Olli Liukkonen

Viestitoiminnon kehitys monialustaiseen mobiilisovellukseen

Opinnäytetyö
Tieto- ja viestintätekniikan koulutus

2019

Kaakkois-Suomen ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tekijä/Tekijät	Tutkinto	Aika
Olli Liukkonen	Insinööri (AMK)	Joulukuu 2019

Opinnäytetyön nimi

Viestitoiminnon kehitys monialustaiseen mobiilisovellukseen

48 sivua
1 liitesivu

Toimeksiantaja

Gleap Health Technologies Oy

Ohjaaja

Lehtori Niina Mässeli

Tiivistelmä

Tämän opinnäytetyön tarkoituksena oli kehittää viestitoiminto toimeksiantajan mobiilisovellukseen. Viestitoiminnon tarkoitus on toimia toimeksiantajan asiakkaiden kommunikoinnin apuna ja helpottaa asiakkaiden välistä viestintää. Työssä perehdyttiin viestitoiminnon vaatimuksiin. Tämän jälkeen viestitoiminto suunniteltiin, sekä toteutettiin vaatimusten pohjalta. Lopuksi käytiin läpi työn tuloksia ja pohdittiin toiminnon jatkokehityksen kohteita.

Tämä opinnäytetyö on kehitystyö ja se toteutettiin ketterää kehitystapaa käyttäen. Työ toteutettiin vaiheittain kehityssykleissä. Ennen jokaista kehitysvaihetta muodostettiin kehitysvaiheen suunnitelma. Kehitysvaiheessa suunnitelma toteutettiin. Kehitysvaiheen jälkeen kaikki kehitetyt toiminnot testattiin käyttäen apuna kehityksen ulkopuolista testiryhmää. Testivaiheessa tehtyjen huomioiden ja testiryhmältä saadun palautteen perusteella laadittiin seuraavan kehitysvaiheen suunnitelma.

Viestitoiminto toteutettiin käyttäen Flutter-ohjelmiston kehitystyökalua ja sen käyttämää Dart-ohjelmointikieltä. Apuna käytettiin Firebase-tietokantapalvelua ja SQLite-tietokannan hallintajärjestelmää. Viestitoiminto kehitettiin toimimaan sekä Android- että iOS-alustalla mahdollisimman samankaltaisesti.

Tämän työn tuotoksena syntyi monialustainen viestitoiminto, joka on liitettävissä joko kokonaan tai osittain toiseen sovellukseen. Joitakin viestitoiminnon alueita on vielä syytä kehittää eteenpäin, jotta niiden laatu saadaan riittävälle tasolle kaupallista levitystä varten.

Asiasanat

mobiilisovellukset, mobiililaitteet, ohjelmistokehitys, ohjelmointi, älypuhelimet, pikaviestiohjelmat

South-Eastern Finland
University of Applied Sciences

Author (authors)	Degree	Time
Olli Liukkonen	Bachelor of Engineering	December 2019

Thesis title

Development of messaging system for multi-platform mobile application

48 pages
1 page of appendices

Commissioned by

Gleap Health Technologies Oy

Supervisor

Niina Mässeli, Senior Lecturer

Abstract

The objective of this thesis was to develop a messaging feature to the commissioner's mobile application. The purpose of the messaging feature is to work as a communication tool and ease the communication between the commissioner's customers. After familiarizing with the requirements, the messaging feature was planned and implemented based on those requirements. Finally, the results were reviewed, and some objectives of further development were discussed.

This thesis was a development assignment and it was carried out using agile software development method. The process advanced in phases of development cycles. A development plan was made before each phase. The plan was executed in the development phase. After the development phase the developed features were tested by a test group outside the development process. The next development plan was designed based on the notes taken during the testing and feedback given by the test group.

The messaging feature was implemented using Flutter software development kit and Dart programming language. It uses Firebase database service and SQLite database management system. The messaging feature was developed to operate as similarly as possible on both Android and iOS platforms.

The outcome of this thesis project is a multiplatform messaging feature. The feature can be attached to another application as a whole or for some details. Some functions must be further developed to meet the quality requirements of a commercially distributed software.

Keywords

mobile application, mobile devices, software development, programming, smartphones, instant message applications

SISÄLLYS

KÄYTETYT TERMIT JA LYHENTEET

Android	Googlen kehittämä käyttöjärjestelmä, muun muassa mobiililaitteille
Build	Lähdekoodin kääntäminen koneen ymmärtämään muotoon ja sovelluksen rakentaminen sen pohjalta
Debug	Virheen etsintä ja korjaus
Getteri	Metodi, joka noutaa muuttujan arvon
GIF	Graphics Interchange Format, bittikarttagrafiikan tallennusformaatti
iOS	Applen kehittämä käyttöjärjestelmä Applen mobiililaitteille
JSON	JavaScript Object Notation, tiedostomuoto, jolla välitetään dataa attribuutti–arvo-pareina
Setteri	Metodi, joka asettaa muuttujan arvon

1 JOHDANTO

Tämä opinnäytetyö sisältää monialustaisen mobiiliapplikaation viestitoiminnon suunnittelun ja toteutuksen vaiheet. Työn tarkoituksena on esitellä mobiiliapplikaation kehitystyön vaiheita, haasteita ja tuloksia pääpiirteittäin, sekä tuottaa toimiva viestitoiminto mobiiliapplikaatioon.

Tuloksena saatava viestitoiminto tulee olla liitettävissä kokonaan tai osittain toimeksiantajan vielä nimeämättömään mobiiliapplikaatioon. Varsinainen applikaatio, johon toiminto mahdollisesti liitetään, on edelleen kehitysvaiheessa, kuten myös monet työssä käytettävistä työkaluista. Tämä opinnäytetyö on toiminnallinen opinnäytetyö ja tarkennettuna tuotekehityshanke.

Opinnäytetyön toimeksiantaja on vuonna 2016 perustettu Gleap Health Technologies Oy, toimitusjohtajanaan Teemu Penttilä. Se on osa Arctic Nutrition Finland Oy:tä ja yhdessä nämä yhtiöt työllistävät alle kaksikymmentä henkilöä Haminan toimipisteellä. Gleap Health Technologies Oy on keskittynyt ohjelmistojen suunnitteluun ja kehitykseen molempien yhtiöiden asiakkaiden käyttöön.

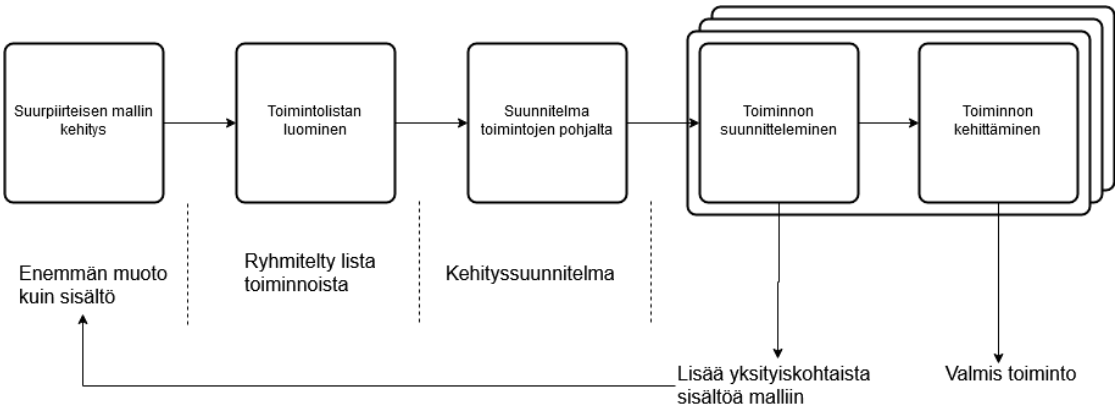
Tämän työn runko-osa sisältää aiheen taustan selvityksen, kehitystyön suunnittelun, kehitystyön toteutuksen ja lopuksi työn tulosten esittelyn ja pohdinnan. Taustan selvityksessä käydään läpi muun muassa yleisimmät käytetyt työkalut ja viestitoiminnon vaatimuksia. Suunnitteluosiossa käydään läpi koko työn kehityssuunnitelma ja pureudutaan työn eri osa-alueiden suunnitelmiin. Toteutusluvussa käydään läpi työn kehitys- ja testausvaiheet sekä kerrotaan kohdatuista haasteista. Luvussa viisi ”Tulokset ja pohdinta” käydään läpi toteutuksen onnistumista ja mahdollisia parannuksia ja korjauksia.

Työn tutkimusmenetelmäksi valikoitui ketterä kehitystapa (Agile software development), sillä sen kehityssyklin pituus ja toteutustapa sopivat käytettyyn työskentelytapaan parhaiten. Myös applikaation jatkuvasti muuttuvat tarpeet, ominaisuudet ja työkalut tukivat ketterän kehitystavan valintaa, sillä esimerkiksi Gallagher ym. (2019) mukaan perinteinen vesiputousmetodi ei sovi muutosherkkään ympäristöön.

Ketteryys kehityksessä on kyky sopeutua muutoksiin. Se on työskentelytapa, jonka päämääränä on menestyminen epävarmassa ja muutosalttiissa ympäristössä (Agile Alliance 2019).

Aluksi toteutettiin viestitoiminnon pohjasuunnitelma ja laadittiin vaadittujen toimintojen lista. Tämän jälkeen alkoi suunnitelman toteutus toiminto kerrallaan. Kun pääapplikaatiossa tapahtui muutos, tai viestitoiminnon vaatimukset muuttuivat, suunnitelman ja toteutuksen muokkaus oli helppoa. Käytetty työskentelymetodi oli sekoitus metodeista toimintokeskeinen kehitys (Feature Driven Development, FDD), jota havainnollistaa Kuva 1 ja adaptoituva ohjelmistokehitys (Adaptive Software Development, ASD), jota havainnollistaa Kuva 2. Käytettyä metodia havainnollistaa Kuva 6.

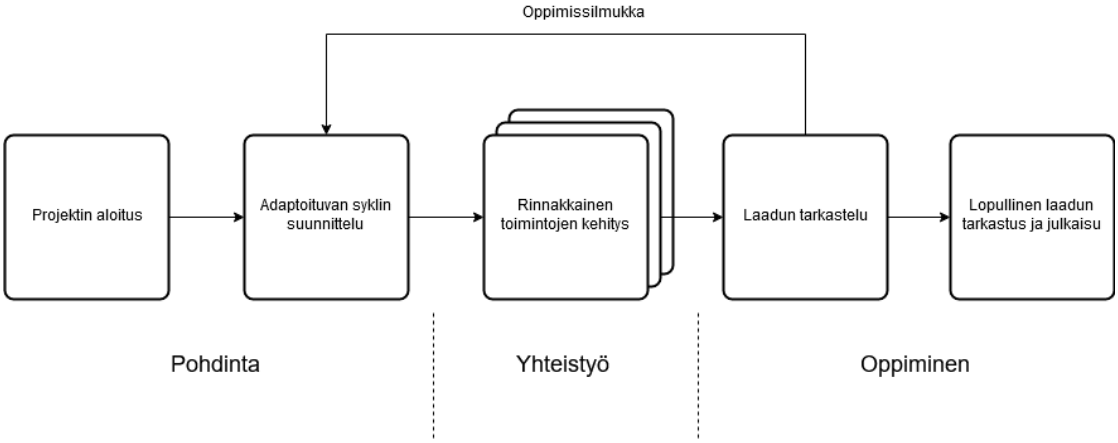
Koch (2004, 249) kertoo, että FDD aloitetaan rakennettavan systeemin suurpiirteisellä suunnitelmalla, jonka tarkoituksena ei ole kerätä kaikkia suunnitelman yksityiskohtia heti, vaan pikemminkin luoda sille pääpiirteet. Yksityiskohtia lisätään suunnitelmaan myöhemmissä kehitysvaiheissa.



Kuva 1. Toimintokeskeinen kehitys. Liukkonen 2019. Palmeria & Felsingiä 2002 mukaillen

Adaptoituva ohjelmistokehitys koostuu Kochin mukaan (2004, 233–236) viidestä osasta, joista ensimmäinen eli projektin aloitus ja viimeinen eli lopullinen laadun tarkastus ja julkaisu suoritetaan vain kerran. Väliin jäävät kolme osaa, adaptoituvan syklin suunnittelu, rinnakkainen toimintojen kehitys ja laadun tarkastelu, muodostavat nk. oppimissilmukan, joka toistetaan useasti projektin edetessä. Hän myös toteaa, että useimmat meistä luultavasti nimeäisivät metodin kolme pääkomponenttia nimillä suunnittelu, rakennus ja tarkastelu, kun

taas sen kehittäjä James Highsmith nimesi ne nimillä pohdinta, yhteistyö ja oppiminen, sillä ne olivat lähempänä hänen näkemystään siitä, kuinka monimutkaiset adaptoituvat systeemit toimivat.

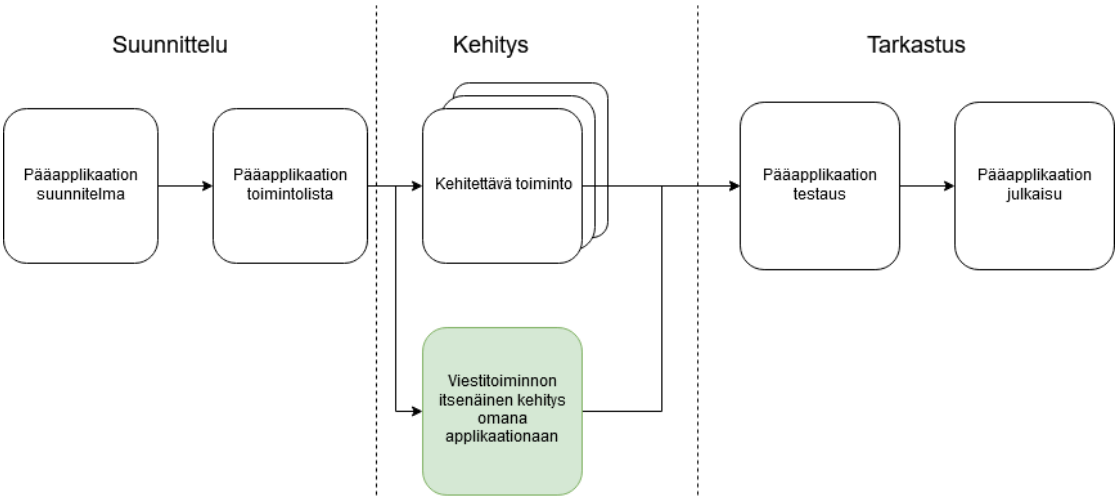


Kuva 2. Adaptoituva ohjelmistokehitys. Liukkonen 2019 Highsmithiä 2000 mukaillen

2 TAUSTA

Tässä luvussa käydään läpi tärkeimmät työssä käytetyt työkalut ja paneudutaan applikaation ja viestitoiminnon vaatimuksiin. Kuva 3 on merkitty vihreällä tämän kehitystyön sijainti suhteessa pääapplikaatioon. Tässä työssä applikaatiosta tai sovelluksesta puhuttaessa, tarkoitetaan viestitoimintoa varten kehitettyä itsenäistä alustaa. Varsinainen applikaatio, johon kehitettävä viestitoiminto mahdollisesti liitetään, kulkee tässä työssä nimellä pääapplikaatio.

Viestitoimintoa kehitetään yhteistyössä pääapplikaation kehittäjän kanssa, jotta mahdollinen liittäminen on myöhemmin mahdollista. Myös mahdollisissa kehityksen aikaisissa ongelmatilanteissa yhteistyöstä on hyötyä.



