Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Frans Vikstén

Maksupalvelut Unity-mobiilisovelluksessa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

25.11.2019

Tekijä Otsikko Sivumäärä Aika	Frans Vikstén Maksupalvelut Unity-mobiilisovelluksessa 35 sivua 25.11.2019
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Pelisovellukset
Ohjaaja	Lehtori Miikka Mäki-Uuro

Insinöörityön tarkoituksena oli tutkia erilaisia maksujärjestelmiä ja maksunvälityspalveluita ja toteuttaa asiakasyrityksen mobiilisovellukseen vaatimusten mukaisesti sovelluksen sisäinen maksujärjestelmä, josta voidaan ostaa lisäsisältöä sovelluksen sisäisesti kuukausimaksutilauksena. Sovelluksen sisäisen maksujärjestelmän tuli noudattaa sääntöjä, jotka on asetettu sen julkaisualustassa, ja olla turvallinen. Sovelluksen sisäisellä maksujärjestelmällä tuli myös olla mahdollisuus seurata tietoja ostoksista ja ostajista. Lopputuloksena tavoitteena oli, että tuotteita pystytään myymään Android- ja iOS-laitteilla, joissa asiakasyrityksen sovellus on julkaistu Google Play- ja Apple App Store -palveluissa.

Insinöörityössä perehdyttiin erilaisiin mahdollisiin maksujärjestelmä- ja maksunvälityspalveluihin, joita voidaan käyttää sovelluksessa, sekä niiden sääntöihin ja hinnoitteluun. Tämän jälkeen voitiin valita mahdolliset maksupalvelut, joita sovelluksen sisäisen maksujärjestelmän toteuttamiseen haluttiin käyttää. Työssä perehdyttiin myös maksujärjestelmän toteuttamiseen ja hallintaan ohjelmakoodissa. Lisäksi tutustuttiin erilaisiin tuotteisiin, tuotteiden myynnin turvallisuuteen, kuittien tarkastamiseen, maksujärjestelmän testaamiseen sekä yksinkertaisen käyttöliittymän luomiseen, jolla voidaan suorittaa myyntikutsuja sovelluksessa.

Työssä toteutettiin valittujen maksupalveluiden käyttöönotto ja tuotteiden lisääminen niiden tarjoamissa hallintanäkymissä. Sovelluksen sisäinen maksujärjestelmä toteutettiin Unity-pelimoottorissa käyttäen Unity IAP -liitännäistä. Maksujärjestelmän tarvittavat ominaisuudet toteutettiin ohjelmakoodissa.

Insinöörityön tavoitteessa onnistuttiin; saatiin toteutettua maksujärjestelmä Unity-pelimoottorissa rakennettuun Android- ja iOS-sovellukseen. Maksujärjestelmään pystyy lisäämään tuotteita, ja niiden myyminen onnistuu käyttäen valittuja maksupalveluita. Työn aikana opittiin erilaisista mahdollisista maksupalveluista, niiden käyttöönotosta ja käytöstä sekä niiden säännöistä. Maksujärjestelmä on tällä hetkellä beetatestausvaiheessa.

Avainsanat	Unity, maksupalvelu, maksujärjestelmä

Author Title Number of Pages Date	Frans Vikstén Mobile payments in Unity-application 35 pages 25 November 2019
Degree	Bachelor of Engineering
Degree Programme	Information and Communications Technology
Professional Major	Game Applications
Instructor	Miikka Mäki-Uuro, Senior Lecturer

The main goal of the thesis was to research several different existing payment services and to implement a store to a mobile application according to client company's specifications. The store should be able to sell additional content for an application with a monthly subscription, follow all the rules set by the publishing platforms and be secure. The in-app store should also be able to track customer purchase analytics. The end goal was that the in-application store can be used to sell products on Android and iOS devices, where the client company's application is published in Google Play and Apple App Store services.

The thesis explains some possibilities between different payment platforms and payment gateways, which can be used in Unity mobile application as well as their rules, restrictions and pricing models. Additionally, initialization of the payment systems and adding products in their management views is explained and the payment service is created in Unity-game engine using Unity IAP plugin. The thesis also goes into the implementation in program code, different product types that can be sold, validating receipts of the purchases, testing the payment system in application and creating a simple user interface from which the client can complete purchases in the application.

The goals of the thesis were reached, and as an end product was produced a working in-application store using a mobile application developed with Unity game engine. It's possible to add and sell products using this in-application store. The information about different possible payment platforms, about their usage and implementation and rules and regulations was useful in selecting and building the in-app payment system for the application.

Keywords	Unity, payment service, in-app purchase

Sisällys

Lyhenteet

1 Johdanto 1

2 Maksupalvelut 2

2.1 Säännöt 3
2.2 Hinnoittelu 4
2.3 Vaihtoehtoiset maksupalvelut 4
2.4 Maksupalvelujen lisäominaisuudet 5

3 Maksupalvelun käyttöönotto 8

3.1 Alustus 8
3.2 Tuotteet 10

4 Sovelluksen sisäisen maksujärjestelmän luonti 15

4.1 Työkalut 15
4.2 Unity IAP:n käyttöönotto 16
4.3 Tuotteet Unity IAP:ssa 18
4.4 Tilaukset 22
4.5 Kuitit ja turvallisuus 23
4.6 Ostostapahtuma 26
4.7 Palauttaminen 28
4.8 Testaus 29
4.9 Lopputoimenpiteet 32

5 Työn tulokset 34

5.1 Maksujärjestelmän käyttö 34
5.2 Kehityskohteet 34

6 Yhteenveto 35

Lähteet 36

Lyhenteet

API	Application programming interface, ohjelmointirajapinta
APK	Android Package, Android-sovellusten käyttämä tiedostomuoto
IAP	In-App Purchase, sovelluksen sisäinen ostos
ID	Identifier, tietojenkäsittelyssä yksilöllinen tunniste
iOS	iPhone OS, Applen kehittämä käyttöjärjestelmä
JSON	JavaScript Object Notation, avoimen standardin tiedostomuoto tiedonvälitykseen

1 Johdanto

Insinöörityön päätarkoituksena oli tutustua maksupalveluihin, joita voidaan käyttää Unity-pelimoottorilla toteutetussa sovelluksessa sovelluksen sisäisenä maksujärjestelmänä asiakasyritykselle, ja vertailla niitä. Maksujärjestelmän toteutuksessa haluttiin ottaa huomioon, että sitä toteutettaessa noudatetaan Googlen ja Applen asettamia sääntöjä ja että maksujärjestelmä on turvallinen.

Sovellus julkaistaan Apple App Storessa [1] ja Google Play Storessa [2]. Myynnissä oleva tuote on tilauspalvelu, jolla voi avata sovellukseen uusia ominaisuuksia esimerkiksi kuukausimaksulla. Sovelluksen kanssa käytetään asiakasyrityksen laitetta, joka ostetaan erikseen.

Sovelluksen sisäiset maksut ovat tärkeä keino rahoittaa mobiilisovelluksia ja pelejä. Aihe valittiin, koska haluttiin parempi ymmärrys sovellusten sisäisten maksujen toiminnasta sekä oppia, miten sovelluksen sisäinen maksujärjestelmä toteutetaan Unity-pelimoottorilla toteutettuun mobiilisovellukseen. Aihe valittiin myös, koska asiakasyrityksen tuotteen spesifikaatiota varten tarvittiin sovelluksen sisäinen maksujärjestelmä, jossa tuotetta tai mahdollisia tulevia tuotteita voidaan myydä.

Insinöörityön toisessa luvussa tutkitaan mahdollisia ratkaisuja sovelluksen sisäisen maksujärjestelmän toteutukseen Unity-sovelluksessa. Luvussa vertaillaan erilaisia olemassa olevia maksupalveluita, ottaen huomioon myyty tuote, myyntialustat, maksupalvelujen hinnastot, säännöt liittyen julkaisualustaan ja erilaiset mahdolliset maksupalvelun tarjoamat lisäominaisuudet.

Luvussa 3 kerrotaan verkkokaupan toteutuksesta, valittujen palveluiden käyttöönotosta ja alustamisesta. Luvussa 4 kerrotaan palveluiden käyttöönotosta Unity-pelimoottorin sisällä ja sen luomisesta ohjelmakoodiin. Luvussa kerrotaan myös, minkälaisia erilaisia tuotteita palveluihin voi asettaa myytäväksi ja miten palvelujen tarjoamat lisäomainaisuudet, kuten analytiikka ja turvallisuus, toimivat. Lopuksi luvussa kerrotaan käyttöliittymän tekemisestä Unity-pelimoottorilla ja siitä, mitä käyttöliittymän tekemisessä tulee ottaa

huomioon. Luvussa 5 esitellään toteutettu maksupalvelu, sen nykyiset ominaisuudet ja mahdolliset kehityskohteet. Viimeisessä luvussa on insinöörityön yhteenveto.

2 Maksupalvelut

Tässä luvussa tutustutaan erilaisiin maksupalveluihin, joita voidaan käyttää mobiilisovelluksessa. Luvussa käydään läpi maksupalveluiden sääntöjä, milloin niitä tulisi käyttää ja miten ne eroavat toisistaan. Luvussa kerrotaan myös maksupalvelujen hinnoista sekä erilaisista vaihtoehtoisista maksupalveluista. Lisäksi luvussa kerrotaan maksupalvelujen tarjoamista lisäominaisuuksista.

Maksupalvelulla tarkoitetaan palvelua, jota voidaan käyttää maksuissa, jotka suoritetaan mobiililaitteen kautta. Mobiilimaksuja voi käyttää laajasti, kuten digitaalisen sisällön tai fyysisten tuotteiden ja palveluiden ostamiseen.

Muutamia erilaisia monetisaatiomalleja, joita sovelluksissa käytetään, ovat maksulliset sovellukset. Maksullisista sovelluksista maksetaan kertamaksu, kun sovelluksen ostaa. Sovellus voi olla myös ilmainen ladata ja käyttää, mutta se noudattaa monetisaatiomalleja, kuten sovellukseen lisätyt mainokset. Käyttäjän katsomat mainokset rahoittavat sovelluksen toiminnan. Ilmainen sovellus voi myös sisältää sovelluksen sisäisiä ostoksia. Sovelluksen sisäiset ostokset voivat olla yksittäisiä ostoksia, joita tekemällä käyttäjä voi avata sisältöä sovellukseen. Sovellus voi sisältää myös tilaustyyppisiä ostoksia, jotka vaativat käyttäjältä esimerkiksi kuukausimaksua, jotta tilauksen sisältö on saatavilla käyttöön sovelluksessa.

Google Play- ja Apple App Store -kaupat tarjoavat omat maksupalveluja, jotka on mahdollista ottaa käyttöön Unity-sovelluksessa käyttäen esimerkiksi Unity IAP -liitännäistä. Myös muita kolmannen osapuolen maksupalveluita on mahdollista käyttää Unity IAP:n kautta, kuten esimerkiksi PayPal [3], Shopify [4] tai Stripe [5]. Näitä palveluita ei kuitenkaan suoraan tueta Unity IAP:n kautta, ja nämä palvelut ovat laajalti tarkoitettu fyysisten tuotteiden myymiseen sovelluksen tai verkkokaupan sisällä. Kolmannen osapuolen maksupalvelut eivät usein ole käytettävissä suoraan sovelluksen sisäisissä maksuissa, mutta

on kuitenkin olemassa poikkeuksia, jolloin kolmannen osapuolen maksupalvelua voidaan käyttää sovelluksen sisäisissä ostoksissa.

Maksupalvelut tarjoavat myös lisäominaisuuksia kaupankäynnin seuraamiseen sovelluksen sisällä. Niistä löytyy kattava analytiikka myynnistä, asiakkaista ja sovelluksen käytöstä. Nämä kauppapaikat myös tallentavat kuitit ostoksista palvelimelleen, jota voidaan käyttää apuna ostosten varmentamiseen. Kuittien varmennus on tärkeää ottaa huomioon tuotteita myydessä.

2.1 Säännöt

Erilaisilla maksupalveluilla on olemassa erilaisia sääntöjä ja rajoitteita niissä myytävien sovellusten ja sovellusten sisällä myytävän sisällön suhteen. Google Play Store -kaupassa myytävien sovellusten tulee käyttää aina maksuissaan Google Playn maksupalvelua. Pelisovelluksen sisäisissä ostoksissa, joissa tarjotaan peliin sisältöä, tulee käyttää Google Play In-app Billing -maksua. Jos kyseessä on sovellus, joka ei ole peli, täytyy sen myös käyttää Google Play In-app Billing -maksua. Jos maksu on pelkästään fyysisestä tuotteesta tai palvelusta, voidaan käyttää kolmannen osapuolen maksupalvelua. Mikäli sovelluksessa myydään digitaalista sisältöä, jota voidaan käyttää sovelluksen ulkopuolella, esimerkiksi musiikki, jota voi soittaa muilla laitteilla, ei tarvitse käyttää Google Play In-app Billing -maksua. [6.]

