OULUN AMMATTIKORKEAKOULU

Jarkko Kuha & Iisa Suonvieri

MONIALUSTAISEN SOVELLUKSEN JULKAISU GOOGLE PLAY - JA APP STORE -KAUPPOIHIN

MONIALUSTAISEN SOVELLUKSEN JULKAISU GOOGLE PLAY - JA APP STORE -KAUPPOIHIN

Jarkko Kuha & Iisa Suonvieri
Opinnäytetyö
Syksy 2021
Tietojenkäsittelyn tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma

Tekijät: Jarkko Kuha ja Iisa Suonvieri
Opinnäytetyön nimi: Monialustaisen sovelluksen julkaisu Google Play - ja Apple Store -kauppoihin
Työn ohjaaja: Jouni Juntunen
Työn valmistumislukukausi ja -vuosi: Syksy 2021
Sivumäärä: esim. 37

Tämän opinnäytetyön tarkoituksena oli toteuttaa jo olemassa olevan Xamarin.Forms-sovelluksen viimeiset tarvittavat toiminnallisuudet sekä julkaista se Google Play - ja App Store -kauppoihin. Opinnäytetyö tehtiin toimeksiantona Nive Information Technology -yritykselle. Julkaistava sovellus oli toimeksiantoyrityksen havaintojen raportointiin tarkoitettu mobiilisovellus. Ennen julkaisua sovellus oli konfiguroitava Google Play - ja App Store -kauppojen ohjeistuksen mukaan julkaisukelpoiseksi. Sovellus myös optimoitiin Microsoftin Xamarin.Forms-kehityksen dokumentaation ohjeistuksen mukaan, jotta julkaistu sovellus toimisi mahdollisimman hyvin loppukäyttäjän laitteessa. Sovelluksen kehitykseen käytettiin Visual Studio-ohjelmaa.

Kaikkia suunniteltuja toiminnallisuuksia ei saatu toteutettua aikataulurajoitteen vuoksi, mutta sovellus pystyttiin julkaisemaan sovelluskauppoihin siitä huolimatta, ja sovellus on nyt saatavilla Google Play - ja App Store -sovelluskaupoista. Sovelluksen julkaisuun käytettiin Google Play - ja App Store -julkaisuportaaleja. Julkaisuprosessissa huomattiin, että Xamarin.Forms-sovelluksen julkaisua koskeva dokumentaatio oli hieman vanhentunut. Julkaisuun löytyi kuitenkin ohjeistusta esimerkiksi YouTube-palvelusta sekä Googlen ja Applen omasta dokumentaatiosta, ja julkaisu saatiin lopulta toteutettua onnistuneesti.

Asiasanat: sovelluksen julkaisu, monialustaisuus, Google Play, App Store, Xamarin.Forms, M-Files, havaintojen raportointi

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems

Authors: Jarkko Kuha & Iisa Suonvieri
Title of thesis: Publishing a cross-platform application to Google Play and App Store
Supervisor: Jouni Juntunen
Term and year when the thesis was submitted: Autumn 2021
Number of pages: 37

The purpose of this thesis was to implement the last needed functionalities in an already existing Xamarin.Forms application and to publish it to Google Play and App Store. The thesis was commissioned by Nive Information Technology. The published application was a mobile application for observation reporting. Prior to release, the app had to be configured to be publishable according to the instructions of Google Play and App Store. The application was also optimized so that it would run as smoothly as possible on the end user's device. The application was developed using Visual Studio.

Not all of the planned functionalities were implemented, but the app was able to be published to app stores regardless, and it is now available in Google Play and App Store. Google Play Console and App Store Connect portals were used for publishing the app. During the process, it was noted that the documentation regarding the publishing of a Xamarin.Forms app was a bit outdated. However, using instructions found on YouTube and information provided by Google and Apple the app was eventually successfully published.

Keywords: publishing application, cross-platform, Google Play, App Store, Xamarin.Forms, M-Files, observation reporting

SISÄLLYS

1 JOHDANTO

Tämän opinnäytetyön aihe on havaintojen raportointisovelluksen kehittäminen Xamarin.Forms-sovelluskehyksellä sekä sen julkaisu Google Play - ja App Store -sovelluskauppoihin. Opinnäytetyö tehdään toimeksiantona Nive Information Technology Oy:lle. Sovelluksen perustoiminnallisuus on jo toteutettu, mutta joitakin tärkeitä ominaisuuksia vielä puuttuu, jotta sovellus voidaan julkaista sovelluskauppoihin. Tämän opinnäytetyön aikana toteutetaan viimeiset tarvittavat ominaisuudet ja sen jälkeen sovellus julkaistaan ladattavaksi App Store - ja Google Play -kaupoista.

1.1 Havaintojen raportointisovellus

Opinnäytetyössä käsitellään Nive Information Technology Oy:n havaintojen raportointiin tarkoitettua mobiilisovellusta. Sovelluksella käyttäjä voi raportoida havainnon omalla mobiililaitteellaan tapahtumapaikalta tai myöhemmin muusta sijainnista. Käyttäjä kirjoittaa havaintoraporttiin tarvittavat tiedot, ja raporttiin voi lisäksi halutessaan liittää kuvan, videon tai manuaalisesti tarkennetut paikannuskoordinaatit. Havainto tallentuu yrityksen M-Files-tiedonhallintajärjestelmään. Havaintoraportin täytettävät kentät on ennalta määritelty M-Files-järjestelmässä, josta sovellus lukee ja piirtää ne käyttöliittymään käyttäjäkohtaisesti.

Koska Xamarin.Forms-sovelluskehys oli alkuperäiselle sovelluskehittäjälle jo ennestään tuttu, se valikoitui sovelluksen kehittämistyökaluksi. Xamarin.Forms on Microsoftin kehittämä avoimen lähdekoodin sovelluskehys monialustaisten sovellusten kehittämiseen. Xamarin.Forms mahdollistaa kehittämisen iOS-, Android- ja Windows-alustoille niin, että lähes kaikki koodaus tehdään yhteiseen koodipohjaan, eikä alustakohtaista kehitystä tarvitse juurikaan tehdä.

1.2 Uusien toiminnallisuuksien toteuttaminen ennen julkaisua

Tällä hetkellä sovellukseen voi kirjautua vain käyttäjän M-Files-tunnuksella. Usein yrityksillä on kuitenkin käytössä käyttäjätilien hallintaan jokin palvelu, joka antaa työntekijöille mahdollisuuden kirjautua yhdellä käyttäjätilillä useisiin yrityksen käyttämiin palveluihin. Tällä palvelulla saatetaan hallita jopa tuhansia käyttäjätilejä, minkä takia toisen palvelun, tässä tapauksessa M-Files-palvelun,

erillinen käyttäjätilien hallinta olisi monille yrityksille kohtuutonta. Tämän takia sovelluksen kirjautumista on monipuolistettava lisäämällä Azure Active Directory -kirjautumismahdollisuus, eli käyttäjä pystyy tällöin kirjautumaan sovellukseen organisaation Microsoft-tunnuksella. Kirjautumisesta on myös tehtävä käyttäjäystävällisempi lisäämällä salasanan ja käyttäjätunnuksen laitekohtainen tallennusmahdollisuus, jotta käyttäjän ei aina tarvitse kirjoittaa käyttäjätunnusta ja salasanaa kirjautuessaan sisään. Tämä tarkoittaa myös sitä, että salasanan tallentamisen turvallisuus on varmistettava.

Toinen sovelluksesta puuttuva ominaisuus on QR-koodin lukeminen mobiililaitteella, ja sen liittäminen tehtyyn havaintoraporttiin. Joillakin aloilla laitteessa olevan QR-koodin lukeminen on tärkeä osa yrityksen omistuksessa olevien laitteiden huollon tai korjauksen ilmoitusprosessia. Tällainen toiminnallisuus myös säästää huomattavasti ilmoitukseen kuluvaa aikaa, sillä laitekoodin kirjoittamisen sijaan lukemalla QR-koodi ilmoitus voidaan kohdistaa nopeasti oikeaan laitteeseen tai laitteen osaan. Tämän ominaisuuden integraatio sovellukseen kasvattaisi sovelluksen potentiaalisten käyttäjien määrää.

Kun viimeiset tärkeät sovellusominaisuudet on toteutettu, sovellus julkaistaan Google Play - ja App Store -kaupoissa. Opinnäytetyössä tarkastellaan, mitä asioita ja vaiheita julkaisemiseen kuuluu ja miten Xamarin.Forms-sovelluksen julkaisu tapahtuu.

2 XAMARIN.FORMS-KEHITTÄMINEN

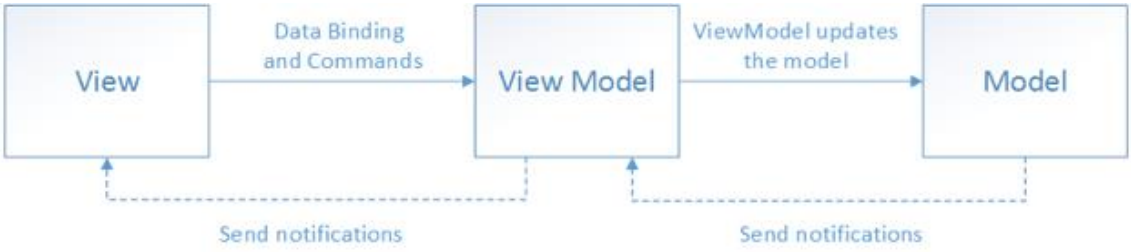
2.1 Xamarin.Forms

Xamarin.Forms on avoimen lähdekoodin UI framework eli käyttöliittymäsovelluskehys Android-, iOS- ja Windows-kehitykseen. Xamarin.Forms mahdollistaa sovelluskehityksen eri alustoille jaetulla koodilla niin, että lopputulos näyttää ja tuntuu natiivilta sovellukselta. Ajaessaan sovellusta Xamarin.Forms muuttaa käyttöliittymäelementit natiiveiksi alustarenderöijien avulla. Joissakin tapauksissa yksittäinen toiminnallisuus saattaa olla olemassa vain tietylle alustalle. Tällaisen toiminnallisuuden käyttöönotto tapahtuu Xamarin.Forms-sovelluksessa alustakohtaisia ominaisuuksia (platform-specifics) käyttämällä. (Microsoft 2020a.)

Xamarin.Forms-sovelluksen kehittäminen tapahtuu Visual Studio -kehitysympäristössä C#-kielellä (Microsoft 2021a). Visual Studio on integroitu kehitysympäristö koodin editointiin ja virheiden korjaukseen. Visual Studion sisäänrakennettu Debug- eli virheenjäljitystila mahdollistaa koodin ja sen muuttujien läpikäymisen lause kerrallaan. (Microsoft 2021b.) iOS-kehittämiseen tarvitaan lisäksi Mac-tietokone, jolle on asennettu Applen määrittämä macOS-käyttöjärjestelmän vähimmäisversio ja Xcode-ohjelma (Microsoft 2021c).

2.2 MVVM-malli

Xamarin.Forms-projektin rakenne koostuu oletuksena XAML-tiedostoista ja niiden code behind -tiedostoista. Vaihtoehto tälle rakenteelle on MVVM-malli (Kuva 1), eli Model-View-ViewModel-malli, jossa sovelluslogiikka toteutetaan ViewModel- ja Model-komponenteissa. Sovellukset, ja näin ollen myös niiden kooditiedostot, voivat helposti kasvaa niin laajoiksi, että vain XAML-tiedostoihin ja code behind -tiedostoihin perustuva rakenne voi haitata koodin ylläpidettävyyttä. MVVM-mallilla on mahdollista selkeästi erottaa sovelluslogiikka ja näkymälogiikka itsenäisiksi tiedostoiksi, jolloin sovellusta on helpompi kehittää, testata ja ylläpitää. (Microsoft 2021d.)



