TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Artur-Andres Heinamäe 180441

Mobiilirakenduste *CI/CD* arendusprotsessi välja töötamine ja juurutamine ettevõttes Handies Solutions OÜ

Bakalaureusetöö

Juhendaja: Jüri Vain

PhD

Tallinn 2021

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Artur-Andres Heinamäe

18.05.2021

Annotatsioon

Käesoleva bakalaureusetöö eesmärk oli leida lahendus, mis hoiaks kokku Handies ettevõtte mobiilirakenduste arendajate aega, mis kulub manuaalseks rakenduse ehituseks ning uuenduste lansseerimisel *Google Play* ja *App Store*-i keskkonda.

Töö käigus võrreldi erinevaid *CI/CD pipeline* arendusprotsessi automatiseerimise lahendusi, millega on võimalik automatiseerida rakenduse lansseerimise protsessi ning leiti lahendus, mis sobib töös kirjeldatud Handies ettevõtte kahe mobiilirakenduse lansseerimiseks. Seejärel juurutati lahendus Handies ettevõtte näitel ning kirjeldati kahe mobiilirakenduse lansseerimise protsessi.

Lõputöö on kirjutatud eesti keeles ning sisaldab 33 teksti leheküljel, 7 peatükki, 9 joonist, 1 tabelit.

Abstract

CI/CD pipeline development for mobile app products – Handies Solutions OÜ

The purpose of this thesis is to find a solution that helps to automate CI/CD pipeline in Handies company. As a part of this thesis, the author finds and implements a solution that automates the company's CI/CD pipeline to deploy iOS and Android mobile apps to the App Store and Google Play, accordingly.

The bachelor's thesis has five parts: problem and background information, introduction and comparison of automation pipeline tools, analysis of automation tools, CI/CD pipeline automation tool implementation, and analysis of results.

The first part describes the current manual app deployment process in Handies company and gives an overview of the problems of manual deployment.

In the analysis part of the thesis, the author describes and compares CI/CD automation tools, which could solve previously defined problems. As a result, one tool is selected that would meet the criteria set by the company.

In the practical part of the thesis, the author implements the selected solution in Handies company. The author analyzes the development time before and after the implementation of the automation tool. The author also describes possible future solutions of pipeline automation.

The method of CI/CD pipeline automation described in this thesis was implemented in a Handies start-up company.

The thesis is in Estonian and contains 33 pages of text, 7 chapters, 9 figures, 1 tables.

Lühendite ja mõistete sõnastik

OTA	*Over the air*, rakenduste uuendamine üle õhu
CI/CD	*Continuous integration and Continuous delivery*, pidev integratsioon ja pidev edastamine
Firebase App Distribution	Firebase poolt hallatav mobiilirakenduste testversioonide jagamise keskkond
Apple Testflight	IOS süsteemi testimiskeskkond
Gitlab	Versioonihaldustarkvara
Yaml	*Ain't Markup Language*, inimloetav andmete jadastamise programmeerimise keel
Google Play	Android rakenduste pood
App Store	iOS rakenduste pood
Build	Rakenduse ehitus
Expo CLI	*Expo* Käsurea rakendus
Codemagic CLI	Codemagicu käsurea rakendus
MR	*Merge request*, lähtekoodi muutmise kontrollimine harus
React Native	*JavaScript* programmeerimiskeele raamistik

Sisukord

Jooniste loetelu

Tabelite loetelu

Sissejuhatus

Käesoleva bakalaureusetöö eesmärgiks on luua *start-up* ettevõtte Handies näitel lahendus, mis aitaks automatiseerida pidevintegratsiooni töövoogu (*CI/CD pipeline*) ehk teisisõnu vähendada toodangukeskkonda jõudvate vigade arvu ja arendajate aega, mis kulub kahe rakenduse lansseerimisele nii *App Store* kui *Google Play* keskkondadesse.

Bakalaureusetöö on jagatud viide ossa: taustinfo ja lähteülesande defineerimine, automaattööriistade tutvustus ja võrdlus, automaattööriistade analüüs, automaattööriista implementatsioon Handies ettevõttes ja tulemuste analüüs.

Töö taustainfo osas kirjeldatakse hetkeolukorda Handies ettevõttes ning selle põhjal koostatakse lähteülesanne. Arendusprotessi *CI/CD* automatiseerimise eesmärgiks on kiirendada tarkvara tootearendust ja uuenduste jõudmist kliendini. Peamine ajakulu on seotud arendusprotsessi sammude kontrollimise, lansseerimise ja vajalike vormingute koostamisega.

Seejärel kirjeldatakse ja võrreldakse võimalikke turul pakutavaid *CI/CD* arendusprotsessi *pipeline'*i automatiseerimist võimaldavaid tööriistu. Koostöös ettevõttega valitakse välja sobiv tööriist.

Töö tehnilises osas juurutab autor koostöös ettevõttega välja valitud *CI/CD pipeline'i* automatiseerimise tööriista. Autor koostab *codemagic.yaml* faili ning seob ära ettevõtte olemasolevate rakendustega. Lisaks kirjutatakse võimalikest edasiarendustest tulevikus, mida käesoleva lõputöö raames ei ole veel kasutatud ning analüüsitakse ajakulu, mis kulus arendajatel manuaalselt rakenduse lansseerimise protsessi sooritamiseks *Google Play* ja *App Store* keskkonda ning võrreldakse seda automatiseeritud tarkvaraga saavutatud ajakokkuhoiuga.

1 Taust ja probleem

Selles peatükis tutvustab autor ettevõtet Handies ja selle vajadusi tarkvara arendusvahendite järele. Kirjeldatakse ettevõtte tarkvaraarendusprotsessi hetkeolukorda, aspekte, mis piiravad uuenduste jõudmise kiirust kliendini ning potentsiaalselt arendajate aega säästvaid lahendusi.

1.1 Handies ettevõtte tutvustus

Handies Solutions OÜ on start-up ettevõtte, mis loodi 2016. aastal. Ettevõtte põhineb Eesti kapitalil. Nagu teisedki ettevõtted sai ka selle ettevõtte idee alguse omanike vajadusest leida erinevate tööde teostamiseks sobiva profiiliga oma ala spetsialiste. Alati kulus sellise spetsialisti leidmiseks liiga palju aega. Handies ettevõte suur soov on viia omavahel kokku väiketööde tegijad ja kliendid, kes vajavad igapäevatöödes abi. Enam otsitavad erialad on näiteks koristaja, ehitaja, torumees jne. [1] Ettevõtte teenuseid saab tellida kogu Eestis.

Ettevõtte on välja arendanud kaks mobiilirakendust, üks on suunatud väiketööde teenuse pakkujale, rakenduse nimeks on „Handies Tegija“. Ning teine rakendus on suunatud teenuse tellijale ehk kliendile ning rakenduse nimeks on „Handies“. Mõlemad rakendused on leitavad ja alla laaditavad *App Store* ja *Google Play* poest.

Bakalaureuse töö autor alustas antud ettevõttes 1. märtsil 2021 praktikandina, et saada parem ülevaade sellest kuidas antud ettevõttes töö käib, kuidas mobiilirakendusi arendatakse ning kuidas praegu rakenduste väljalaskmise protsess *Google Play* ja *App Store* töötab ning kuidas oleks võimalik seda automatiseerida. Autor alustas antud ettevõttes testija ametikohal, et aidata tuvastada mobiilirakenduste arendamise käigus ilmnevaid vigu, ning neid raporteerida ettevõtte arendaja(tele). Lisaks selle kõrvalt õppis autor iseseisvalt programmeerimise keelt nimega *React Native*, millega antud ettevõtte kaks rakendust on loodud. Samuti osaleb autor testimise ning lõputöö kirjutamise vahepeale arendusega seotud töödes, et saada paremini aru kuidas arendus antud ettevõttes käib. Pärast ülikooli lõppu soovib autor jääda edasi ettevõttesse Handies, alguses mobiilirakenduste arendaja-praktikandina ning seejärel arendajana.

1.2 Rakenduse uuenduste tegemise protsess Handies ettevõttes

Handies ettevõttel on kaks mobiilirakendust. Üks rakendus on suunatud kliendile ja teine rakendus firmale endale kui teenusepakkujale ning eraisikutest Tegijatele. Mõlemad rakendused on kodeeritud mobiilirakenduse programmeerimisekeele *Javascript*i *framework*-iga nimega *React Native*. *React Native* on vabavaraline *JavaScripti* raamistik, mis on mõeldud renderdatavate mobiilirakenduste kirjutamiseks iOS-i ja Android-i platformidel. See põhineb *React*il, mis on Facebooki *JavaScript*i teek kasutajaliideste loomiseks. [2]

Rakenduse ehitamiseks (*buildimiseks)* kasutatakse *Expo*t. *Expo* on universaalsete *Reacti* rakenduste raamistik ja platvorm. See on *React Native* ja *native* platvormide ümber ehitatud tööriistade ja teenuste komplekt, mis aitab arendajatel *JavaScripti* / *TypeScripti* koodibaasist *iOS*-i, *Androidi* ja veebirakendustes kiiresti arendada, ehitada, juurutada ja itereerida. [3]

Handies hoiustab enda rakenduse koodi *Gitlab*i-s. *Gitlab* on veebipõhine *Giti* hoidla, mis pakub tasuta avatud- ja erahoidlaid, probleemijälgimise võimalusi ja vikisid. Tegemist on täieliku *DevOpsi* platvormiga, mis võimaldab professionaalidel täita kõiki projekti ülesandeid - alates projekti planeerimisest ja lähtekoodi haldamisest kuni seire ja turbeni. Lisaks võimaldab see meeskondadel koostööd teha ja paremat tarkvara ehitada. [4]

Rakenduste hoidmiseks on ettevõttel kaks repositooriumi. Ühes repositooriumis on kliendi ja teenusepakkuja rakendus. Teises repositooriumis on kliendi kui ka töö tegija jagatud komponendid, mida mõlemas rakenduses kasutatakse.

Handies ettevõttel on rakenduse jaoks ettevõtte konto keskkondades *Google Play* ja *App Store*. Lisaks on ettevõtte arendajatel arendaja kontod mõlemas keskkonnas, mis on ühendatud ettevõtte konto külge, et oleks võimalik tellija ja kliendi rakendust seal hoida ning viia uuendusi/muudatusi klientideni.

Handies ettevõttes on praegu kaks viisi kuidas ettevõtte arendaja (d) rakenduste uuendusi ning parandusi kasutajatele nähtavaks teevad.

Esimeseks viisiks rakenduste uuenduste tegemiseks on nn *OTA* (*over the air*) ehk üle õhu uuendus. *OTA* värskendused võimaldavad arendajal avaldada oma rakenduse *JavaScripti* koodi ja tarkvara uut versiooni ilma uut rakendusefaili ehitamata ja rakenduste

poodidesse uuesti üles laadimata. [5] Üle õhu uuendus tähendab seda, et rakenduse uuendatud versioon tõmmatakse kasutaja seadmesse alla automaatselt järgmisel rakenduse avamisel. Nii jõuavad uuendused üle õhu kasutajani. Üldiselt tehakse *OTA* uuendust pisivigade korral, nt disainis on vaja värvi või teksti muuta. *OTA* uuenduse lansseerimiseks kulub arendajal vaid paar minutit.

Handies arendajad teevad *OTA* uuendust otse läbi käsurea sisestades käsklusi:

- Android (kliendi ja teenusepakkuja rakenduse puhul)

```
ota update: expo publish --release-channel android-production-channel
```

- iOS (kliendi ja teenusepakkuja rakendus):

```
expo publish --release-channel iOS-production-channel
```

Teiseks viisiks on manuaalselt *.ipa, .apk või .aab* faili üles laadimine *Google Play* või siis *App Store* keskkonda. Esmalt peab ettevõtte arendaja enda muudatusi sisaldava koodi põhjal looma rakendusepoodide poolt vastuvõetavad *.ipa, .apk või .aab* laiendiga failid, kasutades selleks vastavaid terminalikäskluseid. Muutma arvutisse sisseehitatud käsurea terminalis käskluste abil ***.ipa*** või ***.apk*** failiks. Siis peab selle enda arvutisse alla laadima, ning *Google Play* või *App Store* keskkonda sisse logima ning siis selle faili sinna üles laadima. Kliendile tuleb teavitus, et Handies rakenduse uuendused on saadaval, uuenduse teavitus tuleb siis *App Store* või *Google Play* keskkonda. Klient saab need kiirelt ja mugavalt enda nutiseadmesse alla laadida.

Rakenduste testimine. Praegu testitakse rakendusi nutiseadmetes, milleks on Androidi nutitelefonid ning tahvelarvuti. iOS testimiseks on kasutusel iPhone erinevad versioonid, iPad, Apple Iphone 8. Androidi rakenduse testimiskeskkonnaks on *Firebase App Distributor*. *Firebase App Distributorit* on võimalik testijal alla laadida *Google Play* poest või alla laadides läbi veebi. IOS rakenduste testimiseks nutiseadmetest kasutatakse *Testflighti*. *Apple TestFlight* on võimalik testijal alla laadida *App Store* poest.

1.3 Manuaalse rakenduseehituse ja lansseerimise protsess ettevõttes

Handies ettevõtte mobiilirakenduste manuaalse süsteemiehituse ja lansseerimise protsess koosneb järgmistest sammudest:

1. Väljalaske haru (*release branch*) loomine *Gitlab*-i kujul
   release/<rakenduse_nimi>/<versioonikood> nt
   /release/customer/2.1.0
2. Koodiredaktori käsureal käskluse *'expo start'* kaudu lokaalse testserveri käivitamine ning seejärel rakenduse testimine lokaalselt, et näha kas protsess toimib (5-10 min).
3. Manuaalselt rakenduse versiooni suurendamine *app.json* failis.