Kuva 3 Pääapplikaation kehityskaavio

2.1 Viestitoiminto

Viestitoiminnolla tarkoitetaan älypuhelimille kehitettävää pikaviestinpalvelua. Pikaviestinpalvelu mahdollistaa kahden tai useamman henkilön lähes reaaliaikaisen tekstimuotoisen kommunikoinnin Internetin välityksellä. Kommunikointi pikaviestinpalvelua käyttäen on luonteeltaan yksityisempää kuin internetin 'chat room' -kommunikointi. Se eroaa luonteeltaan myös sähköpostiviestinnästä, joka on yleensä rakenteeltaan kirjemuotoista, pikaviestin imitoidessa suullista keskustelua. (Rouse s.a.)

Yksinkertaisimmassa muodossaan pikaviestinnän tavoitteena on saavuttaa kaksi asiaa: viestintä ja läsnäolon tarkkailu. Läsnäoloa tarkkaillaan, jotta siitä voidaan tiedottaa muille käyttäjille (Larson 2016).

2.2 Työkalut

Tämän toimeksiannon alussa pääapplikaatio oli jo kehitteillä, joten siinä käytettävät työkalut oli jo osittain valittu. Tässä työssä kehitettävä viestitoiminto on luonteeltaan pääapplikaation lisäosa, joten sen tulee käyttää pääasiassa samoja työkaluja.

Tässä läpikäytävät työkalut olivat myös pääapplikaation käytössä, joten niiden käytön opiskelu ja hallinta olivat pakollisia, jotta toimintoa päästään kehittämään. Flutter, Dart, Firebase ja Android Studio ovat kaikki Googlen kehittämiä

ja toimivat erinomaisesti yhdessä. Tämä yhteensopivuus oli painoarvoltaan suurin, kun pääapplikaatiota suunniteltiin.

2.2.1 Flutter

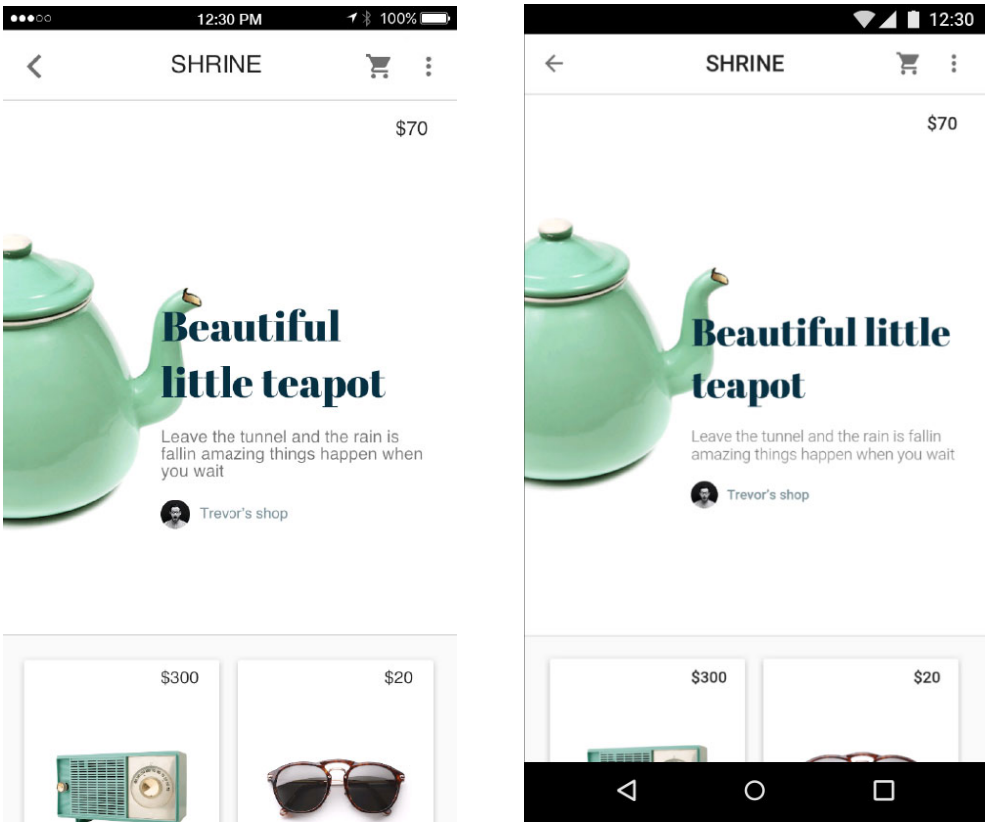
Flutter on Googlen kehittämä monialustainen käyttöliittymätyökalu kauniiden ja natiivina käännettyjen applikaatioiden työstämiseen. Se soveltuu niin mobiili-, web- kuin työpöytäkehitykseen yhdestä koodikannasta. Flutter voi käyttää jo olemassa olevaa natiivia ohjelmakoodia, se on käytössä kehittäjillä ja organisaatioilla ympäri maailman, se on ilmainen ja sillä on avoin lähdekoodi (FAQ s.a.).

Flutter on uusi ja aktiivisen kehityksen alla oleva tuote. Googlen dokumentaation mukaan (Flutter SDK releases s.a.) ensimmäinen vakaa versio (1.0.0) on julkaistu 4.12.2018 ja kirjoitushetkellä viimeisin vakaa versio (v1.9.1+hotfix.6) on julkaistu 24.10.2019. Näiden väliin mahtuu kahdeksan vakaata julkaisua ja lukuisia beta-julkaisua.

Flutter-käyttöliittymätyökalun keskeinen ajatus on niin kutsuttujen widgetien ympärillä. Koko käyttöliittymä on rakennettu yhdistellen erilaisia widgetejä, joista jokainen määrittelee tiettyä yksityiskohtaa. Näitä ovat muun muassa rakenteelliset elementit (kuten nappi tai valikko), tyylillinen elementti (kuten fontti tai väriteema), asettelun muoto (reunapehmuste) ja niin edelleen. Flutter ei käytä natiiveja widgetejä, vaan se sisältää omat valmiit widgetit, jotka näyttävät natiiveilta sekä Androidille (Material Design) että iOS applikaatioille (Cupertino). Siihen on myös mahdollista tehdä omia widgetejä (Nevercode 2019).

Flutter valikoitui työkaluksi tähän projektiin suurimmaksi osaksi sen monialustaisuuden takia. Monialustaisen kehittämisen ajatus on saada ohjelmisto toimimaan hyvin useammassa kuin yhdessä digitaalisessa ympäristössä, jolloin ohjelmistoa voidaan kaupata laajemmalle asiakasjoukolle (Techopedia s.a.). Kehitystyö nopeutui huomattavasti, kun applikaatiota voitiin kehittää samasta koodista usealle alustalle.

Kuva 4 ja Kuva 5 voi nähdä, miltä samasta koodista käännetty ohjelma näyttää eri alustoilla.



Kuva 4. iOS alusta (Technical overview. s.a.) (vasemmalla)
Kuva 5. Android alusta (Technical overview. s.a.) (oikealla)

Muita valintaan vaikuttavia tekijöitä olivat sen liitettävyys Googlen Firebase-tietokantapalveluun ja sen hot reload -toiminto, joka mahdollistaa muutosten tarkastelun simuloiduissa tai fyysisissä testilaitteissa alle sekunnissa. Flutter käyttää uutta ohjelmointikieltä nimeltä Dart. Tämä oli ainoa epäilyksiä herättävä seikka, sillä välillä uuden ohjelmointikielen opettelu saattaa olla kohtuuton vaiva ja hidastava tekijä kehityksessä.

2.2.2 Dart

Dart on Googlen kehittämä C-pohjainen ohjelmointikieli. Dart on uusi, avoimen lähdekoodin ohjelmointikieli, jolla on elinvoimainen yhteisö. Sen kehitys julkistettiin ensimmäistä kertaa 10. lokakuuta 2011 ja ensimmäinen vakaa versio julkaistiin 14. marraskuuta 2013. Se on välittömästi tutun oloinen, jos ohjelmoija on aikaisemmin käyttänyt kieliä kuten Java, C# tai JavaScript/ActionScript. Se ei ole kopio jo olemassa olevista asioista, vaan se yhdistelee eri ohjelmointikielien parhaita puolia. (Ridnjanovic ja Balbaert 2013, 9–10.)

Dart-ohjelmointikielen pääkonsepteihin kuuluu muun muassa se, että mikä tahansa minkä voi sijoittaa muuttujaan on objekti. Jokainen objekti on luokan instanssi. Kaikki objektit periytyvät Object-luokasta. Dart-ohjelmointikielessä ei myöskään ole avainsanoja ’public’, ’protected’ tai ’private’, vaan jos muuttujan nimi alkaa alaviiva-merkillä (_), se on näkyvissä vain omassa kirjastossaan. Muuttujan nimi voi alkaa kirjaimella tai alaviivalla ja sitä voi seurata mikä tahansa näiden merkkien ja numeroiden yhdistelmä. (A tour of the Dart language s.a.)

Ohjelmointikielen valintaan ei voida vaikuttaa, sillä kehitystyökaluksi valittu Flutter käyttää vain Dart-ohjelmointikieltä.

2.2.3 Firebase

Firebase on applikaatioiden kehittämisalustan taustajärjestelmäpalvelu – Backend-as-a-Service eli BaaS. Se tarjoaa ylläpidettyjä taustajärjestelmäpalveluita, kuten reaaliaikaisen tietokannan, tiedostojen tallentamisen pilvipalvelun, todentamispalvelun, kaatumisraportoinnin, koneoppimisen, etäkonfiguroinnin ja ylläpidon staattisille tiedostoille (Firebase s.a.).

Lukuisista tarjolla olevista palveluista tässä työssä käyttöön otetaan todennus-, pilvi-, viestintä- ja tietokantapalvelut. Nämä palvelut ovat välttämättömiä viestitoiminnon kehittämiseksi annettujen vaatimusten pohjalta.

Firebase-tietokannan käyttö on ilmaista seuraavin rajoituksin:

- 1 GB tallennettua dataa
- 10 GB tiedonsiirtoa / kk
- 20 000 datan kirjoituskertaa päivässä
- 50 000 datan lukukertaa päivässä
- 20 000 datan poistamiskertaa päivässä

(Pricing plans s.a.).

Koska viestitoiminto on muusta applikaatiosta erikseen kehitettävä osuus, käytettävissä on myös erillinen ilmainen Firebase-tili sitä varten. Tämä mahdollistaa viestitoiminnon tietoliikenteen tarkan seurannan ja optimoinnin.

2.2.4 Android studio

Android Studio on Android-applikaatioiden virallinen integroitu kehitysympäristö (IDE), joka perustuu IntelliJ IDEA:an (Meet Android Studio s.a.). Opinnäytetyössä ainoa varteenotettava vaihtoehto Android Studion koodieditorille on Visual Studio Code, joka ei kuitenkaan yllä ominaisuuksiltaan samalle tasolle.

Koska Android Studio tukee monialustaista ohjelmistokehitystä, pystytään Android Studiota käyttämään myös iOS-ohjelmiston kehitykseen (Harris 2015).

Android Studion valintaan vaikuttavia tekijöitä ovat muun muassa:

- GIT-versionhallintaohjelmisto, joka mahdollistaa työskentelyn jatkamisen eri työpisteillä, sekä versiohistoriassa takaisin palaamisen.
- Laitesimulointi. Tämä ominaisuus simuloi haluttua Android-laitetta suoraan työaseman näytöllä. Fyysistä testilaitetta ei siis välttämättä tarvita. Kehitettävää ohjelmistoa päästään myös näin ollen testaamaan eri käyttöjärjestelmäversioissa.
- Tehokas Lint, joka analysoi lähdekoodin ja löytää virheet, bugit, tyylilliset ongelmat ja huonot rakenteet.
- Laaja valikoima selkeitä Debug-työkaluja.

2.2.5 Xcode

Xcode on Applen julkaisema kehitysympäristö. Xcode-kehitysympäristöä käytetään sovellusten kehittämiseen Applen laitteille kuten iPad, iPhone, Apple Watch, Apple TV ja Mac. Siinä on työkalut koko kehitystyökokonaisuuden hallitsemiseksi aina sovelluksen luomisesta, testauksesta ja optimoinnista, sen julkaisemiseen App Store -sovelluskaupassa. (Welcome to Xcode 2019.)

Tässä työssä Xcode-kehitysympäristön käyttö rajoittuu vain sen välttämättömiin toimintoihin, kuten kohdelaitteen ja -käyttöjärjestelmän asettamiseen, varmenteen asettamiseen ja sovelluksen buildaamiseen iOS-laitteille.

2.3 Applikaation ja viestitoiminnon vaatimukset

Ensimmäinen ja tärkein vaatimus kehitystyölle on turvallisen kehittämisen varmistaminen. Kehittääkseen ja testatakseen viestitoimintoa ja sen osa-alueita

turvallisesti, täytyy tehdä erillinen applikaatio eli ns. hiekkalaatikko, jotta mahdolliset virhetilanteet eivät vaaranna varsinaisen pääapplikaation kehitystä.

Tämän jälkeen täytyy varmistaa mahdollisimman optimoitu tietoliikenne tietokannan ja sovelluksen välillä, jotta tietokannan maksukynnys ei ylity ja kehitystyö voi jatkua keskeytyksettä. Kehitystyölle ei ole varattu varoja, joita voidaan käyttää tietokannan kustannuksiin.

Viestitoiminnon tulee hyödyntää kirjautumista. Kaikkien viestitoiminnon testikäyttäjien tulee olla rekisteröityjä, sillä pääapplikaation käyttäjät ovat valmiiksi rekisteröityjä ja kirjautuneita. Näin päästään simuloimaan lopullista käyttötilannetta jo testausvaiheessa.

Viimeiseksi viestitoiminnon tulee toimia sekä Android- että iOS-käyttöjärjestelmillä mahdollisimman samankaltaisesti. Tämä helpottaa käyttäjien opastusta toiminnon käytössä ja nopeuttaa mahdollisia virhetilanteiden selvittämistä.

3 SUUNNITTELU

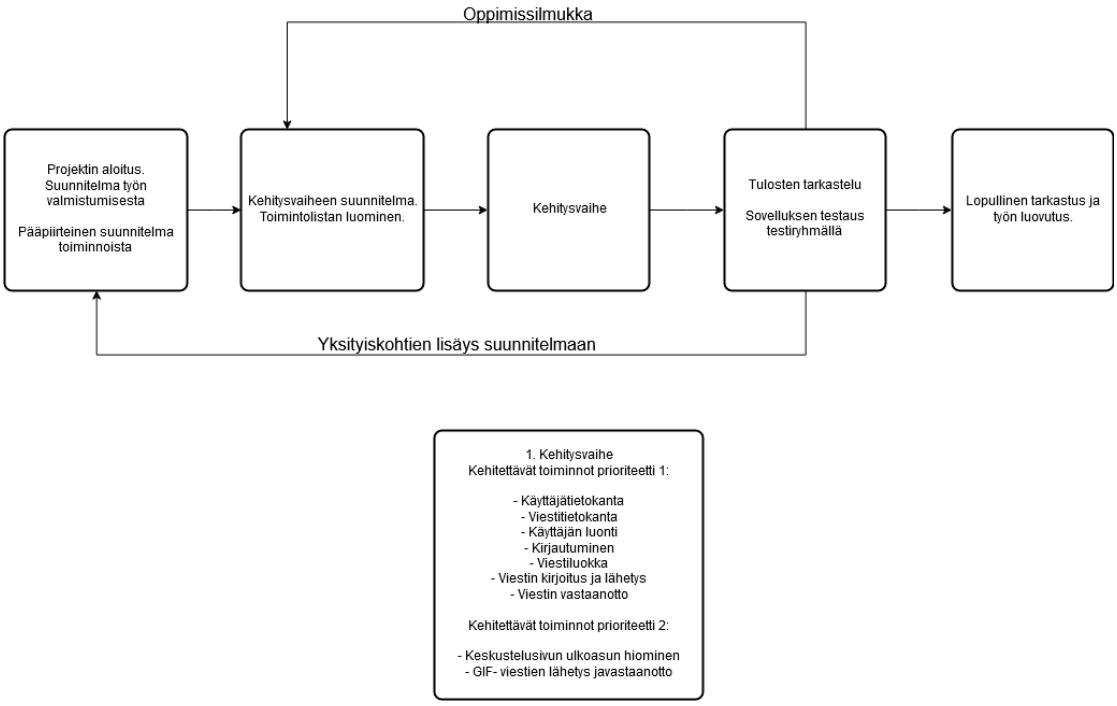
Tässä luvussa käydään läpi sovelluksen ja toimintojen suunnittelun vaiheita ja perustellaan, miksi kyseisiin ratkaisuihin päädyttiin. Kehitystavan valinnasta johtuen suunnitteluvaiheeseen palataan toistuvasti kehitystyön edetessä ja alkuperäistä suunnitelmaa täydennetään.