Kuva 1. MVVM-malli (Microsoft 2021d).

MVVM-mallin komponenteilla on kaikilla oma tehtävänsä ja erilainen suhde toisiinsa. ViewModel-komponentti on yhteydessä sekä View- että Model-komponenttiin, mutta View ja Model eivät kommunikoi keskenään. Näin rakennetulla mallilla on useita hyötyjä. Koska ViewModel-toimii View- ja Model-komponenttien välittäjänä, kehittäjän ei tarvitse tehdä muutoksia itse Model-komponenttiin. Tämä on tärkeää, sillä Model-komponentti sisältää usein tärkeää liiketoimintalogiikkaa, jota on vaikeaa tai riskialtista muuttaa. Tämä malli mahdollistaa myös helpomman yksikkötestauksen siten, että testaukseen ei tarvitse käyttää View-komponenttia eli näkymää, koska ViewModel-komponentilla on mahdollista toteuttaa täsmälleen sama toiminnallisuus kuin View-komponentilla. View-komponentin eristäminen ViewModel- ja Model-komponenteista hyödyttää myös siten, että näkymää on mahdollista muokata ilman varsinaiseen koodiin koskemista. ViewModel-komponentin logiikka toimii View-komponentin muokkauksesta riippumatta. Näin käyttöliittymäsuunnittelijat ja ohjelmoijat voivat työskennellä itsenäisesti ja samanaikaisesti eri komponenteissa. (Microsoft 2021d.)

2.3 Suorituskyvyn optimointi

Ennen sovelluksen julkaisua on tärkeää testata sen suorituskyky. Huonosti optimoitu suorituskyky vaikuttaa merkittävästi käyttäjän kokemukseen sovelluksesta. Xamarin.Formsin dokumentaatio tarjoaa useita ratkaisuja Xamarin.Forms-sovelluksen suorituskyvyn parantamiseen. (Microsoft 2020b.)

Sovelluksen suorituskykyyn vaikuttaa sen yleinen koko. Sovelluksen koon rajoittamiseen on useita keinoja. Suorituskyvyn optimoimiseksi sivujen layout-toteutuksessa on tärkeää, että sovellus ei sisällä turhia tai tyhjiä elementtejä, että elementtejä on mahdollisimman vähän ja että elementit, joita ei tietyllä hetkellä tarvita näkymässä, on piilotettu. Jos mahdollista, elementeille kannattaa määritellä staattinen koko, sillä antamalla sovelluksen laskea elementin koko itse eli määrittämällä elementin kooksi “auto” tarkoittaa, että ohjelman täytyy suorittaa ylimääräisiä laskelmia elementtien

asettelemiseksi. XAML-tiedostot sisältävät aina viittauksia erilaisiin kirjastoihin. Jos samoja viittauksia käytetään useassa eri XAML-tiedostossa, on hyvä käytäntö kerätä viittaukset yhteen lähdetiedostoon, jotta vältytään päällekkäisyyksiltä ja näin ollen lyhennetään myös läpikäytävän XAML-tekstin pituutta. (Microsoft 2020b.)

Edelleen pienentääkseen projektin kokoa kehittäjän tulee lisätä XAML-tiedostoihin komento XAML-kokoamiselle. XAML-kokoaminen tarkoittaa, että XAML-tiedostot tarkistetaan ja kootaan ohjelman kokoamisen yhteydessä, jolloin XAML-elementtien lataamis- ja välitysajat pienenevät. XAML-kokoamisen jälkeen XAML-tiedostot poistetaan, mikä pienentää täten projektin lopullisen kokoonpanon kokoa. Uudempien Xamarin.Forms-projektien template- eli mallitiedostoissa tämä ominaisuus on käytössä oletuksena. (Microsoft 2021e.)

MVVM-mallin mukaan toteutetun Xamarin.Forms-projektin olennainen osa on binding- eli sidosmuuttujien käyttö. Sidosmuuttujat mahdollistavat sovelluksen näkymän (View) ja toiminnallisuuden toteuttavan koodin (ViewModel) kommunikoinnin. (Microsoft 2020c.) Klassisen sidosmuuttujan käyttö voi kuitenkin olla sovelluksen suorituskyvylle kuormittavaa, sillä sidokset ratkaistaan ohjelman ajon aikana. Koottu sidosmuuttuja (compiled binding) tarjoaa vaihtoehdon klassiselle sidosmuuttujalle. Koottuja sidosmuuttujia käyttääkseen kehittäjän on otettava käyttöön XAML-kokoaminen ja sitten määritettävä näkymäelementille (VisualElement) x:DataType-ominaisuus. Kootut sidosmuuttujat ratkaistaan jo sovelluksen koonnin aikana, mikä parantaa suorituskykyä. Lisäksi tämä tapa on hyödyllinen kehittäjille, sillä jos ohjelmasta löytyy virheellisiä sidoksia, ne ilmoitetaan virheinä (build error). Koottua sidosmuuttujaa ei kuitenkaan voi määrittää elementille, joka käyttää source- eli lähdeominaisuutta, sillä lähdeominaisuutta ei voida määrittää ohjelman kokoamisen yhteydessä. (Microsoft 2021f.)

CustomRenderer-luokat ovat tärkeä osa monialustaisen sovelluksen kehitystä. CustomRenderer-luokka mahdollistaa sovelluksen alustakohtaisen mukauttamisen (Microsoft 2021g). Useimmat Xamarin.Forms-renderöintiluokat käyttävät OnElementChanged-metodia, jota kutsutaan, kun mukautettu Xamarin.Forms-ohjausobjekti luodaan renderöimään sitä vastaava natiivi objekti. Joissakin tapauksissa OnElementChanged-menetelmää voidaan kuitenkin kutsua useita kertoja, mikä voi heikentää ohjelman suorituskykyä. Koodiin on siis lisättävä lause, joka tarkistaa, onko elementille jo kerran sovellettu määritetyt ominaisuudet, missä tapauksessa niitä ei toteuteta uudestaan. (Microsoft 2020b.)

2.4 Poikkeusten käsittely

Jotta sovelluksen käyttö olisi mahdollisimman miellyttävää, on koodissa käsiteltävä mahdolliset virheet eli poikkeukset, joita sovelluksen ajon aikana voi tulla vastaan. Hyvin huomioon otetut poikkeustapaukset takaavat, että sovellus ei kaadu normaalin käytön aikana. C#-kielessä, kuten muissakin olio-ohjelmointikielissä, tämän voi tehdä try-catch-ilmaisulla. (Microsoft 2021h.)

Try-catch-ilmaisu tarkoittaa nimensä mukaisesti sitä, että try-osiossa koodi yrittää suorittaa sen sisällä olevat lauseet, kunnes koodi on suoritettu loppuun onnistuneesti tai tapahtuu poikkeus, joka “heitetään” (throw) catch-osioon, joka suorittaa poikkeustilanteelle määritetyn koodin. Halutessaan kehittäjä voi käsitellä erilaiset poikkeustapauksen eri tavoin, jolloin catch-osioita voi ilmaisussa olla useita. Tämä on myös suositeltu try-catch-käyttötapa. On kuitenkin mahdollista käsitellä kaikki mahdolliset poikkeukset yhdessä catch-osiossa. (Microsoft 2021i.)

3 ANDROID-SOVELLUKSEN JAKELU

Android-sovelluksen jakelun voi tehdä Visual Studiossa kahdella tavalla: Ad-Hoc-jakeluna ja Google Play -jakeluna. Ad-Hoc tarkoittaa paketin tallentamista esimerkiksi kovalevylle. Sovelluksen kehittäjä voi jakaa sovelluksen lähettämällä paketin esimerkiksi sitä testaavalle henkilölle, joka voi asentaa paketin suoraan Android-laitteelle ilman välikäsiä. Ad-Hoc-jakeluna tehdyn paketin voi myös jakaa esimerkiksi kehittäjän omilla verkkosivuilla. (Microsoft 2021j.) Verkkosivujen kautta suoritetussa jakelussa tulee ottaa huomioon mahdollinen rahaliikenne. Rahaliikennettä ei voi suorittaa Google Play -työkalun kautta, mikäli jakeluun ei käytä Google Play -kauppaa. (Android 2021a.)

Google Play -jakelu Visual Studion kautta puolestaan tarkoittaa allekirjoitetun paketin julkaisua Visual Studiosta suoraan Google Play -sovelluskauppaan. Ensimmäisellä julkaisukerralla sovelluksen vienti täytyy tosin tehdä manuaalisesti Google Play Consolen kautta, ennen kuin Visual Studion tarjoamaa jakelutapaa voi käyttää. Kun sovellusta julkaistaan Visual Studion kautta, luodaan ensin OAuth-valtuutus Google Play Developer -tilillä, jonka jälkeen tili rekisteröidään Visual Studiossa. (Microsoft 2021k.)

Jotta sovelluksen voi julkaista Google Play -kauppaan, kehittäjällä on oltava käytössään Google-kehittäjätili. Kehittäjätilin luonnin hinta on 25 dollaria. Jokaisen julkaistavan sovelluksen tulee olla allekirjoitettu salatulla avaimella, joka vanhenee 22. päivä syyskuuta vuonna 2033. (Microsoft 2021k.)

3.1 Android ja Google Mobile Services

Android Open Source Project eli AOSP, puhekielessä Android, on tuotantovalmis käyttöjärjestelmä mobiililaitteille ja samalla Googlen johtama avoimen lähdekoodin projekti. Tuotantovalmiilla tarkoitetaan tässä sitä, että Android on jo itsessään toimiva käyttöjärjestelmä, jonka päälle laitevalmistajien tarvitsee vain lisätä omat muutoksensa ja sovelluksensa. Android on avoimen lähdekoodin projekti, jotta vältettäisiin tilanne, jossa yksittäinen alan toimija voisi rajoittaa tai hallita muiden toimijoiden innovaatioita. Androidin muokattavissa oleva lähdekoodi ja sen dokumentaatio ovat siis kaikkien saatavilla. (Android 2021b.)

Google Mobile Services eli GMS on kokoelma Googlen sovelluksia ja ohjelmointirajapintoja (Application Programming Interface, API), jotka tukevat toiminnallisuuksien toimivuutta laitteiden yli. Pakettiin kuuluvia sovelluksia ovat muun muassa Google Search, Google Chrome, YouTube, YouTube Music, Google Play Store, Google Drive, Gmail ja Google Maps. (Android 2021c.) Jos kokoelma puuttuu Android-laitteesta, puuttuvat siitä näin ollen edellä mainitut sovellukset, ja lisäksi esimerkiksi yhteystietojen ja kalenterin synkronointi ei onnistu Google-tilin kautta. Tämä kokoelma ei ole osa Androidin avoimen lähdekoodin projektia, joten jos laitevalmistaja haluaa kokoelman käyttöön omiin laitteisiinsa, on sen haettava tähän lisenssi Googlelta. (Hildenbrand 2019.)

3.2 Xamarin.Android-projektin konfigurointi julkaisua varten

Kun sovellus on julkaisuvalmis, Microsoft ohjeistaa Xamarin.Android-sovelluksen julkaisua tietyin työvaihein. Ensimmäinen vaihe on sovelluksen ikonin määrittäminen. Joissakin sovelluskaupoissa, kuten Google Play -kaupassa, ikonin asettaminen on pakollista, ja ilman sitä sovellusta ei voida julkaista. Ikoni asetetaan Visual Studiossa Android Manifest -nimisessä osiossa Xamarin.Android-projektin asetuksissa. (Microsoft 2021j.)