Apple App Store -kaupassa täytyy käyttää Applen in-app purchase -maksua, jos aikoo avata sisältöä sovelluksen sisällä maksua vastaan, esimerkiksi tilaukset, pelin sisäinen valuutta tai lisäsisällön avaaminen. Sovellukset eivät myöskään saa käyttää omia keinojaan avatakseen sisältöä, kuten lisenssiavaimia tai QR-koodeja. [7.]

Applen sääntöjen mukaan sovellukseen ei saa myöskään luoda näkymää, josta pääsee käsiksi kolmannen osapuolen sovellukseen tai lisäosaan. On siis Applen sääntöjen vastaista luoda IAP-maksun kiertoa käyttäen kolmannen osapuolen verkkokauppaa. [7.]

Applen säännöissä mainitaan myös, että sovellusten, jotka vaativat ulkoisen laitteen toimiakseen tai toimivat yhdessä ulkoisen laitteen kanssa, ei tarvitse käyttää in-app purchase-maksua, mutta in-app purchase-maksun täytyy kuitenkin olla mahdollinen sovelluksessa. [7.]

Esimerkiksi jos rakentaa sovellusta, joka on kuntoilu- ja kalorilaskuri, josta on yksinkertaisempi ilmainen versio, mutta myös paljon monipuolisempi premium-versio, ja sovellusta aiotaan myydä tilauspalveluna asiakkaille, jotta he voivat käyttää premium-versiota, on kyseessä virtuaalisten tuotteiden myynti. Tässä tilanteessa tulisi käyttää IAP-maksua. [7.]

2.2 Hinnoittelu

Eri maksupalveluilla on erilaisia hinnoittelumalleja. Google Play In-app Billing -maksut ja Applen in-app purchase -maksut sisältävät 30 prosentin palvelumaksun ostoksen hinnasta eli myydystä tuotteesta tulee 70 prosenttia myyjälle itselleen. Tilaustuotteiden palvelumaksut laskevat 15 prosenttia, jos asiakas on tilannut onnistuneesti tilaustuotetta 12 maksullista kuukautta. Google Pay -palvelussa palvelumaksu on 2,9 prosenttia tuotteen hinnasta ja 0,30 dollaria, Apple Pay -palvelussa palvelumaksu on 3 prosenttia. Google Pay ja Apple Pay ovat palveluita, jotka on tarkoitettu pääasiassa fyysisten tuotteiden ja palveluiden myymiseen. [8; 9.]

2.3 Vaihtoehtoiset maksupalvelut

Vaihtoehtoisia maksupalveluita Googlen tai Applen tarjoamille palveluille löytyy useita; ne ovat useimmin käytössä verkkosivustoilla. Vaihtoehtoisia maksupalveluita ei käytetä niin usein Google Play- tai Apple App Store -kaupoissa niiden asettamien sääntöjen takia. Fyysisten tuotteiden ja palveluiden sekä esimerkiksi laitetta käyttävien sovellusten kanssa säännöt kuitenkin sallivat niiden käytön. Vaihtoehtoisina maksupalveluina sovellukselle tutkittiin muutamia eri vaihtoehtoja, joita voitaisiin käyttää mahdollisesti Google Play- tai Apple AppStore -maksujen sijasta.

Yhtenä vaihtoehtoisena maksupalveluna oli Shopify-niminen verkkokauppa- ja maksupalvelu. Se mahdollistaa oman verkkokaupan luomisen verkkosivustolle. Shopifyn kautta voi käyttää useita eri maksunvälittäjiä, kuten PayPal tai Stripe. Se toimii samankaltaisesti kuin Google Play tai Apple App Store, mutta sen kautta voidaan luoda oma kauppapaikka verkkosivuna, josta tuotteita voidaan ostaa, ja sen Unity-sovellukseen tarjoamat lisäominaisuudet ovat vähäisemmät. Kauppapaikka voidaan myös lisätä sovellukseen ohjelmakoodin kautta, ja ostoksia voidaan tehdä sovelluksen kautta. Shopify sisältää lisäominaisuuksia, joilla voidaan seurata ostosten myyntiä. [10.]

Shopify voidaan lisätä Unity-sovellukseen käyttämällä Shopify Buy SDK -kirjastoa, jonka voi ladata Shopifyn sivustoilta. Shopify Buy SDK sisältää myös dokumentaation ja ohjeet sen lisäämisestä Unity-sovellukseen.

Shopify on kuitenkin enemmän tarkoitettu verkkosivuilla toimiviin sovelluksiin ja fyysisten tuotteiden myymiseen mobiilisovelluksen sisällä. Shopify Buy SDK:n kautta voidaan hakea myytäviä tuotteita ja kokoelmia tuotteista kaupasta, ja se tarjoaa myös mahdollisuuden suorittaa ostoksia käyttäen suoraa linkkiä tai WebView-komponenttia, jolla näytetään osa verkkosivusta sovelluksen sisällä. Shopifyn maksun voi myös suorittaa käyttäen Apple Pay -palvelua iOS-sovelluksessa. [10.]

Shopify sisältää myös oman maksunvälityspalvelunsa nimeltä Shopify Payments. Tämä palvelu ei kuitenkaan ole käytössä Suomessa, mutta sen tilalla voidaan käyttää muita Suomessa toimivia maksunvälittäjiä, kuten Stripe tai PayPal. [11.]

2.4 Maksupalvelujen lisäominaisuudet

Eri maksupalvelut tarjoavat erilaisia lisäominaisuuksia, joilla voidaan auttaa myynnin helpottamista, turvallisuutta ja onnistumista. Erilaisia palveluita, joita maksupalvelulla on, ovat esimerkiksi valmiit laskutuskumppanit, jotta rahan siirto ei jää sovelluskehittäjän tehtäväksi ja, maksutietojen turvallinen tallentaminen palvelimille, josta niitä voidaan käyttää kaupankäynnissä syntyvissä tilanteissa, kuten palautuksissa tai maksujen hallinnassa.

Maksupalvelut voivat myös sisältää lisäominaisuuksia muunlaisen asiakasdatan tallentamiseen palveluun, kuten miten hyvin tuotteet tai tilaukset ovat myyneet, tietoa asiakkaista ja asiakkaiden käyttötavoista.

Google Play

Google Playn tarjoamat lisäominaisuudet löytyvät Google Play Consolen kautta. Google Play tarjoaa Google Play Billing -kirjaston palvelun käyttöä varten. Google Play Billing sisältää palvelimen puoleisen laskutuksen ominaisuuksia, joilla voidaan validoida ja hallita tilauksia. Tilausten tilaa pystyy seuraamaan, suorittamaan palautuksia ja saamaan reaaliaikaisia päivityksiä tilauksista. Google Play tarjoaa myös testaustyökalut, joilla pystytään testaamaan mobiilisovelluksen sisäisen kaupan luomista ja toimintoja, jotka eivät johda ylimääräisiin maksutapahtumiin. [12.]

Google Play tukee yli 135 maata, joissa voidaan myydä ja ostaa sovelluksen sisäisiä tuotteita. Myytäessä tuotteita Google Play -palvelun kautta ei tarvitse kerätä tietoja maksutapahtumista erikseen, sillä Google Play tallentaa tiedot automaattisesti ja tiedot voidaan hakea palvelusta myöhempää käyttöä varten. Google Play -palvelua käyttäessä ei myöskään tarvitse tutkia jokaisen maan sääntöjä tuotteiden myyntiin erikseen. Google Play -palvelun kautta voi asettaa asetuksia eri maissa myynnille, ja myynti tapahtuu palvelun kautta. [12.]

Asiakkaat voivat maksaa luotto- tai pankkikorteilla, yli 150 eri mahdollisen laskutuskumppanin kautta yli 50 maassa. Lahjakortteja voidaan käyttää yli 30 maassa ja PayPal-maksua yli 20 maassa. Google Play -maksut suoritetaan Google Play Billing -maksupalvelun kautta, jonka se tarjoaa käyttöön. Järjestelmä on hyvin kehittynyt ja luotettu, ja sitä käytetään erittäin laajalti mobiilimaksujen suorittamiseen. [12.]

Apple App Store

Apple App Storen tarjoamat lisäominaisuudet löytyvät Apple Developer Consolesta. Apple App Store Connect tarjoaa Apple App Store Connect API:n käyttöön, ja sen kautta löytyy lisäominaisuuksia, esimerkiksi työkaluja, joilla voidaan seurata myyntitapahtumia ja validoida ja suorittaa mahdollisia palautuksia. Apple App Store Connectin kautta voidaan myös hakea tietoa ostoksista ja tilauksista. [13.]

Sovelluksissa, jotka sisältävät sovelluksen sisäisiä ostoksia, voidaan seurata myynnin trendejä, jotka kertovat, kuinka monta aktiivista tilausta sovelluksella on, kuinka monta uutta tilaajaa saadaan joka päivä ja kuinka paljon sovellus tuottaa niistä. Tilauksista voidaan myös seurata sitä, miten asiakkaat ovat siirtyneet esimerkiksi alennetun tarjoushinnan jälkeen käyttäjiksi ja kuinka usein tilauksia on uusittu tai peruttu. Tilausten perumisesta kerätään myös tietoa käyttäjiltä: miksi tilaukset on mahdollisesti peruttu. [13.]

Apple App Store tarjoaa TestFlight-nimisen ominaisuuden, jolla sovelluksen beetatestausta voidaan helpottaa. Sovelluksen In-app purchase -maksuja voidaan myös testata palvelun tarjoamilla työkaluilla. [13.]

3 Maksupalvelun käyttöönotto

Insinöörityössä valittiin käyttöön Google Play- ja Apple App Store -maksupalvelut, koska ne soveltuivat parhaiten asiakasyrityksen tuotteen myymiseen. Maksujen suorittamiseen käytetään Google Play Billing -maksua ja Apple In-App Purchase -maksua. Nämä palvelut täytyy ottaa käyttöön, ennen kuin niitä voidaan käyttää sovelluksessa. Maksupalvelujen käyttöönotto tehdään niiden tarjoamissa hallintanäkymissä, jotka löytyvät palveluiden verkkosivuilta.

3.1 Alustus

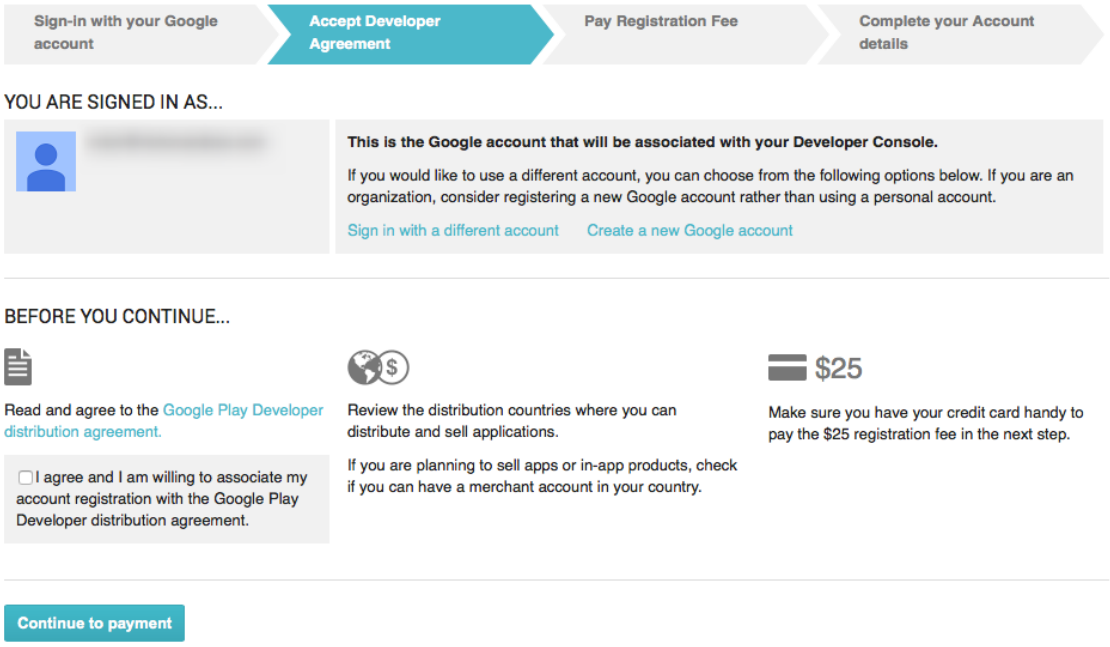
Maksupalveluiden Google Play ja Apple App Store käyttöönotossa täytyy ensiksi tehdä alustuksia, jotta on mahdollista lisätä tuotteita tai tilauksia sovellukseen myytäväksi. Tässä luvussa kerrotaan palveluiden käyttöönotosta ja tuotteiden lisäämisestä niiden kautta, jotta palvelut ovat myöhemmin valmiita käyttää sovelluksessa.

