TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Matten Mätlik 178022IABM

ÄRIPROTSESSIDE PARENDAMINE PILVEPÕHISE TESTIMISPLATVORMI ABIL TESTLIO OÜ NÄITEL

magistritöö

Juhendaja: Ants Torim
PhD

Tallinn 2020

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: Matten Mätlik

08.01.2020

Annotatsioon

Pilvetehnoloogiate turg on kasvavas trendis ning 2021 aastaga kasvavad ettevõtete pilvetaristu põhised kulutused 39,5 miljardilt dollarilt 63 miljardile [1]. Ligi 80% ettevõtetest kasutab mingil kujul pilvelahendusi juba täna [2]. Üks sellistest ettevõtetest on globaalsel turul tarkvara testimisteenuseid pakkuv ja Eesti juurtega Testlio OÜ [3].

Lõputöö käigus luuakse terviklik äriline ning tehnoloogiline disain uue pilvepõhise teenuse väljaarendamiseks, võimaldades ettevõtte klientidel saata Testlio süsteemi oma ehitussüsteemidest pärinevaid artefakte. Seeläbi automatiseerides ärikriitilisi protsesse ja tõstes Testlio kui platvormi üldist atraktiivsust. Töö käigus kirjeldatakse põhjalikult vaadeldavat valdkonda, püstitatakse oodatavad funktsionaalsed ning mitte-funktsionaalsed nõuded, sõnastatakse edukuse kriteeriumid ning luuakse struktureeritud sisend uue süsteemi arhitektuurile ning selle disainile. Tuginedes analüüsile toestatakse uue pilveteenuse väljatöötamist ja liidestamist olemasolevate teenustega, kirjeldatakse ettevõtte kontekstis uudse avaliku rakendusliidese tööpõhimõtteid ning seda kõike viisil, mis oleks firma ärimahtude juures turvaline ja skaleeruv. Lõputöö sisendi tulemusena on valminud reaalne töötav süsteem, mis on võetud edukalt kasutusele nii ettevõttes endas kui ka klientide poolt.

Lõputöö on kirjutatud eesti keeles keeles ning sisaldab teksti 60 leheküljel, 6 peatükki, 27 joonist, 10 tabelit.

Abstract

Business Process Improvement Through Cloud Based Testing Platform: A Testlio OÜ Case Study

The market for cloud technologies is on the rise, with corporate cloud infrastructure spending growing from $ 39.5 billion to $ 63 billion in 2021 [1]. Almost 80% of businesses today use some form of cloud solutions [2]. One of such companies is Testlio OÜ [3], which provides software testing services in the global market and has its roots in Estonia.

The thesis will focus on creating a holistic business and technological design for the development of a new cloud-based service, enabling enterprise customers to upload testable build artifacts directly from their engineering systems to the Testlio system. Thus, automating business-critical processes and increasing the overall attractiveness of Testlio as a platform. The work done describes in detail the scope to be addressed, sets the expected functional and non-functional requirements, formulates the success criteria and provides a structured input to the architecture and design of the new system. Based on this analysis, the development of a new cloud service and its integration with existing services are supported, describing the operating principles of the new public API in a business context, all in a way that is secure and scalable to the business volumes of the company. As a result of the thesis, a real-life working system has been developed, which has been successfully implemented and used daily within the company and by its customers.

The thesis is in Estonian and contains 60 pages of text, 6 chapters, 27 figures, 10 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*; rakendusliides võimaldab reeglistikule vastaval programmil teise programmi teenuseid kasutada
BPMN	*Business Process Modelling Notation*; graafiline notatsioon spetsifitseerimaks äriprotsesse ja töövooge
ERD	*Entity Relationship Diagram*; olemi-suhte diagramm seoste modelleerimiseks
FURPS	*Functionality Usability Reliability Performance Supportability*; meetod funktsionaalsete- ja mitte-funktsionaalsete nõuete modelleerimiseks ja klassifitseerimiseks
HTTPS	*Hypertext Transfer Protocol Secure*; turvaliseks kommunikatsiooniks mõeldud veebiprotokoll
ITIL	*Information Technology Infrastructure Library*; parimate IT halduspraktikate kogum
Kasutusjuht	*use case*; kirjeldus sündmuste jadast, mis loob mingisugust mõõdetavat väärtust süsteemiga suhtlejale
Pidev Integratsioon	Continuous Integration; arenduspraktika, mille kohaselt integreeritakse lähtekoodi jagatud repositooriumisse mitu korda päevas, ning mida seejärel verifitseerivad automaatsed ehitusmehhanismid
SLA	*Service Level Agreement;* kokkulepe IT-teenuste pakkuja ja kliendi vahel
UML	*Unified Modeling Language*; tarkvarasüsteemide spetsifitseerimiseks, visualiseerimiseks, modelleerimiseks ja dokumenteerimiseks mõeldud keel

Sisukord

1 Sissejuhatus 10
2 Valdkond 12
2.1 Ettevõtte tutvustus 12
2.2 Ettevõtte struktuur ja rollid 13
2.3 Testlio süsteem 14
2.4 Uurimisküsimused 14
2.5 Metoodika 14
3 Analüüs 16
3.1 Valdkonna kaardistus (AS-IS) 16
3.1.1 Intervjuud 16
3.1.2 Äridomeen (AS-IS) 18
3.1.3 Äriprotsess (AS-IS) 20
3.1.4 Keskkondade jaotuse probleem 22
3.1.5 Tehnoloogiliste piirangute probleem 25
3.2 Nõuded 26
3.2.1 Rakendusdomeen 26
3.2.2 Kasutusjuhud 28
3.2.3 Välise integratsiooni nõuded 30
3.2.4 Sisemise integratsiooni nõuded 35
3.2.5 Kasutajate nõuded 38
3.3 Edukuse mõõdupuu 42
3.3.1 Süsteemi vastuvõtmine 42
3.3.2 Pilootprojekt 43
3.4 Valdkonna kaardistus (TO-BE) 44
3.4.1 Äridomeen (TO-BE) 44
3.4.2 Äriprotsess (TO-BE) 45
4 Arhitektuur 47
4.1 Infrastruktuuri arhitektuur (AS-IS) 47
4.1.1 Infrastruktuuri pilvekomponendid 48

4.1.2 Veebilehtede vaade ... 49
4.1.3 Serveriga veebiteenuste vaade ... 51
4.1.4 Serverita veebiteenuste vaade ... 54
4.1.5 Ettevõtte pilvearhitektuur ... 56
4.2 Infrastruktuuri arhitektuur (TO-BE) ... 58
4.3 Andmearhitektuur ... 60
4.3.1 Andmemudel ... 60
4.3.2 Tabelite disain ... 62
4.4 Mikroteenuste arhitektuur ... 64
4.4.1 Kasutatavad teenused ... 64
4.4.2 Teenuste avastamine ... 65
4.4.3 Keskkondade automaatne lisamine ... 66
4.4.4 Keskkondade manuaalne lisamine ... 67
4.4.5 Keskkondade taaskasutamine ... 68
5 Valideerimine ja verifitseerimine ... 69
5.1 Süsteemi kasutuselevõtt ... 69
5.2 Süsteemi võimekus ... 70
5.3 Avalik väljapanek ... 71
6 Kokkuvõte ... 73
Kasutatud kirjandus ... 75
Lisa 1 – DynamoDB andmebaasi eksport ... 82
Lisa 2 – Andmebaasi tabelite atribuudid ... 83
Lisa 3 – Avaliku rakendusliidese integratsioonijuhis ... 89

Jooniste loetelu

Joonis 1. Testlio organisatsiooni struktuur 13
Joonis 2. Testlio AS-IS äridomeen 18
Joonis 3. Testlio kliendi AS-IS elutsükkel, tase 0 20
Joonis 4. Testlio kliendi AS-IS testimisprotsess, tase 1 20
Joonis 5. Testlio kliendi AS-IS testimisprotsessi ressursside hange, tase 2 21
Joonis 6. Testlio keskkonna üleslaadimiste jaotused, 08.05.2018 seisuga 22
Joonis 7. Loodava süsteemi kontekstidiagramm 27
Joonis 8. Loodava süsteemi probleemiraamistiku diagramm 27
Joonis 9. Süsteemi kasutusjuhtude diagramm 30
Joonis 10. Erinevate arenduskeskkondade küpsusaste 43
Joonis 11. Testlio TO-BE äridomeen 44
Joonis 12. Testlio kliendi TO-BE testimisprotsessi ressursside hange, tase 2-1 45
Joonis 13. Testlio kliendi TO-BE testimisprotsessi ressursside hange, tase 2-2 46
Joonis 14. Pilvepõhise infrastruktuuri pakkujate paigutus turul, Gartner 2019 48
Joonis 15. Testlio veebilehtede taristu AS-IS abstraktne arhitektuur 50
Joonis 16. Testlio serveriga veebiteenuste taristu AS-IS abstraktne arhitektuur 53
Joonis 17. Testlio serverivabade veebiteenuste taristu AS-IS abstraktne arhitektuur 55
Joonis 18. Testlio pilvepõhise taristu minimeeritud AS-IS arhitektuur 57
Joonis 19. Testlio pilvepõhise taristu minimeeritud TO-BE arhitektuur 59
Joonis 20. Testimisplatvormi uus andmemudel 61
Joonis 21. Testlio teenuste avastamise abstraktne mudel 65
Joonis 22. Keskkondade automaatne lisamine läbi Testlio avaliku rakendusliidese 66
Joonis 23. Keskkondade manuaalne lisamine läbi kasutajaliidese 67
Joonis 24. Keskkondade taaskasutamine läbi kasutajaliidese 68
Joonis 25. Kuvatõmmis keskkondade taaskasutamisest 70
Joonis 26. Avaliku rakendusliidese jõudlus 71
Joonis 27. Kuvatõmmis avaliku rakendusliidese müügist kodulehel 72

Tabelite loetelu

Tabel 1. Testlio äripoole intervjuud 16
Tabel 2. Testlio äridomeeni olemite selgitused 19
Tabel 3. Testlio keskkondade jaotuste ajakulu tabel 24
Tabel 4. Loodava süsteemi rollid ja kasutusjuhud 28
Tabel 5. Testlio näitel kasutusel olevate pilvelahenduste kategooriad 49
Tabel 6. Testlio veebilehtede abstraktse taristu ressursid 50
Tabel 7. Testlio serveriga veebiteenuste abstraktse taristu ressursid 51
Tabel 8. Testlio serverivabade veebiteenuste abstraktse taristu ressursid 54
Tabel 9. Uue andmemudeli unikaalsed indeksid 63
Tabel 10. Uue süsteemi käideldavus 71

1 Sissejuhatus

Käesoleva magistritöö teemaks on äridomeeni ja -protsesside analüüsimine, tehnilise arhitektuuri disain ning ettevõtte teenustele avaliku rakendusliidese väljatöötamine Testlio OÜ näitel.

Lähtutud on vajadusest luua ettevõttele praktilist väärtust. Keskselt on kasutatud disainiteaduse (*research science*) põhimõtteid, kaasates lahenduse väljatöötamisse valdkonna parimaid praktikaid. Arvestatud on ettevõtte ärilise ja tehnoogilise kontekstiga: pakkuda tarkvara testimisteenuseid globaalsel skaalal, pilvetehnoloogiate ja mikroteenuste tehnilises maailmas. Lähtepunktiks on sisuline vajadus terviklikule süsteemidisainile, uue teenuse väljatöötamisele ja liidestamisele, avaliku rakendusliidese loomisele ning seda kõike viisil, mis oleks firma ärimahtude juures turvaline ja skaleeruv.

Magistritööl on järgnevad eesmärgid:

1. Kaardistada ettevõtte äridomeen ja relevantsed äriprotsessid. Sõnastada probleemipüstitus ja -analüüs.
2. Sõnastada vajaliku lahenduse rakendusdomeen. Kirjeldada loodava süsteemi poolt tekkivat väärtust läbi kasutusjuhtude.
3. Modelleerida ja liigitada süsteemi nõudeid, arvestades ettevõtte seni tehtud tehnoloogilisi valikuid.
4. Määratleda parendustega äridomeeni ja -protsessi ning põhjendada selle väärtust.
5. Luua tehniline sisend süsteemi väljaarendamiseks ettevõttes, kattes selleks vajamineva tehnilise lahenduse erinevaid kihte:
    a. pilvepõhise infrastruktuuri edasiarendus;
    b. mikroteenuste ökosüsteemis uue teenuse ehitamine ja integreerimine;
    c. uue andmemudeli disainimine.
6. Pakkuda süsteemi tootmiskeskkonda evitamise strateegiat läbi kvaliteedi versapostide.
7. Mõõdistada loodud lahenduse vastuvõtmist ja relevantsust väljatoodud probleemide lahendamisel.

Magistritöö kirjutaja on ettevõttes olnud arhitekti rollis alates 9. aprill 2018. Lähtudes õppekava iseloomust ja piiratud mahust, on lõputöö koostatud ennekõike analüütilises ja süstemaatilises võtmes. Autor täitis loodud lahenduse juures väga mitmekesiseid rolle – sh analüütik, arendaja, testija ja administraator – olles keskne tehniline lüli lahenduse loomisel.

2 Valdkond

Valdkonnas keskendutakse lõputööle konteksti loomisega. Eesmärk on anda ülevaade vaadeldavast ettevõttest, selle struktuurist, olemasolevast süsteemist ning sellele tuginedes luua seos lõputöö uurimisküsimuste ja metoodikaga.

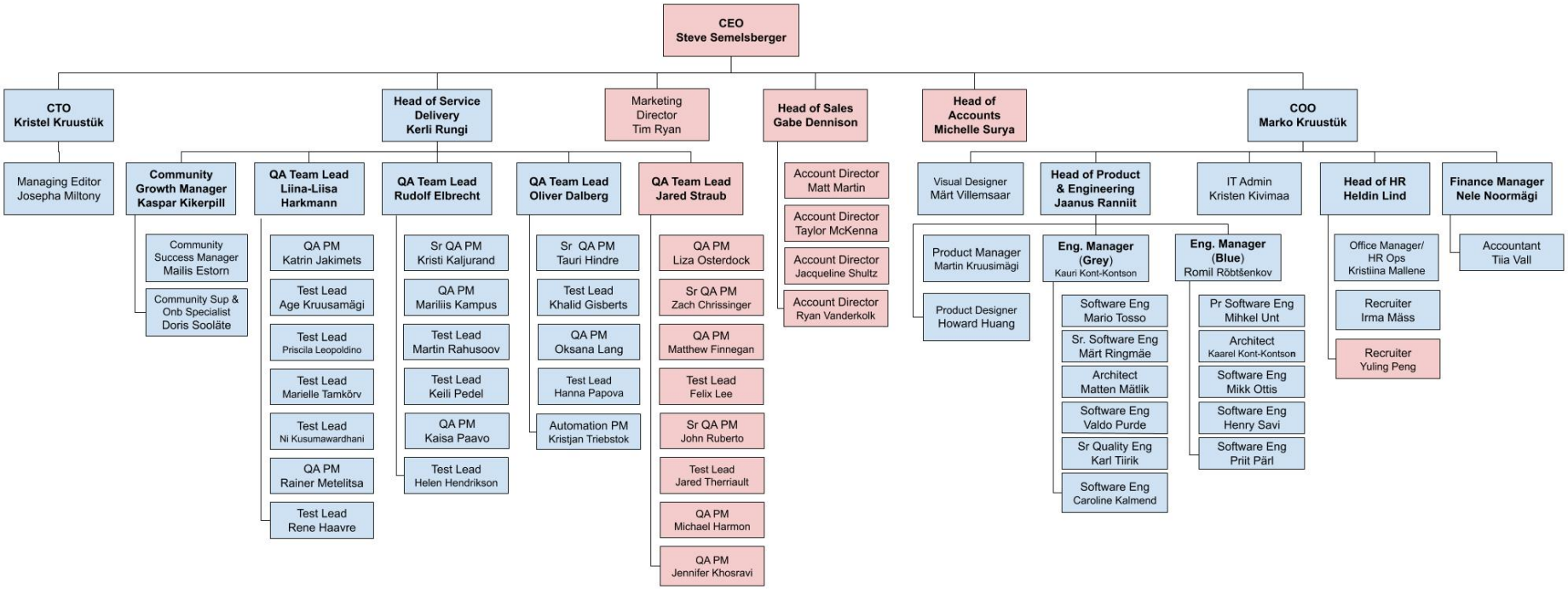
2.1 Ettevõtte tutvustus

Testlio OÜ näol on tegemist 2013 aastal Kristel ja Marko Kruustük poolt asutatud ettevõttega, mille põhitegevusala on ärinõustamine ja muu juhtimisalane nõustamine [4]. Testlio edulugu algas Ameerika Ühendriikides toimunud AngelHack nimelise programmeerimismaratoni (*hackaton*) võiduga. Äriline vundament loodi Texase osariigis, Techstars ettevõtluskiirendis. Tänaseks on ettevõte kaasanud üle 8,5 miljoni USA dollari rahastust ning omab kliente üle terve maailma. Mõned tuntumad nimed on Microsoft, NBA ja Manchester United [5]. Testlio arenduskeskus asub Tallinnas, kus töötab ka enamik ettevõtte *circa* 70-st töötajast. Samuti omab firma kontoreid Tartus ning Ameerika Ühendriikides, Palo Altos. Viimases on vaid ärikesksed rollid kuna ettevõtte enamik kliente on USA taustaga.

Testlio tuumaks on vabakutseliste testijate kogukond. Kogukonnas on testijaid üle terve maailma ning testijaks saamiseks piisab Testlio platvormil kompetentsustestide läbimisest. Testlio põhifookus on mobiilsete rakenduste funktsionaalsel testimisel, kuid täiendavalt pakutatakse ka töölaua (*desktop*) rakenduste ning veebikeskkondade tuge. Lisaks funktsionaalsele testimisele on ettevõtte teenuste hulgas uuriv testimine (*exploratory testing*), regressioonitestimine (*regression testing*), kasutatavuse testimine (*usability testing*), automatiseeritud testimine (*automated testing*) jpm [6].

2.2 Ettevõtte struktuur ja rollid

Joonis 1 kujutab Testlio kui organisatsiooni detailset struktuuri. Sinise markeeringuga on tähistatud Eestis eksisteerivad rollid, punasega Ameerika Ühendriikide omad [7]. Samuti kuuluvad ettevõtte juhtkonda kahe võlausaldajast investorfirma esindajad, vastavalt Altos Ventures ja Vertex Ventures [5].



Joonis 1. Testlio organisatsiooni struktuur

2.3 Testlio süsteem

Testlio kui platvorm on hallatud teenus (*managed service*), mille moodulid on keskkonnad (*builds*), testid (*tests*), testiplaanid (*plans*), meeskonnad (*teams*), testimistsüklid (*runs*), defektid (*issues*) ning raportid (*reports*) [8]. Süsteem ise on pilvetehnoloogiate abil realiseeritud mikroteenuste ning monoliidi hübriidlahendus, mille eesmärk on ettevõtte klientidele pakkuda tarkvara testimisteenuseid ning kõike sellega seonduvat [9]. Peamisteks programmeerimiskeelteks on JavaScript ja PHP. Erinevate tehnoloogiate mitmekesisusest ja sihtotstarbest annab detailsema ülevaate arhitektuuri sektsioon (ptk 4).

2.4 Uurimisküsimused

Fookuses on kolm punkti:

1. Kuidas tõhustada ettevõtte testimisprotsesse?
2. Kuidas disainida süsteemi avalikku rakendusliidest ettevõtte klientide tarbeks?
3. Kuidas rakendada ettevõtte tehnilisi lahendusi pilve- ja mikroteenuste maailmas?

Uuritavaid küsimusi analüüsitakse Testlio OÜ kontekstis. Loodavad lahendused on teatavate kohandustega kasutatavad ka teistes ettevõtetes. Vajalik kohandusaste sõltub eelkõige rakendusdomeeni ja tehnilise taristu sarnasusest.

2.5 Metoodika

Lõputöö metoodika valikul on lähtutud vajadusest luua ettevõttele praktilist väärtust. Disainiteadus (*research science*) on võrdlemisi uus ja disainile orienteeritud uurimismetoodika [10]. Tegevuse eesmärgiks on läbi innovaatiliste artefaktide pakkuda lahendust olemasolevale probleemile - konstrueerides niimoodi uue reaalsuse; selle asemel, et lihtsalt olemasolevat mõtestada [11]. Mitmetahulise ja dünaamilise probleemidomeeni lahendamiseks võimaldab disainiteadus kaasata valdkonna parimaid praktikaid [12].

Lahenduse loomisel kasutatakse täiendavaid vahendeid, sh:

1. BPMN (*Business Process Modelling Notation*) – äriprotsesside modelleerimiseks rakendatud keel, ehk äriprotsesside dünaamiline vaade.
2. ERD (*Entity Relationship Diagram*) – äri- ja andmemudelite kirjeldamine läbi omavaheliste seoste, ehk äri- ja andmeolemite staatiline vaade.
3. Erinevad testimismetoodikad loodava lahenduse kvaliteedi parendamiseks, sh:
    a. A/b testimine.
    b. Integratsioonitestimine.
    c. kasutatavuse testimine.
    d. Regressioonitestimine.
    e. Süsteemitestimine.
    f. Turvalisuse testimine.
    g. Vastuvõtutestimine.
    h. Ühiktestimine.
4. FURPS (*Functionality Usability Reliability Performance Supportability*) – funktsionaalsete- ja mittefunktsionaalsete nõuete modelleerimiseks ning klassifitseerimiseks.
5. ITIL (*Information Technology Infrastructure Library*) – muudatuste halduse (*change management*) ning reliisi- ja evitusprotsessidele (*release and deployment management*) kehtivate nõuete modelleerimine loodavasse lahendusse;
6. Pidev Integratsioon (*Continuous Integration*) – komponendivaates soovitavate ehitus- ja tarbimismustrite määratlemiseks.
7. UML (*Unified Modeling Language*) – üldine modelleerimiseks kasutatud keel, peamiselt kasutusjuhtude- ja jadadiagrammide näol.

3 Analüüs

Analüüs keskendub valdkonna anatoomia lahkamisele, kaardistades ettevõtte *status quo* ning selle eesmärgistatud ideaali konkreetse probleemidomeeni kontekstis. Analüüsi tulemusena luuakse struktureeritud sisend uue süsteemi arhitektuurile ning selle disainile.

3.1 Valdkonna kaardistus (AS-IS)

AS-IS sektsiooni eesmärk on formuleerida nii äridomeeni kui ka -protsesside hetkeseis, luues seeläbi vundamendi edasisele analüüsile. Sõnastatakse ka konkreetsed probleemid ning nende relevantsus lõputöö kontekstis.

3.1.1 Intervjuud

Ettevõtte protsesside ja valdkonna teadmiste lahtimõtestamiseks on läbi viidud mitu intervjuud. Osapoolteks on olnud lõputöö autor ning Testlio teised võtmetöötajad. Intervjuud ise on teostatud vabas vormis. Täpsemaid detaile kirjeldab Tabel 1, millele järgnevad sisu lühikokkuvõtted.

Tabel 1. Testlio äripoole intervjuud

Identifikaator	Intervjueeritav	Roll	Kontakt	Toimumise kuupäev	Pikkus
Intervjuu 1	Kristi Kaljurand	Projektijuht (QA Project Manager)	kristi@testlio.com	18.04.2018	1,5 tundi
Intervjuu 2	Rene Haavre	Testimisjuht (Test Lead)	rene@testlio.com	19.04.2018	1,5 tundi

Intervjuu 1:
Müügiosakond teeb potentsiaalsetele klientidele Testlio keskkonnas demonstratsioone. Peamiselt näidatakse kliendivaadet – kuidas kliendi projekti seadistatakse, kuidas testimistsükleid luuakse, kuidas projekti sobivaid testijaid leitakse, kuidas defekte raporteeritakse ning kuidas üldine aruandlus välja näeb. Enamik voogudest juhtub platvormil, mis on Testlio peamine rakendus. Eduka müügi korral sõlmitakse raamleping,

kus määratakse osutatava teenuse sisu, hind, maht ning tingimused. Testijaid leitakse testijate kogukonnast ning igal testijal on platvormil oma profiil – oskused, tase, seadmete kogu ning ajalooline sooritus. Osadel testijatel on suur hulk eri seadmeid – peamiselt Android ja iOS süsteemidega mobiiltelefone – teistel on vaid mõned üksikud. Testijaid otsitakse testimistsüklitesse seadmete ning nende parameetrite järgi, mis on tavaliselt kliendi poolt ette määratud. Näiteks kui kliendil on Androidi rakendus, siis tavaliselt tellitakse uutemate Androidi seadmete peal testimist. Samuti arvestatakse projekti testija värbamisel tema reitingut ning senist koostöökogemust. Kliendi projekti edukuse eest vastutab projektijuht, kes ühtlasi täidab ka kliendihalduri funktsiooni.

Teenustaseme leppe (*Service Level Agreement*) kohaselt lubatakse kliendile tulemusi 48 tunni ehk kahe ööpäeva jooksul, kus sisemine aeg jaguneb järgmiselt:

1. ettevalmistus, raporteerimine ning puhver – 24 tundi;
2. testimise osa – 20 tundi;
3. ülevaatuse osa – 4 tundi.