- Selleks muudetakse järgmiseid väärtuseid:

   Expo.version
   Expo.icon
   iOS.buildNumber
   android.versionCode

4. Rakenduse süsteemiehitus *(build)* käivitamine kasutades selleks *Expo*t
   4.1. Ehitage androidi rakenduste komplekt (**.aab)** Android operatsioonisüsteemi jaoks (*.apk* failitüüp *Firebase* keskkonnas testimiseks).
   käsklus: *Expo build:android*
   [on võimalik valida ***.apk*** ja .**aab** failitüüpide vahel)
   4.2. Ehitage ***.ipa*** fail IOS operatsioonisüsteemi jaoks:
   käsklus: *Expo build:iOS --release-channel "our_release_channel"*
   [arendajalt küsitakse Apple'ilt autentimist]
   4.3 Veenduge, et süsteemi ehitus (*build)* algas
   4.4 Oodake kuni süsteemi ehitus (*build)* on lõppenud.
   (see võib aega võtta ~30minutit, olenevalt olenevalt järjekorra pikkusest *Expo* serverites ja selle tulemuseks on URL, kust faili saab alla laadida).
5. Laadige fail (***.apk*|.aab|*.ipa*)** alla url pealt, mille saate kätte käsurea aknast või *Expo* enda veebirakendusest.
   5.1. Laadige fail üles pärast faili ehitust (*build)* kas *App Store* või *Google Play*-sse.

<u>Android puhul:</u>

6. Laadige *.apk* failitüüp ülesse *Google Play* konsooli looge uus väljalase (*release)*

6.1. Lisage vajalik teave (nt väljalaskemärkmed) ja avaldage värskendus.

IOS puhul:

7. Laadige ***.ipa*** failitüüp ülesse *App Store* kasutades selleks *App Transporter-it*

7.1 Pärast üleslaadimist töödeldakse faili Apple'i serverites ja pärast seda saab selle avaldada *TestFlightis* ja / või *App Store*'is.

1.4 Probleemi kirjeldus ja lähteülesanne

Probleemi kirjeldus: on ettevõtte tarkvaraarendajate liigne ajakulu kahe rakenduse manuaalsel *uuendamisel Google Play* ja *App Store* keskkonda ning uuenduste jõudmise kiirus klientideni.

Töö eesmärk: kiirendada tarkvara tootearenduset ja uuenduste jõudmist kliendini. Peamine ajakulu on seotud arendusprotsessi sammude kontrollimise, lansseerimise ja vajalike vormingute koostamisega. Lisaks sellele on eesmärk parandada tarkvara uuenduste kvaliteeti.

Eeldatav tulem: automatiseeritud arendusprotsess, mis märgatavalt vähendab aega arendusülesande spetsifitseerimisest muudatuste jõudmiseni kliendini (läbi *Goolge Play*, *App Store* või üle õhu (*OTA*)). Eeldatav tulemus peab toetama arendusprotsessi järgmisi samme:

1. Töö märgistamine arenduse planeerimiskeskkonnas märgendiga „tehtud” (*DONE*).

Siinkohal on juba tehtud arenduse poolel automaattestimine *(Unit Tests)* lokaalses keskkonnas.

2. Arendus teeb avalduse (*MR*) liita versiooni halduses arenduse haru testimise haruga.
3. Avaldus vaadatakse üle pea-arendaja (te) poolt.
4. Harud liidetakse.
5. Läheb käima *CI/CD pipeline*, millega käivitatakse automaattestid (*Unit Tests)*, kontrollitakse koodi kvaliteeti (n. *Eslint*). Eduka läbimise järel:
6. Automaatselt ehitatakse *.apk (.aab) ja .ipa* failid.
7. *.apk (.aab)* fail lansseeritakse *Firebase App Distributor* keskkonda *Android* süsteemil testimiseks.

8. *.ipa* fail lansseeritakse *Apple Testflight* keskkonda *iOS* süsteemil testimiseks.
9. Toimub käsitsi testimine *QA* poolt.
10. Töö liigub tööde süsteemis kvaliteedi kontrolli (*QA*).
11. Töö läbib esmase kontrolli (manuaalne testimine, kui vajalik).
12. Töö märgitakse *QA* poolt kui „kontrollitud" (*ready for launch)* või „puudustega" (*revise*)
13. Kui puuduseid ei esine, tehakse (*MR*) et liita test haru produktsiooni haruga.
14. Avaldus vaadatakse üle pea-arendaja (te) poolt.
15. Harud liidetakse.
16. Läheb käima *CI/CD pipeline*, millega täidetakse automaattestid *(Unit Tests)*, kontrollitakse koodi kvaliteeti (n. *Eslint*). Eduka läbimis järel:
17. Automaatselt ehitatakse *.apk (.aab) ja .ipa* failid.
18. *.ipa ja apk (.aab)* fail lansseeritakse vastavalt *App Store* ja *Google Play* keskkonda.
19. Lõpp: kliendid näevad uuendusi.

Üldjuhul ei ole vaja iga uuendusega läbida protsessi kõiki samme. Korduv protsess läbitakse iga 3 või 5 päeva järel. Kui protsessi kõiki samme ei ole võimalik või mõistlik automatiseerida, tuleb töötada välja asendusprotsess, mis seda aitaks võimalikult kiirelt ja kvaliteetselt teha.

2 Automaattööriistade tutvustus

2.1 *CI/CD* automaattööriistade võrdlus

Järgnevalt kirjeldab autor erinevaid *CI/CD pipeline* automatiseerimise tarkvarasid, mida praegu turul pakutakse. Lisaks koostab autor võrdleva tabeli erinevatest tööriistadest. Võrdlusandmete põhjal valitakse koostöös ettevõttega välja üks tööriist ning implementeeritakse see ettevõttes Handies.

2.1.1 *Gitlab CI*

Gitlab on veebipõhine *Giti* hoidla, mis pakub tasuta avatud ja erahoidlaid, nende jälgimisvõimalusi ja wikisid. *Gitlab* on *DevOpsi* platvorm, mis võimaldab arendajatel täita kõiki projekti ülesandeid - alates projekti planeerimisest ja lähtekoodi haldamisest kuni seire ja turbeni. Lisaks võimaldab see meeskondadel koostööd teha ja paremat tarkvara ehitada. [6]

Põhiomadused

- Tööriista *branching* abil saab vaadata (V*iew*), luua (*Create*), hallata (*Manage*) projekti andmeid ja projekti koodi
- Automatiseerud rakenduseehitus, integratsioon, lähtekoodi verifikatsioon
- Konteineri skaneerimine *(container scanning)*, staatilise rakenduse turvalisuse testimine *(SAST)*
- Dünaamilise rakenduse turvalisuse testimine (*DAST*) [7]

2.1.2 Buddy

Buddy on veebipõhine *CI/CD pipeline* tööriist, mida saab kasutada veebisaitide ja mobiilirakenduste rakenduse ehituseks *(build)*, testimiseks ja juurutamiseks. Ühildub: *GitHub, Bitbucket ja GitLab-ga*. [8]

Põhiomadused

- *Native Dockeri* tugi.
- *Pipeline*de loomisel *UI* ja *Yaml* konfiguratsioonifaili tugi
- Toetab: *FTP / SFTP, AWS, Google Cloud, Digital Ocean, Heroku, Kubernetes*
- Integreerimise võimalus: *GitHubi*, *Gitlab*i ja *BitBucketiga*

- Rakenduse ehitus *(Builds)* käivituvad isoleeritud konteinerites koos vahemälu muutujatega. [9]

2.1.3 *Bitrise*

Bitrise on pidev integreerimise ja kohale toimetamise platvorm kui teenus (*PaaS*), mis keskendub peamiselt mobiilirakenduste (*iOS, Android ja Xamarin*) arendamisele. [10]

Põhiomadused:

- Üle õhu *(Ota)* rakenduse kasutuselevõtt
- *GitHub* tõmbetaotluse (*pull)* tugi
- Toetab Kolmanda osapoole *beta* testimist ja juurutamise teenused *(deployment services)*
- Automaatne koodi allkirjastamine *(code signing)* [11], [12], [13]

2.1.4 *Codemagic*

Codemagic on *CI/CD* automatiseerimise tööriist mobiilirakenduste projektide jaoks. Toetab: *Android, iOS, React Native,* and *Flutter* rakenduse ehitust *(build).* Lisaks on olemas ka ühiktestide (*unit tests),* intekratsioonitestide (*integration tests)* ja seadmetes testimine (*tests on real devices)* toetus. [14]

Põhiomadused:

- Automaatne konfiguratsioon ja seadistamine
- Integreerimise võimalus koodihoidlastega: *Bitbucket*, *GitHub*, *GitLab*
- Mitme samaaegse rakenduse ehituse võimalus
- Automaatne avaldamine teenustesse: *Google Play, iTunes Connect, HockeyApp, Crashlytics, TestFair*
- Heal tasemel dokumentatsioon [15]

2.1.5 *CircleCI*

CircleCI on pideva integreerimise *(CI)* ja pideva edastamise (*CD*) platvorm. *CircleCI* Enterprise'i lahendus on installitav privaatsesse pilve või andmekeskusesse ja seda saab piiratud aja jooksul kasutada tasuta. *CircleCI* automatiseerib tarkvara loomist, testimist ja juurutamist. [16]

Põhiomadused:

- Ühilduvus: *Bitbucket, GitHub, and GitHub Enterprise*
- Käivitab rakenduse ehituse (*builds)* kasutades konteiner või virtuaalset masinat
- Lihtne silumine
- Isikupärastatud e-posti ja IM-teated
- Pidev ja haru spetsiifiline juurutamine
- Automatiseeritud ühendamine *(merging)* ja kohandatud käsud pakettide *(package)* üleslaadimiseks
- Kiire seadistamine ja piiramatu rakenduse ehitamine *(builds)* [17]

2.1.6 *Github Actions*

GitHub on pilvepõhine *(Cloud-Based)* hostimisteenus, mis pakub kasutajatele kasutajasõbralikku kasutajaliidese kasutuskogemust git-versioonide jaoks. *GitHub Actions* võimaldab kasutajal luua otse enda *GitHubi* koodihoidlatesse kohandatud tarkvaraarenduse elutsükli (*SDLC*) töövood. Lisaks võimaldab ka otse koodihoidlasse kasutusele võtta pideva integreerimise (*CI*) ja pideva juurutamise (*CD*). [18]

Põhiomadused:

- Sisse ehitatud pideva integratsiooni (*CI*) ja pidev kasutuselevõtu *(CD)* tööriist
- Näidisfailid pideva integratsiooni *(CI)* ülesseadmiseks
- *Linux, macOS, Windows, ARM* toetus
- Toetab: *Node.js, Python, Java, Ruby, PHP, Go, Rust, .NET*
- Live logid (*Live logs)*
- Mitmekonteinerile testimisvõimalus *(Multi-container testing)* [19]

2.1.7 Visual Studio App Center

Visual Studio App Center võimaldab arendajatel automatiseerida ja hallata IOS, *Androidi, Windowsi ja MacOS-i* rakenduste elutsüklit. Pakub rakenduse ehituse automatiseerimist, rakenduse testimine ettevõtte poolt pakutavate mobiiltelfoni emulaatorite abil, rakenduse reaalse kasutamise monitoorimist. [20]

Põhiomadused:

- Ühilduvus: *GitHubi*, *Bitbucketi* või *Azure* repositooriumitega
- pidev integreerimise (*CI*), pidev edastamine (*CD*) integreerimise võimalus
- rakenduse ehituse toetus: *Swift*, *Objective-C*, *Java, React Native, Xamarin* ja *UWP*
- *UI* testide automatiseerimise võimalus

- Rakenduse jagamise võimalus test kasutajatega [21]

2.1.8 *Appcircle*

Appcircle on mobiilne *(mobile) DevOps*-i *CI/CD* platvorm, mis pakub täielikult automatiseeritud keskkonda mobiilirakenduse elutsükli otsast lõpuni haldamiseks. [22]

Põhiomadused:

- Kiire ja lihtne seadistus *git* repositoorium *app circle* tööriistaga
- Automatiseeritud pilveehitus *(Automated Cloud Build)*
- sisse ehitatud *CI/CD* rakenduseehituse tööriist
- **Paindlik kasutuselevõtt** *(Flexible Deployment)*: binaarse juurutuse automatiseerimine testijatele ja poodidele
- **Veebibrauserisse sisse ehitatud mobiiliseadme emulaator**: Rakenduste eelvaate kuvamine virtuaalsetes iOS- ja Android-seadmetes [22]

2.1.9 *Azure DevOps*

Azure DevOps on Microsofti poolt loodud tarkvara kui teenus (*SaaS*) platvorm, mis pakub tarkvara arendamiseks ja juurutamiseks otsast-lõpuni *DevOps*-tööriistu. [23]

Põhiomadused:

- Kohandatud Visual Studio'ile
- Pidev integreerimine ja juurutamine (CI/CD)
- Toetab: manuaalset, uurivat ja pidev testimist (manual, exploratory ja continuous testing)
- Rakenduse ehituse (build) võõrustamine kasutades Dockerit
- Toetab kahte tüüpi allikakontrolli: Git (hajutatud) või Team Foundationi versioonihaldust (TFVC) [24], [25]