Google Playn alustus

Jotta tuotteita voi myydä Google Play-kaupassa, se täytyy ottaa käyttöön Google Play Developer Consolessa. Ensimmäiseksi tulee luoda kauppiastili, jotta tuotteita voidaan lisätä ja myydä palveluun ladatuissa sovelluksissa. Kauppiastiliä luodessa pitää ilmoittaa myyjän yrityksen laillinen nimi, yhteyshenkilön nimi, yrityksen sijainti ja osoite, puhelin numero sekä muita tietoja miten yritykseen voidaan ottaa yhteyttä ja yrityksen mahdolliset kotisivut.

Lisäksi yrityksen tuotteiden myyntiä varten tulee lisätä yrityksen pankkitili. Tietoja, joita sitä varten tarvitaan, ovat pankkitilin International Bank Account- eli IBAN-numero sekä Swift-BIC-tunnus. [14.]

Kuvassa 1 luodaan uusi Google-kehittäjätili. Näkymässä voi lisätä Merchant-tilin tiedot ja asettaa kehittäjätilin toimintaan. Jos luodaan kokonaan uutta sovellusta Google Play-palveluun, täytyy maksaa 25 dollarin hintainen maksu, jotta kehittäjätili on aktiivinen ja sitä voidaan käyttää. [14.]

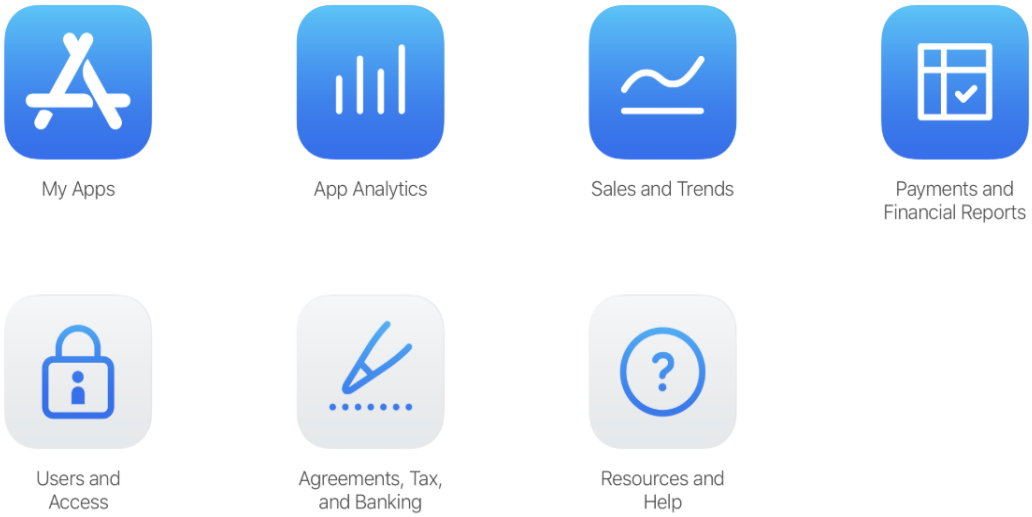


Kuva 1. Sopimusten hyväksyminen ja kehittäjätietojen lisääminen Google Play Storen alustuksessa [14].

Tämän jälkeen Google Play Developer Consolen päänäkymästä voi lisätä tai luoda sinne uusia sovelluksia. Google Play vaatii, että sille lähetetään tässä vaiheessa sovelluksesta APK-tiedosto, jonka manifest-tiedostossa on BILLING-lupa. Sen tulee olla lähetettynä Google Play Developer Consoleen, ennen kuin sovellukseen voi lisätä ostoksia. [14.]

Apple App Store Connectin alustus

Apple App Storeen täytyy ensiksi luoda sovellukselle App ID. Jos sitä ei ole ennalta olemassa. App ID:tä käytetään sovelluksen ja sen sisäisten ostosten linkkaamiseen. App ID:n voi luoda Apple Developer Center-näkymässä valitsemalla Certificates, IDs & Profiles. App ID:tä luodessa pitää antaa sovelluksen nimi, valita sen App ID ja antaa uniikki Bundle ID. Seuraavaksi tulee hyväksyä Applen ”Apple Developer Program License Agreement” -sopimus ja App Store Connectin “Paid Applications” -sopimus. Kuvassa 2 on Apple Developer Consolen näkymä, josta voi valita sopimukset hyväksyttäväksi.



Kuva 2. Päänäkymä Apple Developer Consolessa. Kohdasta Agreements, Tax and Banking voidaan hyväksyä sopimukset [17].

Sopimukset löytyvät kohdasta Agreements, Tax, and Banking. Kun ne on hyväksytty, on mahdollista lisätä uusia maksullisia sovelluksia ja sovellusten sisäisiä tuotteita Apple App Store -kauppaan.

3.2 Tuotteet

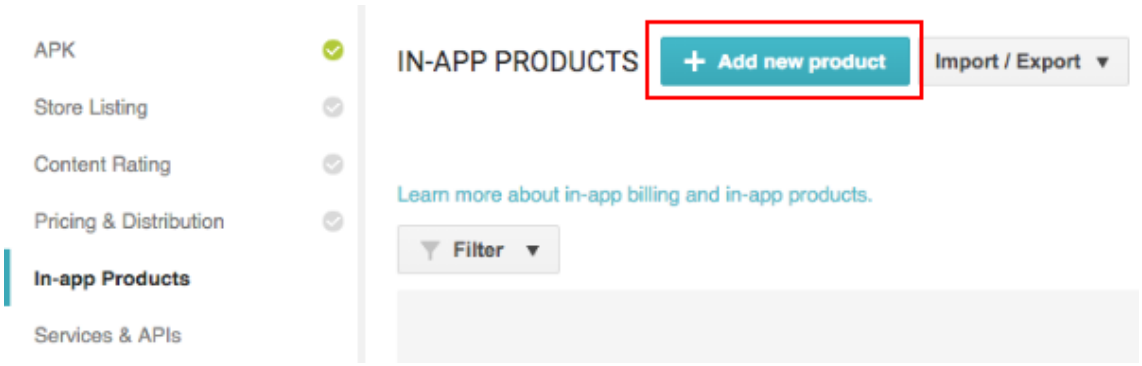
Tuotteita on kolmea eri tyyppiä. Consumable-tuotteita käyttäjä voi ostaa toistuvasti sovelluksen sisällä. Niitä ei voi palauttaa käyttäjälle Google Play- tai Apple AppStore -kauppojen kautta, sillä ne eivät seuraa, omistaako asiakas kyseisen tuotteen. Esimerkkinä Consumable-tuotteesta ovat sovelluksen sisäiset valuutat. Consumable-tuotteista tulee pitää huolta sovelluksen tai oman palvelimen puolella. [15.]

Non-Consumable-tuotteet ovat samankaltaisia kuin Consumabe-tuotteet, mutta niitä voi ostaa vain kerran. Non-Consumable-tuotteen ostamisen jälkeen käyttäjä saa käyttöönsä ostamansa tuotteen pysyvästi. Tuotteet voidaan palauttaa asiakkaalle Google Play- ja Apple AppStore -kauppojen kautta tarkastamalla niiden tiedot. Non-Consumable-tuotteet ovat yleensä esimerkiksi lisäsisältöä sovellukseen tai mainosten poistamista sovelluksesta. [15.]

Subscription-tyyppiset tuotteet ovat tilauksia, joilla käyttäjä saa pääsyn tiettyyn sisältöön tai tuotteeseen rajatuksi aikaa. Tilaukselle voi asettaa, miten usein tilauksesta pitää maksaa, esimerkiksi kuukausimaksua. Subscription-tuotteet on myös mahdollista palauttaa Google Play- ja Apple AppStore -kauppojen kautta asiakkaalle käyttöön. Tuotteita voi olla kahdentyyppisiä: Non-Renewing Subscription ja Auto-Renewing Subscription. Non-Renewing Subscription on vain tietyn määräajan mittainen, kun Auto-Renewing Subscription uusiutuu tietyn kuluneen ajan jälkeen. [15.]

Tuotteet Google Playssa

Google Play Consolen kautta voi lisätä sovellukseen myytäviä tuotteita, jotka on mahdollista hakea myöhemmin sovelluksen kautta ja varmistaa tuotteiden oikea toiminnallisuus. Kuvassa 3 uusia tuotteita voi lisätä Google Play Consolessa valitsemalla sovelluksen, johon tuotteita haluaa lisätä ja valitsemalla In-app Products. [16.]



Kuva 3. Uuden tuotteen lisääminen Google Play Consolen näkymästä [16].

Managed products -kohdassa voidaan luoda uusia Consumable- tai Non-Consumable-tyyppisiä tuotteita sovellukseen valitsemalla näkymästä Create Managed Product. Sovellukseen lisättävien tuotteiden täytyy noudattaa Google Play Developer Program Policies -sääntöjä, ja kaikkiin myytyihin tuotteisiin sisältyy palvelumaksu. [16.]

Uusia tuotteita lisättäessä tuotteille tulee antaa Product ID. Product ID:iden tulee olla sovellukselle uniikkeja, eikä niitä voi muokata enää myöhemmässä vaiheessa. Product ID voi sisältää vain pieniä kirjaimia (a–z), numeroita (0–9), alaviivoja (_) ja pisteitä (.). Tuotetta lisätessä tulee lisätä myös tuotteen nimi, joka näkyy tuotetta ostaessa sovelluksen sisäisessä kaupassa, selitys, mikä tuote on ja mitä sen ostamalla saa, ja tuotteen hinta. [16.]

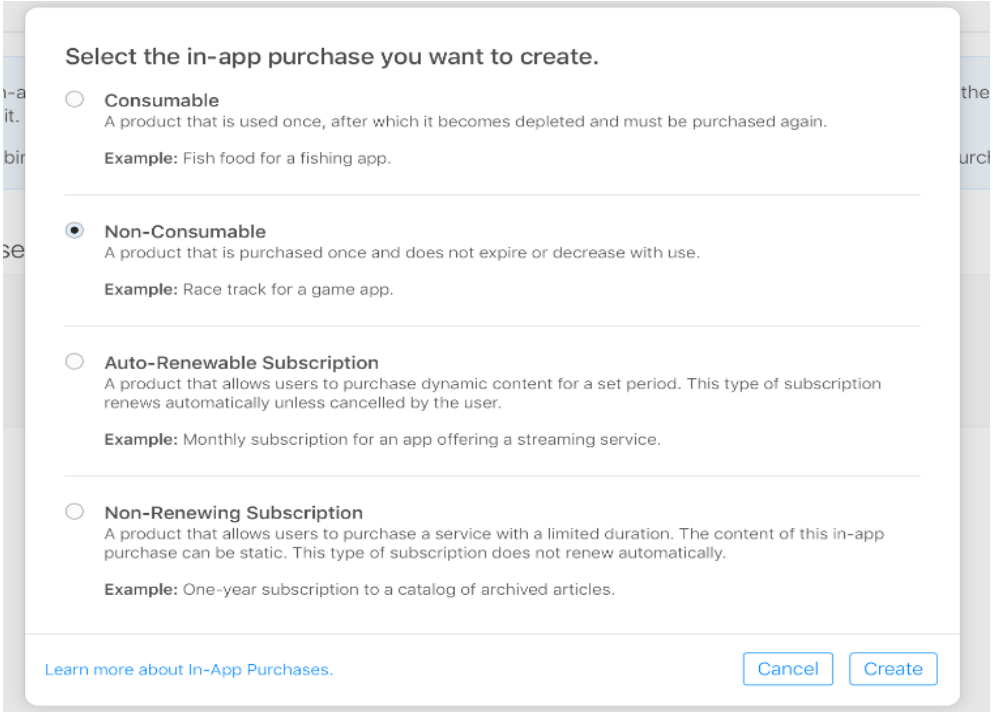
Tuotetta lisättäessä voi myös valita tuotteen tilan, joka voi olla käytössä tai ei käytössä. Jotta tuote on ostettavissa, sen täytyy olla aktiivinen ja sovelluksen, johon tuote on lisätty, täytyy olla julkaistu. Näkymästä voi lisätä myös käännöksiä eri kielille tuotteen nimestä ja selityksestä. [16.]