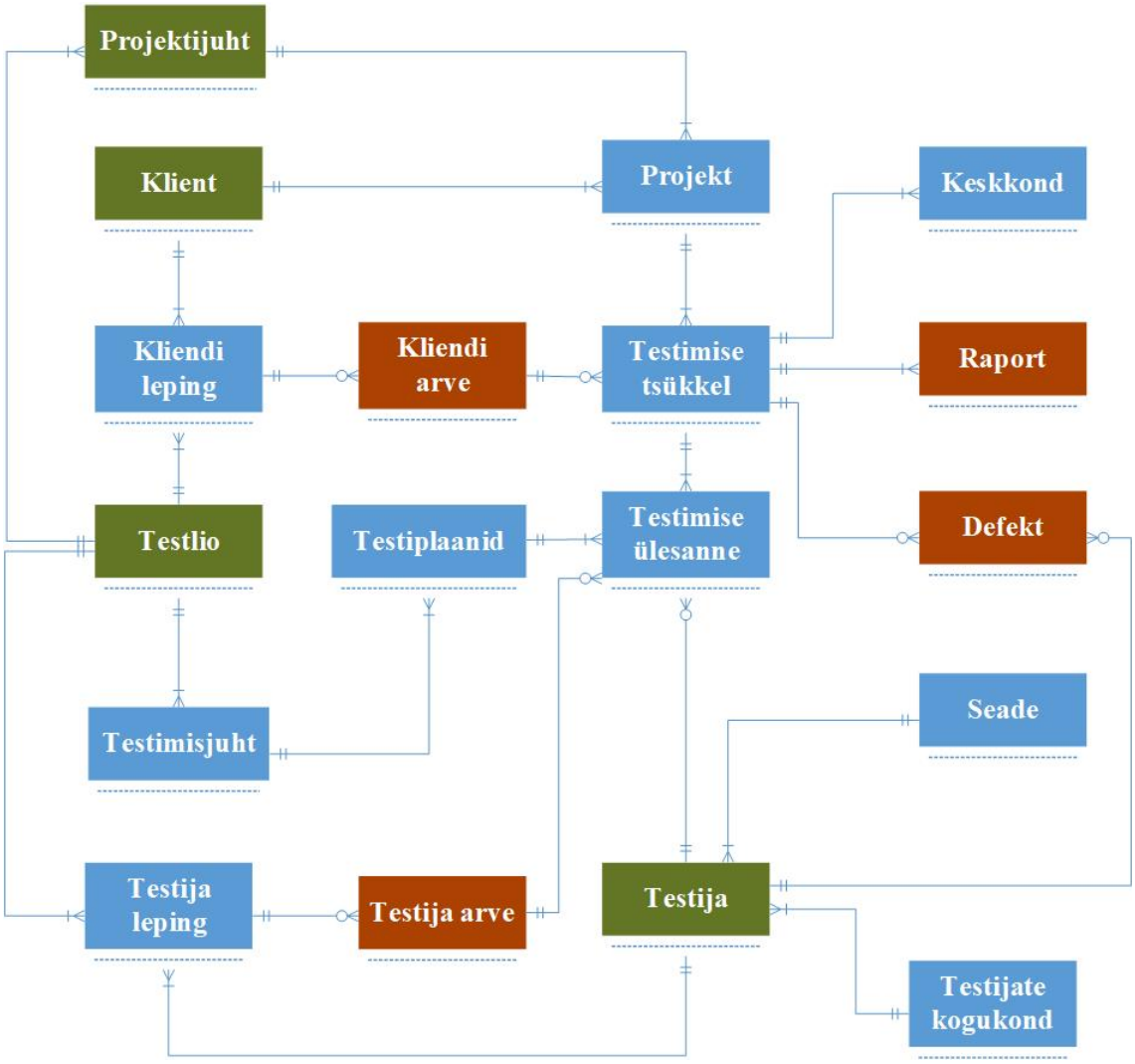
Intervjuu 2:

Testijuhi fookuses on kvaliteetsete testimistulemuste tagamine läbi Testlio protsesside. Kliendi nõuete alusel luuakse projektile testiplaanid, mis omakorda tükeldatakse konkreetseteks testimise ülesanneteks. Testiplaanide ehitus ning haldus käivad kliendi projekti elutsükli järgi. Testimise ülesanded luuakse sisendina testimise tsüklitele ning jaotatakse tsüklis osalevatele testijatele. Edukaks tsükliks on vaja nelja sisendit – testijaid, testimise ülesandeid, keskkonda (*build*) ning juhtimist. Keskkond määratakse kliendi poolt. Keskkonna üleandmise viis varieerub klienditi – levinumad meediumid on kliendi artefaktide süsteem, OneDrive, Dropbox, Skype ja e-mail. Tüüpiliselt kulub keskkonna sõnumi saamise järgselt suurusjärgus 5 – 15 minutit keskkonna importimiseks Testlio süsteemidesse. Probleemid keskkonna tarnega mõjutavad ka testimise tsükli algust. Tsüklis avastatud defektid raporteeritakse platvormil, verifitseeritakse ning kajastatakse kliendiraportis. Testijate tasu ei sõltu edukalt raporteeritud defektide arvust, vaid tsüklis töötatud tundidest, mis omakorda korrutatakse konkreetse testija oskuste koefitsiendiga. Reaalses elus töötavad projektijuht ning testimisjuht tihedalt koos üle erinevate testimiste tsüklite.

3.1.2 Äridomeen (AS-IS)

Testlio ärivaldkond ehk domeen käitleb endas ettevõtte äritegevuse jaoks olulisemaid olemeid ning nendevahelisi suhteid. Informatsiooni talletamisel on suuresti aluseks võetud varasemalt teostatud intervjuud (ptk 3.1.1) ning modelleerimisel on lähtutud ERD (*Entity Relationship Diagram*) ehk olemi-suhte diagrammi notatsioonist [13]. Joonis 2 kujutab domeeni olulisemaid olemeid ning nendevahelisi suhteid. Parema ülevaate andmiseks on täiendavalt rakendatud järgnevat värviskeemi:

1. **roheline** – valdkonna olulisemad rollid;
2. **oranž** – testimisprotsessi artefaktid;
3. **sinine** – kõik muu, mh toetavad olemid.



Joonis 2. Testlio AS-IS äridomeen

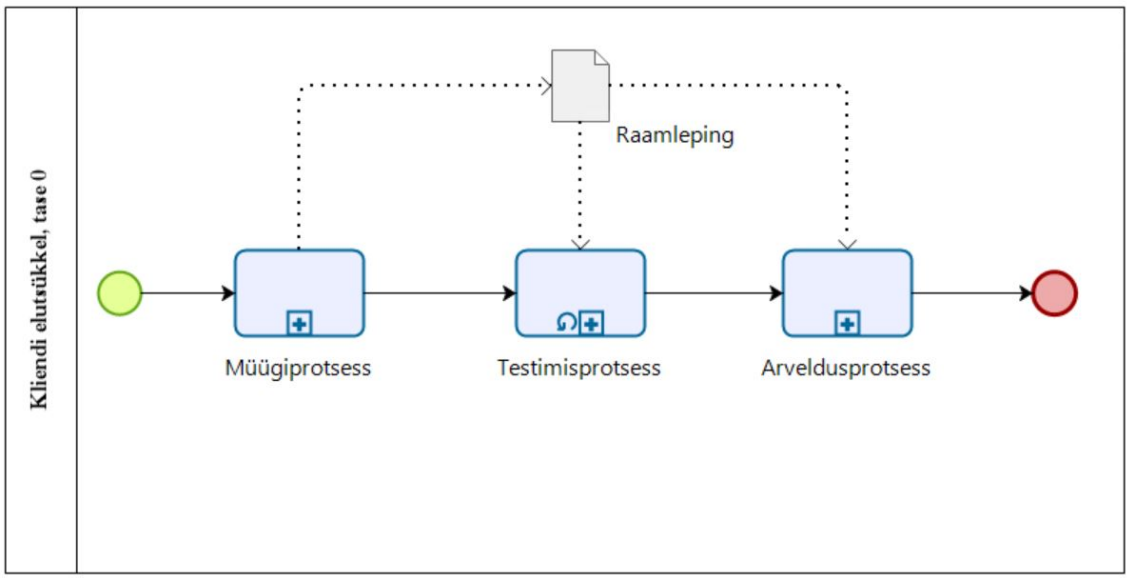
Kliendile müüakse testimise tsükleid ning sõlmitud lepingu alusel toimub arveldus. Testijate kogukonna testijad saavad palka testimise ülesannete kaudu, mille aluseks on leping Testlioga. Testimise tsükli tulemusena tekivad erinevad raportid mis kajastavad testimise sisendeid ja väljundeid. Tabel 2 kirjeldab äridomeeni olemite konteksti ning nendevahelisi seoseid.

Tabel 2. Testlio äridomeeni olemite selgitused

ID	**Olem**	**Kontekst**
BDAS1	Testlio	Vaadeldav ettevõte
BDAS2	Klient	Osapool, kellega *Testlio* omab lepingulist suhet ning kellele osutatakse testimisteenust
BDAS3	Kliendi leping	*Testlio* ja *Kliendi* vahelist suhet ja *Testlio* poolt osutatava testimisteenuse hinnastamist määratlev juriidiline dokument
BDAS4	Kliendi arve	*Kliendile* esitatav arve *Testlio* poolt osutatud teenuste eest, väljamaksega *Testliole*
BDAS5	Projekt	Ühele *kliendi* tootele vastav loogiline üksus
BDAS6	Projektijuht	*Projekti* äriprotsesse omav vastutav isik *Testlios*
BDAS7	Testimisjuht	*Projekti* testimisprotsesse omav vastutav isik *Testlios*
BDAS8	Testija	Tarkvara testimise eest vastutav isik
BDAS9	Testija leping	*Testlio* ja *Testija* vahelist suhet ja testija poolt osutatava testimisteenuse hinnastamist määratlev juriidiline dokument
BDAS10	Testija arve	*Testliole* esitatav arve *Testija* poolt osutatud teenuste eest, väljamaksega *testijale*
BDAS11	Testijate kogukond	Vabaturu tingimustes eksisteeriv *testijate* hulk
BDAS12	Seade	Füüsiline agregaat mille peal testimist teostatakse. Näiteks Apple iPhone XS nutitelefon
BDAS13	Keskkond	Testimise subjektiks olev *kliendi* tarkvara inkrement
BDAS14	Testiplaanid	*Kliendi* tarkvara testimise skoopi ja tegevusi kirjeldav dokument
BDAS15	Testimise tsükkel	Ajaperiood mille jooksul toimub *keskkondade* testimine *testijate* poolt
BDAS16	Testimise ülesanded	Konkreetse *testiplaani* alusel loodud konkreetsed testimisjuhised konkreetsele *testijale* konkreetses *testimistsüklis*
BDAS17	Defekt	Testimise käigus leitud *keskkonna* tarkvaraline puudujääk
BDAS18	Raport	*Testimistsükli* tulemusi koondav dokument

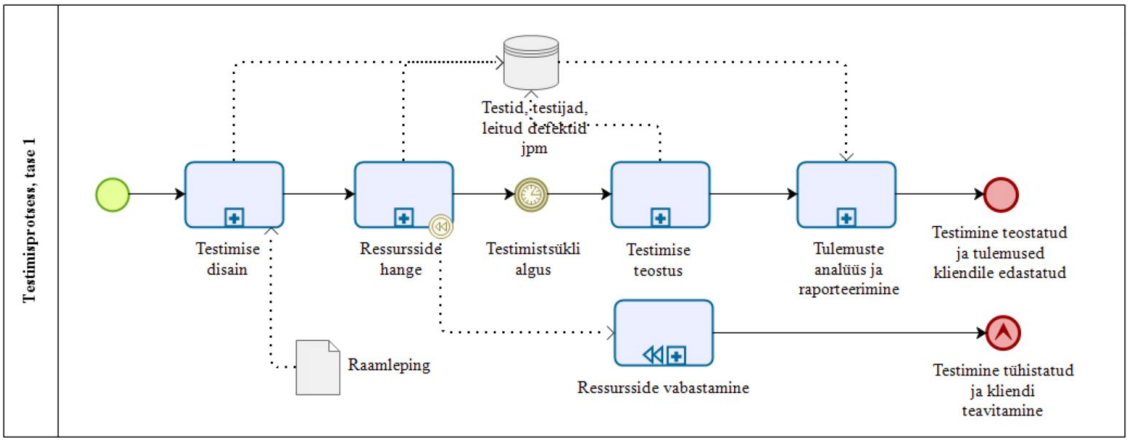
3.1.3 Äriprotsess (AS-IS)

Ettevõtte äriprotsesside modelleerimiseks on kasutatud lahendusena BPMN (*Business Process Modelling Notation*) keelt, mis võimaldab erinevalt UML'ist modelleerida äriprotsesse sündmuskeskse lähenemise järgi [14]. Järgnevad joonised (Joonis 3, Joonis 4 ja Joonis 5) käitlevad endas äriprotsesside kolme erinevat taset, liikudes üldisemast rohkem spetsiifilisemaks. Spetsiifilistematel tasemetel on lahatud ainult lõputöö skoobis relevantseid alamprotsesse.



Joonis 3. Testlio kliendi AS-IS elutsükkel, tase 0

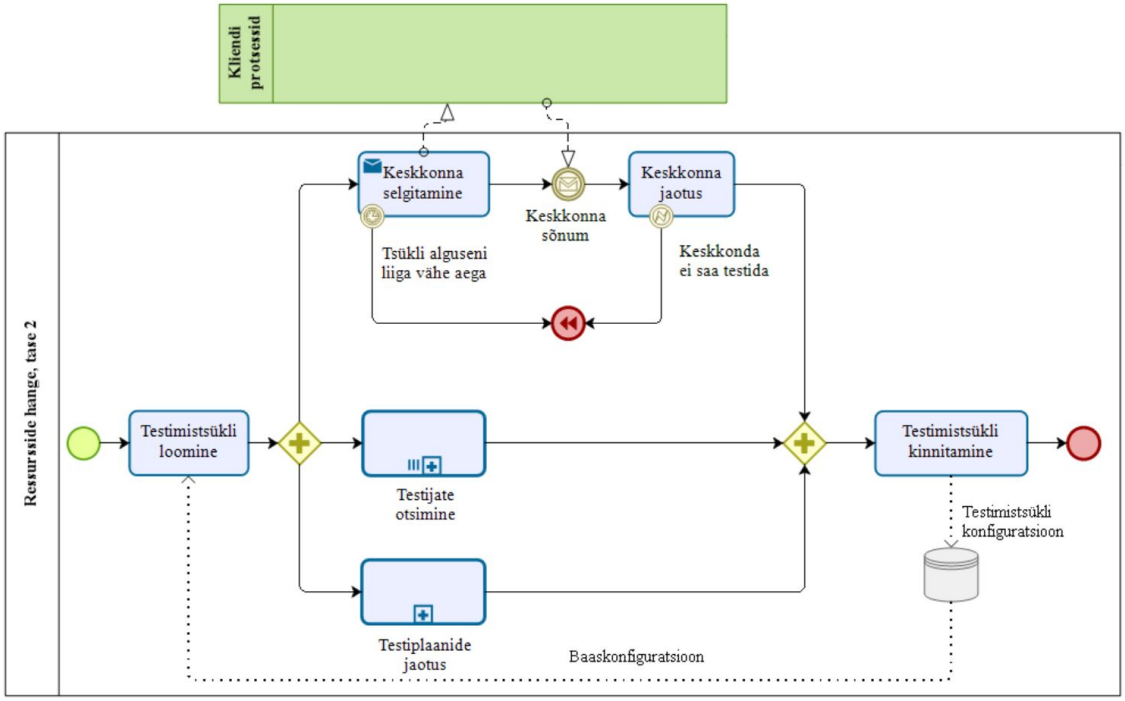
Kliendi elutsükli keskseks osaks on osapoolte vahel sõlmitav raamleping (Joonis 3), mis määrab ära Testlio poolt osutatava testimisteenuse tüüptingimused.



Joonis 4. Testlio kliendi AS-IS testimisprotsess, tase 1

Raamleping on sisendiks testimise disainile (Joonis 4) ning selle alusel tekkinud teste teostatakse testimise alamprotsessis läbi ressursside hankes leitud testijate.

Ressursside hanke ebaõnnestumise korral vabastatakse eelnevalt broneeritud ressursid ning konkreetne testimistsükkel eskaleeritakse kliendile.

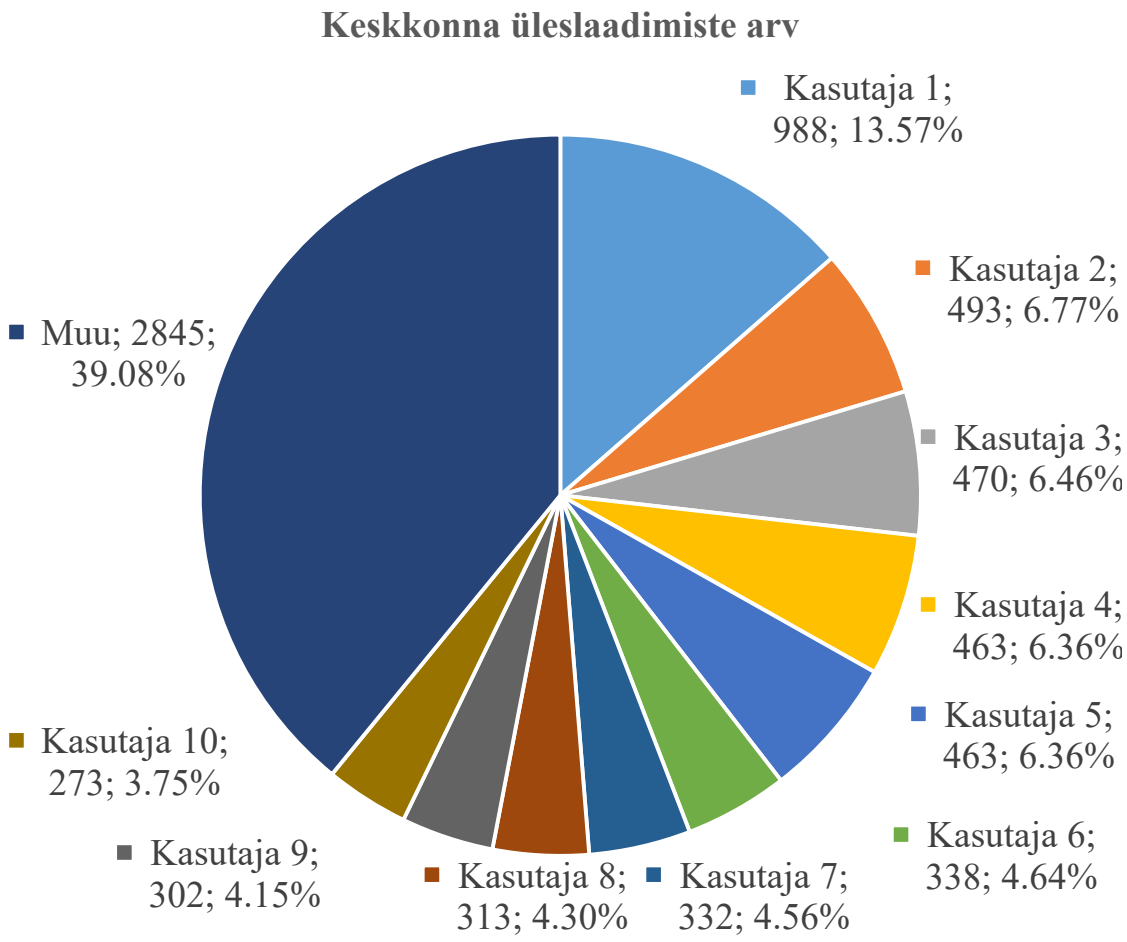


Joonis 5. Testlio kliendi AS-IS testimisprotsessi ressursside hange, tase 2

Joonis 5 kirjeldab ressursside edukaks hangeks vajalikke kolme komponenti: testijaid koos vastavate seadmetega, testiplaane ning keskkonda. Viimane jaguneb omakorda kas veebikeskkonnaks või tarkvaraliseks koosteks (*build*). Kõikide komponentide eduka hanke korral kinnitatakse testimistsükkel, mille käigus seotakse kõik ressurssid omavahel kokku üheks äriloogiliseks tervikuks – testimistsükli konfiguratsiooniks. Suurimaks riskikohaks on keskkondade hange, sest selle ajakohane puudumine võib tühistada terve testimistsükli. Testlio testijate kogukond on globaalne ning mitmekesine; pea igal ajahetkel leidub keegi, kes oleks huvitatud testimisest. Sarnaselt omab Testlio nii projekti- kui ka testimisjuhte eri ajavöötmetes. Kõige raskemini hajutatav on keskkondade hanke risk kuna see informatsioon peab pärinema kliendilt ning on tugevas seoses kliendi arendustsükliga.

3.1.4 Keskkondade jaotuse probleem

Testimise teostuseks vajalik keskkond laekub alati kliendi käest. Testimistsükli loomiseks pole keskkond rangelt nõutud kuid testimise teostuseks on see hädavajalik. Sellest johtuvalt võivad planeeritud tsüklid sattuda kas edasilükkamisele või tühistamisele kui keskkonda pole olemas ettenähtud ajaks. Kuna Testlio enamik kliente on USA turul siis tüüpiliselt täpsustatakse keskkond Eesti aja mõistes töötundide väliselt. Tulenevalt ajavööndite erinevusest (sõltuvalt kliendi asukohast tavaliselt 7 kuni 10 tundi) võivad keskkonnad täpsustuda ka Eesti öötundidel. Tüüpilises kliendi tarkvara arendusprotsessis toodetakse suuremale testimisringile minev keskkond just tööpäeva lõpu poole kuna see võimaldab maksimaalselt arendustunde rakendada. Keskkonna ajaline viide ning ajatsoonide erinevus on ka üks juurpõhjuseid miks klientidele müüakse 48 tunnist tsüklipõhist teenustaseme lepet. Joonis 6 kujutab keskkondade üleslaadimise jaotust erinevate autorite (st üleslaadijate) lõikes. Rakendatud andmeanalüüsi metoodikatest on täpsemalt võimalik juurde lugeda lisadest (Lisa 1).



Joonis 6. Testlio keskkonna üleslaadimiste jaotused, 08.05.2018 seisuga

Ülalpool toodud andmed on ajavahemikus 11.09.2015 kuni 08.05.2018. Tundliku töötajainfo kaitseks on autorite nimed anonümiseeritud. Autorite tegelikud nimed on lõputöö kirjutajale teada. Suurusjärgus ~61% keskkondade jagamisest tehakse kümne kasutaja poolt, kes kokku moodustavad vaid ~13% kõikide kasutajate arvust. Iga kasutaja aliase juures on lisaks välja toodud ka üleslaadimiste arv ning protsentuaalne suhtearv kõigi üleslaadimiste lõikes. Saadud kümme kasutajat on kõik ühel või teisel hetkel Testlio palgal olnud projekti- või testijuhid. Tabel 3 aluseks on keskmise ajakuluna võetud 10 minutit keskkonna kohta (ptk 3.1.1). Tööperioodi arvutamisel on lähtutud esmase ning viimase keskkonna üleslaadimise kuupäevalisest vahest. Töötatud aja arvutusel on arvesse võetud keskmist töötundide arvu kuus [15] ning 28 puhkepäeva aasta lõikes tulenevalt Eesti Vabariigis kehtivast töölepinguseadusest [16].

Tabel 3. Testlio keskkondade jaotuste ajakulu tabel

Autor	**Üleslaadimiste arv**	**Ajaline kulu (tundides)**	**Esmane kasutus**	**Viimane kasutus**	**Tööperiood (kuud ja päevad)**	**Töötatud aeg (tundides)**	**Keskkonna ajakulu suhtearv (protsentides)**
Kasutaja 1	988	164,67	03.06.2016	05.08.2018	26k 2p	4036,41	4,08
Kasutaja 2	493	82,17	03.06.2016	05.05.2018	23k 2p	3570,67	4,30
Kasutaja 3	470	78,33	23.08.2016	05.05.2018	20k 12p	3104,93	2,52
Kasutaja 4	463	77,17	02.06.2016	27.04.2018	22k 25p	3415,43	2,26
Kasutaja 5	463	77,17	26.10.2016	29.01.2018	15k 3p	2328,70	3,31
Kasutaja 6	338	56,33	20.12.2017	08.05.2018	4k 18p	620.99	9,07
Kasutaja 7	332	55,33	19.01.2017	16.04.2018	14k 28p	2173.45	2,55
Kasutaja 8	313	52,17	24.06.2016	06.05.2018	22k 12p	3415.43	1,53
Kasutaja 9	302	50,33	12.10.2016	06.12.2017	13k 24p	2018.21	2,49
Kasutaja 10	273	45,50	14.07.2016	08.09.2017	13k 25p	2018.21	2,25

3.1.5 Tehnoloogiliste piirangute probleem

Ettevõtte tehnoloogilises ökosüsteemis juba eksisteerib keskkondadele fokusseeritud teenus, ehk *build service (v2)*, mis valmis 2016 aasta suvel. Tegemist oli teise katsega luua selline teenus kuna teenuse esimene version jäi poolikuks. V2 tehniline implementatsioon baseerub serverivabadele pilvetehnoloogiatele (ptk 4.1.4); peamiselt rakendusliideste lüüsidele (*API Gateway*), funktsioonidele ja mitte-relatsioonilistele andmebaasidele (*NoSQL*). Tol ajal loodud lahendus täitis oma tehnilist eesmärki, kuid selle kasutuselevõtt avaliku rakendusliidese (*Public API*) ja andmeanalüüsi kontekstis on äärmiselt raskendatud. Juurpõhused on Testlio pilveteenuse pakkuja, Amazon Web Services (*AWS*), enda piirangutes:

1. Veebipäring ei saa kesta kauem kui 29 sekundit [17].
2. Edastatava faili suurus ei saa olla suurem kui 6 megabaiti [18].
3. AWS NoSQL andmebaasi indeksite arv on maksimaalselt 5 lokaalset ja 20 globaalset indeksit [19]. Korrektselt ehitatud indeksid on olulised nii andmebaasi päringute kiiruse kui ka andmete enda korrektsuse säilitamise huvides [20].
4. AWS NoSQL andmebaasis pole võimalik kasutada SQL konstruktsioone [21]. Selliste konstruktsioonide (näiteks `JOIN`) puudumine teeb ebakorrektselt modelleeritud NoSQL andmebaasist andmete agregeerimise äärmiselt keeruliseks [22].

Tuginedes seni üleslaetud keskkondade andmetele, võivad tarkvaralised koostete suurused võivad olla kuni gigabaitides. Loetletud tehnilistest piirangutest osaliselt ümbersaamiseks on senine *de facto* lahendus olnud keskkondade üleslaadimine kliendi (st brauseri) poolel. Selline lahendus pole kõige jätkusuutlikum ettevõtte avaliku rakendusliidese loomise ja andmeanalüüsi eesmärkide kontekstis. Keskkond on testimisteenuste osutamisel hädavajalik komponent, millest johtuvalt on keskkondade teenus on valitud avaliku rakendusliidese pilootprojektiks. Projekti tehnilise teostuse üks võtmekomponent on luua uus keskkondade teenus (v3), mis baseeruks klassikalisele serveriga veebiteenuste arhitektuurile (ptk 4.1.3) ning oleks seeläbi vaba serverivabade pilvetehnoloogiate piirangutest.