Työkalujen ja aihepiirin ollessa tuntemattomia päätetään edetä vaiheittain ja lisätä suunnitelmaan toimintoja ja ominaisuuksia sitä mukaan, kun tiedot ja taidot karttuvat. Näin vältytään turhalta työltä ja aikataulun venymiseltä.

3.1 Applikaation ja viestitoiminnon kehityssuunnitelma

Kehitystyön tehostamiseksi priorisoidaan toimintoja niiden tärkeyden mukaan. Prioriteetilla 1 merkityt toiminnot ovat kunkin kehitysvaiheen tärkeimmät ja näihin keskitytään ensimmäisenä. Kun ne ovat valmiissa tai testaamista vaille olevassa tilassa, siirrytään prioriteetilla 2 merkittyihin toimintoihin. Näiden jälkeen applikaatio luovutetaan testiryhmälle testattavaksi.

Kuva 6 on kaavio viestitoiminnon kehityssyklistä ja ensimmäisessä kehitysvaiheessa kehitettävistä toiminnoista prioriteetteineen. Ensimmäinen kehitysvaihe ja sen toiminnot suunnitellaan ennen kehitystyön aloittamista. Seuraavien kehitysvaiheiden toiminnot muodostuvat testiryhmältä saadun palautteen sekä edellisestä kehitysvaiheesta kerättyjen huomioiden perusteella tulosten tarkasteluvaiheessa.



Kuva 6. Kehityskaavio

3.2 Tietokannat

Firebase-tietokantapalvelun etu normaaliin tietokantaan nähden on se, että mittavaan tietokantasuunnitteluun ei ole tarvetta. Käytettäessä Firebase-tietokantapalvelua hyvin yksinkertainen perussuunnitelma on riittävä. Firebase-palveluun voi lisätä suoraan käyttäjiä ja näiden attribuutteja, näin ollen tietokannan haltijan ei tarvitse luoda käyttäjätaulua erikseen. Palvelun helppokäyttöisyys ja liitettävyys Flutter-työkalulla kehitettyihin applikaatioihin olivat ratkaisevat tekijät tietokantaa valittaessa.

Yksinkertaisessa viestitoiminnossa tarvittavia tietokantatauluja ovat normaalisti käyttäjät ja itse viestit. Kuvia lähetettäessä ja vastaanotettaessa tulee kuville luoda erillinen kuvatietokanta. Firebase-tietokannassa ei kuitenkaan ole tauluja, kuten perinteisissä tietokannoissa. De Croos (2018) kuvailee artikkelissaan Firebase-tietokantaa pysyväksi datarakenteeksi pilvessä. Hän vertaa sitä JSON-objektiin, johon voidaan tallentaa, tai josta voidaan hakea arvoja käyttäen data-avainta. De Croos kertoo Firebase-tietokannan soveltuvan loistavasti reaaliaikaiseen keskustelu- ja viestisovellukseen, jossa reagoidaan datan lisäämiseen tai päivittämiseen.

Pääapplikaation käyttäjät haetaan erillisestä toimeksiantajan asiakastietokannasta. Tässä työssä viestiapplikaation käyttäjät luodaan itse, eikä niitä haeta ulkopuolelta. Jokainen testaaja luo itse käyttäjätilinsä tai vaihtoehtoisesti käyttää Google-tiliään kirjautumiseen, jolloin Google-käyttäjä lisätään käyttäjätileihin.

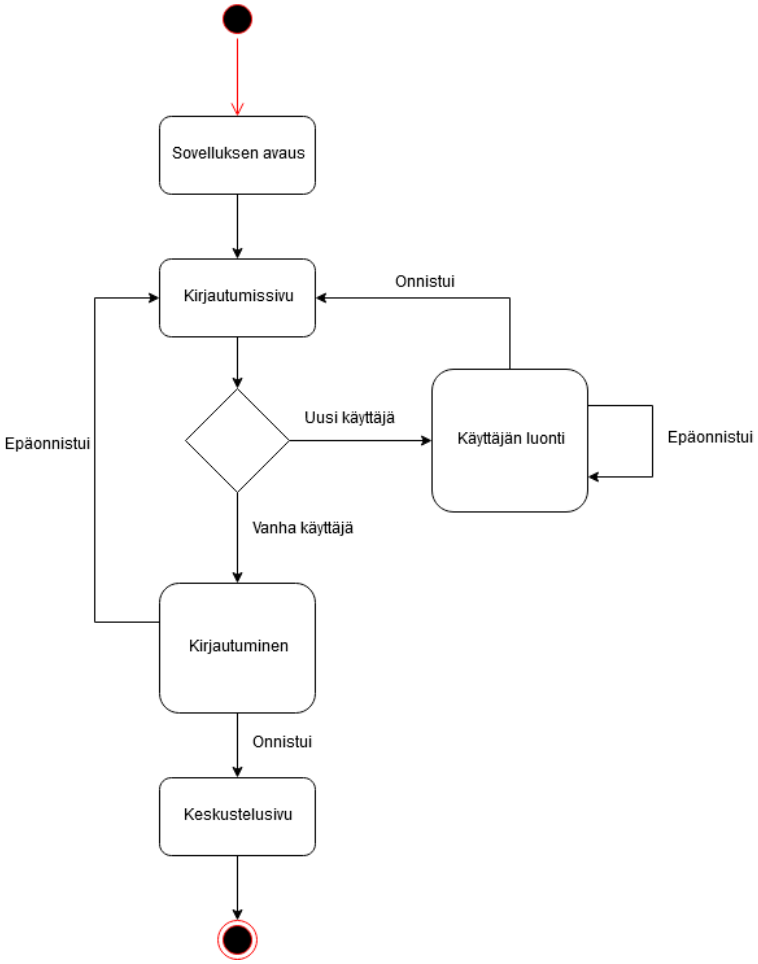
Sovelluksen ensimmäisen testivaiheen jälkeen otettiin käyttöön paikallinen viestitietokanta, jonka tarkoituksena on optimoida tietoliikenteen määrä minimiin. Paikallisen tietokannan suunnitelma ja toteutus löytyvät luvusta 4.1.3.

3.3 Käyttäjän luominen ja kirjautuminen

Lopullinen käyttäjän luominen ja kirjautuminen sovellukseen tapahtuu pääapplikaatiossa ja siihen kehitetyissä toiminnoissa. Jotta kehitettävää viestitoimintoa päästään testaamaan, on sitä varten kuitenkin kehitettävä oma käyttäjän luomis- ja kirjautumistoiminto. Näiden toimintojen kehitys jätetään kuitenkin minimiin.

Opinnäytetyössä kehitettävään applikaatioon tehdään mahdollisuus luoda oma käyttäjä, jolla on sähköpostiosoite, käyttäjänimi sekä salasana. Kirjautumisvaihtoehdot ovat joko itse luodulla käyttäjällä tai Google-tilillä kirjautuminen. Google-tilillä kirjauduttaessa käyttäjän käyttäjänimi muodostetaan Google-tilin nimestä. Käyttäjän kirjautuessa ensimmäistä kertaa, Firebase muodostaa käyttäjälle tilin, joka linkitetään käyttäjän tietoihin tai todentamisen suorittaneen palvelun välittämään informaatioon (Authenticate Using Google Sign-In on Android s.a.).

Kun käyttäjä käynnistää sovelluksen, avautuu ruudulle kirjautumisvalintasivu. Jos käyttäjä valitsee kirjautumisen Google-tilillä, kirjaudutaan palveluun käyttäjän aikaisemmin luodulla Google-tilillä. Käyttäjän valitessa kirjautumisen itse luodulla käyttäjätilillä ohjataan käyttäjä kirjautumissivulle. Tämän jälkeen käyttäjä joko luo uuden käyttäjän tai kirjautuu jo olemassa olevalla käyttäjällä sovellukseen. Tämä toiminta on visualisoitu Kuva 7.

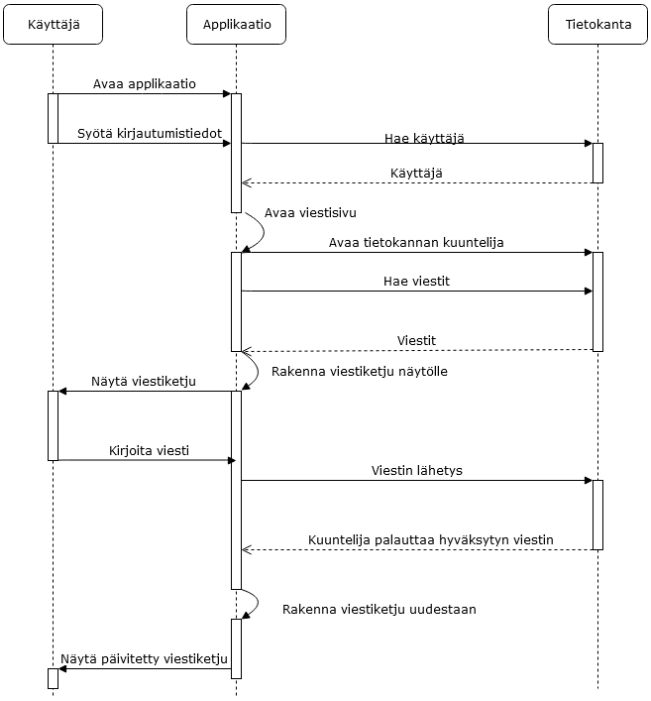


Kuva 7 Kirjautumisen vuokaavio

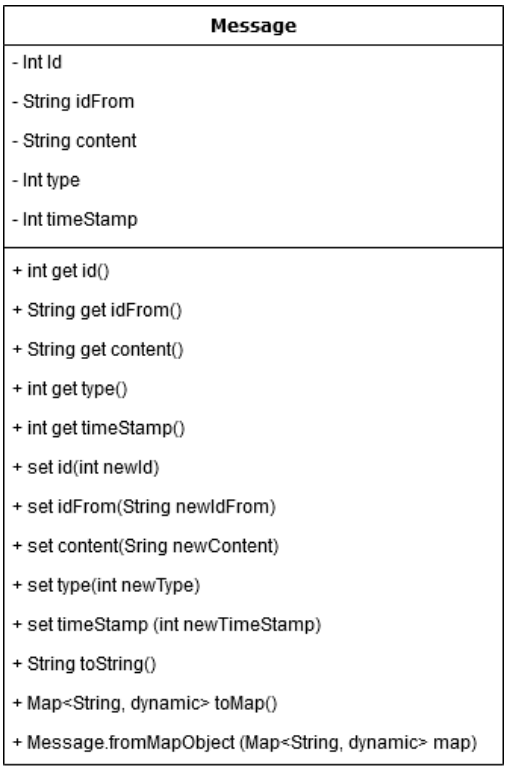
Käyttäjän tulee kirjautua sovellukseen joka kerta kun sovellus avataan. Kirjautumisesta tulee kuitenkin tehdä nopea tapahtuma, jotta se ei heikennä käyttökokemusta. Sovelluksesta tulee myös pystyä kirjautumaan ulos yhdellä napin painalluksella, jotta käytettävää käyttäjätiliä voidaan vaihtaa.

3.4 Viestin lähetys ja vastaanotto

Kun käyttäjä on kirjautunut sovellukseen, sovellus aukaisee keskustelusivun. Kun keskustelusivu aukeaa, kirjataan sovellus kuuntelemaan tietokantaa, jotta saadaan tieto uusista viesteistä. Tämän jälkeen haetaan viestit ja näytetään ne ruudulla. Käyttäjän kirjoitettua viestinsä hänen tulee painaa lähetys symbolia, jolloin viesti lähetetään tietokantaan. Kun viesti on lisätty onnistuneesti tietokantaan, se lisätään näytölle. Viestin lähetyksen toiminnan voi nähdä Kuva 8 sekvenssikaaviossa. Lähetettävä viesti tallennetaan viestiluokan olioksi, jonka suunnitelman voi nähdä Kuva 9.



Kuva 8 Viestin lähetyksen sekvenssikaavio



Kuva 9 Viestiluokan suunnitelma

Viestiluokan muuttujat:

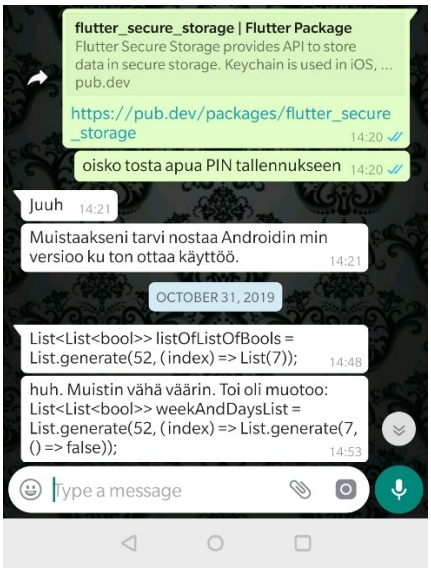
- Id, kokonaislukumuotoinen tunniste. Yksilöllinen jokaiselle viestille.
- idFrom, merkkijono. Pitää sisällään viestin lähettäjän.
- type, kokonaisluku. Viestin tyyppi. Teksti, kuva tai GIF-kuva.
- timeStamp, kokonaisluku. Viestin lähetyksen aikaleima.

Viestiluokan metodit:

- Getterit. Jokaisella muuttujalla on oma getterinsä, joka palauttaa kyseisen muuttujan arvon.
- Setterit. Jokaisella muuttujalla on oma setterinsä, jolla voidaan asettaa kyseisen muuttujan arvo.
- Map-metodit, joilla viestiluokan olio voidaan muuttaa map-objektimuotoon tai map-objektimuodosta viestiluokan olioksi.

