Tampereen ammattikorkeakoulu

Firebase App Distribution Android-sovellusten testiversi-oiden jakelualustana

Miko Kauhanen

OPINNÄYTETYÖ
Marraskuu 2021

Tietojenkäsittelyn tutkinto-ohjelma
Ohjelmistotuotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma
Ohjelmistotuotanto

KAUHANEN, MIKO:
Firebase App Distribution Android-sovellusten testiversioiden jakelualustana

Opinnäytetyö 34 sivua
Marraskuu 2021

Opinnäytetyössä tutkittiin Firebase App Distribution -palvelun soveltuvuutta Android-sovellusten testiversioiden jakeluun. Opinnäytetyön tavoitteena oli luoda yleiskatsaus testiversioiden jakeluun tarkoitetusta palveluista ja saada parempi ymmärrys niiden hyödyllisyydestä mobiilisovellusten kehityksessä. Opinnäytetyön tarkoituksena oli tutustua käytännönläheisesti Firebase App Distribution -palveluun ottamalla se käyttöön Android-sovelluksen testiversion jakelualustana käyttäen Gradle -koontityökalua.

Opinnäytetyö oli konstruktiivinen tutkimus, jossa pyrittiin löytämään parempi toimintamalli testiversioiden jakeluun tutkimalla, miten testiversioiden jakelua voidaan toteuttaa Androidilla, helpottaako Firebase App Distribution -palvelu testiversioiden jakelua ja hallintaa ja selvittämällä, löytyykö Firebase App Distribution -palvelusta kehityskohteita. Tutkimuksessa käytiin läpi hyväksyntätestauksen taustoja ja toteutettiin testiversioiden jakelua Googlen Play-kaupan ja Firebase App Distributionin kautta sekä vertailtiin näiden palveluiden käyttöä keskenään.

Lopputuloksena ei syntynyt uutta toimintamallia, vaan todettiin perinteisen mallin soveltuvan jatkossakin testiversioiden jakeluun. Kokonaisuutena Firebase App Distribution ei tarjoa merkittävää parannusta perinteiseen Play-kaupan kautta toteutettuun jakeluun. App Distributionin hyviä puolia ovat sen soveltuvuus molemmille mobiilialustoille, käyttöönoton helppous kehittäjälle ja yhteensopivuus erilaisten työkalujen kanssa. App Distributionissa on kehitettävää erityisesti testaajien näkökulmasta. Testisovellusten asentaminen on monimutkainen ja monivaiheinen prosessi, jossa testaajalla täytyy olla riittävä luottamus sovelluksen kehittäjään. Tutkimuksen jälkeen voidaan todeta, että Play-kauppa on helppo ja tehokas tapa toteuttaa hyväksyntätestaus Android-sovelluksille.

Asiasanat: firebase, app distribution, android, hyväksyntätestaus

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Option of Software Development

KAUHANEN, MIKO:
Firebase App Distribution as a Distribution Platform for Test Versions of Android Apps

Bachelor's thesis 34 pages
November 2021

The thesis investigated the suitability of Firebase App Distribution service for distributing test versions of Android applications. The aim of the thesis was to provide an overview of test distribution services and to gain a better understanding of their usefulness in mobile app development. The purpose of the thesis was to provide a hands-on introduction to Firebase App Distribution by deploying it as a test distribution platform for an Android app using the Gradle build tool.

The thesis was a constructivist study that aimed to find a better approach to test version distribution. The thesis examined the background of acceptance testing and compared the distribution of test versions via the Play Store and Firebase App Distribution.

The final result of the thesis is that the traditional approach is still suitable for the distribution of test versions. Firebase App Distribution does not offer a significant improvement over Play Store distribution. The strengths of App Distribution are in its suitability for both mobile platforms, ease of deployment for the developer, and compatibility with different tools. App Distribution needs improvement, especially from the testers' point of view. The thesis concludes that the Play Store is easy and efficient way to perform acceptance testing for Android applications.

Key words: firebase, app distribution, android, acceptance testing

SISÄLLYS

1 JOHDANTO

Sovelluskehitysprojektit suunnitellaan ja toteutetaan sovelluskehittämisen elinkaarimallien mukaan. Elinkaarimallit voidaan jakaa vaiheellisiin ja iteratiivisiin malleihin. Vaiheellisissa malleissa sovelluskehitys etenee lineaarisesti vaiheesta toiseen, päätyen lopulta valmiiksi tuotteeksi. Iteratiivisissa malleissa sovellus rakentuu inkrementaalisesti ominaisuus kerrallaan ja lopullinen tuote muotoutuu jatkuvan palautteen ja korjauksien kautta. Mallista riippumatta sovelluskehitykseen kuuluu sovellusten testaaminen ennen julkaisua. Ideaalitilanteessa testaamista suoritetaan koko kehitysprosessin ajan sen eri vaiheissa ja testaamisen eri tasoilla. (Spillner & Linz 2021).

Ennen sovelluksen tuotantoversion julkaisua suoritetaan hyväksyntätestaus, jossa varmistetaan, että sovellus vastaa kehittäjien ja asiakkaan välisen tilaussopimuksen vaatimuksia sekä loppukäyttäjien tarpeita ja mieltymyksiä. Hyväksyntätestausta varten sovelluksesta luodaan testiversio, ja se jaetaan testaajien käyttöön. Testaajien palaute sovelluksesta analysoidaan, ja palautteen mukaan tehdään korjaukset sovellukseen. (Spillner & Linz 2021).

Iteratiivisessa sovelluskehityksessä uusia versioita luodaan jatkuvasti – joskus jopa viikoittain. Nopeassa kehitys- ja julkaisusyklissä uusien versioiden julkaisu ja hallinnointi pyritään automatisoimaan mahdollisimman pitkälle. (Spillner & Linz 2021). Hyväksyntätestaus täytyy valittujen testaajien toteuttaa manuaalisesti, mutta testiversioiden jakelua voidaan helpottaa jakelupalveluita hyödyntämällä.

Projektin aikana testaajat voivat vaihtua ja niiden määrä voi muuttua. Joskus sovelluksesta luodaan eri testiversioita eri testaajajoukoille – esimerkiksi verrattaessa erilaisia käyttöliittymän toteutuksia tai sovelluksen palvelumalleja keskenään. Testaajien ja testiversioiden hallinnointi käsin on työlästä, mikä on synnyttänyt tarpeen testiversioiden jakelupalveluille. Palvelujen yleisenä tavoitteena on tehdä hyväksyntätestauksen toteuttaminen ja testiversioiden jakelu mahdollisimman yksinkertaiseksi (Firebase App Distribution 2021a; Microsoft App Center Distribute 2021).

Android-sovelluksien julkaisuun on saatavilla useita eri jakelukanavia. Yksi jakelukanavista on Androidin virallinen Google Play -sovelluskauppa. Virallisen asemansa ja julkaistujen sovelluksien määrän perusteella Play-kauppaa voidaan kutsua pääsääntöiseksi julkaisualustaksi Android-sovelluksille. Vuoden 2021 ensimmäisellä neljänneksellä Play-kaupassa olikin saatavilla lähes 3,5 miljoonaa sovellusta verrattuna toisen sovelluskaupan, Amazon Appstoren, noin puoleen miljoonaan sovellukseen (Statista 2021a; Statista 2021b).

Play-kauppa tarjoaa sovelluskehittäjille testiversioiden hallintaan työkaluja, ja sen kautta voidaan sovelluksien testiversioita jakaa testaajien käyttöön hyväksyntätestausta varten. Testiversioiden hallintaan on myös saatavilla erillisiä selainpohjaisia palveluja, joiden tavoitteena on helpottaa entisestään hyväksyntätestauksen toteuttamista ja testaajien hallinnointia. Yksi näistä palveluista on tämän opinnäytetyön aiheena oleva Firebase App Distribution -palvelu.

2 OPINNÄYTETYÖN TAUSTAA

Tämän opinnäytetyön tavoitteena on luoda yleiskatsaus testiversioiden jakeluun tarkoitetusta palveluista ja saada parempi ymmärrys niiden hyödyllisyydestä mobiilisovellusten kehityksessä. Opinnäytetyön tarkoituksena on tutustua käytännönläheisesti Firebase App Distribution -palveluun ottamalla se käyttöön Android-sovelluksen testiversion jakelualustana käyttäen Gradle -koontityökalua.

Tämä opinnäytetyö on konstruktiivinen tutkimus, jossa pyrin löytämään paremman toimintamallin testiversioiden jakeluun. Opinnäytetyössä haluan saada vastaukset seuraaviin tutkimuskysymyksiin:

1. Miten testiversioiden jakelua voidaan toteuttaa Androidilla?
2. Helpottaako Firebase App Distribution -palvelu testiversioiden jakelua ja hallintaa?
3. Löytyykö Firebase App Distribution -palvelusta kehityskohteita?

Valitsin saatavilla olevista palveluista juuri App Distributionin tarkastelun kohteeksi, sillä Firebase ja sen muut palvelut ovat itselleni aikaisemmin tuttuja. Firebase on työssäni yleisesti käytetty palvelualusta, jolloin App Distribution on helppo ottaa tarvittaessa käyttöön muissa Firebasen palveluita käyttävissä sovelluksissa.

Pyrin hyödyntämään opinnäytetyön tuloksia suoraan eri projekteissa ja sen kautta saatua osaamista työelämässä nopeuttamaan sovellusten julkaisua, optimoimaan sovellusten julkaisuputkea ja helpottamaan hyväksyntätestauksen toteuttamista. Opinnäytetyön tuloksien myötä on mahdollista tehdä parempia päätöksiä siitä, onko erillisen palvelun käyttöönotolle tarvetta. Jos palvelulla nähdään olevan myönteisiä vaikutuksia, voidaan se ottaa asiakasprojekteissa käyttöön, mikä vastaavasti näkyy asiakkaalle parempana palveluna. Hyödyn voidaan nähdä jakautuvan myös koko organisaatiolle hyväksyntätestauksesta ja julkaisuputkesta lisääntyneen tiedon, käytännön osaamisen ja asiakastyytyväisyyden myötä.