3.2 Nõuded

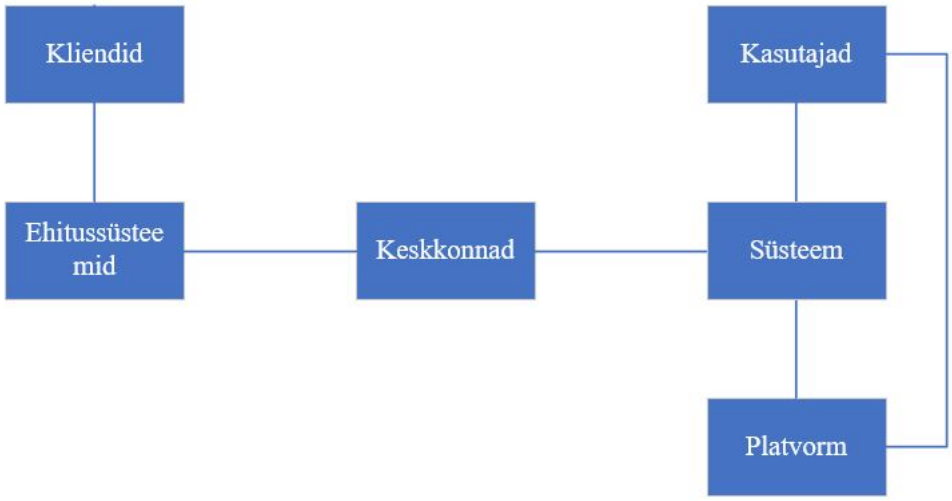
Nõuete modelleerimisel ja klassifitseerimisel rakendatakse FURPS mudelit, mille alusel jaotatakse nõuded funktsionaalseteks ja mitte-funktsionaalseteks nõueteks [23]. Funktsionaalsed nõuded vastavad küsimusele: „Mida süsteem tegema peab?“, mitte-funktsionaalsed küsimusele: „Kuidas süsteem töötab?“. Eesmärgiks on täiendada varasemalt kirjeldatud kasutusjuhtude nõudeid ning luua täiendav süsteemne dimensioon, eelkõige mittefunktsionaalsete nõuete aspektis. Nõuete struktrueerimisel kasutatakse M. Jacksoni probleemiraamistiku (*problem frame*) mudelit. M. Jackson väidab, et probleemi lahenduse olemus ei seisne nii väga loodava süsteemi kirjelduses, vaid rakendusdomeeni (*application domain*) struktuuris ja omadustes ning kliendi poolt esitatavates nõuetes sellele domeenile [24]. Nõuete alampeatükis keskendutakse domeeni lahkamisele ning formuleeritakse loodavale lahendusele püstitatavad nõuded.

3.2.1 Rakendusdomeen

Rakendusdomeen on piiritletud maailm, kus loodavat süsteemi rakendatakse. Domeeni kirjelduste mõistmiseks on vaja mõista domeenis asetsevaid ilminguid (*phenomena*), mida teostatakse läbi määratluste (*designations*) [25]. Varasemalt AS-IS äridomeenis (ptk 3.1.2) kirjeldatud klientide (olem **BDAS2**) ja keskkondade (olem **BDAS13**) olemitele lisanduvad uuritava rakendusdomeeni olulisemaid olemid:

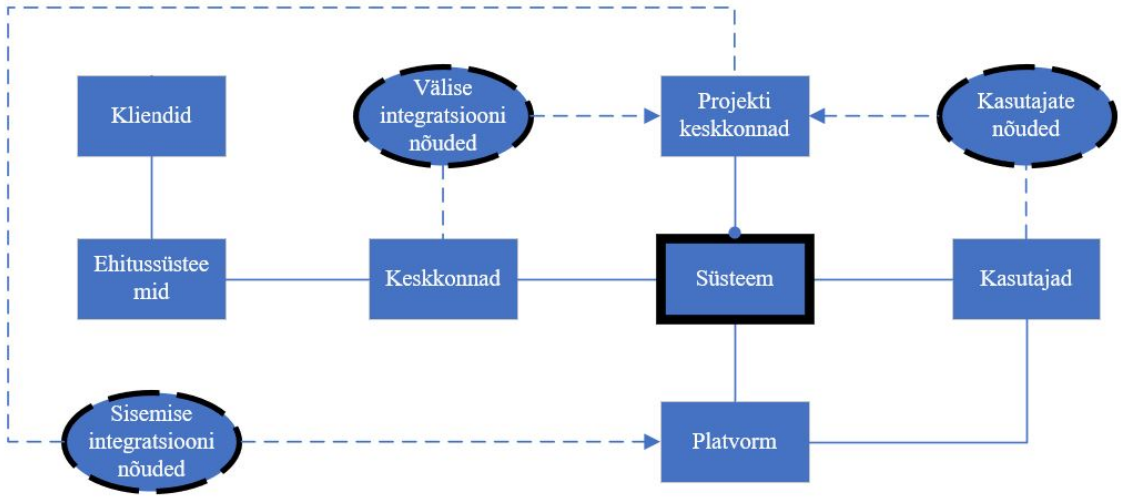
1. Süsteem – lõputöö raames loodav uus tarkvaraline süsteem.
2. Kasutajad – süsteemi kasutajad.
3. Ehitussüsteemid – kliendide CI (*Continious Integration*) süsteemid, kus keskkondi ehitatakse;
4. Platvorm – eksisteeriv Testlio keskkond, mille ümber ärikriitilised tegevused on põimunud;

Modelleerides loetletud olemid kontekstidiagrammile saame visualiseeritud rakendusdomeeni (Joonis 7).



Joonis 7. Loodava süsteemi kontekstidiagramm

Raamistades kontekstidiagrammi erinevate nõudete järgi tekib probleemiraamistiku diagramm (*problem frame diagram*). Joonis 8 laiendab eelnevat kontekstidiagrammi (Joonis 7) ning keskendub rakendusdomeenide sidususele nõuetega.



Joonis 8. Loodava süsteemi probleemiraamistiku diagramm

Süsteemile esitatavad nõuded võib jagada kolme kategooriasse:

1. välise integratsiooni nõuded – kuidas ning mis tingimustel peaksid keskkonnad jõudma uude süsteemi;
2. sisemise integratsiooni nõuded – kuidas peaks toimima liidestus eksisteerivate Testlio süsteemidega;
3. kasutajate nõuded – kuidas Testlio kasutajad soovivad uut süsteemi kasutada ning millist funktsionaalsust oodatakse.

3.2.2 Kasutusjuhud

Kasutusjuht on kirjeldus sündmuste jadast, mis loob mingisugust mõõdetavat väärtust süsteemiga suhtlejale [26]. Uue pilvepõhise lahenduse skoobis vaadeldud rollid baseeruvad varasemalt AS-IS äridomeenis (ptk 3.1.2) kirjeldatud kliendi (olem **BDAS2**), projektijuhi (olem **BDAS6**), testijuhi (olem **BDAS7**) ning testija (olem **BDAS8**) olemitele, mille osas on tehtud rollide näol vastavaid üldistusi:

1. kasutaja – baasroll kõikidele teistele rollidele;
2. juht – projektijuht või testimisjuht;

Tabel 4 puhul on vastavalt rollidele ja nende üldistustele välja toodud süsteemi peamised kasutusjuhud. Pariteetsus tähendab vanas keskkondade süsteemis (ptk 3.1.5) eksisteerivat funktsionaalsust, mille väljavahetamiseks on vajalik samaväärse või parema funktsionaalsuse pakkumine uues süsteemis. Kasutusjuht mis pole pariteetne, on kasutajate mõistes uhiuus süsteemi funktsionaalsus.

Tabel 4. Loodava süsteemi rollid ja kasutusjuhud

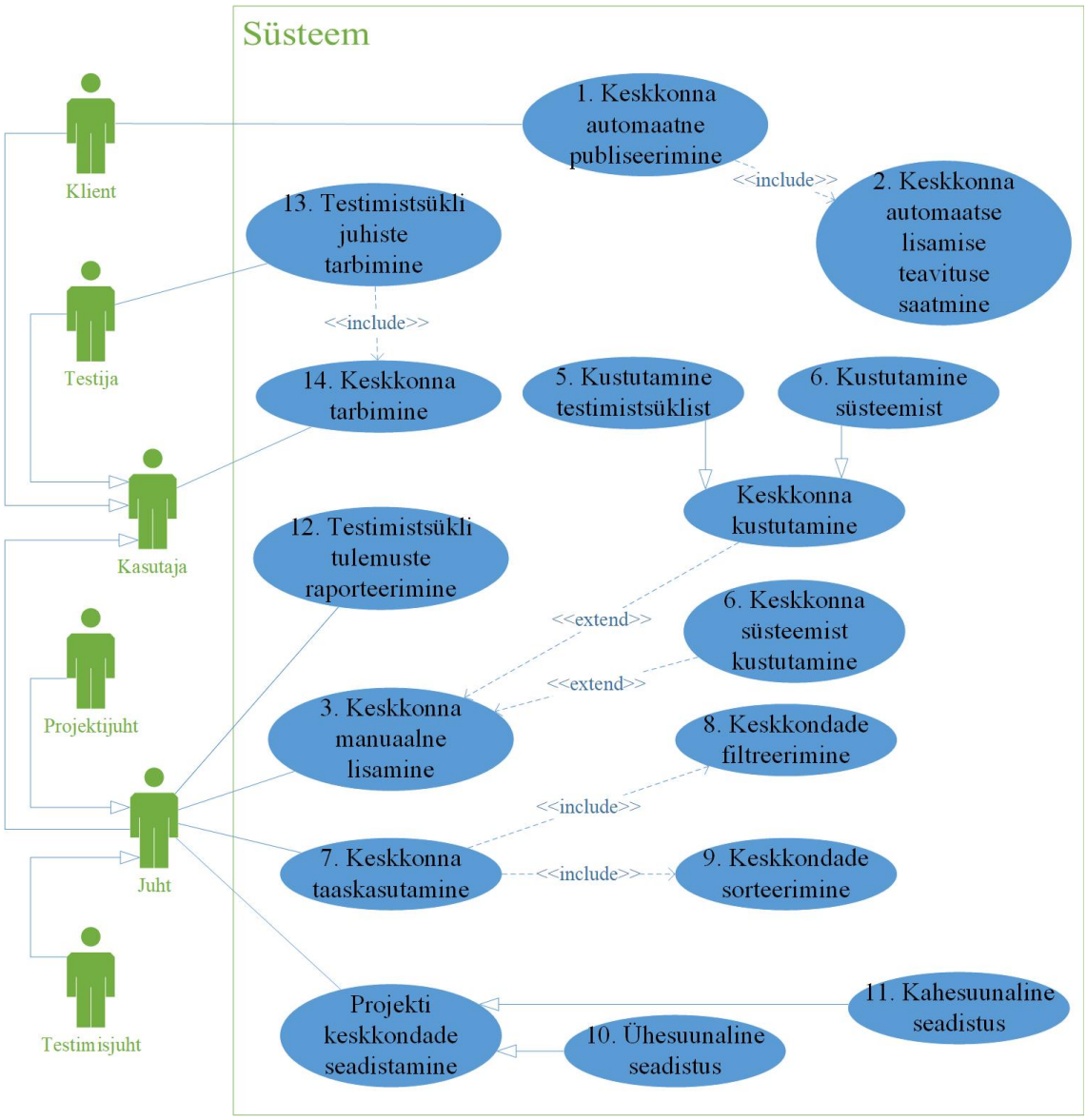
ID	Roll	Kasutusjuht	Pariteetsus
UC1	Klient	Keskkonna automaatne lisamine	Ei
UC2		Keskkonna automaatse lisamise teavituse saatmine	Ei
UC3	Juht	Keskkonna manuaalne lisamine	Jah
UC4		Keskkonna andmete muutmine	Ei
UC5		Keskkonna kustutamine testimistsüklist	Jah
UC6		Keskkonna kustutamine süsteemist	Ei
UC7		Keskkonna taaskasutamine	Ei
UC8		Keskkondade filtreerimine	Ei
UC9		Keskkondade sorteerimine	Ei
UC10		Projekti keskkondade seadistamine – ühesuunaline	Ei
UC11		Projekti keskkondade seadistamine – kahesuunaline	Ei
UC12		Testimistsükli tulemuste raporteerimine	Jah
UC13	Testija	Testimistsükli juhiste tarbimine	Jah
UC14	Kasutaja	Keskkonna tarbimine	Jah

Nimetatud kasutusjuhud on loetletud lühiformaadis (*brief level*), demonstreerides lahenduse põhiväärtuste loomet kuid jättes samal ajal suurema disaini- ja implementeerimisvabaduse [27]. Tavapärasema (*casual level*) detailsustase juures on näitena lahatud kasutusjuhtu number seitse, keskkonna taaskasutamine (**UC7**):

1. Juht kasutab Testlio platvormi
2. Juht valib vastava projekti
3. Juht valib projektis testimistsükli
4. Juht valib projekti keskkondade taaskasutuse
5. Juht valib projekti varasemalt üleslaetud keskkonna
6. Juht kinnitab valiku
7. Juhile kuvatakse testimistsüklisse lisatud keskkond

Alternatiiv (**UC7a**): Projektis pole ühtegi uut testimistsüklit. Juht loob uue testimistsükli ning seejärel valib projekti keskkondade taaskasutuse.
Alternatiiv (**UC7b**): Projekti pole ühtegi keskkonda varasemalt üles laetud. Juht algatab vajadusel keskkonna manuaalse lisamise kasutusjuhu (**UC3**).

Joonis 9 puhul on kasutusjuhtude tabeli (Tabel 4) baasil näidatud kasutusjuhtude diagrammi. Oluline on eraldi rõhutada, et kuna igas rollis on tegemist Testlio platvormi kasutajaga, on kasutaja roll võetud baasrolliks, st igal muul rollil on lisaks enda kasutusjuhtudele päritud kasutaja kasutusjuhud. Samuti on eraldi kasutaja tüübiks juhi roll, mis omakorda on projekti- ning testijuhi rollidele vahetuks baasrolliks. Lähtudes intervjuude (ptk 3.1.1) käigus selgunud tavapraktikast, on projekti- ning testijuhi vastutusalad projektiti tihti vahetuses. Sellest johtuvalt on diagrammil kujutatud süsteemiga interaktsioonis olevat üldisemat juhi rolli.



Joonis 9. Süsteemi kasutusjuhtude diagramm

3.2.3 Välise integratsiooni nõuded

Välise integratsiooni eesmärgiks on luua turvaline, skaleeruv ja hõlpsasti kasutatav integratsioonipunkt ettevõtte klientide jaoks. Integreeritavaks objektiks on kliendi keskkonnad: tarkvaralised koosted (*builds*) või veebilingid.

Funktsionaalsus (*functionality*):

1. Funktsionaalsed nõuded (*functional requirements*):
    a. **EIRFF1:** Integratsioon kliendipoolse pideva integratsiooni süsteemiga – võimaldades klientidel automaatset keskkondade publitseerimist Testlio süsteemi (kasutusjuht **UC1**). Publitseerimise tulemusena oleks keskkond koheselt rakendatav.

b. **EIRFF2:** Keskkonna kliendipoolse lisamise korral reaalajas teavituste saatmine (kasutusjuht **UC2**). Automaatsed sõnumid võimaldavad huvitatud osapooltel jälgida erinevate keskkondade publitseerimist. Eelistatud kommunikatsioonivahend ettevõttes on Slack [28].

2. Taaskasutatavus (*reusability*):
    a. **EIRFR1:** Integratsioonimehhanism peab olema klienditi taaskasutatav. See tähendab, et Testlio avaliku rakendusliidese kasutus erineb klienditi vaid rakenduspääse väärtuse ja päritava ressurssi aadressi võrra.
3. Turve (*security*):
    a. Rakendusepääse (*API token*) haldus:
        i. **EIRFS1:** Pääse formaadiks peab olema JWT või samaväärselt tugev autentimismehhanism (*JSON Web Token*) [29]. JWT pakub turvalist informatsioonivahetust osapoolte vahel ning võimaldab saata JSON objekti näol täiendavat informatsiooni [30].
        ii. **EIRFS2:** JWT signeerimiseks kasutatud salasõna peab olema erinev privaatsete ehk majasiseste teenuste salasõnast. Sellisel juhul ei mõjuta ühe salasõna leke teiste teenuste kättesaadavust.
        iii. **EIRFS3:** JWT salasõna peab olema vähemalt 64 juhusliku karakteritega sõne. See võimaldab kaitse jõhkra jõu rünnakute (*brute force attacks*) vastu [31].
        iv. **EIRFS4:** JWT salasõna peab olema turvaliselt hoiustatud ega kõrvalistele isikutele kättesaadav. Vastasel juhul poleks salasõna keerukusest kasu ning pahatahtlik osapool saaks ise rakenduspääsmeid signeerida.
        v. **EIRFS5:** JWT salasõna ei jagata kliendiga. Kliendile jagatakse tema jaoks relevantse sisuga ning Testlio poolt eelnevalt signeeritud räkenduspääse.
        vi. **EIRFS6:** Rakenduspääsmele peab olema rakendatud autoriseerimist – ühe avaliku rakenduspääsmega saab ligi ainult kindlatele ressurssidele.
        vii. **EIRFS7:** Kahe erineva aga kehtiva avaliku rakenduspääsega ei tohi ligi saada samadele ressursidele.

viii. **EIRFS8:** Avaliku rakenduspääsme aegumistähtaeg on seotud salasõna vahetamise elutsükliga, milleks on üks aasta. Paroolide perioodiline vahetus aitab tõsta süsteemi turvalisust.

ix. **EIRFS9:** Salasõna konfidentsiaalsuskao korral tuleb tunnistada kõik selle salasõnaga signeeritud rakenduspääsmed kehtetuks, genereerida uued pääsmed ning edastada need klientidele.

b. Rakendusliidese (*API*) disain:

i. **EIRFS10:** Avalik ja privaatne tee peavad olema rakendusliidese tasemel eristatud – niimoodi on võimalik vastavalt vajadusele avalik tee sulgeda ilma sisemisi teenuseid mõjutamata.

c. Infrastruktuuri disain:

i. **EIRFS11:** Avalik rakendusliides peab olema infrastruktuuri tasemel privaatsest eraldatud – sellisel juhul ei kandu potentsiaalsed jõudlusprobleemid ega turvaründed automaatselt üle sisemistele teenustele.

ii. **EIRFS12:** Avaliku rakendusliidese päringud peavad liikuma läbi veebitulemüüri (WAF – Web Application Firewall), mis seirab, filtreerib ja blokeerib sellest läbiminevat kogu avalikku veebiliiklust [32].

iii. **EIRFS13:** Loodava süsteemi ning kliendi ehitussüsteemi vaheline liiklus peab liikuma ainult üle krüpteeritud ja turvalise HTTPS veebiprotokolli.

d. Andmebaasi disain:

i. **EIRFS13:** Andmebaas ise ei tohi olla avalikult kättesaadav. Andmeid on võimalik pärida ainult läbi selleks ettenähtud teenuse.

ii. **EIRFS14:** Andmebaas peab olema krüpteeritud.

Kasutatavus (*usability*):

1. Dokumentatsioon (*documentation*):
    a. **EIRUD1:** Kliendile suunatud dokumentatsioon – lühike tehniline dokumentatsioon Testlio avaliku rakendusliidese integratsiooni jaoks (Lisa 3).
2. Lokaliseerimine (*localization*):
    a. **EIRUL1:** Rakendusliidese, andmebaasi, andmebaasi tabelite ja muu tehnilise teostuse puhul kasutatakse ainult inglise keelt.

3. Aruandlus (*reporting*):
    a. **EIRUR1:** Avaliku rakendusliidese kaudu lisatud keskkonnad raporteeritakse selleks ettenähtud uude Slack kanalisse.
    b. **EIRUR2:** Avaliku rakendusliidese krahhid peavad olema reaalajas raporteeritud ettevõtte arendusüksusele.
    c. **EIRUR3:** Süsteemi jõudlus peab olema reaalajas jälgitav.

Töökindlus (*reliability*):

1. Käideldavus (availability):
    a. **EIRRA1**: Avaliku rakendusliidese teenus peab olema kõrge käideldavusega (*high availability*) süsteem, mille korral on teenuse kumulatiivne maasoleku aeg aasta lõikes minimaalne [33]. Esialgseks eesmärgiks on kokkuleppeliselt 99,9% käideldavus, mille juures on maksimaalne lubatud maasoleku aeg kumulatiiivselt 8,77 tundi aasta kohta.
    b. **EIRRA2**: Avaliku rakendusliidese teenus peab olema kättesaadav ühelt kindlalt avalikult veebiaadressilt ning seda ei tohi edaspidi muuta ilma klientide kooskõlastuseta
2. Stabiilsus (*stability*):
    a. **EIRRS1:** Avaliku rakendusliidese teenusel peab olema rakendatud erinditöötlust (*exception handling*) ning teenuse mitte-sihtotspärane kasutus ei tohi põhjustada teenuse instantsi krahhe.
3. Taastuvus (*recoverability*):
    a. **EIRRR1:** Avaliku rakendusliidese teenuse teenuse instantsi krahhi korral käivitatakse koheselt uus instants
    b. **EIRRR2:** Andmebaasist tehakse regulaarselt varukoopiaid, vähemalt kord 24 tunni jooksul.

Jõudlus (*performance*):

1. Tõhusus (*efficiency*):
    a. **EIRPE1:** Avaliku rakedusliidese teenuse erinevate instantside koormuse jaotamiseks peab kasutama koormusjaoturit (*load balancer*). Koormuse ühtlane jaotamine aitab tagada teenuse koormuse paremat utiliseerimist.
2. Ressursikasutus (*resource consumption*):
    a. **EIRPR1:** Avaliku rakedusliidese teenuse mälu ja arvutusvõimsuse kasutus on piiratud konteineri enda piirangutega (ptk 4.1.3).

3. Skaleeritavus (*scalability*):
    a. **EIRPS1:** Koormuse kasvul lisatakse avaliku rakendusliidese teenuse instantse klastrisse juurde.
    b. **EIRPS2:** Koormuse vähenemisel eemaldatakse avaliku rakendusliidese teenuse instantse klastrist.
4. Suutlikkus (*capacity*):
    a. **EIRRC1:** Avaliku rakendusliidese teenuse instantste peab tootmiskeskkonnas (*production*) misiganes ajahetkel olema minimaalselt kaks.
    b. **EIRRC2:** Avaliku rakendusliidese teenus ei tohi jagada eksisteerivat platvormi vahemälu (*cache*), vaid selleks peab olema eraldiseisev mäluinstants. See võimaldab avaliku rakedusliidesele täiendavat jõudlust ning vähendab jagatud komponentidest tulenevaid riske.

Toetatavus (*supportability*):

1. Auditeeritavus (*auditability*):
    a. **EIRSA1:** Iga avaliku rakendusliidese kaudu lisatud keskkonna puhul peab olema nähtav lisaja identiteet ning teostuse aeg.
    b. **EIRSA2:** Avaliku rakendusliidese teenuse iga kasutuse (st sissetuleva veebipäringu) kohta peab tekkima logikirje süsteemi.
2. Konfigureeritavus (*configurability*):
    a. **EIRSC1:** Avaliku rakendusliidese teenuse ehitamise hetkel peab teenuse skoop (st avalik või privaatne) olema konfigureeritav. Ehitussüsteemides kasutatakse selleks vastavaid keskkonna muutujaid (*environment variable*), läbi mille on võimalik ehitatavat teenust seadistada [34, 35].
3. Testitavus (*testability*):
    a. **EIRST1:** Teenust peab olema võimalik terve süsteemi tasemel testida eelnevalt tootmiskeskkonda minekut.

3.2.4 Sisemise integratsiooni nõuded

Sisemise integratsiooni eesmärgiks on tagada edukas kooseksisteerimine ettevõtte olemasolevate süsteemidega. Integratsioonipunktiks on Testlio platvorm, teised mikroteenused ning kasutajaliides.

Funktsionaalsus (*functionality*):

1. Funktsionaalsed nõuded (*functional requirements*):
    a. **IIRFF1:** Testimistsüklisse lisatud keskkonnad peavad olema loetletud testimistsükli raportil (**kasutusjuht UC12**). Platvormil eksisteeriv raport koondab enda alla testimistsüklisse lisatud keskkonnad ja nende andmed. Raporti mõistes on keskkonna lisamise mehhanism sekundaarne: kui keskkond lisati mingil viisil testimistsüklisse ning oli aluseks testimisele, peab see kajastuma raportil.
    b. **IIRFF2:** Testimistsüklisse lisatud keskkonnad peavad olema loetletud testimistsükli juhistel (**kasutusjuht UC13**). Platvormil eksisteerivad juhised on suunatud tsüklis osalevatele testijatele ning loodava süsteemi kaudu lisatud keskkond ja selle andmed peavad olema testijatele nähtavad.
    c. **IIRFF3:** Testimistsükli juhiste kontekstis loetletud keskkond peab olema allalaetav või avatav veebibrauseri aknas (**kasutusjuht UC14**). Esimesel juhul on tegemist failidega, teisel juhul aga veebilinkidega.
2. Taaskasutatavus (*reusability*):
    a. **IIRFR1:** Platvorm peab uue süsteemi kaudu lisatud keskkondade puhul taaskasutama uue süsteemi enda funktsionaalsust. Keskkond ja selle andmed päritakse uuest süsteemist ning kuvatakse platvormil.
3. Turve (*security*):
    a. Rakendusliidese (*API*) disain:
        i. **IIRFS1**: Platvorm ja teised sisemised teenused peavad keskkondade pärimiseks kasutama sisemisi teid, st sisemised päringud ei tohi läbida avalikku rakendusliidest.
        ii. **IRFS2**: Sisemine liiklus ei tohi liikuda üle avaliku interneti vaid üle turvatud võrkude.
    b. Andmebaasi disain:

i. **IIRFS3:** andmebaasis peab olema eraldi teenuspõhised kasutajad, mille õigused on kohandatud vastavalt andmebaasi skeemapõhisele ligipääsule. See tähendab, et läbi loodava teenuse pole võimalik andmebaasist lugeda teenuse väliseid teisi andmebaase ega sõltumatuid tabeleid.
ii. **IIRFS4:** andmebaasi teenuskasutajatele peab olema rakendatud vähimate õiguste printsiipi (*principle of least privilege*) [36]
iii. **IIRFS5:** uue teenuse andmebaas peab olema eraldatud eksisteerivast platvormi andmebaasist**.**

Kasutatavus (*usability*):

1. Dokumentatsioon (*documentation*):
    a. **IIRUD1:** Majasisene dokumentatsioon – Confluence'i dokumentatsioon peab andma ülevaate uue teenuse iseloomust ning integratsioonist teiste majasiseste teenustega.
2. Aruandlus (*reporting*):
    a. **IIRUR1:** Iga privaatse rakendusliidese kaudu lisatud või muudetud keskkonna puhul peab olema nähtav lisaja/muutja identiteet ning teostuse aeg.
    b. **IIRUR2:** Privaatse rakendusliidese krahhid peavad olema reaalajas raporteeritud ettevõtte arendusüksusele.
    c. **IIRUR3:** Süsteemi jõudlus peab olema reaalajas jälgitav.