Tilaus-tyyppistä tuotetta lisättäessä aluksi asetetaan tilauksen nimi, selvennys, minkälainen tilaus on kyseessä, ja sen hinta. Tilaustuotteesta pitää myös kertoa sen tilauskauden pituus: kuinka usein tilauksen hinta täytyy maksaa uusiksi, jotta tilaus pysyy aktiivisena. Tilaustuotetta lisätessä voi myös valita alustavan hinnoittelun tilaukselle, jossa tilauksen ensimmäiset kuukaudet ovat halvempia asiakkaalle. Minimiaika alustettuun hinnoitteluun on 3 päivää ja maksimi 12 kuukautta. Alustavan hinnoittelun hinta ei voi ylittää tuotteen normaalia hintaa. [16.]

Tilaustuotteeseen voidaan myös tarjota tilauksen ilmainen kokeilujakso. Se kestää sille asetetun ajan, jonka jälkeen se muuttuu automaattisesti normaaliksi maksulliseksi tilaukseksi. Käyttäjä voi valita ilmaisen kokeilujakson vain kerran, jos hän ei ole aikaisemmin ostanut kyseistä tilaustyypin tuotetta. [16.]

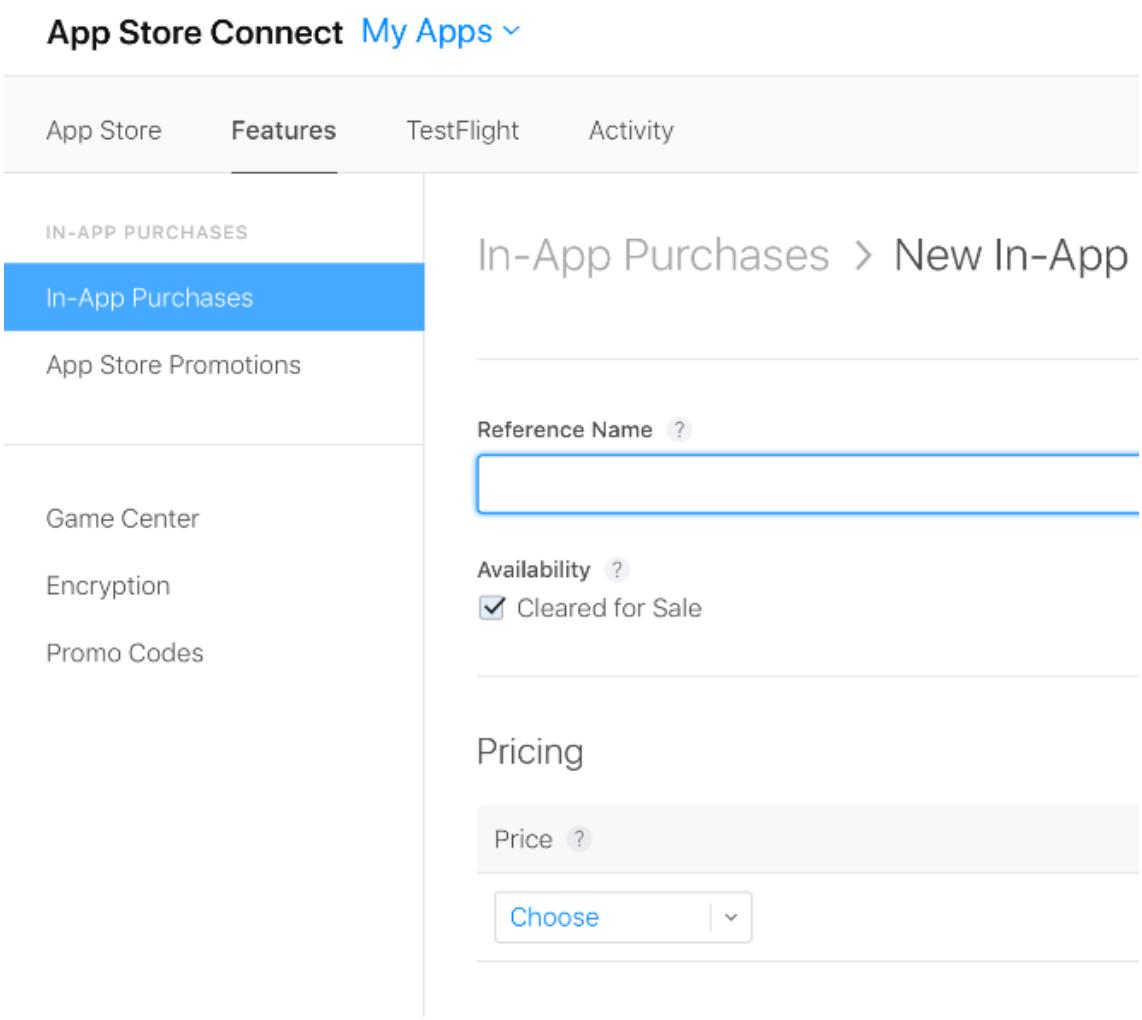
Tuotteet Apple App Storessa

Apple App Storeen voi lisätä tuotteita näkymästä Features. In-App Purchases -näkymä löytyy, kun valitaan haluttu sovellus Apple App Store Connect -näkymästä. Kuvassa 4 olevassa näkymässä, halutaanko lisätä Consumable-, Non-Consumable-, Auto-Renewable Subscription- vai Non-Renewing Subscription -tyyppinen tuote. Tuotteet, joita tästä näkymästä lisätään, voivat olla pelkästään digitaalisia tuotteita Applen sopimuksen mukaisesti. [17.]



Kuva 4. Tuotteen tietojen syöttäminen App Store Connectin konsolinäymässä [17].

Kuvassa 5 olevassa New In-App purchase -näkymässä Apple App Store -tuotteille tulee antaa tietoja, joita käytetään tuotteen myymisessä. Vaadittuja tietoja ovat Reference Name, tuotteen nimi, joka näytetään Apple App Storessa, Product ID, tuotteen uniikki tunniste, jolla tuote tunnistetaan sovelluksen sisällä. Cleared for Sale -kohdassa voi valita, onko tuote myytävänä IAP:ssa. Pricing-kohdassa voidaan valita tuotteen hinta valitsemalla sen Pricing Tier.



Kuva 5. Tuotteen tietojen syöttäminen App Store Connectin konsolinäkymässä [17].

Localizations-kohdassa voidaan valita tuotteen näytettävä nimi ja selitys tuotteesta sekä lisätä käännöksiä eri kielille. Viimeiseksi lisättäessä tuotetta Apple App Storeen pitää tuotteen lisäyksen yhteydessä antaa tietoja tuotteen App Review -arviointiin, joka täytyy läpäistä, ennen kuin tuote tulee myytäville sovelluksessa. Tuotteesta pitää olla kuva-kaappaus sekä selitys siitä, mikä tuote on ja mitä ollaan myymässä. [17.]

4 Sovelluksen sisäisen maksujärjestelmän luonti

Tässä luvussa perehdytään itse maksujärjestelmän toteutukseen Unity-pelimoottorilla ja siihen miten Unity IAP -liitännäinen otetaan käyttöön pelimoottorin sisällä. Ensimmäiseksi luvussa käydään läpi työkalut, joita toteutuksessa käytettiin, alustetaan Unity IAP -liitännäinen käyttöön ohjelmakoodissa ja Unity-editorissa sekä perehdytään erilaisten tuotteiden lisäämiseen sovelluksen sisäiseen maksujärjestelmään. Luvussa perehdytään myös ostosten kuittien varmistamiseen ja turvalliseen tapaan myydä tuotteita omassa sovelluksessa. Lopuksi käydään läpi maksujärjestelmän toiminnan testaaminen ja käyttöliittymä.

4.1 Työkalut

Maksujärjestelmä toteutettiin Unity-sovellukseen, joka on saatavilla Android- ja iOS-laitteille Google Play- ja Apple App Store -kauppojen kautta. Unity IAP -liitännäistä käytettiin sovelluksen sisäisten toimintojen toteuttamiseen.

Sovelluskoodi toteutettiin käyttäen C#-ohjelmointikieltä ja Unity-pelimoottorin, sekä Unity IAP:n tarjoamia kirjastoja. Kauppapaikan käyttöliittymä toteutettiin käyttäen Unity-pelimoottoria. Kauppapaikka toteutettiin käyttäen Google Play- ja Apple App Store - maksupalveluita maksunvälittäjänä ja maksupalveluna.

Unity

Unity on pelimoottori, jota kehittää Unity Techonologies [18]. Sitä voidaan käyttää pelien kehittämiseen yli 25:lle eri alustalle. Pelimoottoria voi käyttää laajasti erilaisten pelien ja sovellusten tekemiseen, kuten kolmiulotteisten sovellusten ja virtuaalitodellisuuden tai lisätyn todellisuuden sovelluksiin. Unity-pelimoottori tarjoaa paljon erilaisia työkaluja valmiiksi sovelluskehitykseen, sovelluksen kaupallistamiseen ja analytiikan keräämiseen. [18.]

Unity sisältää myös Unity Asset Store -kauppapaikan, josta voi ostaa pelimoottoriin liitännäisiä, kuten työkaluja tai valmiita osia pelistä tai sovelluksesta. Tämä nopeuttaa huomattavasti kehittäjän työtä, sillä osan tarvittavasta toiminnallisuudesta voi hankkia Unity Asset Storen kautta.

Unity IAP

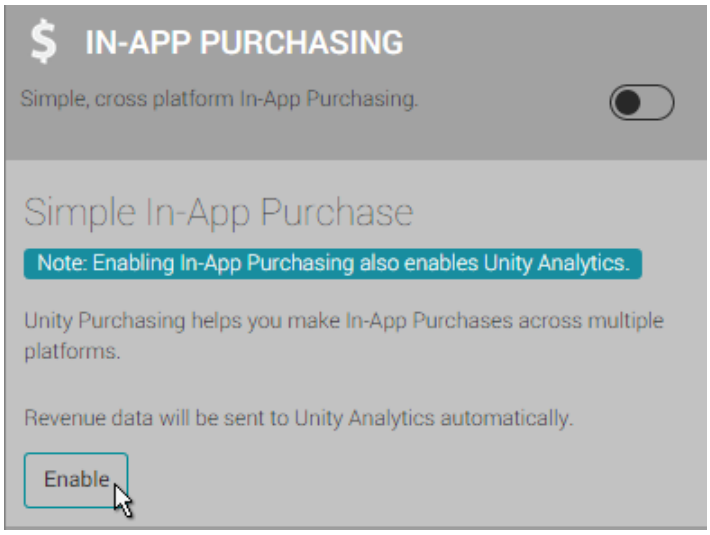
Unity IAP on Unity-pelimoottorin liitännäinen, jonka pystyy lataamaan Unity-pelimoottorin kautta. Unity IAP täytyy konfiguroida käyttöön mahdollisten maksupalveluiden kanssa ja ohjelmoida toimimaan sovelluksen sisälle, jotta sovelluksen sisäistä maksujärjestelmää voidaan käyttää. [19.]

Unity IAP -liitännäisellä voi mahdollistaa maksujärjestelmän toimivuuden useilla eri alustoilla, ja se tukee valmiiksi ostoksia kymmenellä alustalla. Unity IAP -liitännäisellä pystyy myös seuraamaan sovelluksen myyntiä siihen liitetyn Unity Analytics -kirjaston kautta. [19.]

4.2 Unity IAP:n käyttöönotto

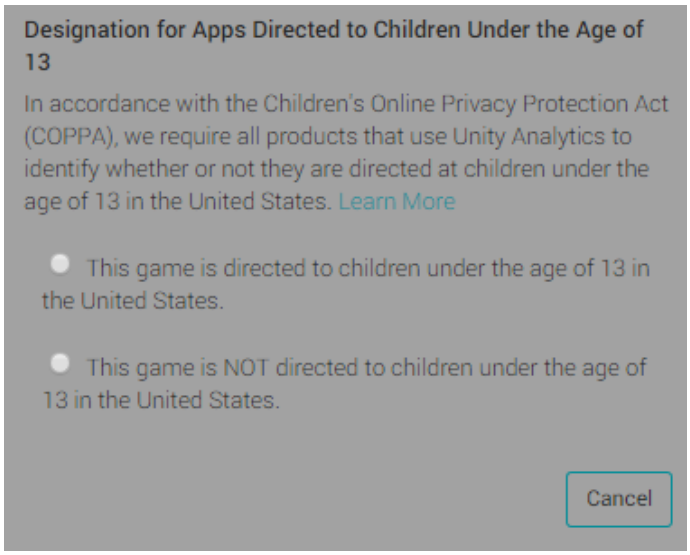
Unity IAP -liitännäisen voi lisätä Unity-editorin sisältä Services-välilehdestä. Unityyn täytyy olla kirjautunut sisään käyttääkseen Services-välilehden toimintoja. Tämän jälkeen täytyy valita organisaatio, johon projekti kuuluu, ja luoda projektille oma Unity Project ID painamalla ”Create”. Mikäli projektilla on jo olemassa Unity Project ID, sen voi valita myös olemassa olevaan projektiin kohdasta ”I already have a Unity Project ID”. [20.]