3 HYVÄKSYNTÄTESTAUS

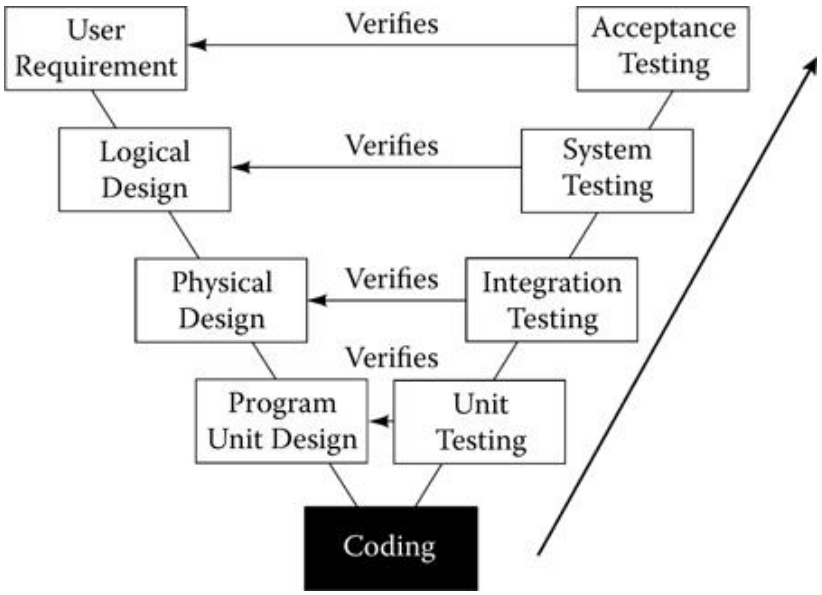
Tämä luku käsittelee sovellustestaamisen taustoja ja hyväksyntätestauksen osaa sovellustestuksessa. Luvussa selvitän mitä hyväksyntätestaus on, mikä on sen tavoite ja miten se voidaan toteuttaa. Luku auttaa ymmärtämään miksi sovelluksista luodaan testiversioita ja miten jakelupalvelut voivat auttaa hyväksyntätestauksen toteuttamisessa.

3.1 Hyväksyntätestaus osana sovelluskehitystä

Sovellustestaus on kokonaisuutena hyvin moniulotteinen aihe joka sisältää määritelmiä, jotka eivät aina ole tarkkaan rajattuja. William E. Lewis (2017) määrittelee teoksessaan *Software Testing and Continuous Quality Improvement, 3rd Edition* sovellustestauksen sovellusten laadunvarmistukseen kuuluvana aktiviteettina. Hän tarkentaa sovellustestauksen olevan riskienhallintastrategia, jolla varmistetaan, että sovellus vastaa toiminnallisia vaatimuksia. (Lewis 2017).

Andreas Spillner ja Tilo Linz (2021) taas määrittelevät teoksessaan *Software Testing Foundations, 5th Edition* sovellustestausta tarkemmin monivaiheisena prosessina, jossa varsinaisten testien suorittamisen ja tulosten tarkastelun lisäksi siihen kuuluu testauksen suunnittelua, testien analysointia ja testitapauksien (test cases) suunnittelua ja toteutusta sekä raportointia ja riskien analysointia. (Spillner & Linz 2021).

Määrittelyeroista huolimatta molemmat teokset jakavat sovellustestauksen neljään eri tasoon: komponenttitestaukseen (myös nimellä yksikkötestaus), integraatiotestaukseen, systeemitestaukseen ja hyväksyntätestaukseen. Testaamisen tasot eroavat toisistaan testauksen kohteen, tavoitteiden, tekniikoiden ja niiden vastuualueiden mukaan. (Spillner & Linz 2021; Lewis 2017).



KUVA 1. Testaamisen tasot ja kehityksen vaiheet V-mallissa. (Lewis 2017).

Testausprosessin viimeisenä oleva hyväksyntätestaus voidaan jakaa käyttäjän hyväksyntätestaukseen, systeemin tai sovelluksen operoijan hyväksyntätestaukseen, sopimus- ja sääntelykohtaiseen hyväksyntätestaukseen sekä kenttätestaukseen eli alpha- ja betatestaukseen (Spillner & Linz 2021).

Kenttätestaus voidaan jakaa sisäiseen alpha-testiin ja ulkoiseen beta-testiin. Alpha-testissä sovelluksesta luodaan testiversio ja testaajajoukkona toimii yleensä kehittäjäorganisaatio ja asiakkaan edustajat. Beta-testissä sisäisen vaiheen läpäissyt sovellusversion testaus voidaan laajentaa julkiseksi rajoitetulle määrälle loppukäyttäjiä. (Spillner & Linz 2021).

Kenttätestaus on halpa ja käytännöllinen tapa testata sovellusta todellisessa ympäristössä ja todellisilla laitteilla. Kenttätestauksella pyritään löytämään tuntemattomat ja osaksi-tuntemattomat muuttujat, joilla voi olla vaikutusta sovelluksen toimintaan. (Spillner & Linz 2021).

Testiversioiden lataamista testaajien käyttöön voidaankin tämän määrityksen mukaan kutsua kenttätestaukseksi. Asiakasprojekteissa olen kuitenkin huomannut, että rajaus sisäisen alpha-testin ja ulkoisen beta-testin välillä ei usein ole käytännön tasolla näin sitova. Beta-testausta voidaan toteuttaa myös suljetusti ilman loppukäyttäjiä tai sisäisesti kehittäjäorganisaatiossa. Alphatestaus voidaan myös laajentaa asiakkaan organisaatioon.

Lewis (2017) määrittelee hyväksyntätestauksen edellä mainittua laajemmin testeiksi, joilla varmistetaan, että sovellus vastaa alkuperäisiin vaatimuksiin. Lewis rajaa vastuun hyväksyntätestauksen suorittamisesta vain loppukäyttäjälle. Hän myös mainitsee, että hyväksyntätestaus ei ole pakollista suorittaa, jos asiakas on tyytyväinen systeemitestin tuloksiin, jos aikataulu pakottaa jättämään hyväksyntätestauksen pois tai silloin, jos käyttäjät ovat olleet aktiivisesti mukana sovelluksen kehityksessä. (Lewis 2017).

3.2 Hyväksyntätestauksen tavoite

Spillner ja Linz (2021) kertovat määrittelydokumenttien vaatimuksista ja hyväksyntätestauksen tavoitteista. Sovelluksen suunnitteluvaiheessa luodut määrittelydokumentit sisältävät funktionaalisia ja ei-funktionaalisia vaatimuksia. Funktionaaliset vaatimukset määrittelevät mitä sovelluksen kuuluu tehdä ja millä keinoin. Ei-funktionaaliset vaatimukset koskevat mm. sovelluksen suorituskykyä, luotettavuutta, skaalautuvuutta ja käytettävyyttä. Näin ollen funktionaalinen testaus vastaa kysymykseen “Mitä sovellus tekee?” ja ei-funktionaalinen testaus vastaa kysymykseen “Kuinka hyvin sovellus toimii?”. (Spillner & Linz 2021).

Hyväksyntätestauksen yhtenä tavoitteena on varmistaa, että sovellus toimii oikein ja sitä voidaan käyttää tarkoituksen mukaisella tavalla, toteuttaen sille asetetut funktionaaliset ja ei-funktionaaliset vaatimukset. (Spillner & Linz 2021).

Hyväksyntätestauksen toisena tavoitteena on luoda tietoa, joka mahdollistaa sidosryhmien – kuten asiakkaan – arvioida riskejä ja tehdä tietoon perustuvia päätöksiä sovelluksen lopullisesta julkaisusta. (Spillner & Linz 2021).

Lewis (2017) kertoo teoksessaan, että hyväksyntätestauksen motiivi on positiivinen eli tavoitteena on osoittaa, että sovellus toimii oletetulla tavalla. Huomiota ei kiinnitetä teknisiin ongelmiin, joiden kuuluisi olla löydettynä alemman tason testauksella, vaan siihen onko sovellus sopiva käyttäjille. (Lewis 2017).

Kiteytettynä hyväksyntätestauksella halutaan varmistaa, että sovellus vastaa kehittäjän ja sovelluksen tilanneen asiakkaan välisen sopimuksen vaatimuksia ja toiveita ominaisuuksien ja laadun kannalta. Hyväksyntätestauksella pyritään myös varmistamaan, että sovellus vastaa loppukäyttäjien tarpeisiin heidän toivomallaan tavalla. Hyväksyntätestaus on viimeinen varmistus ennen sovelluksen lopullista tuotantojulkaisua.

3.3 Hyväksyntätestauksen toteutus

Sovelluskehityksessä prosessia, jossa sovellus siirtyy koodista versionhallinnan kautta julkaisuksi pääosin automaattisesti, kutsutaan julkaisuputkeksi. Julkaisuputkeen sisällytetään testit, jotka täytyy läpäistä ennen sovelluksen lopullista julkaisua. Alemman tason testit voidaan suorittaa pääosin automaattisesti, mutta viimeinen hyväksyntätestaus täytyy testaajien suorittaa manuaalisesti.

Hyväksyntätestaus toteutetaan aina läpäistyjen systeemitestien jälkeen, jolloin sovelluksen tekninen toimivuus on varmistettu mahdollisimman kattavasti. Testien perustana toimivat sopimuksessa määritellyt hyväksynnän kriteerit eli tilanteet sovelluksen toiminnassa, jotka luokitellaan virheeksi. (Spillner & Linz 2021).

Lähtökohtaisesti alphatestaus suoritetaan kehittäjäorganisaation ympäristössä ja betatestaus suoritetaan asiakkaan tai loppukäyttäjän tiloissa, ympäristössä ja laitteilla. Näin pyritään luomaan mahdollisimman todenmukainen ympäristö, jossa sovellusta voidaan testata sen lopullisessa käyttötarkoituksessa ja tositilannetta vastaavalla käyttötavalla. Rajaukset organisaatioden välillä eivät aina ole näin sitovia, vaan eri vaiheessa olevia versioita voidaan testata myös ristiin.

Mobiilisovelluksilla hyväksyntätestaus voidaan toteuttaa jakelualustojen kautta luomalla sovelluksesta testiversioita ja lataamalla ne jakelualustoille asiakkaan ja loppukäyttäjien testattavaksi. Testaajat antavat sovelluksesta palautetta, jonka kehittäjät analysoivat. Palautteen pohjalta tehdään tarvittavat muutokset sovellukseen, jonka jälkeen luodaan uusi testiversio testattavaksi. Asiakkaan hyväksynnän myötä testiversio voidaan siirtää tuotantoon eli julkaista valmiina sovelluksena.

4 SOVELLUKSEN TESTIVERSIO

Tässä luvussa esittelen jakelualustojen testaamiseen käytettävän Android-sovelluksen ja sen kehitysympäristön. Luvussa selvitän, miten koontiversiot määritellään ja miten testiversioita luodaan Android Studiossa.