2.1.10 *Travis CI*

Travis CI on võõrustatud pideva integreerimise *(CI)* teenus, mis on avatud lähtekoodiga projektide jaoks tasuta ja mida kasutatakse tarkvaraprojektide ehitamiseks ja testimiseks. Pakub nii väikeste kui ka suurte koodimuudatuste tuge, muudatuste avastamise jälgimist. Lisaks koodimuudatuse õnnestumise või ebaõnnestumise tagasisidet. [26]

Põhiomadused:

- Kiire ülesseadmine
- *macOS*, *Linux*, ja *iOS* toetus

- Automaatne tõmbetaotluse *(pull requests)* kontrollimine
- E-posti, *Slacki*, *HipChati* ja muude koostöövahendite integreerimise võimalus
- *API* ja *CMD* tööriista valikud halduse kohandamiseks
- Toetab mitmeid üldlevinud programmeerimiskeeli: *C#, Android, Java, C++, JavaScript (with Node.js), PHP, Perl, Python, R, Ruby* [9], [24]

2.2 Ise-hallatava *(Self-hosted)* ja pilvepõhise *(Cloud-based) CI/CD* võrdlus

2.2.1 Ise-hallatav *(Self-hosted) CI/CD*

Ise-hallatav *(Self-hosted)* *CI/CD*-lahendused töötavad kasutaja kohalikus arvutis koos sellele masinale installitud tarkvarapakettidega. Mõned näited ise hallatavatest *CI/CD* tööriistadest on *Jenkins*, *TeamCity* jne. [27]

Plussid

- Ise-hallatava lahenduse üks suurimaid eeliseid on laiendatavus. Mõnda ise-hallatavat teenust saab kohandada pistikprogrammidega, et võimaldada funktsionaalsust, mis pole komplektis.
- Kogu *CI / CD* süsteemi turvalisuse haldamine on kasutaja vastutusel. See võib olla äärmiselt aeganõudev ja keeruline töö, kuid te ei pea lootma väliste pakkujate turvalisusele. Kui kaob vajadus minna väljapoole oma eravõrke, aitab see vähendada turvaohte. [27]

Miinused

- Tulevikus rakenduse ehitustõrgete vältimiseks peab kasutaja oma riistvarasüsteeme hooldama ja tarkvara regulaarselt värskendama. [27]
- Võib olla väga aeganõudev seadistusprotsess projektide ühendamiseks VCS-iga, probleemide jälgimise ja teavitussüsteemidega.
- Kasutaja peab haldama kogu autentimist, mida meeskonnaliikmed vajavad projektidele juurdepääsemiseks. [27]

2.2.2 Pilvepõhine *(Cloud-based) CI/CD*

Pilvepõhine (*Cloud-based)* *CI/CD (*hästi tuntud kui **SaaS tööriist**) kasutab riistvarasüsteeme ja pilveserverisse salvestatud tarkvara. Mõned näited pilvepõhistest *CI / CD* tööriistadest on *Travis CI, CircleCI, Codemagic* jne. [27]

Plussid

- Pilvepõhise *CI/CD*-lahenduse kasutamise suurim eelis on see, et kasutaja ei pea haldama riistvara ega tarkvara infrastruktuuri. Kõigi tarkvarauuendustega tegeleb ka pilvepõhine *CI/CD* pakkuja. [27]
- Esmane seadistamine on kiire ja lihtne. Samuti on enamikul *SaaS-i CI/CD*-lahendustest hea integreerimine *git*-põhiste *VCS*-idega nagu *GitHub, Gitlab, Bitbucket* jne [27]

Miinused

- Kasutatava teenuse kohta võib tekkida lisakulusid. Kasutaja meeskonna suurenedes võib tõusta ka kasutatava pilvepõhise *CI/CD*-teenuse hind. [27]
- Kõik pilvepõhised *CI/CD*-lahendused ei toeta kõiki platvorme, tööriistu ja keskkondi. [27]

Tabel 1. Automatiseerimistööriistade võrdlus

***CI/CD* tööriistad**	*Gitlab CI*	Buddy	*Bitrise*	*Codemagic*	*CircleCI*	*Github Actions*	*VS App Center*	*Appcircle*	*Azure DevOps*	*Travis CI*
Asutatud	2011 [28]	2015 [29]	2015 [28]	2017 [28]	2011 [28]	-	2016 [28]	2019 [30]	-	2011 [28]
Toetab operatsioonisüsteeme:	Linux, Windows, Mac	Winodws, Linux, Mac	Windows, Linux, Mac	Windows, Linux, Mac	Windows, Linux, Mac	Linux, *macOS*, and Windows.	Windows, Linux, Mac	Windows, Linux, Mac	Windows, Linux, Mac	Windows, Linux, Mac
Süsteemi ehituse kiirus (*Build speed)* Android & IOS	(07m44s / 12m47s); - [31]	**(01m32s / 04m20s);** - [31]	(03m16s / 10m57s); (32m36s / 39m/43s) [31]	(05m18s / 06m58s); (19m55s / 30m27s) [31]	(**02m22s / 04m19s**); - [31]	**(02m44s / 04m56s);** (13m53s / 26m58s) [31]	(03m25s / 06m22s); (19m21s / 22s12s) [31]	(07m16s / 10m53s); (20m05s / 15m44s) [31]	(04m04s / 06m26s); 16m19s / 20m20s [31]	(03m29s / 06m38s); (35m41s / 41m28s) [31]
Ühildumine	Campfire, Flowdock, Jira, Pivotal Tracker, and Slack	*Gitlab*, Github, Bitbucket	*Gitlab*, Github, Jira, Bitpucket,Slack	Github,*Gitlab* , Butbucket	*Gitlab*, github	Azure pipeline	Github, Bitbucket, Visual Studios Team Services, Slack, Microsoft Teams	*Gitlab*	*Gitlab*	*Gitlab*
Ülesseadmine **(lihtne/keskmine/raske)**	lihtne	lihtne	lihtne	lihtne	raske	raske	raske	lihtne	keskmine	keskmine
Pilvepõhine *(SAAS)* vs ise hallatav *(Self-Hosted)*	SaaS, ise hallatav	Ise hallatav	SAAS	SAAS	SAAS, Ise hallatav	SAAS, Ise hallatav	SAAS	SAAS, Ise hallatav	SAAS, Ise hallatav	Ise hallatav
Android & IOS automatiseerimise võimalus	IOS Android	Android	IOS Android	IOS Android	IOS Android	IOS Android	IOS Android	IOS Android	IOS Android	IOS Android
Hinnaplaan	Tasuta versioon,	Tasuta versioon, Professionaa	Üksik arendaja: $40/ kuus,	Tasuta versioon, Maksa siis kui	$30/ kuus [36]	Tasuta versioon, Meeskond	$40/$99 kuus [38]	Tasuta versioon, arendaja:$	Tasuta versioon, maksa siis	Tasuta versioon, 1 töö *(job)*

	$19/$99 [32]	lne:$75/kuu, Hüper:$200/ kuus, *On-premises* $35/kuus ühe kasutaja kohta [33]	Org Standard:$ 100/ kuus, Org Eliit:$300/ kuus [34]	kasutad, Ettevõttele alates $299/ kuus [35]		: €3,31per kasutaja/k uus, Ettevõte € 17 kasutaja kohta/ kuus [37]		39/ kuus, profession aalne:$ kuus [39]	kui kasutad [40]	Plan: $69/ kuus, 2 tööd *(jobs)* $129/kuus, 5 tööd (*jobs)*: $249/kuus [41]
Tasuta prooviversioon	30 päeva	30 päeva	14 päeva	puudub	puudub	puudub	30 päeva	30 päeva	puudub	puudub
***Node.js* toetus**	Toetab	Toetab	Toetab	Toetab	Toetab	Toetab	Toetab	Toetab	Toetab	Toetab
Virtualiseerimise keskkond (*macOS*)	Ei toeta	Ei toeta	Toetab	Toetab	Toetab	Toetab	Toetab	Ei toeta	Toetab	Toetab
Veebileht	https://docs .*Gitlab*.co m/ee/ci/	https://buddy .works/docs	https://ww w.*Bitrise*.io /why/techn ologies/rea ct-native-continuous-integration	https://docs.*C odemagic*.io/	https://*Cir cleCI*.com /*CircleCI*-react/#:~:t ext=and%20Androi d%20apps%20on%20React,yml%20file.	https://doc s.github.c om/en/acti ons	https://doc s.microsof t.com/en-us/appcent er/*build*/re act-native/	https://*App circle*.io/bl og/guide-to-automated-mobile-ci-cd-for-react-native-*Appcircle*/	https://azur e.microsoft .com/en-us/services /devops/pi pelines/	https://trav is-ci.com/pla ns

3 Automaattööriistade analüüs

Selles peatükis võrdleb autor eelnevalt kirjeldatud kolmanda osapoole automatiseerimise tarkvarasid ning leiab sobiva tööriista, mille abil lahendada eelnevalt püstitatud probleemi.

3.1 Tööriista valiku põhikriteeriumid

Autor lähtub lahenduste võrdlemisel seitsmest püstitatud põhikriteeriumist mis tulid sisendina ettevõtte peaarendaja poolt.

Põhikriteeriumid on järjestatud prioriteedi järgi.

1. lahenduse maksumus;
2. *macOS* toetus;
3. Pilvepõhine *(Cloud-based) CI* tööriist;
4. ühilduvus *Gitlab*i-ga;
5. *Node.js* toetus;
6. ühilduvus *React Native* ja *Expo*ga;
7. lahenduse kasutuselevõtu keerukus;

3.2 Võrdlusprotsessi kirjeldus

Autor alustab võrdlust sellest, mis tarkvarad toetavad nii iOS (*App Store)* kui ka Androidi (*Google Play)* mobiilirakenduste automatiseeritud avaldamise võimalust. *CI/CD* automatiseerimise tööriist nimega *Buddy* langeb võrdlusest välja, kuna *Buddy* ei paku hetkel enda tööriistas sees *IOS App Store* automatiseerimise võimalust ega *macOS* keskkonda.

Võrdlusesse on jäänud alles üheksa tarkvara: *Gitlab CI, Bitrise, Codemagic, CircleCI, Github Actions, VS App Center, Appcircle, Azure DevOps, Travis CI.*

Järgnevalt analüüsib autor *Gitlab*'i sisse ehitatud *CI/CD* automatiseerimistööriista võimalust kuna rakenduse koodi hoiustatakse *Gitlab*is. *Gitlabi* sisse ehitatud tööriist langeb võrdlusest välja, kuna *Gitlab* ei paku sisse ehitatud *macOS* tuge, mida oleks vaja IOS rakenduse kasutuselevõtmiseks ja üles laadimiseks *App Store*. Teiste sõnadega Windowsi kasutajad peaksid kasutama *Mac*-i virtuaalmasinat ning läbi virtuaalmasina seadistama *Gitlab*-i *pipeline.* Sealt edasi peab iga kord, kui on soov *IOS* rakendust juurutada, olema virtuaalmasin töös.

Autor võrdleb *Azure DevOps*, *Appcircle* ja *VS App Center* automatiseerimise tarkvarasi. Automatiseerimise tööriistad *Appcircle* ja *VS App Center* on omakorda rakenduse ehituse **(*build*)** protsessi automatiseerinud, mis sobiks hästi siis, kui Handies ettevõtte arendajad kasutaksid käskluste käivitamiseks ja rakenduse ehituseks „puhast *React Native* 'it". Praegu on kasutusel rakenduse ehitusel *Expo*, ehk rakenduse ehitus käib läbi *Expo* serveri, automatiseerimistarkvarasse mille sees käib rakenduse ehitus sinna külge *Expo* ühendamise protsess on keeruline. Lisaks langeb võrdlusest välja ka *Azure DevOps* lahendus, kuna *macOS* virtuaalkeskkonna ülesseadmine on seal keerukas ja aeganõudev.

Võrdlusesse on alles jäänud tööriistad, mis pakuvad sisse ehitatud *macOS* virtuaalkeskkonda: *Bitrise*, *Codemagic*, *CircleCI ja TravisCI*. Sisse ehitatud *macOS* virtuaalkeskkond tähendaks Handies arendajale seda, et arendajal piisab ainult automatiseerimistööriistale lanseerimise *pipeline* üles ehitamisest ja enda rakenduse repositooriumi külge integreerimisest.

Viimaseks võrdluse kriteeriumiks on tööriista hind. *CircleCI* tööriista maksusmus on 30 dollarit kuus [36] *Bitrise* maksumus 40 dollarit kuus [34] TravisCI maksumus 69 dollarit kuus [41] *Codemagic* tööriist pakub tasuta versiooni või siis maksta kasutuspõhiselt. [35]

Võttes arvesse kõiki kriteeriume: lahenduse maksumus (prooviversiooni olemasolu), ühilduvus *Gitlabi*-ga, pilvepõhine *ci*, ühilduvus *React Native* ja *Expo*ga, sisse ehitatud *macOS* tugi, *Node.js* tugi, autor koostöös ettevõttega otsustasid *Codemagic* automatiseerimise tarkvara kasuks. Seega alustatakse Handies ettevõttes *Codemagic*u tarkvara paigaldust.

