihaku -tapahtuma puolestaan lähetetään, kun käyttäjä yrittää hakea

reittiä, jolla ei kulje junia. Huono reittihaku lähettää tunnisteena reitin, jota yritettiin hakea.

Taulukko 1. Reittien ja junalistan tapahtumien rakenne.

Kategoria	**Reitti**	**Junat**
toiminto	reitti valittu	junat päivitetty
toiminnon tunniste	reitti	
toiminto	reitti lisätty	seuraavat junat painettu
toiminnon tunniste	reitti	
toiminto	reitti poistettu	edelliset junat painettu
toiminnon tunniste	reitti	
toiminto	paluureitti lisätty	huono reittihaku
Toiminnon tunniste	Reitti	Reitti

Taulukko 2 kuvaa junan ja asetuksien tapahtumien rakennetta. Junalle on määritelty kaksi tapahtumaa: juna valittu ja juna päivitetty. Juna valittu -tapahtuma lähetetään, kun juna valitaan listasta ja juna päivitetty -tapahtuma, kun käyttäjä päivittää yksittäisen junan yksityiskohtaiset tiedot. Molemmat tapahtumat lähettävät tunnisteena sen junan numeron, jolle toiminto suoritettiin. Asetuksissa on määritelty yksi tapahtuma, joka lähetetään, kun käyttäjä muuttaa listassa näytettävien junien määrää. Tunnisteena toiminnon mukana lähetetään käyttäjän valitsema määrä.

Taulukko 2. Junan ja asetuksien tapahtumien rakenne.

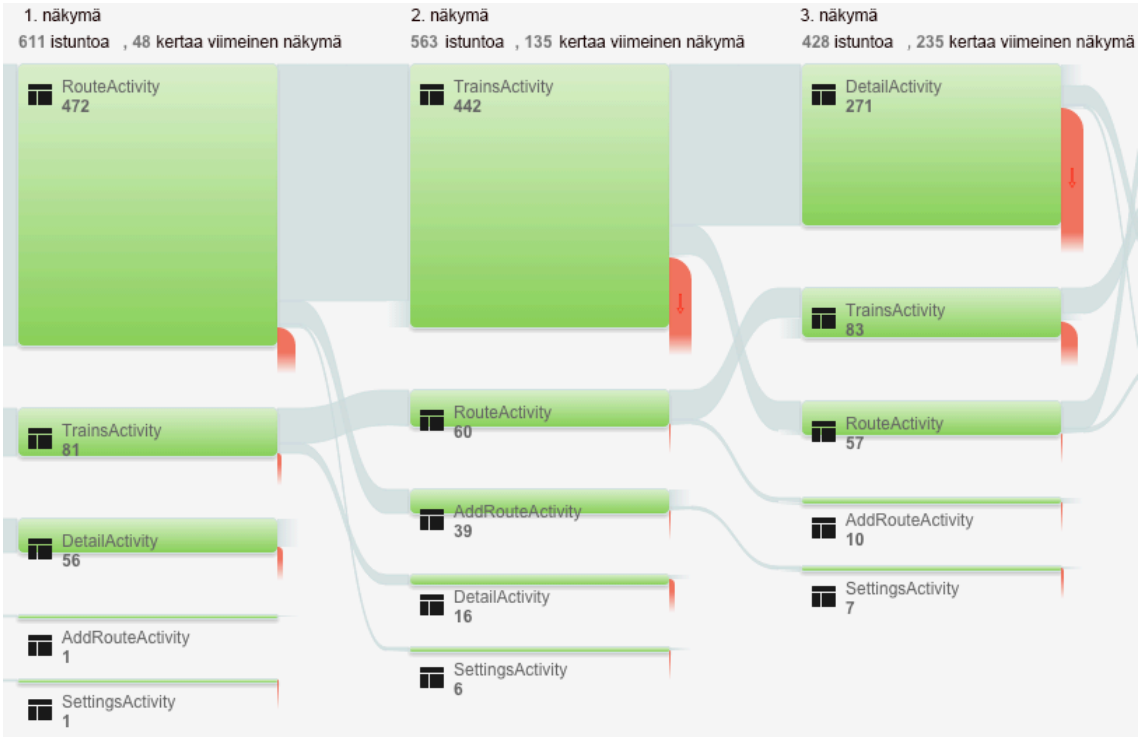
Kategoria	**Juna**	**Asetukset**
toiminto	juna valittu	junien määrä muutettu
toiminnon tunniste	juna	määrä
toiminto	juna päivitetty	
toiminnon tunniste	juna	