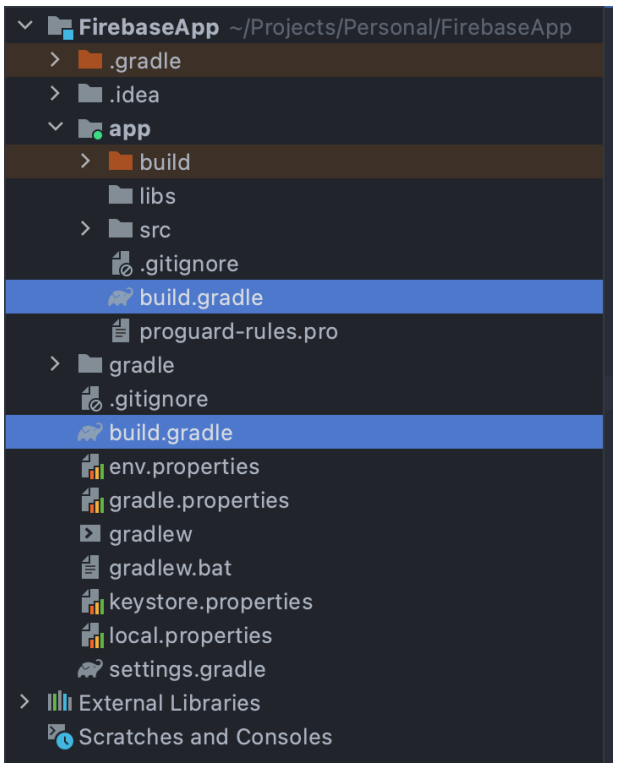
4.1 Sovelluksen ja kehitysympäristön esittely

Kehitysympäristönä toimii Android Studio (v.4.2.2), joka on Android-sovellusten kehittämiseen suositeltu integroitu kehitysympäristö (IDE, Integrated Development Environment). Opinnäytetyötä varten luon yksinkertaisen Android-sovelluksen Android Studiosta löytyvällä valmiilla sovelluspohjalla, sillä varsinainen sovellus tai sen ominaisuudet eivät ole olennaisia opinnäytetyön tuloksien kannalta. Pohja on Kotlin -ohjelmointikielellä luotu sovellus, joka sisältää kaksi näkymää ja valmiit projektin ja koonnin konfiguroinnit, joilla sovelluksen kehitysversiota voidaan ajaa fyysisellä Android-laitteella tai emulaattorilla.

Android Studioon sisältyy oletuksena oma koontijärjestelmä. Koontijärjestelmä kääntää sovelluskoodin ja resurssit ja paketoi ne asennuskelpoiseksi APK (Android Package Kit) tai AAB (Android App Bundle) sovelluspaketiksi. Koontijärjestelmä käyttää oletuksena Gradle-työkalua koonnin automatisointiin and hallinnointiin. Gradle mahdollistaa koontiasetusten konfiguroinnin erikseen kehitys-, testi- ja julkaisuversioille. (Android Developers 2021b).

4.2 Testiversion määrittely

Projektihakemisto sisältää kaksi koonnin konfigurointiin tarkoitettua build.gradle -tiedostoa ja Gradlen asetuksia määrittävät settings.gradle, gradle.properties ja local.properties -tiedostot. Ylemmän tason build.gradle -tiedosto määrittää projektitason konfiguroinnit, jotka vaikuttavat kaikkiin projektin moduuleihin. Moduulit voivat olla esimerkiksi saman projektin sisällä olevia erillisiä sovelluksia.



KUVA 2. Yhden moduulin Android -projekti build.gradle tiedostoineen.

Ylemmän tason build.gradlessa määritellään koko projektissa käytetyt arkistot ja riippuvuudet sekä Androidin versiot. Alempi moduulitason build.gradle sisältää yksittäisen moduulin koontiin liittyvät määritykset kuten koontityypit, koontivariantit, tuotevariantit, moduulin version ja sen tukemat Android -versiot. (Android Developers 2021b).

Tuotevariantti (product flavor) määrittää eri asetuksia esimerkiksi sovelluksen ilmaiselle ja maksulliselle versiolle tai sovelluksen kokeilu- ja täysversiolle (Android Developers 2021b). Tuotevariantin asetuksissa voidaan esimerkiksi määrittää mitä ohjelmointirajapinnan avaimia (API key) ja varmenteita (signing key) käytetään eri kehitysversiossa ja mitkä moduulit ovat käytössä missäkin variantissa.

Koontityyppi (build type) määrittää mitä asetuksia Gradle käyttää sovelluksen koontivaiheessa. Tyypillisessä sovelluksessa koontityyppit on jaettu kehitysvaiheen mukaan debug-, testaus- ja julkaisuversioon. Koontivariantti (build variant) koostuu tuotevariantista ja koontityypistä eikä sitä voida suoraan muokata. (Android Developers 2021b).

```
buildTypes {
    debug {
        applicationIdSuffix ".debug"
        debuggable true
    }
    release {
        signingConfig signingConfigs.releaseConfig
        minifyEnabled true
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
    }
    alpha {
        signingConfig signingConfigs.releaseConfig
        versionNameSuffix '-alpha'
    }
}
flavorDimensions 'version'
productFlavors {
    demo {
        dimension 'version'
        resValue("string", "app_name", "FirebaseApp Demo")

    }
    full {
        dimension 'version'
        resValue("string", "app_name", "FirebaseApp")
    }
}
```

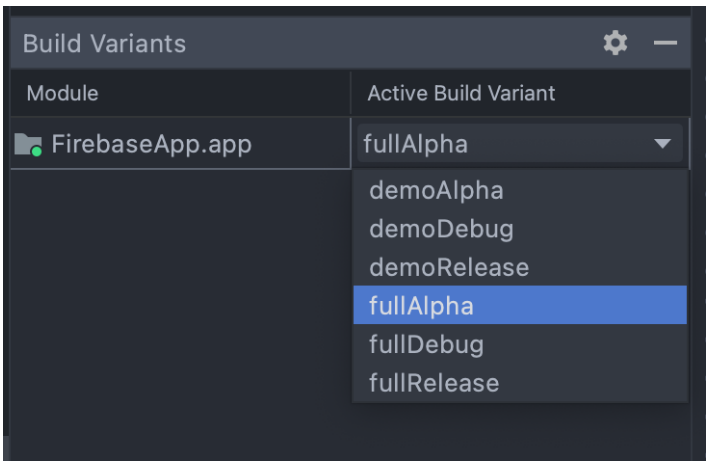
KUVA 3. Moduulitason build.gradlessa määritellyt koontityypit ja tuotevariantit.

4.3 Testiversion koonti

Testiversiota varten luodaan koontityyppi “alpha” moduulitason build.gradle -tiedostoon. Koontityypin asetuksissa määritellään versionimessä näkymään loppuliite “alpha” ja sovelluksen allekirjoituksessa käytettävä varmenne.

Play kauppaan tarkoitetut sovellukset allekirjoitetaan aina varmenteella. Varmenne on digitaalinen sertifikaatti joka varmistaa, että koodia ei ole muokattu sovelluksen koonnin jälkeen. (Android Open Source Project 2021).

Tuotevariantin “full” ja koontityypin “alpha” yhdistelmästä muodostuu koontivariantti fullAlpha. Sovellus voidaan koota joko Android App Bundleksi (AAB) tai Android Package Kit (APK) -sovelluspaketiksi. Android App Bundle on Googlen kauppasovelluksille tarkoitettu julkaisuformaatti, joka kokoaa sovelluskoodin ja resurssit, mutta jättää eri laitteille optimoitujen APK-pakettien ja sovellusten varmenteiden luomisen Play-kaupan vastuulle. (Android Developers 2021a).



KUVA 4. Sovelluksen koontivariantit.

Android-sovellusten eri koontivarianttien luominen ja määrittely on tehty helpoksi ja saatavilla olevat määrittelyvaihtoehdot ovat monipuolisia. Monipuolisuus mahdollistaa lukuisten erilaisten sovellusversioiden luomisen eri käyttötarkoituksia ja testaajajoukkoja varten. Hyväksyntätestausta varten voidaankin pienillä muutoksilla luoda eri testiversioita eri testaajajoukoille. Tämä mahdollistaa sovelluksen kokeilu- ja täysversion testauttamista ja vertailua eri testaajilla ja testiversioiden määrittelyä siten, että esimerkiksi testiversion kaatumis- ja käyttödata eivät sekoitu julkaisuversiosta saatuun dataan. Tämä taas helpottaa vain testiversiossa esiintyvien ongelmien tunnistamista ja ratkomista ennen julkaisua.

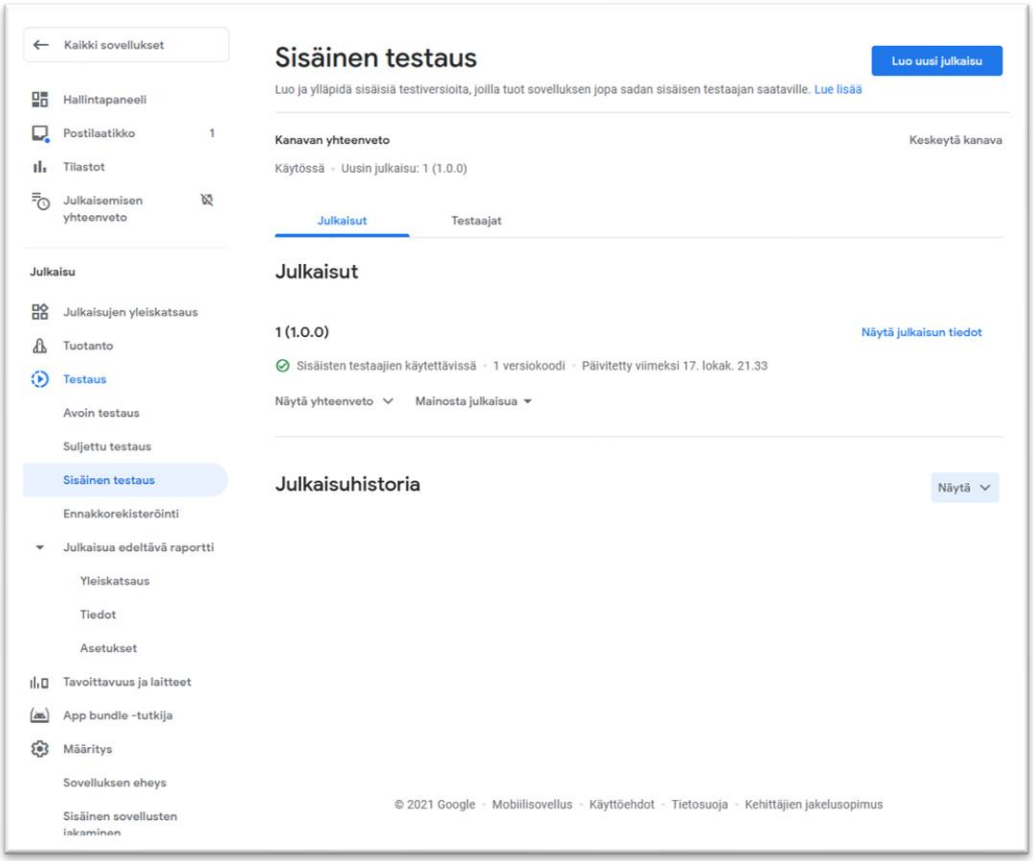
5 TESTIVERSION JAKELU PLAY KAUPASSA

Tässä luvussa selvitän miten Googlen Play kauppaa voi hyödyntää sovellusten testiversioiden jakelualustana julkaisemalla sovelluksen testiversion Play-kaupassa valikoiduille testaajille Play Consolen kautta. Luvun tarkoituksena on esittää, miten testiversioiden jakelu voidaan toteuttaa ilman erillistä palvelua.