Toinen vaihe on sovelluksen versiointi. Versioinnin avulla on mahdollista seurata, mikä versio sovelluksesta on käytössä, miten sovellus päivitetään ja milloin päivitys tulee suorittaa. Android-projektin versiota kuvaamaan käytetään kahdenlaista tietotyyppiä: version numeroa ja version nimeä. Version numero on kokonaisluku, joka yleensä alkaa numerosta 1 ja kasvaa joka koontiversion luonnilla. Numerolla ei ole yhteyttä version nimen kanssa, eikä sen ole suotavaa näkyä käyttäjälle. Android käyttää version numeroarvoa version seurantaan, eli numeron tarkistamalla Android tietää, tuleeko sovellus päivittää tarjolla olevaan uuteen versioon. Version nimi taas on merkkijono, joka näkyy käyttäjälle ja josta näkee, mikä versio sovelluksesta laitteessa on käytössä. Version nimi näytetään käyttäjille Google Play -kaupassa, mutta Android itse ei käytä sitä versionseurantaan. (Microsoft 2021j.)

Kolmantena vaiheena Xamarin.Forms-dokumentaatio mainitsee APK-paketin pienentämisen (Microsoft 2021j). Tältä osin dokumentaatio on kuitenkin vanhentunut, sillä Google Play on siirtynyt elokuussa vuonna 2021 käyttämään lähetettävänä pakettimuotona APK-pakettien sijaan Android App Bundle - eli AAB-formaattia. Google Play tekee AAB-paketista optimoituja ja laitekohtaisia

APK-paketteja, mikä mahdollistaa loppukäyttäjän lataaman APK-paketin koon pienentämisen. Ennen AAB-uudistusta kehittäjien oli luotava ja ylläpidettävä useita APK-paketteja eri laitteille. (Android 2021d.)

Seuraava vaihe on sovelluksen suojaaminen hyökkäyksiltä. Visual Studiossa testaamista helpottaa debug- eli virheenjäljitystila. Sovellusta julkaistaessa debug-tila täytyy poistaa käytöstä, sillä sen päälle jättäminen altistaa sovelluksen riskeille. Debug-tilan käytöstä poistamisen jälkeenkin sovellus voi silti olla altis hyökkäyksille. Microsoftin dokumentaatiossa suositellaan käyttämään Dotfuscator-nimistä työkalua tunnistamaan, yritetäänkö sovellusta käyttää rootatulla laitteella. (Microsoft 2021j.) Rootatulla laitteella hyökkääjän on mahdollista päästä käsiksi sovelluksen binääriin, takaisinmallintamaan (reverse engineer) sen, poimiakseen arkaluontoisia tietoja tai muokatakseen sovelluksen toimintaa (PreEmptive Solutions 2021).

Edellä mainittujen vaiheiden jälkeen on hyvä tarkistaa, että sovellus rakentuu onnistuneesti release - eli julkaisutilassa (Microsoft 2021j). Lisäksi Androidin sovelluksen julkaisuun liittyvässä dokumentaatiossa mainitaan julkaisua edeltävänä vaiheena log- eli lokikutsujen poistaminen (Android 2021a). Lokikutsut ovat kehittäjän koodissa tekemiä komentoja, joilla kehittäjä voi seurata suoritetun koodin kulkua ja toimivuutta. Lokikutsut tulee poistaa julkaistavan sovelluksen koodista, sillä niiden avulla myös mahdolliset hyökkääjät voivat seurata koodin suorittamista ja saadulla informaatiolla käyttää sovellusta luvatta. Pahimmassa tapauksessa lokikutsuihin voi olla kirjattu jopa salasanoja tai tokeneita, joiden ei missään nimessä tule päätyä ulkopuolisten tietoon. (Adasiewicz 2018.)

3.3 Sijainnin haku ja Google Maps

Sijainnin hakuun sovelluksessa on kaksi eri tapaa: etualalla ja taustalla tapahtuva haku. Molemmissa tavoissa sovelluksen tulee kysyä käyttäjältä lupa sijainnin hakuun. Etualalla tapahtuva sijainnin haku on kyseessä silloin, kun sijainti haetaan vain kerran tai hetkellisesti. Etualalla tapahtuva sijainnin haku on käyttäjälle näkyvää toimintaa. Esimerkiksi reittiohjeita antava sovellus, joka ohjaa käyttäjää tämän sijainnin perusteella, tai pikaviestisovellus, jolla sovelluksen käyttäjä jakaa hetkellisesti sijaintinsa toiselle käyttäjälle katsotaan etualalla tapahtuvan sijainnin hakua käyttäviksi sovelluksiksi. (Android 2021e.)

Taustalla tapahtuvaksi sijainnin hauksi katsotaan kaikki tilanteet, jotka eivät sisälly etualalla tapahtuvat sijainnin hakuun. Esimerkki taustalla tapahtuvasta sijainnin hausta sovelluksessa on perheille suunnattu sovellus, joka jakaa käyttäjänsä sijainnin jatkuvasti muille perheenjäsenille. (Android 2021e.) Taustalla tapahtuvaa sijainnin hakua ei saa käyttöön ilman erillistä hakemusta Googlelta, jossa on tarkkaan kerrottava sijainnin käyttötarkoitus sekä perustelut sijainnin jatkuvan seurannan tarpeelle (Google 2021a). Ennen Android 10 -versiota (API-taso 29) tätä hakemusta ei tarvinnut, sillä käyttäjän sijainnin hakuun antama lupa päti myös taustalla tapahtuvaan sijainnin hakuun (Android 2021e).

Jotta Google Maps -toiminnallisuutta voi käyttää sovelluksessa, kehittäjän on lisättävä koodiin viittaus API-avaimeen. API-avaimen hankkiakseen rekisteröitynyt Google-käyttäjä ensin hankkii sovelluksen allekirjoitukseen käytetyn SHA-1-sormenjäljen ja sitten luo uuden projektin Googlen API-konsoliin. Kun projekti on luotu, sille luodaan API-avain credentials- eli pääsytiedot-osiossa. Avain rajoitetaan toimimaan vain määritetyissä sovelluksissa SHA-1-sormenjäljen ja sovelluksen paketin nimen perusteella. (Microsoft 2021l.)

3.4 Android-julkaisualustat

Android-sovelluksen voi julkaista useassa eri sovelluskaupassa (Android 2021a). Google Play on selvästi laajin alusta sovellusten jakamiseen (Statista 2021). On kuitenkin laitevalmistajia, joiden laitteilta ei löydy Google Play -sovelluskauppaa. Esimerkiksi Huawein Android-laitteille ei voi asentaa Google-ohjelmia. Tämä johtuu Yhdysvaltojen asettamasta kauppakiellosta. Huawei laitteisiin sovellukset ladataan Huawei AppGallery-nimisestä sovelluskaupasta. (Grabham 2021.) Toinen laitevalmistaja, joiden tablettilaitteilta ei löydy Google Play -sovelluskauppaa on Amazon. Tarkkaa syytä tälle Amazon ei ole kertonut, mutta syy on todennäköisesti kilpailullinen, sillä Amazon tarjoaa käyttäjilleen omaa sovelluskauppapalvelua nimeltä Amazon Appstore. (Nield 2019.)

Vertaamaan eri sovelluskauppojen suosiota voidaan tarkastella kauppojen tarjoamien sovellusten ja niiden aktiivisten käyttäjien määrää. Google Play -sovelluskaupasta sovelluksia löytyy noin 3.48 miljoonaa, ja sillä on aktiivisia käyttäjiä yli miljardi (Android 2021f, Statista 2021). Amazon Appstore -kaupassa taas sovelluksia on vain noin 460 tuhatta (Statista 2021). Amazon Appstoren aktiivisten käyttäjien määrästä ei löydy tietoa. Huawei AppGallery -kaupassa on sovelluksia noin 55 tuhatta ja aktiivisia käyttäjiä noin 450 miljoonaa (Amit 2020). Google Play - ja Amazon Appstore -kauppojen

tiedot ovat vuoden 2020 toiselta kvartaalilta ja Huawein luvut helmikuulta 2020 (Amit 2020, Statista 2021).

4 IOS-SOVELLUKSEN JAKELU

iOS-sovelluksen jakeluun on olemassa neljä vaihtoehtoa: App Store -jakelu, in-house-jakelu, ad hoc -jakelu sekä yritys- ja opetusinstituutioiden kustomoitu sovelluksen jakelu. In-house-jakelussa sovellus on mahdollista jakaa sisäisesti organisaation jäsenille. Tätä jakelutapaa varten kehittäjän tai sovelluksen kehittäjäyrityksen on oltava Apple Developer Enterprise -ohjelman jäsen. Ad-hoc-jakelussa taas sovelluksen voi jakaa testaustarkoitukseen maksimissaan sadalle iOS-laitteelle ilman sen viemistä sovelluskauppaan. Apple sallii myös sovellusten mukautetun jakelun tilanteissa, joissa sovellus on tarkoitettu tietyn yrityksen tai opetusinstituution käyttöön. Suosituin tapa jakaa iOS-sovellus on kuitenkin App Store -kauppaan jakelu. (Microsoft 2021m.)

4.1 Apple Developer -ohjelma

Sovelluksen testaamiseen fyysisellä laitteella ja myöhemmin jakelua varten kehittäjällä on oltava Applen myöntämä provisioning profile eli säännöstelyprofiili. Sovelluksen testaamiseen on mahdollista käyttää ilmaista säännöstelyprofiilia, mutta sovelluksen jakelua varten kehittäjän tai yrityksen on oltava osa Apple Developer -ohjelmaa (Apple Developer Program). (Microsoft 2021n.) Ohjelman jäseneksi liittymisen vuosimaksu on tällä hetkellä 99 dollaria tai sama määrä paikallista valuuttaa (Apple 2021a).

