Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

JONI RÖNTYNEN

DevOps mobiilisovelluksen kehityksessä

TIETOJENKÄSITTELYN TUTKINTO-OHJELMA
2022

Tekijä Röntynen, Joni	Julkaisun laji Opinnäytetyö, AMK	Päivämäärä marraskuu 2022
	Sivumäärä 24	Julkaisun kieli Suomi
Julkaisun nimi **DevOps mobiilisovelluksen kehityksessä**		
Tutkinto-ohjelma Tietojenkäsittelyn koulutusohjelma		

Tiivistelmä

Tässä työssä tutustuttiin DevOpsiin yleisesti, sekä avattiin sen historiaa. Työssä esitellään myös mobiilikehityksen ydinpiirteet Flutter ohjelmistokehyksellä. Flutter on tällä hetkellä suosituin mobiilisovellusten ohjelmistokehys. Mobiilisovellusten lähdekoodit sijaitsevat Bitbucketissa.

Projektissa vertaillaan eri jatkuvan integroinnin ja julkaisun palveluntarjoajia, sekä niihin liitettäviä työkaluja ja ominaisuuksia. Lopuksi valitaan yksi palveluntarjoaja, jolla työ toteutetaan.

Työssä valittiin versiointiin semanttinen versiointi, sekä päätettiin julkaisuiden synkronisuudesta iOS ja Android käyttöjärjestelmille.

Avainsanat
Mobiilikehitys, DevOps, automatisointi, CI/CD

Author
Röntynen, Joni

Type of Publication
Bachelor's thesis

Date
November 2022

Number of pages
24

Language of publication:
Finnish

Title of publication
DevOps in mobile development

Degree programme
Business Information Systems

Abstract

In this work, DevOps was introduced in general, as well as its history. In the work, I also present the core features of mobile development with the Flutter software framework. Flutter is currently the most popular software framework for mobile applications. Source codes for mobile applications will be available on Bitbucket.

I compare different continuous integration and continuous delivery online providers and their tool and properties. At the end best provider is chosen for the project.

Versioning was decided to follow semantic versioning and keep deployments synchronized for iOS and Android operating systems.

Keywords
Mobile development, DevOps, automation, CI/CD

SISÄLLYS

SYMBOLI- JA LYHENNELUETTELO

AAB - Android App Bundle

AWS - Amazon Web Services pilvipalveluntarjoaja

AWS Codecommit - Amazon Web Services pilvipohjainen versionhallintajärjestelmä

AWS S3 - Amazon Web Services pilvipalveluntarjoajan tietosäiliö

Bitbucket - Atlassianin ylläpitämä pilvipohjainen versionhallintajärjestelmä

Build - Lähdekoodin kääntäminen ajettavaan versioon

Branch -Versionhallintajärjestelmän lähdekoodista(master) eriytetty haara (oksa)

Framework - Ohjelmistokehys

GIT Repository - Hajautetun versionhallintajärjestelmän tietosäiliö

GIT PUSH - Päivittää pilvipohjaisen versiohallinnan lähdekoodin

GIT PULL REQUEST - Pyyntö lähdekoodin omistajalle lisätä avustajan koodi osaksi lähdekoodia

HTTP - Selaimien sekä palvelimien siirtoprotokolla

IPA - iOS sovelluksen arkistotiedosto

JSON - Tiedostomuoto tallennukseen ja tiedonvälitykseen

Linter - Koodin tarkistustyökalu

XCode – Applen ohjelmointiympäristö

1 JOHDANTO

Mobiilisovellukset tarjoavat monia hyötyjä suhteessa web-sovelluksiin ja niiden suosio onkin ollut kasvussa (Mansoor 2022). Vuonna 2017 alan liikevaihto oli 176,6 miljardia dollaria ja vuoteen 2022 mennessä se oli kasvanut jo 428,1 miljardiin dollariin ja kasvun ennustetaan jatkuvan. Suurimmat hyödyt tulevat mobiilisovellusten tarjoamista personointiominaisuuksista, joissa käyttäjän on mahdollista helposti räätälöidä sovelluksen sisällön näyttäminen omanlaisekseen. Tärkeässä osassa on myös mobiililaitteiden notifikaatioiden hyödyntäminen halutuista tapahtumista. Mobiilisovellukset mahdollistavat myös laitteiden ominaisuuksien käytön, näitä ovat muun muassa laitteen kamera, yhteystiedot, GPS, kiihtyvyysmittarit, kompassi, puhelutiedot yms. (Nitin 2022.)

Mobiilisovellusten kehitysvaiheessa voi olla tarpeellista julkaista useita versioita sovelluksesta Applen App Storeen tai Googlen Play Storeen, jotta testaajat voivat testata sovelluksen ominaisuuksia. Perinteisesti jokaisella kauppajulkaisun tekijällä tulee olla tili kyseiseen sovelluskauppaan, sekä oikeudet julkaisuihin.

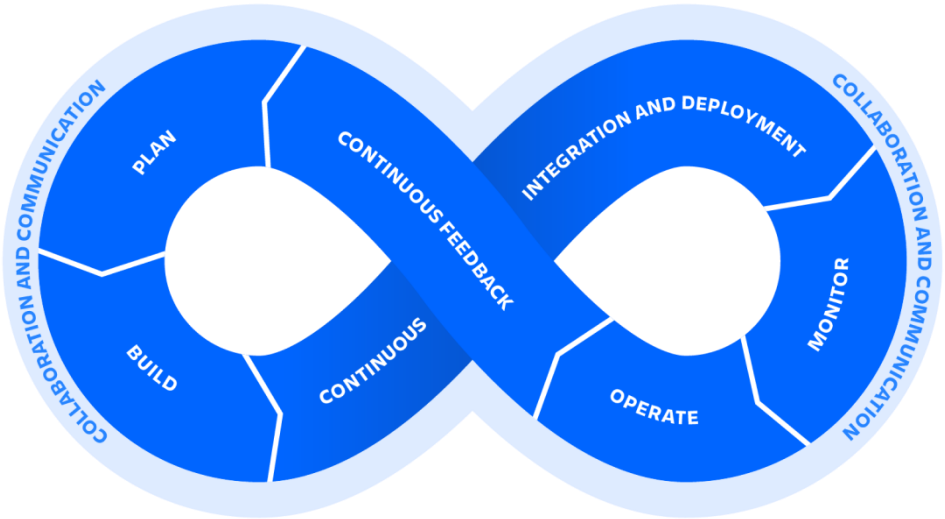
DevOps-mallissa automatisoimalla toiminnot, jokaisella kehittäjällä ei tarvitse enää olla tilejä sovelluskauppoihin. Automatisoinnilla on mahdollista myös tehostaa julkaisuihin kuluvaa aikaa ja varmistaa tuotteen julkaisujen laatu sekä julkaisunopeus. Sovellusten julkaiseminen käsityönä on aikaa vievää sekä kaavamaista toistoa, johon on ratkaisuna CI/CD-työkalujen (Continuous Integration/Continuous Deployment) käyttöönotto. Tällä hetkellä toimeksiantajalla sovellusten julkaisut tehdään itse manuaalisesti yleensä vastaavan kehittäjän toimesta. DevOpsin myötä kaikilla projektissa mukana olevilla kehittäjillä on mahdollisuus tehdä julkaisuja. Tämän opinnäytetyön tavoitteena on automatisoida kohdeyrityksen jo olemassa oleva infrastruktuuri, johon sisältyy sovellus ja sen julkaisut Googlen Playssa, sekä Applen AppStoressa. Teoriaosuudessa kerätään tietoa eri palveluntarjoajien palveluiden ominaisuuksista, valitaan vertailun perusteella parhaiten kohdeyrityksen kriteereitä vastaava palvelu ja työn

osuudessa käydään läpi. Sovelluskauppatilien perustaminen ei kuulu tämän työn piiriin.