Töökindlus (*reliability*):

1. Käideldavus (availability):
    a. **IIRRA1**: Uus loodav teenus peab sisemiselt olema kättesaadav läbi teenuste avastamise mudeli (ptk 4.4.2).
2. Stabiilsus (*stability*):
    a. IIRRS1: Privaatse rakendusliidese teenusel peab olema rakendatud erinditöötlust (*exception handling*) ning teenuse mitte-sihtotspärane kasutus ei tohi põhjustada teenuse instantsi krahhe.
3. Taastuvus (*recoverability*):
    a. IIRRR1: Privaatse rakendusliidese teenuse teenuse instantsi krahhi korral käivitatakse koheselt uus instants.
    b. **IIRRR2:** Andmebaasist tehakse regulaarselt varukoopiaid, vähemalt kord 24 tunni jooksul.

Jõudlus (*performance*):

1. Tõhusus (*efficiency*):
    a. **IIRPE1:** Privaatse rakendusliidese teenuse erinevate instantside koormuse jaotamiseks peab kasutama koormusjaoturit (*load balancer*). Koormuse ühtlane jaotamine aitab tagada teenuse koormuse paremat utiliseerimist.
2. Ressursikasutus (*resource consumption*):
    a. **IIRPR1:** Privaatse rakendusliidese teenuse mälu ja arvutusvõimsuse kasutus on piiratud konteineri enda piirangutega (ptk 4.1.3).
3. Skaleeritavus (*scalability*):
    a. **IIRPS1:** Koormuse kasvul lisatakse privaatse rakendusliidese teenuse instantse klastrisse juurde.
    b. **EIRPS2:** Koormuse vähenemisel eemaldatakse privaatse rakendusliidese teenuse instantse klastrist.
4. Suutlikkus (*capacity*):
    a. **IIRRC1:** Privaatse rakendusliidese teenuse instantste peab tootmiskeskkonnas (*production*) misiganes ajahetkel olema minimaalselt kaks.
    b. **IIRRC2:** Lugemiskoormus peaks olema suunatud lugemiseks mõeldud andmebaasi instantsi pihta. Kirjutamiskoormus (lisamised, kustutamised, muutmised) kirjutamiseks mõeldud andmebaasi instantsi pihta. Operatsioonipõhine (lugemine *versus* kirjutamine) erisus aitab optimeerida üldist andmekoormust, andmebaasi instantside jõudlust ning ka ettevõtte pilvekulusid.

Toetatavus (*supportability*):

1. Auditeeritavus (*auditability*):
    a. **IIRSA1:** Privaatse rakendusliidese teenuse iga sisemise kasutuse (st sissetuleva veebipäringu) kohta peab tekkima logikirje süsteemi.
2. Konfigureeritavus (*configurability*):
    a. **IIRSC1:** Privaatse rakendusliidese teenuse ehitamise hetkel peab teenuse skoop (st avalik või privaatne) olema konfigureeritav. Ehitussüsteemides kasutatakse selleks vastavaid keskkonna muutujaid (*environment variable*), läbi mille on võimalik ehitatavat teenust seadistada [34, 35][env vars].
3. Testitavus (*testability*):
    a. **IIRST1:** Teenust peab olema võimalik terve süsteemi tasemel testida eelnevalt tootmiskeskkonda minekut.

3.2.5 Kasutajate nõuded

Kasutaja nõuete eesmärgiks on formuleerida loodava süsteemi poolt oodatav uus kasutajaliidese funktsionaalsus. Kasutajaliidese kontekstis mõeldakse mitte-platvormi ehk eraldiseisvat veebilehte aadressil https://app2.testlio.com (ptk 4.1.5).

Funktsionaalsus (*functionality*):

1. Funktsionaalsed nõuded (*functional requirements*):
    a. **URFF1:** Keskkondi peab olema võimalik manuaalselt süsteemi lisada (**kasutusjuht UC3**). Testlio avaliku rakendusliidese integratsioon võtab klienditi erinevalt aega ning süsteem peab pakkuma võimalust käsitsi keskkondade lisamist läbi kasutajaliidese.
    b. **URFF2:** Keskkonna andmete muutmine (**kasutusjuht UC4**). Keskkonnaga kaasneb hulk andmeid: nimi, silt, versioon, tüüp, loomiskuupäev jpm (ptk 4.3.1). Versioon (*version*) ja silt (*tag*) on ajalooliselt olnud juhi rolli poolt muudetavad keskkonna tunnused ning uus süsteem peaks seda toetama.
    c. **URFF3:** Keskkonna testimistsüklist kustutamine (**kasutusjuht UC5**). Varasemalt tsüklisse lisatud keskkondi peab olema võimalik vajadusel ka tsüklist. Eemaldada. See tähendab testimistsükli ja keskkonna vahelise seose kustutamist.
    d. **URFF4:** Keskkondi peab olema võimalik süsteemist kustutada (**kasutusjuht UC6**). Varasemalt süsteemi lisatud keskkondi peab olema võimalik vajadusel ka süsteemist eemaldada. See tähendab projekti ja keskkonna vahelise seose kustutamist.
    e. **URFF5:** Süsteemi lisatud keskkondi peab olema võimalik taaskasutada (**kasutusjuht UC7**), ehk luua testimistsükli ja varasemalt üleslaetud keskkonna vaheline seos ilma keskkonda uuesti üles laadimata.
    f. **URFF6:** Süsteemis olemasolevaid keskkondi peab olema võimalik keskkonna andmete alusel filtreerida (**kasutusjuht UC8**). Keskkonna otsing selle nime, versiooni, sildi või loomiskuupäeva järgi aitab juhil kiiremini leida tema jaoks relevantset keskkonda.
    g. **URFF7:** Süsteemis olemasolevaid keskkondi peab olema võimalik keskkonna andmete alusel sorteerida (**kasutusjuht UC9**). Sarnaselt filtreerimisele aitab

keskkonna otsing selle nime, versiooni, sildi või loomiskuupäeva järgi kiiremini tuvastada relevantset keskkonda.

h. **URFF8:** Projekti keskkondade jagamine peab olema vajadusel ühesuunaliselt seadistatav üle erinevate projektide (**kasutusjuht UC10**). Ühesuunaline jagamine võimaldab vaid ühel projektil taaskasutada teise keskkondi.

i. **URFF9:** Projekti keskkondade jagamine peab olema vajadusel kahesuunaliselt seadistatav üle erinevate projektide (**kasutusjuht UC11**). Kahesuunaline jagamine võimaldab mõlemal projektil taaskasutada teineteise keskkondi.

j. **URFF10:** Kasutajaliideses loetletud keskkond peab olema allalaetav või avatav veebibrauseri aknas (**kasutusjuht UC14**). Esimesel juhul on tegemist failidega, teisel juhul aga veebilinkidega.

2. Turve (*security*):
    a. **URFS1:** Erinevate kasutusjuhtude realiseerimisel süsteemis tuleb alati kontrollida vastava kasutaja rolli. Enamik uue süsteemi funktsionaalsusest on ette nähtud juhi rollile ning tavakasutajal ei tohiks olla sellele ligipääsu.
    b. **URFS2:** Projekti keskkondade seadistusel saab juht jagada keskkondi ainult üle projektide kus tal on samuti juhi roll.
    c. **URFS3**: Kasutaja ei tohi jääda kasutajaliidesesse igavesti sisse logituks. Piisava aja jooksul tegevusetus olekus viibimine peab lõppema kasutaja automaatse väljalogimisega süsteemist.

Kasutatavus (*usability*):

1. Dokumentatsioon (*documentation*):
    a. **URUD1:** Majasisene dokumentatsioon – Confluence'is dokumenteeritud kasutajaliidese põhivood võimaldades kasutajatel uut funktsionaalsust võimalikult hõlpsasti avastada.
2. Lokaliseerimine (*localization*):
    a. **URUL1:** Kasutajaliidesel peab olema minimaalselt ingliskeelne tugi.
3. Aruandlus (*reporting*):
    a. **URUR1:** Iga kasutajaliidese kaudu manuaalselt lisatud või muudetud keskkonna puhul peab olema nähtav lisaja või muutja identiteet ning teostuse aeg.
    b. **URUR2:** Kasutajaliidese krahhid peavad olema reaalajas raporteeritud ettevõtte arendusüksusele.

4. Esteetika (*aesthetics*):
    a. **URUA1:** Kasutajaliides peab olema joondatud Testlios kasutatava disainiteegiga – Ant Design [37].

Töökindlus (*reliability*):

1. Käideldavus (availability):
    a. **URRA1:** Keskkonnad peab olema kättesaadavad ka mobiiliseadmetest**.**
    b. **URRA2:** Kui keskkonna tüübiks on mobiilirakendus, ehk kui tegemist on tarkvaralise koostega kas Android või iOS mobiiltelefonidele, peab keskkond olema paigaldatav vastavale seadmele. iOS faile on vaja eelnevalt signeerida: failide signeerimine vastava sertifikaadiga on Apple poolt kehtestatud kohustuslik turvanõue, kaitsmaks mobiilirakendusi tervikluskao eest [38]. Esmane signeerimine toimub kliendi ehitussüsteemis kliendi sertifikaadiga. Resigneerimine ehk uuesti signeerimine on iOS faili signeerimine Testlio Apple sertifikaadiga. Protsessi eesmärk on tagada, et fail oleks paigaldatav misiganes iOS seadme peale kuna ettevõttel puudub kontroll klientide poolt kasutatavate sertifikaatide üle.
2. Stabiilsus (*stability*):
    a. **URRS1:** Kasutajaliidesel peab olema rakendatud erinditöötlust (*exception handling*) ning kasutajaliidese mitte-sihtotspärane kasutus ei tohi põhjustada kasutaja brauseri krahhe.
3. Taastuvus (*recoverability*):
    a. **URRR1**: Kasutajaliidese kaudu lisatud keskkonnad peavad olema kao korral taastatavad.
4. Suutlikkus (*capacity*):
    a. **URRC1:** Uus süsteem (v3) peab olema vähemalt sama kiire kui väljavahetatav vana süsteem (v2).

Jõudlus (*performance*):

1. Tõhusus (*efficiency*):
    a. **URPE1:** Kasutajaliidese tehniline majutus peab olema teenusepakkuja sisuedastusvõrguga (*CDN – Content Delivery Network*) kaetud. See võimaldab serveerida sisu kasutajale kõige lähedamal olemas geograafilises sisuedastusvõrgu punktis [39].
2. Ressursikasutus (*resource consumption*):

a. **URPR1**: Kasutajaliidese ressursikasutus on piiratud kasutaja brauseri sätetega.

3. Skaleeritavus (*scalability*):
    a. **URPS1**: Kasutajaliidese skaleeritavus on piiratud sisuedastusvõrgu enda võimekusega (ptk 4.1.2).

Toetatavus (*supportability*):

1. Auditeeritavus (*auditability*):
    a. **URSA1:** Iga manuaalselt lisatud keskkonna ning selle andmete muutmise kohta peab süsteemi kirje, mille järgi on võimalik nii ajahetke kui ka reaalset kasutajat tuvastada.
2. Jätkusuutlikkus (sustainability):
    a. **URSS1:** Vana keskkondade teenus (v2) peab olema toetatud seni kuni sealsed keskkonnad on veel kasutuses. V2 tootmiskeskkonnast mahavõtmine eeldab äripoole valmisolekut.
    b. **URSS2:** Kasutajaliides peab toetama vana keskkondade teenusest (v2) pärit olevate keskkondade haldamist (andmete muutmist ja keskkonna kustutamist). Keskkonna uuesti lisamise korral lisatakse see uude süsteemi (v3).
3. Testitavus (*testability*):
    a. **URST1:** Kasutajaliidest peab olema võimalik kasutajate poolt terve süsteemi tasemel testida eelnevalt tootmiskeskkonda.

3.3 Edukuse mõõdupuu

Edukuse mõõdupuu kirjeldab süsteemi vastuvõtmise strateegiat ning selle kasutusulevõttu. Eesmärk on kehtestada tootmiskeskkonna kvaliteedikvoodid ning süsteemi integratsiooniks vajalikud pidepunktid.

3.3.1 Süsteemi vastuvõtmine

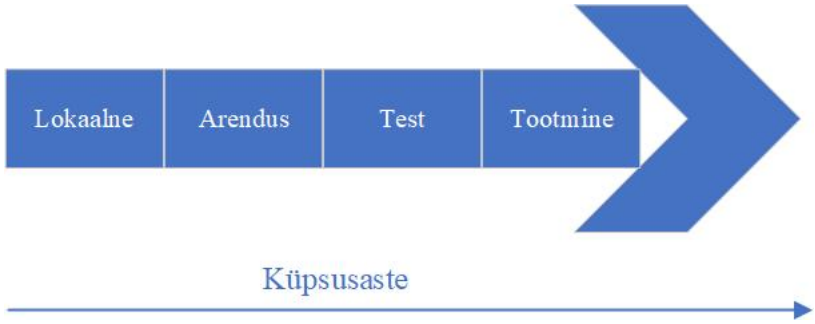
Uuendatud süsteem antakse üle Testlio äripoolele. Äripoolse vastuvõtu eest vastutab teenuste tarne (*service delivery*) üksus (Joonis 1). Eduka vastuvõtu aluseks on üksuse poolt teostatud vastuvõtutestid (*acceptance tests*), mis baseeruvad varasemalt defineeritud kasutusjuhtudele (ptk 3.2.2) ning süsteemile esitatud nõuetele (ptk 3.2.3, ptk 3.2.4 ja ptk 3.2.5). Kokkuleppeliselt peab äripoolelt vastuvõtuprotsessis osalema vähemalt üks testi- ja üks projektijuht, kuid osalejate arvu ei piirata. Vastuvõtu perioodiks on esialgselt planeeritud kaks nädalat.

Arendusüksuse vastutuseks on tagada vastuvõtutestideks vajalik stabiilne testkeskkond, toetada äripoolt tehnilistes küsimustes ning adresseerida süsteemi kontekstis raporteeritud defekte. Testimiskeskkond peab süsteemi vastuvõtuks rahuldama järgmisi nõudeid:

1. **Testitavus:** keskkond peab omama vastuvõtutestimise subjektiks olevat funktsionaalsust.
2. **Stabiilsus:** keskkonna funktsionaalset seisu ei muudeta ilma äripoole kooskõlastuseta.
3. **Võrdväärsus:** testimiskeskkonna seadistus, mh andmed, peavad toetama tootmiskeskkonnaga võrreldavat kogemust. Mida väiksem on erinevus nimetatud kahe keskkonna vahel, seda lihtsam on korreleerida testimiskeskkonna kvaliteeti tootmiskeskkonnaga.

Reliisihalduse peamine eesmärk on tagada tootmiskeskkonna terviklus ning korrektsete komponentide evitus [40]. Joonis 10 visualiseerib ettevõttes kasutusel olevat arendusprotsessi, mille järgi läbivad muudatused neli eri küpsusastet. Lokaalne arenduskeskkond on iga arendaja jaoks individuaalselt eksisteeriv keskkond, mida käivitatakse ainult füüsilises tööarvutis. Edasised keskkonnad on ettevõtte üleselt jagatud pilveressurssid.

Ühest keskkonnast teisse liikumise eelduseks on pideva integratsiooni ehitusahelas kehtestatud kvaliteedikvootidele vastamine. Eduka süsteemi vastuvõtmise järel paigaldatakse lahendus globaalsesse tootmiskeskkonda.



Joonis 10. Erinevate arenduskeskkondade küpsusaste

3.3.2 Pilootprojekt

Kliendi keskkondade suurema väärindamise aluseks on integratsioon Testlio uue süsteemiga. Läbi avaliku rakendusliidese saatetud keskkondade loodakse uued väärtused:

1. **Vabadus:** Kliendid võivad saata testimiseks mõeldud keskkondi, misiganes kliendile sobival ajal. Keskkondade vastuvõtmine ei sõltu enam testi- või projektijuhi tööst.
2. **Kokkuhoid:** varasemalt süsteemi sisenenud keskkondi pole vaja uuesti üles laadida. Samuti on võimalik eksisteerivaid keskkondi filtreerida ja otsida erinevate tunnuste järgi.
3. **Selgus:** süsteemi sisenevad keskkonnad on läbi automaatsete teavituste kohe nähtavad kõikidele osapooltele, koos vajaliku tehnilise infoga.

Esmase integratsiooni algatajaks on iga kliendi vastav Testlio poolne projektijuht. Kliendile pakutav tehniline tugi sisaldab uue süsteemiga integreerimiseks vajalikke juhiseid (Lisa 3). Süsteemi pilootprojekti raames valitakse koostöös äripoolega üks sobiv klient. Projekti edukuse aluseks on järgmised versapostid:

1. **Ehitussüsteemi integratsioon:** läbi avaliku rakendusliidese saadetakse keskkonnad Testlio süsteemi.
2. **Süsteemi integratsioon:** uued keskkonnad lisatakse automaatselt vastava kliendi projekti keskkondade kogumisse. Uued keskkonnad on otsitavad, filtreeritavad ja taaskasutatavad läbi süsteemi kasutajaliidese.
3. **Platvormi integratsioon:** uued keskkonnad on nähtavad platvormil kuvatavatel testimisjuhistel ja – raportitel. Platvormi kasutaja seisukohast pole tähtis millist

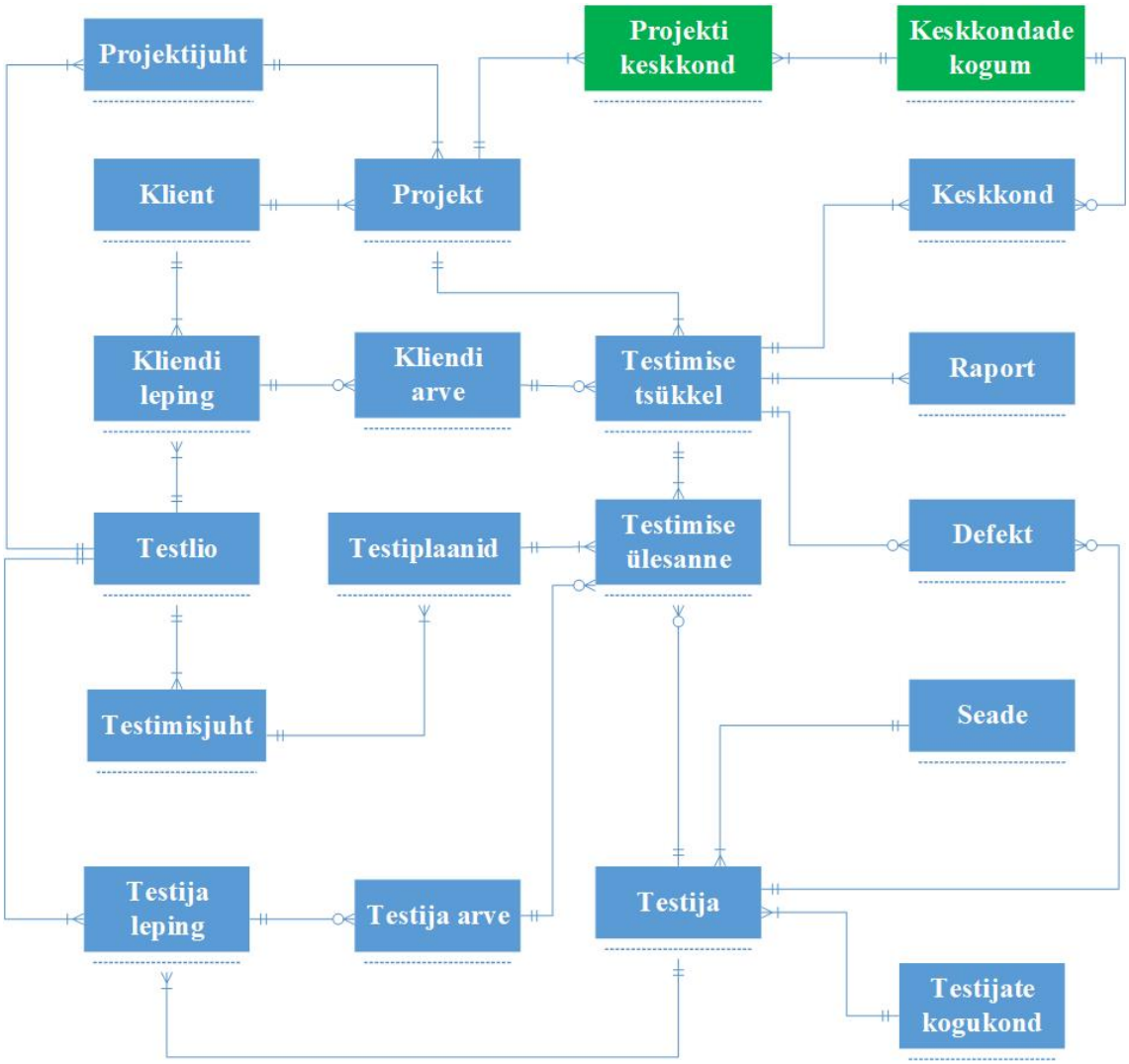
teed pidi sisenesid keskkonnad Testlio ökosüsteemi. Eduka integratsiooni korral on toetatud nii vana kui ka uus süsteem.

3.4 Valdkonna kaardistus (TO-BE)

TO-BE sektsiooni eesmärk on formuleerida nii äridomeeni kui ka -protsesside soovitud seis tulevikus, tuginedes varasemalt kirjeldatud AS-IS hetkeseisule (ptk 3.1) ning nõuetele (ptk 3.2).

3.4.1 Äridomeen (TO-BE)

Joonis 11 kujutab parendatud äridomeeni, kus on sinisega tähistatud varasemad AS-IS olemid, rohelisega uued TO-BE olemid. Äridomeeni mudeli täiendamisel on lähtutud esialgsest AS-IS domeenimudelist (ptk 3.1.2).

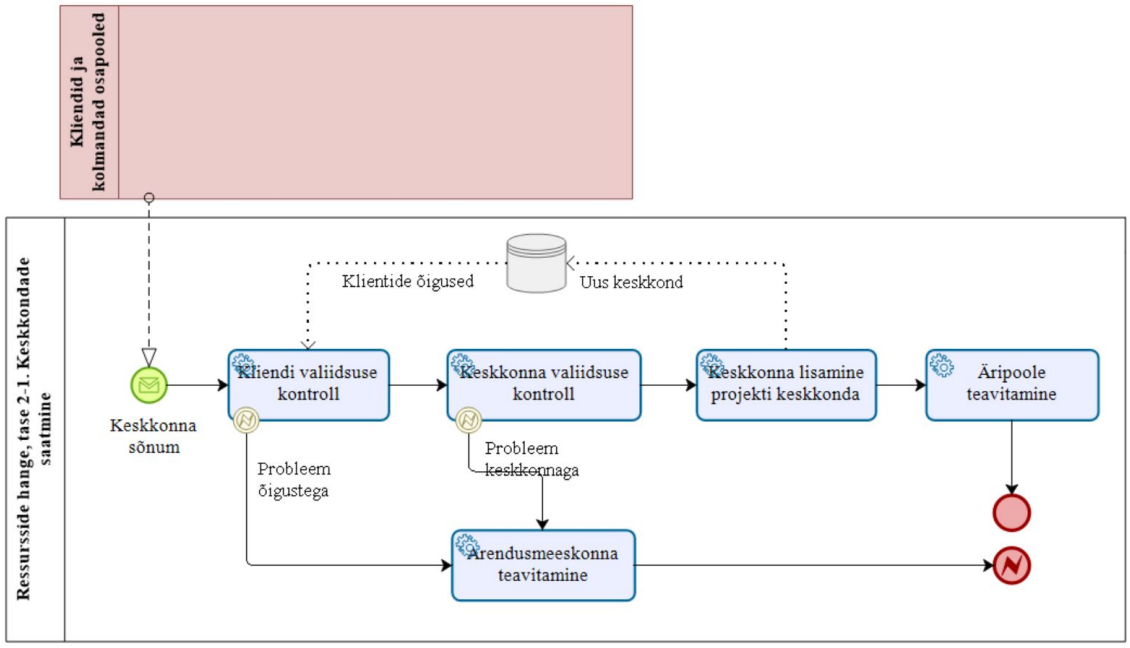


Joonis 11. Testlio TO-BE äridomeen

Täiendavad olemid – vastavalt „Projekti keskkond“ ja „Keskkondade kogum“ – loovad keskkondadele testimise tsüklite ülese elutsükli ning võimaldab keskkondi hallata eraldiseisvate artefaktidena. Igal projektil on täpselt üks primaarne keskkondade kogum, mis määrab asukoha kuhu projekti keskkondi kirjutatakse nii automaatse (nõue **EIRFF1**) kui ka manuaalse (nõue **URFF1**) lisamise korral. Primaarne kogum on ühtlasi ka esmane kogum, kust projekt keskkondi loeb (nõuded **IIRFF1** kuni **IIRFF3** ning **URFF2** kuni **URFF10**) ning kuhu kirjutatakse keskkonna muudatusi (nõuded **URFF2** kuni **URFF4**). Samuti võib projektil olla null kuni mitu sekundaarset kogumit millest on projektil võimalik keskkondi täiendavalt lugeda, pakkudes seeläbi keskkondade jagamist üle mitme projekti (nõuded **URFF8** ja **URFF9**). Primaarsus ja sekundaarsus on keskkonna kogumi omadused konkreetse projekti keskkonna suhtes. Sekundaarsed kogumid on oma olemuselt teiste projektide primaarsed kogumid, kuhu on konkreetsel projektil antud lugemisõigus.