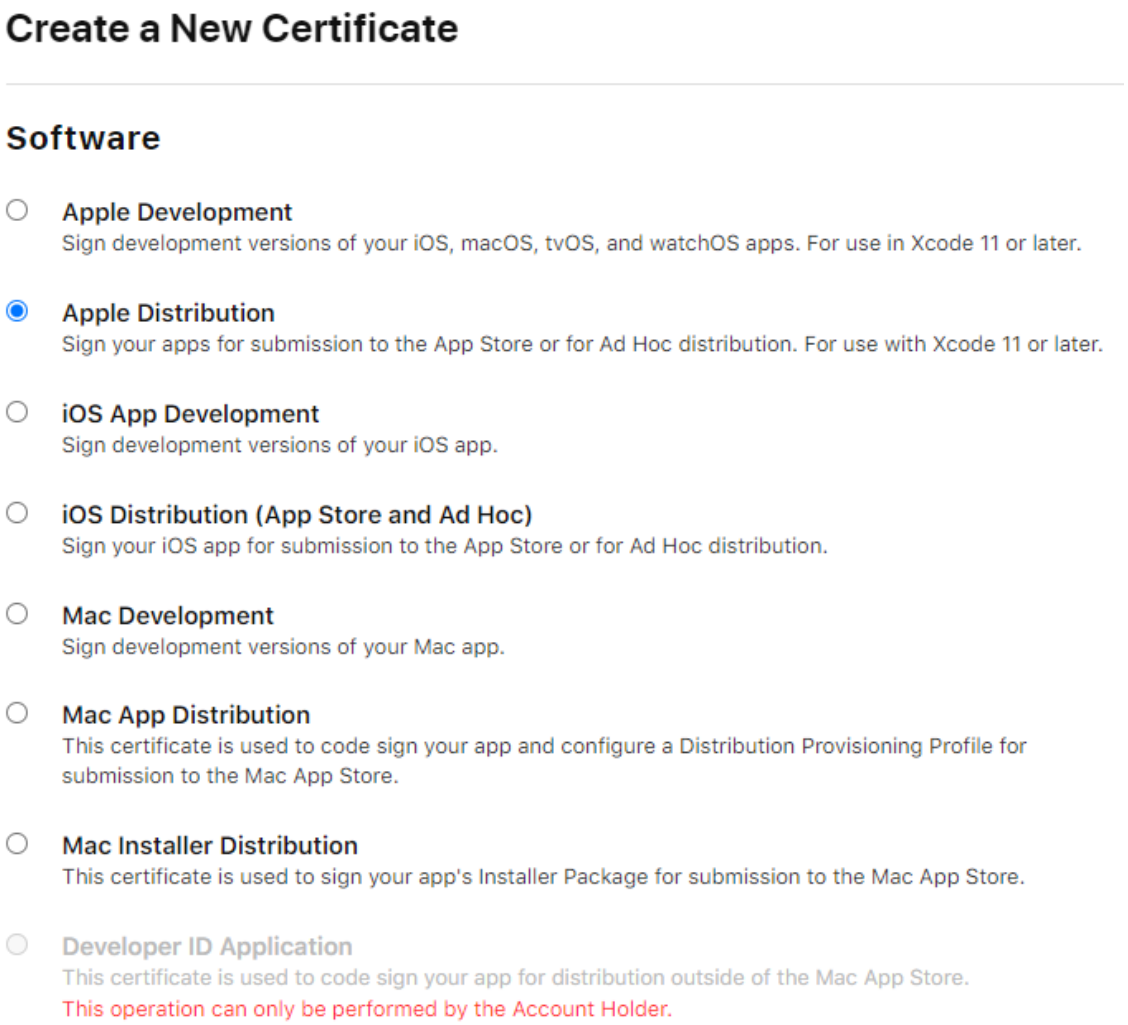
Apple Developer -ohjelma tarjoaa kehittäjän käyttöön kaikki työkalut, joita iPhone, iPad, Mac, Apple TV ja Apple Watch -sovellusten kehittämiseen tarvitsee. Lisäksi ohjelma mahdollistaa sovellusten julkaisun ja jakelun. Ohjelman kautta kehittäjillä on myös pääsy esimerkiksi Apple-kehittäjien verkostointitapahtumiin ja erilaisiin tukipalveluihin. (Apple 2021b.)

4.2 App Store -jakelu

App Store -jakelua varten täytyy rakentaa erityinen distribution- eli jakeluprofiili. Jakeluprofiilia varten tarvitaan jakelusertifikaatti ja sovellustunnus. Kehittäjä voi halutessaan käyttää kehittäjäprofiilia varten aikaisemmin luotua sovellustunnusta tai vaihtoehtoisesti luoda uuden sertifikaatin ja tunnuk-

sen. Jakeluprofiili mahdollistaa sovelluksen digitaalisen allekirjoituksen, mikä puolestaan mahdollistaa sovelluksen asennuksen iOS-laitteille. Profiilin rakentaminen tapahtuu Apple Developer -portaalin kautta. (Microsoft 2021o.)

Jakelusertifikaatti luodaan Apple Developer -portaalin Certificates- eli sertifikaatit-osiossa (Microsoft 2021o). Sertifikaatin voi luoda vain tilillä, jolla on Account Holder - tai Admin-tason oikeudet. Kaikki kehittäjätiimin jäsenet voivat kuitenkin käyttää luotua sertifikaattia. (Apple 2020a.)

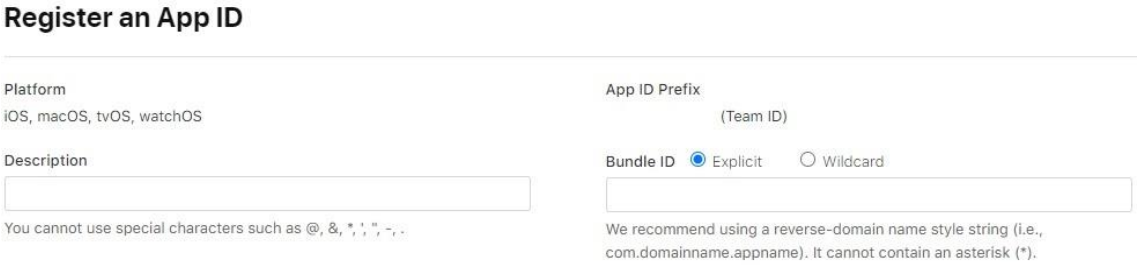


Kuva 2. Jakelusertifikaatin hankkiminen.

Uuden jakelusertifikaatin tyypiksi valitaan “Apple Distribution” eli Apple-jakelu (Kuva 2) (Apple 2020b). Tämän jälkeen käyttäjä ohjataan luomaan allekirjoituspyyntö sertifikaattia varten. Allekirjoituksen pyyntö tapahtuu Mac-tietokoneen Keychain Access - eli Avainnipun käyttö -ohjelmalla.

Sertifikaatin luomisen jälkeen se ladataan Apple Developer -portaaliin sekä asennetaan Mac-tietokoneelle. (Microsoft 2021o.)

Sovellustunnuksen hankkiminen (Kuva 3) tapahtuu saman työkalun kautta Identifiers- eli tunnisteet-kohdasta. Lisättäessä uutta sovellusta sille määritellään nimi ja Bundle-tunnus muodossa: “com.[domain nimi].[sovelluksen nimi]” tai “com.[domain nimi].*”. Tämän jälkeen sovellukselle valitaan sen mahdollisesti tarvitsemat palvelut, kuten Maps-karttapalvelu tai Avainnippu-palvelu salasanojen ja tilitietojen hallintaan. (Microsoft 2021o.)



Kuva 3. Sovellustunnuksen rekisteröinti.

Kun sertifikaatti ja sovellustunnus on hankittu, luodaan jakeluprofiili. Jakeluprofiilia luodessa valitaan, miten sovellus halutaan julkaista. Jakeluprofiiliin liitetään aikaisemmin hankitut sertifikaatti ja sovellustunnus. Lopuksi profiilille annetaan nimi. (Microsoft 2021o.)

Kun sovellus halutaan viedä App Store -jakeluun, projektin julkaisuversio rakennetaan viimeisen kerran. Visual Studiossa projektin iOS-ominaisuuksien valikossa “Bundle Signing” -kohdasta valitaan säännöstelyprofiiliksi luotu jakeluprofiili. Kun jakelu on suoritettu onnistuneesti, sovellus konfiguroidaan Applen sovelluskauppaportaalissa. (Microsoft 2021o.) Microsoftin dokumentaatiossa tästä portaalista käytetään nimeä iTunes Connect, mutta vuodesta 2018 lähtien App Store -julkaisu on tapahtunut App Store Connect -nimisen työkalun kautta (Apple 2018). Kun sovellus on ladattu App Store Connect -portaaliin ja kehittäjä on lisännyt App Store -kauppaa varten tarvittavat tiedot, suoritetaan sovellukselle tarkastus Applen toimesta. Sovelluksen on läpäistävä Applen sovellusarviointi, jotta se on mahdollista julkaista App Store -kauppaan. (Microsoft 2021o.)

4.3 App Store -julkaisun ohjeistus

App Store -kauppaan julkaistavien sovellusten on täytettävä tietyt kriteerit. Tärkeimmät kriteerit sovellukselle ovat, että sovelluksen ominaisuudet vastaavat niitä toiminnallisuuksia, jotka sen kuvaukseen on listattu, ja että sovellus ei kaadu normaalikäytön puitteissa. (Microsoft 2021p.) Näiden kriteerien lisäksi Apple listaa useita tärkeitä ohjeita sovellusten julkaisuun dokumentissa App Store Review Guidelines (Apple 2021c). Dokumentti on mittava, sillä sen on tarkoitus tarjota ohjeistusta lukuisten erilaisten sovellusten julkaisuun. Tässä opinnäytetyössä luetellaan vain työn aiheena olevan sovelluksen kannalta oleelliseksi katsotut ohjeistukset.

Sovelluksessa käytettävän kielen on oltava asiallista, eikä se saa sisältää muun muassa loukkaavaa tai syrjivää sisältöä. Myöskään sovelluksen kuva- tai videomateriaali ei saa sisältää vahingollista sisältöä. Sovellus ei saa vaarantaa käyttäjän fyysistä turvallisuutta. Vaarantamiseksi katsotaan esimerkiksi lääketieteelliseen käyttöön tarkoitetut sovellukset, jotka tarjoavat käyttäjälle virheellistä tietoa, tai sovellukset, jotka kannustavat huumausaineiden käyttöön. (Apple 2021c.)

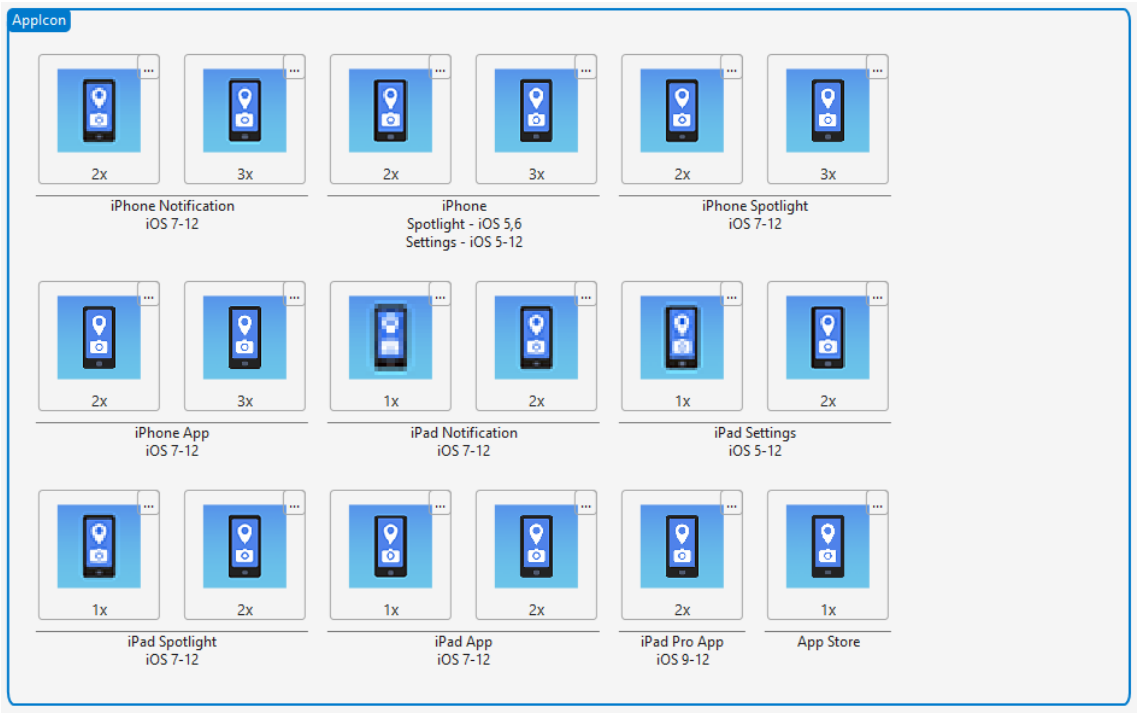
Julkaistavan sovelluksen on oltava viimeistelty, eli siitä on poistettava esimerkiksi paikanpitäjätekstit ja tyhjät verkko-osoitteet. Sovelluksen on myös oltava testattu, jotta mahdollisimman monet virheet ja kaatumistilanteet on poistettu. Sovellus on suunniteltava ja toteutettava niin, että se ei vahingoita sitä käyttävää laitetta esimerkiksi käyttämällä akkua liian nopeasti tai lämmittämällä laitetta liikaa. (Apple 2021c.)