2 DEVOPS

2.1 DevOps yleisesti

DevOps-termiä käytetään hyvin laajasti, eikä sillä ole vain yhtä oikeaa käyttötapaa. DevOps ei ole yksittäinen tuote tai palvelu vaan toimintatapa (Pilotcore 2021). Tässä työssä DevOps-termillä tarkoitetaan kehityksen sekä ylläpidon ja tuotannon yhteistyön parantamista (Contribyte www-sivut, n.d.). DevOps-liikehdintä alkoi vuosien 2007–2008 välissä ja sitä käytetään ratkaisemaan perinteisen ohjelmointimallin ongelmat, joissa ohjelmoijat sekä tuotteen julkaisusta ja ylläpidosta vastaavat tahot ovat ”siiloutuneet” omiin osa-alueisiinsa. Perinteisessä mallissa tuotteen kehittäjät (Dev) työskentelevät erikseen suhteessa tuotteen julkaisijoihin (Ops). DevOps-mallissa (kuva 1) nämä kaksi ryhmää yhdistetään yhdeksi monialaiseksi ryhmäksi, joka vastaa tuotteen julkaisuista sekä ylläpidosta koko tuotekehityksen elinkaaren ajan. (Atlassian www-sivut, n.d.)



Kuva 1. Tuotekehityksen elinkaari (Atlassian www-sivut n.d.)

DevOpsin selkeimmät hyödyt ovat suoralinjaisuus ja jatkuva korkeatasoisten päivitysten julkaiseminen. DevOps myös nopeuttaa ja helpottaa tuotteiden julkaisua. Menetelmä takaa päivitysten laadun, koska järjestelmään on mahdollista lisätä automatisoituja testejä ennen julkaisua. DevOpsin käyttöönotto vaatii sekä yritykseltä että henkilöstöltä uuden toimintamallin hyväksymistä osaksi yrityksen kulttuuria. (Hall n.d.)

2.2 Mobiilisovellusten kehitys yleisesti

Mobiilisovellusten kehityksessä tulee ottaa huomioon eri käyttöjärjestelmät ja niiden versiot. Androidilla on 12 eri versiota, joista viimeisin versio on Android 13 julkaistu vuonna 2022 (Chandan 2022). Applen iOS-versioita on 16, joista viimeisin on iOS 16. Se on julkaistu vuonna 2022 (Costello 2021.) Uudemmat versiot rajoittavat kehittäjien mahdollisuutta hyödyntää puhelimien ominaisuuksia ja lisäksi käyttäjältä on kysyttävä oikeuksia käyttää puhelimen muita sovelluksia ja toimintoja (Android www-sivut 2022).

Sovellus on mahdollista julkaista eri toimijoiden alustoilla, joista yleisimmät ovat Googlen Play Store sekä Applen App Store. Edellä mainittuihin kauppoihin tulee luoda kehittäjätilit, jotta sovelluksen julkaiseminen on mahdollista. Googlen kehittäjätilin hinta on 25 $ (Google www-sivut n.d.a) ja Applen vastaava 99 $ / vuosi (Apple www-sivut n.d.a). Ensimmäiset versiot suositellaan julkaistavaksi suljettuihin testiryhmiin kuten Googlen internal testing (Google www-sivut n.d.b) ja Applen TestFlightiin (Apple www-sivut n.d.b).

Kokemukseeni perustuva julkaisunopeus Googlella on ensimmäisen version julkaisun jälkeen muutama tunti tai joitakin päiviä, kun taas Applella puhutaan päivistä ja jopa viikoista. Pitkä julkaisuaika on haastava, mikäli sovellus tarvitsee nopeaa julkaisua esimerkiksi bugin korjaamiseksi.

Mobiilisovellusten kauppajulkaisuja ei voi peruuttaa, vaan versiointi kulkee vain eteenpäin. Mikäli vanha versio halutaan kuitenkin ottaa käyttöön, tulee sitä tehdä uusi julkaisu. (Google www-sivut n.d.c; Apple www-sivut n.d.c.) Lisäksi kehitysvaiheessa tulee huomioida sovellusten toimivuus myös sovelluksen vanhemmissa versioissa tai

pakottaa käyttäjä päivittämään sovelluksen uusin toimiva versio. Kehittäjän tulisi myös huomioida, että käyttäjä ei välttämättä lataa uusinta versiota. (Jagtap 2018.) Uudemman version päivittämisen tarve voi tulla esimerkiksi rajapinnan muuttuessa, jolloin tietojen hakeminen ei välttämättä enää toimi.

3 PROJEKTIN TYÖVÄLINEET

Flutter on Googlen kehittämä avoimen lähdekoodin framework, jonka ohjelmointikielenä on Dart. Flutterilla on mahdollista kehittää sovelluksia useille alustoille yhdellä koodikannalla.

Dart on tyyppiturvallinen, mikä tarkoittaa, että muuttujan arvo vastaa muuttujan tyyppiä aina. Käyttäjän on mahdollista tyypittää muuttuja dynaamiseksi (dynamic), jolloin on käyttäjän vastuulla varmistaa, sopiiko muuttuvan arvo muuttujaan. Mikäli dynamic-tyypin arvo ei vastaa toiseen muuttujaan asetettavaa tyyppiä, seuraa siitä ajonaikainen virhe. Dynaamista tyyppiä käytetään varsinkin JSON-tietoja haettaessa, jolloin JSON-avaimen arvon tyyppiä ei välttämättä ole mahdollista saada tietää etukäteen (String vs. List). Dartilla on mahdollista ottaa käyttöön Sound null safety, jolloin framework tarkistaa, ettei arvo voi olla null, ellei käyttäjä sitä erikseen salli. (Dart www-sivut n.d.)

Flutterin keskeinen luokkarakenne on widget-luokat, muilla frameworkeilla puhutaan komponenteista. Widgetit vastaavat käyttöliittymän piirtämisestä käyttäjälle. Rakenne on usein puumainen, jolloin yksittäisellä widgetillä voi olla useita widgettejä lapsena (child)/sisaruksena (sibling). (Flutter www-sivut 2022.a.)

Flutterilla on mahdollista ajaa automaattisia testejä, joita ovat yksikkötesti, Widgettesti ja Integraatiotesti.