5.1 Google Play Consolen esittely

Google Play on Androidin virallinen sovellusten jakelualusta. Google Play Console on kehittäjille tarkoitettu hallintapaneeli, jonka kautta sovelluksia julkaistaan Play kaupassa. Sovellusten lataaminen Play-kauppaan Play Consolen kautta vaatii maksullisen Google-kehittäjätilin.

Play Consolen kautta hallinnoidaan Play-kaupassa saatavilla olevien Android-sovelluksien sovellussivuja ja julkaisujen jakelua. Play Consolen kautta voidaan myös toteuttaa testiversioiden jakelua. Play Consolessa on saatavilla kolme eri testiversioiden jakelupolkua: sisäinen testaus, suljettu testaus ja avoin testaus. (Android Developers 2021c).



KUVA 5. Play Consolen sisäisen testauksen hallintasivu.

Sisäinen testaus on tarkoitettu organisaation sisällä käytettäväksi kanavaksi, jolla testiversiot saadaan nopeasti laadunvarmistustiimien ja kehittäjien testattavaksi ilman Googlen tarkastusprosessia. (Google Play Console n.d.b).

Suljettu testaus on alpha- ja betatestausta varten, jossa sallitut testaajat määritellään erikseen postituslistojen kautta. Suljetussa testauksessa on mahdollista luoda useita kanavia, joilla jakelua voidaan rajoittaa kanavakohtaisesti tiettyihin maihin ja alueisiin sekä tietyille sovellusversioille. (Google Play Console n.d.a).

Avoimessa testauksessa olevat julkaisut ovat vapaasti jokaisen Play kaupan käyttäjän löydettävissä ja saatavilla kun käyttäjä ilmoittautuu testaajaksi Play kaupan kautta. Avoimet testit voidaan myös rajata maa- tai aluekohtaisesti ja testaajien lukumäärää voidaan rajoittaa. (Google Play Console n.d.c).

Play Consolen käyttöliittymä on pääosin selkeä mutta valikoissa navigointi ja eri vaihtoehtojen löytäminen vaatii perehtymistä. Testikanavien eroavaisuudet eivät ole mielestäni riittävän selkeästi esitetty. Koen, että asiaan perehtymätön ei osaisi käyttöliittymän perusteella päätellä eroa sisäisen testauksen ja suljetun testauksen välillä. Myös se, että suljettu testaus voidaan jakaa useaan eri testikanavaan luo lisää monimutkaisuutta jo valmiiksi sekavaan ja vaihtoehtoja täynnä oleviin testausvalintoihin.

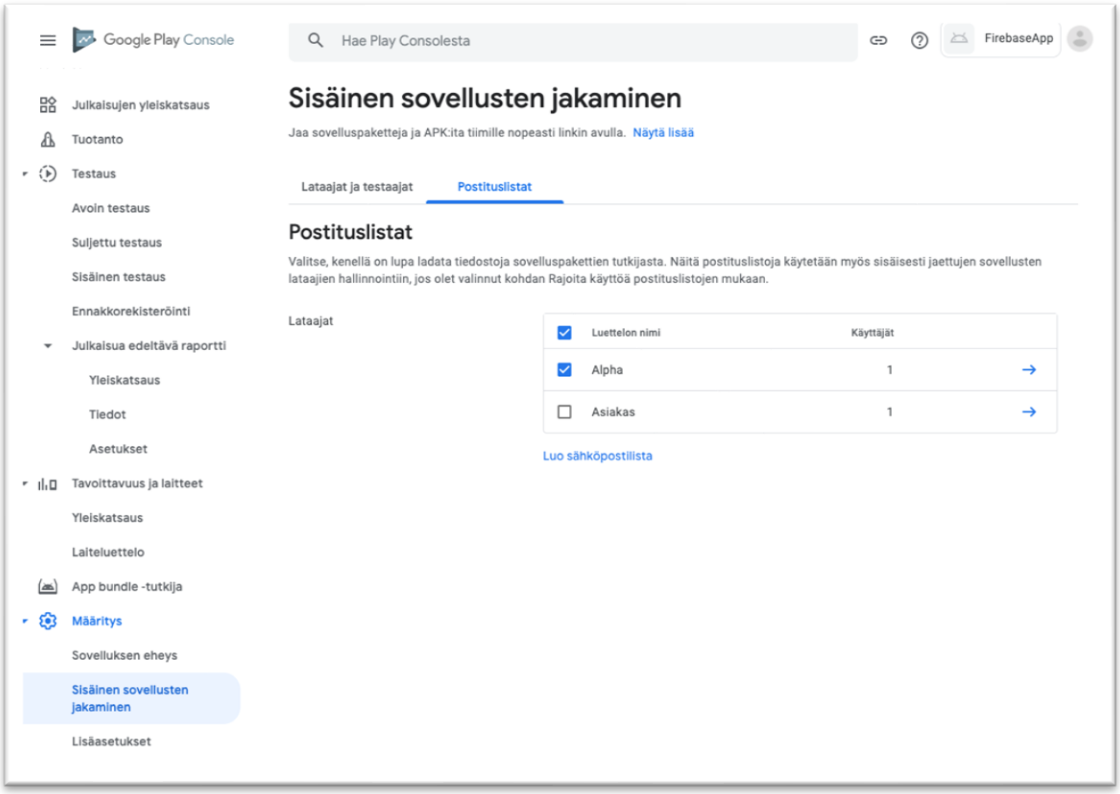
Toisaalta Play Console mahdollistaa osaavalle käyttäjälle erittäin yksityiskohtaisesti määritellyt ja rajatut hyväksyntätestit eri sovellusversioiden ja maa-alueiden perusteella. Näkisin kuitenkin hyödyllisenä saada lisätietoja eri testiversioista ja niiden latausmääristä näiden rajausten perusteella.

5.2 Testiversion julkaisu Play Consolessa

Play Consoleen kirjautumisen jälkeen, avautuu ensimmäisenä sovelluslistaus, jossa luodaan uusi sovellus. Sovelluksen luomisen jälkeen avataan sovelluksen hallintapaneeli, jossa voidaan hallinnoida ja tarkastella sovelluksen kauppasivua ja julkaisua.

Sovelluksen alphaversion julkaisua varten seurataan Google Supportista löytyvää virallista ohjeistusta (n.d.). Ensimmäiseksi täydennetään sovelluksen tiedot, kuten sovelluksen nimi, oletuskieli ja maksullisuus Play-kauppaa varten. Kauppasivun luomisen jälkeen määritellään testaajajoukko luomalla uusi postituslista testaajien sähköposteista ja mahdollistamalla sovelluksen lataaminen vain tämän postituslistan jäsenille. Listaan voidaan lisätä vain Google-tilejä, mikä voi olla ongelmallista asiakkaan suorittaman hyväksyntätestauksen kannalta. Monesti asiakasosapuolen sähköpostitilit eivät ole Google-tilejä, jolloin asiakkaan testaajien täytyy luoda erilliset tilit testausta varten. Tilien luonti ja määrittely voi olla aikaa vievää ja aiheuttaa liikaa ylimääräistä työtä silloin, kun hyväksyntätestaus halutaan saada nopeasti käyntiin.

Testaajajoukon luomisen jälkeen luodaan Play Consolessa suljettuun testipolkuun uusi Alpha-testikanava ja määritellään sille juuri luotu testaajajoukko. Lopuksi testikanavalle luodaan uusi julkaisu lataamalla kehitysympäristössä luotu Android App Bundle -sovelluspaketti.



KUVA 6. Testaajajoukkojen hallinnointipaneeli Play Consolessa.

Pääsääntöisesti lataaminen toteutetaan käsin etsimällä sovelluspaketti käyttöjärjestelmän hakemistosta. Lataaminen on myös mahdollista toteuttaa kolmannen osapuolen tarjoamalla Fastlane-työkalulla, jolloin lataaminen voidaan suorittaa komentoriviltä tai automatisoida suoraan versionhallinnasta.