Kuvassa 6 In-App Purchasing -välilehdellä pystyy ottamaan Unity IAP -liitännäisen käyttöön projektissa ja lisäämään liitännäisen projektiin näppäimestä ”Import”. In-App Purchasing -välilehdestä löytyy myös kohta, johon voi lisätä sovelluksen Google Play -avaimen. Sen avulla liitännäinen pystyy välittämään oikeat tiedot Google Play -kauppaan sovelluksen tuotoista ja muusta analytiikasta. Tämä kohta ei ole pakollinen, jos ei halua käyttää Google Play -palveluja sovelluksen sisäisten ostosten toteuttamiseen.



Kuva 6. Unity IAP-liitännäisen käyttöönotto Unity-editorissa.

Kuvan 7 näkymässä Unity IAP -liitännäistä lisätessä täytyy myös sitoutua COPPA (Children’s Online Privacy Protection Act)-säädöksiin ja ilmoittaa, onko sovellus suunnattu lapsille, jotka ovat alle 13 vuotta vanhoja. Tämä laki määrittää Yhdysvalloissa, ettei sovellus, joka on suunnattu alle kolmetoistavuotiaille, saa kerätä heistä tietoja. [20.]

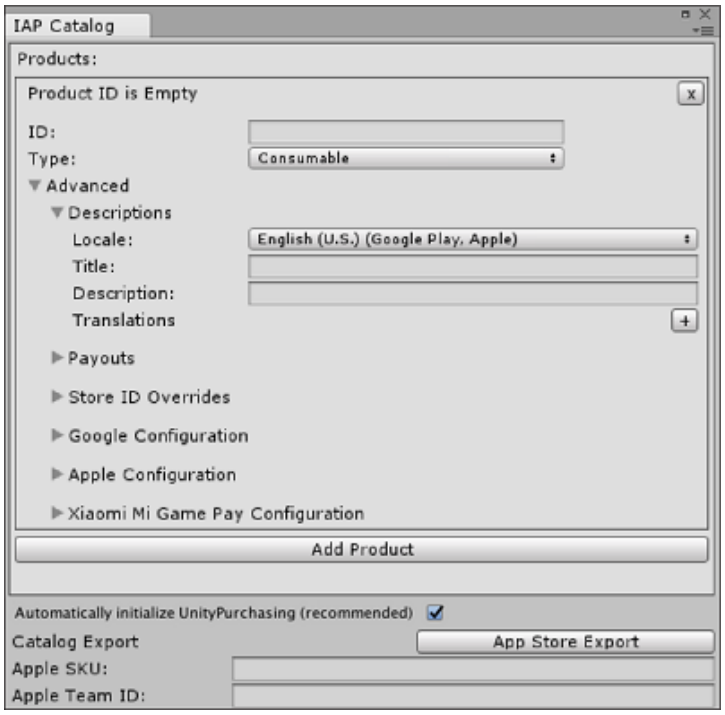


Kuva 7. COPPA (Children's Online Privacy Protection Act)-sopimuksen hyväksyminen Unity IAP -liitännäisen alustuksessa.

Unity IAP -liitännäinen ei toimi ilman, että ottaa käyttöön myös Unity Analytics -liitännäisen. Unity Analytics -liitännäisen voi ottaa käyttöön Services-välilehdeltä valitsemalla sen, jos Unity Analytics ei automaattisesti aktivoidu Unity IAP -liitännäisen mukana.

4.3 Tuotteet Unity IAP:ssa

Tuotteita joita sovelluksen sisällä myydään, tulee lisätä myös käyttöön sovelluksen sisällä käyttäen Unity IAP:ta. Tuotteita voidaan lisätä sovelluksen sisäiseen maksujärjestelmään muutamalla eri tavalla. Ensimmäinen tapa on käyttää Unity IAP:n tarjoamaa Codeless IAP Catalog -näkymää, joka löytyy Unity-editorin sisältä kohdasta Window > Unity IAP > IAP Catalog. Kuvassa 8 olevassa IAP Catalog -näkymässä voidaan luoda uusia tuotteita sovelluksen sisäiseen koodiin. IAP Catalog -näkymässä asetetaan tuotteen julkaisualustasta riippumaton ID, jolla se pystyy olemaan yhteydessä tuotteen maksupalveluun, sekä tuotteen tyyppi. Tämän ID:n tulisi vastata Google Play- ja Apple App Store -palveluissa lisättyjen tuotteiden ID:itä. [21.]



Kuva 8. Unity IAP -liitännäisen Codeless IAP Catalog -näkymä, jossa voidaan lisätä tuotteita lisäämättä niitä ohjelmakoodissa.

Advanced-kohtaan voidaan lisätä tuotteelle myös tieto, missä maissa tuotetta myydään, tuotteen nimi, joka näkyy sovelluksen sisäisessä kaupassa, ja kuvaus tuotteen sisällöstä. Payouts-kohdassa voidaan luoda tuotteelle valmiiksi erilaisia määritteitä, mitä toimitetaan sovellukseen onnistuneen ostotapahtuman jälkeen, kuten tuotteen tyyppi ohjelmakoodissa ja sen määrä. Payouts-kohdan Data-kenttään voi myös lisätä tietoa, joka halutaan toimittaa sovellukselle ostotapahtuman onnistuttua, esimerkiksi jos jokin UI-elementti halutaan korostaa.

Unity IAP otetaan ohjelmakoodissa käyttöön UnityEngine.Purchasing kirjastosta. Ohjelmakoodissa täytyy tehdä IStoreListener-luokan aliluokka, jolla Unity IAP -liitännäistä käytetään. Ensimmäiseksi luokan tulee kutsua InitializePurchasing-metodia, jossa luodaan StandardPurchasingModule- ja ConfigurationBuilder-luokkien oliot. Tämän jälkeen kutsutaan UnityPurchasing.Initialize-metodia, jolloin liitännäinen suorittaa alkutoiminnot.

Tuotteiden alustaminen ohjelmakoodissa tehdään InitializePurchasing-metodissa sen jälkeen, kun ConfigurationBuilder on alustettu. Tuotteita voidaan lisätä ConfigurationBuilderiin sen metodilla AddProduct. AddProduct-metodille annetaan parametreiksi tuotteen ID, kuten se on Google Play- tai Apple Appstore -konsolissa asetettu, sekä tuotteen tyyppi.

Esimerkkikoodissa 1 InitializePurchasing-metodissa tarkastetaan myös mahdollinen alusta, jolla sovellus on käytössä. Se tehdään tarkastamalla Unity Enginen sisäinen muuttuja Application.platform ja StardardPurchasingModulen käytössä olevan instanssin App Store. InitiatePurchasing-metodissa myös alustetaan sovelluksen sisäisten salausavaimien häivytys käyttäen CrossPlatformValidator-luokkaa.

```
//Initialize the different products from play store to Unity IAP
public void InitializePurchasing()
{
      if(IsInitialized())
      {
            return;
      }
      var module = StandardPurchasingModule.Instance();
      var builder = ConfigurationBuilder.Instance(module);

      //Add products and subscriptions from Google Play and Apple App Store
      builder.AddProduct(productName, ProductType.Consumable);
      builder.AddProduct(subscriptionName, ProductType.Subscription, new IDs(){
            {kProductNameAppleSubscription, AppleAppStore.Name },
            {kProductNameGooglePlaySubscription, GooglePlay.Name },
      });
string appIdentifier = Application.identifier;
//Load CrossPlatformValidator for receipt validation
validator = new CrossPlatformValidator(GooglePlayTangle.Data(),
      AppleTangle.Data(), UnityChannelTangle.Data(), appIdentifier)

//Initialize UnityPurchasing
UnityPurchasing.Initialize(this, builder);

}
```

Esimerkkikoodi 1. Unity IAP:n alustaminen koodissa Purchaser-nimikseksi luokaksi, joka perii IStoreListener-luokan sekä tuotteiden lisäämisen.

Jos alustus onnistuu, kutsutaan automaattisesti callback-metodina luokan OnInitialized-metodia. OnInitialized-metodissa luokalle Purchaser asetetaan IStoreController- ja IExtensionsProvider -oliot paikalleen, ja ostaminen on mahdollista. Ostaminen on sovelluksessa hyvä lukita alustusprosessin ajaksi, jottei viallisia ostoksia pääse tapahtumaan alustusvaiheessa. Jos alustus epäonnistuu, kutsutaan callback-metodina OnInitializeFailed-metodia. Esimerkkikoodissa 2 esitellään BuyProductID-metodi, jolla voidaan aloittaa ostostapahtuma valittuun tuotteeseen.

```
void BuyProductID(string productid){
{
      if(IsInitialized())
      {
            Product product = controller.products.withID(productid);
            if(product != null && product.availableToPurchase)
            {
                  controller.InitiatePurchase(product);
            }
            else
            {
                  //Failed purchasing product, product is not found or available
                  for purchase
            }
      }
      else
      {
      //Failed purchasing product, Purchaser is not initialized
      }
}
```

Esimerkkikoodi 2. BuyProductID-metodi, jolla ostotapahtuma voidaan aloittaa.

Ohjelmakoodissa kutsutaan IStoreControllerin InitiatePurchase-metodia käyttäen parametrinä tuotteen Product ID:tä. Tämä komento voidaan lisätä esimerkiksi käyttöliittymän näppäimeen, jolla aloitetaan ostotapahtuma.

4.4 Tilaukset

Unity IAP SDK 1.19:ssä alettiin tukea tilaustuotteita SubscriptionManager-luokan kautta. SubscriptionManager-luokalla pystyy ohjelmakoodissa saamaan tietoja tilauksesta, kuten milloin tilaus vanhenee tai uusiutuu tai milloin tilaus on ostettu. Esimerkkikoodissa 3 olevassa OnInitialized-metodissa voidaan tarkastaa tuotteiden tietoja IStoreController-oliolta. Tuotteet voidaan hakea listasta käymällä läpi controller.products.all-lista. [22.]

```
//Callback when the store has been initialized
public void OnInitialized(IStoreController controller, IExtensionProvider ex-
tensions)
{
      Purchaser.controller = controller;
      Purchaser.extensions = extensions;

      Dictionary<string, string> introductory_info_dict = extensions.GetExten-
sion<IAppleExtensions>().GetIntroductoryPriceDictionary();

      foreach(var item in controller.products.all)
      {
            if(item.availableToPurchase)
            {
                  //Usage of SubscriptionManager class
                  if(item.recept != null)
                  {
                        if(item.definition.type == ProductType.Subscription)
                        {
                              if(CheckSubscriptionManagerAvailabily(item.receipt=
                              {
                                    string intro_json = (introductory_info_dict
==null || !introductory_info_dict.ContainsKey(item.definition.storeSpeci-
ficId))? null : introductory_info_dict[item.definiton.storeSpecificId];
                                    SubscriptionManager sm = new SubscriptionMaa-
ger(item, intro.json);
                                    SubscriptionInfo info = sm.getSubscription-
Info();
                              else
                              {
                                    //This product is not available for Subscrip-
tionManager class, purchased procuts need to use 1.19+ SDK.
                              }
                        else{
                              //The product is not a subscription product
                        }
                  else{
                        //The product should have a valid receipt
                  }
            }
      }
}
```

Esimerkkikoodi 3. OnInitialized-metodi, jossa käytetään SubscriptionManageria.

Listasta voidaan hakea Subscription-tyyppistä tuotetta. Mikäli Subscription-tyyppinen tuote löytyy listasta SubscriptionManager voi luoda SubscriptionInfo tyyppisen olion tuotteesta, käyttämällä getSubscriptionInfo-metodia. SubscriptionInfo-metodilta voidaan hakea tietoja tilaustuotteesta.

4.5 Kuitit ja turvallisuus

Unity IAP tarjoaa kuitin sovelluksen sisäisistä ostoksista JSON-tauluna. Tämä kuitti sisältää seuraavat arvot: Store on käytössä olevan maksupalvelun nimi, kuten Google Play tai Apple App Store. TransactionID on tapahtuneen maksutapahtuman uniikki tunniste, jonka kauppa tarjoaa. Payload-muuttujan sisältö vaihtelee alustoittain: Google Playssä Payload on JSON-taulu, joka sisältää Google Playn tarjoaman kuitin JSON-datana sekä allekirjoituksen JSON-muuttujaan. [23.]