Yksikkötestillä (unit test) testataan yksittäistä funktion, metodin tai luokan toimintoa, sekä toiminnon tuloksen oikeellisuutta. Ulkoiset riippuvuudet (rajapintojen palauttama data) korvataan mock-datalla, joka on itse luotua tai generoitua tietoa.

Widget-testi (widget test) testaa yksittäisen widgetin käyttöliittymän näkymää sekä toimivuutta. Testillä varmistetaan yksittäisen widgetin elinkaaren oikea kesto.

Integraatiotestillä (integration test) testataan joko koko järjestelmän toimivuutta tai isoa osaa sovelluksesta. Tavoitteena on testata kaikkien testiin sisältyvien widgettien toimivuutta keskenään, sekä mahdollistaa suorituskyvyn testaaminen. Yleensä integraatiotestit ajetaan fyysisillä laitteilla tai emulaattoreilla. Testissä oleva sovellus erotetaan koodikannasta, jotta koodikanta ei vaikuta testin tulokseen. (Flutter www-sivut n.d.b.) Flutterilla on oma linter, joka voidaan suorittaa ennen buildia. Lintteri varmistaa, ettei koodista löydy ongelmia tai esimerkiksi käyttämättömiä muuttujia, sekä antaa mahdollisia korjausehdotuksia. (Maureen 2019.)

Projektin lähdekoodit sijaitsevat Atlassianin ylläpitämässä *Bitbucke*t pilvipalvelussa, joka pohjautuu Git-versionhallintajärjestelmään. Vastaavanlaisia palveluita tarjoavat muun muassa Github ja Gitlab, sekä AWS CodeCommit.

4 CI/CD-PALVELUNTARJOAJAT

4.1 Fastlane

Fastlane on avoimen lähdekoodin työkalu julkaisuiden automatisointiin. Fastlane on lokaalisti käytettävä työkalu, jonka voi integroida muihin palveluihin. (Fastlane www-sivut n.d.) Käyttöönotossa luodaan tiedosto Fastfile jossa määritellään Ruby -ohjelmointikielellä halutut toiminnot (Kuva 2). Jotkin palveluntarjoajat vaativat käytettäväksi Fastlanea, mutta tässä projektissa sitä ei käytetä.

```
lane :beta do
  increment_build_number
  build_app
  upload_to_testflight
end

lane :release do
  capture_screenshots
  build_app
  upload_to_app_store       # Upload the screenshots and the binary to iTunes
  slack                     # Let your team-mates know the new version is live
end
```

Kuva 2. Fastfilen sisältämät toiminnot (Fastlane www-sivut n.d.)

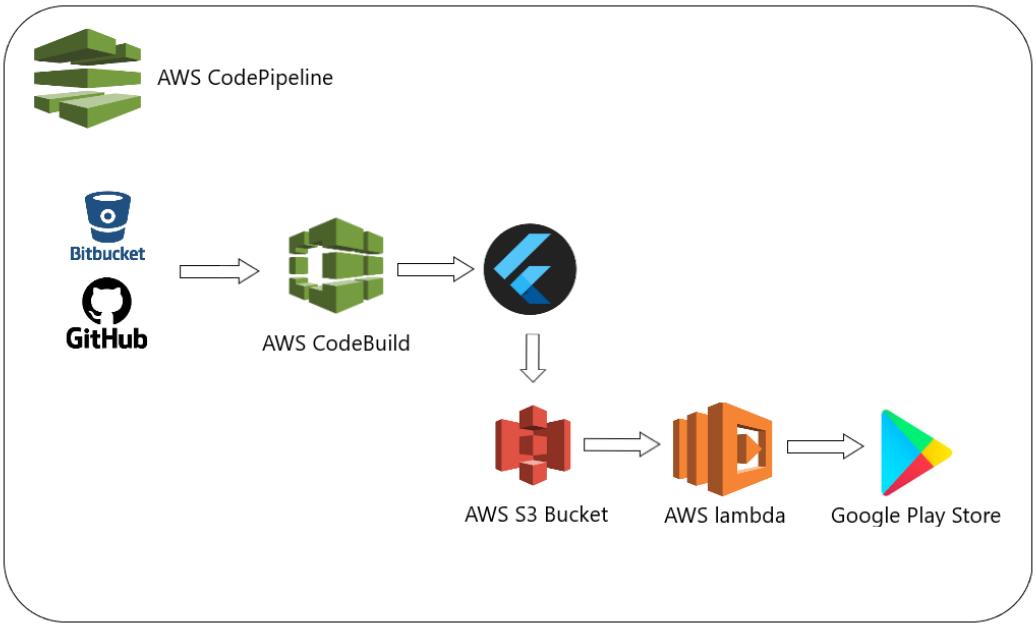
Haluttu toiminto suoritetaan kuvan 3 mukaisesti. Mikäli yrityksellä on jo käytössä Fastlane on se mahdollista integroida osaksi joidenkin palveluntarjoajien julkaisuputkea. Itsessään Fastlane ei tarjoa tämän työn määritelmän mukaista palvelua, joten se yksistään ei riitä.

```
fastlane release
```

Kuva 3. Fastlane käynnistyskomento (Fastlane www-sivut n.d.)

4.2 Amazonin verkkopalvelut

Amazon Web Services -ominaisuuksilla (AWS) on mahdollista tehdä CI/CD-julkaisuputki (Kuva 4). Julkaisuputkeen on mahdollista liittää GitHub/BitBucket repository. Tämän lisäksi tarvitaan AWS S3, johon ladataan Google Play -kaupan vaatima keystore.properties -tiedosto. Se sisältää julkaisemisen vahvistamiseen vaaditut avaimet. Kun julkaisuputkeen liitettyyn repositoryyn viitattuun branchiin tehdään muutos, laukaisee tämä julkaisuputken, jossa tapahtuu sovelluksen kääntäminen(build) sekä tämän jälkeen käännetty sovellus voidaan tallentaa haluttuun paikkaan, joka tässä tapauksessa on AWS S3.



Kuva 4. Amazon verkkopalvelulla toteutettu sovelluksen julkaisukaavio

Android App Bundle(.aab) on mahdollista lisätä Googlen Play -kauppaan muun muassa asettamalla triggeri aikaisemmin tehtyyn S3 valittuun buckettiin ja triggeri laukaisee AWS-lambdan, jolla on mahdollista siirtää tiedosto Play-kauppaan Googlen API:a hyödyntäen (Humayun 2021.). Tosin ongelmaksi tässä muodostuu iOS, joka on käännettävä XCodella/macOS, johon AWS:llä ei ole imagea/serveriä (Amazon www-sivut n.d.).