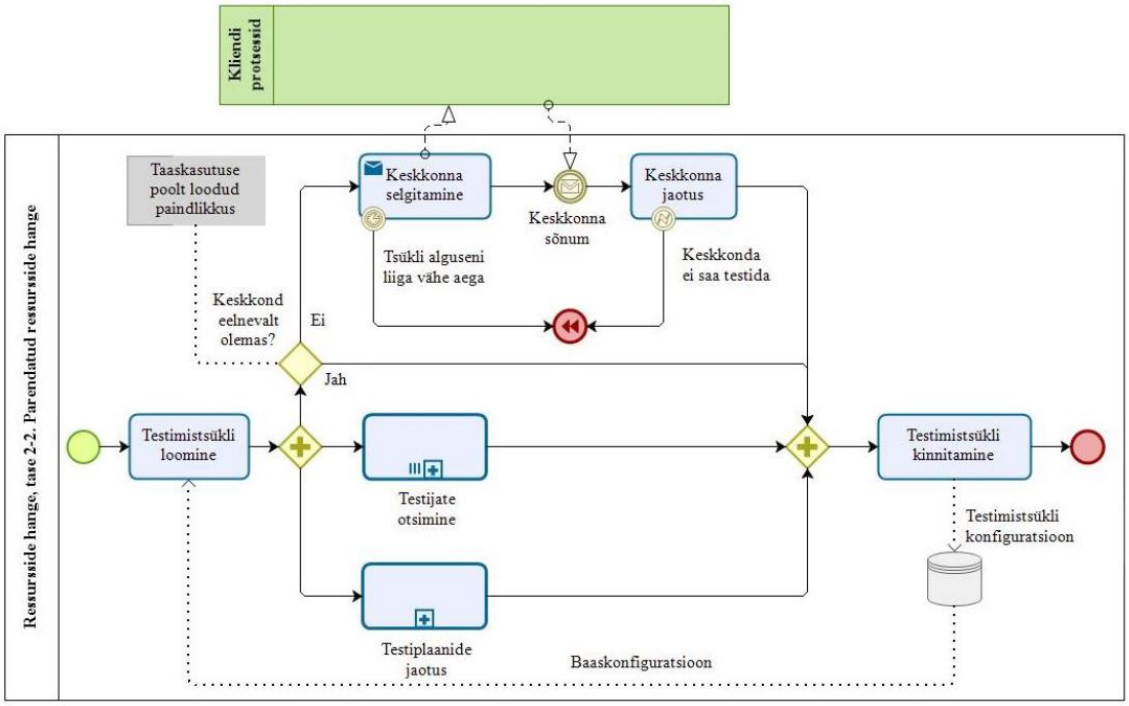
3.4.2 Äriprotsess (TO-BE)

Äriprotsessi täiendamisel on lähtutud esialgsest AS-IS äriprotsessist (ptk 3.1.3). Parendused seisnevad AS-IS ressursside hanke (Joonis 5) alamprotsessi automatiseerimise ning suurema paindlikkuse tagamisel. Fookuses on keskkondade jaotuse probleem (ptk 3.1.4) ning võrreldes varasemaga on ressursside hange eraldatud kahte iseseisvasse alamprotsessi, mida kajastavad vastavalt Joonis 12 ja Joonis 13.



Joonis 12. Testlio kliendi TO-BE testimisprotsessi ressursside hange, tase 2-1

Avalik rakendusliides loob kliendile vabaduse Testlio süsteemidesse automaatselt saata keskkondi läbi kliendile kuuluva ehitussüsteemi integratsiooni (nõue **EIRFF1**). Igal kliendil on oma unikaalne ja autoriseeritud rakenduspääse (nõue **EIRFS1**), millega tal on õigus saata keskkondi vaid oma projekti keskkondade kogumisse (nõue **EIRFS6**). Avalik rakendusliides võimaldab keskkondade või misiganes muu sõnumi saatmist ka kolmandatele osapooltele, kuid täiendav kliendi valiidsuse kontroll lubab selle funktsionaalsuse kasutust vaid Testlio klientidele.



Joonis 13. Testlio kliendi TO-BE testimisprotsessi ressursside hange, tase 2-2

Parendatud ressursside hanke äriprotsess (Joonis 13) võimaldab vähendada keskkondade hankest tulenevaid riske. Avaliku rakendusliidese eduka integratsiooni korral on vajalik keskkond süsteemis olemas enne vastava testimistsükli loomist (nõue **EIRFF1**). Täiendava paindlikkuse aitab tagada keskkondade taaskasutamine läbi projekti keskkondade ja nende kogumite – varasemalt üles laetud keskkondi saab kasutada ka tulevastes testimistsüklites (nõuded **EIRFF1** ja **URFF1**). Näiteks kui eesmärgiks on erineva fookusega testimistsüklid sama keskkonna tarbeks või varasemalt luhtunud testimistsükli kordamine. Taaskasutuse subjektiks kõik süsteemi sisenenud keskkonnad üle erinevate kanalite, mh ka käsitsi üles laetud keskkonnad.

4 Arhitektuur

Arhitektuuri eesmärk on tuginedes analüüsile luua terviklik süsteemidisain, toestada uue pilveteenuse väljatöötamist ja liidestamist olemasolevate teenustega, kirjeldada ettevõtte kontekstis uudse avaliku rakendusliidese tööpõhimõtted ning seda kõike viisil, mis oleks firma ärimahtude juures turvaline ja skaleeruv.

4.1 Infrastruktuuri arhitektuur (AS-IS)

Testlio testimisteenuse strateegilisel hinnakujundusel on üheks oluliseks kulukomponendiks omatava IT tartistu arendus- ja opereerimiskulud. Pilvepõhiste lahenduste eduka kasutuselevõtu korral on ettevõtetel võimalus vähendada opereerimiskulusid, parendada klientidega interaktsioone ning lühendada uute toodete/teenuste turuletuleku aega [41]. Sellest johtuvalt on Testlio tehniline infrastruktuur laias osas pilvetehnoloogiate abil lahendatud. Peamisteks erandiks on interneti teenusepakkuja poolt pakutud ruuterid, kommutaatorid (*switches*), pääsupunktid (*access points*) ning muu ettevõtte füüsiliste kontoriruumide teenindamisega seotud seadmed.

Maailma ühe suurema uurimis- ja konsultatsioonifirma Gartner andmeteil on 2019 seisuga juhtivaks pilvepõhise infrastruktuuri pakkujaks Amazon Web Services [42]. Joonis 14 kujutab Gartneri poolt hindamisel lähtutud maagilise kvadrandi (*magic quadrant*) metodoloogia tulemusi, mis jaotab tehnoloogiate pakkujad nelja erineva tururolli vahel [43]. Tuginedes turujaotusele on ka Testlios kasutusel AWS (*Amazon Web Services*).

Joonis 14. Pilvepõhise infrastruktuuri pakkujate paigutus turul, Gartner 2019

4.1.1 Infrastruktuuri pilvekomponendid

Terve ettevõtte infrastruktuuri lahkamine oleks antud lõputöö skoobis liiga mahukas ning osaliselt tähtsusetu. Fookuses on taristu üldine abstraktne vaade (tase 0), mida seejärel kitsendatakse alamvaatesse ainult relevantsete komponentide lõikes (tase 1). Tabel 5 kirjeldab ettevõttes kasutusel olevate erinevate pilvelahenduste üldised kategooriad.

Tabel 5. Testlio näitel kasutusel olevate pilvelahenduste kategooriad

ID	Kategooria	Tehniline näide	Kommentaar
CC1	Veebilehed (*web pages*)	JavaScript, HTML ja CSS	Eessüsteemid (*front-end applications*), mis suhtlevad kasutajaga vahetult
CC2	Serveriga veebiteenused (*web services*)	Docker, AWS ECS ja AWS EC2	Tagasüsteemid (*back-end applications*), mille käitlemiseks rakendatakse veebiservereid
CC3	Serverivabad veebiteenused (*serverless web services*)	Serverless ja AWS Lambda	Tagasüsteemid (*back-end applications*), kus pilveteenuse pakkuja toimib ise serverina [44][Serverless]
CC4	Andmebaasid (*databases*) ja vahemälud (*caches*)	MySql, DynamoDB ja Redis	Relatsiooniliste (*SQL*) ja mitte-relatsiooniliste (NoSQL) andmete hoiustamine
CC5	Komplimenteerivad pilveressursid	AWS Route 53 (DNS) ja AWS ELB (koormusjaotur)	Pilveteenuse pakkuja erinevad komponendid mis aitavad luua terviklikke lahendusi

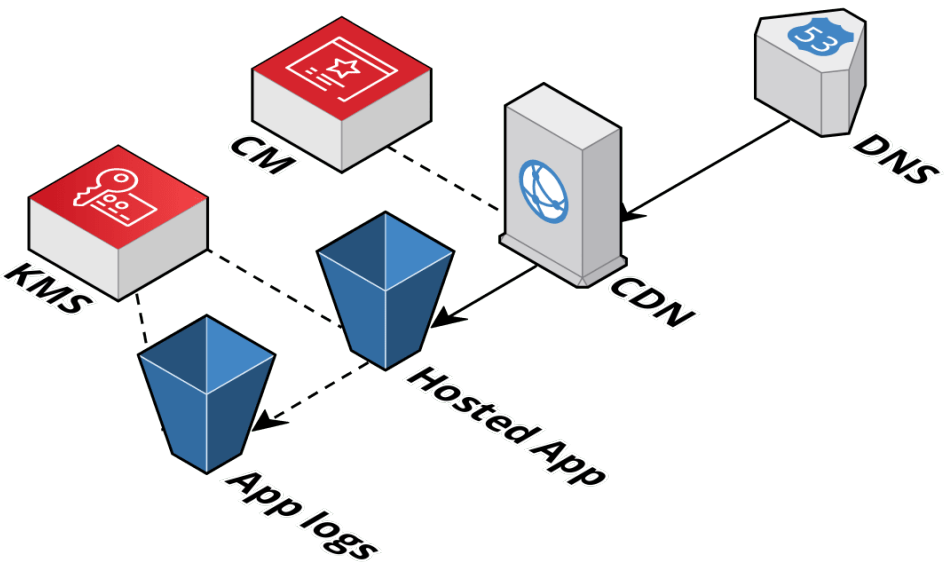
Pilvelahenduste arhitektuuri sidusama vaate loomise huvides pole andmebaase, vahemälusid ega komplimenteerivaid pilveressursse vaadeldud isolatsioonis vaid eelnevate kategooriate kompositsioonis. Järgnevates peatükkides kirjutatakse lahti kategooriate üks kuni kolm abstraktsed vaated, mis käitlevad endas vastava pilvelahenduse üldist arhitektuuri. Jooniste teostusel on kasutatud CloudCraft tarkvara [45]. CloudCraft on üks AWS pilveplatvormi arhitektuuri joonistamise ametlikke tööriiste ning rakendab kokkulepitud graafilist sümboolikat [46].

4.1.2 Veebilehtede vaade

Joonis 15 kujutab veebilehtede taristu abstraktset arhitektuuri, mille järgi on ettevõttes arendatud sisuliselt kõik arendatud veebilehed. Mõistagi on iga konkreetse veebilehe puhul pilves olevate ressursside täpne nimetus ja seadistus üksteisest erinev, kuid loetletud komponendid, nende funktsioonid ning omavaheline suhtlus on läbivalt sarnane. Tabel 6 loetleb kõik abstraktses lahenduses kasutatavad ressursid.

Tabel 6. Testlio veebilehtede abstraktse taristu ressursid

ID	Ressurss	AWS teenus	Funktsioon
CCW1	*DNS*	*Route 53*	*Domain Name System* ehk domeeninimede süsteem; domeeni haldamine ja sisuedastusvõrgu (CDN) sidumine kasutajasõbralikemate internetiaadressidega [47]
CCW2	*CDN*	*CloudFront*	*Content Delivery Network* ehk sisuedastusvõrk; võimaldab kasutajatele veebisisu kiiremat serveerimist [48]
CCW3	*CM*	*Amazon Certificate Manager*	*Certificate Manager* ehk sertifikaatide haldur; võimaldab turvalist internetiliiklust üle HTTPS protokolli [49]
CCW4	*Hosted App*	*Simple Storage Service (S3)*	Veebilehe koodi hoiustamine ja serveerimine [50]
CCW5	*App logs*	*Simple Storage Service (S3)*	Sihtkoht veebilehe enda logide kirjutamiseks ja hoiustamiseks [50]
CCW6	*KMS*	*Key Management Service*	S3's hoiustatud andmete krüpteerimine ja dekrüpteerimine [51]



Joonis 15. Testlio veebilehtede taristu AS-IS abstraktne arhitektuur

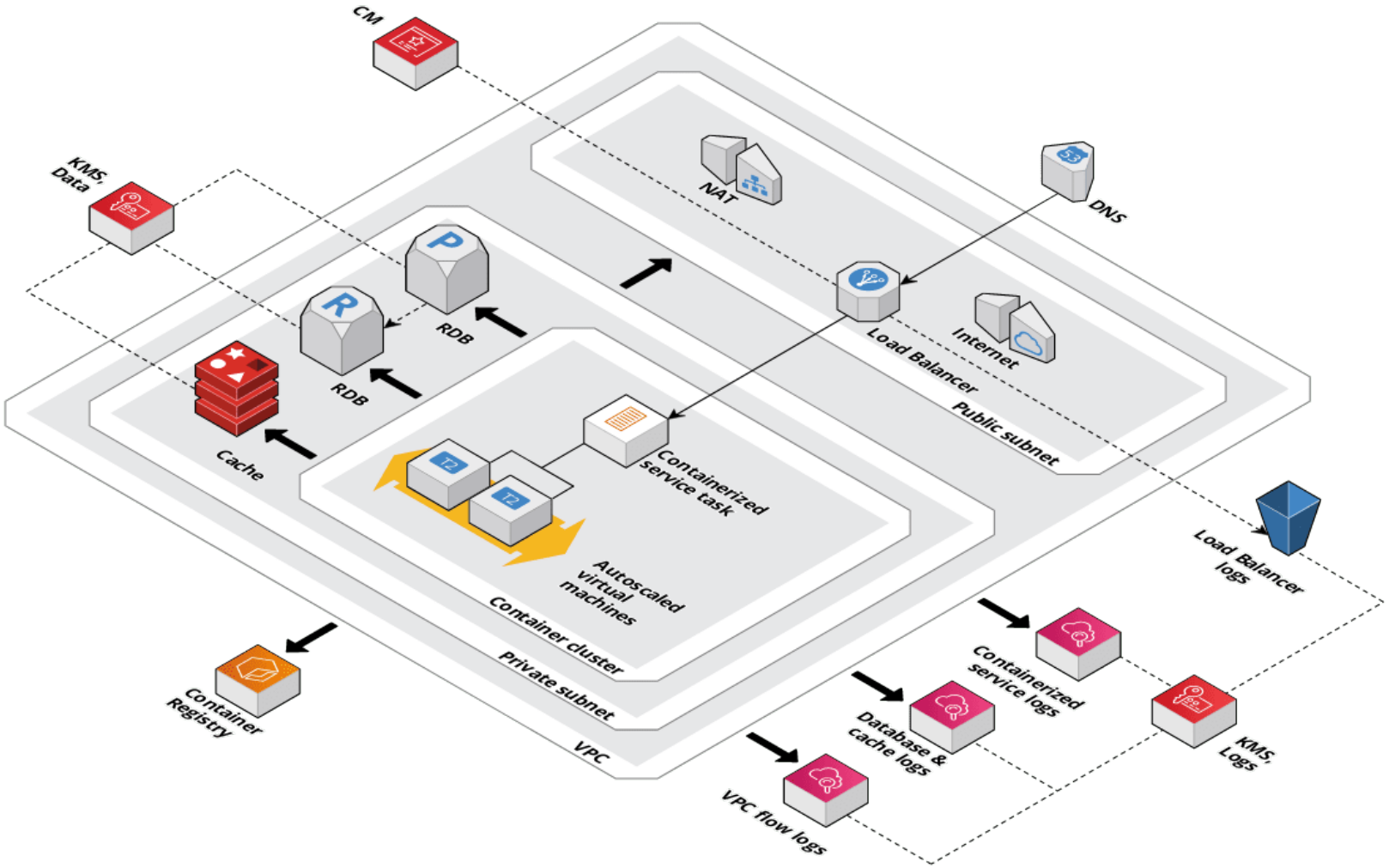
4.1.3 Serveriga veebiteenuste vaade

Tabel 7 keskendub serveriga veebiteenuste ressurssidele mis on võrreldes veebilehtede ressursitabeliga (Tabel 6) uudsed. Veebiteenuste puhul on turvatud HTTPS ühenduste võimaldamiseks koormusjaoturiga seotud vastavad SSL/TLS sertifikaadid. Domeeninimede süsteemis on omatav domeen seotud koormusjaoturi DNS aadressiga, mille AWS väljastab ressursi loomisel. Logid, andmebaaside ning vahemälude andmed on krüpteeritud erinevate KMS sümmeetriliste krüptovõtmetega, millel on rätseplahenduse korras seadistatud pääsupoliitikad (*access control*). Joonis 16 on visualiseeritud serveriga veebiteenuste abstraktne arhitektuur.

Tabel 7. Testlio serveriga veebiteenuste abstraktse taristu ressursid

ID	Ressurss	AWS teenus	Funktsioon
CCS1	*VPC*	*VPC*	*Virtual Private Cloud* ehk virtuaalvõrk pilves; loogiline erinevate võrkude kogum [52]
CCS2	*Public subnet*	*VPC. Subnets*	Avalik võrk, kus ressurssidel on avalik IP aadress ja vaikimisi ligipääs internetile [53]
CCS3	*Private subnet*	*VPC. Subnets*	Privaatne võrk, kus ressurssidel on privaatne IP aadress ja vaikimisi puudub ligipääs internetile. Piiratud ligipääs võimaldatakse läbi avaliku võrgu konfiguratsiooni [53]
CCS4	*Internet*	*Internet Gateway*	Võimaldab VPC's olevatele pilveressurssidele internetiühenduse [54]
CCS5	*NAT*	*NAT Gateway*	*Network Address Translation* ehk võrguaadresside teisendus; privaatses alamvõrgus olevatele ressurssidele internetti ligipääsu andmine. Eeldab Internet Gateway olemasolu [55]
CCS6	*Load Balancer*	*EC2. Load Balancers*	*Elastic Compute Cloud (EC2)* ehk pilvepõhised arvutusressursid. Koorusjaotur (*load balancer*) hajutab koormust ja võimaldab veebipäringute suunamist konfigureeritud reeglite alusel, konkreetsetele veebiteenuse instantsidele [56]
CCS7	*Load Balancer logs*	*Simple Storage Service (S3)*	Sihtkoht koormusjaoturi enda logide kirjutamiseks ja hoiustamiseks [50]
CCS8	*Cache*	*ElastiCache*	*Cache* ehk vahemälu; Tihti kasutatavate andmete hoiustamine nende puhverdamise eesmärgil erinevate teenuste jaoks [57]

ID	Ressurss	AWS teenus	Funktsioon
CCS9	*RDB*	*Relational Database Service (RDS)*	*Relational Database* (RDB) ehk relatsiooniline andmebaas; relatsiooniliste andmete hoiustamine pilves. Ühele andmebaasi instantsile vastab üks RDB. Eristatakse primaarseid (P) ja replikeerivaid (R) instantse, kus primaarse andmed kopeeritakse automaatselt replikeeritavasse instantsi. Replikeeriva instantsi puhul on kasutaja jaoks ainult lugemisoperatsioonid lubatud [58]
CCS10	*Container Cluster*	*Elastic Container Service (ECS)*	*Container Cluster* ehk konteinertehnoloogiaid kasutavate rakenduste klaster, st kogum. Eesmärgiks on konteinerite orkestreerimine viisil, mis rahuldaks erinevaid käideldavuse ja ressursitarbimise nõudeid konkreetsete konteineriseeritud teenuste lõikes [59]
CCS11	*Container Registry*	*Elastic Container Registry*	*Container Registry* ehk konteinerite register erinevate konteineriseeritud teenuste tõmmiste (*images*) hoiustamiseks. Registris olevad tõmmised on pilves uute konteinerite käivitamise aluseks ning iga konteineriseeritud teenus peab oma ehitusprotsessi käigus publitseerima loodud tõmmise vastavasse registrisse [60]
CCS12	*Conteinerized service task*	*ECS. Task Definitions*	Konteineriseeritud teenuse instants. Igal teenusel on ühe tõmmise kohta null kuni mitu konteinerit töös [61]
CCS13	*Autoscaled virtual machines*	*EC2. Auto Scaling*	Automaatselt skaleeritavate virtuaalmasinate grupp; võimaldamaks vajalike virtuaalmasinate hulka mis oleks suuteline käitlema vajamineva teenuse erinevaid kontainereid [62]
CCS14	*Logs*	*CloudWatch*	Reaalajas erinevate pilvekomponentide ja -teenuste tegevuste logimine. S3 ja CloudWatch pakuvad mõlemad logide kirjutamise võimalust aga iga kasutusjuhtum oleneb konkreetse pilveteenuse toest, hinnatundlikkusest ja muudest asjaoludest. Tavapäraselt on CloudWatch võimsam ja kallim lahendus [63]



Joonis 16. Testlio serveriga veebiteenuste taristu AS-IS abstraktne arhitektuur

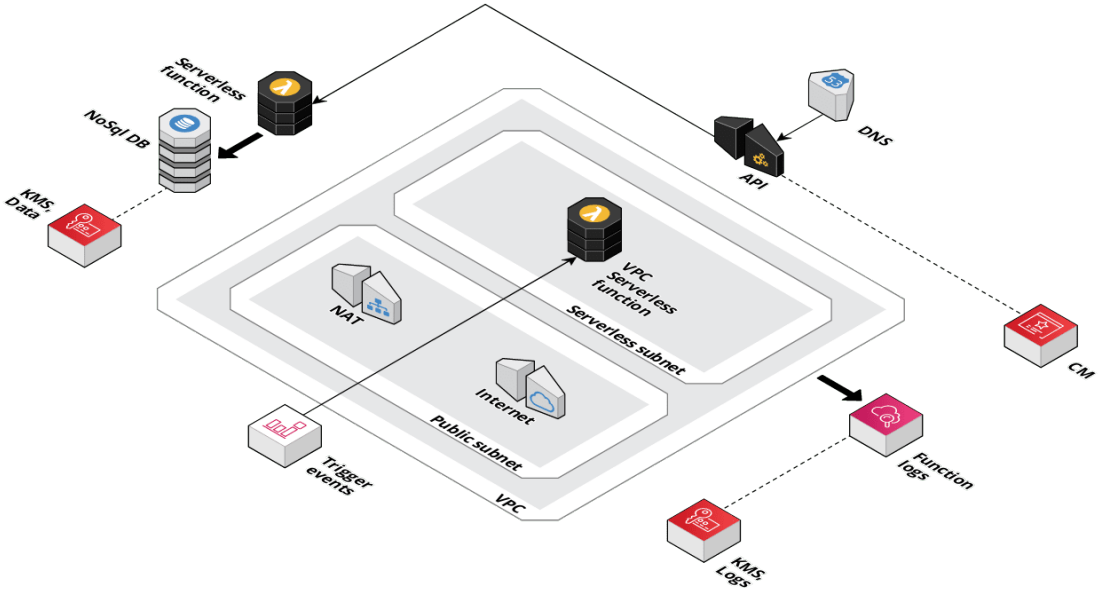
4.1.4 Serverita veebiteenuste vaade

Serverita lahendused võimaldavad erinevaid teenuseid jooksutada pilvepõhiste funktsioonidena, ilma servereid omamata. Funktsioonid käivitatakse vastava sündmuse peale ning nende eluiga on ühe funktsiooni tööaeg. Kirjeldatud pilvelahendusi nimetatakse kollektiivselt FaaS (*Function-as-a-Service*) platvormiks [44]. Tabel 8 kirjeldab serverivabu resursse, mis on võrreldes eelnevate pilveressursside uudsed (Tabel 6 ja Tabel 7).

Tabel 8. Testlio serverivabade veebiteenuste abstraktse taristu ressursid

ID	Ressurss	AWS teenus	Funktsioon
CCSL1	*Serverless subnet*	*VPC. Subnets*	Privaatne serverivabad funktsioonidele mõeldud võrk, kus vaikimisi puudub ligipääs internetile. Piiratud ligipääs võimaldatakse läbi avaliku võrgu konfiguratsiooni [53]
CCSL2	*Serverless function*	*AWS Lambda*	Serverivaba arvutuslik funktsioon etteantud lähtekoodi avalikuks käitlemiseks [64]
CCSL3	*VPC serverless function*	*AWS Lambda*	Serverivaba arvutuslik funktsioon etteantud lähtekoodi privaatseks käitlemiseks [64]
CCSL3	*API*	*API Gateway*	Application Programming Interface ehk rakendusliides; Võimaldab erinevad serverivabad funktsioonid kokku siduda ühte liidesesse [65]
CCSL4	*NoSql DB*	*DynamoDB*	Mitte-relatsiooniline andmebaas, mille skaleerimine ja turvaline majutus on AWS poolt eelnevalt lahendatud. Andmestruktuurid baseeruvad võti-väärtus (*key-value*) põhimõttele [66]
CCSL5	*Trigger Events*	*CloudWatch Events, AWS Lambda etc*	Erinevad sündmused, mis käivitavad serverivabu funktsiooni. Vastavate sündmuste kuulamine on iga funktsiooni kohta eraldi seadistatud
CCSL6	*Function logs*	*CloudWatch*	Reaalajas erinevate serverivabade funktsioonide tegevuste logimine. Kehtib nii avalike kui ka privaatsete (VPC) funktsioonide kohta [67]

Ettevõtte serverivabas taristus eristatakse kahte sorti lambda funktsioone – privaatsed ja avalikud. Privaatsetel ehk VPC's olevatel funktsioonidel on vajadusel ka ligipääs privaatse võrgu (*Private Subnet*) ressurssidele Avalikud funktsioonid töötavad VPC väliselt ning neid on võimalik hõlpsasti üle rakendusliidese (API) päringute käivitada. Avalik, serverivaba ja privaatne võrk on ettevõtte näol VPC osa. Joonis 17 vaates on visualiseeritud serverivabade veebiteenuste arhitektuur.