Suljettuun ja avoimeen testaukseen jaetut sovellukset käyvät läpi Googlen tarkastusprosessin, jolloin sovellus on saatavilla vasta sen läpäistyään. Ensimmäisen ladatun sovellusversion tarkistus voi kestää muutamasta päivästä yli viikkoon, jolloin sovellus on saatavilla vain sisäisen kanavan kautta. Saman sovelluksen myöhemmät versiot eivät käy läpi samaa tarkistusprosessia, vaan ovat yleensä saatavilla saman päivän sisällä. Tarkistusaika onkin hyvä huomioida sovelluksen julkaisun ajoituksessa. On suositeltavaa varmistaa, että sovellus siirretään suljettuun testaukseen vähintään viikkoa ennen sovittua tuotantojulkaisua,

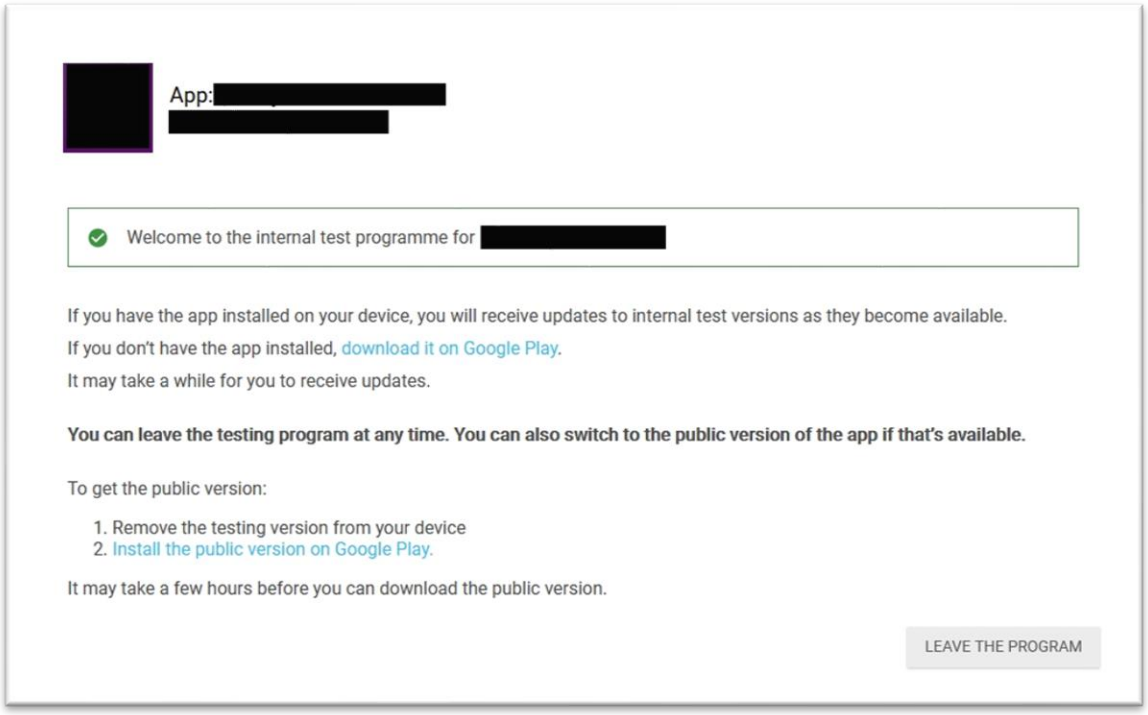
välttäen näin pitkän tarkistusajan myöhemmin. Myös testiversioiden jakelun aikana on hyvä huomioida ja ilmoittaa testaajille, että uusi versio ei näy välittömästi lataamisen jälkeen Play-kaupassa.

Tarkastusprosessin jälkeen sovellus on saatavilla alphatestiin liittyneille testaajille. Suljetun alphatestin julkaisut voidaan myöhemmin siirtää helposti avoimeen testaukseen tai tuotantoon julkisesti saataville.

5.3 Testiversion lataaminen

Testaajien täytyy hyväksyä kutsu suljettuun testiin kehittäjän antamasta linkistä. Linkkiä ei toimiteta automaattisesti, eikä sitä ole mahdollista lähettää testaajien sähköpostiin suoraan Play Consolen kautta, vaan se täytyy käsin jakaa jokaiselle testaajalle erikseen. Tämä on selkeä puute hyväksyntätestauksen toteuttamisen kannalta. Mitä enemmän testaajia on mukana ja mitä hajanaisempi testaajajoukko on kyseessä, sitä työläämpää linkkien lähettäminen on.

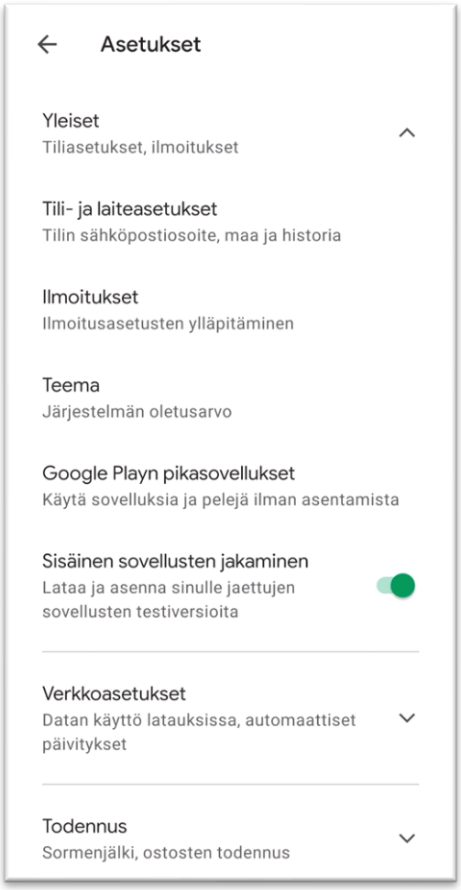
On suositeltavaa, että kehittäjä antaa testaajille suoran linkin kauppasivulle. Suora linkki helpottaa testaajan työtä, sillä sisäisessä tai suljetussa testauksessa oleva sovellus ei usein löydy Play-kaupan hakutoiminnon kautta. Kutsulinkissä on linkki sovelluksen Play-kaupan sivulle, mutta mielestäni se on liian helppo jättää huomioimatta. Testaajien työtä varmasti helpottaisi, jos testisovellukset näkyisivät erillisessä valikossa Play-kaupassa heti kun testaaja on hyväksynyt kutsun.



KUVA 7. Testisovelluksen kutsulinkki.

Testaaja tarvitsee Google-tiliin yhdistetyn mobiililaitteen ennen kuin sovelluksen saa ladattua. Kuten julkisesti saatavilla olevien tuotantosovelluksien kanssa, lataus onnistuu sovelluksen Play-kaupan kautta vain sovelluksen uusimmalle versiolle. Pidän tätä erikoisena ratkaisuna, sillä ajoittain uusin testiversio voi olla rikki, jolloin testaajalla olisi hyvä olla mahdollisuus ladata myös vanhempi toimiva versio. Tämä mahdollistaisi myös peräkkäisten versioiden vertailun keskenään.

Erikoisena seikkana sisäisen testausversion lataaminen vaatii muutaman ylimääräisen lisävaiheen. Ennen sovelluksen lataamista testaajien täytyy edellä mainittujen vaiheiden lisäksi käydä asettamassa laitteensa Play-kaupan asetuksista sisäisten sovellusten jakaminen päälle. Asetus ei ole suoraan näkyvissä vaan vaatii kehittäjätilan aktivoimista asetusten sovellustiedoissa. Ohjeet sisäisten sovellusten jakamisen aktivoimiseen löytyy vain dokumentaatiosta, jolloin kehittäjän on usein ohjeistettava erikseen testaajia tämän aktivoimisesta. Tämä hankaloittaa osaltaa hyväksyntätestauksen sujuvaa toteutusta.



KUVA 8. Sisäisen sovellusten jakaminen Play -kaupan asetuksissa.

Alkuvaiheen asetusten jälkeen testiversioiden lataaminen on vaivatonta ja tuttua kenelle tahansa Android-laitteen käyttäjälle. Uusien sovellusversioiden näkyminen kaupassa ja sovelluksien päivittäminen viimeisimpään testiversioon on ajoittain haastavaa. Usein Play Consoleen ladattu uusi sovellusversio ei näy heti testaajille edes sovelluksen omalta sivulta tai sovelluslistauksesta. Uusista versiosta ei myöskään tule ilmoituksia, jolloin testaaja voi epähuomiossa käyttää vanhaa versiota. Mielestäni paras ratkaisu olisi näyttää käyttäjälle hiljainen ilmoitus uudesta saatavilla olevasta versiosta.

6 TESTIVERSION JAKELU FIREBASESSA

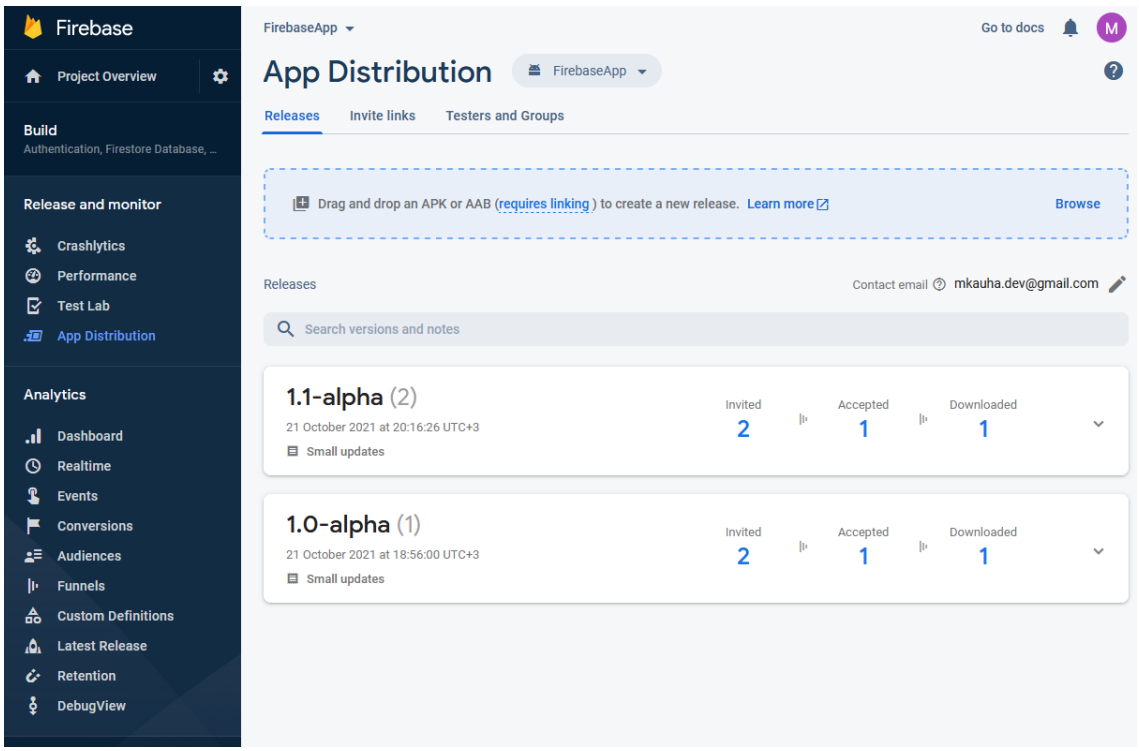
Tämä luku käsittelee Firebase -alustaa ja Firebase App Distribution -palvelua kokonaisuutena. Luvussa otan Firebase App Distribution -palvelun käyttöön Android-sovelluksen testiversion jakelualustana käyttäen Gradle -koontityökalua.