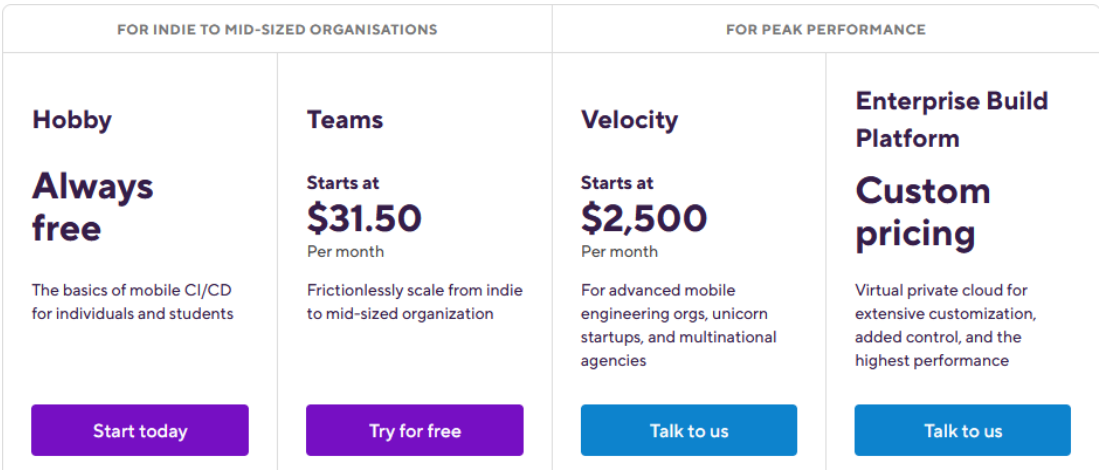
Tavoitteena on automatisoida sovelluksen testaus ja julkaiseminen, joka on mahdollista tehdä itse muun muassa AWS-ominaisuuksilla sekä vuokratulla iOS-serverillä, mutta koska pyrkimyksenä on mahdollisimman vaivaton ja helppo ratkaisu, tämä ratkaisumalli ei ole tavoitteiden mukainen, sillä ympäristön pystytys ja ylläpito on liian työlästä.

4.3 Bitrise

Bitrise on perustettu vuonna 2014 (Bitrise n.d.a) ja yritys on erikoistunut mobiilisovellusten CI/CD-palvelun tuottamiseen (Bitrise n.d.b). Bitrise tukee muun muassa seuraavia kääntäjiä: Flutter ja React Native. Bitrise tukee notifikaatioita yleisimpiin

palveluihin, esim. Comment on GitHub Pull Request, Discord message, Email with Mailgun, Microsoft Teams, Slack ja jopa Twitter message on mahdollinen.

Yrityksellä on neljä eri palveluluokkaa tarjolla ja jokainen luokka sisältää eri ominaisuuksia (kuva 3). Käyttö perustuu krediitteihin, joita tulee jokaisen kuukausittain tehtävän tilauksen mukainen määrä, sekä tarvittaessa krediittejä on mahdollista ostaa lisää. Jokainen kääntämiseen kuluva minuutti kuluttaa krediittivarastoa. Kääntäminen macOS-virtuaalikoneella on tuplasti kalliimpaa kuin Linuxin virtuaalikoneella.



Kuva 3. Bitrise-palveluiden hintataulukko (Bitrise www-sivut n.d.c.)

Halvin on harrastuskäyttöön tarkoitettu Hobby, jolla on mahdollista tehdä yksinkertaisia toimenpiteitä, kuten kääntää sovellus ja julkaista se kauppoihin. Hobby tarjoaa 300 krediittiä/kuukausi. Yrityskäyttöön edullisin sopiva luokka on Teams. Alimpaan Tier-1 tasoon sisältyy 500 krediittiä 35 $ per kuukausi.

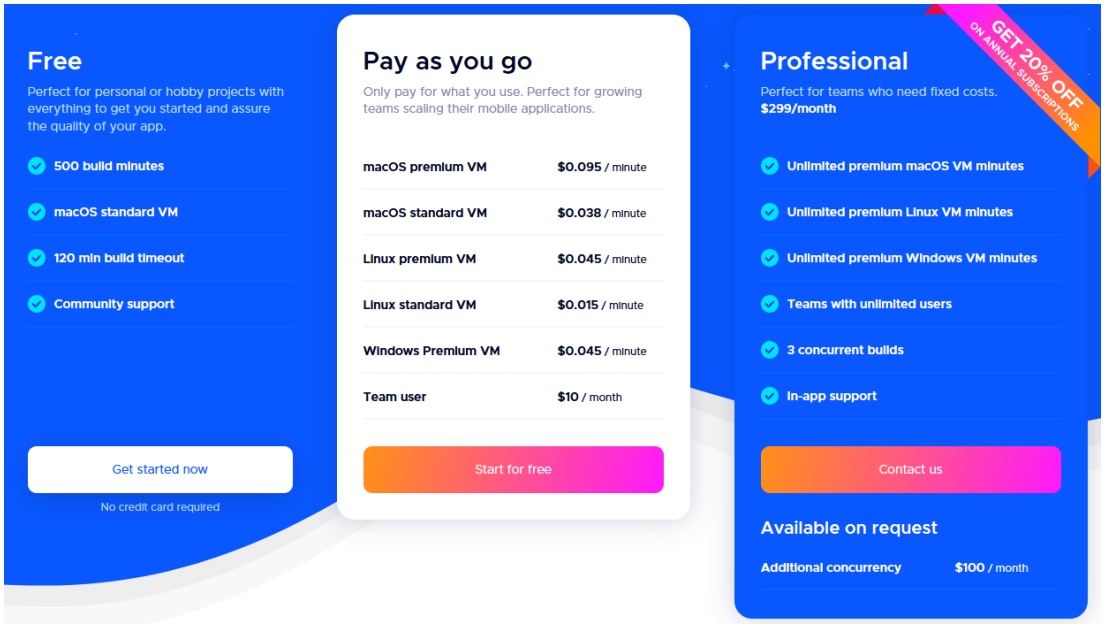
Velocity on suunnattu kansainvälisille toimijoille ja tarjonta on huomattavasti laajempi verrattuna edullisimpiin vaihtoehtoihin. Kuukausiveloitus tällä palvelulla on alkaen 2500 $/kuukausi. Lisäksi on mahdollista räätälöidä Custom-luokka ottamalla yhteyttä Bitrisen asiakaspalveluun.

4.4 Codemagic

Nevercode on vuonna 2012 Virossa perustettu yritys. Vuonna 2017 yritys julkaisi Nevercode CI/CD-palvelun mobiilisovelluksille. Tuki Flutterille tuli vuonna 2018 ja

yhteistyö Googlen kanssa alkoi 2019. Vuoden 2021 alussa Nevercode yhdistettiin Codemagiciin. Codemagic tukee muun muassa seuraavia frameworkeja: Android (Java & Kotlin), Flutter, iOS (Objective-C & Swift) ja React Native. (Codemagic www-sivut n.d.a.)