Google Analyticsin hallintapaneelista selviää, kuinka monta kertaa jokainen tapahtuma on tapahtunut sovelluksessa. Tapahtumat voidaan järjestää myös määrän perusteella, minkä avulla selviää suosituimmat ja vähiten tapahtuneet tapahtumat. Suosituimpia tapahtumia ovat reitin valinta ja junan valinta, mikä ei ole yllätys, sillä nämä tapahtumat

ovat sovelluksen käytön kannalta kaikkein oleellisimpia. Niiden avulla käyttäjä saa selville haluamansa tiedon, eli milloin lähtee juna halutulle reitille. Selkeästi vähiten sovelluksessa on tapahtunut reitin poistoja ja paluureitin lisäyksiä. Molemmat toiminnot on mahdollista suorittaa painamalla reittiä pohjaan, jolloin aukeaa valikko, josta voi valita toimintojen välillä. Tapahtumien vähäinen määrä kertoo joko siitä, etteivät käyttäjät ole löytäneet toimintoja, tai siitä, etteivät käyttäjät lisää vahingossa vääriä reittejä, joita poistaa.

Jokaiselle sovelluksen Acitivity-komponentille eli käyttöliittymän näkymälle määritettiin oma näkymä, joka lähetetään Google Analyticsille, kun näkymä näytetään sovelluksessa. Näkymät nimettiin Java-luokkien mukaan: AddRouteActivity, DetailActivity, RouteActivity, SettinsActivity ja TrainsActivity. Näkymien avulla voidaan selvittää, miten käyttäjät siirtyvät sovelluksessa näkymästä toiseen.

Kuva 20 esittää käyttäjien kulkua sovelluksessa. 1. näkymä tarkoittaa näkymää, joka avautuu ensimmäisenä, kun käyttäjä on käynnistänyt sovelluksen. 2. näkymä tarkoittaa toista ja 3. kolmatta näkymää, jossa käyttäjä on käynyt sovellusta käyttäessään. Näkymien lukumäärä jatkuu myös eteenpäin, mutta istuntojen määrä puolittuu siirryttäessä kolmannesta näkymästä neljänteen, joten iso osa käyttäjistä lopettaa sovelluksen käytön kolmannen näkymän jälkeen. Näkymän nimen alla oleva luku kertoo näkymän katselukertojen määrän. Siniharmaat alueet kuvaavat siirtymisiä näkymästä toiseen ja punaiset sovelluksen sulkemista näkymän kohdalla. Kuvasta näkyy, miten suurin osa käyttäjistä valitsee ensin reitin, sen jälkeen junan listasta ja päätyy lopulta junan yksityiskohtaisiin tietoihin. Tämä vastaa sovelluksen pääkäyttötarkoitusta, eikä siis ole yllättävää. Lähes puolet käyttäjistä, jotka päätyvät junalistaan, eivät valitse yksityiskohtaisia tietoja. Tämä johtunee siitä, että listasta selviävät junan tärkeimmät tiedot, eivätkä yksityiskohtaiset tiedot siksi ole välttämättömiä käyttäjälle. Siirtymät näkymissä edes takaisin selittyvät takaisinpainikkeella.



Kuva 20. Google Analyticsin näkymien perusteella muodostettu kävijän kulku [muokattu lähteestä 29].

Latausnopeuksien mittaaminen sovelluksissa jaettiin kahteen eri kategoriaan: junalistan ja yksittäisen junan lataukseen. Listan latauksessa erilaisia mittauksen kohteita ovat reitille kulkevat junat tällä hetkellä, menneisyydessä ja tulevaisuudessa sekä JSON-vastauksien jäsentely ja latauksen ja jäsentelyn yhteenlaskettu aika. Yksittäistä junaa ladatessa mitataan latauksen, jäsentelyn ja molempien yhteenlaskettua kestoa. Lähetetyt latausajat sisältävät myös ladatun JSON-vastauksen koon. Latausaikoja haluttiin mitata, jotta voitaisiin huomata, jos latausajat jostain syystä kasvaisivat kohtuuttoman suuriksi.