Jos sovellus tarjoaa kolmannen osapuolen kirjautumisvaihtoehtoja, Applen dokumentaation mukaan sen on tällöin tarjottava käyttäjälle mahdollisuus kirjautua myös Apple-tunnuksilla. Poikkeus tähän ohjeistukseen on kuitenkin, jos sovellus on tietyn kolmannen osapuolen asiakas tai kumppani ja näin ollen käyttää kirjautumiseen tämän kolmannen osapuolen tilejä. (Apple 2021c.)

4.4 Xamarin.iOS-projektin konfigurointi App Store -julkaisua varten

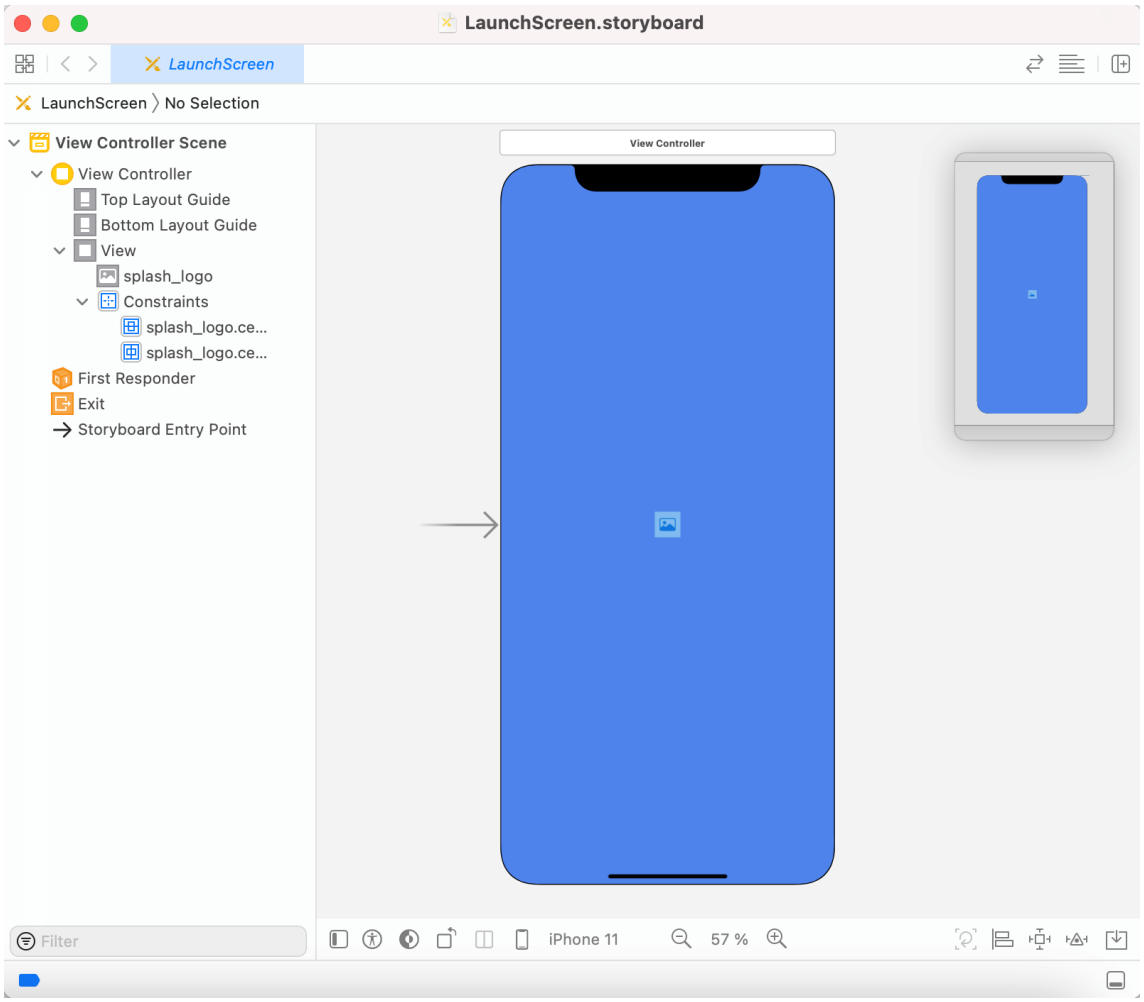
App Store -sovelluskaupan vaatimat ikonit ja aloitusnäyttö määritellään Xamarin.iOS-projektissa Visual Studiossa. Sovellusikonit lisätään oletuksena Assets.xcassets-tiedoston AppIcon-nimiseen kuvasarjaan (Kuva 4), ellei kehittäjä ole määritellyt ikonikuvasarjalle projektissa jotain muuta nimeä.

AppIcon-kuvasarjaan lisätään useita eri kokoisia ikonikuvia, joita iOS-laitteet käyttävät eri tilanteissa. (Microsoft 2021q.)



Kuva 4. AppIcon-kuvasarja.

Aloitusnäytön toteuttaminen (Kuva 5) tapahtuu Mac-tietokoneella Xcode-ohjelmalla. Visual Studio 2019 -versiosta lähtien iOS .storyboard -tiedostoja ei ole voinut muokata Windows-ympäristössä. Aloitusnäytön muokkaamiseen käytetään Xcode Interface Builder -työkalua. Työkalulla näytölle voi lisätä elementtejä, kuten kuvia, muokata niiden sijaintia ja kokoa ja määrittää, miten ne asettuvat laitteiden näytöille. On suotavaa, että asettelussa otetaan huomioon iOS-laitteiden erilaiset näyttökoot. (Microsoft 2021r.)



Kuva 5. Aloitusnäytön toteuttaminen Xcode-ohjelmalla.

Vuoden 2020 huhtikuusta lähtien Apple ei ole julkaissut sovelluksia, jotka käyttävät käytöstä poistettua UIWebView-ohjelmointirajapintaa. Xamarin.Formsin uudet versiot käyttävät uutta WKWebView-rajapintaa, mutta joissakin tapauksissa Xamarin.Forms-projektista voi vielä löytyä viittauksia vanhaan rajapintaan. Sen lisäksi, että kehittäjän on poistettava koodista kaikki viittaukset UIWebView-ohjelmointirajapintaan, viittaukset on hyvä poistaa myös iOS-projektin ominaisuudet-valikon kautta lisäämällä komennon “--optimize=experimental-xforms-product-type” mtouch-argumentteihin. (Microsoft 2021s.)

5 SOVELLUKSEN JULKAISUN TOTEUTTAMINEN

5.1 Sovelluksen ikonin ja aloitusnäytön suunnittelu ja toteutus

Jokaisella sovelluskaupalla on erilaisia määrittelyjä käyttämiensä ikonien suhteen. Google Play - ja App Store -kauppojen ohjeistukset ikonien tekoon ovat kattavat ja määrittävät tarkasti, millainen ikonin tulisi olla mittasuhteiltaan, kooltaan ja sisällöltään. App Store - ja Google Play -ikonien suuntaviivat eivät juurikaan poikkea toisistaan. Sovellusta varten suunniteltiin ikoni, jota voitaisiin käyttää molemmissa sovelluskaupoissa.

Sekä App Store - että Google Play -sovelluskaupoissa ikoni on lopulliselta muodoltaan pyöreäkulmainen neliö. Ikoni kuitenkin lähetetään palveluihin neliönä ilman pyöristettyjä kulmia, ja sovelluskaupat asettavat itse kuvalle muodon määrittävän maskin. Näin taataan ikonin yhdenmukaisuus kullakin alustalla. Google Play -kauppaa varten riittää yksi kooltaan 512 pikseliä leveä ja korkea ikoni, mutta App Store -kauppaa varten on tarjottava useita eri kokoisia ikoneja, joista suurin on kooltaan 1024 x 1024 pikseliä. (Android 2021g, Apple 2021d.) Ikoni ei mielellään saa sisältää läpinäkyvyyttä, eikä sillä saa olla varjoa (Android 2021g).

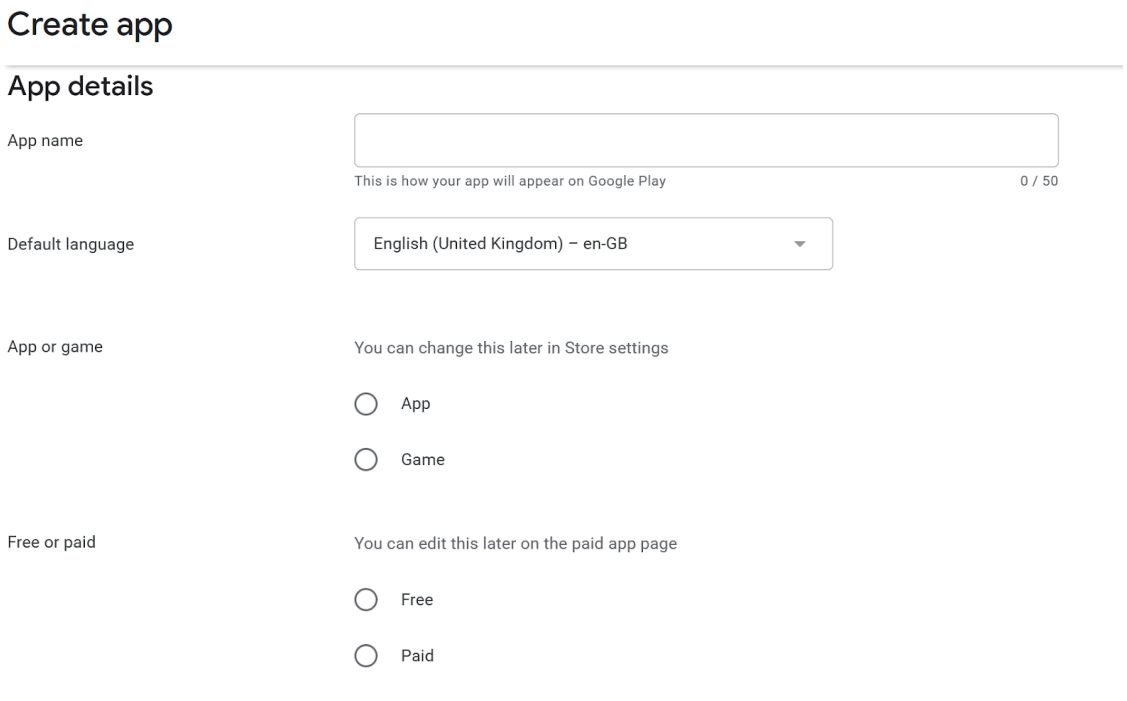
Kuva 6. Ikonivaihtoehtoja.

Toimeksiantajan havaintojen raportointisovellukselle oli jo olemassa työskentelyvaiheessa käytetty ikoni. Koska tämän ikonin elementtien katsottiin sopivan hyvin sovellukselle ottaen huomioon sovelluksen käyttötarkoitus, sitä käytettiin pohjana suunniteltaessa varsinaista sovelluskauppaikonia. Ikonin värit otettiin toimeksiantajan logon värimaailmasta. Erilaisia ikonivaihtoehtoja luotiin ensin kahdeksan (Kuva 6), joista valittiin yksi lopullinen versio ensin rajaamalla eniten silmää miellyttävät versiot ja sitten testaamalla niitä laitteessa. Ikonia valitessa on tärkeää huomioida sen lopullinen koko laitteen näytöllä. Liian pienet elementit tai monimutkainen sisältö eivät erotu pienestä ikonikuvasta. (Apple 2021d.)