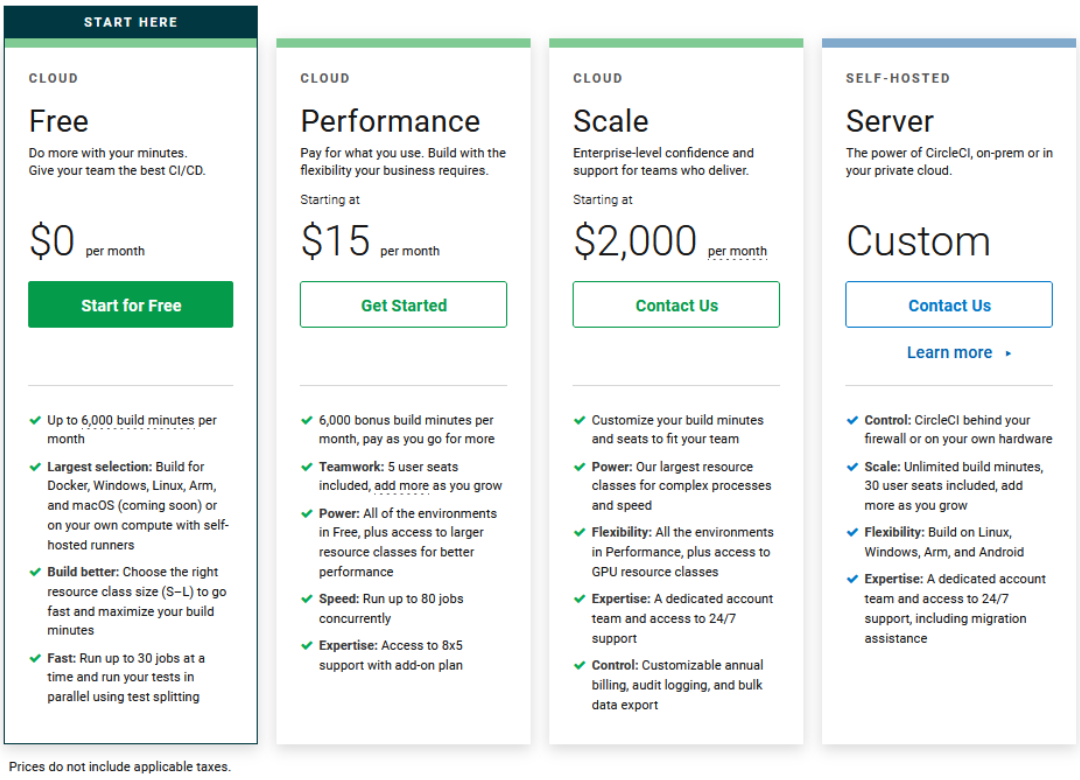
Yritys tarjoaa kolmea eri palveluluokkaa, joista edullisin on Free (kuva 4). Free-palveluluokassa tulee mukana 500 minuuttia build-aikaa. Pay as you go on nimensä mukaisesti palvelu, jota käyttämällä maksat vain kääntämiseen kuluvasta ajasta minuuttiperusteisesti. Tämän lisäksi tulee maksaa 10 $ kuukaudessa käyttäjää kohden. Käyttäjäksi lasketaan henkilö, joka käynnistää kääntämisen.



Kuva 4. Codemagic-palveluiden hintataulukko (Codemagic www-sivut n.d.b.)

4.5 CircleCI

CircleCi on vuonna 2011 perustettu yritys, jonka pääkonttori sijaitsee San Franciscossa (CircleCi www-sivut n.d.a). CircleCI ei suoraan tue Flutteria, vaan vaatii myös käytettäväksi Fastlanea. CircleCI tarjoaa neljää eri palvelualuokaa (kuva 5). Hinnoittelussa halvimmalla vaihtoehdolla on 15 $ kiinteä kuukausikulu, johon sisältyy 6000 minuuttia kääntämiseen tarvittua aikaa. Mikäli tulee tarvetta ostaa lisäaikaa, se on mahdollista. (CircleCi www-sivut n.d.a.)



Kuva 5. CircleCI-palveluiden hintataulukko (CircleCI www-sivut n.d.b)

4.6 Palveluntarjoajan valinta

Palveluntarjoajaksi valikoitui Codemagic. Valinta perustuu vahvaan Flutter-tukeen sekä hyvään dokumentaatioon. Kilpailijoihin nähden kustannuksissa pystytään säästämään sillä uusissa projekteissa aktiivisen kehitysvaiheen jälkeen palvelua ei tarvita niin usein, mutta ylläpitovaiheessa palvelusta ei kuitenkaan haluta luopua kokonaan ja muilla kustannuksia kertyy silti vaikka palvelua ei käytettäisi useaan kuukauteen.

Codemagicissa on mahdollista luoda tiimejä, jossa käyttäjille voidaan antaa eri oikeuksia ja rajoituksia. Tiimit luodaan projektikohtaisesti ja tiimiin voidaan määritellä halutut henkilöt, joilla on käyttöoikeus käynnistää julkaisuputki. Aikaisemmin julkaisujen tekeminen vaati kehittäjältä tilejä sovelluskauppoihin, sekä sovelluksen julkaisemiseen vaaditut allekirjoitusavaimet. Palveluntarjoajan mallissa jokaisella kehittäjällä ei tarvitse olla tietämystä sovelluskauppojen toiminnoista tai määrittelyistä.
Laskutus perustuu sovelluksen kääntämiseen kuluvaan aikaan, sekä aktiivisten käyttäjien määrään kuukaudessa. Kuluja voidaan hallita myös rajoittamalla buildaamiseen kuluvaa aikaa tai buildien määrää.

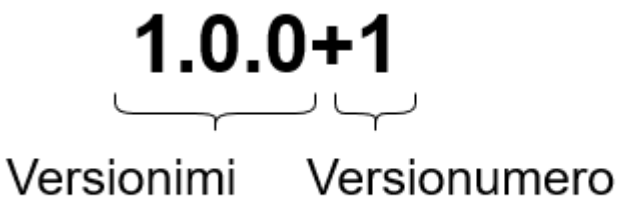
5 TOTEUTUS

Tässä projektissa noudatetaan semanttista versiointia (Sematic Versioning www-sivut n.d). Sovelluksen versioinnissa on hyvä noudattaa yleisiä ohjeita (Bruno 2022). Versionimi on myös yleisesti käyttäjien näkyvissä. Versionimessä on kolme eri numeroa pisteellä eroteltuna (Kuva 6). Version nostossa nostetaan haluttua ”tasoa” ja tasot määritellään seuraavasti:

- Major versio sisältää rikkovia muutoksia esimerkiksi API:in
- Minor versio lisää toimintoja ja ominaisuuksia, jotka eivät riko API:a
- Patch versio, mikäli tehdään bugikorjauksia ilman uusia ominaisuuksia

Kuva 6. Semanttinen versionti (Narek 2020)

Versioinnissa on kaksi osaa, toinen on käyttäjille näkyvä versionnimi, josta käytetään puhekielessä termiä ”versio” ja toinen on versionumero, jolla tarkoitetaan buildien määrää (Kuva 7).



Kuva 7. Semanttinen versio, jonka perään on lisätty versionumero

Versioinnit ovat järkevää pitää synkronoituna sovelluskaupoissa ja julkaista sama versio molemmille käyttöjärjestelmille samanaikaisesti, jotta mahdollisten ongelmien seuraaminen on helpompaa. Versiota ylläpidetään Flutterin pubspec-tiedostossa, joskin versionumeron ylläpitäminen jätetään Julkaisuputken hallintaan.