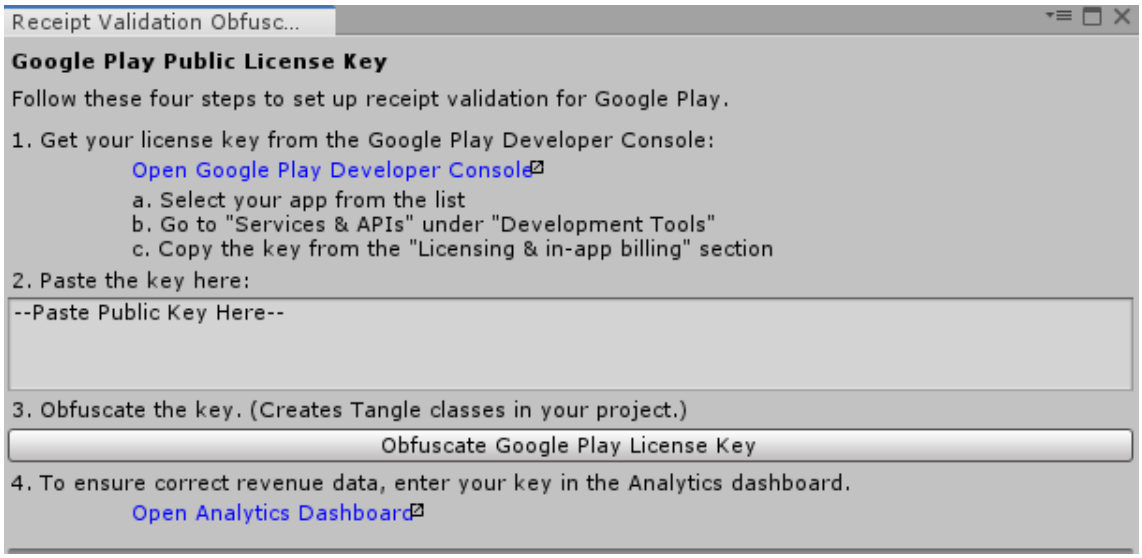
iOS-versiossa 7.0 ja sen jälkeen Payload-arvona on base 64 -tyyppinen kuitti Apple App Storesta. Aikaisemmassa versiossa iOS käyttää Payloadina SKPaymentTransaction-tyyppistä objektia, joka sisältää kuitin tiedot.

Tehtyjen ostosten kuitit kannattaa validoida, etteivät käyttäjät pääse mahdollisesti sisältöön, jota he eivät ole ostaneet. Parasta on validoida kuitti siellä, missä sovelluksen sisältö on tarjolla. Etävalidoinnissa sisällölle, joka on palvelimen puolella ja ladataan sovellukseen, kun se on ostettu, validoinnin tulisi tapahtua palvelimella, ennen kuin tuote on julkaistu asiakkaalle. Unity IAP ei tue etävalidontia, mutta sitä varten on kolmannen osapuolen kirjastoja, joita voi käyttää. [23.]

Paikallisessa validoinnissa sisällölle, joka on asiakkaan puolella, jossa kaikki sisältö sovelluksessa on ostettu ja käytössä, validoinnin tulisi sijaita laitteella, ilman että on tarvetta ottaa yhteyttä palvelimeen. Jos sisältö, jonka käyttäjä on ostanut, sijaitsee jo valmiiksi laitteella, sovelluksen täytyy tehdä päätös, että se avaa sisällön asiakkaalle. Unity IAP tarjoaa työkaluja, jotka auttavat sisällön piilottamisessa ja kuittien käsittelyssä sekä validoinnissa. [23.]

Kuitin validointi tehdään käyttäen tunnettuja salausavaimia. Sovellukselle tämä on joko salattu Google Playn julkinen avain tai Applen pääsertifikaatti. Jos käyttäjä pystyy vaihtamaan nämä avaimet, on mahdollista, että käyttäjä voi kiertää kuitin validointitarkastukset, joten on tärkeää, että käyttäjä ei voi helposti löytää tai muokata näitä avaimia.

Unity IAP tarjoaa työkalun, jolla voi häivyttää sovelluksen sisäiset salausavaimet. Tämä työkalu sekoittaa avaimet siten, että käyttäjän on paljon vaikeampi päästä niihin käsiksi. Työkalu löytyy kohdasta Window > Unity IAP > IAP Receipt Validation Obfuscator. Receipt Validation Obfuscator (kuva 9).



Kuva 9. Receipt Validation Obfuscator -näkymä Unity IAP -liitännäisessä.

Näkymässä voidaan koodata Google Playn julkinen avain ja Applen pääsertifikaatti kahdeksi uudeksi C#-tiedostoksi, AppleTangle ja GooglePlayTangle. Ne on lisätty tämän jälkeen käyttöön projektissa. Google Playn julkinen avain löytyy Google Play Developer Consolen Services & APIs-näkymästä. Applen pääsertifikaatti on valmiiksi käytössä Unity IAP:n kanssa. [23.]

Esimerkkikoodissa 4 ProcessPurchase-metodissa tarkastetaan kuitti käyttäen CrossPlatormValidator-luokan metodia Validate. CrossPlatformValidator-luokkaa voidaan käyttää molempien, Google Play- ja Apple App Store -kaupan kanssa.

```
public PurchaseProcessingResult ProcessPurchase(PurchaseEventArgs e)
{
      bool validPurchase = true;

      //Unity IAP's validation logic is only included on these platforms
#if UNITY_ANDROID || UNITY_IOS || UNITY_STANDALONE_OSX
      //Prepare the validator with the secrets we prepared in the editor obfus-
cation window
      var validator = new CrossPlatformValidator(GooglePlayTangle.Data(), Ap-
pleTangle.Data(), Application.identifier);
      try
      {
            var result = validator.validate(e.purchasedProduct.receipt);
      }
      catch
      {
            //Invalid recept, not unlocking content
            validPurchase = false;
      }
#endif
      if(validPurchase)
      {
            //Unlock the appropriate content here.
            if(String.Equals(e.purchasedProduct.definition.id, subscriptionName,
StringComparison.Ordinal)
                  //Validation successful with subscription product.
            }
            else if(String.Eqauls(e.purchasedProduct.definition.id, productName,
StringComparison.Ordinal)
            {
                  //Validation successful with product.
            }
            else
            {
                  //Validation failed, unrecognized product
            }
      }
      return PurchaseProcessingResult.Complete;
}
```

Esimerkkikoodi 4. ProcessPurchase-metodi, jossa tarkastetaan kuitti käyttäen CrossPlatform-Validator-luokkaa

CrossPlatformValidator-luokka suorittaa kaksi tarkastusta. Kuitin aitous tarkastetaan validoimalla kuitin allekirjoitus. Kuitin sovellustunnistetta verrataan sovelluksen omaan sovellustunnisteeseen, ja mikäli ne eivät vastaa toisiaan, syntyy InvalidBundleId-poikkeus. CrossPlatformValidator-luokka toimii vain Google Play- ja Apple App Store -palveluiden kanssa. Muista palveluista tuotetut tai virheelliset kuitit tuottavat IAPSecurityExeptionin.

Jos kuittia yrittää validoida ilman että on asettanut siihen salausavainta, seuraa MissingStoreSecretExeption-poikkeus.

On myös tärkeää, että tarkastaa kuitin validoinnin lisäksi, mitä tietoja kuitti sisältää. Yleinen keino yrittää päästä käsiksi sisältöön ilman tapahtunutta ostosta on tarjota sille kuitteja toisista tapahtuneista ostoksista. Nämä kuitit vaikuttavat aidoilta ja läpäisevät kuitin validoinnin, joten päätöksiä tulisi tehdä esimerkiksi kuitissa olevan tuotteen ID-numeron perusteella, jonka CrossPlatformValidator tuottaa. [23.]

Eri maksupalveluilla on erilaiset kuitit. Jotta pääsee käsiksi kauppakohtaisiin tietoihin, IPurchaseReceipt-luokkaa voidaan käyttää kahdella eri aliluokalla, GooglePlayReceipt ja AppleInAppPurchaseReceipt. Mikäli käytössä on iOS 7.0 tai uudempi, tulee käyttää tähän AppleValidator-luokkaa, jolla voidaan saada tarkat tiedot kuitista.

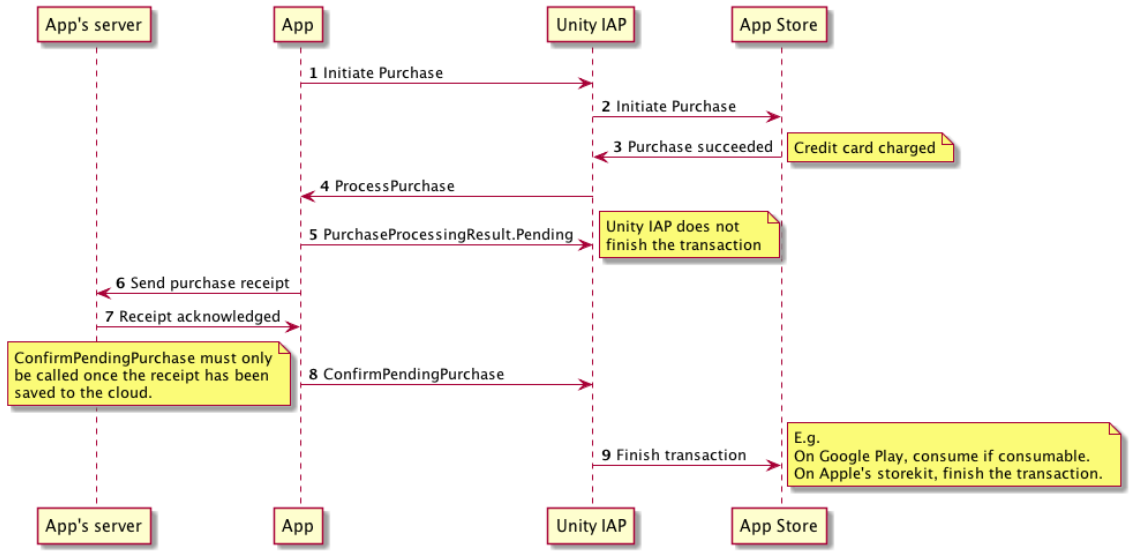
4.6 Ostostapahtuma

Ostostapahtuma aloitetaan kutsumalla InitiatePurchase-metodia antamalla sille parametriksi ostettavan tuotteen ID, minkä jälkeen Unity IAP -liitännäinen aloittaa ostostapahtuman. ProcessPurchases-metodi palauttaa tämän jälkeen PurchaseProcessingResult-arvon, joka kertoo, onko sovellus suorittanut ostostapahtuman loppuun saakka. PurchaseProcessingResult-arvo voi olla Complete, jolloin tuotteen ostaminen on suoritettu eikä siitä ilmoiteta enää uudestaan, tai Pending, jolloin sovellus vielä selvittää ostostapahtumaa. ProsessPurchase-metodia kutsutaan seuraavan kerran automaattisesti, kun sovellus käynnistetään uudestaan, ellei ConfirmPendingPurchase-metodia IStoreControllerista ole kutsuttu.

Unity IAP -liitännäinen vaatii tarkat tiedot ostosten nykyisestä tilasta, jotta voidaan varmistaa ostosten toiminta häiriötilanteissa, kuten internetyhteyden katketessa tai sovelluksen kaatumisessa. Jos ostostapahtuma on onnistunut internetyhteyden katketessa, tieto onnistuneesta ostoksesta lähetetään sovellukselle seuraavalla sovelluksen käynnistyskerralla. [24.]

Ostoksia voidaan suorittaa heti, ilman että tarkastetaan ostoksen kuittia ja onnistunutta tapahtumaa palvelimen kautta. Tätä ei kuitenkaan tulisi käyttää Consumable-tyyppisten tuotteiden kanssa, koska on olemassa riski, että tuotteet häviävät asiakkaalta, jos asiakas poistaa sovelluksen, ennen kuin pilvitallennus tapahtuu. [24.]

Kuvassa 10 suoritetaan ostotapahtuma käyttäen palvelinta kuitin validoimisessa. Jos ostoksia suoritetaan tarkastamalla kuitti palvelimen kautta, tulee palauttaa PurchaseProcessingResult.Pending.