Kun sovelluksen ikoni oli toteutettu, suunniteltiin sen pohjalta myös sovelluksen aloitusnäyttö, josta käytetään englanniksi sanoja “launch screen” tai “splash screen”. Aloitusnäyttö näytetään käyttäjälle ennen kuin varsinainen sovellus avautuu. Aloitusnäytön tarkoitus on parantaa käyttäjäkokemusta antamalla vaikutelman siitä, että sovellus avautuu nopeasti, vaikka tosiasiassa sovellus ei avaudu heti käyttäjän painaessa käynnistysikonia. (Apple 2021e.)

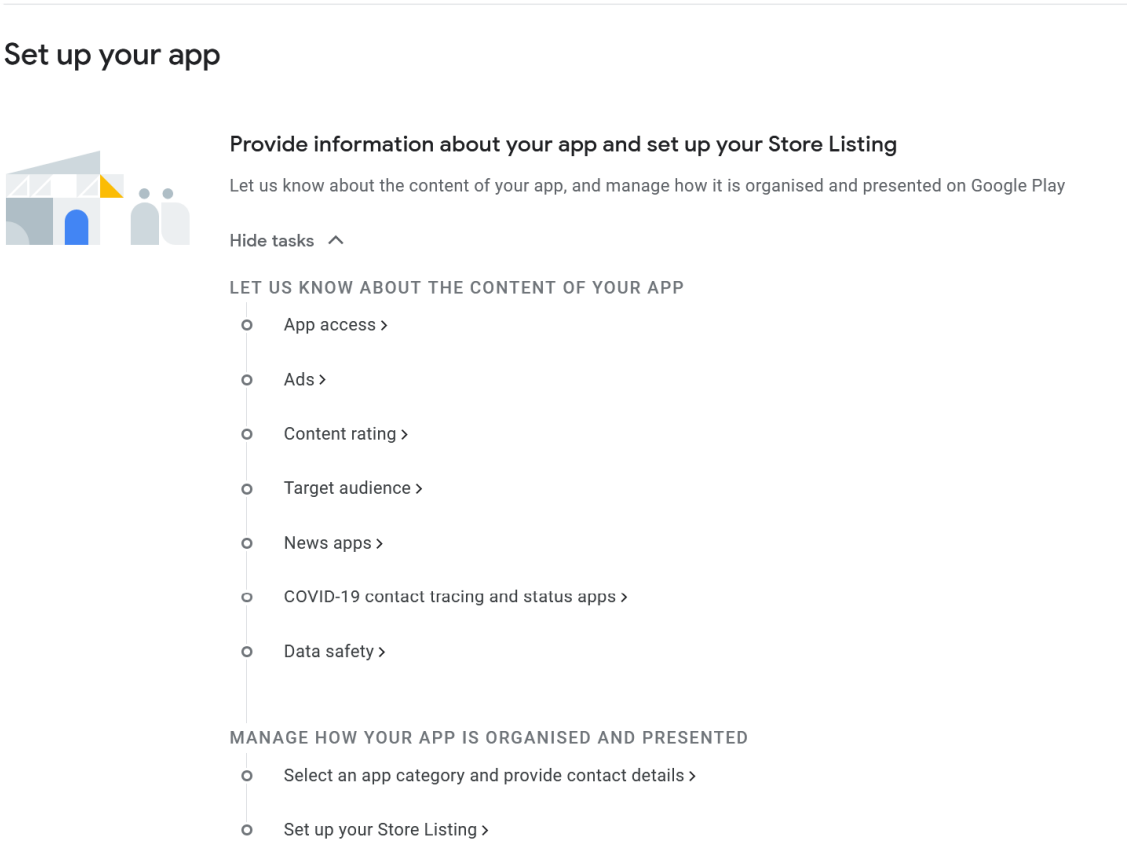
5.2 Google Play Console

Kun sovellus on Visual Studiossa konfiguroitu Androidin ohjeiden mukaiseksi, se voidaan julkaista Google Play -kauppaan. Sovelluksen vienti sovelluskauppaan tapahtuu Google Play Console -portaalin kautta. Uuden sovelluksen julkaisu aloitetaan painamalla portaalissa kohtaa “Create app” eli luo sovellus (Kuva 7). (Google 2021b.)



Kuva 7. Sovelluksen luonti Google Play Consolessa.

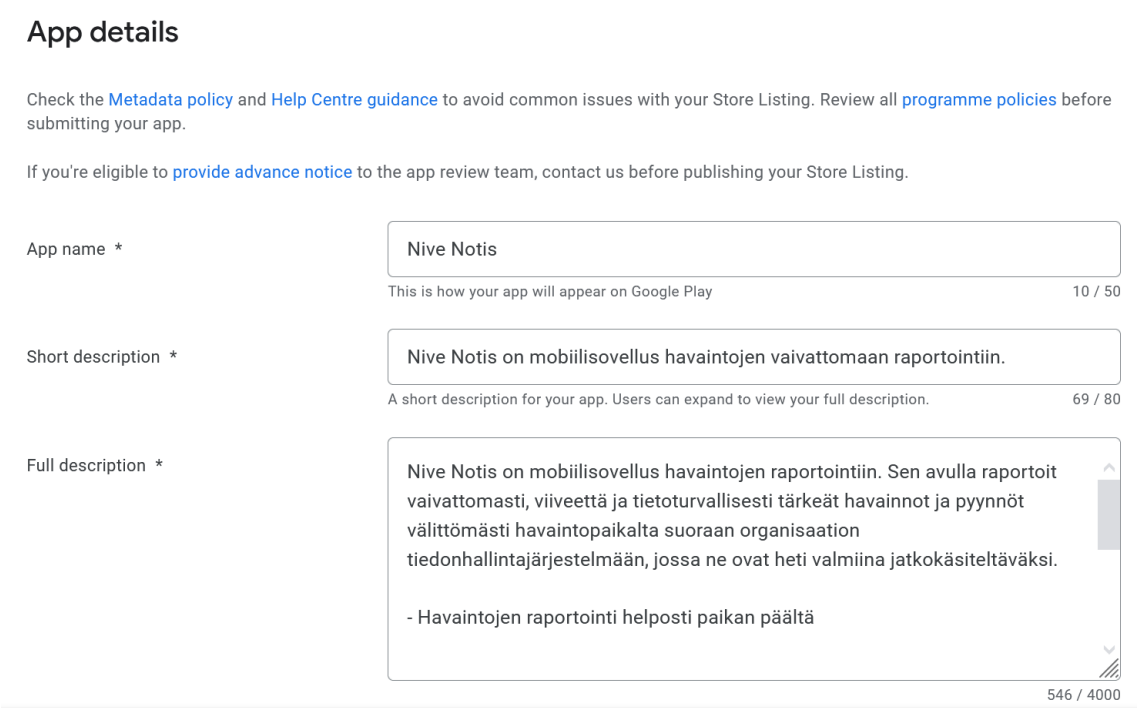
Uutta sovellusta Google Play Consolessa luodessa portaali kehottaa valitsemaan, onko sovellus maksullinen vai ei. Tämän valinnan voi muuttaa vielä ennen julkaisua, mutta julkaisun jälkeen ilmaista sovellusta ei voi enää muuttaa maksulliseksi. Kun pyydetyt tiedot on syötetty ja uusi sovellus on luotu, siirrytään Dashboard-sivulle. Sivulla opastetaan julkaisun etenemistä (Kuva 8), sen vaatimuksia sekä testiversioiden julkaisua.



Kuva 8. Google Play Consolen kokoelma julkaisua edeltävistä vaiheista.

Ennen kuin sovelluksen voi julkaista tuotantoon, se on testattava vähintään yhdellä testaustavalla. Erilaisia testaustapoja ovat sisäinen, suljettu ja avoin testaus. Sisäinen testaus on maksimissaan sadalle testaajalle tehtävä testausmuoto, jossa julkaisija itse jakaa linkin valitsemilleen testaajille. Sisäinen testaus on myös testausmuodoista nopein, sillä siinä sovellusta ei tarkisteta Googlen toimesta etukäteen, toisin kuin kaikissa muissa testausmuodoissa. Suljetussa testauksessa testaajien määrä on moninkertainen sisäiseen testaukseen verrattuna, ja tarkkaa rajaa testaajien määrälle ei ole annettu. Suljetun testausmuodon ero sisäiseen testaukseen on testaajien määrän lisäksi mahdollisuus lisätä testaajia Google Group -nimisen työkalun kautta. Avoimessa testausmuodossa taas sovelluksen testiversio julkaistaan Google Play -kaupassa kaikkien halukkaiden testattavaksi. Testaajat voivat antaa palautetta sovelluksesta jokaisessa testausmuodossa heille annetun linkin tai sähköpostin kautta. Testiversioita voi julkaista useita, ja julkaisu voi tapahtua yhtä aikaa muiden testiversioiden ja tuotantoversion julkaisun kanssa. Julkaisijan on otettava huomioon, että kun sovellus julkaistaan ensimmäistä kertaa millä tahansa testaustavalla tai kun testattavaa sovellusversiota muokataan, voi testattavan sovelluksen toimimiseen mennä jopa useampi tunti. (Google 2021c.)

Seuraavaksi sovellukselle tulee määrittää Google Play -kauppaan tarvittavat tiedot sovelluksen sisällöstä (Kuva 9). Julkaisijan tulee määrittää sovelluksen käytön rajoitukset, eli tarvitseeko käyttäjä erityisiä käyttäjätunnuksia sovelluksen käyttöön tai toimiiko sovellus esimerkiksi vain tietyssä sijainnissa. Mikäli rajoituksia on, Googlelle on järjestettävä mahdollisuus testata sovellusta näiden rajoitusten puitteissa, esimerkiksi tarjoamalla testaukseen tarkoitettu käyttäjätili. Julkaisijan on kerrottava, käyttääkö sovellus kolmannen osapuolen mainoksia, sekä täytettävä kysely sovelluksen sisällöstä, jotta sille osataan määrittää oikea ikäraja. Sovellukselle määritetään myös kohdeyleisö, kategoria ja avainsanat. Kaikki näistä auttavat käyttäjiä löytämään sovelluksen Google Play -kaupassa. Tämän lisäksi sovelluksen myymälätietoihin kuuluu sovelluksen nimi, lyhyt kuvaus sovelluksesta sekä sovelluksen täysi kuvaus, missä kerrotaan mahdollisimman tarkkaan sovelluksen tarkoitus ja sen päätoiminnallisuudet.



Kuva 9. Myymälätiedot-osio Google Play Consolessa.