3.5 Käyttöliittymä

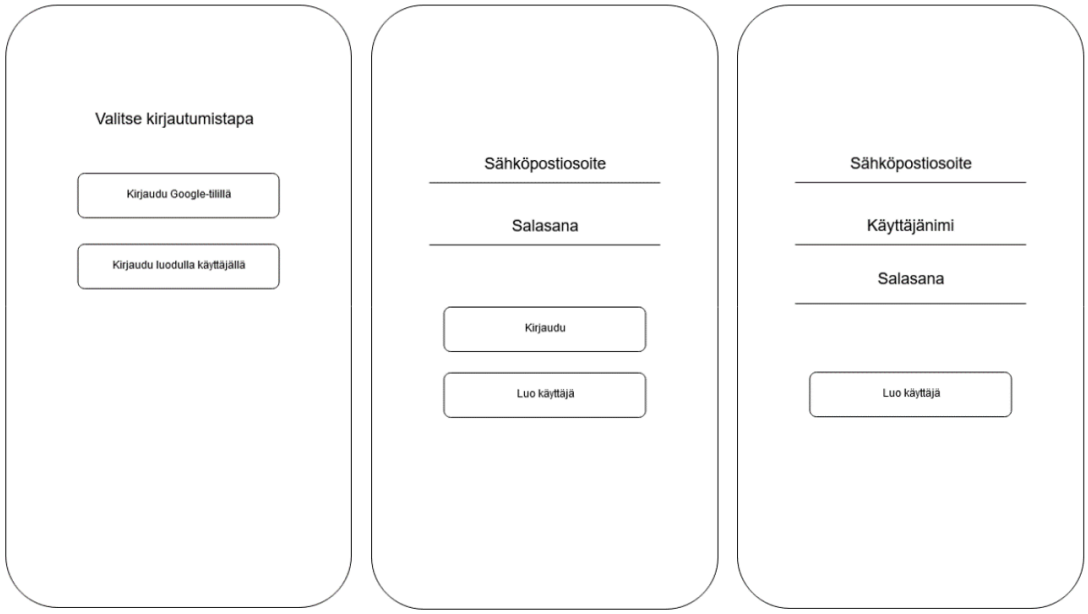
Keskustelunäkymän käyttöliittymäsuunnittelussa käytetään apuna perinteistä viestisovelluksen asettelua, jonka voi nähdä Kuva 10. Tässä omat viestit sijoitetaan ruudun oikeaan laitaan ja muiden keskusteluun osallistujien lähettämät viestit sijoitetaan ruudun vasempaan laitaan. Sen lisäksi viestit näytetään vanhimmasta uusimpaan siten, että uusin on alimaisena.



Kuva 10 WhatsApp keskustelusovelluksen asettelu

Käyttöliittymässä tullaan käyttämään värejä käyttäjän informoimiseen näkymän eri osa-alueista. Tausta tulee olemaan hyvin tumma. Viestikuplat ja niiden sisällöt tullaan erottelemaan hillityin värein toisistaan. Värien käytöllä tähdätään käyttöliittymän intuitiivisuuteen ja helppokäyttöisyyteen. Oikeat värivalinnat tukevat informaation luettavuutta ja lisäävät käytettävyyttä eli parantavat käyttökokemusta (Cherryhappygirl 2018).

Muiden näkymien wireframe- eli rautalankamallit ovat alla Kuva 11. Vasemmalla kirjautumistavan valintanäkymä, keskellä kirjautumisnäkymä ja oikealla käyttäjän luontinäkymä. Koska näitä toimintoja ei tulla käyttämään mihinkään muuhun, kuin testikäyttäjien luontiin ja kirjautumiseen, jätetään nämä toiminnot ajan säästämiseksi visuaalisesti yksinkertaisiksi ja niiden toiminnallisuus pelkistetyksi.



Kuva 11 Kirjautumisen rautalankamallit

3.6 Käyttökokemus

Käyttökokemuksen suunnittelussa kaiken keskiössä on tunne, jota käyttäjä tuntee käyttäessään palvelua. Käyttökokemusta suunnitellessa keskitytään siihen, kuinka palvelun käyttö saadaan käyttäjälle mahdollisimman luontevaksi ja helpoksi. (Virtanen 2016.)

Käyttökokemuksen parantamiseksi sovelluksessa otetaan käyttöön visuaalista palautetta antavia elementtejä. Näitä ovat esimerkiksi kehitysindikaattori, joka viestittää käyttäjälle jonkin prosessin olevan käynnissä, sekä käyttäjäpohjaisten virheiden ilmoitus tekstinä näytöllä. Nämä elementit selkeyttävät applikaation käyttöä ja pitävät käyttäjän tietoisena sovelluksen tilasta. Näin ollen vältytään negatiivisilta tunteilta, kuten epätietoisuudelta ja epävarmuudelta, sekä vähennetään väärin ymmärryksen riskiä. Järjestelmän tilan näkyvyys on yksi tärkeimmistä tekijöistä käyttökokemuksen suunnittelussa (Babich 2018).

3.7 Keskusteluryhmät

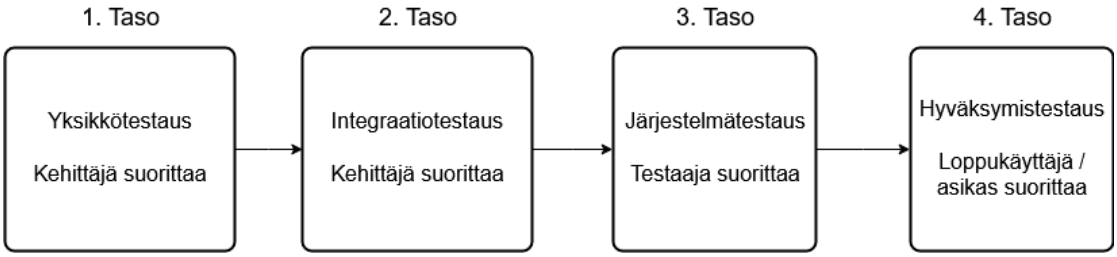
Keskustelutoiminnon päätarkoituksena on toimia toimeksiantajan asiakkaiden työkaluna. Asiakkaat muodostavat tiimejä. Jokainen asiakas kuuluu johonkin tiimiin. Asiakkaan oma tiimi on kirjattu asiakastietokantaan. Jokainen tiimi muodostaa oman keskusteluryhmänsä, eikä muiden tiimien keskusteluja pääse lukemaan. Kun asiakas asentaa applikaation ja kirjautuu siihen tunnuksillaan, tarkistetaan hänen tiiminsä ja kirjataan sovellus kyseisen tiimin keskusteluryhmään. Asiakas ei siis itse pysty valitsemaan keskusteluryhmäänsä.

Kehityksen ja testauksen aikana testiryhmän kaikki jäsenet kirjataan samaan keskusteluryhmään. Täten viestitoimintoa saadaan testattua oikealla ryhmäkoolla ja testiin saadaan luotua mahdollisimman aito käyttökokemus.

3.8 Testaus

Puhuttaessa testiryhmän testauksesta tässä työssä tarkoitetaan käyttäjätestausta. Työssä tehtävien käyttäjätestien tarkoituksena on kerätä testikäyttäjien kokemuksia applikaation käytöstä ja paikallistaa mahdollisia virhetilanteita ja ongelmakohtia. Käyttäjätestien taso vaihtelee järjestelmätestauksesta hyväksymistestaukseen. Näiden lisäksi testataan myös käytettävyyttä. Testien tasot ovat nähtävissä Kuva 12.

Järjestelmätestauksessa testataan kokonaista integroitua sovellusta ja sen tarkoituksena on testata kuinka hyvin sovellus noudattaa annettuja vaatimuksia. Hyväksymistestauksessa testataan täyttääkö sovellus yrityksen antamat vaatimukset ja arvioidaan, onko se valmis julkaistavaksi. (ProfessionalQA 2019.)



Kuva 12 Testauksen tasot

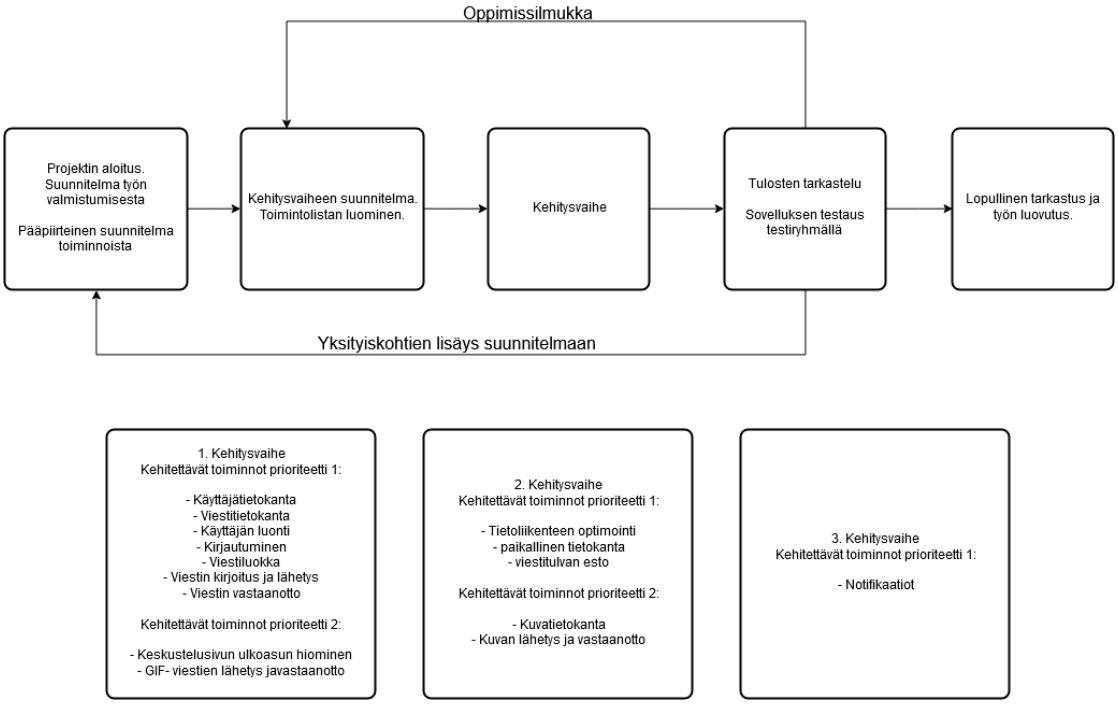
Kehitysvaiheessa applikaatio on jatkuvan alempien tasojen testauksen, eli yksikkötestauksen ja integraatiotestauksen alaisena, kehittäjän toimesta. Jokaista uutta ominaisuutta testataan toistuvasti sekä Android-, että iOS-alustalla sen toimivuuden takaamiseksi.

Jokaisen kehitysvaiheen päätyttyä applikaatio annetaan testiryhmän testattavaksi. Applikaation uusin versio ladataan Microsoftin OneDrive-palveluun, ja latauslinkki lähetetään testiryhmälle.

Testiryhmä koostuu applikaation kehittäjän lisäksi neljästä ohjelmointiin suuntautuneesta tieto- ja viestintätekniikan neljännen vuoden opiskelijasta, joilla on suoritettuna pelitestauksen 1. ja 2. kurssit. Lisäksi ryhmässä on yksi tietoverkkotekniikkaan suuntautunut neljännen vuoden opiskelija. Testaus on vapaamuotoista ja palaute kerätään suullisesti. Testauksen suuntaviivoina käytetään Foggian artikkelin (2018) 2. vaiheen seitsemän kohtaista muistilistaa. Testaajilta ei täten kysytä mielipiteitä tarkoilla kysymyksillä, sillä se olisi johdattelua ja saattaisi muokata testaajan näkemystä. Objektiivisuuden ja neutraalin suhtautumisen säilyttämiseksi testiryhmän jäsenten annetaan kertoa itse kokemuksistaan ja mielipiteistään. Huomiot ja palaute kirjataan ylös ja niistä muodostetaan seuraavan kehitysvaiheen suunnitelma.

4 TOTEUTUS

Työn toteutus tapahtui vaiheittain. Ensimmäinen kehitysvaihe suunniteltiin ennen toteutuksen aloittamista ja 2. ja 3. vaihe suunniteltiin toteutuneiden vaiheiden ja testitulosten ja palautteen pohjalta. Kuva 13 kaavio kuvaa työn vaiheita ja siihen on listattu eri kehitysvaiheet prioriteetteineen. Tässä luvussa käydään läpi eri kehitysvaiheet ja niitä seuranneet testivaiheet ja pureudutaan kehitettyihin toimintoihin.



Kuva 13 Kehityskaavio kehitysvaiheineen

Kun kehitysvaihe saatiin päätökseen, siirryttiin testausvaiheeseen, jossa testattiin kehitetyt toiminnot ja niiden toimivuus. Pääpaino testauksessa oli edellisessä kehitysvaiheessa tehdyissä toiminnoissa.

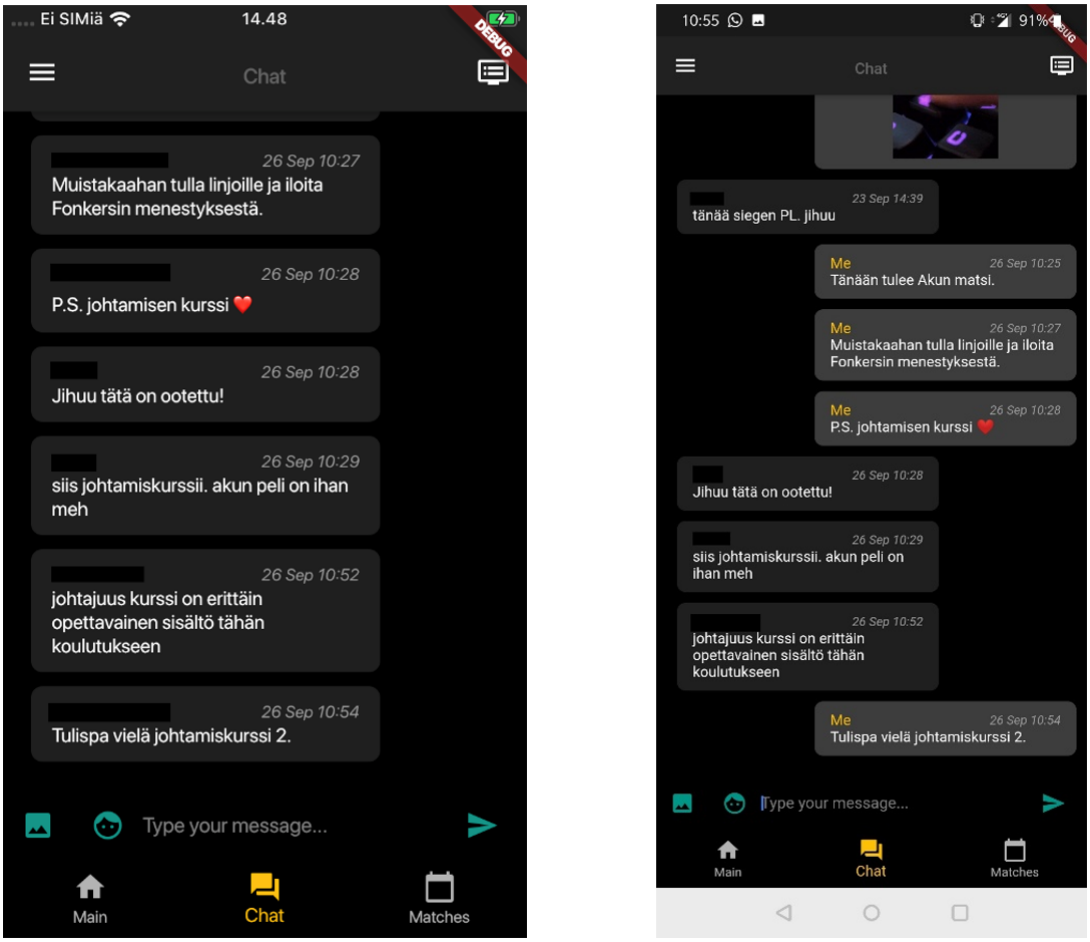
4.1 Ensimmäinen kehitysvaihe

Ensimmäinen kehitysvaihe oli laajin monesta syystä. Siinä oli eniten kehitettäviä toimintoja ja useat näistä toiminnoista olivat kytköksissä toisiinsa, joten niitä täytyi kehittää samanaikaisesti. Kehitysvaiheen aluksi täytyi myös tutustua Flutter-käyttöliittymätyökaluun ja sen käyttämään Dart-ohjelmointikieleen,

jotta toimintojen kehitys ylipäänsä olisi mahdollista. Työkalujen käytön opiskelu ja kehitysvaiheen valmistelut kestivät yhteensä useamman viikon.