Versionimi on vain käyttäjää varten, eikä sillä ole varsinaista toiminnallista käyttötarkoitusta (Android www-sivut n.d). Käytännössä Androidilla ja iOSilla on samat versionimi ominaisuudet, mutta eri nimillä. Androidilla käytetään termiä “versionName” ja vastaavasti iOS on vain ”version” (Apple www-sivut n.d.d). Uuden versionumeron nostamisen määrittely jää käyttäjän vastuulle, koska järjestelmä ei tiedä ohjelmakoodiin tehtyjen muutosten vaikutusta.

Versionumero on järjestelmiä varten, eikä käyttäjät usein sitä näe (Android www-sivut n.d). Numero esitetään usein sulkumerkkien sisällä kokonaislukuna, esimerkiksi 1.0.0(3) tai 1.0.0+3, joka tässä ilmoittaa, että kyseessä on kolmas versio kyseisestä sovelluksesta. Kääntämisen yhteydessä tulee myös huomioida versionumero, joka androidille kääntäessä tulee nostaa yhdellä jokaista kääntämistä kohden. Luku on juokseva ja sitä nostetaan jokaisella kääntämisellä yhdellä. Apple sallii versionumeron resetoinnin jokaisen version muutoksen välissä, joskin tämä saattaa aiheuttaa hämmennystä (Chris 2020). Tästä syystä on mielekästä noudattaa versionumeron nostamista samoin kuin Googlen määrittelemällä tavalla. Joillakin palveluntarjoajilla on mahdollisuus käyttää kauppojen rajapintaa, josta saa haettua sovelluksen viimeisimmän julkaistun versiokoodin, joka mahdollistaa aina uusimman version hakemisen. Ilman automaattista versionumeron hallintaa tulee käyttäjän huolehtia siitä, että versiokoodi on aina ajan tasalla pubspec-tiedostossa.

5.1 Codemagic käyttöönotto

Codemagicin käyttöönotossa on mahdollista käyttää joko ilmaista palvelua tai maksullisia palveluita. Käyttäjän tulee rekistöröidä itsensä sekä valita mahdollisesti haluttava maksullinen osio. Mikäli käyttäjä haluaa luoda tiimejä, on valittava maksullinen osio ja annettava maksutiedot sekä hyväksyttävä ehdot. Projekti tulee liittää osaksi Codemagicia ja tässä projektissa käytetään Atlassianin BitBucketissa olevaa

repositorya, jossa lähdekoodit sijaitsevat. Ne kehittäjät, joille halutaan antaa oikeus käynnistää julkaisuputki tulee liittää osaksi projektin tiimiä.

Toteutuksena voi käyttää joko graafista käyttöliittymää tai YAML-tiedostoa. Mikäli Flutter sovelluksen tiedoston juuresta löytyy codemagic.yaml-tiedosto, käyttää Codemagic kyseistä tiedostoa konfiguraatioon automaattisesti. (Codemagic www-sivut 2022.)

5.2 Workflow

Yksittäinen workflow tekee yhden asian esimerkiksi Androidin kääntämisen ja julkaisun Play-kauppaan (Kuva 8). Workflowlle määritellään kuvaava nimi esimerkiksi android-release, joka tässä tapauksessa tekisi Play-kaupan tuotantojulkaisun. Workfloweja voi olla useita yhdessä tiedostossa, riippuen haluttujen toimintojen määrästä esimerkiksi:

- android_internal
- android_release
- ios_testflight
- ios_release
- unit_tests

Kääntämiselle on mahdollista asettaa aikarajat, jolla varmistetaan, ettei kääntämiseen kulu kohtuuttomasti aikaa.

```yaml
workflows:
  android-workflow:
    name: Android Workflow
    max_build_duration: 120
    environment:
      android_signing:
        - keystore_reference
      groups:
        - google_play
      vars:
        PACKAGE_NAME: "io.codemagic.fluttersample"
        GOOGLE_PLAY_TRACK: alpha
      flutter: stable
    scripts:
      - name: Set up local.properties
        script: |
          echo "flutter.sdk=$HOME/programs/flutter" > "$CM_BUILD_DIR/android/local.properties"
      - name: Get Flutter packages
        script: |
          flutter packages pub get
      - name: Flutter analyze
        script: |
          flutter analyze
      - name: Flutter unit tests
        script: |
          flutter test
        ignore_failure: true
      - name: Build AAB with Flutter
        script: |
          BUILD_NUMBER=$(($(google-play get-latest-build-number --package-name "$PACKAGE_NAME" --tracks="$GOOGLE_PLAY_TRACK") + 1))
          flutter build appbundle --release \
            --build-name=1.0.$BUILD_NUMBER \
            --build-number=$BUILD_NUMBER
    artifacts:
      - build/**/outputs/**/*.aab
      - build/**/outputs/**/mapping.txt
      - flutter_drive.log
    publishing:
      email:
        recipients:
          - user_1@example.com
          - user_2@example.com
        notify:
          success: true
          failure: false
      google_play:
        credentials: $GCLOUD_SERVICE_ACCOUNT_CREDENTIALS
        track: $GOOGLE_PLAY_TRACK
        submit_as_draft: true
```

Kuva 8. Esimerkki worklowsta, jossa julkaistaan sovellus kauppaan testaajille (Codemagic www-sivut 2022)

5.3 Ympäristömuuttujat

Codemagiciin määriteltyjä ympäristömuuttujia on järkevää käyttää sellaisten tietojen käyttöön, joita ei haluta jakaa versionhallintaan. Codemagic tarjoaa joko sovelluskohtaisia tai globaaleja ympäristömuuttujia. Globaalit muuttujat soveltuvat, mikäli on tarve useamman projektin jakaa sama muuttuja. Sovelluskohtaisia ympäristömuuttujia on kahdenlaisia, yksittäisiä ja ryhmiä. Ryhmämuuttujat määritellään graafisen käyttöliittymän avulla ja salaisuudet salataan valitsemalla Secure checkbox. Binääritiedostot tulee base64 enkoodata ennen kuin ne voidaan tallentaa ympäristömuuttujaksi.

Myöhemmin käyttöönoton yhteydessä tiedostot base64 dekoodataan. Ryhmät ryhmitellään toimintojen mukaan (Kuva 9).

```
environment:
  groups:
    - keystore_credentials
    - google_play
    - appstore_credentials
```

Kuva 9. Ryhmämuuttujien esittely

Yksittäiset muuttujat määritellään kuvan 10 mukaisesti. Ympäristömuuttujiin on mahdollista tallentaa haluttuja tietoja käyttäjän toimesta ja lisäksi Codemagic tarjoaa omien sisäisten ympäristömuuttujien toimintoja muun muassa versionumeron ylläpitämistä sekä tarvittavien tiedostojen staattisia polkuja ja lisäksi useita muita toimintoja.

```
vars:
  PACKAGE_NAME: 'com.joni.thesis'
  XCODE_WORKSPACE: 'Runner.xcworkspace'
  XCODE_SCHEME: 'Runner'
  BUNDLE_ID: 'com.joni.thesis'
  APP_STORE_ID: 1617500000
```

Kuva 10. Yksittäisten ympäristömuuttujien määrittely tiedostossa

5.4 Triggeri

Julkaisuputki halutaan käynnistää jonkin tapahtuman takia, joka tässä tapauksessa on muutos lähdekoodissa, joka sijaitsee versionhallinnan pilvipalvelussa. Tämän projektin lähdekoodit sijaitsevat Bitbucketissa. Lähdekoodit voivat sijaita muissakin versionhallintaa tarjoavissa pilvipalveluissa muun muassa GitHubissa tai GitLabissa. Bitbucketissa tapahtuvia muutoksia ”kuunnellaan” webhookin avulla. Mikäli palvelussa tapahtuu muutos, esimerkiksi muutos lähdekoodissa, lähettää kyseinen pilvipalvelu viestin haluttuun kohdeosoitteeseen, jotta kohdeosoite tietää tapahtuneesta muutoksesta. Viesti kulkee HTTP POST payloadissa, joka sisältää tarvittavat tiedot tapahtumasta.