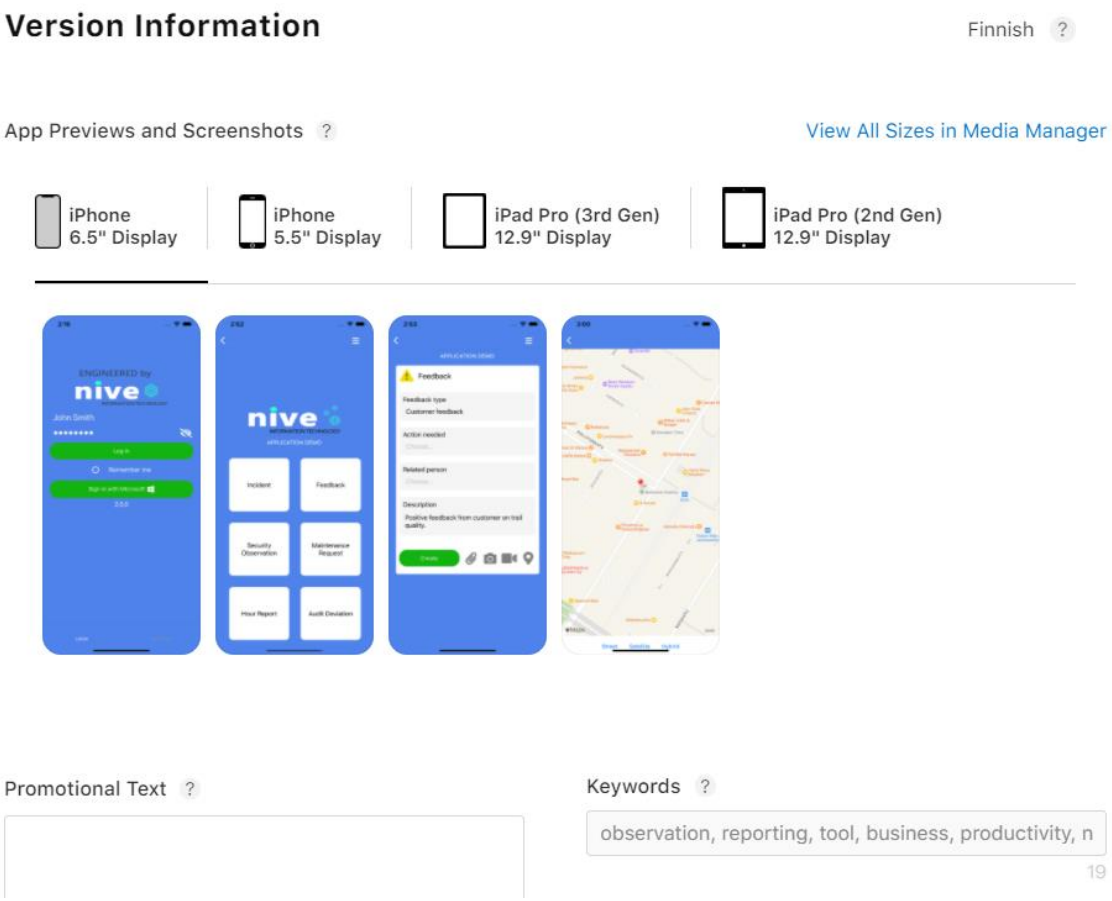
Sovellukselle lisätään myös sovellusikoni, näyttökaappauksia, ominaisuuskuva sekä halutessa video sovelluksesta. Ominaisuuskuva on huomiota ja kiinnostusta herättävä kuva, jonka tarkoitus on ilmaista sovelluksen tarkoitusta ja tärkeimpiä ominaisuuksia. Sitä käytetään myös videon esikatselukuvana, mikäli julkaisija on sellaisen lisännyt. (Apptamin 2021.)

Viimeinen kohta sovelluksen tietojen määrittelyssä koskee tietoturvallisuutta. Erillisessä lomakkeessa selvitetään, mille asioille sovellus tarvitsee lupaa käyttäjältä. Käyttäjältä lupa täytyy kysyä esimerkiksi laitteen kameran, äänitallenteiden tai tiedostojen käyttöön. Jos sovellus kerää tietoa, julkaisijan tulee tarkasti määrittää mihin kerättyä tietoa käytetään, jaetaanko sitä kolmansille osapuolille ja onko tieto suojattua sovelluksessa ja sitä siirrettäessä. Tämän jälkeen annetaan sovelluksen julkaisseen yrityksen yhteystiedot ja valitaan, annetaanko Google Playlle lupa mainostaa sovellusta Google Playn ulkopuolella.

Kun sovelluksen tiedot on määritetty, sovellus voidaan julkaista sovelluskaupassa. Julkaisija valitsee, mihin maahan tai maihin sovellus julkaistaan. Tämän jälkeen valitaan, hyväksytäänkö Googlen tarjoama allekirjoitusavain sovellukselle. Jos sovellus julkaistaan AAB-pakettina, julkaisijan on pakko hyväksyä Googlen allekirjoitusavain. Lopuksi sovellukselle määritetään julkaisutiedot (release notes) sekä julkaisunimi, jonka tarkoitus on erottaa eri julkaisut toisistaan. Julkaisutietojen tarjoaminen ei ole pakollisia, mutta päivittäessä sovellusta on hyvä kertoa, miten sovellusta on päivitetty.

5.3 App Store Connect

IOS-sovelluksen julkaisu tapahtuu App Store Connect -palvelun kautta. App Store Connect -portaalissa voi julkaista sovelluksen, päivittää sitä sekä seurata esimerkiksi sovelluksen analytiikkaa ja myyntiä julkaisun jälkeen. Sovelluksen ensimmäistä versiota julkaistaessa uusi sovellus lisätään Apps-välilehdellä painamalla +-ikonia. Avautuvassa ikkunassa määritellään julkaistavan sovelluksen nimi, joka näkyy App Store -kaupassa, ja lisätään sille aikaisemmin luotu Bundle-tunnus. Kun uusi sovellus on lisätty, käyttäjälle avautuu App Store -välilehti, jossa kaikki App Store -kaupassa näkyvä tieto määritellään (Kuva 10).



Kuva 10. Sovelluksen tietojen määrittely App Store Connect -portaalissa.

App Store -kauppaa varten julkaisijan on tarjottava vähintään yksi mutta mielellään useampi näytökaappaus sovelluksesta sovelluskauppaa varten. Lisäksi julkaisija voi tarjota videoita sovelluksen käytöstä tai muuta mainosmateriaalia. Sovellukselle määritetään kuvausteksti ja mahdollisesti lyhyt mainosteksti, avainsanat ja linkki sivulle, josta sovelluksen käyttäjän on mahdollista pyytää sovellustukea. Nämä tiedot julkaisija täyttää App Store Connect -palvelun App Store -välilehden etusivulla. Tälle sivulle ladataan myös varsinainen sovellus, joka halutaan julkaista. App Store Connect hakee ikonin App Store -kauppaa varten ladatun sovelluksen asset-tiedostoista. Jos julkaistava sovellus toimii vain sisäänkirjautumisella, julkaisijan on tarjottava sovellukselle käyttäjätili, jolla Applen on mahdollista suorittaa sovelluksen testaus ennen sen julkaisua.

App Information -sivulla julkaisija määrittää sovelluksen lokalisaatiotiedot, eli esimerkiksi sovelluksen nimen toisella kielellä, mahdollisesti useille alueille, sen ensisijaisen ja toissijaiset kielet sekä kategoriat, joilla sovellus on mahdollista löytää sovelluskaupasta. Sovelluksen ikärajoitus määritetään vastaamalla lyhyeen kyselyyn. Sovelluskauppaa varten julkaisijan on määritettävä End-user license agreement (EULA) eli loppukäyttäjän lisenssisopimusehdot. Julkaisijan on mahdollista

käyttää omaa sopimusehtodokumenttia tai Applen tarjoamaa vakiolisenssisopimusta. Sovelluksen hinta asetetaan Pricing and Availability -sivulla. Jos sovelluksesta halutaan tehdä maksullinen, on tehtävä Applen kanssa maksullisten sovellusten sopimus (Paid Applications Agreement).

Sovelluksen yksityisyysasetukset asetetaan App Privacy -sivulla. Sovellusta ladatessa käyttäjän tulee tietää, kerätäänkö hänestä tietoja, siirretäänkö tietoja kolmansille osapuolille ja voidaanko tietoja yhdistää käyttäjän identiteettiin. App Privacy -sivulta löytyvällä lomakkeella valitaan ensin, mitä tietoja sovelluksella kerätään. Sitten jokaiselle kerätylle tietotyypille määritellään, mitä tarkoitusta varten tietoa kerätään, voidaanko kerättyä tietoa yhdistää käyttäjän identiteettiin ja lopuksi, käytetäänkö kerättyä tietoa mainostamistarkoitukseen.

Koska App Store -kaupan palvelimet sijaitsevat Yhdysvalloissa, sitä pidetään Yhdysvaltojen vientituotteena ja näin ollen siihen sovelletaan Yhdysvaltain vientilakeja. Julkaisijan on siis otettava huomioon Yhdysvaltojen salausohjelmia koskeva vientilaki. Jos julkaistava sovellus sisältää salausta missään muodossa, tämä katsotaan salausohjelmiston vienniksi, ja siksi siihen sovelletaan Yhdysvaltojen vientiä ja muiden maiden tai alueiden tuontia koskevia vaatimuksia. Salauksen käyttö tarkoittaa esimerkiksi kutsujen tekemistä suojatulla kanavalla, kuten HTTPS, tai salausalgoritmien käyttöä. Yhdysvaltojen vientivaatimuksiin on kuitenkin olemassa poikkeuksia, jotka vapauttavat julkaisijan vaatimuksien noudattamisesta. Tällaiseksi poikkeukseksi katsotaan esimerkiksi tilanne, jossa sovellus käyttää matalan tason salausta. On julkaisijan vastuulla lukea tarkasti Salaus- ja vientihallinnon määräykset (Encryption and Export Administration Regulations, EAR) ja huolehtia määräysten oikeasta tulkinnasta. Jos sovelluksen salausmenetelmät eivät kuulu poikkeustapauksiin, julkaisijan on toimitettava Applelle dokumentti vientivaatimuksien noudattamisesta (export compliance documentation). Poikkeustapauksissa taas julkaisija voi joutua lähettämään vuosittaisen itseluokitteluraportin (annual self-classification report) Yhdysvaltain hallitukselle. (Apple 2021f.)

6 POHDINTA

Opinnäytetyön ensisijaisena tavoitteena oli julkaista työn aiheena oleva sovellus, ja se saatiin toteutettua. Sovellus on nyt ladattavissa sekä Google Play - että App Store -sovelluskaupoista. Tärkein opinnäytetyön aikana opittu asia oli, että julkaisuun liittyvien asioiden selvittäminen kannattaa aloittaa hyvissä ajoin. Julkaisua valmistellessa kävi ilmi useampikin asia, mikä ei kehityksen aikana tuntunut tärkeältä, mutta julkaisuvaiheessa osoittautuikin välttämättömäksi. Julkaisun työnkulku ja eri julkaisualustojen vaatimukset on hyvä olla tiedossa jo sovellusta kehittäessä, jotta julkaisuprosessia aloittaessa ei kulu turhaan aikaa muokatessa projektia julkaisuvaatimusten mukaiseksi. Sovelluksen julkaisua ja näin ollen myös opinnäytetyötä hidasti se, että Xamarin.Forms-dokumentaatio oli paikoin puutteellista. Usein dokumentaatiota seuratessa piti etsiä aiheesta lisätietoa keskustelupalstoilta tai esimerkiksi YouTube-palvelusta.

App Store -julkaisuun tarvittavia sertifikaattia, sovellustunnusta ja jakeluprofiilia luodessa kävi ilmi, että Microsoftin Xamarin.iOS-julkaisua koskeva dokumentaatio oli hieman vanhentunut. Applen oma dokumentaatio taas oli epäselvä eikä tarkasti listannut todellisia vaiheita, mitä App Store -julkaisuun sisältyy. Julkaisussa tärkeimmäksi apuvälineeksi muodostuikin YouTube-palvelusta löytynyt Xamarin Guy -tekijän “Publish iOS App to App Store Xamarin forms”-video, joka kuvaa Xamarin.Forms-projektin App Store -julkaisun vaiheet selkeästi ja järjestyksessä (Xamarin Guy 2020). Videon perusteella sertifikaatin, sovellustunnuksen ja jakeluprofiilin luominen sekä itse sovelluksen julkaisu onnistui ongelmitta ja nopeasti. Kun App Store -kaupan määrittelyt oli tehty ja sovellus lähetettiin arvioitavaksi, sovellus tarkistettiin samana päivänä. Sovellus näkyi sovelluskaupassa tarkastusta seuraavana päivänä.