4.2.3 Fabric sovelluksessa

Fabric valittiin työkaluksi sovellukseen ensisijaisesti sen ilmeisen helpon kaatumisten hallinnan ja ilmaisuuden vuoksi. Työkalu valittiin osittain kokeilumielessä, sillä yrityksellä ei ole aikaisempaa kokemusta mobiilisovelluksen kaatumisten seuraamisesta. Työkalua olikin tarkoitus aluksi käyttää vain kaatumisten seuraamiseen, mutta kehityksen edetessä haluttiin kokeilla myös muita työkalun tarjoamia palveluita.

Crashlytics oli ensimmäinen Fabricin osa, joka otettiin sovelluksessa käyttöön. Kun Crashlytics otettiin käyttöön, oli sovellus vain parin ihmisen käytössä. Tänä aikana kokeiltiin, miten Crashlytics toimii. Crashlyticsin käyttöönotto onnistui lisäämällä sovelluksen lähdekoodiin vain muutama rivi koodia. Tämän jälkeen kaikista sovelluksen kaatumisista lähti tieto Fabricin hallintapaneeliin, jonka Crashlytics-osassa listataan kaikki sovelluksessa tapahtuneet poikkeukset ja niiden kuvaukset sekä koodirivi, jolla virhe tapahtui. Hallintapaneelissa poikkeuksia pystyy merkitsemään korjatuiksi, jolloin ne siirtyvät korjattujen poikkeusten alle. Crashlytics oli testien aikana erittäin hyödyllinen työkalu, koska sen avulla saatiin korjattua monia virheitä, joita ei olisi välttämättä löydetty ilman Crahlyticsin käyttöä ennen julkaisua.

Kun tehdään testi- tai julkaisuversiota Android-sovelluksesta, tulee ottaa huomioon ProGuardin käyttö. ProGuard on työkalu, joka lyhentää muuttujien, luokkien ja metodien nimiä ja poistaa käyttämätöntä koodia. Tällä tavoin ProGuard pienentää lopullisen APK:n kokoa ja vaikeuttaa sen muuntamista takaisin Android Studio -projektiksi ja luettavaksi koodiksi. ProGuard myös hävittää koodista rivien numerot, mikä hankaloittaa Crashlyticsin listaamien poikkeuksen paikantamista. [50] ProGuardia käytettäessä kannattaakin määritellä proguard-rules.pro-tiedostossa sääntö, joka estää rivinumeroiden hävittämisen.

Kun sovellusta haluttiin testata suuremmalla määrällä käyttäjiä, otettiin käyttöön Fabricin Beta-testijakelupalvelu. Betan avulla sovellus jaettiin yrityksen sisällä vapaaehtoisille testaajille. Ensin valmis APK raahattiin ja pudotettiin Fabricin Android Studion liitännäiseen, minkä jälkeen julkaisulle pystyi antamaan kommentteja. Myöhemmin testiversiota päivittäessä kommenteissa listattiin korjatut ongelmat ja uudet ominaisuudet. Testaajien sähköpostiosoitteet syötettiin liitännäiseen testiversion julkaisun jälkeen, jolloin Fabric lähetti testaajille sähköpostina kutsun. Kutsun kautta testaajat pystyivät asentamaan laitteisiinsa Crashlyticsin sovelluksen, jonka kautta he pystyivät asentamaan testisovelluksen. Hallintapaneelin kautta nähtiin, kuka oli hyväksynyt kutsun, asentanut sovelluksen ja käynnistänyt sen. Hallintapaneelista selvisi myös, jos testaaja ei ollut asentanut uutta versiota sovelluksesta, sekä ketkä testaajista olivat aktiivisia. Aktiivisuudella tarkoitetaan Answersissä sovelluksen käynnistämistä. Kun testiversioita oli julkaistu muutama ja halutut ominaisuudet sovellukseen lisätty sekä virheet korjattu, sovellus päätettiin julkaista Google Play Store -sovelluskaupassa. Testaaminen Beta-sovelluksen avulla sujui hyvin. Testaajien lisääminen ja sovelluksen päivittäminen oli todella helppoa ja nopeaa. Yhdessä Crashlyticsin kanssa Beta muodostaa otollisen

työkaluparin, jonka avulla sovelluksen jako ja kaatumisten hallinta onnistuu sujuvasti testivaiheessa.