4.1.1 Keskustelutoiminnon käyttöliittymä

Keskustelun käyttöliittymä tehtiin suunnitelman mukaan mahdollisimman samankaltaiseksi molemmille alustoille. Käyttöliittymien vertailukuvassa (Kuva 14) on sama keskustelu kuvattuna eri puhelimilla. Sovellukseen ovat kirjautuneet eri puhelimilla eri käyttäjät. Tämä selviää viestien asettelusta. Kuva 14 vasemmassa laidan viestiketjussa nähdään, että ruudulla ei näy ainuttakaan kyseisen käyttäjän viestiä, kun taas oikean laidan viestiketjussa erottuu kirjautuneen käyttäjän viisi omaa viestiä.



Kuva 14 Viestiketjuvertailu alustoittain. Vasemmalla iOS- ja oikealla Android-alustalla.

Käyttöliittymän mallina käytettiin erilaisista viestisovelluksista tuttua asettelua, jossa omat viestit tulevat ruudun oikeaan laitaan ja muiden viestit ruudun vasempaan laitaan. Tämän lisäksi viestit kasautuvat vanhimmasta uusimpaan, siten, että uusin on alimaisena.

Viestikuplan leveys on vakio, riippumatta viestin pituudesta. Se sisältää lähettäjän, lähetysajan ja viestin sisällön. Jokainen osa on eroteltu eri värillä. Viestikupla erottuu taustasta harmaana. Käyttäjän oman viestin kupla on hieman vaaleampi kuin muiden käyttäjien viestikuplat. Viestikuplan vasempaan yläreunaan asetetaan viestin lähettäjän nimi syaanin värisellä fontilla. Käyttäjän omissa viesteissä lukee keltaisella fontilla englannin kielinen teksti ”Me”, eli minä. Kuplan oikeaan yläreunaan merkitään viestin lähetysajankohta harmaalla fontilla. Ajankohtaan merkitään päivämäärä, kuukausi ja kellonaika minuutin tarkkuudella.

Kun käyttäjä painaa tekstistä ”Type your message…”, aukeaa ruudun alareunaan näppäimistö. Tämän jälkeen käyttäjä voi kirjoittaa haluamansa viestin ja lähettää sen ryhmälle. Jos muilla ryhmän jäsenillä on samaan aikaan keskusteluikkunoita auki, uusi viesti ilmestyy ruudulle lähes reaaliajassa.

4.1.2 Tietokannat

Firebase-tietokannan luonti tapahtui Internet-selaimen kautta. Luonnin jälkeen oli linkitettävä tietokanta ja kehitettävä applikaatio toisiinsa. Linkitys tapahtuu applikaation alustakohtaisella tunnisteella, joka rekisteröidään Firebase-tietokantaan. (Add Firebase to your Flutter app s.a.)

Firebase-tietokannan luonnin jälkeen asetettiin tietokannan turvallisuussäännöt. Firebasen turvallisuussääntöjä käytetään suojaamaan dataa vahingollisilta käyttäjiltä. Säännöt voidaan kirjoittaa yksinkertaisiksi tai monimutkaisiksi sovelluksen vaatiman turvallisuustason mukaan. (Firebase Security Rules s.a.)

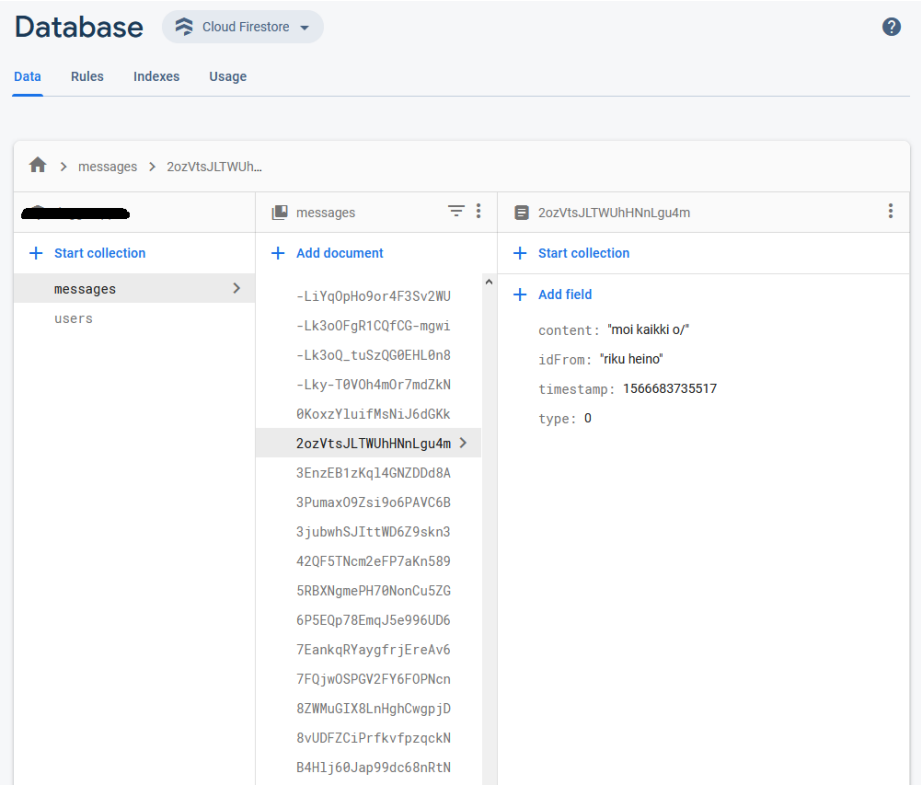
Kuva 15 on tietokannan dokumentteja koskeva sääntö. Säännössä annetaan kirjautuneille käyttäjille oikeus lukea ja kirjoittaa dokumentteja. Tämän jälkeen

käyttäjät pystyivät lähettämään viestejä tietokantaan ja lukemaan muiden lähettämiä viestejä.