4 Rakenduse ehitusfaili *(build)* konfigureerimine

Siin on näidis ***codemagic.yaml*** rakenduse ehitusfaili konfiguratsioonist, *CI/CD* arendusprotsessi automatiseerimise jaoks, mida vastavalt enda rakenduse tüübile ning vajadustele on võimalik sobivaks kohandada. [42]

```
workflows:
  my-workflow:
  name: My workflow name
  instance_type: mac_mini
  max_build_duration: 60
  environment:
  vars:
  PUBLIC_ENV_VAR: "value here"
  flutter: stable
  xcode: latest
  cache:
  cache_paths:
  - ~/.pub-cache
  triggering:
  events:
  - push
  branch_patterns:
  - pattern: '*'
  include: true
  source: true
  cancel_previous_builds: false
  scripts:
  - ...
  artifacts:
  - build/**/outputs/**/*.aab
  publishing:
  email:
  recipients:
  - name@example.com
  scripts:
  - echo 'Post-publish script' [42]
```

Joonis 1. Codemagic.yaml näidis [42]

4.1 Yaml faili osade kirjeldused

Töövoog (*workflow)* - võtke aluseks ***codemagic.yaml*** näidisfail**,** et defineerida enda rakenduse töövood *(workflows)* enda projekti jaoks. Iga töövoog *(workflow)* kirjeldab otsast-lõpuni rakenduse ehituse (*build) pipeline-i* käivitamisest kuni avaldamiseni. Näiteks võivad siin failis kirjas olla rakenduse arendamiseks, testimiseks ja avaldamiseks eraldi töövood *(workflows)*. [42]

```
workflows:
 my-workflow: # workflow ID
 name: My workflow name # workflow name displayed in
Codemagic UI
 instance_type: mac_mini # machine instance type
 max_build_duration: 60 # build duration in minutes (min 1, max
120)
 environment:
 cache:
 triggering:
 scripts:
 artifacts:
 publishing: [42]
```

Joonis 2. Rakenduse ehituse töövoog [42]

4.2 Töövoo *(workflow)* osade kirjeldused

4.2.1 Eksemplari tüüp *(Instance Type)*

Eksemplari tüüp (Instance_type): määratleb rakenduse ehitamise masina tüübi, mida läheb vaja rakenduse ehitamiseks. [42]

Codemagic toetab rakenduse ehitamisel *(build)* järgnevaid tööjaamu:

- *mac_mini – macOS standard VM,*
- *mac_pro- macOS premium VM,*
- *linux- Linux standard VM,*
- *linux_x2- Linux premium VM.* [42]

NB: kõik **premium** kategooria virtuaalmasinad on kasutusel ainult tasuliste pakettide puhul. [42]

4.2.2 Keskkond (*Environment*)

Keskkond (*environment)*: sisaldab kõiki keskkonnamuutujaid ning võimaldab määrata *Flutter, Xcode, CocoaPods, Node ja npm* versiooni, mis kasutatakse rakenduse ehitamiseks (***buildimiseks)***. Samamoodi on võimalik lisada ka koodi allkirjastamiseks *(code signing)* vajalikud mandaadid ja API-võtmed. Võimalusel krüpteerige tundlikke andmeid hoidvate muutujate väärtused. [42]

```
environment:
 vars: # Define your environment variables here
 PUBLIC_ENV_VAR: "value here"
 SECRET_ENV_VAR: Encrypted (...)

 # Android code signing
 FCI_KEYSTORE: Encrypted (...)
 FCI_KEYSTORE_PASSWORD: Encrypted (...)
 FCI_KEY_PASSWORD: Encrypted (...)
 FCI_KEY_ALIAS: Encrypted (...)

 # iOS automatic code signing
 APP_STORE_CONNECT_ISSUER_ID: Encrypted (...)
 APP_STORE_CONNECT_KEY_IDENTIFIER: Encrypted (...)
 APP_STORE_CONNECT_PRIVATE_KEY: Encrypted (...)
 CERTIFICATE_PRIVATE_KEY: Encrypted (...)

 # iOS manual code signing
 FCI_CERTIFICATE: Encrypted (...)
 FCI_CERTIFICATE_PASSWORD: Encrypted (...)
 FCI_PROVISIONING_PROFILE: Encrypted (...)

 # Firebase secrets
 ANDROID_FIREBASE_SECRET: Encrypted (...)
 IOS_FIREBASE_SECRET: Encrypted (...)

 SSH_KEY_GITHUB: Encrypted (...) # defining an ssh key used to download
private dependencies
 CREDENTIALS: Encrypted (...) # publishing a package to
pub.dev
 APP_CENTER_TOKEN: Encrypted (...) # publishing an
application to App Center

 flutter: stable # Define the channel name or version (e.g.
v1.13.4)
 xcode: latest # Define latest, edge or version (e.g. 11.2)
 cocoapods: 1.9.1 # Define default or version
 node: 12.14.0 # Define default, latest, current, lts, carbon
(or another stream), nightly or version
 npm: 6.13.7 # Define default, latest, next, lts or version
 ndk: r21d # Define default or revision (e.g. r19c)
 java: 1.8 # Define default, or platform version (e.g. 11).
The default platform version is 1.8 [42]
```

Joonis 3. Rakenduse ehituse keskkond [42]

4.2.3 Vahemälu (*Cache)*

Vahemälu (*cache)*: määratleb ära vahemällu salvestatavad ja *Codemagic*u salvestatavad teed *(paths)*. Näiteks võite kaaluda järgmiste teede *(paths)* vahemällu salvestamist [42]:

Path	Description
`$FLUTTER_ROOT/.pub-cache`	Dart cache
`$HOME/.gradle/caches`	Gradle cache. Note: do not cache `$HOME/.gradle`
`$HOME/Library/Caches/CocoaPods`	CocoaPods cache

```
cache:
 cache_paths:
 - ~/.gradle/caches
 - ... [42]
```

Joonis 4. Vahemälu teekond [42]

4.2.4 Käivitamine (*Triggering*)

Käivitamine *(triggering)*: defineerib automaatse rakenduse ehituse *(automatic build)*käivitamise ja vaadatud harude *(watched branches*) sündmused *(events)*. Kui ühtegi sündmust *(events)* pole määratletud, on võimalik käivitada rakenduse ehitus *(builds)* ainult manuaalselt.

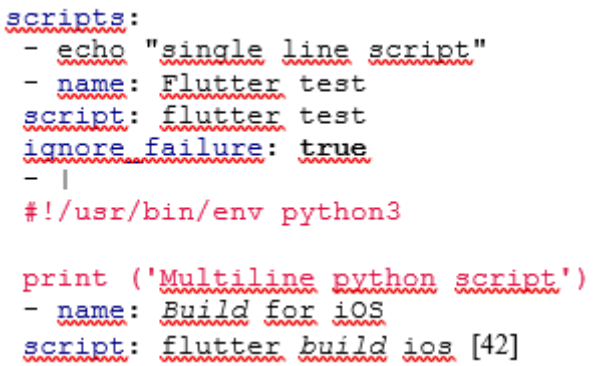
Haru muster *(branch pattern)* võib ühilduda konkreetse haru nimega või võimalik on kasutada metamärkide *(wild cards)* sümboleid mitme haruga sobiva mustri loomiseks. [42]

Joonis 5. Automaatse rakenduse käivitus [42]

4.2.5 Skriptid (*Scripts*)

Skriptid täpsustavad, milline rakendus on ehitatud *(built)*. Siin on võimalik määrata käsklused projekti testimiseks *(test)*, rakenduse ehitamiseks *(build)* ja enda projekti koodimärgid *(code sign)*. [42]

Kui **ignore_failure** väärtuseks on määratud ***true***, töötab töövoog ka siis, kui skript ebaõnnestub. [42]



Joonis 6. Rakenduse ehituse skriptid [42]

4.2.6 Väljundfailid (Artifacts)

Konfigureerige enda teekonnad *(paths)* ja nimed mida soovite enda väljundfailis *(artifacts)* kasutada. Kõik teed *(paths)* on relevantsed kloonide kataloogiga *(clone directory)* absoluutsed teed (*absolute paths)* on samamoodi lubatud. [42]

```
artifacts:
 - build/**/outputs/**/*.apk # relative path for a project
in root directory
 - subfolder_name/build/**/outputs/**/*.apk # relative path
for a project in subfolder
 - build/**/outputs/**/*.aab
 - build/**/outputs/**/mapping.txt
 - build/ios/ipa/*.ipa
 - build/macOS/**/*.pkg
 - /tmp/xcodebuild_logs/*.log
 - flutter_drive.log [42]
```

Joonis 7. Väljundfailide konfigureerimine [42]

4.2.7 Avaldamine (*Publishing*)

Kõik loodud väljundfailid *(artifacts)* on võimalik saata väliteenustesse *(external services)*. Toetatud väliteenused on email, *Slack*, *Google Play* ja *App Store Connect*. [43]

```
publishing:
 email:
 recipients:
 - name@example.com
 notify:
 success: false # To not receive a notification when a build
succeeds
 failure: false # To not receive a notification when a build
fails [43]
```

Joonis 8. Väljundfailide avaldamise võimalused [43]

4.3 Edasised sammud pärast *codemagic.yaml* valmimist

1. Kui ***Codemagic*-u süsteemi ehituse *(build)*** konfiguratsioonifail ***codemagic.yaml*** on valmis, laadige see fail enda rakenduse repositooriumisse, mis asub *Gitlab*is. [44]

2. Pöörduge tagasi *Codemagic*u rakenduse seadete (*app settings)* juurde, otsige üles *codemagic.yaml* fail, *codemagic.yaml* faili skannerimiseks, valige haru *(branch)* mida soovite skaneerida *(scan)* ning vajutage nuppu ***Check for configuration file***, mis asub lehe ülaosas. [44]

3. Kui *codemagic.yaml* fail on tuvastatud *Codemagicu* süsteemi poolt sellest harust *(branch)*, klõpsake nupul *"Start your first build"* ja valige töövoog *(workflow)* ja haru *(branch)* mida soovite kasutada rakenduse ehitusel. [44]
4. Lõpuks, vajutage nupule "Start new *build*" et käivitada rakenduse ehitamine. [44]

5 *CI/CD pipeline-i* implementatsioon Handies ettevõttes

Käesolevas peatükis tutvustab autor üldist protsessi kuidas käib ettevõtte *React Native* rakenduste lansseerimine *Codemagic*-u kolmanda osapoole tööriistaga. Lisaks tutvustab autor Handies ettevõtte vajadustele kohandatud *codemagic.yaml* faili.

5.1 Üldkasutatavad *CI* skriptid

Handies ettevõtte jaoks loodud CI skript kasutab rakenduste ehitamiseks järgmiseid käsurealiideseid (CLI): npm, *Expo*, Fastlane, Yarn.

- ***NPM (node Package Manager)***- *npm* on Node JavaScripti platvormi paketihaldur. Seda kasutatakse erinevate serveripoolsete sõltuvuste haldamiseks. [45]

- ***Expo CLI*** - *Expo CLI* on käsurea rakendus, mis on arendaja ja *Expo* tööriistade peamine liides. Kasutatakse mitmesuguste ülesannete täitmiseks, näiteks uute projektide loomiseks, rakenduse arendamisel projekti serveri käitamine, logide vaatamine, rakenduse avamine simulaatoris. [46]

- ***Fastlane*** – on lihtsam viis kuidas enda *iOS ja Android* rakenduste juurutamist ja väljalaskeid automatiseerida. See võimaldab arendajal automatiseerida enda arenduse töövoogu. [47]

- ***Yarn***- *„Yet Another Resource Negotiator“,* see on avatud lähtekoodiga paketihaldur nagu *npm*. Ning aitab hallata projekti sõltuvusi. [48]

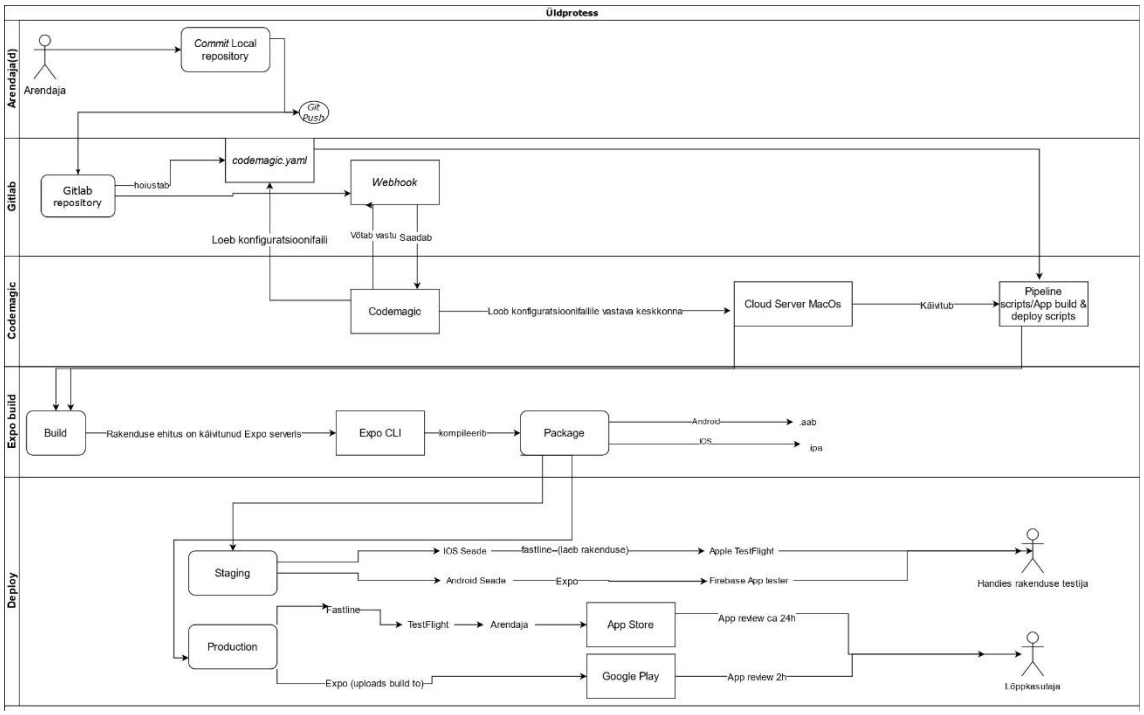
5.2 Juurutusprotsess kasutades *Codemagic* tarkvara

Järgnevalt on kirjeldatud esialgne kavand Handies *CI/CD pipeline* loomiseks ***(vt. Joonis 9).***

1) Ettevõtte arendaja sooritab enda arvutis *commiti* lokaalsesse repositooriumisse, seejärel sisestab käskluse *“git push”* (mille tulemusena laadidakse lokaalsed *commititud* koodimuudatused *Gitlab*-i repositooriumisse üles).
2) Kood jõuab *Gitlab*-i repositooriumisse, repositooriumis on ***codemagic.yaml*** fail (kus on rakenduse lansseerimiseks vajalikud käsud *pipeline* sees ära defineeritud).