Jonkin aikaa julkaisun jälkeen päätettiin Fabricista ottaa käyttöön myös Answers. Koska Answers asentuu Crashlyticsin mukana, ei itse sovellukselle tarvinnut tehdä mitään. Sen sijaan Answers kytkettiin päälle Fabricin hallintapaneelista. Answers otettiin käyttöön kokeilumielessä, sillä tarvittavat tiedot sovelluksen hankinnasta ja käytöstä olivat jo saatavilla Google Analyticsin keräämästä tiedosta. Answersin avulla kerättiin vain sen automaattisesti keräämää tietoa, eikä erillisiä tapahtumia sen avulla määritelty. Answersin käyttöliittymä on hyvin paljon yksinkertaisempi kuin Google Analyticsin ja samalla myös pinnallisempi. Raportteja on viisi erilaista, ja ne sisältävät diagrammeja erilaisista mittareista. Raportteja ei voi muokata, sovelluksen käyttäjiä ei voi segmentoida, eikä diagrammeissa tarkasteltavaa aikaväliä pysty muuttamaan. Answers näyttää siis nopeasti sovelluksen oleellisimmat mittarit, mutta ei paljon muuta. Työtä tehdessä koettiin, ettei Answers antanut lisää informaatiota Google Analyticsiin verrattuna. Yksittäin käytettynä Answers on kuitenkin erittäin hyvä lisä sovellukseen, jos aikaa analytiikan suunnitteluun ja toteuttamiseen ei ole paljon.

5 Yhteenveto

Analytiikka mahdollistaa mobiilisovelluksen menestymisen ja helpottaa huomattavasti sovelluksen kehitystä. Ilman analytiikkaa on mahdotonta tietää, miten sovellusta käytetään ja miten se toimii käyttäjien laitteilla. Sovelluskaupoista selviävät latausten määrä ja sovellukselle annetut arvosanat, mutta pelkästään ne eivät riitä sovelluksen menestymisen analysoimiseen ja jatkokehityksen suunnitteluun. Analytiikan avulla kehittäjät voivat saada syvällisempää tietoa sovelluksen käytöstä ja apua paikantamaan poikkeuksia ja suunnitteluvirheitä sovelluksessa. Nykypäivänä sovelluskauppojen kilpailun ollessa äärimmäisen kovaa analytiikka onkin välttämätöntä sovelluksessa, mikäli sitä halutaan parannella ja sen saavan lisää käyttäjiä.

Insinöörityön tavoitteena oli toteuttaa Android-sovellus, joka näyttää käyttäjälle ajan tasalla olevaa aikataulutietoa junista käyttäjän määrittelemille reiteille hyödyntämällä rata.digitraffic.fi-rajapintaa. Tämän lisäksi tavoitteena oli toteuttaa sovellukselle kaatumisten raportointi ja betatestaus sekä seurata sovelluksen käyttöä ja hankintaa kahden analytiikkatyökalun avulla. Sovellus toteutettiin Android Studiossa ja analytiikka ja tes-

taus Google Analyticsin ja Fabricin avulla. Tämän lisäksi sovellusta testattiin ennen julkaisua Testdroid Cloud -palvelussa.

Työn toteutus onnistui erittäin hyvin, ja lopputuloksena saatiinkin toimiva sovellus, jonka avulla käyttäjät voivat muun muassa selata lähteviä junia valitsemalleen reitille. Sovellus myös julkaistiin Play Storessa, mikä toi odotetusti pienen määrän käyttäjiä sovellukselle. Sovelluksen testaus ja analytiikkatiedon kerääminen onnistuivat myös hyvin.

Analytiikkatyökalut otettiin onnistuneesti käyttöön sovelluksessa, ja ne suoriutuivat niille tarkoitetuista toiminnoista kuten odotettiin. Ennen julkaisua sovellus jaettiin testikäyttöön Fabricin Beta-moduulin avulla, mikä sujui vaivattomasti. Samaan aikaan alettiin seurata sovelluksissa tapahtuvia poikkeuksia, minkä avulla testissä löydettiin ja korjattiin monia virheitä ennen sovelluksen julkaisua ja sen jälkeen. Julkaisun jälkeen sovelluksen hankintaan ja käyttöön liittyvää tietoa kerättiin Google Analyticsin ja Fabricin moduulien avulla. Kerätyn tiedon perusteella onnistuttiin tekemään johtopäätöksiä muun muassa siitä, miten käyttäjät käyttivät sovellusta, sekä siitä, miten paljon uusia käyttäjiä minäkin päivänä sovellukselle tuli.