6.1 Yleiskatsaus palveluun

Firebase on Googlen ylläpitämä BaaS (Backend-as-a-service) -alusta mobiili- ja web-kehitykseen. BaaS-alustoilla voidaan ulkoistaa palvelimet ja niiden hallinnointi kolmansille osapuolelle. Ne myös tarjoavat työkaluja sovellusten backend-ominaisuuksien kuten tietokantojen, rajapintojen, pilvitoimintojen, tiedostonhallinnan, sosiaalisen median integraatioiden ja push-ilmoituksien toteuttamista varten. (Batschinski n.d.)

Firebasessa palvelut jakautuvat kolmeen osa-alueeseen: Build, Release & Monitor ja Engage. Osa palveluista on tarkoitettu varsinaiseen sovellusten kehittämiseen, osa sovellusten julkaisuun ja julkaisujen seurantaan ja osa käyttäjäkokemuksen optimointiin. Iso osa Firebasen palveluista on käytettävissä ilmaiseksi ja loput ovat käytön mukaan skaalautuvasti maksullisia. (Firebase n.d.).

Vuonna 2019 julkistettu ja toistaiseksi vielä beta-vaiheessa oleva Firebase App Distribution on yksi uusimmista alustan palveluista. App Distributionin tavoitteena on nopeuttaa ja helpottaa testiversioiden hallinnointia ja jakelua valituille testaajille. App Distributionilla voidaan toteuttaa Android- ja iOS-sovellusten testiversioiden jakelua. Jakelua voidaan tehdä selaimessa, komentoriviltä tai se voidaan integroida suoraan Fastlane- ja Gradle-työkaluihin. App Distributionin kanssa voidaan myös ottaa käyttöön Firebasen Crashlytics-palvelu, jolloin testiversioista saadaan reaaliaikaista diagnostiikka- ja kaatumisdataa. App Distribution pyrkii myös tarjoamaan testaajille helpon tavan antaa palautetta sovelluksista. (Ma 2019; Firebase n.d.).



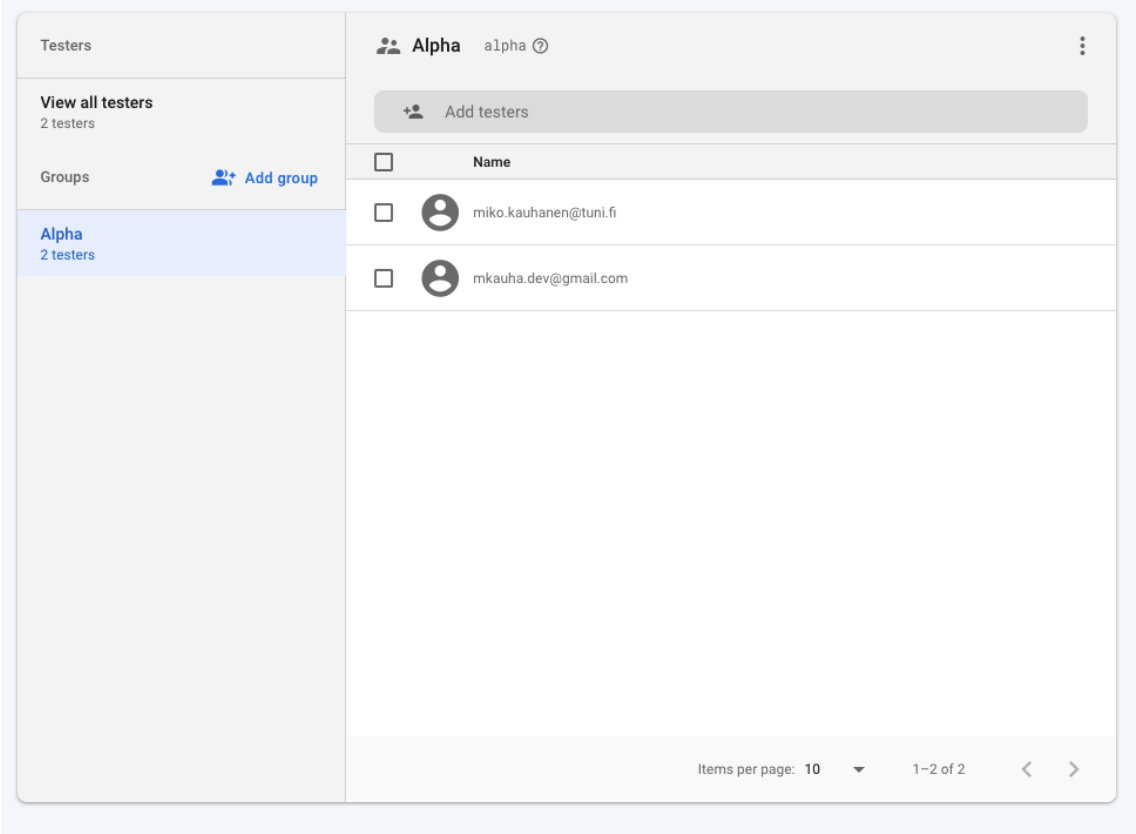
KUVA 9. Firebase App Distribution hallintapaneeli.

App Distributioniin lisättiin toukokuussa 2021 yhteensopivuus Google Android App Bundle (AAB) sovelluspakettien kanssa. Lisäyksen myötä App Distributionin kautta on mahdollista ladata testiversioita suoraan Google Play Consolen sisäiselle testipolulle. (Berry 2021).

6.2 Käyttöönotto ja konfigurointi

Testiversioiden jakelua pääsee hallinnoimaan Firebase Consolen App Distribution hallintasivun kautta. Hallintasivu on huomattavasti yksinkertaisempi ja karsitumpi verrattuna Play Consoleen vastaavaan. Hallintasivulla voi tarkastella kaikkia testiversioiden julkaisuja, testaajien kutsulinkkejä ja testaajajoukkoja. Testiversioiden jakeluun ei ole saatavilla useita kanavia vaan kaikki versiot näkyvät samassa näkymässä.

Sovelluksen hyväksyntätestiä varten luodaan Testers and Groups välilehdellä uusi testaajajoukko Alpha, johon lisätään testaajien sähköpostiosoitteet. Erona Play-kauppaan, sähköpostiosoitteiden ei tarvitse olla Google-tiliin liitettyjä ja testiversioiden jakelua ei pysty rajaamaan muuten kuin testaajajoukkojen kautta.



KUVA 10. Testaajajoukkojen hallinnointipaneeli App Distributionissa.

Firebasen (2021b) virallinen dokumentaatio tarjoaa erinomaiset ohjeet palvelun käyttöönotolle Android-sovelluksessa. Sovellusprojektissa palvelu otetaan käyttöön luomalla Firebaseen uusi projekti ja rekisteröimällä sovellus Firebaseen syöttämällä moduulitason build.gradle -tiedostosta löytyvä sovelluksen applicationId-tunniste palveluun. Rekisteröinnin jälkeen lisätään App Distribution -riippuvuus projektitason Gradle-tiedostoon ja lisäosa moduulitason Gradle-tiedostoon. Kirjautumalla komentorivin kautta projektiin Google-tilillä, annetaan Gradlelle oikeudet muokata Firebasen projektia.

Moduulitason build.gradle -tiedostossa alpha-koontityypille lisätään firebaseAppDistribution -kenttä, jossa määritellään Firebasen appId -tunniste, sovelluspaketin tyyppi, sovelluspaketin polku, muutoslista ja testaajajoukon tunniste.

```
alpha {
    signingConfig signingConfigs.releaseConfig
    versionNameSuffix '-alpha'
    buildConfigField "String", "API_KEY", envProperties['ALPHA_API_KEY']

    // App distribution määritykset
    firebaseAppDistribution {
        appId="1:61048806095:android:88ab437b3bfda0aca975ba"
        artifactType = "APK"
        artifactPath = "/Users/mkauha/Projects/Personal/FirebaseApp/app/build/outputs/apk/full/alpha/app-full-alpha.apk"
        releaseNotes = "Small updates"
        groups = "alpha"
    }
}
```

KUVA 11. App Distributionin määritys moduulitason build.gradlessa.

6.3 Testiversioiden jakelu ja hallinnointi

Sovellus ladataan palveluun ajamalla komentoriviltä koonti- ja latauskomento halutulle koontivariantille. Jos koontipaketin tyypiksi on valittu APK, voidaan sovellus julkaista suoraan App Distributioniin, jolloin palvelu lähettää testaajajoukolle sähköpostilla linkin uuden version lataamiseen. Sähköpostilmoitus lähetetään jokaisen uuden version latauksen yhteydessä, jolloin vaarana on, että nopeassa kehityssyklissä ilmoituksia tulee jatkuvasti. Hallintapaneelin kautta voidaan myös luoda erilliset kutsulinkit testaajille.