3) *GitLab* suhtleb läbi *webhooki Codemagic*uga, mis käivitab rakenduse ehituse protsessi.
4) *Codemagic* loeb repositooriumist ***codemagic.yaml*** konfiguratsioonifaili ning loob konfiguratsioonifailile vastava virtuaalkeskkonna, kus käivitab ette antud käsud (nende hulka kuuluvad *Expo*, *npm* ja *fastline* käsud).
5) Vastavalt skriptis defineeritud käsule käivitub rakenduse ehitus *Expo* serveris (*Expo CLI)*, mille tulemuseks on vastavalt kas iOS paketifail (.*ipa*) või Androidi rakendusefail .*aab.*
6) ***.ipa*** ja ***.aab*** andmetüübid laetakse üles kas
    1. **Testkeskonna puhul (***Staging)*
        - *Apple TestFlight*: testkasutaja iOS seade
        - *Firebase App Distributor*: testkasutaja Android seade

        Või
    2. **Toodangukeskkonna puhul** (*Production)*
    - IOS rakenduse fail (.*ipa*) laadikatse *Fastlane* käsurealiidest kasutades üles *TestFlight* keskkonda, kust edasi toimub rakenduse uue versiooni avaldamise protsess manuaalselt, et teha rakendusele viimane kontroll ja lisada kasutajatele info uuenduse kohta (nt *release notes*).
    - Android rakenduse üles laadimine *Google Play* poodi- läbinud rakenduse ehituse *Expo* serveris, kasutame taas *Expo* käsurealiidest, et laadida üles 6. sammus genereeritud .*aab* Android rakenduse faili *Google Play* poodi.
7) Rakenduse uuendus vaadatakse ning kinnitatakse manuaalselt *Google Play* või *App Store* poolt. (*App Store* puhul võib see aega võtta 24h, *Google Play* poe puhul 2h).
8) Lõppkasutajal on võimalik Handies rakenduse uuendus endale alla laadida, kas *Google Play* või *App Store* keskkonnas.



Joonis 9. Handies ettevõtte rakenduse lansseerimise protsess

5.3 Konfiguratsioonifaili *codemagic.yaml pipeline* tutvustus

Autor tutvustab ***codemagic.yaml*** *pipeline* konfiguratsioonifaili, mis on kohandatud täpselt Handies ettevõtte hetkevajadusi arvesse võttes. Handiese rakenduste lähtekoodi hoitakse *Gitlab*-is. *Codemagic*-u kasutamiseks on vaja esmalt konfigureerida rakenduse *pipeline* konfiguratsioonifail *codemagic.yaml* ning see üles laadida rakenduse juurkataloogi Gitlabis. See fail sisaldab käsklusi, koodi üleslaadimine kindlasse (*release branch*) *git* harusse käsuga `git push`.

Antud ettevõttes kasutusele võetud *codemagic.yaml* fail *on ära toodud* ***Lisa 3.***

Codemagic-u konfiguratsioonifailis ***(codemagic.yaml)*** on ära defineeritud eraldi ettevõtte *React Native* rakenduste töövood (*workflows*). Alustatud on kliendirakendusest (*Client)*, ära kirjeldatud kliendirakenduse Android rakenduse ehitus *(build)* ning rakenduse üleslaadimiseprotsess *(upload) Google Play* poodi. Ning kliendirakenduse IOS rakenduseehitus *(build)* ning üleslaadimisprotsess *(upload) App Store* keskkonda.

Järgnevalt on eraldi kirjeldatud teenusepakkuja *(Tasker)*rakenduse töövoogu *(workflow)*. Alustatud Android rakenduseehituse ja üleslaadimisprotsessiga *Google Play* poodi. Ning lõpetades IOS rakenduse rakenduseehituse ja üleslaadimisprotsessiga *App Store*-i.

5.4 Kliendirakenduse (*Client)* Android-i ja IOS skriptide seletused

5.4.1 Kliendirakenduse (*Client*) Android skriptid

```
scripts:
- name: Prepare Client app for Build
```
(**skripti osa nimi:** Android-i kliendirakenduse rakenduse ettevalmistus)
```
script: |
```

- **cd ./client-app** (kataloogi muutmine *(change directory*), liikumine Handies kliendirakenduse *(client-app)* juurkausta. [49]

- **npm install --global *Expo-cli*** *Expo* CLI käsurea rakenduse installeerimine globaalselt [50], [51]

- **yarn** (alias/lühend käsklusele *'yarn install'*, selle käigus installeeritakse kõik välised teegid (*library-d*), mis on projekti poolt kasutatud ning defineeritud ***package.json*** failis, alternatiiv npm-ile (*node package manager*), Handies arendajate poolt rohkem eelistatud kui npm oma parema töökindluse tõttu. [52]

- **wait -** Unix käsk, ootab kuni eelmine käsk on lõpetanud. [53]

- **cd ./app/components/cross-app && git pull origin master**

(see käsk liigub eraldi git alammoodulina käsitletavale rakenduse globaalsete komponentide “ kausta ning ütleb süsteemile, et „tõmba alla Gitlab rakenduse repositooriumust“ *origin master* harust (branch) enda arvutisse *cross-app* rakenduse komponendid) [49]

```
- name: Build Client Android
```
(**skripti osa nimi:** Android-i kliendirakenduse rakenduseehitus)
```
script: |
```

- **cd ./client-app** – kataloogi muutmine *(change directory)*, liikumine handies kliendirakenduse *(client-app)* juurkausta [49]

- `echo $SERVICE_ACCOUNT | base64 --decode > service-account.json` – See käsk dekodeerib *env variables* muutuja $SERVICE_ACCOUNT all oleva service-account.json faili ning salvestab selle faili nimega *service-account.json* (mida hiljem kasutatakse *Expo upload* käsu poolt sisendina)

- **npx *Expo* login --username $*EXPO*_USERNAME --password $*EXPO*_PW -** see käsklus teostab sisse logimise *Expo* serverisse (kuna *Expo* server nõuab kasutajakontot on Handies ettevõtte arendajal see ka olemas), seega ka see käsklus

- ***Expo* build:android -t app-bundle --release-channel ci-test --non-interactive**

(see käsk käivitab rakenduse ehituse *Expo* serveris ning ehitab valmis paketifaili, mis sisaldab ***.aab*** Android rakenduse failitüüpi, *Expo* rakenduse *release-channelit* nimega 'ci-test', mis eristab arendaja samanimelisi, kuid erinevates keskkondades loodud rakendusi *Expo* serverites.
--non-interactive on CI puhul kasutatav *tag*, ehk käsk, mis ei küsi kasutaja sisendit

- name: **Upload client Android (skripti osa nimi**: Android-i kliendirakenduse üleslaadimine *Google Play* keskkonda)
script: |

- **cd ./client-app -** kataloogi muutmine (*change directory*), liikumine handies kliendirakenduse (client-app) juurkausta [49]
- ***Expo* upload:android --latest --type aab --key service-account.json --track internal -** *Expo* abil kõige viimasena genereeritud ***.aab*** kliendirakenduse *(Client)* üles laadimine Google keskkonda, **--track internal** määrab *Google Play* sisese keskkonna, kuhu fail üles laetakse (*internal* testimine, *beta* testimine, *production* vms) tüübi määramine ***(aab).*** Lisaks antakse käsule kaasa ***service-account.json*** fail, mis sisaldab endas vajalikku infot, et anda *Expo*le ligipääs faili üles laadimiseks vastavale *Google Play* kontole. [54]

5.4.2 Kliendirakenduse *(Client)* IOS skriptide seletus

- name: **Build and upload customer iOS app** (**skripti osa nimi:** IOS-i kliendirakenduse rakenduseehitus, üleslaadimine)
script: |

- **cd ./client-app** – kataloogi muutmine *(change directory*), liikumine handies kliendirakenduse *(client-app)* juurkausta. [49]
- **sudo gem install fastlane** – Fastline käsurealiidese installeerimine [55]
- **npx *Expo* login --username $*EXPO*_USERNAME --password $*EXPO*_PW** see käsklus teostab sisse logimise *Expo* serverisse (kuna *Expo* server nõuab kasutajakontot on Handies ettevõtte arendajal see ka olemas), seega ka see käsklus [56]
- ***Expo* build:ios --release-channel ci-test --non-interactive** – see käsk käivitab rakenduse ehituse *Expo* serveris ning ehitab valmis pakki mis sisaldab **.**IOS rakenduse failitüüpi gitlab haru (*branch*) nimega

–release-channel, ning lisaks on *ci test—non-interactive* mis on *Google Play* testkeskkonna nimi [57]

- **`curl -o app.ipa "$ (Expo url:ipa --non-interactive)"`**

See käsurida sisaldab kahte käsku ühe sees:

1) sulgudes olev käsk (*Expo url:ipa --non-interactive)* -> küsib *Expo* serverilt viimase genereeritud ***.ipa*** faili URL-i (ehk see mis eelmise käsureaga sai tehtud)

2) *curl -o app.ipa (*IPA_URL) -> tõmbab URL'i pealt ***.ipa*** faili ja salvestab selle failisüsteemi **app*.ipa*** nime all. [58]

- **`fastlane pilot upload --username $FASTLANE_USER --ipa "app.ipa" -`** Fastlane otsib ise automaatselt arendaja poolt defineeritud keskkonnamuutujate seast vajalikud fastlane muutujad, mida kasutab Apple'i serveritega autentimiseks, et saada ligipääs rakenduse üles laadimiseks rakenduse omaniku Apple'i kontole. [59]

Kuna rakendused on mõlemad sarnaselt üles ehitatud, on ka teenusepakkuja *(Tasker)* rakenduse ehitusprotsess *(build)* ja üleslaadimise protsess identsed kliendirakendusega *(Client)*.

6 Tulemuste analüüs

Autori hinnangul on peatükis 2.4 püstitatud probleem ning lähteülesanne saanud töö jooksul suures osas lahendatud. Automaatsest juurutusprotsessist on välja jäänud rakenduste testimiskeskkonda *(Staging)* saatmise haru automatiseerimine, kuna hetkel kasutavad suurem osa ajast arendajad testrakenduste uuendamiseks *OTA* uuendusi, mille ajakulu on võrreldes rakendusefaili ehitamisega marginaalne ning ei vajanud antud hetkel ettevõtte arendajate sõnul automatiseerimist.

Töö käigus koostöös Handies ettevõttega seadis autor üles *Codemagic*-u ettevõtte poolt pakutava *CI/CD pipeline* automatiseerimise tööriista mobiilirakenduste rakenduse ehituse automatiseerimiseks ning lansseerimiseks *Google Play* ja *App Store* keskkondadesse. Esmalt tuli *Codemagic*-u tööriist siduda Handies ettevõtte mobiilirakenduste repositooriumiga, mis asub *Gitlab*-is. See võttis autoril maksimaalselt 10 minutit aega. Järgmiseks tuli koostada kahe mobiilirakenduse töövoogude *(workflows) CI/CD pipeline* konfiguratsioonifail *(codemagic.yaml)* automaatseks lansseerimiseks *Google Play* ja *App Store*-i keskkondadesse. Konfiguratsioonifaili *codemagic.yaml* loomist alustati kõigepealt ettevõtte Android kliendirakenduse töövoo koostamisest. Et alguses testimiseks kõige lihtsama automaatne juurutus *Google Play* keskkonda tööle saada. Pärast seda loodi sinna kõrvale iOS kliendirakenduse töövoo *(workflow)* automaatne juurutus *App Store*-i. Kui kliendirakenduse nii iOS kui Andorid automaatne rakenduse ehitusprotsess ning lansseerimine *App Store* ja *Google Play*sse oli valmis saanud, siis koostati analoogselt ka Teenusepakkuja iOS ja Androidi rakenduste töövoog *(workflow)* automaatse rakenduse ehituse ja juurutusprotsessi jaoks. Kui konfiguratsioonifail oli valminud tuli see üles laadida ettevõtte rakenduste juurkataloogi mis asub *Gitlabis*.

Codemagic konfiguratsioonifaili *codemagic.yaml* koostamiseks ning ka tööle saamiseks kulus autoril koostöös ettevõttega kaks tööpäeva. Mis oli ka *CI/CD pipeline* mobiilirakenduste automatiseerimise juures kõige keerulisem.