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if request.auth.uid != null;
    }
  }
}
```

Kuva 15 Firebase-tietokannan säännöt

Sääntöjen asetuksen jälkeen tietokantaan luotiin messages-kokoelma, johon tallentuvat kaikki testiryhmän viestit. Kuva 16 nähdään tietokannan kokoelma ja sen sisältämiä viestejä. Selainpohjaisesta konsolista päästään myös tarkastelemaan yksittäisiä viestejä ja niiden sisältämiä muuttujia. Jokaisella viestillä on oma yksilöllinen tunnus, jonka tietokanta luo automaattisesti, kun viesti lisätään kokoelmaan (Add data to Cloud Firestore s.a.).

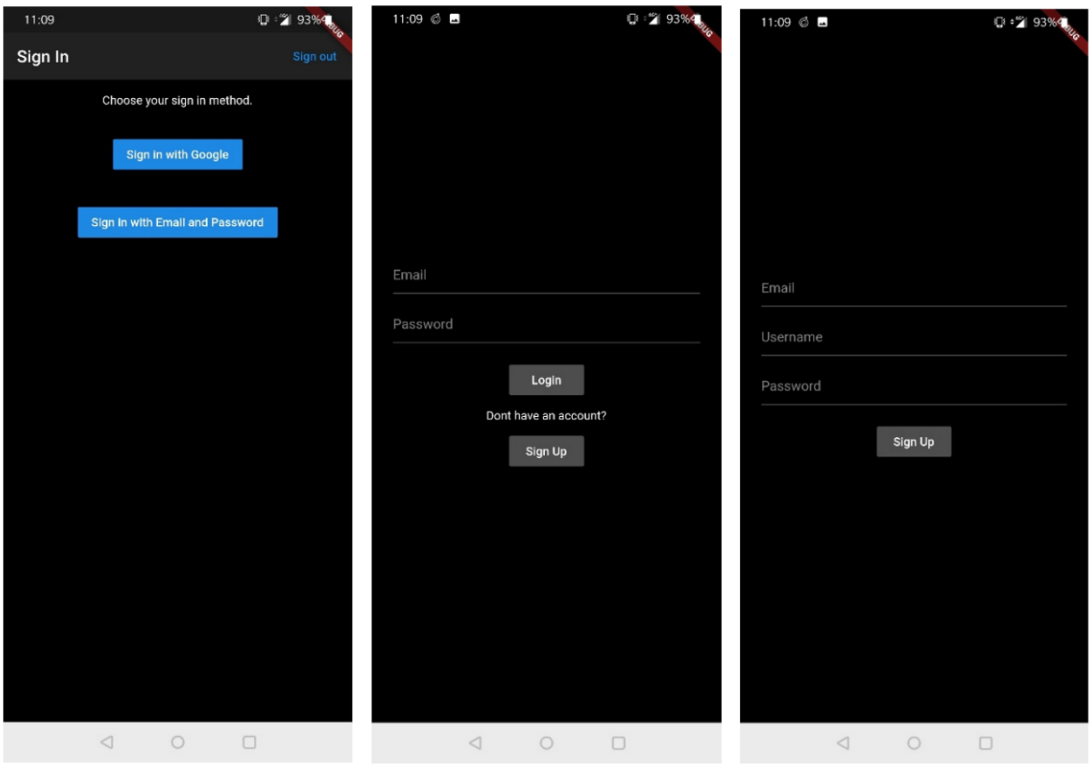


Kuva 16 Firebase-tietokannan viestikokoelma

4.1.3 Kirjautuminen ja käyttäjän luonti

Kuten jo aiemmin todettiin, käyttäjien luonti on vain sovelluksen testausta varten luotu ominaisuus, joten esimerkiksi sähköpostiosoitteiden olemassaoloa ei todennettu varmistussähköposteilla. Käyttäjätilien luonnista saatiin näin ollen nopea ja vaivaton operaatio. Tilien luomisen syynä oli saada viesteille eri lähettäjiä, jotta voidaan tarkastella, kuinka käyttöliittymä ja tietokanta käyttäytyvät.

Kuva 11 ovat wireframe-mallit kirjautumiseen ja käyttäjän luontiin tarvittaville sivuille. Kuva 17 on ruutukaappauksia kyseisistä sivuista valmiina.



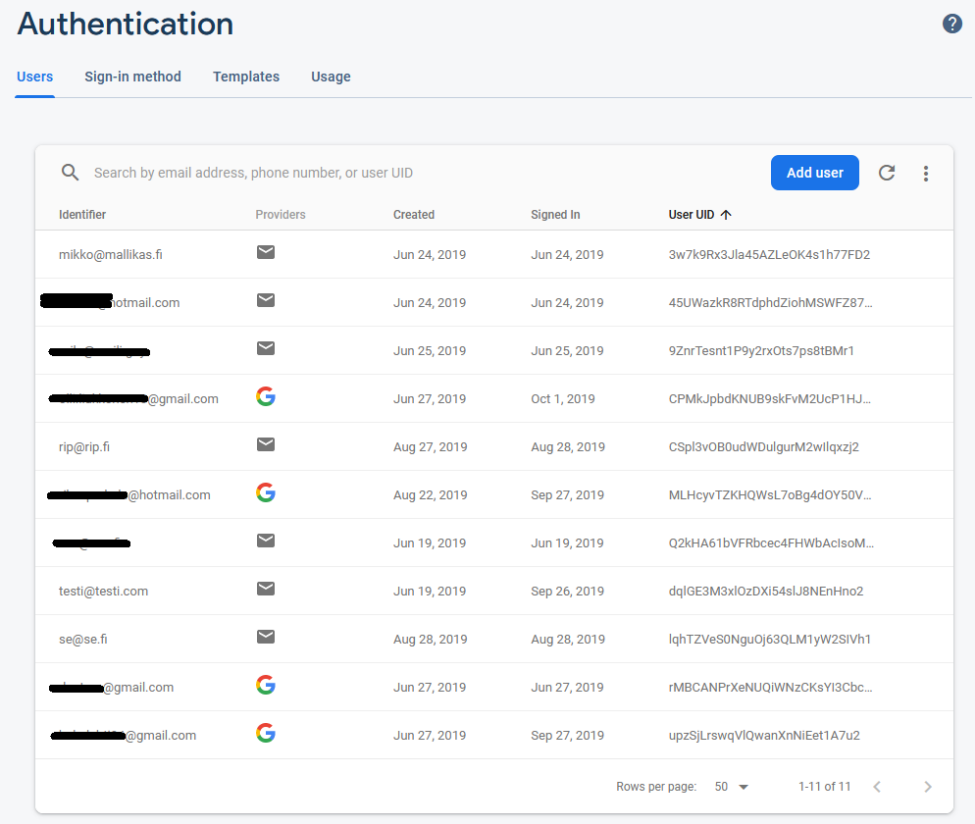
Kuva 17 Kirjautumis- ja käyttäjänluontinäkymät Android-alustalla

Testikäyttäjät pystyvät luomaan omia käyttäjätilejä tai kirjautumaan henkilökohtaisella Google-tilillään. Kehitysvaiheessa käyttäjätilejä luotiin siis enemmän kuin käyttäjiä oikeasti oli.

Kirjautumisen nopeuttamiseksi sovellus muistaa edellisen kirjautumisen tiedot ja käyttää niitä, mikäli käyttäjä valitsee kirjautumisen itse luodulla käyttäjällä. Google-tiliä käytettäessä käyttäjän laite ja Google-varmennepalvelu pitävät

huolen käyttäjätilin muistamisesta. Näin kirjautumisesta saadaan nopea toimenpide, joka ei heikennä käyttökokemusta.

Kuva 18 on tietokannan hallinnointikonsolin todennusvälilehdeltä. Siinä on lista sovelluksen käyttäjistä, joista jokaisen tilin tyypin voi nähdä sarakkeesta ”Providers”. Harmaa kirjekuori kuvaa sähköpostiosoitteen ja salasanan perusteella itse luotua käyttäjätiliä ja iso, kirjava g-kirjain kuvaa Google-tiliä. Listassa näkyvät myös käyttäjän luonti- ja viimeisin kirjautumispäivämäärä, sekä käyttäjän yksilöllinen tunniste UID.



Kuva 18 Firebase-konsolin todennusvälilehti

4.1.4 GIF-tiedostojen lähetys ja vastaanotto

GIF on kuvaformaatti, joka on animoitu yhdistämällä useita kuvia yhdeksi tiedostoksi. Tämä tiedosto on koodattu grafiikan vaihtoformaattiin (graphics interchange format) eli tunnetummin GIF-tiedostoksi. GIF-tiedoston sisältämät kuvat esitetään peräkkäin, joka saa aikaan animoidun leikkeen tai lyhyen elokuvan. GIF-kuvat toimivat viihdyttävinä toteamuksina, vastauksina tai kommentteina verkkokeskusteluissa. Niitä käytetään myös kuvaamaan reaktioita tai

tuotteita ja selittämään konsepteja hauskalla, luovalla ja ytimekkäällä tavalla. (Williams 2016)

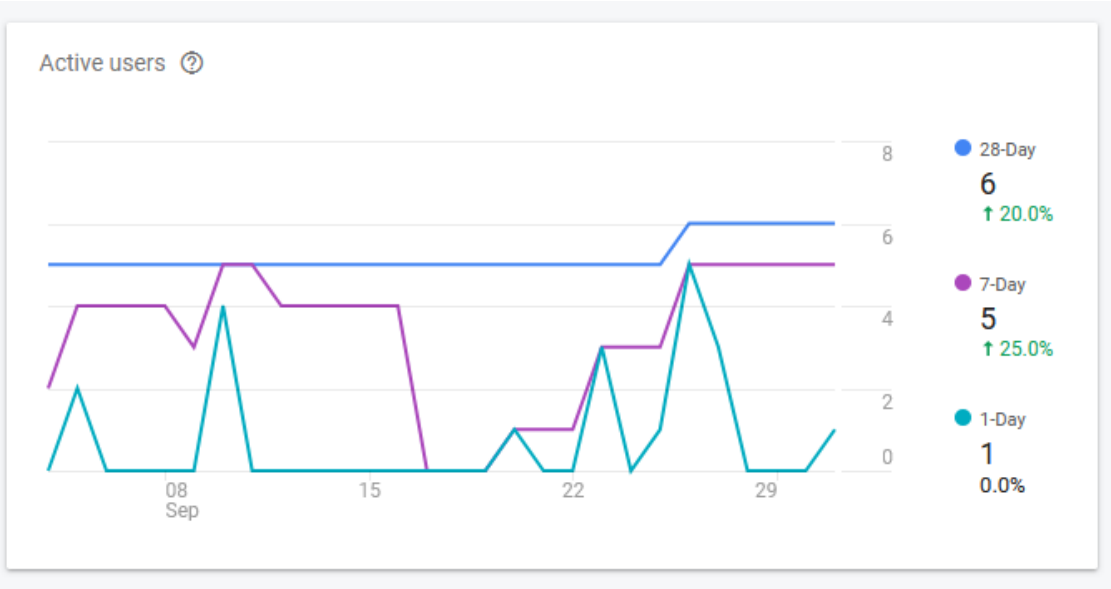
GIF-tiedostojen lähetys ja vastaanotto toteutettiin sisällyttämällä sovellukseen valmiiksi valittuja GIF-tiedostoja. Jokaisessa sovelluksessa on varastoituna samat valmiiksi valitut GIF-tiedostot. Tällöin niitä ei tarvitse lähettää käyttäjien välillä. Riittää kun välitetään viite, mikä GIF-tiedosto halutaan näyttää. Kun sovellus tunnistaa viitteen GIF-tiedostoon, se osaa automaattisesti näyttää sen ruudulla hakien kuvan omasta GIF-varastostaan.

Tämä menetelmä säästää suuria määriä tiedonsiirtokapasiteettia. Neljän megatavun kokoisen GIF-tiedoston lähettämisen sijaan lähetetään yhden tavun kokoinen viite kyseiseen tiedostoon, joka löytyy jo vastaanottajan sovelluksesta.

GIF-tiedoston sisältö näytetään viestiketjussa, samaan tapaan viestikuplan sisällä, kuin tekstikin. GIF-tiedoston sisältämällä viestillä on samat muuttujat, kuin tekstiä sisältävällä viestillä.

4.2 Ensimmäinen testausvaihe

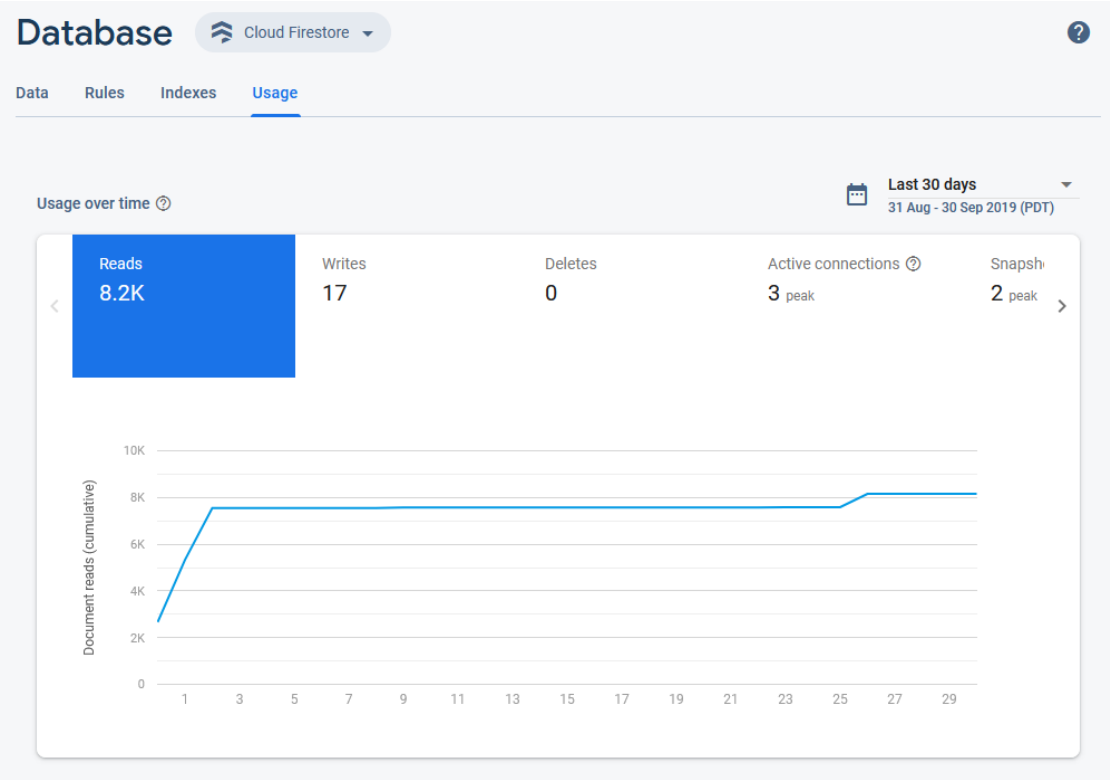
Testaajien asennettua sovelluksen laitteilleen tietoliikennettä, online-tietokannan käyttäytymistä ja käyttäjien aktiivisuutta seurattiin Firebase-tietokannan selainpohjaisesta konsolista. Kuva 19 on kuvaaja applikaation päivittäisten aktiivisten käyttäjien määrästä.



Kuva 19 Aktiivisten käyttäjien kuvaaja Firebase-konsolin seurantatyökaluista

Ryhmän jäseniltä kerättiin suullista palautetta sovelluksen toiminnoista, ulkonäöstä, toimivuudesta ja käyttömukavuudesta. Palautteen mukaan yleinen käytettävyys ja käyttöliittymä olivat hyvällä tasolla. Kenelläkään testiryhmän jäsenistä ei ollut vaikeuksia löytää tai käyttää toimintoja. Virheitä ei testauksen aikana löydetty. Toiveena esitettiin kuvien lähetys- ja vastaanottotoimintoa.

Seuraamalla testiryhmän sekä tietokannan käyttäytymistä saatiin selville seuraavat asiat. Sovellus tarvitsee viestitulvan, eli niin kutsutun ’spammin’ tai ’floodauksen’ eston, sekä tietokannan tietoliikenteen optimointia. Tarkastelemalla Kuva 20 näkyvää tietokannan käyttövälilehteä saatiin selville, että sovellus lukee tietokantaa liian paljon muuhun käyttöön nähden. Välilehdeltä voi tarkastella tietokannan luku-, kirjoitus- ja poistokertoja. Tämän lisäksi siitä näkee aktiiviset yhteydet ja nk. kuuntelijoiden lukumäärän. Käytön seurantaa jatkettiin hetken varsinaisen testivaiheen päätyttyä.



Kuva 20 Tietokannan käytön kuvaaja Firebase-konsolin seurantatyökaluista

4.3 Toinen kehitysvaihe

Toisen kehitysvaiheen kehitettävät toiminnot muodostuivat testiryhmän palautteen sekä testauksen aikana itse tehtyjen huomioiden perusteella. Tietoliikenteen optimointi ja viestitulvan esto priorisoitiin korkeammalle kuin kuvien lähetys- ja vastaanottotoiminto. Kun toisen kehitysvaiheen suunnitelma toimintolistoineen saatiin valmiiksi, pyydettiin testiryhmän jäseniä poistamaan sovellus laitteiltaan.

4.3.1 Tietoliikenteen optimointi

Tässä opinnäytetyössä tietoliikenteen optimoinnilla tarkoitetaan viestisovelluksen ja Firebase-tietokannan välisen tietoliikenteen minimoimista. Optimointi tarvitsi erittäin tarkan ja kontrolloidun kehitysympäristön, jotta jokainen suoritettu toiminto ja sen vaikutukset tietoliikenteeseen pystyttiin kirjaamaan ylös. Tätä edesauttoi se, että optimointivaiheessa sovellusta pääsi käyttämään vain sen kehittäjä.

Kuten luvussa 4.2 mainittiin, suurimpana ongelmana oli tietokannan lukukertojen suuri määrä varsinaiseen käyttöön nähden. Jotta määrän muodostumista

pystytään kontrolloimaan, tulee ensin selvittää mistä tekijöistä lukukertojen summa muodostuu. Lukuoperaatioita ei ole eritelty tietokannan analysointityökaluissa, joten erittely täytyi toteuttaa itse. Analysoinnin perusteella löytyi kaksi suurta tekijää, jotka kasvattivat lukuoperaatioiden summaa.

Ensimmäinen summaa kasvattava tekijä oli tietokannan tarkastelu selainpohjaisessa konsolissa. Viestiketjunäkymän aukaiseminen suoritti koko viestikokoelman luvun. Kokoelman sisältäessä 100 viestiä lisättiin lukuoperaatioiden summaan 100 joka kerta kun näkymä avattiin. Tämän lisäksi yksittäisen viestin tarkasteleminen lisäsi lukukertoja yhdellä. Tälle seikalle ei ollut mitään muuta tehtävissä kuin rajoittaa näkymän avausta.

Toinen ja suurempi tekijä oli sovelluksen toimintatapa. Kun käyttäjä avasi sovelluksen, se luki tietokannan viesti kerrallaan ja loi tämän perusteella viestiketjun näytölle. Kun käyttäjä lähetti viestin, tietokannan kuuntelija ilmoitti sovellukselle, että uusia viestejä on ladattavissa ja sovellus latasi kaikki tietokannan viestit uudelleen. Näin ollen viestihistorian ollessa 100 viestiä pitkä, yhden käyttäjän sovelluksen käynnistys ja yhden viestin kirjoitus lisäsi lukuoperaatioiden summaa 201 lukukerralla. Jos esimerkiksi viisi käyttäjää yksi kerrallaan avaa sovelluksen ja kirjoittaa yhden viestin, tulisi lukukertoja 1025. Jos tähän lisätään vielä se, että käyttäjät käyvät lukemassa jokaisen uuden viestin erikseen olisi lukukertojen summa 3085, kun taas kirjoituskertoja kertyisi vain 5.

Kuva 21 nähdään, että päivittäinen lukukertojen määrä ilmaiselle tietokannalle on 50 000. Sovelluksen toiminnan optimointi oli siis pakollista, sillä kiintiön täyttyessä tietokantaa ei voi enää käyttää ennen seuraavaa päivittäiskiintiön nollautumista.

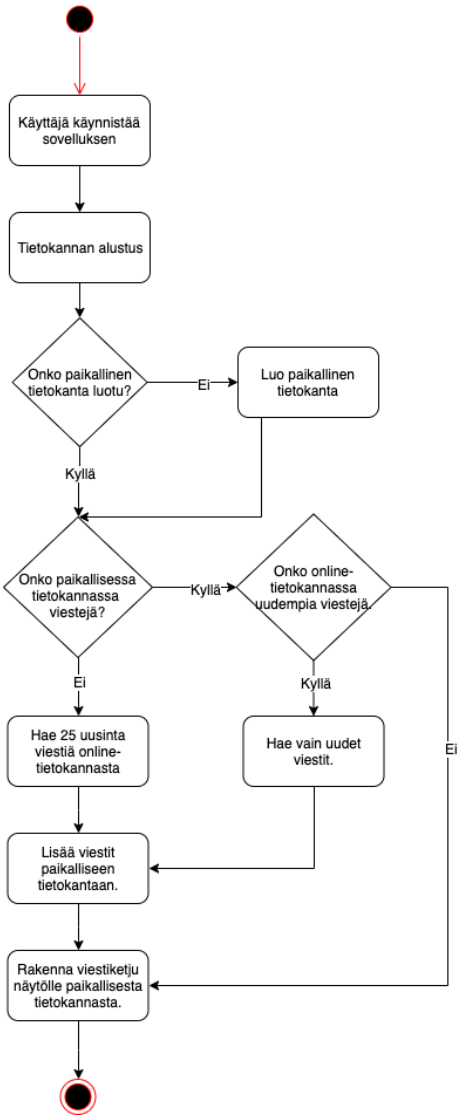
Storage

Resource	Usage today	Daily quota
Cloud Firestore Read Operations	0.0011 of 0.05 Million Ops	2%
Cloud Firestore Small Operations	0.000002 of 0.05 Million Ops	0%
Cloud Firestore API Calls	1,242	–
Cloud Firestore Stored Data	0.00026 of 1 GB	0%
Data Sent to Cloud Firestore API	0.000005 GB	–
Data Received from Cloud Firestore API	0.00005 GB	–
Cloud Firestore Entity Fetch Ops	1,038	–
Cloud Firestore Query Ops	60	–
Cloud Firestore Key Fetch Ops	2	–
Cloud Firestore Network Egress	0.00017 GB	–
Cloud Storage Standard Storage	0.01 of 5 GB	0%