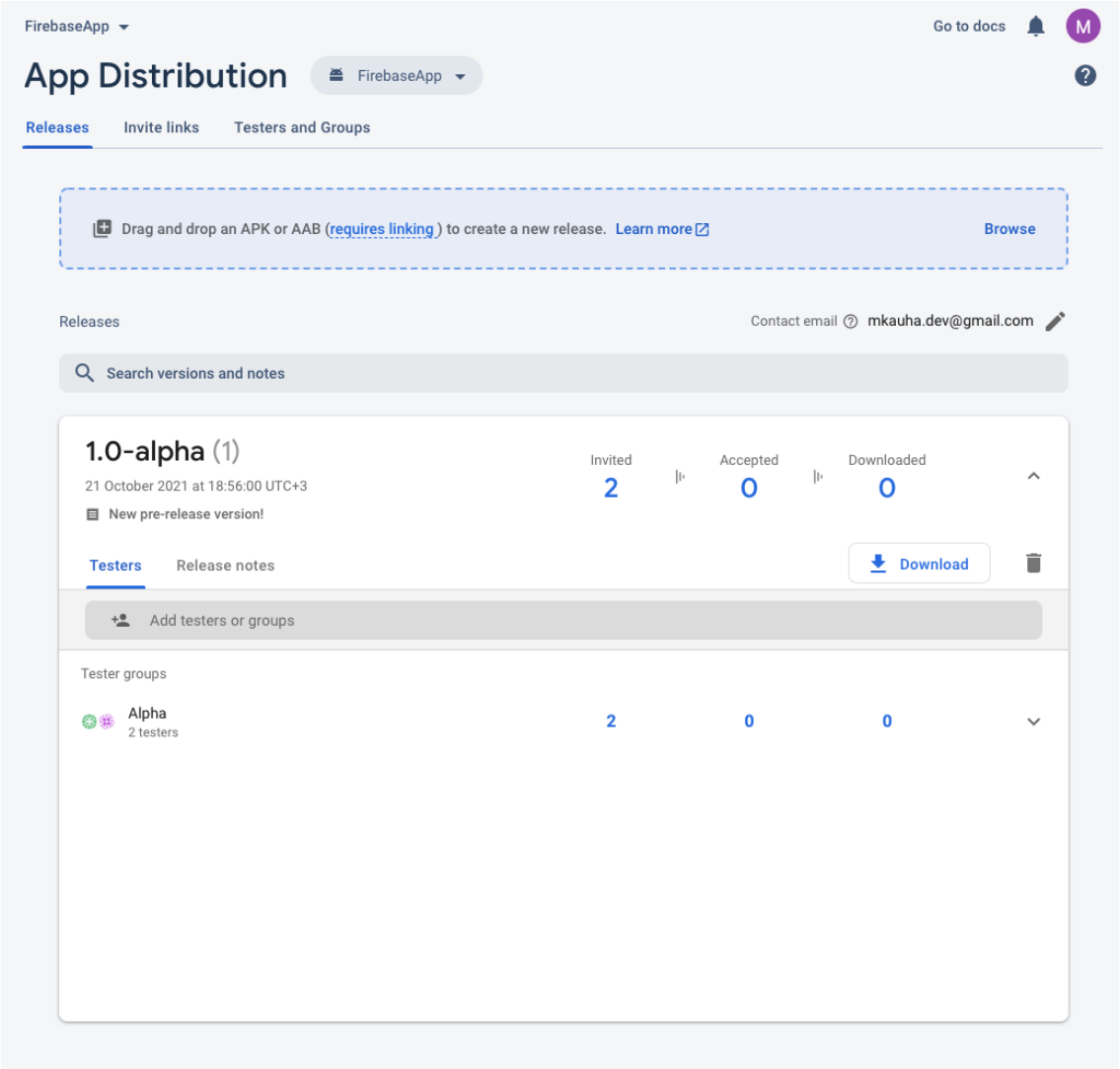
```
Terminal:  Local  +
mkauha@mmbp FirebaseApp % ./gradlew assemblefullAlpha appDistributionUploadfullAlpha

> Task :app:appDistributionUploadFullAlpha
Using APK path specified by the artifactPath parameter in your app's build.gradle: /Users/mkauha/Projects/Personal/FirebaseApp/
Uploading APK to Firebase App Distribution...
Getting appId from build.gradle file
Using cached Firebase CLI credentials
Uploading the APK.
Uploaded APK successfully 200
Added release notes successfully 200
Added testers/groups successfully 200
App Distribution upload finished successfully!

BUILD SUCCESSFUL in 26s
38 actionable tasks: 16 executed, 22 up-to-date
mkauha@mmbp FirebaseApp %
```

KUVA 12. Sovelluksen koonti ja lataaminen App Distributioniin.

Hallintapaneelista voidaan tarkastella yksittäisen sovellusversion tietoja kyseisen version hyväksyneistä ja ladanneista testaajista. Versio voidaan myös poistaa palvelusta, jolloin se ei ole enää saatavilla testaajille.



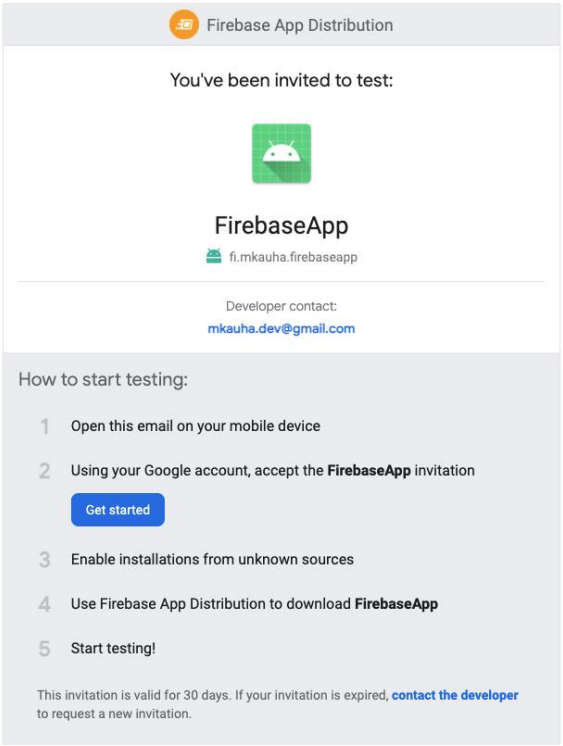
KUVA 13. Sovellusversion tiedot.

Jos sovelluspaketin tyypiksi on valittu AAB, täytyy Firebase olla yhdistettynä Google Play Consoleen sovellustunnisteen kautta. Tätä ennen sovellus täytyy olla julkaistuna Play Consolessa testipolulla tai tuotannossa. Yhdistetyn sovelluksen testiversiot ovat saatavilla suoraan Play-kaupasta samaan tapaan kuin käytettäessä Play Consolea julkaisualustana. Tässä kohtaa herääkin kysymys mitä tarkoitusta App Distribution palvelee, jos se toimii vain kehitysympäristön ja Play Consolen välikätenä.

6.4 Testiversion lataaminen

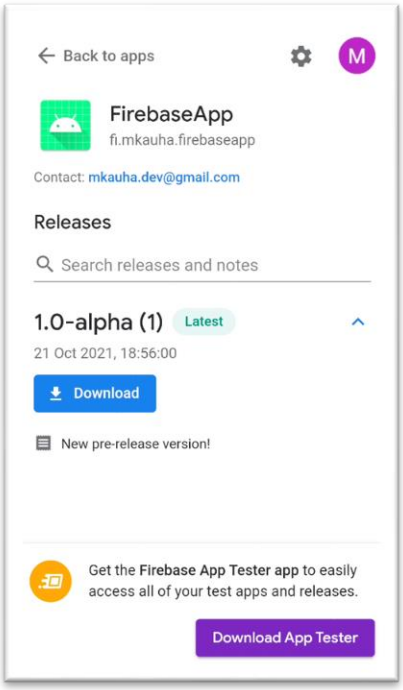
Testaaja saa jokaisesta uudesta sovellusversiosta sähköpostiin latausohjeet ja latauslinkin. Linkki avaa lataussivuston johon täytyy kirjautua Google-tilillä. Linkin

kautta ladatun sovelluksen asentaminen vaatii ulkoisista lähteistä ladattujen sovellusten asennuksen sallimista Android-laitteella. Koen, että tämän asetuksen käyttöönotto voi olla tavalliselle käyttäjälle liian hankala. Kutsulinkki ei tarjoa ohjeita asetuksen käyttöönottoon, vaan kehittäjän täytyy ohjeistaa testaajia erikseen. Useiden testaajien ohjeistaminen vie ylimääräistä aikaa ja hankaloittaa hyväksyntätestauksen toteuttamista.



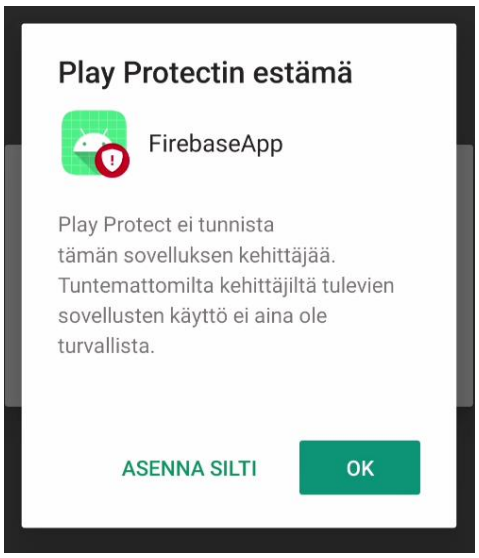
KUVA 14. Testaajan ohjeet sähköpostissa.

Testaajan on myös mahdollista ladata erillinen Firebase App Tester sovellus, joka näyttää kootusti kaikki sovellusversiot kaikista sovelluksista joissa käyttäjä on testaajana. App Tester näyttää myös automaattisesti uudet päivitykset testisovelluksiin muutoslistan kanssa. Tämä on merkittävä ja myös yllättävä etu Play-kauppaan verrattaessa. App Tester ei kuitenkaan tarjoa mitään säätömahdollisuuksia testaajalle esimerkiksi ilmoitusten osalta ja on ominaisuuksien osalta hyvin rajattu.



KUVA 15. Sovelluksen latausnäkymä mobiililaitteella.

App Distributionista ladattujen sovellusten asentaminen aiheuttaa lisähämmennystä. Koska sovellus ei ole ladattuna Play kauppaan, ja siten Googlen varmentama, ilmoittaa laite testaajalle sovelluksen tulevan tuntemattomalta kehittäjältä ja varoittaa sovelluksen asentamisesta. Testaajalla täytyykin olla riittävä luottamus sovelluksen kehittäjiin, salliakseen asentamisen mahdollisesti tuntemattomalta kehittäjältä. Jos luottamusta ei ole, tai testaaja ei tunne kehittäjää, voi sovellus helposti jäädä asentamatta.



KUVA 16. Varoitus tuntemattoman kehittäjän sovelluksesta.

7 JOHTOPÄÄTÖKSET JA POHDINTA

Opinnäytetyön myötä halusin löytää paremman toimintamallin testiversioiden jakeluun. Halusin selvittää, miten testiversioiden jakelua voidaan toteuttaa Androidilla, helpottaako Firebase App Distribution testiversioiden jakelua ja löytyykö Firebase App Distributionista kehityskohteita.

Opinnäytetyössä käytetty kirjallinen aineisto oli suurelta osin palveluiden virallista dokumentaatiota ja aihetta käsittelevää kirjallisuutta viimeisen viiden vuoden sisältä, joten pidän niiden kautta saatuja tuloksia luotettavana ja ajantasaisena. Aineisto rakentui myös palveluiden käytöstä saadusta kokemuksesta, jonka pyrin tuomaan esille mahdollisimman todenmukaisesti ja neutraalisti ilman ennakkoasenteita. On kuitenkin hyvä huomioida, että olen aikaisemmin käyttänyt Firebasea ja Play Consolea, jolloin ennakkoasenteilla voi olla vaikutusta tutkimuksen lopputulokseen. Palvelut myös kehittyvät jatkuvasti, jolloin on vaarana, että opinnäytetyön tulokset eivät ole täysin päteviä tulevaisuudessa.