Työkaluilla olisi mahdollista tehdä myös syvällisempää analyysiä sovelluksen käytöstä, mutta sille ei nähty tarvetta. Mobiilianalytiikkaa varten on olemassa myös monia muita työkaluja, jotka saattavat tarjota toimintoja, joita tässä työssä ei otettu huomioon. Sovelluksen analytiikka onkin mahdollista toteuttaa monella tavalla, joista tässä työssä käytetty oli yksi. Onnistuneen tiedonkeräyksen ja helpon käyttöönoton nojalla sanoisin, että tämä tapa oli toimiva. Sen ainoana heikkoutena oli Google Analyticsin käyttöönoton hitaus verrattuna muihin, nopeammin asennettaviin työkaluihin.

Jatkossa sovellusta kehitetään lähinnä mahdollisten käyttäjien lähettämien palautteiden ja sovelluksessa tapahtuneiden kaatumisten perusteella. Sovelluksessa saatetaan ottaa käyttöön lisää analytiikkaa tarpeen vaatiessa tai kokeilujen vuoksi. Haasteena on uusien käyttäjien hankkiminen, sillä sovellusta ei aiota markkinoida julkisesti.

Lähteet

1 Lella, Adam, Lipsman, Andrew & Martin, Ben. 2015. The 2015 U.S. mobile app report. ComScore. E-kirja, saatavissa yritykseltä: https://www.comscore.com/Insights/Presentations-and-Whitepapers/2015/The-2015-US-Mobile-App-Report.

2 Number of mobile app downloads worldwide from 2009 to 2017 (in millions). 2013. E-kirja. Statista. <http://www.statista.com/statistics/266488/forecast-of-mobile-app-downloads/>. Luettu 26.10.2015.

3 Stop guessing, start measuring: 5 mobile metrics for great apps. 2015. Appcelerator. E-kirja, saatavissa yritykseltä: http://www.appcelerator.com/resource-center/white-papers/5-mobile-metrics-for-great-apps/

4 Cooper, Adam. CETIS analytics series volume 1, no 9: A brief history of analytics. 2012. E-kirja. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.269.6470&rep=rep1&type=pdf>. Luettu 27.10.2015.

5 Ledford, Jerri, Teixeira, Joe & Tyler, Mary E. 2010. Google Analytics. Third Edition. Indiana: Wiley Publishing.

6 The beginner's guide to app analytics. 2014. Verkkodokumentti. Localytics. <http://info.localytics.com/hs-fs/hub/367525/file-1252512142-pdf/eBooks/A_Beginners_Guide_to_App_Analytics_.pdf?t=1445893159415>. Luettu 29.10.2016.

7 Dykes, Brent. 2013. Web analytics vs. mobile analytics: what's the difference? Verkkodokumentti. Web Analytics Action Hero. <http://www.analyticshero.com/2013/07/24/web-analytics-vs-mobile-analytics-whats-the-difference/>. Luettu 29.10.2016.

8 Cutroni, Justin. 2012. Cohort analysis with Google analytics. Verkkodokumentti. <http://cutroni.com/blog/2012/12/11/cohort-analysis-with-google-analytics/>. Luettu 16.12.2015.

9 Top 10 KPIs to measure mobile app development success. 2015. Verkkodokumentti. Neosperience. <http://blog.neosperience.com/top-10-kpis-to-measure-mobile-app-development-success>. Luettu 12.12.2015.

10 Drell, Lauren. 2013. 9 mobile app KPIs to know. Verkkodokumentti. Mashable. <http://mashable.com/2013/09/04/mobile-app-metrics/#5u.SnQg8Fiqm>. Luettu 12.12.2015.