Kuva 21 Firebase-tietokannan päivittäiskiintiöt käytettäessä maksutonta tiliä

Lukukertojen määrää vähennettiin rajoittamalla sovelluksessa näytettävää viestiketjun pituutta kahteenkymmeneenviiteen. Tämän lisäksi luotiin paikallinen tietokanta käyttäjän laitteeseen. Paikallinen tietokanta toimii varastona vanhoille viesteille, ja näin ollen online-tietokannan koko viestikokoelmaa ei ole tarvetta lukea.

Uuden toimintatavan periaatteena on verrata paikallisen tietokannan viimeisimmän viestin aikaleimaa online-tietokannan viimeisimmän viestin aikaleimaan. Aikaleiman ollessa sama tiedetään, että uusia viestejä ei ole, joten toimenpiteitä ei tarvita. Jos aikaleimat eriävät toisistaan on online-tietokantaan ilmestynyt uusia viestejä ja ne luetaan online-tietokannasta ja lisätään paikalliseen tietokantaan. Näytöllä oleva viestiketju rakennetaan uudestaan paikallisen tietokannan sisällöstä. Aikaleiman tarkkuus määritellään millisekunteina, joten yhtä aikaa lähetettyjen viestien mahdollisuus on melko pieni. Kuva 22 on vuokaavio tietokantojen käytöstä sovelluksen käynnistyessä.



Kuva 22 Tietokantojen yhteiskäytön vuokaavio

Lukukertojen määrää saatiin vähennettyä myös vaihtamalla viestin lähetyksen yhteydessä käytettävää kommunikointimetodia transaction-metodista add-metodiin. Ensin mainittu suoritti tietokannassa ensin yhden lukuoperaation ja tämän jälkeen yhden kirjoitusoperaation, kun taas add-metodi suoritti vain yhden kirjoitusoperaation. Kuva 23 on vertailutaulukko eri käyttöjärjestelmillä suoritetuista ja eri metodeilla toteutetuista viestin lähetyksistä.

Käyttäen 'transaction'-metodia	iOS	android
sovelluksen avaus		
viestisivun avaus ilman uusia viestejä	1 luku	1 luku
viestin lähetys	2 lukua / 1 kirjoitus	2 lukua / 1 kirjoitus
viestisivun avaus jos 1 uusi viesti	2 lukua	2 lukua

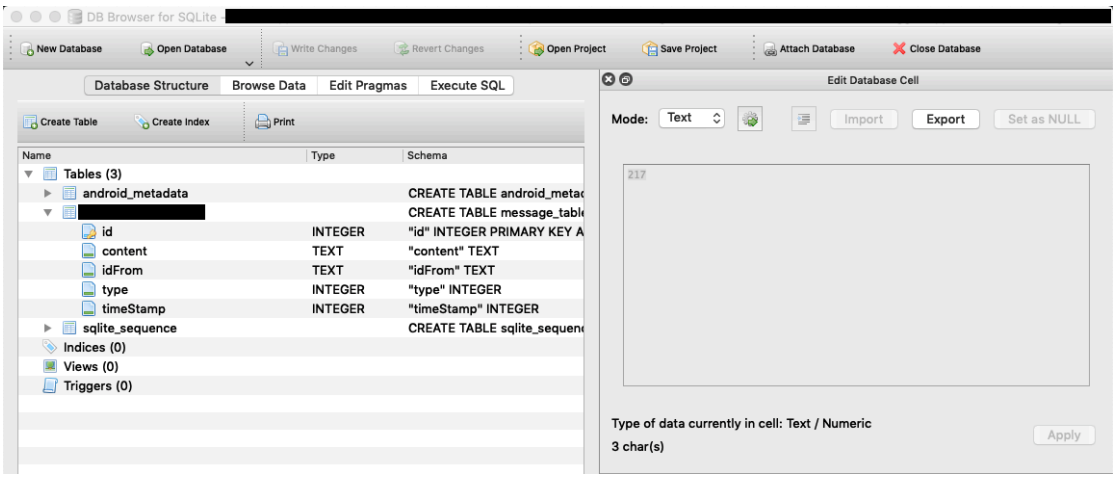
Käyttäen 'add'metodia	iOS	android
sovelluksen avaus		
viestisivun avaus ilman uusia viestejä	1 luku	1 luku
viestin lähetys	1 luku / 1 kirjoitus	1 luku / 1 kirjoitus
viestisivun avaus jos 1 uusi viesti	2 lukua	2 lukua

Kuva 23 Metodien vertailutaulukot

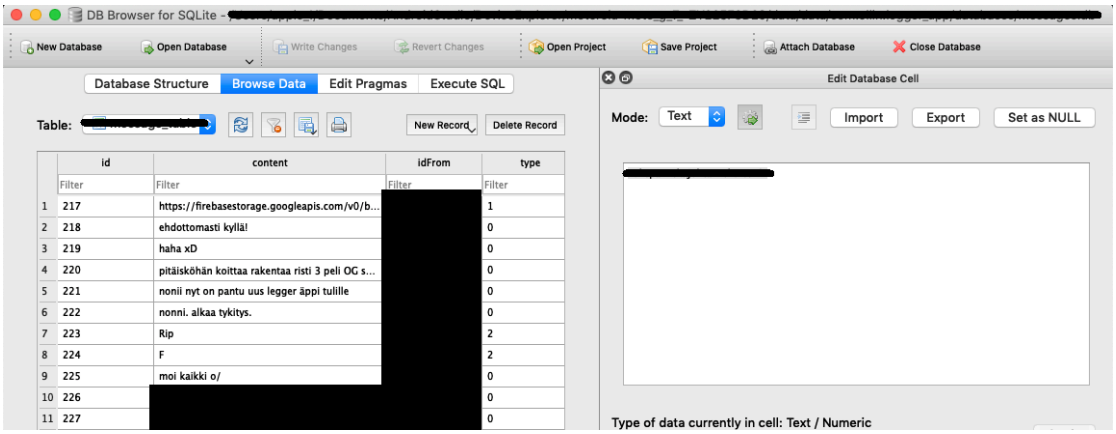
4.3.2 Paikallinen tietokanta

Käyttäjän laitteeseen sijoitettu paikallinen tietokanta toteutettiin SQLite-relaatiotietokantajärjestelmällä. SQLite ei tarvitse konfigurointia tai serveriä toimiakseen. Se on erittäin kompakti vaatien vain noin 500 kilobittiä levytilaa. SQLite toimii eri alustoilla ja se sisältää vain yhden tiedoston, joka voidaan kopioida, siirtää tai jakaa sähköpostilla. (SQLite s.a.)

Viestisovellus luo paikallisen tietokannan ensimmäisellä käynnistyskerralla. Tietokantaan luodaan viestipöytä, joka sisältää tallennetut viestit. Viestien muuttujat ovat täsmälleen samat kuin online-tietokannan viesteissä. Ne sisältävät siis yksilöllisen tunnuksen, sisällön, lähettäjän, tyypin ja aikaleiman. Kuva 24 on laitteeseen tallennetun paikallisen tietokannan rakenne, jota voidaan verrata Kuva 16 online-tietokannan rakenteeseen. Kuva 25 nähdään paikallisen tietokannan viestitaulun sisältöä.



Kuva 24 Paikallisen tietokannan rakenne

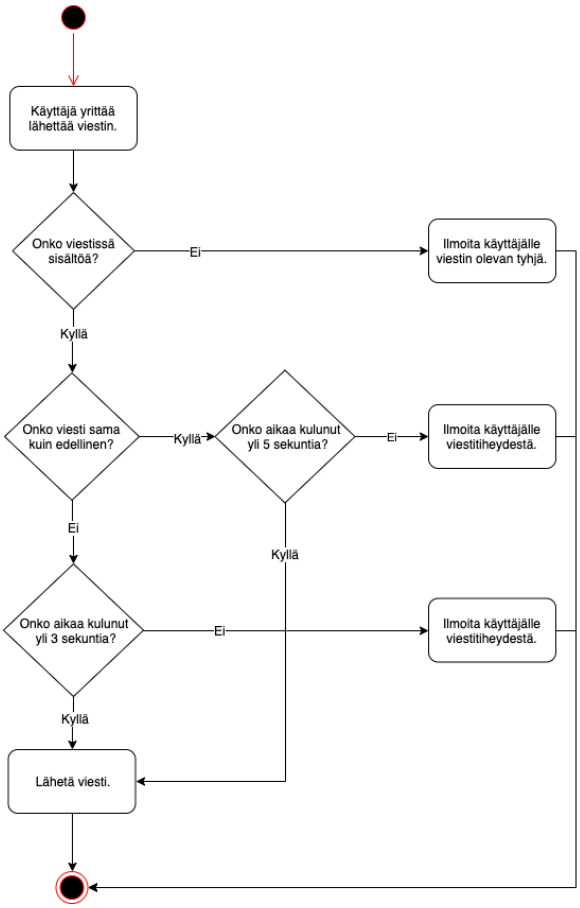


Kuva 25 Paikallisen tietokannan viestitaulu

4.3.3 Viestitulvan esto

Viestitulva on käyttäjäperäinen ongelma. Käyttäjä saattaa tarkoituksella tai vahingossa lähettää useita viestejä lyhyen ajan kuluessa. Tällainen toiminta kuormittaa tietokantoja, on haitaksi viestiketjun luettavuudelle ja heikentää käyttökokemusta.

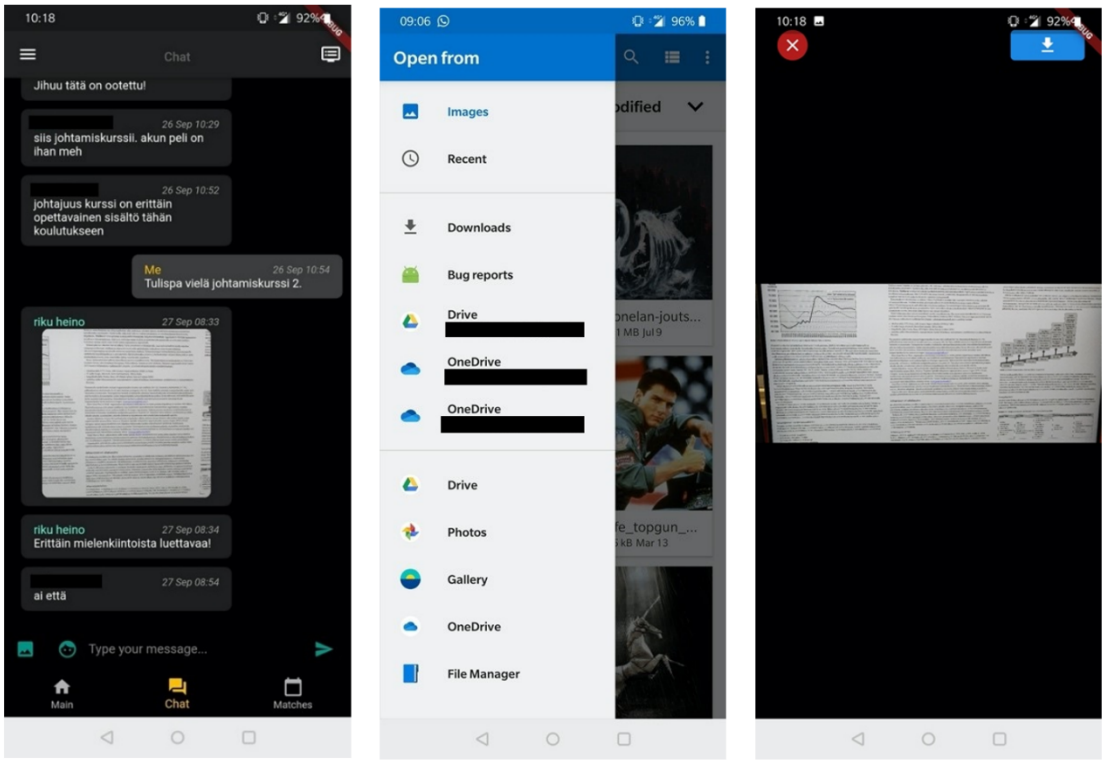
Jotta viestitulvatapaukset voidaan estää, tulee määritellä viestien lähetykseen sopiva intervalli, sekä viestin sisällön tarkastus. Sovelluksen viestin lähetysmetodiin kirjoitettiin ehto, joka tarkastaa, onko lähetettävässä viestissä sisältöä. Jos viesti on tyhjä, sitä ei lähetetä. Jos viesti läpäisee ehdon, tarkastetaan, ettei viestin sisältö ole sama kuin edellisessä kyseisen käyttäjän lähettämässä viestissä. Jos viestin sisältö ei ole sama, tarkastetaan vielä, onko käyttäjän edellisen viestin lähetyksestä kulunut 3 sekuntia. Edellisen viestin ollessa vanhempi kuin 3 sekuntia, uusi viesti lähetetään. Jos viestin sisältö on sama kuin edellisessä viestissä, tarkastetaan, onko käyttäjän edellisen viestin lähetyksestä kulunut 5 sekuntia. Edellisen viestin ollessa vanhempi kuin 5 sekuntia, uusi viesti lähetetään. Kuva 26 on vuokaavio sovelluksen toiminnasta viestiä lähetettäessä.



Kuva 26 Viestitulvan eston vuokaavio

4.3.4 Kuvien lähetys ja vastaanotto

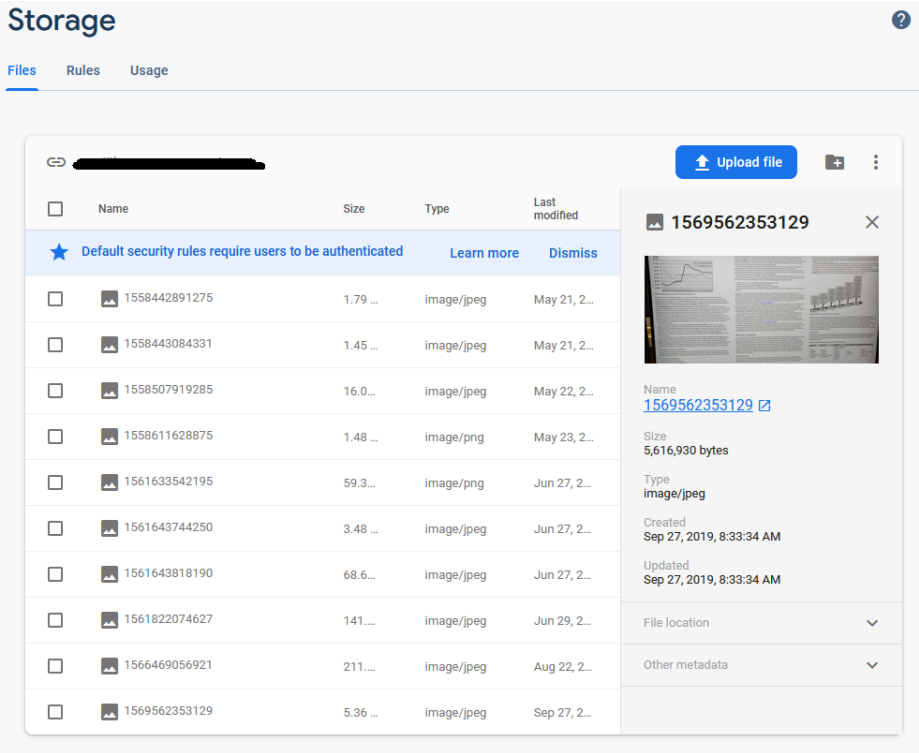
Kuvien lähettämistoiminnon kehittämisen taustalla oli testiryhmältä saatu palaute. Toiminnon kehittämiseksi täytyi tietokantaan avata varasto lähetettyjä kuvia varten. Tämä varasto on nimeltään Firebase Storage ja se on erillään viestitietokannasta. Kuva 27 vasemmassa laidassa nähdään, kuinka kuvaviestitoiminnon ulkoasu toteutettiin samankaltaisesti yleisimpien viestisovellusten kanssa.



Kuva 27 Kuvan lähetysnäkymät Android-alustalla

Lähettääkseen kuvan käyttäjän tulee painaa tekstikentän vasemmalla puolella sijaitsevaa maisemaikonia. Tämä avaa tiedostoselaimen, jonka voi nähdä Kuva 27 keskellä. Tiedostoselaimen kautta käyttäjä valitsee lähetettävän kuvan. Tiedosto on mahdollista valita käyttäjän laitteelta tai pilvipalveluista, kuten Google Drive tai OneDrive.

Kun käyttäjä lähettää kuvan, se tallennetaan tietokannan kuvavarastoon, joka voidaan nähdä Kuva 28. Firebase Storage luo kuvalle osoitteen, josta se on nähtävissä. Tämän jälkeen kuvan osoite välitetään viestinä jokaiselle käyttäjälle. Sovellus osaa lukea kuvan osoitteen ja hakea sen automaattisesti tietokannasta. Kuvaa ei lähetetä jokaiselle käyttäjälle suoraan laitteeseen vaan sovellus näyttää sen tietokannasta.



Kuva 28 Firestore-tietokannan kuvavarasto

Alkuperäisestä, lähetetystä kuvasta tehdään esikatseluversio ja se asetetaan viestikuplan sisälle, jotta se saadaan mahtumaan keskustelunäkymään. Esikatselukuva on nähtävissä Kuva 27 vasemman laidan näkymässä. Kun käyttäjä painaa esikatselukuvasta, aukeaa koko näytön kokoinen katselunäkymä, jonka voi nähdä Kuva 27 oikeassa laidassa. Katselunäkymään asetetaan tarkasteltava kuva skaalattuna niin, että se mahtuu kokonaisuudessaan näytölle. Käyttäjän painaessa kuvaa avataan kuvan päälle peitetaso. Peitetaso lisää ruudun vasempaan yläreunaan rasti-ikonin, jota painamalla poistutaan katselunäkymästä. Ruudun oikeaan yläreunaan peitetaso lisää latausikonin, jota painamalla käyttäjä voi ladata kyseisen kuvan omalle laitteelleen. Peitetason voi nähdä Kuva 27 oikean laidan näkymässä.

4.4 Toinen testausvaihe

Toisen kehitysvaiheen päätyttyä sovellus jaettiin testiryhmälle testausta varten. Käyttäjien ja tietokannan käyttäytymistä seurattiin samoilla tietokantatyökaluilla kuin edellisessä testausvaiheessa.

Tietokannan käytön kuvaajaa tarkastelemalla voitiin todeta tietoliikenteen optimoinnin onnistuneen. Tietokannan lukukertojen summa pysyi määrältään huomattavasti pienempänä kuin ennen optimointia. Tähän vaikutti osaltaan myös viestitulvan estotoiminto.

Viestitoiminnon käytettävyys oli kiitettävällä tasolla, eikä virheitä löytynyt. Uudeksi toiminnoksi testiryhmä toivoi notifikaatioita eli ilmoituksia uusista viesteistä. Toisen testausvaiheen lopuksi testiryhmää pyydettiin poistamaan sovellus laitteiltansa, jotteivat he pääsisi käyttämään sovellusta kesken kehitysvaihetta ja näin mahdollisesti häiritsemään kehitystä.

4.5 Kolmas kehitysvaihe

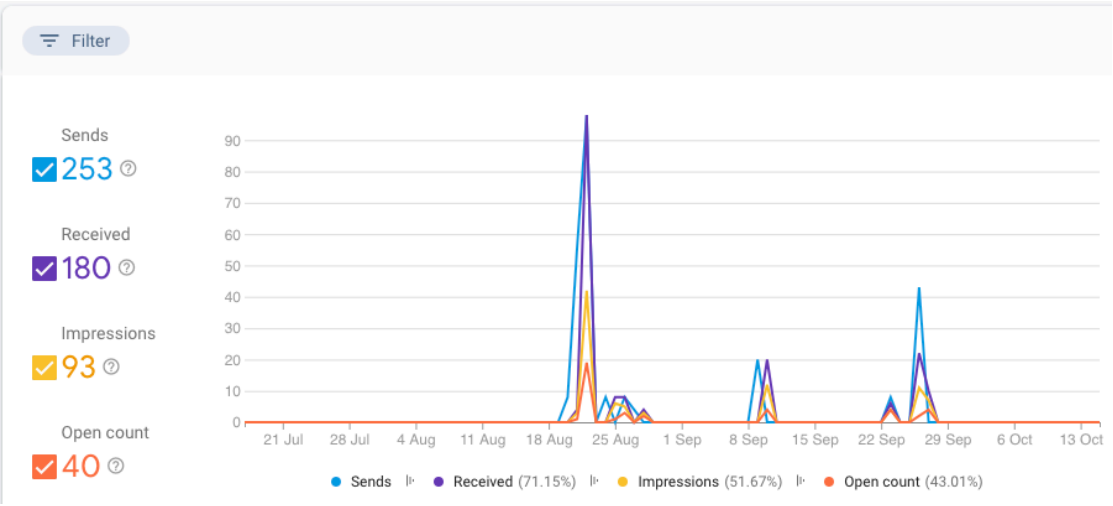
Testiryhmältä kerätyn suullisen palautteen perusteella muodostettiin kolmannen kehitysvaiheen ainoa kehitettävä toiminto, ilmoitukset. Muita uusia toimintoja ei ideoitu tai kehitetty, jotta työ pysyisi aikataulussa, eikä se laajenisi kohtuuttomasti.

4.5.1 Ilmoitukset

Ilmoitukset rakennettiin toimimaan ryhmän tunnuksen perusteella. Kehitysvaiheessa kaikki testaajat kirjattiin samaan ryhmään, kun he asensivat sovelluksen. Tämä tarkoittaa sitä, että jokainen testaaja saa ilmoituksen uudesta viestistä ryhmän keskustelusivulla.

Ilmoituksen lähetys tapahtuu Firebase-tietokannan viestityspalvelun (Messaging) välityksellä. Ilmoituksen lähetysmetodia kutsutaan sovellukseen toteutetussa viestin lähetysmetodissa. Ilmoituksen lähetysmetodissa välitetään palvelimelle tiedot viestin lähettäjästä, viestin sisällöstä ja ryhmästä, johon lähettäjä kuuluu. Viestityspalvelu lähettää ilmoitukset edelleen jokaiselle ilmoitetun ryhmän jäsenelle.

Ilmoituksista kertyvää dataa tarkkailtiin Firebase-tietokannan selainpohjaisen konsolin kautta. Kuva 29 on eriteltynä kuvaajina ilmoitusten lähetykset, vastaanotot ja muita tapahtumia. Näitä kuvaajia seuraamalla voitiin tarkkailla ilmoitusten toimivuutta ja käyttäjien toimintaa heidän saadessaan ilmoituksen.



Kuva 29 Ilmoitustapahtumien kuvaajat Firebase-konsolin seurantatyökaluista