Joonis 17. Testlio serverivabade veebiteenuste taristu AS-IS abstraktne arhitektuur

Privaatseid funktsioone käivitatakse standardselt läbi järgmiste sündmuste:

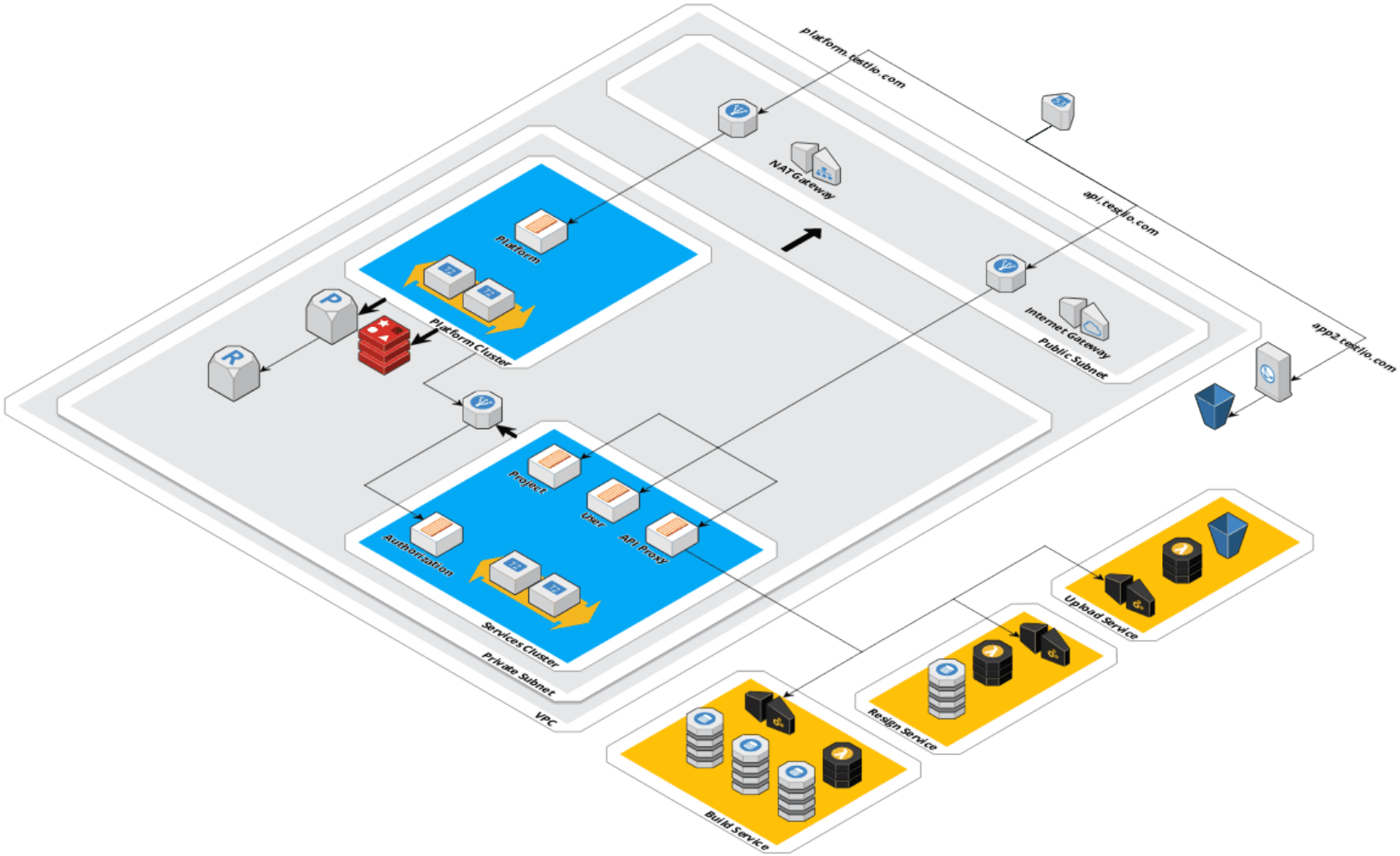
1. CloudWatch reeglid; näiteks ajareeglite (*cron*) järgi käituv päästik [68].
2. Autoriseeritud VPC välised teised serverivabad funktsioonid [69].
3. Andmebaasi triggerid; näiteks RDS konkreetse kirje muudatuse peale käivitatakse lambda funktsioon [70]
4. Mõni muu veebipäringuid tegev kliendi rakendus

4.1.5 Ettevõtte pilvearhitektuur

Ettevõtte pilvearhitektuuri alustaladeks on varasemas kolmes peatükis lahatud pilvekomponentide vaated – vastavalt veebilehed, serveriga ning serverita veebiteenused. Samuti eksisteerib piiratud hulk erilahendusi, kuid vastava detailsuse modelleerimine pole antud lõputöö skoobis. Fookuses on ainult relevantsed tuumikteenused. Joonis 18 kirjeldab ettevõtte üldise pilvearhitektuuri, mille visualiseerimiseks on kasutatud järgnevad värviskeemi:

1. **Sinine** – konteineriseeritud serveriga veebiteenuste klastrid (ptk 4.1.3);
2. **Kollane** – olemasolevad serverivabad veebiteenused (ptk 4.1.4) mis on vajalikud lõputöö eesmärgi realiseerimiseks;
3. **Hall** –võrgundusega seostuvad segmendid.

Ettevõttes eksisteerib kaks klastrit: üks uute veebiteenuste, teine pärandvarast platvormi tarbeks. Selline jaotus aitab tagada ressursside lahususe [71]. Platvormi klastri fookuses on monoliitne PHP tehnoloogiates realiseeritud platvorm, teenuste klaster majutab erinevaid mikroteenuseid. Mikroteenuste eripärasi kõnetab täpsemalt mikroteenuste arhitektuuri peatükk (ptk 4.4). Teenuste klastris (*services cluster*) täidab arhitektuuriliselt olulist rolly *API proxy* nimeline teenus, mis on oma olemuselt tagurpidine proxy (*reverse proxy*). Tagurpidise proksi eesmärk on teenindada avalikust veebist tulevaid veebipäringuid teiste teenuste nimel ja luues klientidele ühese kontaktpunkti erinevate teenustega suhtlemiseks [72]. Samuti on API proxy eesmärk pakkuda erinevate mikroteenuste avastamist. Ettevõtte API proxy teenus on kättesaadav aadressilt https://api.testlio.com/. Eksisteeriva mudeli järgi suhtlevad platvorm ja enamik teenuseid teineteisega üle avaliku veebi. Turvakaalutlustel on ainukeseks mõjuvaks erandiks on autoriseerimisteenus (*Authorization*), millega suhtlus käib läbi privaatses võrgus oleva koormusjaoturi (*load balancer*). Lisaks puudub teenuste klastris välja toodud teenustel oma andmebaasi ja puhvri instants. Andmete kirjutamise ja lugemise eest relatsioonilisse andmebaasi vastutab platvorm (*Platform*), mille vastavaid REST rakendusliideseid kutsuvad välja teenuse klastri erinevad teenused (*User, Project, Authorization*). Eraldi vahemälu komponendi kasutus on platvormi enda kiiruse parendamiseks ning ükski teenus seda aktiivselt ei kasuta. AS-IS joonise kompaktsuse huvides on välja jäetud ka reaalsuses eksisteerivad erinevad komplimenteerivad pilveressurssid (KMS, CM jpm), mille rakendust on kirjeldatud varasemates abstraktsetes vaadetes (Joonis 15, Joonis 16 ja Joonis 17).

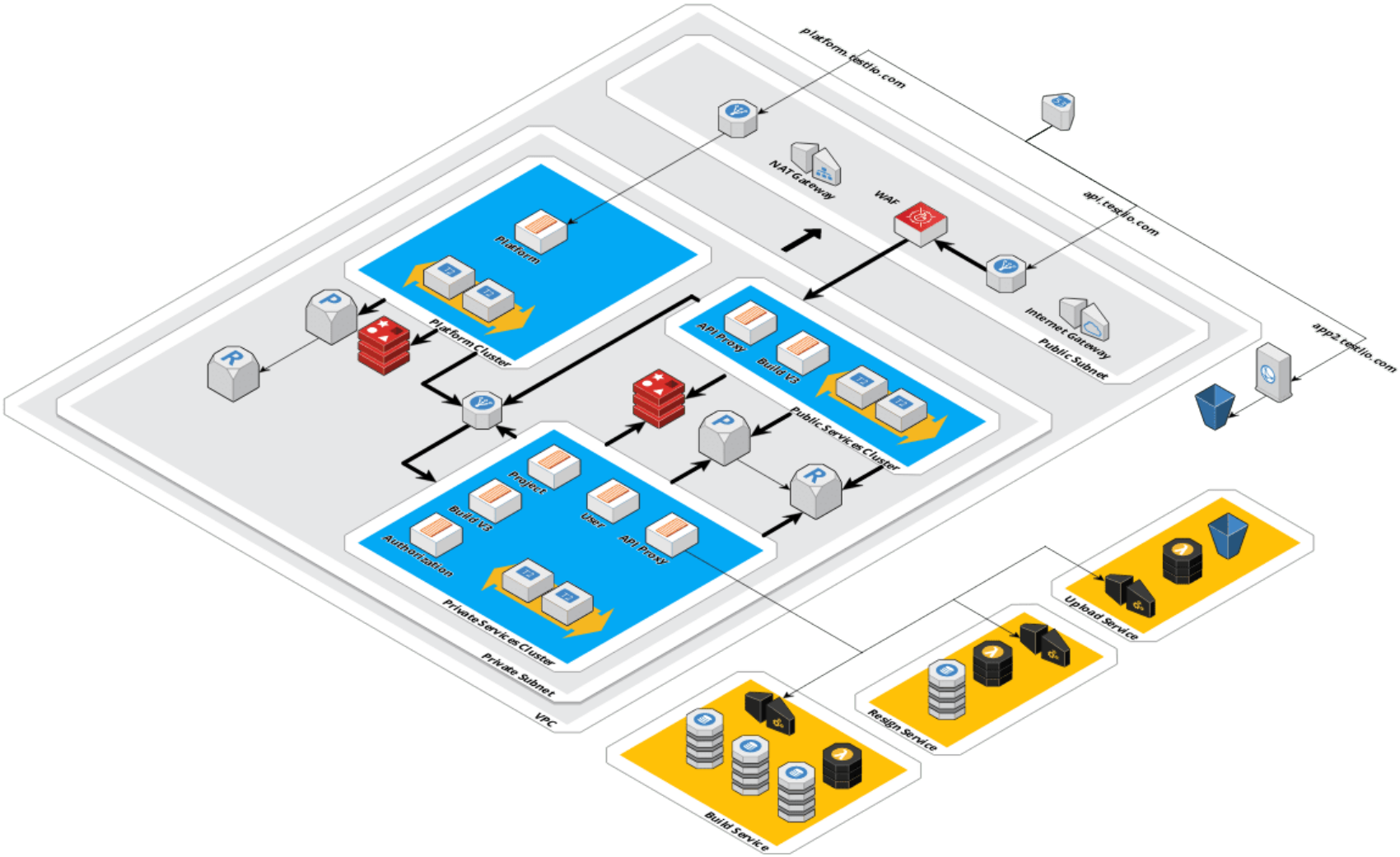


Joonis 18. Testlio pilvepõhise taristu minimeeritud AS-IS arhitektuur

4.2 Infrastruktuuri arhitektuur (TO-BE)

Joonis 19 kirjeldab loodavat uut infrastruktuuri, mille puhul on järgnevad täiendused:

1. Teenuste klaster (*services cluster*) on löödud kaheks: avalike teenuste klaster (*public services cluster*) ja privaatsete teenuste klaster (*private services cluster*). See võimaldab infrastruktuuri tasemel teenuste eraldust ning vajadusel avalike teenuste piiramist, ilma privaatseid mõjutamata (nõue **EIRFS11**).
2. Privaatses klastris töötab lõputöö käigus loodav uus teenus, *Build V3*. Sarnane *Build V3* teenuse instants on käitluses ka avalike teenuste klastris. Erinevus privaatse ja avalike teenuste instantside vahel määratakse teenuse ehitushetkel läbi *NODE_ENV* keskkonna muutuja (*environment variable*). Ehituspõhine eraldus võimaldab ainult relevantse funktsioonaalsuse kaasamist rakendusliideste tasemel (nõuded **EIRSC1** ning **IIRSC1**).
3. Avalike teenuste avastamine käib läbi avaliku klastri API Proxy teenuse. Privaatsete teenuste avastamine läbi privaatse klastri API Proxy teenuse (nõue **IIRRA1**).
4. Teenuste klastritel on oma vahemälu (*cache*) ja andmebaaside (*primary & replica*) instantsid. See võimaldab teenuste andmekoormus eraldada platvormi omast (nõuded **IIRFS5** ning **EIRRC2**).
5. Teenuste klastrite andmete lugemiskoormus on suunatud replikeerivasse (*R – Replica*) andmebaasi instantsi, kirjutamiskoormus primaarsesse (*P- Primary*) (nõue **IIRRC2**). Replikatsioon võimaldab sisuliselt reaalajas andmete varundamise (nõuded **EIRRR2** ning **IIRRR2**).
6. Avalikus võrgus oleva koormusjaoturi ülesanne on hallata ainult avaliku rakendusliidese (*public api*) kaudu avaliku teenuse klastrisse minevaid päringuid (nõue **EIRPE1**). Päringute suunamisel liigub kogu liiklus läbi veebitulemüüri (nõue **EIRFS12**)
7. Sisemises võrgus oleva koormusjaoturi ülesanne on hallata ainult sisemisi päringuid (nõue **IIRPE1**). Siinkohal loetakse erinevate sisevõrgus olevate klastrite vahelist kommunikatsiooni. Sellisel juhul pole ükski sisemine rakendusliides ega nendevaheline kommunikatsioon väljaspoolt kättesaadav (nõue **IRFS2**).



Joonis 19. Testlio pilvepõhise taristu minimeeritud TO-BE arhitektuur

4.3 Andmearhitektuur

Andmearhitektuur keskendub uue süsteemi jaoks vajaliku andmearhitektuuri loomisele. Laiendatakse eksisteerivat andmebaasi, täiendavalt luuakse uus andmebaas ning kirjeldatakse detailset kõikide uute tabelite sisemist ehitust.

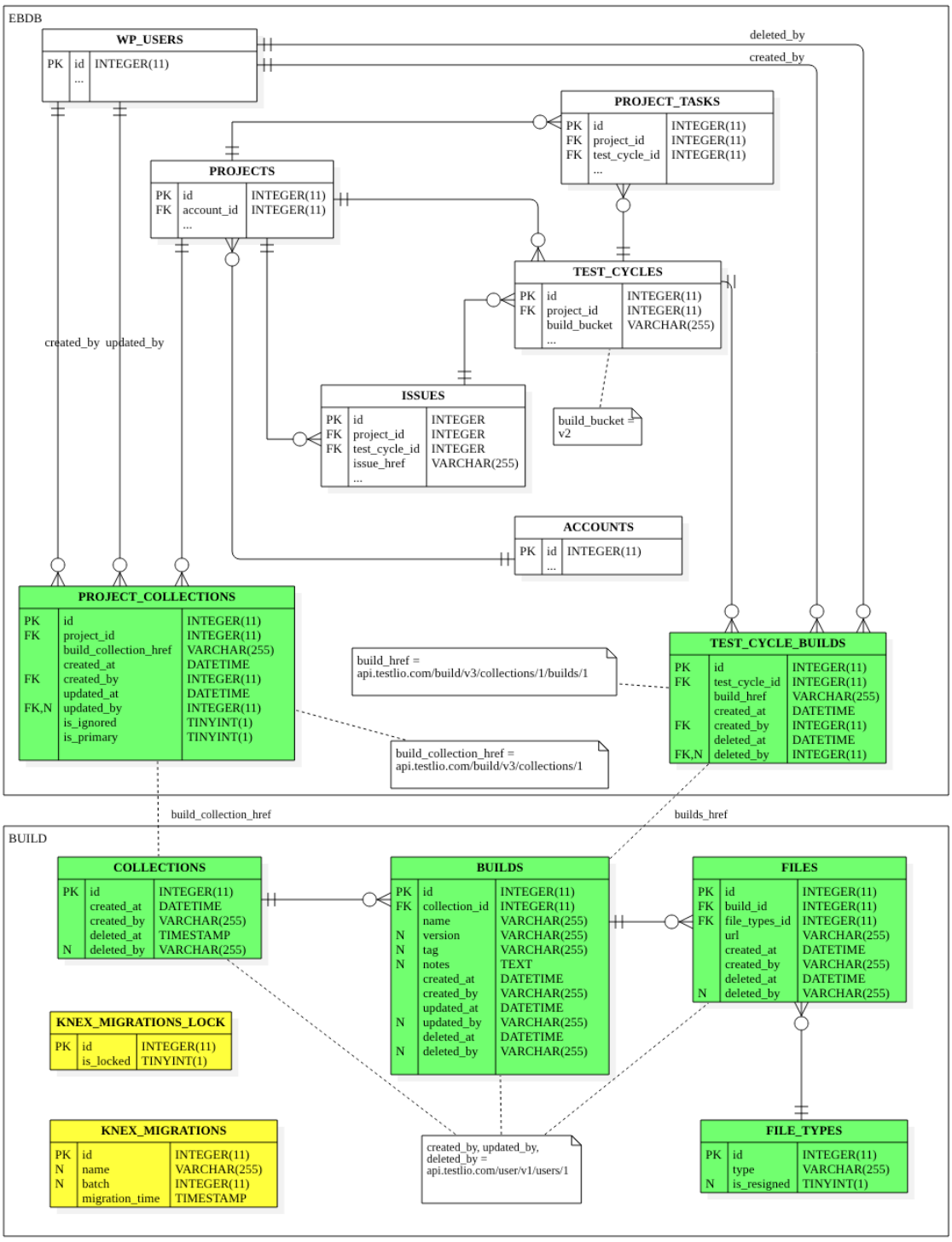
4.3.1 Andmemudel

Joonis 20 kujutab testimisplavormi uut andmemudelit, mis on kavandatud infoloogilisel tasemel ERD notatsioonis [13]. Tabelite fookuses on ainult relevantsed atribuudid. Tabelite värviskeem on järgnev:

1. **valge** – originaalsel kujul muutmata tabelid;
2. **roheline** – modelleeritud uued tabelid;
3. **kollane** – *Knex* raamistiku poolt automaatselt loodud uued tabelid [73].

Loodud mudelil on kirjeldatud kahte andmebaasi - uue keskkondade teenusele (*Build V3*) mõeldud andmebaas, *BUILD*, ning platvormi eksisteeriv andmebaas, *EBDB*. Andmebaasipõhine eraldus loob selge vastutuspõhise domeeni (nõue **IIRFS5**) – *BUILDS* andmebaas tegeleb ainult keskkondadega ning nende seosed projektide ja testimistsüklitega pole nimetatud andmebaasi skoobis. See võimaldab disainida, hallata ning skaleerida vastavaid andmebaase konkreetse teenuse individuaalse elutsükli järgi. Keskkondade seoseid projektide ning testimistsüklitega hoitakse täiendavates põhiandmebaasi *EBDB* tabelites, vastavalt *PROJECT_COLLECTIONS* ning *TEST_CYCLE_BUILDS*. *PROJECT_COLLECTIONS* tabel võimaldab koostete jagamist üle eri projektide ning kontode ning vastab äridomeeni TO-BE "projekti keskkond" kontseptsioonile, *COLLECTIONS* aga "keskkondade kogum" kontseptsioonile (ptk 3.4.1). *TEST_CYCLE_BUILDS* tabel võimaldab keskkondade taaskasutamise üle erinevate testimistsüklite (nõue **URFF5**). Keskkonna lisamisel projekti (nõuded **EIRFF1** ning **URFF11**) tekitatakse vastava *COLLECTIONS* kohta uued kirjed *BUILDS* ja *FILES* tabelitesse. Kui keskkonna tüübiks on iOS mobiilirakenduse fail, tuleb fail resigneerida (nõue **URRA2**). Resigneerimise tulemusena tekitakse vastava *BUILDS* kohta uus *FILES* kirje, kus *FILE_TYPES* tabeli *is_resigned* atribuudi tõeväärtus on tõene (arvväärtus 1). Andmebaaside üleselt hoitakse viiteid rakendusliidese disaini loogika alusel, kasutades selleks rakendusliidese linke (*build_collection_href* ning *build_href* näited).

Sarnaselt hoitakse ka kasutajate viiteid *BUILD* andmebaasis (*BUILD* andmebaasi *created_by*, *updated_by* ning *deleted_by* näited) (nõuded **EIRSA1** ning **URSA1**).



Joonis 20. Testimisplatvormi uus andmemudel

4.3.2 Tabelite disain

Tuginedes loodud andmemudelile (Joonis 20) on järgnevates tabelites välja toodud tabelite atribuudid, semantika ning kontekst. Atribuudid ja nende väärtused on modelleeritud relatsioonilise andmebaasimootori MySql 5.6 [74] baasil, millest olulisemad on:

1. *tinyint* – numbriline väärtus vahemikus 0 kuni 127;
2. *integer* – numbriline väärtus vahemikus 0 kuni 4 294 967 295;
3. *text* – tekstiline väärtus pikkusega 0 kuni 65 535;
4. *varchar* – tekstiline väärtus pikkusega 0 kuni 65 535, soovitav maksimaalne pikkus määratakse atribuudi loomise hetkel;
5. *datetime* – kuupäevaline väärtus vahemikus 1000-01-01 00:00:00 kuni 9999-12-31 23:59:59;
6. *CURRENT_TIMESTAMP* – kuupäevaline hetkeline ajaväärtus etteantud formaadis;
7. *NULL* – ehk tühiväärtus.

Rahuldamiseks unikaalsete indeksite nõudeid on seotud datetime andmetüübi veergude puhul kasutatud vaikeväärtust 0000-00-00 00:00:00. Vastavalt MySql 5.6 spetsifikatsioonile pole NULL väärtused samaväärsed [75]. Konkreetsete väärtuste kasutamine indeksi veergudel võimaldab vältida kahe kontekstuaalselt võrdse kirje ehk duplikaatide teket. Tabel 9 kirjeldab vajaminevaid unikaalseid indekseid. Iga uue tabeli tehniline vaade on kirjeldatud andmebaasi atribuutidele keskenduvas lisas (Lisa 2).

Tabel 9. Uue andmemudeli unikaalsed indeksid

Andmebaas	Tabel	Atribuut	Selgitus
EBDB	PROJECT_COLLECTIONS	project_id, build_collection_href	Garantii, et ühel projektil poleks mitu samaväärset keskkondade kogumit. Üks keskkondade kogum võib projekti suhtes olla kas primaarses või sekundaarses rollis. Mitte kunagi mõlemas. Vastava omaduse määrab ära EBDB.PROJECT_COLLECTIONS.is_primary atribuudi tõeväärtus.
EBDB	TEST_CYCLE_BUILDS	test_cycle_id, build_href, deleted_at	Garantii, et ühe testimistsükli ajahetkel ei saa konkreetne keskkond olla aktiivses olekus mitu korda.
BUILD	FILE_TYPES	type, is_resigned	Garantii, et iga failitüüp on unikaalne. Failitüübid on eeldefineeritud loetelu, millest täpselt ühele peab iga keskkond vastama.

4.4 Mikroteenuste arhitektuur

Mikroteenuste arhitektuur kirjeldab uue loodava teenuse ehitust, tööpõhimõtteid ja kommunikatsiooni ettevõtte mikroteenuste ökoruumis. Loetletakse erinevate teenuste vastutusalad ning vastavad liidestuspunktid, mis on vajalikud uue süsteemi funktsionaalsuse ehitamisel.

4.4.1 Kasutatavad teenused

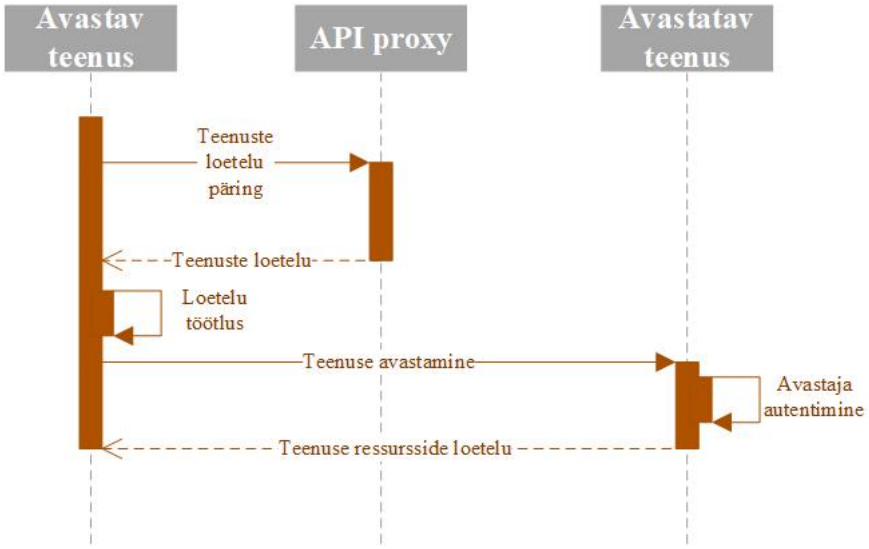
Loodava süsteemi kontekstis rakendatakse Testlio platvormi ning ettevõtte mikroteenuste ökosüsteemi alamosa. Lahenduse skoobis olevad relevantsed mikroteenused ja nende eesmärgid on järgnevad:

1. **Authorization service:** ehk ressursikasutuste autoriseerimiseks mõeldud teenus. Eesmärgiks on tagada, et ainult volitatud kasutajad ja/või teenused saaks konkreetseid ressursse kasutada ning üksnes lubatud viisidel [76].
2. **API proxy:** ehk tagurpidise proksi teenus. API proxy on pidepunkti erinevate teenuste avastamiseks [77]. Täiendavalt on lahatud teenuste avastamise mustrit teenuste avastamise peatükis (ptk 4.4.2).
3. **Build service:** ehk vana keskkondade teenus, mille uus teenus (*build-service-v3*) asendab [78]. Kõik keskkondade lisamised on mõeldud minema uude süsteemi, st nii avaliku rakendusliidese kui ka manuaalselt läbi kasutajaliidese lisatud keskkonnad.
4. **Project service:** ehk projektihalduseks mõeldud teenus. Projektiteenus on oma olemuselt Testlio platvormiga jäigalt seotud teenus, mille eesmärk on kapseldada platvormi projektidega seotud loogikat väiksemasse mikroteenusesse [79].
5. **Resign service:** ehk resigneerimise teenus. Teenus on vajalik ainult juhtudel mil tarkvaraliseks koosteks on iOS mobiiliplatvormile mõeldud fail. Teenus ise jaotub kahest alamteenusest: serverivaba pilveteenus ja füüsilised OSX operatsioonisüsteemi peal töötavad töölised (*workers*). Esmane on vastutav tööliste orkestratsiooni eest; teine aga resigneerimise ja failide üleslaadimise eest tagasi pilve [80].
6. **Upload service**: ehk failide pilve üleslaadimiseks mõeldud mikroteenus. Vastava päringu peale tagastab teenus ajutise volitusega AWS S3 ressursi aadressi, kuhu kliendil on piiratud aeg faili üleslaadimiseks. Vastavad viited kirjutatakse teenuse poolt omatavatesse AWS DynamoDB tabelitesse [81].