Konfiguraatiossa määritellään mitä repositoryn branchia halutaan ”kuunnella” ja mikäli kuunnellussa branchissa tapahtuu haluttu tapahtuma, joko push, pull request tai

tag, ilmoittaa se Codemagicille muutoksesta webhookilla. Build on mahdollista myös käynnistää graafisen käyttöliittymän avulla.

```
triggering:
  events:                     # List the events that trigger builds
    - push
    - pull_request
    - tag
  branch_patterns:            # Include or exclude watched branches
    - pattern: '*'
      include: true
      source: true
    - pattern: excluded-target
      include: false
      source: false
    - pattern: included-source
      include: true
      source: true
  tag_patterns:               # Include or exlude watched tag labels
    - pattern: '*'
      include: true
    - pattern: excluded-tag
      include: false
    - pattern: included-tag
      include: true
  cancel_previous_builds: false  # Set to `true` to automatically cancel outdated webhook builds
```

Kuva 11. Triggereiden määrittelyt (Codemagic www-sivut 2022)

5.5 Scriptit

Scripts -osiossa suoritetaan varsinaiset toiminnot, joita käyttäjä ajaa normaalisti paikallisesti terminaalilla. Toimintoihin sisältyy tarvittavien tiedostojen luominen, sekä sovelluksen ulkopuolisten pakettien hakeminen ja mahdollisten testien suorittaminen. Tämän lisäksi suoritetaan itse sovelluksen buildaaminen, joka tuottaa sovelluskauppaan tarvittavat artifaktit. Sovelluskaupat tarjoavat rajapinnan, jolla on mahdollista hakea viimeisein käytetty versionumero ja nostamalla sitä yhdellä, jolloin vältetään mahdolliset päällekkäiset versionumerot ja siitä seuraava virhetilanne (Kuva 12).

```
--build-number=$(($(google-play get-latest-build-number --package-name 'com.example.app') + 1))
```

Kuva 12. Viimeisimmän versionumeron hakeminen sovelluskaupasta (Codemagic www-sivut 2022)

5.6 Julkaisut

Julkaisut ovat mielekästä jakaa kuluttajille sovelluskauppojen avulla. Android julkaisuiden yleisin alusta on Googlen Play (kuva 13). Kiinan markkinoita ei ole huomioitu tässä kuvassa, sillä Google Play ei ole sallittu Kiinassa. (Artyom 2022.)

Name	Available Apps
Google Play	2.7 million
Amazon AppStore	0.48 million
GetJar	0.85 million
Aptoide	0.7 million
Opera Mobile Store	0.3 million

Kuva 13. Yleisimmät sovelluskaupat Androidille (Artyom 2022)

Kuvassa 14 esitellään yleisimmät sovelluskaupat iOS sovelluksille. Julkaisut kauppoihin tehdään joko manuaalisesti käsin Googlen Play Consolen kautta tai Applen XCodella. Vaihtoehtoisesti on mahdollista käyttää sovelluskauppojen rajapintoja ja luoda julkaisut näiden avulla. Automatisoinnin palveluntarjoajat käyttävät rajapintoja hyödyksi julkaisuiden tekemiseen ja tästä syystä palveluntarjoajilla on tarve saada pääsy kauppoihin.

Name	Available Apps
App Store	2.2m
Cydia	n/a
GetJar	0.85m
Appland	0.13m

Kuva 14. Yleisimmät jakelualustat iOS sovelluksille (Artyom 2022)

Sovelluskauppojen rajapintojen käyttö vaatii oikeuksien antamisen ja haasteena tässä on, että oikeuden voi myöntää vain tilin omistaja. Tilin omistajan tietoteknisestä osaamisesta riippuen, voi olla tarve opastaa oikeuksien antamisessa, sillä varsinkin Google Playssa se on hieman monimutkaisempi. Googlella luodaan palveluun käyttäjä, jonka avulla on mahdollista ladata sovellustiedosto (.aab) Play-kauppaan. Applella käytetään rajapinnan avainta ja muita tarvittavia tietoja, jolla mahdollistetaan rajapinnan käyttö. Toki olisi mahdollista itsekin käyttää sovelluskauppojen rajapintoja julkaisuihin dokumentaatioiden mukaisesti, mutta tässä tapauksessa se ei ole mielekästä, koska tavoitteena on yksinkertaistaa julkaisuiden tekemistä ilman ylläpidollisia toimia.

Julkaisuihin on hyvä liittää informaatio mitä muutoksia uusi julkaisu sisältää. Codemagic tarjoaa mahdollisuuden lisätä muutokset pipelinen yhteydessä. Käytännössä sovelluksen tiedoston juureen lisätään release_notes.json -tiedosto (Kuva 15), johon syötetään halutut kentät.

```
[
  {
      "language": "en-GB",
      "text": "British English release notes "
  },
  {
      "language": "fi",
      "text": "Suomenkieliset julkaisutiedot"
  }
]
```

Kuva 15. Julkaisun tiedot

Julkaisujen onnistumisesta tai epäonnistumisesta on hyvä informoida kehittäjiä. Ilmoituskanavia on useita ja kyseisessä projektissa käytetään Slack-viestintäalustaa ja mahdollisesti yksittäisen kehittäjän sähköpostiosoitetta. Viestit voidaan myös jakaa siten että epäonnistunut julkaisu ilmoitetaan kehittäjän sähköpostiin ja onnistunut julkaisu Slack pikaviestintäsovelluksen projektin kanavalle.

6 POHDINTA

Opinnäytetyö on mielenkiintoinen, sillä olen tehnyt ja ylläpitänyt Flutterilla tehtyjä mobiilisovelluksia. Sovelluksien julkaisutahti varsinkin kehityksen alkuvaiheessa on nopea. Uusia versioita julkaistaan kehityksen alussa viikoittain ja tähän kuluu aikaa, sillä sovellus julkaistaan sekä Applen että Googlen kauppoihin, mikä edellyttää sovelluksen kääntämisen iOS-käyttöjärjestelmällä ja Windows/Linux-käyttöjärjestelmällä. Lisäksi julkaisuista yleensä vastaa vastaava kehittäjä, jolla on muutenkin usea projekti samanaikaisesti kesken. DevOpsilla on mahdollista jakaa julkaisuiden tekemistä usealle kehittäjälle, jolloin työtaakka vastaavalla kehittäjällä vähenee, sekä julkaisunopeus kasvaa.