6.1 Ajakulu

Järgnevalt on võrreldud ajakulu mis kulus enne Handies ettevõtte arendajal rakenduste manuaalseks ehituseks ja juurutamiseks nii *Google Play* ja *App Store* keskkondadesse, sellega mis kulub pärast Codemagic-u automatiseerimistarkvara kasutuselevõttu. Automatiseeritud protsessiga pärast Codemagic tööriista kasutuselevõttu on võimalik tutvuda joonisel 9.

Eelnevalt kulus Handies ettevõtte arendajal ühe rakenduse ja ühe platvormi rakenduse ehituseks ja manuaalseks lansseerimiseks **umbes 30-45 minutit.** (manuaalse lansseerimise protsess on välja toodud peatükis 2.3). Kliendirakenduse nii *Android-i* kui ka i*OS* rakenduse ehituseks kui ka lansseerimiseks (*release*) kulus eelnevalt **umbes 1h,** halvemal juhul võis kuluda isegi **1h30min.** Teenusepakkuja rakenduse lansseerimise ajakulu on võrdväärne kliendirakenduse ajakulu näitega.

Seega võttis nii kliendirakenduse kui ka teenusepakkuja iOS ja Android rakenduste lansseerimise manuaalne protsess *App Store* ja *Google Play* keskkondadesse ~~võttis~~ ettevõtte arendajal kokku ligikaudu **2h.** Siinkohal peab ära mainima, et uuenduste tegemine on väga varieeruv ning see toimub vastavalt vajadusele. Vahel tehakse uuendusi kord kuus, vahel kaks korda nädalas. Kui ettevõtte arendajal juhtumisi oli vaja sooritada nädalas kaks rakenduse uuendust kliendi kui ka teenusepakkuja (iOS ja Android), siis see võis koguni aega võtta kokku **4h** arendaja väärtusliku aega mida ta oleks saanud kasutada olulistema ülesannetega tegelemiseks.

Pärast *Codemagic*-u *CI/CD pipeline* automatiseerimise tööriista kasutuselevõttu võttis rakenduse ehitus ja rakenduse üleslaadimine *App Store*-i ning *Google Play* poodi kokku aega 1h16min. (vt. *Lisa 6.)* Ehk ühe iOS või Androidi rakenduse lansseerimise automatiseeritud protsess võttis eraldi aega umbes **37.5 minutit,** võrreldes manuaalse juurutusega, kui arendajal kulus selleks **30-45 minutit** oli ühe mobiilirakenduse platvorm rakenduse ehituse jaoks kulunud arendaja tööaeg. See protsess tervikuna toimus automaatselt Codemagic serveris, st. et selle aja sees kulus arendajal reaalset tööaega vaid paar minutit kui ta pidi sooritama (*git push branchi)* käivituse.

Võrreldes manuaalset rakenduse juurutusprotsessi automatiseeritud rakenduse juurutusprotsessiga saab järeldada seda, et kui arvestada sellega, et keskmiselt kulus arendajal 1h15minutit manuaalse rakenduse ehituse ja lansseerimise peale vastavasse keskkonda, siis peale protsessi automatiseerimist kulus arendajal selleks reaalset tööaega 5 minutit (*release branchi* loomide, versioonide muutmine). Seega on arendaja ajavõit 1h15 min - 5 min = 1h10minutit. Seega ajavõit 1h16 - 5minutit = 1h11minuti, ümardatult 1h10 minutit. Ning kliendi kui ka teenusepakkuja, mõlema rakenduse väljalaske ajavõit juba 2h 20minutit.

Lisaks rakenduseehitusele ja automaatsele lansseerimise protsessile *Google Play* ja *App Store* poodidesse väheneb uute arendajatele rakenduse ehituse (*build*) protsessi õpetamisele kuluv aeg ja vastavate ligipääsude andmine, ning ettevõtte arendajad saavad keskenduda olulistema ülesannete sooritamisele.

6.2 Tulevikulahendusi

Esmalt võiks Handies ettevõtte kaaluda *CI/CD pipeline*-i juurde implementeerida rakenduse versiooninumbri automaatse suurendamise (*app.json*) failis, kuna kui on soov uus lansseerimine teha *Google Play* või *App Store* keskkonda peab iga kord arendaja manuaalselt suurendama rakenduse versiooninumbrit.

Praegune mobiilirakenduste rakenduseehitus ning lansseerimine toimub eraldiseisvas *Expo* serveris. See on halb seetõttu, et *Expo*l on enda järjekord, mis sõltub hetkel *Expo* servereid kasutavate arendajate arvust. Järgnevalt võiks ettevõtte kaaluda *Expo* serveri loobumisest.

***Expo*st serveri loobumiseks on kaks võimalust:**

- Esimeseks võimaluseks on ettevõttel endal üles seada arendusprotsessi kiirendav automatiseeritud *pipeline* kasutades selleks näiteks *Turtle CI'd*, et võita *Expo* serverite ooteaja arvelt (15-30minutit), mis tuleneb teiste *Expo* kasutajate hulgast rakenduse ehituse (*build*-i) loomise hetkel.

- Teiseks võimaluseks on ettevõttel kasutada rakenduse ehitamisel Codemagic poolt ette valmistatud *React Native* rakenduse ehituse protsessi ning rakenduseehitus *Expo* keskkonnast tuua üle täielikult *Codemagic*-u keskkonda. Lisaks automatiseerida

teekond rakenduse testkeskkonda (*Firebase App Distributor* ja Testflight) saatmiseks, et Handies ettevõtte testijad saaksid sealt selle kätte ning oleksid võimelised seda rakendust kohe testima asuma ning ettevõtte arendajatele testtulemustest tagasisidet andma.

7 Kokkuvõte

Käesoleva lõputöö eesmärk oli leida lahendus, mis hoiaks kokku Handies ettevõtte arendaja(te) aega, mis kulub rakenduste uuenduste/paranduste üleslaadimiseks *Google Play* või *App Store* keskkonda ning rakenduste uuenduste jõudmist klientideni.

Töö esimeses osas tutvustas autor start-up ettevõtet nimega Handies ning lahendavat probleemi ning defineeris lähteülesande. Lisaks tutvustas autor turul pakutavaid *CI/CD* arendusprotsessi automatiseerimise tööriistu, koostas eelnevalt väljatoodud tööriistadest võrdluse. Töö teises osas analüüsiti erinevaid automatiseerimise tööriistu ning lähtudes ettevõtte vajadustest valiti välja *Codemagic*-u automatiseerimise tööriist, mida siis praktiliselt implementeeriti antud ettevõttes. Töö tulemuste osas võrdles autor ajakulu, mis kulus enne ettevõtte manuaalselt rakenduste lansseerimiseks *Google Play* ning *App Store* keskkondadesse ajakuluga, mis saavutati pärast *CI/CD* arendusprotsess automatiseerimist. Lisaks pakkus autor välja tulevikulahendusi, mida ettevõte võiks realiseerida, kuid mis praegusest implementatsioonist välja jäid.

Autori hinnangul vastab bakalaureusetöö tulemus suures osas töö alguses püstitatud lähteülesandele ning ettevõtte poolt seatud kriteeriumitele. Töö käigus juurutatud lahendus võimaldab ettevõttel säästa arendajate väärtuslikku tööaega, mis kulus enne manuaalselt rakenduse ehituse ning lansseerimise peale *Google Play* ning *App Store* keskkonda. Lisaks on tänu töö analüüsi osas olevale võrdlusele võimalik mõista millist lahendust oleks mõistlik juurutada, kui on soov automatiseerida mobiilirakenduse *CI/CD* lantseerimise arenduseprotsess.

Kasutatud kirjandus

[1] „Handies ettevõtte tutvustus,“ [Võrgumaterjal]. URL: https://handies.com/meist/. [Kasutatud 17.05.2021].

[2] „React Native tutvustus,“ 17 05 2021. [Võrgumaterjal]. URL: https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html.

[3] „Expo rakenduseehituse tutvustus,“ 17 05 2021. [Võrgumaterjal]. URL: https://docs.Expo.io/.

[4] „Gitlab-i tutvustus,“ 17 05 2021. [Võrgumaterjal]. URL: https://www.simplilearn.com/tutorials/git-tutorial/what-is-Gitlab.

[5] „Ota uuenduse juhend,“ 17 05 2021. [Võrgumaterjal]. URL: https://docs.Expo.io/guides/configuring-ota-updates/.

[6] „What is Gitlab,“ [Võrgumaterjal]. URL: https://www.simplilearn.com/tutorials/git-tutorial/what-is-Gitlab. [Kasutatud 01.05.2021].

[7] „Blog: Best 14 CI/CD Tools You Must Know,“ [Võrgumaterjal]. URL: https://www.katalon.com/resources-center/blog/ci-cd-tools/. [Kasutatud 12.05.2021].

[8] H. Sheth, „Blog: 31 of The Top CI/CD Tools Available Today,“ 15 10 2020. [Võrgumaterjal]. URL: https://www.lambdatest.com/blog/31-best-ci-cd-tools/. [Kasutatud 15.05.2021].

[9] E. Gift, „Buddy vs. Travis CI: A Detailed Comparison,“ [Võrgumaterjal]. URL: https://medium.com/the-devops-corner/buddy-vs-travis-ci-a-detailed-comparison-8be02266876a. [Kasutatud 17.05.2021].

[10] „Top Mobile Continuous Integration (CI/CD) Tools,“ 05 01 2021. [Võrgumaterjal]. URL: https://instabug.com/blog/continuous-integration-tools/. [Kasutatud 17.05.2021].

[11] A. Altvater, „Top Continuous Integration Tools: 51 Tools to Streamline Your Development Process, Boost Quality, and Enhance Accuracy,“ 26 03 2017. [Võrgumaterjal]. URL: https://stackify.com/top-continuous-integration-tools/. [Kasutatud 17.05.2021].

[12] „20 Best Continuous Integration(CI/CD) Tools in 2021,“ [Võrgumaterjal]. URL: https://www.guru99.com/top-20-continuous-integration-tools.html. [Kasutatud 16.05.2021].

[13] „Bitrise koduleht,“ [Võrgumaterjal]. URL: https://www.bitrise.io/why/technologies/ios-continuous-integration. [Kasutatud 09.05.2021].

[14] „Codemagic,“ [Võrgumaterjal]. URL: https://www.capterra.com/p/152573/Nevercode/. [Kasutatud 12.05.2021].

[15] M. Jaksman, „Best Jenkins alternatives. Top CI/CD Tools for your Android and iOS projects,“ 22 08 2018. [Võrgumaterjal]. URL: https://hackernoon.com/top-ci-cd-tools-for-your-android-and-ios-projects-8d356b983b3b. [Kasutatud 12.05.2021].

[16] „Overview,“ [Võrgumaterjal]. URL: https://CircleCI.com/docs/enterprise/overview/. [Kasutatud 12.05.2021].

[17] „Best 14 CI/CD Tools You Must Know | Updated for 2021,“ [Võrgumaterjal]. URL: https://www.katalon.com/resources-center/blog/ci-cd-tools/. [Kasutatud 17.05.2021].

[18] „What Is GitHub Action and How Do They Help,“ [Võrgumaterjal]. URL: https://medium.com/swlh/what-is-github-action-and-how-do-they-help-c8b254118fa5. [Kasutatud 14.05.2021].

[19] „GitHub Actions Website,“ [Võrgumaterjal]. URL: https://docs.github.com/en/actions. [Kasutatud 14.05.2021].

[20] „Visual Studio homepage,“ [Võrgumaterjal]. URL: https://visualstudio.microsoft.com/app-center/faq/. [Kasutatud 16.05.2021].

[21] „Visual Studio App Center,“ [Võrgumaterjal]. URL: https://visualstudio.microsoft.com/app-center/. [Kasutatud 16.05.2021].

[22] „Appcircle Homepage,“ [Võrgumaterjal]. URL: https://appcircle.io/. [Kasutatud 16.05.2021].

[23] „What is Azure DevOps,“ [Võrgumaterjal]. URL: https://www.devopsgroup.com/insights/resources/tutorials/all/what-is-azure-devops/. [Kasutatud 16.05.2021].

[24] B. Patterson, „20 Best Continuous Integration Tools: A Guide to Optimizing Your CI/CD Processes,“ [Võrgumaterjal]. URL: https://www.threatstack.com/blog/20-best-continuous-integration-tools-a-guide-to-optimizing-your-ci-cd-processes. [Kasutatud 16.05.2021].

[25] „Azure DevOps: The Next Big Thing in Application Lifecycle Management,“ [Võrgumaterjal]. URL: https://www.simplilearn.com/azure-devops-article. [Kasutatud 16.05.2021].

[26] E. Gift, „Buddy vs. Travis CI: A Detailed Comparison,“ [Võrgumaterjal]. URL: https://medium.com/the-devops-corner/buddy-vs-travis-ci-a-detailed-comparison-8be02266876a. [Kasutatud 16.05.2021].

[27] „Blog: CI/CD for mobile apps: the complete guide,“ [Võrgumaterjal]. URL: https://blog.Codemagic.io/the-complete-guide-to-ci-cd/#choosing-between-self-hosted-and-cloud-based-cicd. [Kasutatud 12.05.2021].

[28] „Blog: Top Mobile Continuous Integration (CI/CD) Tools,“ [Võrgumaterjal]. URL: https://instabug.com/blog/continuous-integration-tools/. [Kasutatud 16.05.2021].

[29] „Crunchbase homepage,“ [Võrgumaterjal]. URL: https://www.crunchbase.com/organization/buddy-llc. [Kasutatud 16.05.2021].