Google Play -julkaisun dokumentaatio oli samaan tapaan vanhentunut. Xamarin.Android-julkaisuun liittyvästä dokumentaatiosta oli kuitenkin mahdollista seurata julkaisuprosessia hyvin pitkälle ja vain muutamat asiat olit tarkastettava Googlen tarjoamasta Google Play -kauppaa koskevasta dokumentaatiosta. Koko julkaisuprosessia ei toteutettu Googlen dokumentaation perusteella, sillä sen todettiin olevan vaikealukuisempi kuin Microsoftin dokumentaation. Määritysten tekemiseen Google Play -kaupassa meni noin yksi työpäivä. Kun sovellus lähetettiin arvioitavaksi, arvioinnissa kesti kolme päivää. Arvioinnin jälkeen sovellus näkyi sovelluskaupassa välittömästi. Julkaisun jälkeen sovelluksessa havaittiin pieni ongelma, jolloin julkaisuportaaliin ladattiin uusi versio sovelluksesta. Tämän version tarkastuksessa kesti vain yksi päivä.

Syy siihen, miksi osa Microsoftin Xamarin.Formsia koskevista dokumenteista oli vanhentuneita saattaa olla se, että seuraavien vuosien aikana Microsoft on siirtymässä Xamarin.Formsista uuteen sovelluskehykseen nimeltä .NET Multi-platform UI, eli lyhyesti .NET Maui tai Maui. Microsoftin dokumentaation mukaan Xamarin.Forms-sovelluksen päivittäminen Maui-sovellukseksi tulee onnistumaan .NET-päivitysassistenttia käyttämällä ilman tarvetta suurille koodimuutoksille. (Microsoft 2021t.) Tämän opinnäytetyön aiheena oleva sovellus tullaan todennäköisesti päivittämään Maui-sovellukseksi, kun Maui julkaistaan.

Google vaatii Play App Signing -allekirjoitusavaimen käyttöä uusille sovelluksille. Microsoftin dokumentaatiossa on ohjeet allekirjoitusavaimen hankkimiselle, mutta dokumentaatiossa ei kuitenkaan mainita, että kun Googlen allekirjoitusavain otetaan käyttöön, sovelluksen allekirjoitusavaimen SHA-1-sormenjälki muuttuu. Tämä on kuitenkin tärkeä tieto kehittäjälle, sillä kyseisen sormenjäljen muuttuessa Google Maps -palvelu ei enää toimi sovelluksessa.

Tietotekniikan alalla harvoilla termeillä on vakiintunut suomenkielinen käännös. Google Play -julkaisuprosessia toteutettiin aluksi Google Play Console -portaalin suomenkielisellä versiolla, mutta hyvin pian kieli täytyi vaihtaa englanniksi, sillä suomennokset sivulla olivat kankeita ja hankalalukuisia. Asia hankaloitti myös opinnäytetyön kirjoittamista, ja useissa kohdissa päädyttiinkin kirjoittamaan termistä sekä alkuperäinen englanninkielinen sana että sen suomennos.

Tämän opinnäytetyön aikana oli tarkoitus toteuttaa sovellukseen kolme uutta toiminnallisuutta, mutta yksi niistä jäi toteuttamatta. Tämä toteuttamatta jäänyt toiminnallisuus on laitteen QR-koodin lukeminen sovelluksella. QR-koodin luku on kuitenkin sovelluksen käyttäjien kannalta tärkeä ominaisuus, joten se tullaan toteuttamaan lähitulevaisuudessa.

LÄHTEET

Adasiewicz, Rafał 2018. 3 Reasons Why You Should Always Remove Logs From Production. Netguru. Hakupäivä 27.11.2021. https://www.netguru.com/blog/3-reasons-why-you-should-always-remove-logs-from-production-mobile-app.

Amit 2020. Huawei AppGallery vs Google Play Store. Huawei Update. Hakupäivä 27.11.2021. https://www.huaweiupdate.com/huawei-appgallery-vs-google-play-store/.

Android 2021a. Publish your app. Hakupäivä 27.11.2021. https://developer.android.com/studio/publish.

Android 2021b. Android Open Source Project. Hakupäivä 27.11.2021. https://source.android.com/.

Android 2021c. The best of Google, right on your devices. Hakupäivä 27.11.2021. https://www.android.com/gms/.

Android 2021d. About Android App Bundles Hakupäivä 27.11.2021. https://developer.android.com/guide/app-bundle.

Android 2021e. Request location permissions. Hakupäivä 28.11.2021. https://developer.android.com/training/location/permissions.

Android 2021f. Google Play. Hakupäivä 28.11.2021. https://developer.android.com/distribute.

Android 2021g. Google Play icon design specifications. Hakupäivä 28.11.2021. https://developer.android.com/distribute/google-play/resources/icon-design-specifications.

Apple 2018. Introducing App Store Connect. Hakupäivä 28.11.2021. https://developer.apple.com/news/?id=06042018.

Apple 2020a. What is app signing?. Hakupäivä 28.11.2021. https://help.apple.com/xcode/mac/current/#/dev3a05256b8.

Apple 2020b. Certificate type. Hakupäivä 28.11.2021. https://help.apple.com/xcode/mac/current/#/dev80c6204ec.

Apple 2021a. Choosing a Membership. Hakupäivä 28.11.2021. https://developer.apple.com/support/compare-memberships/.

Apple 2021b. Membership Details. Hakupäivä 28.11.2021. https://developer.apple.com/programs/whats-included/.

Apple 2021c. App Store Review Guidelines. Hakupäivä 28.11.2021. https://developer.apple.com/app-store/review/guidelines/.

Apple 2021d. App Icon. Hakupäivä 28.11.2021. https://developer.apple.com/design/human-interface-guidelines/ios/icons-and-images/app-icon/.

Apple 2021e. Launch Screen. Hakupäivä 28.11.2021. https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/launch-screen/.

Apple 2021f. Export compliance overview. Hakupäivä 28.11.2021. https://help.apple.com/app-store-connect/#/dev88f5c7bf9.

Apptamin 2021. Google Play Feature Graphic Examples and Best Practices. Hakupäivä 27.11.2021. https://www.apptamin.com/blog/feature-graphic-play-store/.

Google 2021a. Requesting access to location in the background. Hakupäivä 28.11.2021. https://support.google.com/googleplay/android-developer/answer/9799150.

Google 2021b. Create and set up your app. Hakupäivä 28.11.2021. https://support.google.com/googleplay/android-developer/answer/9859152.

Google 2021c. Set up an open, closed, or internal test. Hakupäivä 28.11.2021. https://support.google.com/googleplay/android-developer/answer/9845334?hl=en.

Grabham, Dan 2021. What does Huawei's trade ban mean for your present or future Huawei phone?. Pocket-lint. Hakupäivä 27.11.2021. https://www.pocket-lint.com/phones/news/huawei/148102-what-does-huawei-s-google-ban-mean-for-your-huawei-or-honor-phone.

Hildenbrand 2019. What are Google Mobile Services (GMS) and why does my phone need them?. Android Central. Hakupäivä 27.11.2021 https://www.androidcentral.com/what-gms-and-why-does-my-phone-need-it.

Microsoft 2020a. What is Xamarin.Forms?. Hakupäivä 28.11.2021. https://docs.microsoft.com/fi-fi/xamarin/get-started/what-is-xamarin-forms.

Microsoft 2020b. Improve Xamarin.Forms App Performance. Hakupäivä 27.11.2021. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/deploy-test/performance.

Microsoft 2020c. Xamarin.Forms Data Binding. Hakupäivä 28.11.2021. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/data-binding/.

Microsoft 2021a. Create a Xamarin.Forms application quickstart. Hakupäivä 28.11.2021. https://docs.microsoft.com/en-us/xamarin/get-started/quickstarts/app.

Microsoft 2021b. Welcome to the Visual Studio IDE. Hakupäivä 28.11.202). https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019.

Microsoft 2021c. Xamarin.Forms supported platforms. Hakupäivä 28.11.2021. https://docs.microsoft.com/en-us/xamarin/get-started/supported-platforms.

Microsoft 2021d. The Model-View-ViewModel Pattern. Hakupäivä 27.11.2021. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm.

Microsoft 2021e. XAML Compilation in Xamarin.Forms. Hakupäivä 27.11.2021. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/xaml/xamlc.

Microsoft 2021f. Xamarin.Forms Compiled Bindings. Hakupäivä 28.11.2021. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/data-binding/compiled-bindings.

Microsoft 2021g. Introduction to Custom Renderers. Hakupäivä 27.11.2021. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/custom-renderer/introduction.

Microsoft 2021h. Exceptions and Exception Handling. Hakupäivä 28.11.2021. https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/exceptions/.

Microsoft 2021i. Try-catch (C# Reference). Hakupäivä 27.11.2021. https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/try-catch.

Microsoft 2021j. Preparing an Application for Release. Hakupäivä 27.11.2021. https://docs.microsoft.com/en-us/xamarin/android/deploy-test/release-prep.

Microsoft 2021k. Publishing to Google Play. Hakupäivä 27.11.2021. https://docs.microsoft.com/en-us/xamarin/android/deploy-test/publishing/publishing-to-google-play.

Microsoft 2021l. Obtaining a Google Maps API Key. Hakupäivä 28.11.2021. https://docs.microsoft.com/en-us/xamarin/android/platform/maps-and-location/maps/obtaining-a-google-maps-api-key.

Microsoft 2021m. Xamarin.iOS app distribution overview. Hakupäivä 28.11.2021. https://docs.microsoft.com/en-us/xamarin/ios/deploy-test/app-distribution/.

Microsoft 2021n. Device provisioning for Xamarin.iOS. Hakupäivä 28.11.2021. https://docs.microsoft.com/en-us/xamarin/ios/get-started/installation/device-provisioning/.

Microsoft 2021o. App Store Distribution. Hakupäivä 28.11.2021. https://docs.microsoft.com/en-us/xamarin/ios/deploy-test/app-distribution/app-store-distribution/.

Microsoft 2021p. Publishing Xamarin.iOS apps to the App Store. Hakupäivä 28.11.2021. https://docs.microsoft.com/en-us/xamarin/ios/deploy-test/app-distribution/app-store-distribution/publishing-to-the-app-store.

Microsoft 2021q. Application Icons in Xamarin.iOS. Hakupäivä 28.11.2021. https://docs.microsoft.com/en-us/xamarin/ios/app-fundamentals/images-icons/app-icons.

Microsoft 2021r. Designing user interfaces with Xcode. Hakupäivä 28.11.2021. https://docs.microsoft.com/en-us/xamarin/ios/user-interface/storyboards/.

Microsoft 2021s. Xamarin.Forms WebView. Hakupäivä 28.11.2021. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/webview.

Microsoft 2021t. What is .NET MAUI?. Hakupäivä 28.11.2021. https://docs.microsoft.com/en-us/dotnet/maui/what-is-maui.

Nield, David 2019. Why Some Android Phones Don't Have the Play Store. Gizmodo. Hakupäivä 27.11.2021. https://gizmodo.com/why-some-android-phones-dont-have-the-play-store-1835087772.

PreEmptive Solutions 2021. Root Check. Hakupäivä 27.11.2021. https://www.preemptive.com/dotfuscator/pro/userguide/en/protection_checks_root.html.

Statista 2021. Number of apps available in leading app stores as of 1st quarter 2021. Hakupäivä 27.11.2021. https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/. Vaatii käyttöoikeuden.

Xamarin Guy 2020. Publish iOS App to App Store Xamarin forms. Hakupäivä 28.11.2021. https://www.youtube.com/watch?v=fkBRXzotbzw&t=229s&ab_channel=XamarinGuy.