Uskon kuitenkin saaneeni vastaukset esittämiini tutkimuskysymyksiin. Opinnäytetyön myötä kävi ilmi, että testiversioiden jakelua voidaan toteuttaa sekä Googlen Play-kaupan, että Firebase App Distributionin kautta onnistuneesti. Opinnäytetyössä käytiin läpi yksityiskohtaisesti testiversioiden jakelun toteutus molemmilla alustoilla. Opinnäytetyössä tuotiin esiin myös hyväksyntätestauksen teoreettista taustaa, pyrkimyksenä avata jakelualustojen tarpeellisuutta hyväksyntätestauksen tehokkaassa toteutuksessa.

Molempia alustoja käyttäessä ja vertaillessa kävi selväksi, että Firebase App Distributionista ja Play-kaupasta löytyy monia hyviä ominaisuuksia, mutta myös selkeitä kehityskohteita. Selvää oli kuitenkin se, että jakelualustat helpottavat testiversioiden jakelua ja siten hyväksyntätestauksen toteuttamista merkittävästi.

Yksi suurimmista App Distributionin vahvuuksista on sen saatavuus molemmille mobiilialustoille. Palvelun kautta voidaan hyväksyntätestaus toteuttaa yksittäisen sovelluksen Android ja iOS -versioilla siten, että testaajien ja testiversioiden hallinnointi on keskitettynä yhteen paikkaan. Firebasessa valmiina oleva sovellus on

helppo yhdistää Crashlytics -palveluun, jolloin sovellusten testiversioista on mahdollista saada reaaliaikaista diagnostiikka- ja kaatumisdataa. Näin mahdolliset testauksessa löydetyt ongelmat on helppo tunnistaa ja korjata.

Kehittäjän näkökulmasta erityisen hyvää oli App Distributionin käyttöönoton ja käytön helppous. Käyttöönotto ei vaatinut merkittävän suurta konfigurointia ja virallinen dokumentaatio ja ohjeistus oli kattava. Uutta sovellusta varten ei tarvitse luoda erillistä sovellussivua tai odotella Googlen tarkistusprosessin valmistumista. App Distributioniin ladattu uusi versio näkyi ja oli saatavilla testaajille välittömästi.

Testiversioiden lataaminen palveluun Gradlen ja komentorivin avulla oli helpompaa ja tehokkaampaa verrattuna Play Consoleen, jossa testiversiot siirrettiin selaimen latausikkunaan käsin. Molemmat palvelut voidaan myös integroida Fastlane-työkalun kanssa, jolloin uudet versiot saadaan luotua automaattisesti versionhallinnasta.

Myös Firebase App Tester -sovellus oli positiivinen yllätys. Sovellusta ei mainostettu App Distributionin hallintapaneelissa vaan se oli näkyvillä vain testattavan sovelluksen latauslinkin kautta. Testaajalle sovellus oli erittäin hyödyllinen, sillä se helpotti testiversioiden hallinnointia ja lataamista. Sovellus tarjosikin versiohistorian ja muutoslistan myötä paremman toiminnallisuuden tällä osa-alueella Play-kauppaan verrattuna.

Testaajan näkökulmasta katsottuna nousee esiin selkeitä kehityskohteita App Distributionissa. Palvelu lähettää jokaisesta uudesta versiosta testaajalle sähköpostiviestin, jolloin erityisesti versionhallinnan kanssa yhdistettynä voi viestejä tulla päivän aikana useita. Tämä on suora vastakohta Play-kaupan tilanteeseen, jossa ilmoituksia ei tule ollenkaan.

Ratkaisuna tähän näkisin sen, että kehittäjille annetaan mahdollisuus muuttaa ilmoitusasetuksia hallintapaneelin kautta. Näin ilmoitukset voitaisiin muokata eri tilanteisiin sopivaksi. Olisi myös aiheellista antaa testaajalle itselleen mahdollisuus muuttaa ilmoitusasetuksia App Tester -sovelluksen kautta esimerkiksi näkymään Push-notifikaatioina sähköposti-ilmoitusten sijasta.

Toisena kehityskohteena on itse testiversion asentaminen. Koska sovelluksen lähde ei ole Googlen varmentama, on sovelluksen asennus monimutkainen prosessi testaajalle. Sovelluksen asentaminen vaatii, että testaaja kirjautuu oikealle Google-tilille, osaa muuttaa tuntemattoman lähteen sovelluksen asetuksia, sekä luottaa kehittäjään riittävästi asentaakseen varmentamattomia sovelluksia.

Myös tämän vuoksi on suositeltavaa, että testaaja lataa App Tester sovelluksen, jolloin ensimmäisen version jälkeen uusien testiversioiden asennus sujuu helpommin. On erikoista, että App Tester -sovellus ei ole näkyvämmässä roolissa App Distributionissa. Koen, että monet palvelun ongelmista olisivat vähemmän haitallisia, jos testaajat pakotettaisiin käyttämään tätä sovellusta.

Opinnäytetyön kautta saadun kokemuksen perusteella en koe App Distributionia tarpeellisena palveluna yhdelle alustalle tähtäävän sovelluksen testiversioiden jakelussa. Palvelusta saatu hyöty keskittyy vain sovellusprojektin alkuvaiheeseen, jolloin on todennäköistä, että virallisia jakelukanavia ei ole vielä mahdollista ottaa käyttöön. Ainoastaan projektin alkuvaiheessa käyttöönoton helppous ja uusien testiversioiden nopea saatavuus tuo etua perinteiseen Play-kauppa jakeluun verrattuna.

Huomionarvoisena seikkana on Androidin uusi julkaisuformaatti Android App Bundle. Vuoden 2021 elokuusta lähtien kaikki uudet sovellukset täytyy koota App Bundleksi, jotta ne voidaan ladata Play-kauppaan (Android Developers 2021a). App Bundleksi kootun sovelluksen asentaminen laitteelle vaatii Play-kaupan, jolloin sitä ei voi asentaa muista lähteistä. Erillisten jakelualustojen on siis pakko olla intergroitavissa Play-kauppaan, jotta ne voivat hyödyntää tätä julkaisuformaattia. App Bundle voi siis lopettaa erillisten jakelualustojen edellytykset toimia vaihtoehtona Play-kaupalle. Jos erillinen alusta toimii vain välikätenä todelliselle jakelualustalle, sillä ei mielestäni ole syytä olla olemassa.

Kokonaisuutena App Distribution ei vielä tarjoa selkeää parannusta perinteiseen Play Consolen kautta suoritettuun testiversioiden jakeluun. Play Consolen kautta testiversioiden ja testaajajoukkojen hallinnointi on lähes yhtä helppoa tai jopa hel-

pompaa kuin App Distributionissa. Play Console tarjoaa myös monipuolisesti keinoja rajata testiversioiden jakelua tietyille joukoille ja alueille. Testaajille Play-kauppa on tutumpi ja sen kautta ladatut sovellukset ovat turvallisempia. Sovellusten asentaminen on myös merkittävästi helpompaa Play-kaupasta, vaikkakin testiversion löytäminen voi aluksi olla turhan hankalaa.

Jatkotutkimusta voisi suorittaa Firebase App Distributionin käytöstä iOS-sovelluksilla ja Fastlane-työkalulla. Fastlane vaikuttaa erittäin hyödylliseltä työkalulta, sillä se mahdollistaisi palveluiden yhdistämisen versionhallintaan.

Mielenkiinnolla seuraan muodostuuko beta-vaiheessa olevasta App Distributionista pelkästään ominaisuuksien osalta varteenotettava vaihtoehto virallisille testiversioiden jakelualustoille, vai katoaako alustan tarjoama hyöty kokonaan uuden julkaisuformaatin ja Play-kaupan kehittymisen myötä.

LÄHTEET

Android Developers. 2021. Android Studio. About Android App Bundles. Luettu 25.10.2021. https://developer.android.com/guide/app-bundle/

Android Developers. 2021. Android Studio. Configure your build. Luettu 20.10.2021. https://developer.android.com/studio/build

Android Developers. 2021. Google Play. Google Play Console. Luettu 20.10.2021. https://developer.android.com/distribute/console

Android Open Source Project. 2021. Security. Application Signing. Luettu 5.11.2021. https://source.android.com/security/apksigning/

Batschinski, n.d. G. Back4App. Backend as a Service – What is a BaaS? Luettu 19.10.2021. https://blog.back4app.com/backend-as-a-service-baas/

Berry, L. Firebase Blog. App Distribution Adds Support to Android App Bundles. Julkaistu 19.5.2021. Luettu 19.10.2021. https://firebase.googleblog.com/2021/05/app-distribution-adds-support-to-android-app-bundles.html

Google Firebase. n.d. Luettu 19.10.2021. https://firebase.google.com/

Google Firebase. 2021. Firebase documentation. Release and Monitor. Firebase App Distribution. Luettu 19.10.2021. https://firebase.google.com/docs/app-distribution

Firebase documentation. 2021. Distribute Android apps to testers using Gradle. Luettu 19.10.2021. https://firebase.google.com/docs/app-distribution/android/distribute-gradle

Google Play Console. n.d. Internal testing. Luettu 20.10.2021. https://play.google.com/console/about/internal-testing/

Google Play Console. n.d. Closed testing. Luettu 20.10.2021. https://play.google.com/console/about/closed-testing/

Google Play Console. n.d. Open testing. Luettu 20.10.2021. https://play.google.com/console/about/opentesting/

Google. Play Console Help. Set up an open, closed, or internal test. Luettu 20.10.2021. https://support.google.com/googleplay/android-developer/answer/9845334?hl=en&ref_topic=7071528

Lewis, W. 2017. Software Testing and Continuous Quality Improvement, 3rd Edition. Auerbach Publications.

Ma, F. Firebase Blog. What's new at Firebase Summit 2019. Julkaistu 26.9.2019. Luettu 19.10.2021. https://firebase.googleblog.com/2019/09/Whats-new-at-Firebase-Summit-2019.html

Microsoft. Visual Studio App Center. 2021. Distribute. Luettu 9.11.2021. https://docs.microsoft.com/en-us/appcenter/distribution/

Spillner, A. Linz, T. 2021. Software Testing Foundations, 5th Edition. Rocky Nook.

Statista. 2021a. Number of available apps in the Google Play Store from 2nd quarter 2015 to 1st quarter 2021. Luettu 8.11.2021. https://www.statista.com/statistics/289418/number-of-available-apps-in-the-google-play-store-quarter/

Statista. 2021b. Number of available apps in the Amazon Appstore from 1st quarter 2015 to 1st quarter 2021. Luettu 8.11.2021. https://www.statista.com/statistics/307330/number-of-available-apps-in-the-amazon-appstore/