Ilmoitusten toimivuutta testattiin kehitysvaiheessa molemmilla alustoilla toistuvasti, sillä ilmoitustoiminto otettiin käyttöön eri alustoilla eri tavalla.

4.6 Kolmas testausvaihe

Kun ilmoitukset saatiin toimimaan testilaitteissa toivotulla tavalla, jaettiin sovellus kolmannen kerran testiryhmälle. Testausvaiheen pääpaino oli ilmoitusten testauksessa. Tällä kertaa uusia kehitysideoita ei syntynyt, eikä virheitä toimivuudessa löytynyt. Kaikki suunnitellut toiminnot oli implementoitu ja ne testattiin eri alustoilla. Suullisesti kerätty palaute oli myönteistä, eikä toiminnoista löytynyt akuutteja hiomista vaativia kohtia. Näin ollen kaikki työn suunnitelman kohdat oli suoritettu ja työ oli valmis luovutettavaksi.

5 TULOKSET JA POHDINTA

Suunniteltu viestitoiminto saatiin kehitettyä ja kehittämistyö pysyi hyvin oletetussa aikaikkunassa. Toimintojen määrän pitäminen minimissä antoi mahdollisuuden keskittyä ydintoimintojen hiomiseen mahdollisimman hyvin toimiviksi. Projektin aloituksen ja alkuperäisen suunnitelman laatimisen jälkeen tehtävälistaan lisättiin vain erittäin tärkeitä toimintoja. Graafisen ulkoasun jättäminen hiomattomaksi nopeutti työtä ja auttoi pitämään fokuksen toimintojen helppokäyttöisyydessä ja koodin virheettömyydessä. Työn tarkka rajaaminen ja auttoi myös työn jäsentelyssä ja sen dokumentoinnissa.

Työn alkuun saaminen oli työvaiheista pitkäkestoisin ja raskain. Kokemuksen puute Dart-ohjelmointikielestä ja kaikista käytetyistä työkaluista aiheutti aluksi odotettua aikataulun venymistä. Flutter-käyttöliittymätyökalun toiminnan ja Dart-ohjelmointikielen syntaksin tullessa tutummaksi työn edistyminen nopeutui.

Yhteistyö pääapplikaation kehittäjän kanssa nopeutti oppimista ja se oli todella arvokas tekijä oppimisessa ja kehittymisessä. Yhteistyön ansiosta vältyttiin aikataulun liialliselta venymiseltä ja viestitoiminnon mahdollinen jatkokehitys on sujuvampaa.

Työtä vaikeuttivat työkalujen jatkuva kehittyminen. Sekä Flutter että sen käyttämä ohjelmointikieli Dart ovat varsin uusia ja aktiivisen kehityksen alaisia työkaluja. Työn aikana nämä molemmat saivat useita versiopäivityksiä aiheuttaen korjaustarpeita koodissa. Flutter-käyttöliittymätyökalun ja Dart-ohjelmointikielen kattavan ja ajantasaisen dokumentaation ansiosta korjaukset olivat kuitenkin pääasiassa nopeasti tehtäviä.

Viestitoimintoa on suotavaa jatkokehittää, jotta sen laatu saadaan riittävälle tasolle kaupallista levitystä varten. Jatkokehityksen kohteita ovat ainakin viestin aikaleiman tarkkuus ja toiminnon graafinen ulkoasu. Aikaleima määritellään tässä työssä millisekunnin tarkkuudella, joka jättää pienen virhemarginaalin samanaikaisille viesteille. Toiminnon graafinen ulkoasu tulee suunnitella uudestaan värien käytön ja joidenkin ikonien osalta. Ulkoasun tulee olla myös

yhtenäinen pääapplikaation ulkoasun kanssa. Tietokantojen käyttöä tulee jatkokehittää, mikäli päivittäin lähetettävien viestien määrä nousee useisiin tuhansiin. Myös useiden keskusteluryhmien yhtäaikaista käyttöä tulee kehittää ja testata, jotta saadaan kuva tietokannan käyttäytymisestä, kun käyttäjien ja välitettävien viestien ja ilmoitusten määrä kasvaa. Viestitoiminnon mahdollinen lisääminen pääapplikaatioon tulee vaatimaan lisää työtunteja ja jatkokehitystä.

Työ antoi arvokasta kokemusta ohjelmiston kehittämisen eri osa-alueista, kuten suunnittelu, toteutus, testaus, virheen korjaus ja uudelleen suunnittelu. Se opetti myös ohjelmiston suunnittelun tärkeyden. Kehityssuunnitelman rajaaminen helposti hallittavaksi kokonaisuudeksi oli avaintekijä työn ajoissa valmistumisen kannalta.

Ohjelmiston kehityksen lisäksi työ antoi kokemusta uusista ja kehittyvistä työkaluista kuten Flutter, Dart ja Firebase, jotka olivat aiemmin tuntemattomia. Nämä työkalujen käytön opettelu syvensi monialustaisen mobiiliapplikaation kehitystyöhön tarvittavaa osaamista ja opetti applikaation ja tietokannan välisen tietoliikenteen periaatteita ja hallintaa.

LÄHTEET

A tour of the Dart language. s.a. Google. WWW-dokumentti. Saatavissa: https://dart.dev/guides/language/language-tour [viitattu 11.11.2019].

Add data to Cloud Firestore. s.a. Google. WWW-dokumentti. Saatavissa: https://firebase.google.com/docs/firestore/manage-data/add-data [viitattu 16.10.2019].

Add Firebase to your Flutter app. s.a. Google. WWW-dokumentti. Saatavissa: https://firebase.google.com/docs/flutter/setup [viitattu 16.10.2019].

Agile Alliance. 2019. Agile 101. WWW-dokumentti. Saatavissa: https://www.agilealliance.org/agile101/ [viitattu 25.9.2019].

Authenticate Using Google Sign-In on Android. s.a. Google. WWW-dokumentti. Saatavissa: https://firebase.google.com/docs/auth/android/google-sign-in [viitattu 16.10.2019].

Babich, N. 2018. 10 Creative Loading Indicators. Medium. Blogi. Saatavissa: https://uxplanet.org/10-creative-loading-indicators-1a15c562b75a [viitattu 24.10.2018].

Cherryhappygirl. 2018. How To Use Color In UI Design Wisely to Create A Perfect UI Interface? Medium. Blogi. Saatavissa: https://blog.prototypr.io/how-to-use-color-in-ui-design-wisely-to-create-a-perfect-ui-interface-6e524bd023bc [viitattu 24.10.2019].

De Croos, P. 2018. When You Should (and Shouldn't) Use Firebase. WWW-dokumentti. Päivitetty 3.7.2018. Saatavissa: https://www.codementor.io/cultofmetatron/when-you-should-and-shouldn-t-use-firebase-f62bo3gxv [viitattu 30.9.2019].

FAQ. s.a. Google. WWW-dokumentti. Saatavissa: https://flutter.dev/docs/resources/faq [viitattu 18.9.2019].

Firebase. s.a. Google. WWW-dokumentti. Saatavissa: https://flutter.dev/docs/development/data-and-backend/firebase [viitattu 23.9.2019].

Firebase Security Rules. s.a. Google. WWW-dokumentti. Saatavissa: https://firebase.google.com/docs/rules [viitattu 16.10.2019].

Flutter SDK releases. s.a. Google. WWW-dokumentti. Saatavissa: https://flutter.dev/docs/development/tools/sdk/releases?tab=windows [viitattu 19.9.2019].

Foggia, L. 2018. Usability testing: what is it and how to do it? WWW-dokumentti. Saatavissa: https://uxdesign.cc/usability-testing-what-is-it-how-to-do-it-51356e5de5d [viitattu 1.10.2019].

Gallagher, A., Dunleavy, J., Reeves, P. 2019. The Waterfall Model: Advantages, disadvantages, and when you should use it. WWW-dokumentti. Päivitetty 23.4.2019. Saatavissa: https://developer.ibm.com/articles/waterfall-model-advantages-disadvantages/ [viitattu 5.11.2019].

Harris, R. 2015. Intel INDE Lets You Develop iOS Apps in Android Studio. Verkkolehti. Saatavissa: https://appdevelopermagazine.com/intel-inde-lets-you-develop-ios-apps-in-android-studio/ [viitattu 17.10.2019].

Koch, A. 2004. Agile Software Development: Evaluating the Methods for Your Organization. E-kirja. Norwood: Artech House. Saatavissa: https://kaakkuri.finna.fi/ [viitattu 27.9.2019].

Larson, G. 2016. Instant messaging. WWW-dokumentti. Saatavissa: https://britannica.com/topic/instant-messaging [viitattu 17.10.2019].

Meet Android Studio. s.a. Google. WWW-dokumentti. Saatavissa: https://developer.android.com/studio/intro [viitattu 17.10.2019].

Nevercode. 2019. What is Flutter? Benefits and limitations. WWW-dokumentti. Päivitetty 18.1.2019. Saatavissa: https://blog.codemagic.io/what-is-flutter-benefits-and-limitations/ [viitattu 20.9.2019].

Pricing plans. s.a. Google. WWW-dokumentti. Saatavissa: https://firebase.google.com/pricing [viitattu 11.11.2019].

ProfessionalQA. 2019. Software Testing Levels. WWW-dokumentti. Saatavissa: http://www.professionalqa.com/levels-of-testing [viitattu 30.9.2019].

Ridjanovic, D. Balbaert, I. 2013. Learning Dart. E-kirja. Birmingham: Pact Publishing. Saatavissa: https://kaakkuri.finna.fi/ [viitattu 23.9.2019].

Rouse, M. s.a. Instant messaging (IM). WWW-dokumentti. Päivitetty 2017. Saatavissa: https://searchunifiedcommunications.techtarget.com/definition/instant-messaging [viitattu 17.10.2019].

SQLite. s.a. Distinctive Features Of SQLite. WWW-dokumentti. Saatavissa: https://www.sqlite.org/different.html [viitattu 14.10.2019].

Technical overview. s.a. Google. WWW-dokumentti. Saatavissa: https://flutter.dev/docs/resources/technical-overview [viitattu 20.9.2019].

Techopedia. s.a. Cross-Platform Development. WWW-dokumentti. Saatavissa: https://www.techopedia.com/definition/30026/cross-platform-development [viitattu 21.10.2019].

Virtanen, J. 2016. Mistä muodostuu loistava käyttökokemus eli User Experience (UX)? Blogi. Saatavissa: https://contrast.fi/hyvan-kayttokokemuksen-ux-kolme-tarkeinta-elementtia/ [viitattu 30.9.2019].

Welcome to Xcode. 2019. Apple. WWW-dokumentti. Saatavissa: https://help.apple.com/xcode/mac/current/#/devc8c2a6be1 [viitattu 17.10.2019].

Williams, D. 2016. What is a GIF? Small Business Trends. WWW-dokumentti. Päivitetty 11.1.2019. Saatavissa: https://smallbiztrends.com/2016/03/what-is-a-gif.html [viitattu 7.11.2019].

KUVALUETTELO