11 Measuring the success of your mobile strategy. 2010. Verkkodokumentti. Webtrends. <http://static1.1.sqspcdn.com/static/f/497415/7402918/1276980902067/kfann+white+paper.pdf?token=IZTRfzfzY1LxpBsFwgieapdSkvY%3D>. Luettu 29.10.2016.

12 Cropper, Erin. 2012. Mobile whitepaper: Overcoming primary obstacles ot effective mobile analysis. Verkkodokumentti. Stratigent. <http://www.waafiles.org/whitepapers/Mobile_Whitepaper.pdf>. Luettu 29.10.2016.

13 Skidmore, Stephen. 2014. What is your mobile analytics strategy to drive adoption (and ROI)? Verkkodokumentti. Apperian. <https://www.apperian.com/mam-blog/mobile-analytics-strategy-drive-adoption-roi/>. Luettu 16.12.2015.

14 Montgomery, Justin. 2010. Why is mobile marketing & advertising so hard to measure? Verkkodokumentti. Mobile Marketing Watch. <http://mobilemarketingwatch.com/why-is-mobile-marketing-advertising-so-hard-to-measure-5243/>. Luettu 11.12.2015.

15 Usage of traffic analytics tools for websites. 2015. Verkkodokumentti. W3Techs. <http://w3techs.com/technologies/overview/traffic_analysis/all>. Luettu 3.10.2015.

16 Rouse, Margaret. 2011. Google Analytics Definition. Verkkodokumentti. TechTarget. <http://searchbusinessanalytics.techtarget.com/definition/Google-Analytics>. Luettu 3.10.2015.

17 Waisberg, Daniel. 2011. Google Analytics Premium: Better Support & Goodbye Data Sampling. <http://searchengineland.com/google-analytics-premium-better-support-goodbye-data-sampling-94997>. Luettu 3.10.2015.

18 Platform components. 2015. Verkkodokumentti. Learn about Google Analytics. <https://developers.google.com/analytics/devguides/platform/>. Luettu 3.10.2015

19 How Google Analytics works. 2015. Verkkodokumentti. Google Analytics Help. <https://support.google.com/analytics/answer/6081186?hl=en&ref_topic=6080724&vid=1-635792996529093677-2914465513>. Luettu 3.10.2015.

20 Mobile app data collection. 2015. Verkkodokumentti. Google Analytics Help. <https://support.google.com/analytics/answer/6083697?hl=en&ref_topic=6083692&vid=1-635792996529093677-2914465513>. Luettu 3.10.2015.

21 Google Analytics Platform Principles – Lesson 1.2 The platform components. 2014. Verkkodokumentti. Google Analytics. <https://www.youtube.com/watch?t=157&v=WlgESwo55xk>. Katsottu 3.10.2015.

22 Overview of Google Play Services. 2015. Verkkodokumentti. Google APIs for Android. <https://developers.google.com/android/guides/overview>. Luettu 4.10.2015.

23 Setting Up Goole Play Services. 2015. Verkkodokumentti. Google APIs for Android. <https://developers.google.com/android/guides/setup>. Luettu 4.10.2015.

24 Tracker. 2015. Verkkodokumentti. Google APIs for Android. <https://developers.google.com/android/reference/com/google/android/gms/analytics/Tracker>. Luettu 16.10.2015.

25 Analytics for Android Apps. 2010. Verkkodokumentti. Android Developers Blog. <http://android-developers.blogspot.fi/2010/12/analytics-for-android-apps.html>. Luettu 15.10.2015.

26 Screens – Android SDK v4. 2015. Verkkodokumentti. Analytics for Android. <https://developers.google.com/analytics/devguides/collection/android/v4/screens>. Luettu 16.10.2015.

27 About Events. 2015. Verkkodokumentti. Analytics Help. <https://support.google.com/analytics/answer/1033068?hl=en>. Luettu 16.10.2015.

28 User Timings – Android SDK v4. 2015. Verkkodokumentti. Analytics for Android. <https://developers.google.com/analytics/devguides/collection/android/v4/usertimings>. Luettu 17.10.2015.

29 Sovelluksen Google Analytics -tili. 2015. Verkkodokumentti. Google Analytics. <https://developers.google.com/android/reference/com/google/android/gms/analytics/Tracker>. Luettu 20.10.2015.

30 Mobile App Overview report. 2015. Verkkodokumentti. Analytics Help. <https://support.google.com/analytics/answer/2621230?hl=en>. Luettu 20.10.2015.