Kuva 10. Ostotapahtuman suoritus käyttäen palvelinta ostotapahtuman tarkastamiseen, jotta sisältö avataan asiakkaan laitteella ostotapahtuman onnistuttua [24].

Kun palvelin on tarkastanut tapahtuman kuitin ja vastannut kuitin olevan oikea, voidaan kutsua ConfirmPendingPurchase-metodia ostostapahtuman suorittamiseksi loppuun asti. Kun käytetään PurchaseProcessingResult-arvoa Pending, Unity IAP -liitännäinen pitää ostotapahtuman odottamassa, kunnes se on suoritettu loppuun saakka. Unity IAP varmistaa, että ostotapahtuma ei katoa, vaikka käyttäjä asentaisi sovelluksen uudestaan ostotapahtuman aikana.

4.7 Palauttaminen

Kun käyttäjä uudelleenasentaa sovelluksen, käyttäjälle tulee antaa oikeudet kaikkiin Non-Consumable- ja uusiutuviin Subscription-tuotteisiin, jotka käyttäjä omistaa. Maksupalvelut pitävät kirjaa jokaisen asiakkaan omistamista tuotteista, joten Unity IAP -liitännäinen pystyy hakemaan tiedot tuotteista Google Play- tai Apple App Store -palveluista. Ei-uusiutuvia tilauksia ei voi hakea Apple-alustoilla Unity IAP:n kautta, joten aktiivisista tilauksista tulee pitää kirjaa ja synkronisoida ne laitteiden välillä. [25.]

Tuotteiden palauttaminen tehdään kutsumalla esimerkkikoodissa 5 olevaa ProcessPurchase-metodia IStoreListener-luokassa. OnTransactionsRestored-metodia kutsutaan, kun tuotteet on onnistuneesti palautettu maksupalvelun kautta.

```
public void RestorePurchases()
{
      if(IsInitialized())
      {
            if(IsGooglePlayStoreSelected)
            {
                  IGooglePlayStoreExtensions googleExtensions = exten-
sions.GetExtensions<IGooglePlayStoreExtensions>();
                  googleExtensions.RestoreTranstactions(OnTransactionsRestored);
            }
            else
            {
                  IAppleExtensions appleExtensions = extensions.GetExten-
sions<IAppleExtensions>();
                  appleExtensions.RestoreTransactions(OnTransasctionsRestored);
            }
      }
}
```

Esimerkkikoodi 5. RestorePurchases-metodi, jota kutsumalla voidaan palauttaa ostokset Apple App Store -tuotteisiin.

Google Play-alustalla Unity IAP palauttaa automaattisesti kaikki tuotteet, joita käyttäjä omistaa Unity IAP:n alustusvaiheessa, sovelluksen käynnistyessä. iOS-alustalla käyttäjän täytyy syöttää salasana, jotta aikaisemmat ostotapahtumat voidaan palauttaa. Sovelluksen pitää sisältää esimerkiksi näppäin, joka mahdollistaa, että salasana voidaan syöttää ja aikaisemmat ostokset voidaan palauttaa.

4.8 Testaus

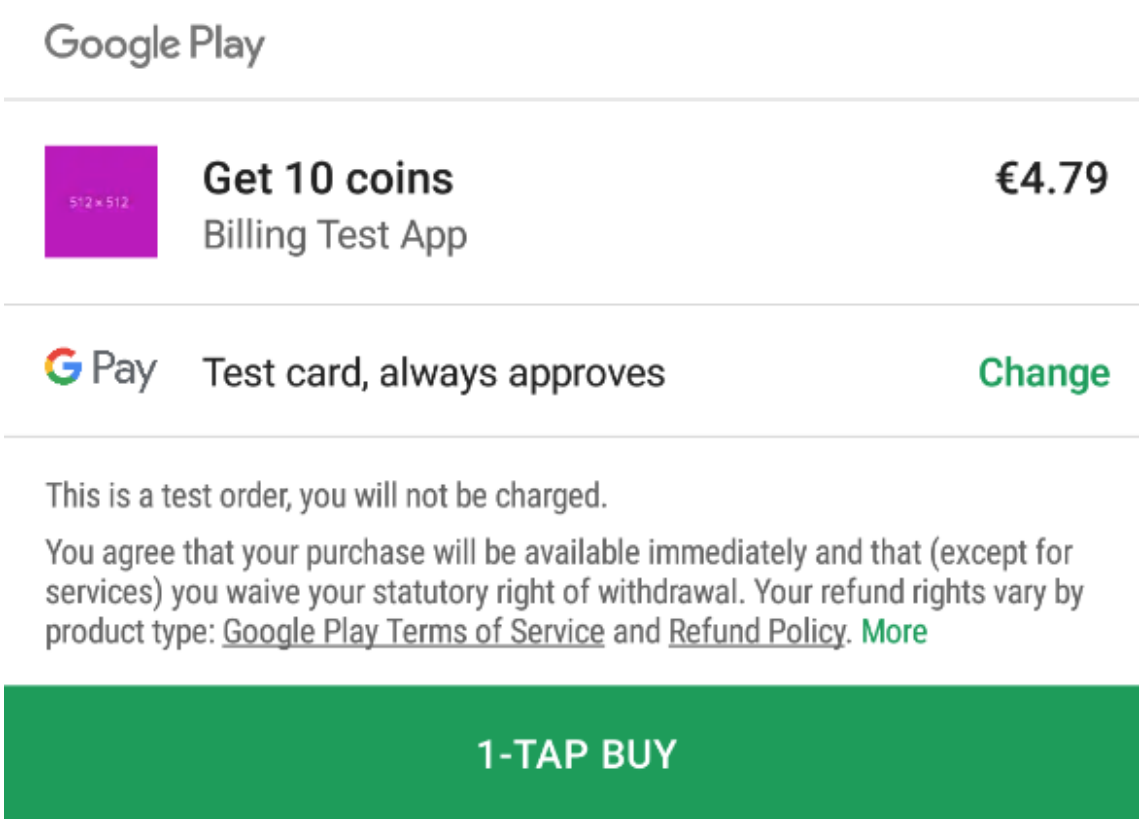
Google Play Billing tarjoaa valmiiksi muutamia varattuja Product ID:itä, joita käyttämällä sovellusta voi testata, ennen kuin antaa mahdollisille testaajille pääsyn sovellukseen. Näitä testi-Product ID:itä käyttämällä voi tarkistaa, että ostamiseen ja palauttamiseen liittyvät sovelluskutsut toimivat halutulla tavalla. Valmiita Product ID:itä käyttäessä tehdään kutsu varattuun Product ID-arvoon. Jokainen niistä palauttaa ennalta määritetyn vastauksen, eikä ostoksissa siirry rahaa. Testaamiseen tarkoitettuja Product ID-avaimia ei voi käyttää tilaustyyppisten ostosten testaamiseen. [26.]

Varattuja Product ID:itä on kolme erilaista: Kun tekee Google Play Billing -kutsun käyttäen android.test.purchased-product ID:tä, Google Play vastaa onnistuneella tuotteen ostoksella, ja vastaus sisältää myös JSON-muuttujan, joka sisältää tietoa ostoksesta. Kun lähettää kutsun android.test.canceled-tuotteeseen, Google Play vastaa kertomalla tuotteen ostoksen olevan peruutettu. Tämä voi tapahtua, jos ostoksen yhteydessä tapahtui virhe, esimerkiksi annettu luottokortti oli virheellinen. Tuotteesta android.test.item_unavailable tulee vastaus, jossa haettu tuote ei ole sovelluksen tuotelistalla saatavilla. Näiden valmiiden tuotekoodien testaamisen jälkeen on hyvä siirtyä koko ostoksen testaamiseen. Kun on havaittu, että valmiiksi tarjotut Product ID:t toimivat ja että allekirjoituksen varmenne toimii, voidaan sovellusta testata oikeilla ostoksilla. [26.]

Kun Google Play on asetettu toimimaan sovelluksen puolella, sitä voidaan testata Google Play Billingin kautta valmiilla testausmenetelmällä. Sovellus täytyy julkaista suljettuna tai avoimena beetatestauksena Google Play Consolessa. Sovelluksen julkaisun jälkeen voi kestää muutamia tunteja, ennen kuin sovellus on testattavissa.

Jokaisen testaajan tulisi liittyä sovelluksen testaamiseen Google Play Consolen kautta lähetettävällä testausviestillä. Viestissä selitetään, mitä merkitsee olla sovelluksen testaaja, ja se sisältää linkin, jolla testaajaksi voi liittyä. Testattava sovellus tulee asentaa Android-laitteeseen. Testaajia voidaan myös asettaa Google Play Developer Consolen kautta asettamalla testaajan Gmail-tilin osoite arvoon Tester.

Testaajat voivat nyt ostaa tuotteita sovelluksen sisältä normaalisti. Testatessa tulisi tehdä ainakin kaksi ostosta, yksi käyttäen maksukeinoa, joka aina hyväksyy ostoksen, joka on esitelty kuvassa 11, toinen ostos tulisi tehdä käyttäen keinoa, joka hylkää aina ostoksen. Nämä maksukeinot näkyvät testaajiksi merkityille tileille. Käytettäessä näitä testausmaksukeinoja ostoksista ei siirry rahaa eikä veroja lasketa ostotapahtuman yhteydessä. Google Play kertoo testiostoksesta näyttämällä ilmoituksen ostosnäkymässä. Tämän jälkeen maksujärjestelmää voidaan testata oikeilla käyttäjillä ja ostoksilla.



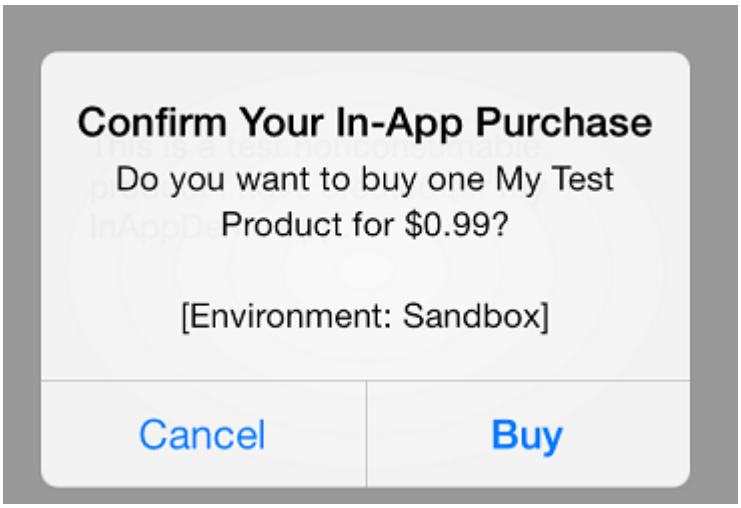
Kuva 11. Google Play -kaupan kautta tapahtunut testisostos. Ostoksessa ilmoitetaan testaajalle, ettei tuotteesta laskuteta.

Tilauksien testaaminen toimii samankaltaisesti kuin kertaostosten testaaminen, mutta tilauksissa on muutamia eroavia tilanteita, kuten onnistunut tai epäonnistunut tilauksen uusiminen. Tilauksia testatessa voidaan kuitenkin käyttää samaa menetelmää kuin kertaostosten kanssa. Testausta varten tehdyt tilaukset uusiutuvat huomattavasti normaalia nopeammin auttaakseen testaamisessa. Esimerkiksi viikon mittainen tilaus uusiutuu kerran viidessä minuutissa. Testitilaukset uusiutuvat enintään kuusi kertaa.

AppStore Connect -konsolista voidaan suorittaa iOS-alustalla olevan Apple AppStore -sovelluksen testaus käyttämällä Sandbox Tester -käyttäjätilejä, joita voi luoda sovelluksen konsolista kohdasta Users and Roles. Sanbox Tester -käyttäjälle pitää luoda uusi testaamiseen tarkoitettu tili, jolle annetaan samankaltaiset tiedot kuin oikealle olemassa olevalle Apple ID -tilille. [27.]

Jos Sandbox Tester -tilillä halutaan testata Non-Consumable-tuotteen ostamista useampaan kertaan, pitää olla luotuna aina uusi testaamista varten tarkoitettu tili. Samalla käyttäjätilillä Non-Consumable-tuotteen ostaminen johtaa vain tuotteen palauttamiseen tilille. Jos tuotteen testaamisessa pitää useita kertoja suorittaa ostoksen koodi läpi, voidaan Non-Consumable-tuottetta testata Comsumable-tuotteena. [27.]