[30] „Appcircle,“ [Võrgumaterjal]. URL: https://www.capterra.com/p/204535/Appcircle/. [Kasutatud 16.05.2021].

[31] „Best 11 CI/CD Services for building React Native Project iOS & Android,“ [Võrgumaterjal]. URL: https://dev.to/retyui/best-11-ci-cd-services-for-building-react-native-project-ios-android-534h. [Kasutatud 16.05.2021].

[32] „Gitlab Homepage,“ [Võrgumaterjal]. URL: https://about.gitlab.com/pricing/. [Kasutatud 15.05.2021].

[33] „Buddy Homepage,“ [Võrgumaterjal]. URL: https://buddy.works/pricing. [Kasutatud 15.05.2021].

[34] „Bitrise Homepage,“ [Võrgumaterjal]. URL: https://www.bitrise.io/pricing. [Kasutatud 16.05.2021].

[35] „Codemagic Homepage,“ [Võrgumaterjal]. URL: https://codemagic.io/pricing/. [Kasutatud 15.05.2021].

[36] „Circle Ci Homepage,“ [Võrgumaterjal]. URL: https://circleci.com/pricing/?utm_source=gb&utm_medium=SEM&utm_campaign=SEM-gb-200-Eng-ni&utm_content=SEM-gb-200-Eng-ni-CirclePricing&utm_term=a2&gclid=CjwKCAjwqIiFBhAHEiwANg9sznHQu_Wgjtr_Y_iGs5K_MT_Kdo6U4QbUPu876WcG6GCledzriuvBXxoChmkQAvD_BwE. [Kasutatud 16.05.2021].

[37] „Github Actions Homepage,“ [Võrgumaterjal]. URL: https://github.com/features/actions. [Kasutatud 16.05.2021].

[38] „Visual Studio Homepage,“ [Võrgumaterjal]. URL: https://visualstudio.microsoft.com/app-center/pricing/. [Kasutatud 17.05.2021].

[39] „Appcircle Homepage,“ [Võrgumaterjal]. URL: https://appcircle.io/pricing/. [Kasutatud 16.05.2021].

[40] „Microsoft Azure Homepage,“ [Võrgumaterjal]. URL: https://azure.microsoft.com/en-us/services/devops/pipelines/. [Kasutatud 15.05.2021].

[41] „Travis CI Homepage,“ [Võrgumaterjal]. URL: https://travis-ci.com/plans. [Kasutatud 16.05.2021].

[42] „Using codemagic.yaml,“ [Võrgumaterjal]. URL: https://docs.codemagic.io/getting-started/yaml/. [Kasutatud 12.05.2021].

[43] „Publishing and deployment,“ [Võrgumaterjal]. URL: https://docs.codemagic.io/publishing-yaml/distribution/. [Kasutatud 16.05.2021].

[44] „Building a React Native app,“ [Võrgumaterjal]. URL: https://docs.codemagic.io/getting-started/building-a-react-native-app/. [Kasutatud 17.05.2021].

[45] „npm Docs,“ [Võrgumaterjal]. URL: https://docs.npmjs.com/cli/v6/commands/npm. [Kasutatud 15.05.2021].

[46] „Expo CLI,“ [Võrgumaterjal]. URL: https://docs.expo.io/workflow/expo-cli/. [Kasutatud 15.05.2021].

[47] „Fastline Docs,“ [Võrgumaterjal]. URL: https://docs.fastlane.tools/. [Kasutatud 16.05.2021].

[48] „Difference between npm and yarn,“ [Võrgumaterjal]. URL: https://www.geeksforgeeks.org/difference-between-npm-and-yarn/. [Kasutatud 16.05.2021].

[49] „Cd Command in Linux,“ [Võrgumaterjal]. URL: https://linuxize.com/post/linux-cd-command/#:~:text=The%20cd%20%28%E2%80%9Cchange%20directory%E2%80%9D%29%20command%20is%20used%20to,in%20which%20the%20user%20is%20currently%20working%20in.. [Kasutatud 16.05.2021].

[50] „npm Docs,“ [Võrgumaterjal]. URL: https://docs.npmjs.com/cli/v6/commands/npm-install. [Kasutatud 16.05.2021].

[51] „Expo CLI,“ [Võrgumaterjal]. URL: https://docs.expo.io/workflow/expo-cli/. [Kasutatud 16.05.2021].

[52] „CLI Introduction,“ [Võrgumaterjal]. URL: https://classic.yarnpkg.com/en/docs/cli/. [Kasutatud 16.05.2021].

[53] „Wait Command in Linux,“ [Võrgumaterjal]. URL: https://linuxhint.com/wait_command_linux/ . [Kasutatud 16.05.2021].

[54] „Uploading Apps to the Apple App Store and Google Play,“ [Võrgumaterjal]. URL: https://docs.expo.io/distribution/uploading-apps/. [Kasutatud 16.05.2021].

[55] „Getting started with fastlane for iOS,“ [Võrgumaterjal]. URL: https://docs.fastlane.tools/getting-started/ios/setup/. [Kasutatud 16.05.2021].

[56] „Setting up Continuous Integration,“ [Võrgumaterjal]. URL: https://docs.expo.io/guides/setting-up-continuous-integration/. [Kasutatud 15.05.2021].

[57] P. Leja, „How to build an iOS standalone app with Expo on Windows and Linux?,“ 29 07 2020. [Võrgumaterjal]. URL: https://tdcm.io/blog/building-an-iOS-standalone-app-with-Expo-on-Windows-and-Linux#:~:text=%20%20%201%20The%20last%20step%20to,Distribution%20Certificate.%20Choose%20%E2%80%9CI%20want%20to...%20More%20. [Kasutatud 15.05.2021].

[58] „Curl command in Linux with Examples,“ [Võrgumaterjal]. URL: https://www.geeksforgeeks.org/curl-command-in-linux-with-examples/#:~:text=curl%20command%20in%20Linux%20with%20Examples.%20curl%20is,LDAP%20or%20FILE%29.%20curl%20is%20powered%20by%20Libcurl.. [Kasutatud 15.05.2021].

[59] „Fastline docs,“ [Võrgumaterjal]. URL: https://docs.fastlane.tools/actions/pilot/. [Kasutatud 13.05.2021].

[60] „Codemagic Documentation,“ [Võrgumaterjal]. URL: https://docs.codemagic.io/. [Kasutatud 16.05.2021].

[61] „Using codemagic.yaml,“ [Võrgumaterjal]. URL: https://docs.codemagic.io/getting-started/yaml/. [Kasutatud 16.05.2021].

[62] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

[63] „Automate your workflow,“ [Võrgumaterjal]. URL: https://github.com/features/actions. [Kasutatud 17.05.2021].

[64] „An Introduction to Github Actions,“ [Võrgumaterjal]. URL: https://gabrieltanner.org/blog/an-introduction-to-github-actions. [Kasutatud 16.05.2021].

Lisa 1 - Mobiilirakenduse CI/CD pipeline automatiseerimisel kasutatavad metoodikad, tarkvarad ja keskkonnad

- Continuous delivery
- Continuous integration
- *Gitlab* – version control and pipelines
- App *Distributor* – Android app distribution for testing network
- TestFlight – iOS app distrubution for testing network
- macServer – for *build*ing IPA files automatically
- YAML format for configurations
- **Cross-Platform Mobile (IOS, Andorid):** React-Native, *Expo*
- **Back-end:** PHP, Symphony, Kubernetes (K8S), NodeJS (for chat and notification service), MySQL (for core), MongoDB (for chat)
- **DevOps:** Google Cloud Platform, Google SQL, Google Kubernetes licence, Google Storage, Amazon Web Services (AWS), S3,- RabbitMQ, Redis

Lisa 2 - *Codemagic*u paigaldamise kasutusjuhend

Antud juhend õpetab Handies ettevõtte juhtivarendajat/tehnoloogiajuhti kuidas enda koodiprojekt ühendada *Codemagic CI/CD* automaattööriistaga.

1. Avada leht nimega ***https://codemagic.io/start/***

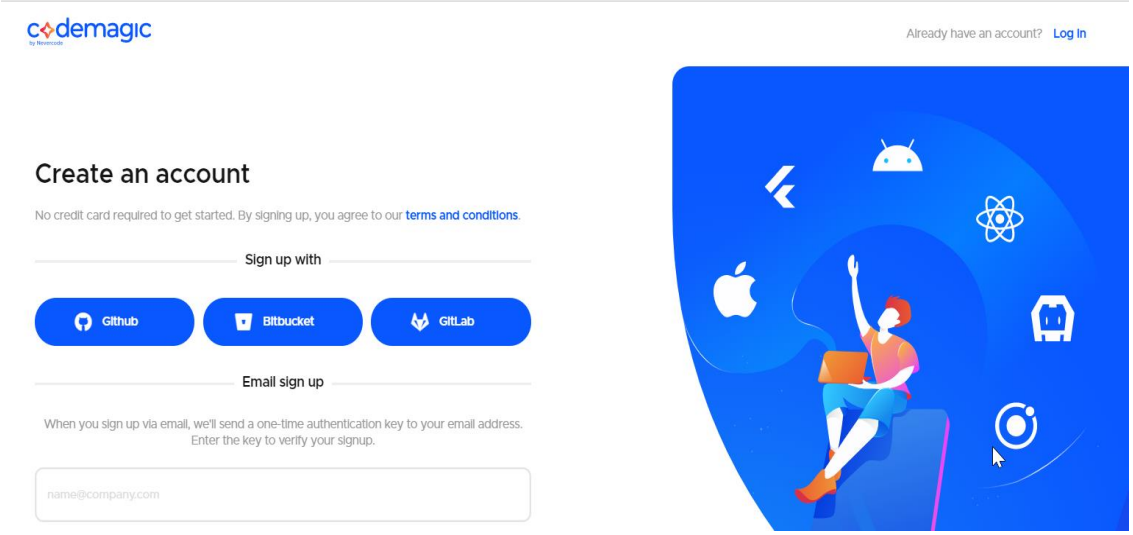
2. Valige paremalt üleval nurgast *"Sign up"*

3. Siin on kaks võimalust *Codemagic*u konto registeerimiseks:

   3.1. Registeerida konto *Gitlab*-iga

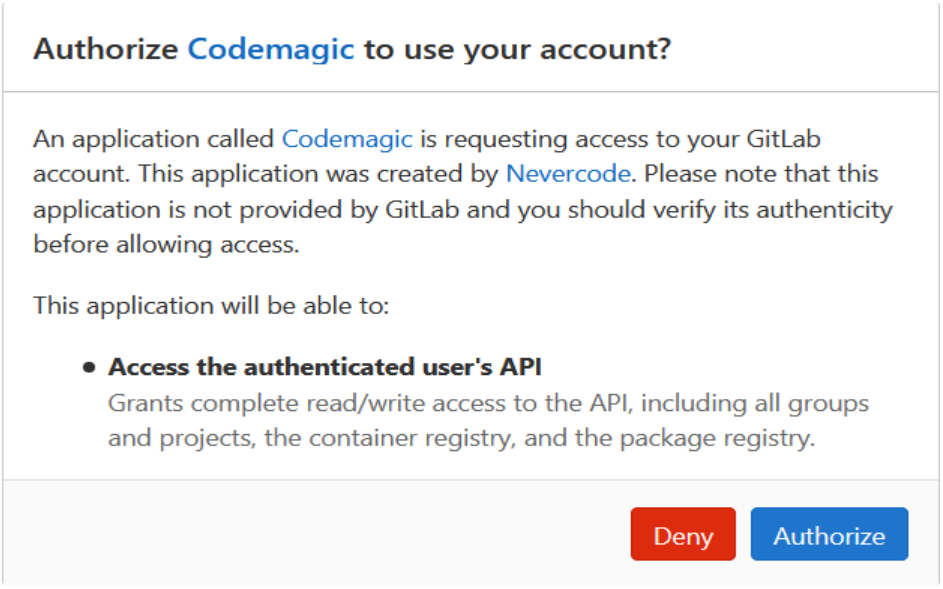
   või

   3.2. Registeerida konto emaili aadressiga [60]



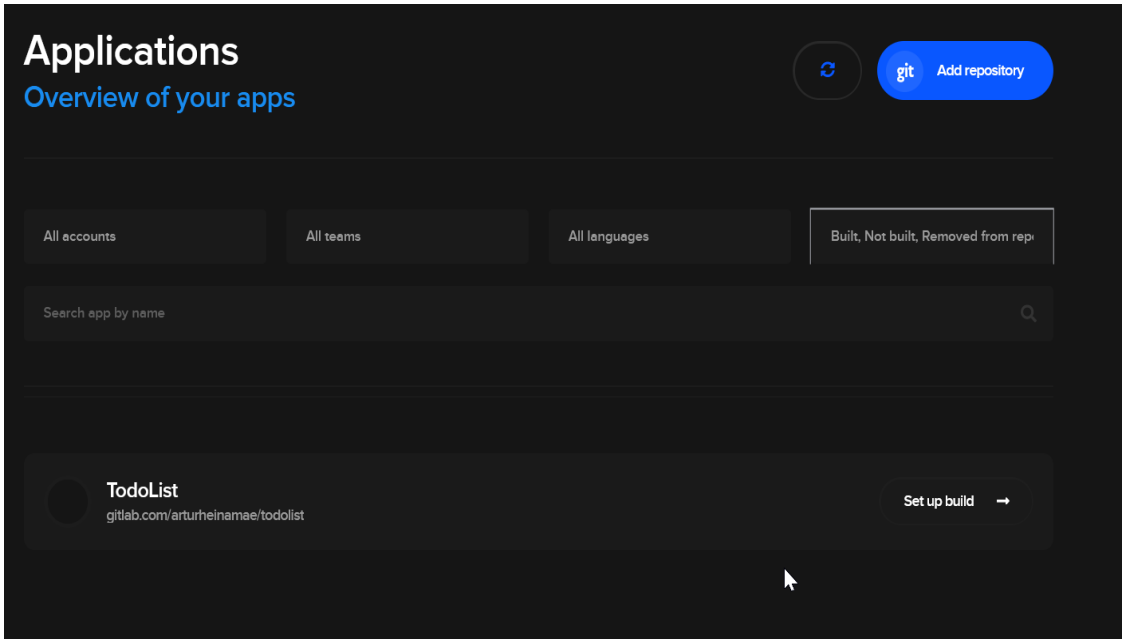
Valisin esimese variandi (*Gitlab*iga registeerimise kuna Handies ettevõtte hoiustab rakenduse koodi *Gitlabis*).