31 Mobile App Audience. 2015. Verkkodokumentti. Analytics Help. <https://support.google.com/analytics/answer/2568874?hl=en>. Luettu 20.10.2015.

32 Mobile App Acquisition. 2015. Verkkodokumentti. Analytics Help. <https://support.google.com/analytics/answer/2568794?hl=en>. Luettu 20.10.2015.

33 Mobile App Behavior. 2015. Verkkodokumentti. Analytics Help. <https://support.google.com/analytics/answer/2568878?hl=en>. Luettu 20.10.2015.

34 Segmentit. 2015. Verkkodokumentti. Analytics Ohjeet. <https://support.google.com/analytics/answer/3123951?hl=fi>. Luettu 20.10.2015.

35 Tietoja omista raporteista. 2015. Verkkodokumentti. Analytics Ohjeet. <https://support.google.com/analytics/answer/1033013?hl=fi&ref_topic=1012046&vid=1-635810236519416709-1362891274>. Luettu 20.10.2015.

36 Seibert Jeff. 2014. Introducing Fabric. Verkkodokumentti. Twitter. <https://blog.twitter.com/2014/introducing-fabric>. Luettu 4.1.2015.

37 DeAmicis, Carmel. 2014. Twitter introduces Fabric, its first platform for mobile app developers. Verkkodokumentti. Gigaom. <https://gigaom.com/2014/10/22/twitter-introduces-fabric-its-first-platform-for-mobile-app-developers/>. Luettu 4.1.2015.

38 Chang, Wayne. 2013. Crashlytics is joining forces with Twitter. Verkkodokumentti. Crashlytics. < https://www.crashlytics.com/blog/crashlytics-is-joining-forces-with-twitter/>. Luettu 4.1.2016.

39 Sovelluksen Fabric-tili. 2015. Verkkodokumentti. Fabric. <https://fabric.io/solita/android/apps/fi.solita.junailija/>. Luettu 4.1.2016.

40 Beta Process Walkthrough. 2015. Verkkodokumentti. Fabric. <https://docs.fabric.io/android/beta/introduction.html#distributing-the-app>. Luettu 4.1. 2016.

41 Answers. 2015. Fabric. Verkkodokumentti. <https://docs.fabric.io/android/answers/index.html>. Luettu 4.1.2016.

42 Answers Metrics. 2015. Fabric. Verkkodokumentti. <https://docs.fabric.io/android/answers/answers-metrics.html>. Luettu 5.1.2016.

43 Kaasila, Jouko, Ferreira Denzil, Kostakos, Vassilis & Ojala, Timo. 2015. Testdroid: automated remote UI testing on Android. Verkkodokumentti. Oulun yliopisto. <http://www.ee.oulu.fi/~vassilis/files/papers/mum12a.pdf>. Luettu 5.1.2016.

44 Helppi, Ville-Veikko. 2014. Automatic iOS application testing with Testdroid App Crawler – available now!. Verkkodokumentti. Testdroid. <http://testdroid.com/news/automatic-ios-app-testing-with-testdroid-app-crawler>. Luettu 5.1.2016.

45 Yrityksen Testdroid-tili. 2016. Testdroid Cloud. Verkkodokumentti. <https://cloud.testdroid.com>. Luettu 5.1.2016.

46 Crashes & exceptions – Android SDK v4. 2015. Verkkodokumentti. Analytics For Android. <https://developers.google.com/analytics/devguides/collection/android/v4/exceptions>. Luettu 11.1.2016.

47 Rata.digitraffic.fi. 2015. Verkkodokumentti. Liikennevirasto. <http://rata.digitraffic.fi/api/v1/doc/index.html>. Luettu 11.1.2016.

48 IntentService. 2016. Verkkodokumentti. Android developers. <http://developer.android.com/reference/android/app/IntentService.html>. Luettu 12.1.2016.

49 Simply Singleton. 2003. Verkkodokumentti. JavaWorld. <http://www.javaworld.com/article/2073352/core-java/simply-singleton.html>. Luettu 15.1.2016.

50 ProGuard. 2016. Verkkodokumentti. Android developers. <http://developer.android.com/tools/help/proguard.html>. Luettu 12.1.2016.