Sandbox Tester -tilillä testattaessa on tärkeää muistaa kirjautua ensiksi pois Apple App Storesta käyttäjän aikaisemmalta, oikealta Apple ID -tililtä. Sitten ostosta voidaan testata avaamalla sovellus ja painamalla esimerkiksi ostosnäppäintä, joka suorittaa ostoksen ohjelmakoodin. Tässä vaiheessa tulee ilmoitus, jossa pyydetään kirjautumaan sisään ostoksen suorittamista varten, ja sovellukseen tulee kirjautua Sandbox Tester -tilillä. Kuvassa 12 on ilmoitus, joka näkyy Sandbox Tester -tilillä ostosta suorittavalle käyttäjälle. Ilmoituksessa kerrotaan, että kyseessä on Sandbox-testi.



Kuva 12. Apple App Storen ilmoitus testiostoksesta kun testiostos suoritetaan Sandbox Tester -käyttäjätilillä.

Nyt ostamisen ohjelmakoodia voidaan testata valitsemalla Buy-vaihtoehto. Sandbox Tester -käyttäjätilejä käytettäessä kehittäjän ei tarvitse käyttää omia maksutietojaan sovelluksen sisäisten ostosten testaamiseen eikä ostoksista laskuteta.

4.9 Lopputoimenpiteet

Jotta maksujärjestelmää voidaan käyttää tuotteiden ostamiseen sovelluksessa, sitä varten täytyy olla luotuna käyttöliittymä. Sovelluksen käyttöliittymä rakennettiin Unity-pelimoottorissa Canvas-tyyppiseen objektiin. Ostotapahtuma aloitetaan ohjelmakoodista kutsumalla BuyProductID-metodia, johon on asetettu parametriksi ostettavan tuotteen Product ID -avain. Käyttöliittymässä on tärkeää ilmoittaa tarkasti tuotteen tai tilauksen hinta ja mitä se sisältää. Applen sääntöjen mukaan se ei saa olla piilotettuna alavalikkoon, vaan sen pitää olla suoraan näkyvissä ostotapahtuman yhteydessä. Kuvassa 13 on esimerkki ostovalikosta, jossa kerrotaan hinta ja tiedot tuotteesta samassa näkymässä, jossa ostosnäppäin sijaitsee ja ostotapahtuma aloitetaan.

Kuva 13. Amazon Appsin esimerkki tilauspalvelun ostonäkymästä [28].

Ostaminen on esimerkissä kiinnitetty näppäimeen, ja näppäimen painallus avaa käyttäjän laitteen mukaisesti joko Google Play- tai Apple App Store -ostonäkymän, jossa itse maksu tapahtuu.

Kokonaisuudessaan Unity IAP:lla toteutettua sovelluksen sisäistä maksujärjestelmää, joka on toiminnassa Google Play- ja Apple App Store -kaupoissa voidaan, kuvata seuraavasti:

- Unity IAP -liitännäinen otetaan käyttöön Unity-editorista ja lisätään projektiin.
- Liitännäinen alustetaan ohjelmakoodissa ottamalla käyttöön Unity IAP:n luokka UnityPurchasing.
- Kuittien tarkistamiseen tarkoitetut salausavaimet Google Play- ja Apple App Store -kauppoihin voidaan luoda ja ottaa käyttöön alustuksessa.
- Sovelluksen sisäisessä maksujärjestelmässä myytävät tuotteet lisätään alustuksen aikana.
- Luodaan ostostapahtuma ja ProcessPurchasing-metodi, joka hoitaa kuitin tarkastamisen.
- Tehdään mahdolliseksi ostosten palauttaminen iOS-laitteille luomalla käyttöliittymään palautus näppäin.
- Luodaan käyttöliittymä, josta ostostapahtuman voi aloittaa.

Kun tuotteet on asetettu aktiivisiksi Google Play- ja Apple App Store -konsoleissa, sovelluksen sisäinen maksujärjestelmä on valmis käytettäväksi. Ostostapahtumista tallentuvat tiedot löytyvät myös maksupalveluiden konsoleiden kautta.

5 Työn tulokset

Insinöörityössä sovellukseen lisättyä sovelluksen sisäistä maksujärjestelmää voi käyttää sinne erilaisten tuotteiden ja tilausten luomiseen. Uusia tuotteita tai tilauksia tehdessä niiden aktivointi pitää hoitaa sovelluksen puolella.

5.1 Maksujärjestelmän käyttö

Uusia tuotteita tai tilauksia voidaan lisätä eri kauppapaikoille Google Play Consolen ja Apple Developer Consolen kautta. Ne pitää tämän jälkeen aktivoida käyttöön ohjelmakoodissa tai Unity-pelimoottorin Codeless IAP -näkymässä. Tuotteille pitää tämän jälkeen ohjelmakoodissa lisätä toiminnallisuus, miten asiakkaan omistama tuote vaikuttaa sovelluksen toimintaan. Insinöörityön tapauksessa kyseessä oli tilauspalvelu, joka avaa erilaisia osia sovelluksesta, mikäli asiakas omistaa aktiivisen tilauksen. Uusia tuotteita lisätessä pitää myös luoda tuotteille käyttöliittymään graafinen elementti, jota kautta mahdollinen asiakas näkee, mitä hän on ostamassa ja paljonko se maksaa.

Ostotapahtumista tarkastetaan kuitti oston yhteydessä, jotta osto on oikeasti tapahtunut, ja vältytään väärinkäytöltä. Palautettavat ostotapahtumat myös palautetaan asiakkaan laitteelle siihen yhdistettäessä. iOS-laitteilla tämän suorittaa käyttöjärjestelmään liitetty näppäin. Ostotapahtumista myös tallennetaan palvelimelle analytiikkatietoja, joilla voidaan seurata tapahtuneita ostoksia.

5.2 Kehityskohteet

Maksujärjestelmä osaa nyt suorittaa kaiken maksamiseen liittyvän koodin ja avata tuotteita asiakkaalle palautusten ja ostamisen yhteydessä sekä varmistaa turvallisuuden. Ostamiseen suunnitellussa käyttöliittymässä on vielä kehittämistä, että siitä saadaan mahdollisen selkeä ja helppokäyttöinen asiakkaalle. Kolmannen osapuolen maksupalvelun tai maksunvälittäjän käyttöönotto tilanteissa, jotka ovat sallittuja Google Play- ja Apple App Store -palveluissa julkaistuille sovelluksille ja tuotteeseen liittyvien yksittäisten

sääntöjen mukaan, voi olla toteutettavissa tulevaisuudessa, kun sovellusta ja tuotetta kehitetään eteenpäin.

Sovelluksen sisäinen maksujärjestelmä on tällä hetkellä toiminnassa, mutta ei vielä tuotantokäytössä. Maksujärjestelmän toiminta ja sen kautta kautta tehtävät ostostapahtumat sekä kuittien varmennus on testattu Android- ja iOS-laitteilla.

6 Yhteenveto

Insinöörityössä kehitettiin sovelluksen sisäinen maksujärjestelmä asiakasyrityksen Android- ja iOS -sovellukseen, jota kautta voidaan myydä tilaustyyppisiä tuotteita sovelluksen sisältä asiakkaille. Työn tuloksena saatiin rakennettua sovelluksen sisään Google Play- ja Apple App Store -kauppapaikkoja käyttävä käyttöliittymä, jolla asiakkaat pystyvät ostamaan tuotteita sovelluksesta. Käyttöliittymä toteutettiin käyttäen Unity IAP -liitännäistä Unity-pelimoottoriin.

Sovellukseen haluttiin lisätä tilaustyyppisiä tuotteita, mutta muunkinlaisia tuotteita voidaan nyt lisätä jatkossa myyntiin käyttäen samoja menetelmiä. Sovelluksen sisäinen kauppa tarkastaa asiakkaiden ostoksien kuitit ja estää tuotteiden saamisen virheellisesti haltuun. Laskutuskumppanina toimivat Google ja Apple. Maksut kulkevat turvallisesti niiden palvelimien kautta. Sovellus osaa myös muistaa asiakkaalla olleet tuotteet ja palauttaa ne sovelluksen uudelleenasennuksen jälkeen.

Insinöörityössä ei toteutettu ostoksille etävalidointia eikä käytetty kolmannen osapuolen maksupalveluita tai maksunvälittäjiä, mutta nämä ovat asioita, jotka ovat mahdollisesti edessä tulevaisuudessa, sillä maksujärjestelmää kehitetään eteenpäin sovelluksen käyttötarkoituksiin.

Lähteet

1 Apple App Store. Verkkoaineisto. Apple. <https://www.apple.com/fi/ios/app-store/>. Luettu 7.9.2019.

2 Google Play Store. Verkkoaineisto. Google. <https://play.google.com/store/>. Luettu 7.9.2019.

3 PayPal. Verkkoaineisto. Paypal. <https://www.paypal.com/fi/home>. Luettu 7.9.2019.

4 Shopify. Verkkoaineisto. Shopify. <https://www.shopify.ca/>. Luettu 7.9.2019.

5 Stripe. Verkkoaineisto. Stripe. <https://stripe.com/en-fi>. Luettu 7.9.2019.

6 Monetization and Ads, Payments. Verkkoaineisto. Google. <https://play.google.com/about/monetization-ads/payments/>. Luettu 7.9.2019.

7 App Store Review Guidelines. Verkkoaineisto. Apple. < https://developer.apple.com/app-store/review/guidelines/#introduction >. Luettu 8.9.2019.

8 Transaction Fees. Verkkoaineisto. Google. < https://support.google.com/google-play/android-developer/answer/112622?hl=en >. Luettu 11.9.2019.

9 App Store Principles and Practices. Verkkoaineisto. Apple. < https://www.apple.com/ios/app-store/principles-practices/ />. Luettu 22.9.2019.

10 Shopify Unity Buy SDK. Verkkoaineisto. Shopify. < https://help.shopify.com/en/api/storefront-api/tools/unity-buy-sdk>. Luettu 7.9.2019.

11 Ways to accept payments in Finland. Verkkoaineisto. Shopify. < https://www.shopify.com/payment-gateways/finland>. Luettu 5.9.2019.

12 Google Play Store. Verkkoaineisto. Google. <https://developer.android.com/distribute/google-play>. Luettu 7.9.2019.

13 In-App Purchase. Verkkoaineisto. Apple. < https://developer.apple.com/in-app-purchase/>. Luettu 11.9.2019.

14 Play Consolen Käyttö. Verkkoaineisto. Google. <https://support.google.com/googleplay/android-developer/answer/6112435?hl=fi>. Luettu 20.9.2019.

15 Defining products. Verkkoaineisto. Unity. <https://docs.unity3d.com/Manual/UnityIAPDefiningProducts.html />. Luettu 7.9.2019.

16 Create a managed product. Verkkoaineisto. Google. < https://support.google.com/googleplay/android-developer/answer/1153481?hl=en>. Luettu 7.9.2019.

17 Apple Store Connect Help. Verkkoaineisto. Apple. < https://help.apple.com/app-store-connect/en.lproj/static.html>. Luettu 7.9.2019.

18 Unity Editor. Verkkoaineisto. Unity. < https://unity3d.com/unity/editor>. Luettu 7.9.2019.

19 Unity IAP. Verkkoaineisto. Unity. < https://docs.unity3d.com/Manual/UnityIAP.html >. Luettu 11.9.2019.

20 Setting up Unity IAP. Verkkoaineisto. Unity. <https://docs.unity3d.com/Manual/UnityIAPSettingUp.html>. Luettu 7.9.2019

21 Codeless IAP. Verkkoaineisto. Unity. < https://docs.unity3d.com/Manual/UnityIAPCodelessIAP.html>. Luettu 7.9.2019

22 Subscription Product support. Verkkoaineisto. Unity. < https://docs.unity3d.com/Manual/UnityIAPSubscriptionProducts.html>. Luettu 7.9.2019

23 Receipt validation. Verkkoaineisto. Unity. < https://docs.unity3d.com/Manual/UnityIAPValidatingReceipts.html>. Luettu 7.9.2019

24 Processing Purchases. Verkkoaineisto. Unity. < https://docs.unity3d.com/Manual/UnityIAPProcessingPurchases.html>. Luettu 7.9.2019

25 Restoring Transactions. Verkkoaineisto. Unity. < https://docs.unity3d.com/Manual/UnityIAPRestoringTransactions.html>. Luettu 7.9.2019

26 Test Google Play Billing. Verkkoaineisto. Google. < https://developer.android.com/google/play/billing/billing_testing>. Luettu 15.9.2019

27 Sandbox Testing. Verkkoaineisto. Apple. < https://developer.apple.com/apple-pay/sandbox-testing/>. Luettu 15.9.2019

28 Amazon In-App Purchasing Component. Verkkoaineisto. Amazon. < https://developer.amazon.com/docs/fire-app-builder/amazon-in-app-purchase-component.html>. Luettu 15.9.2019