Kõik loetletud teenused eksisteerivad tootmiskeskkonnas. Uue teenusena lisandub **build service v3** (*Build V3*), mis asendab vana keskkondade teenuse (*build service*). Teenuste arhitektuurilise vaate annavad edasi varasemalt kirjeldatud Testlio pilvepõhise taristu minimeeritud AS-IS ja TO-BE joonised (vastavalt Joonis 18 ja Joonis 19).

4.4.2 Teenuste avastamine

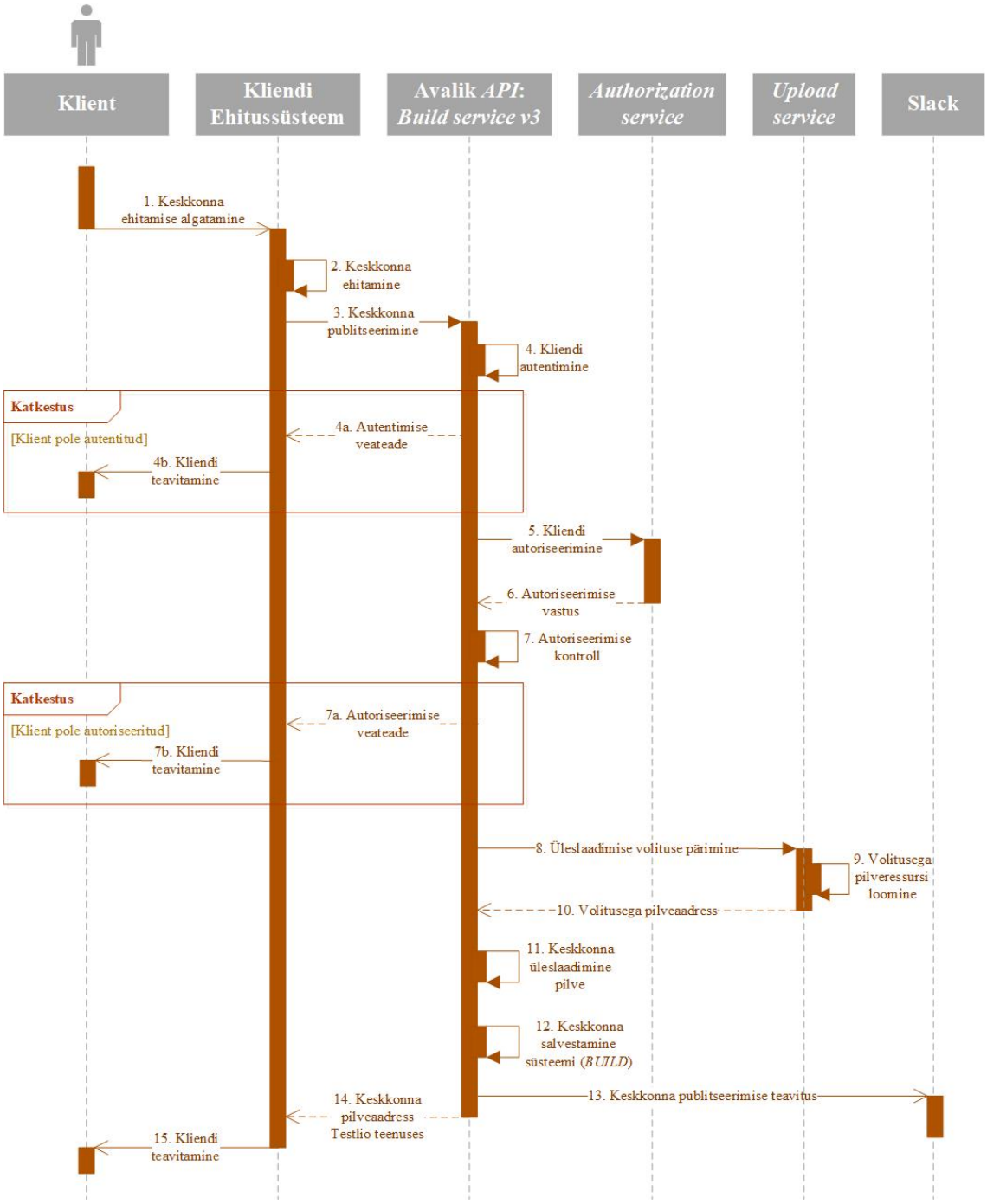
Teenuste avastamise (*service discovery*) tsentraalne mudel lahendab mikroteenuste maailmas ühe olulise probleemi: kuidas tagada suure hulka erinevate teenuste puhul omavahelise suhtluse töökindlus ja paindlikkus? Lahenduseks on kasutada teenuse registrit. Registrisese registreeritakse avastatavate teenuste poolt pakutav funktsionaalsus ja veebiaadress. Kliendi rollis olevad teenused saavad seejärel dünaamiliselt teisi teenuseid avastada, omamata jäiki viiteid avastatud teenuse asukohale [82]. Testlio mikroteenuste ökosüsteemi kontekstis täidab teenuste registri rolli API proxy nimeline teenus. Joonis 21 kirjeldab teenuste avastamise põhimõtteid abstraktsel tasemel. Teenuste avastamine on osa sisemise integratsiooni käideldavuse nõuetest (nõue **IIRRA1**).



Joonis 21. Testlio teenuste avastamise abstraktne mudel

4.4.3 Keskkondade automaatne lisamine

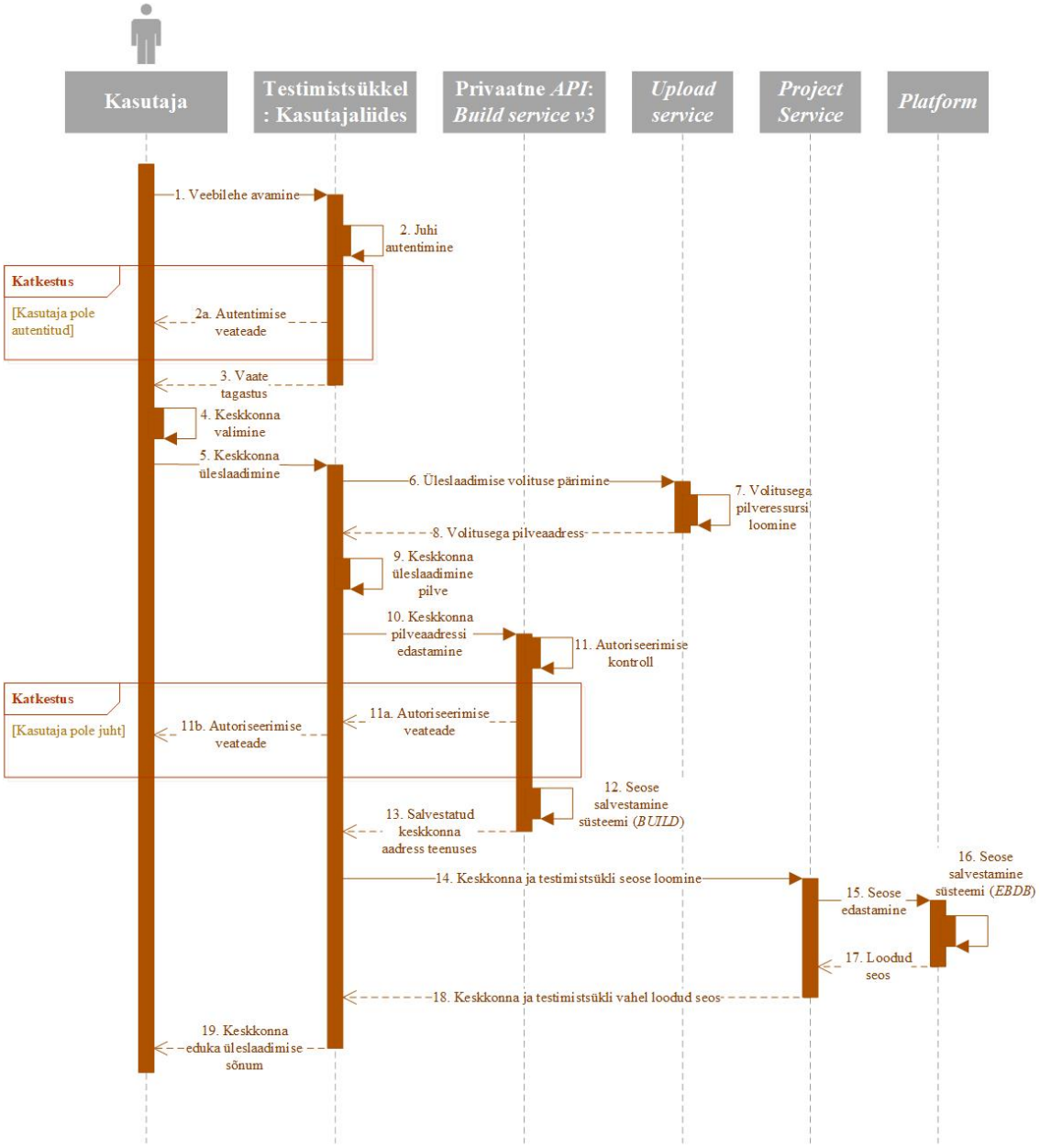
Keskkonna automaatse lisamise eelduseks on avaliku rakendusliidese kasutus Testlio klientide poolt. Vastav loogika on modelleeritud avaliku rakendusliidese jadadiagrammil (Joonis 22) ning baseerub kliendi kasutusjuhtudele **UC1** ja **UC2**. Diagrammi skoobist on välja jäetud teenuste avastamise mudel (ptk 4.4.2) kuna avastamine toimub taustal ning kõigi Testlio sisemiste teenuste vahel. Eduka keskkondade automaatse lisamise tulemusena tekivad vastavad kirjed *BUILD* andmebaasi (samm 12)



Joonis 22. Keskkondade automaatne lisamine läbi Testlio avaliku rakendusliidese

4.4.4 Keskkondade manuaalne lisamine

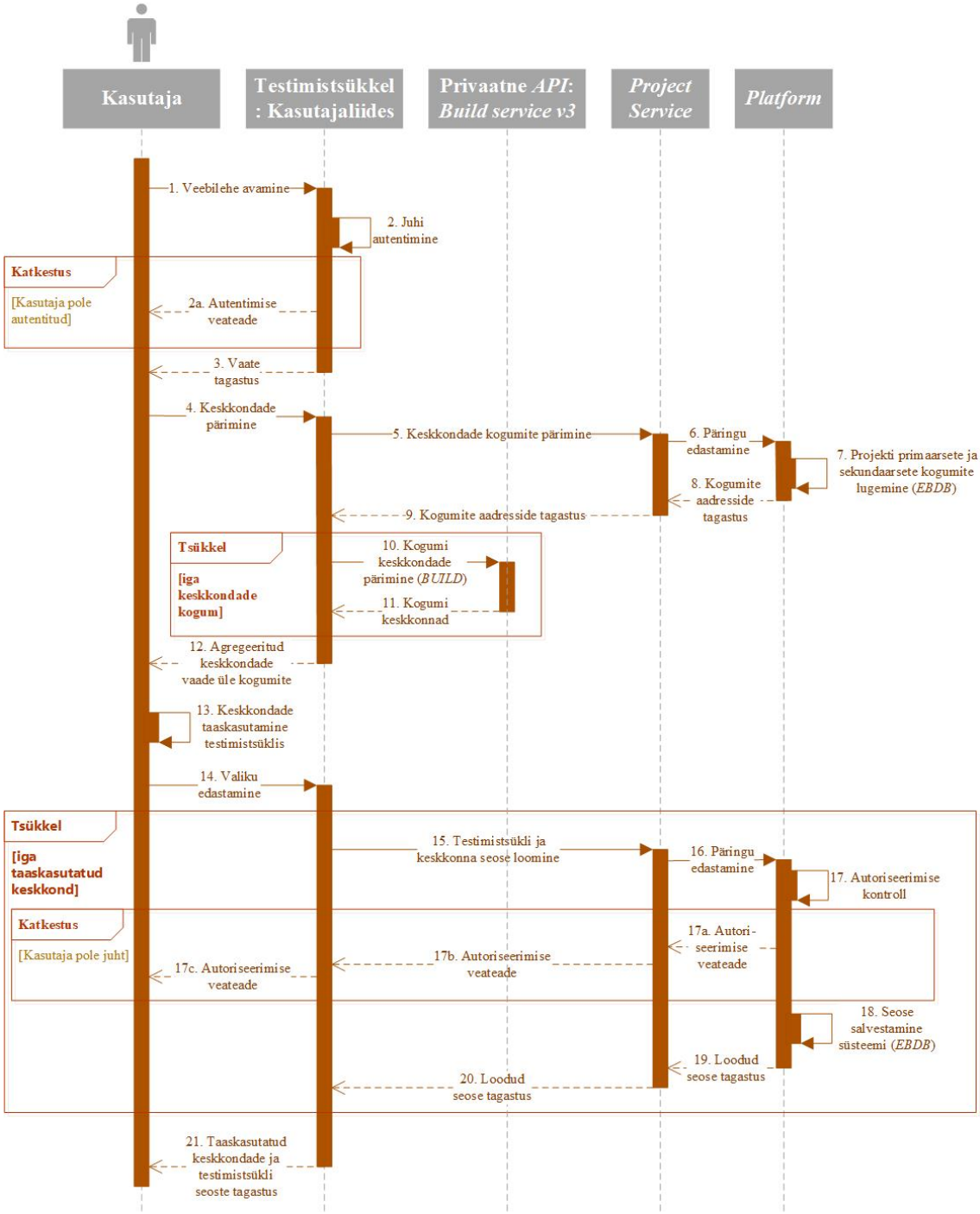
Keskkonna manuaalne lisamise võimalus on vajalik nii avaliku rakendusliidese integratsiooni puudumise korral kui ka üldise TO-BE äriprotsessi osana (ptk 3.4.2). Lisaja rollis on tavapäraselt kas projekti- või testimisjuht (üldisemalt *juht*) ning vastav lisamise loogika on modelleeritud kasutajaliidese jadadiagrammil (Joonis 23) ning baseerub ning baseerub juhi kasutusjuhule **UC3.** Kasutajaliidese kaudu edukalt testimistsüklisse lisatud keskkonna vastavad kirjed lisatakse *BUILD* (Samm 12) ning *EBDB* andmebaasidesse (Samm 16).



Joonis 23. Keskkondade manuaalne lisamine läbi kasutajaliidese

4.4.5 Keskkondade taaskasutamine

Keskkondade taaskasutamise funktsionaalsus baseerub kasutusjuhule **UC7** on üheks võtmekohaks keskkonna jaotuse probleemi lahendamisel (ptk 3.1.4). Selle tulemusena piisab keskkonna ühekordsest üleslaadimisest Testlio süsteemi, peale mida on keskkond kasutatav sisuliselt misiganes projekti testimistsüklites. Projektide ülese jagamise skoop oleneb projekti keskkondade jagamise seadistusest. Vastavat loogikat kujutab Joonis 24.



Joonis 24. Keskkondade taaskasutamine läbi kasutajaliidese

5 Valideerimine ja verifitseerimine

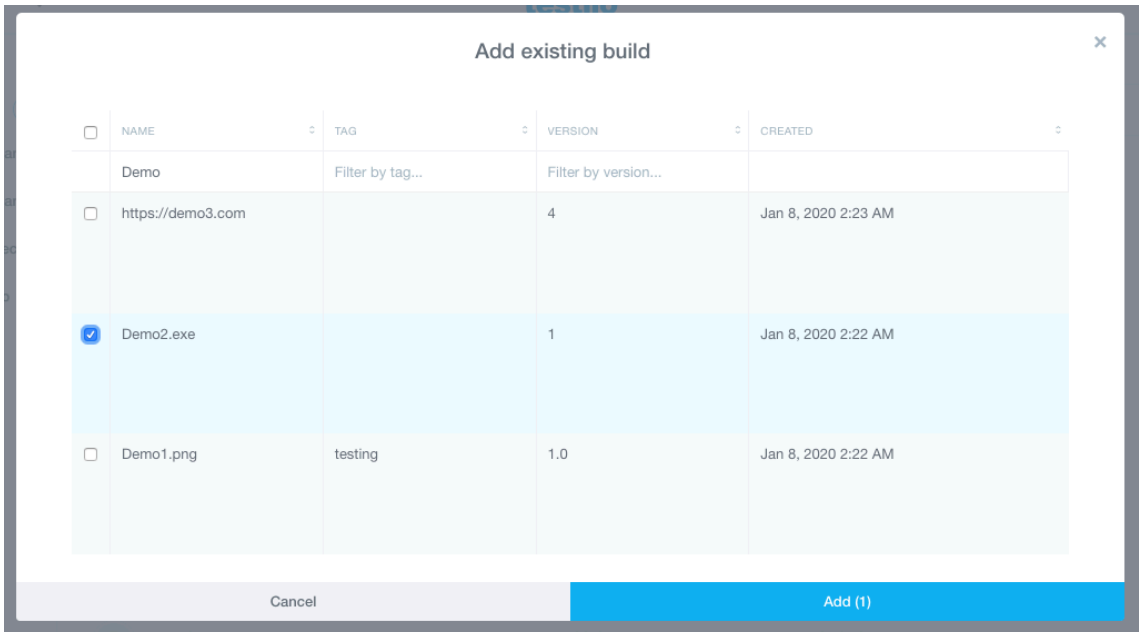
Valideerimine ja verifitseerimise eesmärk on loodava süsteemi headusele hinnangu andmine. Vaadeldakse teostatud töö tulemuse vastavust nõuetele ning süsteemi võimet lahendada reaalseid probleeme.

5.1 Süsteemi kasutuselevõtt

Loodud süsteemidisain on olnud arendusüksusele sisendiks uue süsteemi väljaarendamisel. Koos ärilise toetusega allokeeriti mitmeks kuuks osaliselt ühe alamüksuse arendusresurss. Süsteemi vastuvõtmisel lähtuti vastuvõtu kriteeriumitest (ptk 3.3.1) ning teostati põhjalik testimine ettevõtte äripoole poolt. Pilootprojekti (ptk 3.3.2) raames valiti välja sobiv klient, kes oli ka üks Testlio avaliku rakendusliidese huvigruppidest just keskkondade saatmise kontekstis. Kui varasemalt võttis keskkondade majandamine aega suurusjärgus 10 minutit keskkonna kohta (ptk 3.1.4), siis avaliku rakendusliidest kasutava kliendi puhul on vastav aeg äripoole sõnul suurusjärgus 30 sekundit kuni minut. Põhjendused on järgnevad:

1. Testlio poolne juht oli juba varasemalt teadlik keskkonna olemasolust, asukohast, versioonist ja muudest andmetest ning seda tänu läbi avaliku rakendusliidese integratsiooni (kasutusjuhud **UC1** ning **UC2**).
2. Läbi keskkondade taaskasutamise uue funktsionaalsuse polnud vaja enam keskkondi uuesti üles laadida. Keskkondade jagamine eri projektide vahel on juhi poolt konfigureeritav, millest tulenevalt on süsteemis juba olemasolevat keskkonda võimalik taaskasutada mitmes eri projektis. Taaskasutamine tähendab veebipäringute abil andmebaasis seoste loomist eksisteeriva keskkonna ja testimistsükli vahel, mistõttu on loodav funktsionaalsus kordades kiirem kui failide uuesti üleslaadimine (kasutusjuht **UC7**).

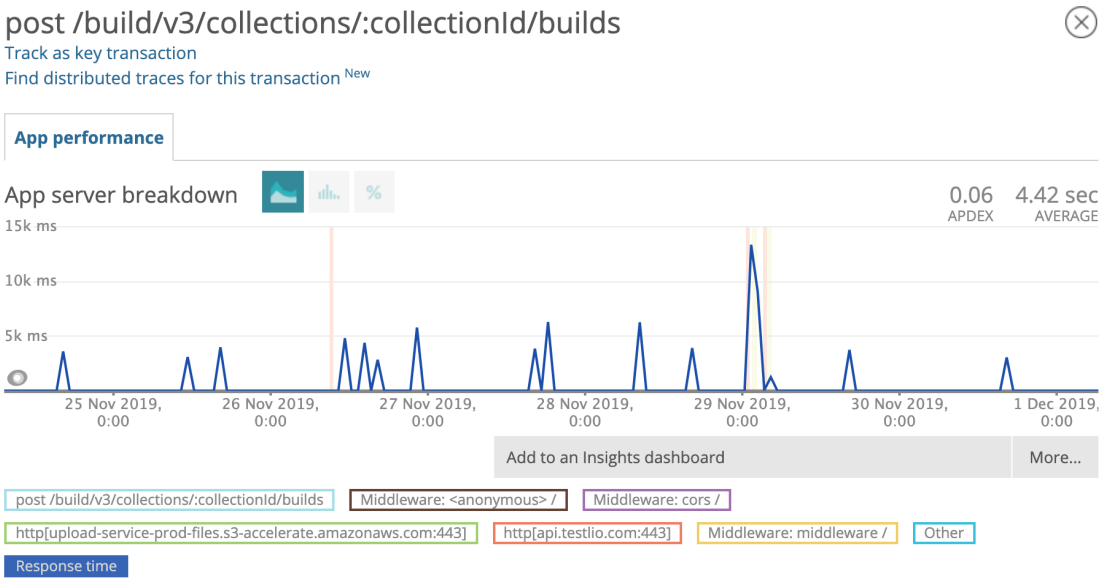
Joonis 25 on kuvatõmmis keskkondade taaskasutamise modaalaknast.



Joonis 25. Kuvatõmmis keskkondade taaskasutamisest

5.2 Süsteemi võimekus

Ettevõttes kasutatakse tootmiskeskkonnas rakenduste toimivuse seireks (*APM – Application Performance Monitoring*) NewRelic nimelist töövahendit. Võrreldes klassikaliste monitooringulahendusega on APM'il suur eelis: APM pakub multidimensionaalset vaadet, võrguliikluse analüüsist kuni serverite jälgimiseni [83]. Ettevõttes kasutatav NewRelic edastab reaalajas rakenduse jõudluse andmeid ning aitab seeläbi visualiseerida lõppkasutajate kogemust [84]. Lõputöö käigus loodud lahenduse jõudluspõhist meetrikat on tänu eelnevale teenuse konfigureerimisele samuti võimalik näha reaalajas (nõueded **EIRUR3** ning **IIRUR3**). Avalike ja privaatsete päringute teed on eraldatud rakendusliidese tasemel (nõue **EIRFS10**), seega konkreetse *REST* rakendusliidese jõudlust on võimalik lihtsamini analüüsida. Joonis 26 vaatleb nädalaajalist perioodi, kus tuuakse välja avaliku rakendusliidese kasutus ettevõtte klientide poolt: keskmise päringu kiiruseks on 4,42 sekundit. Edukaks päringuks loetakse siinkohal keskkona üleslaadimist Testlio süsteemi. Ajaperioodil üleslaetud keskkondade puhul on failide suurusjärgud tavapäraselt olnud kas 100MB või 250MB.



Joonis 26. Avaliku rakendusliidese jõudlus

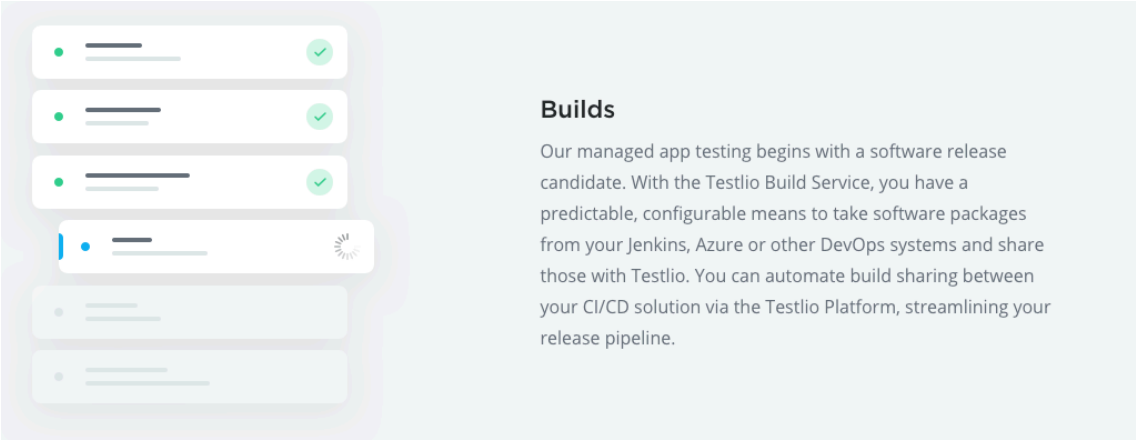
Tuginedes NewRelic'u poolt pakutavatele andmetele oli teenuse käideldavus 2019 detsemberis 99,9% juures, rahuldades seeläbi süsteemi oodatava käideldavuse nõuet (nõue **EIRRA1**). Tabel 10 kirjeldab käideldavuse meetrikat alates süsteemi tootmiskeskkonda paneku hetkest, tuginedes süsteemi rahulolu indeksile [85].