Codemagic sopi tähän projektiin, mutta vastaavia palveluntarjoajia on useita ja onkin järkevää valita sopiva palveluntarjoaja sen hetkiseen projektiin. Projektin luonteesta sekä laajuudesta riippuen onkin mielekästä selvittää tarkasti mitä palveluntarjoajalta halutaan, sekä mahdollisesti räätälöidä tarpeisiin sopiva paketti palveluntarjoajalta. Kohdeyritys jolle työ toteutettiin, hyötyi toistuvan työn vähenemisestä, jolloin työn kannattavuus lisääntyi.

LÄHTEET

Amazon www-sivut n.d. Build and test iOS apps with AWS CodeCommit, AWS CodePipeline, and AWS Device Farm. Viitattu 1.2.2022. https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/build-and-test-ios-apps-with-aws-code-commit-aws-codepipeline-and-aws-device-farm.html

Android www-sivut 2022. Behavior changes: all apps. Viitattu 1.2.2022. https://developer.android.com/about/versions/12/behavior-changes-all

Android www-sivut n.d. <manifest>. Viitattu 20.11.2022. https://developer.android.com/guide/topics/manifest/manifest-element.html#vcode

Apple www-sivut n.d.a. Enrollment. Viitattu 1.2.2022. https://developer.apple.com/support/enrollment/

Apple www-sivut n.d.b. Beta Testing Made Simple with TestFlight. Viitattu 1.2.2022. https://developer.apple.com/testflight

Apple www-sivut n.d.c. Create a new version. Viitattu 1.2.2022. https://help.apple.com/app-store-connect/en.lproj/static.html

Apple www-sivut n.d.d. Version Numbers and Build Numbers. Viitattu 20.11.2022. https://developer.apple.com/library/archive/technotes/tn2420/index.html

Artyom, D. 2022. App Stores List. Viitattu 20.11.2022. https://www.businessofapps.com/guide/app-stores-list/

Atlassian www-sivut n.d. DevOps: Breaking the development-operations barrier. Viitattu 1.2.2022. https://www.atlassian.com/devops

Chandan, P. 2022. Android 13 Update Tracker. Viitattu 1.2.2022. https://www.dealntech.com/android-13-update/

Bitrise www-sivut n.d.a. Our story. Viitattu 1.2.2022. https://www.bitrise.io/about

Bitrise www-sivut n.d.b. Why do mobile app companies need to adopt Mobile DevOps? Viitattu 1.2.2022. https://www.bitrise.io/why/features/mobile-devops

Bitrise www-sivut n.d.c. Powering the world’s best mobile engineering teams. Viitattu 1.2.2022. https://www.bitrise.io/pricing

Bruno, R. 2022. How - and why – to think about mobile app versioning. Viitattu 20.11.2022. https://www.runway.team/blog/how-why-mobile-app-versioning

Chris, G. 2020 Versioning for iOS and Android Apps -Part 1. Viitattu 20.11.2022. https://www.pullrequest.com/blog/semantic-versioning-ios-android-apps/

CircleCI www-sivut n.d.a. About CircleCI. Viitattu 1.2.2022. https://circleci.com/careers/

CircleCI www-sivut n.d.b. Build with the best. Get the most, for free. Viitattu 1.2.2022. https://circleci.com/pricing/

Codemagic www-sivut n.d.a. Viitattu 1.2.2022. https://codemagic.io/start/

Codemagic www-sivut n.d.b. Viitattu 1.2.2022. https://codemagic.io/pricing/

Codemagic www-sivut 2022. Using Codemagic.yaml. Viitattu 20.11.2022. https://docs.codemagic.io/yaml-basic-configuration/yaml-getting-started/

Codemagic www-sivut 2022. Flutter apps. Viitattu 20.11.2022. https://docs.codemagic.io/yaml-quick-start/building-a-flutter-app/

Codemagic www-sivut 2022. Starting builds automatically with codemagic.yaml. Viitattu 20.11.2022. https://docs.codemagic.io/yaml-running-builds/starting-builds-automatically

Codemagic www-sivut 2022. Starting builds automatically with codemagic.yaml. Viitattu 20.11.2022. https://docs.codemagic.io/knowledge-codemagic/build-versioning/

Costello, S. 2022. The History of iOS, from Version 1.0 to 16.0. Viitattu 24.11.2022. https://www.lifewire.com/ios-versions-4147730

Dart www-sivut n.d. Dart overview. Viitattu 1.2.2022. https://dart.dev/overview

Fastlane www-sivut n.d. Viitattu 1.2.2022. https://docs.fastlane.tools/

Flutter www-sivut n.d.a. Widget class. Viitattu 1.2.2022. https://api.flutter.dev/flutter/widgets/Widget-class.html

Flutter www-sivut n.d.b. Testing Flutter apps. Viitattu 1.2.2022. https://docs.flutter.dev/testing

Google www-sivut n.d.a. Viitattu 1.2.2022. https://support.google.com/google-play/android-developer/answer/9845334?hl=en

Google www-sivut n.d.b. Difference between an internal, closed, and open test? Viitattu 1.2.2022. https://support.google.com/googleplay/android-developer/answer/6112435?hl=en&ref_topic=3450769#zippy=%2Cstep-pay-registration-fee

Google www-sivut n.d.c. Prepare and roll out a release. Viitattu 1.2.2022. https://support.google.com/googleplay/android-developer/answer/9859348

Hall, T. n.d. What is DevOps culture. Viitattu 1.2.2022. https://www.atlassian.com/

Humayun, Z. 2021. How to Automatically Upload an Android App Bundle to the Play Store. Viitattu 1.2.2022. https://www.freecodecamp.org/news/automatically-upload-an-android-app-bundle-to-the-play-store/

Jagtap, S. 2018. Why You Need to Keep Your Mobile DevOps In-House in 2018. Viitattu 20.11.2022. https://medium.com/xcblog/why-you-need-to-keep-your-mobile-devops-in-house-in-2018-f6bc6300b15d

Mansoor, I. 2022. App Revenue Data. Viitattu 20.11.2022. https://www.businessofapps.com/data/app-revenues/

Maureen, J. 2019. Setting up Lint Rules in Dart-Flutter. Viitattu 20.11.2022. https://medium.com/podiihq/setting-up-lint-rules-in-dart-flutter-1ebbed0418a6

Narek, K. 2020. Semantic Versioning. Viitattu 20.11.2022. https://spokeo-engineering.medium.com/semantic-versioning-fff27bce9759

Nitin, D. 2022. Mobile App Or Website? 10 Reaseons Why Apps Are Better. Viitattu 20.11.2022. https://vwo.com/blog/10-reasons-mobile-apps-are-better/

Pilotcore 2021. What Is the Difference Between CI/CD and DevOps? Viitattu 1.2.2022. https://pilotcoresystems.com/insights/what-is-the-difference-between-ci-cd-and-devops

Sematic Versioning www-sivut n.d. Semantic Versioning 2.0.0. Viitattu 20.11.2022. https://semver.org/