3.3. Valides registeerimise *Gitlab*-iga, suunatakse edasi *Gitlab*-i, edasi tuleb valida **"Authorize",** ning sellega toimub *Gitlab-i* ja *Codemagic*u sidumine.

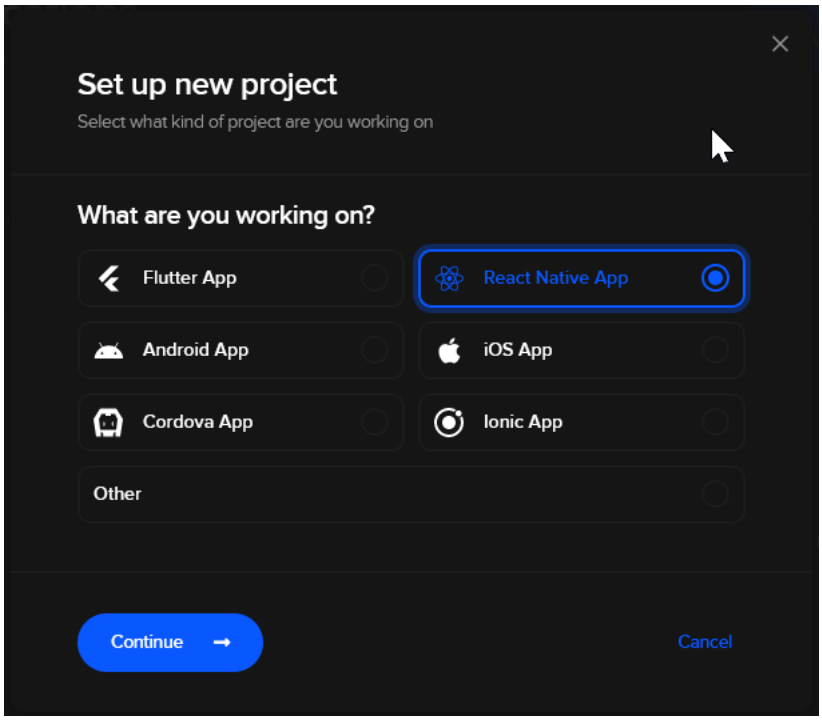


3.4. Rakenduse **(*Applications*) aknas**

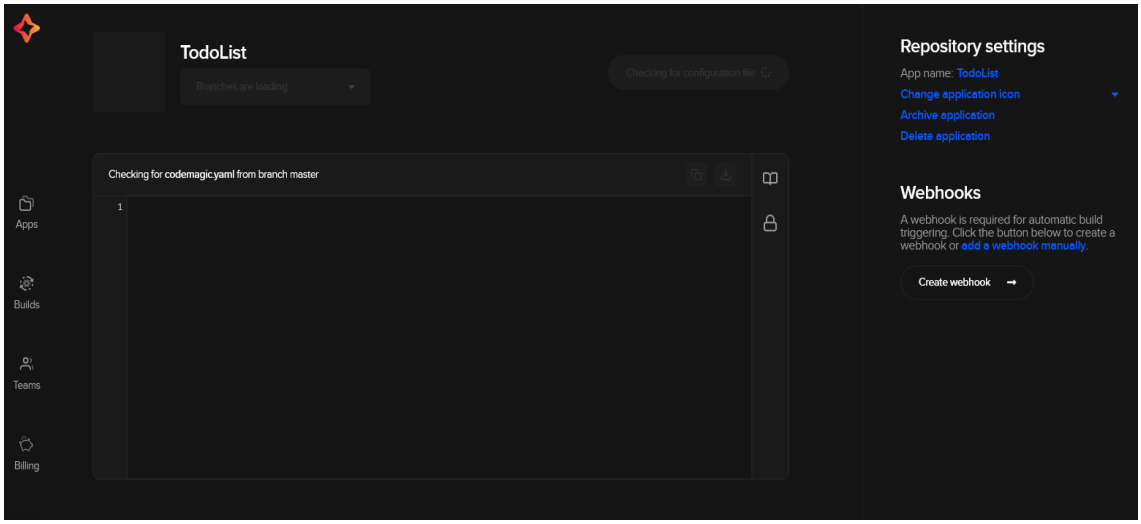
Kõik kontod (*All accounts)*, kõik tiimiliikmed (*All teams*), kõik keeled *(All languaeges*)-- Olles valinud antud aknast endale sobivad valikud -- vali ***"Set up build"*** [60]



3.5. Hüpikaknas tuleb valida ***React Native*** **App** --ning *"Continue"* [60]



3.6. Pärast projekti koodi sidumist *Codumagicuga* on vaja konfigureerida rakenduse ehituse *(build)* fail. Rakenduse ehituse (*Build)* failiks on ***codemagic.yaml*** fail.



Kui rakenduse ehituse *(build)* **(*codemagic.yaml*)** konfiguratsioonifail on valmis saadud ning see on üles laetud *Gitlab*-i rakenduse repositooriumisse, siis muutub see automaatselt nähtavaks antud aknasse (pildil). [61]

Lisa 3 - Handies ettevõtte *Codemagic.yaml* fail

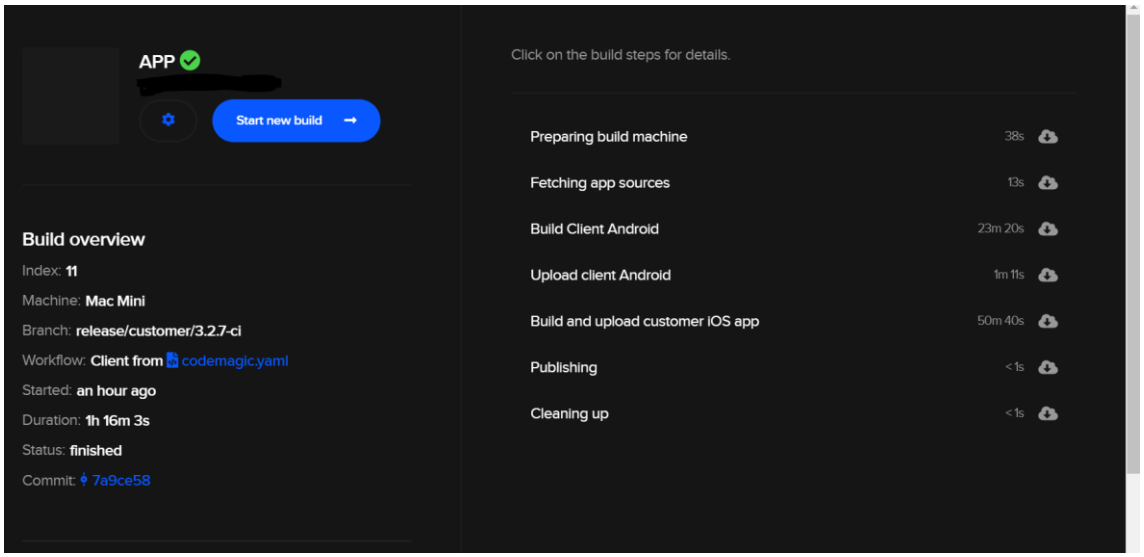
```
workflows:
 react-native-customer:
 name: Client
 max_build_duration: 120
 instance_type: mac_mini
 environment:
 vars:
 SERVICE_ACCOUNT: <service_account.json>
 EXPO_USERNAME: <Expo_username>
 EXPO_PW: <Expo_pw>
 FASTLANE_USER: <appstore_username>
 FASTLANE_PASSWORD: <appstore_pw>
 FASTLANE_APPLE_APPLICATION_SPECIFIC_PASSWORD: <appstore_app_specific_pw>
 FASTLANE_SESSION: <fastlane_session_string>
 FASTLANE_ITC_TEAM_ID: <appstore_team_id>
 node: latest
 cache:
 cache_paths:
 - ~/client-app/node_modules
 - ~/tasker-app/node_modules
 triggering:
 events:
 - push
  branch_patterns:
 - pattern: release/customer/*
 include: true
 source: true
 scripts:
 - name: Prepare Client app for Build
 script: |
 cd ./client-app
 npm install --global Expo-cli
 yarn
 wait
 cd ./app/components/cross-app && git pull origin master
 - name: Build Client Android
 script: |
 cd ./client-app
 echo $SERVICE_ACCOUNT | base64 --decode > service-account.json
 npx Expo login --username $EXPO_USERNAME --password $EXPO_PW
 Expo build:android -t app-bundle --release-channel ci-test --non-
interactive
 - name: Upload client Android
 script: |
 cd ./client-app
 Expo upload:android --latest --type aab --key service-
account.json --track internal
 - name: Build and upload customer iOS app
 script: |
 cd ./client-app
 sudo gem install fastlane
 npx Expo login --username $EXPO_USERNAME --password $EXPO_PW
 Expo build:ios --release-channel ci-test --non-interactive
 curl -o app.ipa "$(Expo url:ipa --non-interactive)"
 fastlane pilot upload --username $FASTLANE_USER --ipa "app.ipa"
```

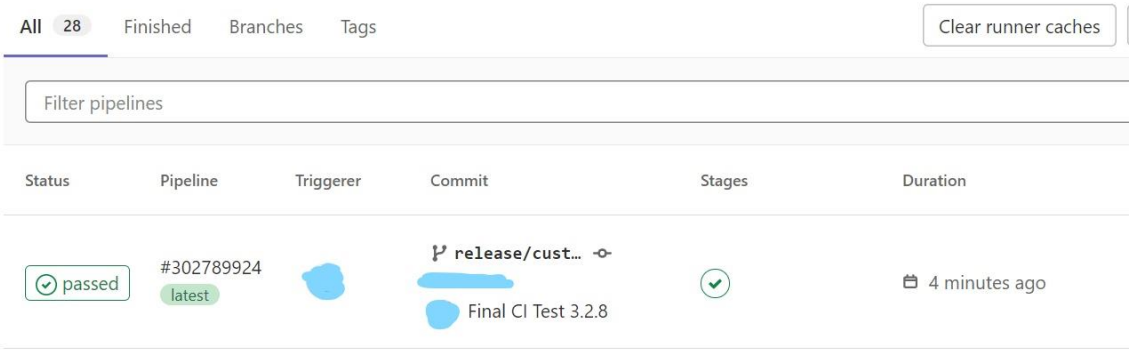
```
react-native-tasker:
name: Tasker
max_build_duration: 120
instance_type: mac_mini
environment:
vars:
SERVICE_ACCOUNT: <service_account.json>
EXPO_USERNAME: <Expo_username>
EXPO_PW: <Expo_pw>
FASTLANE_USER: <appstore_username>
FASTLANE_PASSWORD: <appstore_pw>
FASTLANE_APPLE_APPLICATION_SPECIFIC_PASSWORD: <appstore_app_specific_pw>
FASTLANE_SESSION: <fastlane_session_string>
FASTLANE_ITC_TEAM_ID: <appstore_team_id>
node: latest
triggering:
events:
- push
# - tag
branch_patterns:
- pattern: release/tasker/*
include: true
source: true

scripts:
- name: Prepare Tasker app for build
script: |
cd ./tasker-app
npm install --global Expo-cli
yarn
wait
cd ./app/components/cross-app && git pull origin master
- name: Build and upload Tasker Android
script: |
cd ./tasker-app
echo $SERVICE_ACCOUNT | base64 --decode > service-account.json
npx Expo login --username $EXPO_USERNAME --password $EXPO_PW
Expo build:android -t app-bundle --release-channel ci-test
Expo upload:android --latest --type aab --key service-
account.json --track internal
- name: Build and upload Tasker iOS app
script: |
cd ./tasker-app
sudo gem install fastlane
npx Expo login --username $EXPO_USERNAME --password $EXPO_PW
Expo build:ios --release-channel ci-test --non-interactive
curl -o app.ipa "$(Expo url:ipa --non-interactive)"
fastlane pilot upload --username $FASTLANE_USER --ipa "app.ipa"

publishing:
# See the following link for details about email publishing - https
://docs.Codemagic.io/publishing-yaml/distribution/#email
email:
recipients:
- <email@email.com>
notify:
success: true # To receive a notification when a build succeeds
failure: false # To not receive a notification when a build fails
```

Lisa 4 - Eduka rakenduse pipeline-i joonis





Lisa 5 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Artur-Andres Heinamäe

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Mobiilirakenduste *CI/CD* arendusprotsessi välja töötamine ja juurutamine ettevõttes Handies Solutions OÜ“, mille juhendaja on Jüri Vain
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

18.05.2021

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.