Tabel 10. Uue süsteemi käideldavus

Kriteerium	**Sep 2019**	**Oct 2019**	**Nov 2019**	**Dec 2019**	**Jan 2020**
Päringute arv (tuhandetes)	1040	955	1080	1190	1180
Keskmine päringu aeg (millisekundites)	3.68	4.13	5.19	3.56	3.17
Rahulolu indeks	1.0	1.0	1.0	1.0	1.0
Heade päringute osakaal (%)	99.9	99.6	99.5	99.9	100.0
Neutraalsete päringute osakaal (%)	0.1	0.3	0.3	0.1	0.0
Halbade päringute osakaal (%)	0.0	0.0	0.1	0.1	0.0

5.3 Avalik väljapanek

Keskkonnad on Testlio ökosüsteemi üheks põhimooduliks ning ilma keskkondadeta poleks klientide jaoks võimalik teste teostada. Lõputöö tulemusena loodud süsteem on heaks kiidetud ettevõtte äripoole poolt ning on osa Testlio avalikust müügipakkumisest [86]. Joonis 27 on firma kodulehelt tehtud kuvatõmmis (vt "*You can automate build sharing…*").



Joonis 27. Kuvatõmmis avaliku rakendusliidese müügist kodulehel

6 Kokkuvõte

Testlio OÜ poolt on olnud peamiseks ülesandepüstituseks töötada välja viis kuidas ettevõtte kliendid saaks Testlio süsteemidesse automaatselt saata keskkondi (*builds*) ning seda viisil, mis oleks pilvetehnoloogiate kontekstis turvaline ja skaleeruv. Lõputöös kajastatud töö tulemusena on mõtestatud äridomeeni (ptk 3.1.2) ning -protsesside (ptk 3.1.3) hetkeseis. Tõstatatud on keskkondadega seonduvad nii ärilised (ptk 3.1.4) kui ka tehnoloogilised (ptk 3.1.5) probleemid. Sõnastatud on vajaliku lahenduse rakendusdomeen (ptk 3.2.1) ning selle välise integratsiooni, sisemise integratsiooni ja kasutajate nõuded. Valitud lahendusele on modelleeritud kasutusjuhud (ptk 3.2.2) ning püstitatud on lahenduse põhiväärtused ja nende loomise instrumentatsioon:

1. Kliendi ehitussüsteemist automaatselt keskkondade saatmine: kiirendades kogu testimisprotsessi ning võimaldades niimoodi eemaldada keskkondade süsteemidevaheliseks majandamiseks seni tehtud manuaalne projekti- või testimisjuhi töö.
2. Automaatselt lisatud keskkondade teavitussüsteem: võimaldades huvigruppidel reaalajas jälgida Testlio süsteemi lisatud keskkondi ning saatvat informatsiooni.
3. Kasutajaliideses manuaalselt keskkondade lisamise testimistsüklisse: nii avaliku rakendusliidese integratsiooni puudumise korral kui ka üldise äriprotsessi osana.
4. Kasutajaliideses keskkonna andmete muutmine (versioon, märgend jpm): varasemalt nimetatud funktsionaalsuse puudumine põhjustas identsete keskkondade taasüleslaadimisi kuna keskkonda saatvate andmete kirjeldamine oli võimalik ainult üleslaadimise hetkel.
5. Keskkondade kustutamine kas testimistsüklist või süsteemist tervikuna: pakkudes vajalikku osa keskkondade haldamise elutsüklist.
6. Kasutajaliideses keskkondade taaskasutamine testimistsüklites: andes ülevaate projektis olemasolevatest keskkondadest ning võimaldades süsteemis olemasolevaid hõlpsasti taaskasutada, ilma midagi uuesti üles laadimata.
7. Kasutajaliideses projekti keskkondade sorteerimine ja filtreerimine: pakkudes mugavust kindla tunnuse järgi kiiresti üles leida kõige relevantsem keskkond.

8. Projekti keskkondade jagamise seadistus: luues projekti- ja testijuhtidele vabaduse taaskasutada keskkondi üle erinevate projektide.
9. Testimistulemuste raporteerimine: taaskasutades eksisteerivat automaatse raporteerimise funktsionaalsust, võimaldades kliendi raportile genereerida kõik testimistsüklisse pandud keskkonnad.
10. Testimistsükli juhiste tarbimine: luues ülevaate testimistsüklisse lisatud keskkondadest ning nende andmetest.
11. Keskkonna tarbimine: lubades testimistsüklisse pandud keskkondi alla laadida autentitud kasutajate poolt ja vajadusel paigaldada oma seadmetesse.

Lahenduse instrumentatsioonile on püstitatud funktsionaalsed ja mitte-funktsionaalsed nõuded kogu rakendusdomeeni lõikes (ptk 3.2.3, 3.2.4, 3.2.5). Ettevõtte täiendatud äridomeen (ptk 3.4.1) ning -protsessid (ptk 3.4.2) kajastavad keskkondade oluliselt suuremat väärindamist, luues lisaks ka keskkondade taaskasutamise vabaduse. Tuginedes nõuetele on loodud terviklik arhitektuuriline sisend (ptk 4), mis toestab keskkondade jaoks uue pilveteenuse ehitamist, selle liidestamist Testlio eksisteeriva tehnoloogilise ökosüsteemiga ning avaliku rakendusliidese rajamist. Eriline rõhuasetus on infrastruktuuri uue pilvearhitektuuri väljatöötamisel *Amazon Web Services* platvormil. Läbi *Docker* konteineriseerimistehnoloogia on ehitatud eraldi pilveklastrid ettevõtte siseste *versus* väliste teenuste jaoks. Vastava klastri konfiguratsioonid, koos koormusjaoturite (*load balancer*), veebitulemüüride (*web application firewall*) ning ehitushetkel defineeritud teenuseprofiilidega võimaldavad luua turvalise ja skaleeritava pilvearhitektuuri (ptk 4.2). Uue andmemudeli disainilt on lähtutud MySql andmebaasimootorist ning uue teenuse jaoks on loodud uus *BUILD* andmebaas koos täiendavate tabelitega Testlio platvormi *EBDB* andmebaasis (ptk 4.3.1 ja 4.3.2). Kirjeldatud on uue teenuse toimimist mikroteenuste kontekstis ning suhtlust teiste teenustega, andes funktsionaalse ülevaate erinevate teenuste vahelisest kommunikatsioonist (ptk 4.4). Väljatöötatud süsteemi vastuvõtmise (ptk 3.3.1) strateegia alusel on loodud süsteem tootmiskeskkonnas ning esimene pilootprojekt on edukalt teostatud (ptk 3.3.2). Loodud lahendus on omaks võetud ettevõtte äriüksuse poolt ning on osa firma ametlikust väärtuspakkumisest. Vastavat pakkumist võib näha ettevõtte kodulehel: https://testlio.com/platform/#platform-builds. Kirjeldatud lahendus ning selle funktsionaalsus on lõputöö esitamise hetkeks arvestaval määral reaalselt implementeeritud ning lõputöö autor on olnud ettevõtte arendusüksuses selle tehnilises keskmes.

Kasutatud kirjandus

[1] C. Pettey, "Cloud Shift Impacts All IT Markets," 2019. [Online]. Available: https://www.gartner.com/smarterwithgartner/cloud-shift-impacts-all-it-markets/. [Kasutatud Jaanuar 2020].

[2] B. Kleyman, "The State of Cloud: Adoption Trends, Managing Infrastructure, and Empowering Choice," 19 Juuli 2019. [Online]. Available: https://www.datacenterknowledge.com/cloud/state-cloud-adoption-trends-managing-infrastructure-and-empowering-choice. [Kasutatud Juuli 2019].

[3] "Testlio platform architecture," [Online]. Available: https://testlio.com/platform/#platform-architecture. [Kasutatud Jaanuar 2020].

[4] "Testlio OÜ," Creditinfo Eesti AS, [Online]. Available: https://www.e-krediidiinfo.ee/12449969-TESTLIO%20OÜ?lang=en. [Kasutatud September 2019].

[5] "About Us," Testlio OÜ, [Online]. Available: https://testlio.com/about-us/. [Kasutatud November 2019].

[6] "Testlio Capabilities," Testlio OÜ, [Online]. Available: https://testlio.com/solutions/capabilities. [Kasutatud August 2019].

[7] "Org chart," Testlio OÜ, [Online]. Available: https://testlions.atlassian.net/wiki/spaces/GEN/overview. [Kasutatud Detsember 2019].

[8] "How Testlio works?," Testlio OÜ, [Online]. Available: https://testlio.com/how-testlio-works/. [Kasutatud August 2019].

[9] "Architecture," Testlio OÜ, [Online]. Available: https://testlions.atlassian.net/wiki/spaces/EN/pages/29895056/Architecture. [Kasutatud Detsember 2019].

[10] J. Brezet, H. Christiaans and J. Diehl, "To craft, by design, for sustainability : Towards holistic sustainability design for developing-country enterprises," 12 12 2016. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid:0c2c14c8-9550-449d-b1ff-7e0588ccd6c2?collection=research. [Kasutatud Detsember 2019].

[11] J. Iivari and J. R. Venable, "Action research and design science research - Seemingly similar but decisively dissimilar," 2009. [Online]. Available: https://aisel.aisnet.org/ecis2009/73/. [Kasutatud Detsember 2019].

[12] R. Gleasure, J. Feller and B. F. O'Flaherty, "Procedurally Transparent Design Science Research: A Design Process Model," 14 12 2012. [Online]. Available: https://aisel.aisnet.org/icis2012/proceedings/ResearchMethods/10/. [Kasutatud Detsember 2019].

[13] "Relatsiooniliste andmemudelite koostamise juhend," Riigi Infosüsteemide Amet, 2015. [Online]. Available: https://www.ria.ee/sites/default/files/content-

editors/publikatsioonid/relatsiooniliste_andmemudelite_koostamise_juhend_ver._1.0.pdf. [Kasutatud August 2019].

[14] C. Haisjackl, P. Soffer, S. Y. Lim and B. Weber, "How do humans inspect BPMN models: an exploratory study," Springer Open Choice, 7 10 2016. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5910471/. [Kasutatud Mai 2019].

[15] "Kalendaarse tööajafondi arvestus," Kutseliit, 2018. [Online]. Available: https://www.kutseliit.eu/wp-content/uploads/kalendaarse-tooajafondi-arvestus-2018.pdf. [Kasutatud August 2019].

[16] "Töölepingu seadus," Riigikogu, 17 12 2008. [Online]. Available: https://www.riigiteataja.ee/akt/13120899. [Kasutatud August 2019].

[17] "Amazon API Gateway Limits and Important Notes," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/apigateway/latest/developerguide/limits.html. [Kasutatud November 2019].

[18] "AWS Lambda Limits," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/limits.html. [Kasutatud November 2019].

[19] "Secondary Indexes," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Limits.html#limits-secondary-indexes. [Kasutatud November 2019].

[20] J. M. Hellerstein and M. Stonebraker, Readings in Database Systems (The MIT Press), The MIT Press, 2005.

[21] "Reading Data from a Table," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SQLtoNoSQL.ReadData.html. [Kasutatud November 2019].

[22] "Best Practices for Modeling Relational Data in DynamoDB," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-relational-modeling.html. [Kasutatud November 2019].

[23] C. Larman, Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (2nd Edition), Prentice Hall PTR, 2001, pp. 41-44.

[24] M. Jackson, in *Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices*, Addison-Wesley Professional, 1995, p. 10.

[25] M. Jackson, in *Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices (ACM Press)*, Addison-Wesley Professional, 1995, p. 63.

[26] A. Modeling, "UML 2 Use Case Diagrams: An Agile Introduction," [Online]. Available: http://www.agilemodeling.com/artifacts/useCaseDiagram.htm. [Kasutatud Juuni 2019].

[27] C. Larman, Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (2nd Edition), Prentice Hall PTR, 2001, pp. 45-58.

[28] "What is Slack?," Slack Technologies, Inc., [Online]. Available: https://slack.com/intl/en-ee/help/articles/115004071768-What-is-Slack-?eu_nc=1. [Kasutatud August 2019].

[29] M. Jones, J. Bradley and N. Sakimura, "JSON Web Token (JWT)," Mai 2015. [Online]. Available: https://tools.ietf.org/html/rfc7519. [Kasutatud Mai 2019].

[30] "Introduction to JSON Web Tokens," Auth0, Inc., [Online]. Available: https://jwt.io/introduction/. [Kasutatud Mai 2019].

[31] D. J. Bernstein, "Understanding brute force," The University of Illinois at Chicago, [Online]. Available: https://cr.yp.to/snuffle/bruteforce-20050425.pdf. [Kasutatud Juuli 2019].

[32] D. Anders, "Introduction to Web Application Firewalls," [Online]. Available: https://www.owasp.org/images/d/d1/Intro-WebApplicationFirewalls.pdf. [Kasutatud 2019].

[33] E. Vargas, "High Availability Fundamentals," Enrique Vargas Sun BluePrints™ OnLine, November 2000. [Online]. Available: https://pdfs.semanticscholar.org/3830/8c355bd0d36f9cae33113e34c83fdbd3e568.pdf. [Kasutatud Juuli 2019].

[34] "Using Environment Variables," Circle Internet Services, Inc., [Online]. Available: https://circleci.com/docs/2.0/env-vars/. [Kasutatud 2019].

[35] "Environment Variables," Travis CI, GMBH, [Online]. Available: https://docs.travis-ci.com/user/environment-variables/ . [Kasutatud 2019].

[36] M. Rouse, "principle of least privilege (POLP)," TechTarget, [Online]. Available: https://searchsecurity.techtarget.com/definition/principle-of-least-privilege-POLP. [Kasutatud August 2019].

[37] "Ant Design," Ant Design, [Online]. Available: https://ant.design/. [Kasutatud August 2019].

[38] "Code Signing," Apple Inc, [Online]. Available: https://developer.apple.com/support/code-signing/. [Kasutatud August 2019].

[39] D. Sarkar, N. Rakesh and K. K. Mishra, "Content delivery networks: Insights and recent advancement," Detsember 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7913113. [Kasutatud September 2019].

[40] S. Kempter, "Release and Deployment Management," Veebruar 2018. [Online]. Available: https://wiki.en.it-processmaps.com/index.php/Release_and_Deployment_Management. [Kasutatud Juuli 2019].

[41] "Exploring the Cloud: A Global Study of Government's Adoption of Cloud," KPMG International Cooperative, Veebruar 2012. [Online]. Available: https://images.forbes.com/forbesinsights/StudyPDFs/exploring-cloud.pdf. [Kasutatud Juuli 2019].

[42] "Gartner Report: Magic Quadrant for Cloud Infrastructure as a Service, Worldwide," Amazon Web Services, Inc., 2019. [Online]. Available: https://pages.awscloud.com/Gartner-Magic-Quadrant-for-Infrastructure-as-a-Service-Worldwide.html. [Kasutatud August 2019].

[43] "Gartner Magic Quadrant," Gartner, [Online]. Available: https://www.gartner.com/en/research/methodologies/magic-quadrants-research. [Kasutatud August 2019].

[44] M. Roberts, "Serverless Architectures," 22 Mai 2018. [Online]. Available: https://martinfowler.com/articles/serverless.html. [Kasutatud September 2019].

[45] "Cloudcraft," Cloudcraft Inc, [Online]. Available: https://cloudcraft.co/. [Kasutatud August 2019].

[46] "AWS Architecture Icons," Amazon Web Services, Inc., [Online]. Available: https://aws.amazon.com/architecture/icons/. [Kasutatud August 2019].

[47] "What Is Amazon Route 53?," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/Welcome.html. [Kasutatud August 2019].

[48] "What Is Amazon CloudFront?," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html. [Kasutatud August 2019].

[49] "What Is AWS Certificate Manager?," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/acm/latest/userguide/acm-overview.html. [Kasutatud August 2019].

[50] "What is Amazon S3?," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome.html. [Kasutatud August 2019].

[51] "What is AWS Key Management Service?," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/kms/latest/developerguide/overview.html. [Kasutatud August 2019].

[52] "What Is Amazon VPC?," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html. [Kasutatud August 2019].

[53] "VPCs and Subnets," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Subnets.html. [Kasutatud September 2019].

[54] "Internet Gateways," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Internet_Gateway.html. [Kasutatud August 2019].

[55] "NAT Gateways," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html. [Kasutatud August 2019].

[56] "What Is Elastic Load Balancing?," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/what-is-load-balancing.html. [Kasutatud August 2019].

[57] "What Is Amazon ElastiCache for Redis?," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/WhatIs.html. [Kasutatud August 2019].

[58] "What Is Amazon Relational Database Service (Amazon RDS)?," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html. [Kasutatud September 2019].

[59] "What is Amazon Elastic Container Service?," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html. [Kasutatud September 2019].

[60] "What Is Amazon Elastic Container Registry?," Amazon Web Services, Inc., [Online]. Available:

https://docs.aws.amazon.com/AmazonECR/latest/userguide/what-is-ecr.html. [Kasutatud September 2019].

[61] "Amazon ECS Task Definitions," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task_definitions.html. [Kasutatud September 2019].

[62] "What Is Amazon EC2 Auto Scaling?," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html. [Kasutatud September 2019].

[63] "What Is Amazon CloudWatch?," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html. [Kasutatud September 2019].

[64] "What Is AWS Lambda?," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html. [Kasutatud September 2019].

[65] "What Is Amazon API Gateway?," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html. [Kasutatud September 2019].

[66] "What Is Amazon DynamoDB?," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html. [Kasutatud September 2019].

[67] "What Is Amazon CloudWatch?," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html. [Kasutatud September 2019].

[68] "Tutorial: Schedule AWS Lambda Functions Using CloudWatch Events," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/RunLambdaSchedule.html. [Kasutatud September 2019].

[69] "Using Amazon SNS for System-to-System Messaging with an AWS Lambda Function as a Subscriber," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/sns/latest/dg/sns-lambda-as-subscriber.html. [Kasutatud September 2019].

[70] "Using AWS Lambda with Amazon RDS," Amazon Web Services, Inc., [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/services-rds.html. [Kasutatud September 2019].

[71] M. Featonby, "Amazon ECS availability best practices," 8 November 2019. [Online]. Available: https://aws.amazon.com/blogs/containers/amazon-ecs-availability-best-practices/. [Kasutatud Oktoober 2019].

[72] Y. Tao and G. Chen, "An Extensible Universal Reverse Proxy Architecture," IEEE, 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7945939. [Kasutatud Oktoober 2019].

[73] "KnexJS," [Online]. Available: http://knexjs.org. [Kasutatud Oktoober 2019].

[74] "MySQL 5.6 Reference Manual. Data Types," MySql, [Online]. Available: https://dev.mysql.com/doc/refman/5.6/en/data-types.html. [Kasutatud Oktoober 2019].

[75] "MySQL 5.6 Reference Manual. CREATE INDEX Statement," MySql, [Online]. Available: https://dev.mysql.com/doc/refman/5.6/en/create-index.html. [Kasutatud Kasutatud 2019].

[76] "Authorization service," Testlio OÜ, [Online]. Available: https://testlions.atlassian.net/wiki/spaces/EN/pages/32065312/Authorization+service. [Kasutatud Juuli 2019].

[77] "API proxy," Testlio OÜ, [Online]. Available: https://testlions.atlassian.net/wiki/spaces/EN/pages/33199887/API+proxy. [Kasutatud Juuli 2019].

[78] "Build service," Testlio OÜ, [Online]. Available: https://testlions.atlassian.net/wiki/spaces/EN/pages/63996033/Build+service. [Kasutatud Juuli 2019].

[79] "Project service," Testlio OÜ, [Online]. Available: https://testlions.atlassian.net/wiki/spaces/EN/pages/64029042/Project+service+V2. [Kasutatud Juuli 2019].

[80] "Resign service," Testlio OÜ, [Online]. Available: https://testlions.atlassian.net/wiki/spaces/EN/pages/63864998/Resign+service. [Kasutatud Juuli 2019].

[81] "Upload service," Testlio OÜ, [Online]. Available: https://testlions.atlassian.net/wiki/spaces/EN/pages/33197737/Upload+service. [Kasutatud Juuli 2019].

[82] P. Jamshidi, C. Pahl, N. C. Mendonça and S. T. James Lewis, "Microservices: The Journey So Far and Challenges Ahead," Mai 2018. [Online]. Available: https://www.researchgate.net/publication/324959590_Microservices_The_Journey_So_Far_and_Challenges_Ahead. [Kasutatud September 2019].

[83] R. Sturm, C. Pollard and J. Craig, Application Performance Management (APM) in the Digital Enterprise: Managing Applications for Cloud, Mobile, IoT and eBusiness, Morgan Kaufmann, 2017.

[84] "Introduction to New Relic APM," NewRelic, [Online]. Available: https://docs.newrelic.com/docs/apm/new-relic-apm/getting-started/introduction-new-relic-apm. [Kasutatud Detsember 2019].

[85] "Apdex: Measure user satisfaction," NewRelic, [Online]. Available: https://docs.newrelic.com/docs/apm/new-relic-apm/apdex/apdex-measure-user-satisfaction. [Kasutatud Jaanuar 2020].

[86] "Testlio Platform," Testlio OÜ, [Online]. Available: https://testlio.com/platform/. [Kasutatud Detsember 2019].

[87] E. Dasque, "DynamoDBtoCSV," [Online]. Available: http://www.frenchguys.com/wordpress/exporting-dynamodb-data-to-csv/. [Kasutatud Jaanuar 2020].

[88] "Sequel Pro," Sequel Pro, [Online]. Available: https://www.sequelpro.com/. [Kasutatud Jaanuar 2020].

[89] T. Preston-Werner, "Semantic Versioning 2.0.0," [Online]. Available: https://semver.org/. [Kasutatud August 2019].

Lisa 1 – DynamoDB andmebaasi eksport

Tulenevalt varasema keskkondade teenuse tehnoloogilistest probleemidest (ptk 3.1.5) on andmete pärimine kindla struktuuri järgi NoSql andmebaasist äärmiselt keeruline. Probleemist ülesaamiseks on skänneeritud iga NoSql DynamoDB eraldi *AWS* konsoolirakenduse abil, sisuliselt tehes täiemahulise andmete mahakirjutamise (*full dump*) CSV formaati. Antud ülesande kontekstis on võimalik kasutada vabavaralisi tööriistu nagu DynamoDBtoCSV [87]. Tekkinud csv faile on võimalik importida mõnda CSV toega Sql klienti, nagu näiteks SqlPro [88]. Kui erinevate CSV failide tulbad omavad teineteisse viiteid võõr- ja primaarvõtmete näol, siis vastava struktuuri loomisel on võimalik kasutada Sql päringuid originaalis mitte-Sql andmete peal.

Lisa 2 – Andmebaasi tabelite atribuudid

Uue andmemudeli realiseerimiseks on allpool toodud tabelites (Tabel 11, Tabel 12, Tabel 13, Tabel 14, Tabel 15 ja Tabel 16) välja toodud erinevad tabelid, nende atribuudid ning kontekst.

Tabel 11. BUILD.FILE_TYPES atribuudid

Atribuut	**Nullitav**	**Tüüp**	**Pikkus**	**Vaikeväärtus**	**Selgitus**
id	Ei	Integer	11		Primaarvõti ehk failitüüpide identifikaator
type	Ei	Varchar	255		Failitüüp. A la *Android*
is_resigned	Jah	Tinyint	1	NULL	Tõeväärtus (*boolean*) kajastamaks, kas keskkond on resigneerutud (*resigned*). Resigneerimine on kohaldatav ainult Apple iOS mobiilse platvormi koostetele. Muudel juhtudel on väärtuseks NULL.

Tabel 12. EBDB.PROJECT_COLLECTIONS atribuudid

Atribuut	Nullitav	Tüüp	Pikkus	Vaikeväärtus	Selgitus
id	Ei	Integer	11		Primaarvõti ehk projekti keskkonna identifikaator
project_id	Ei	Integer	11		Võõrvõti EBDB.PROJECTS tabeli id väljale ehk viide kirjega seotud projekti identiteedile
build_collection_href	Ei	Varchar	255		BUILDS.COLLECTIONS rakendusliidese põhine ressursilink ehk viide kirjega seotud keskkondade kogumi identiteedile. A la *api.testlio.com/builds/v3/collections/1*
created_at	Ei	Datetime	MySql vaikepikkus	CURRENT_TIMESTAMP	Kirje loomise hetke ajatempel
created_by	Ei	Integer	11		Võõrvõti EBDB.WP_USERS tabeli id väljale ehk viide kirje looja identiteedile
updated_at	Ei	Datetime	MySql vaikepikkus	CURRENT_TIMESTAMP	Iga kirje uuendamise hetke ajatempel, mh ka kirje loomine
updated_by	Jah	Integer	11	NULL	Võõrvõti EBDB.WP_USERS tabeli id väljale ehk viide kirje uuendaja identiteedile
is_ignored	Ei	Tinyint	1	0	Tõeväärtus (*boolean*) kajastamaks, kas seos on kehtiv või mitte
is_primary	Ei	Tinyint	1	0	Tõeväärtus (*boolean*) kajastamaks, kas kollektsioon on ainult lugemiseks (0) või lugemiseks ja kirjutamiseks (1).

Tabel 13. EBDB.TEST_CYCLE_BUILDS atribuudid

Atribuut	Nullitav	Tüüp	Pikkus	Vaikeväärtus	Selgitus
id	Ei	Integer	11		Primaarvõti ehk testimistsükli keskkondade identifikaator
test_cycle_id	Ei	Integer	11		Võõrvõti EBDB.TEST_CYCLES tabeli id väljale ehk viide kirjega seotud testimistsükli identiteedile
build_href	Ei	Varchar	255		BUILD.BUILDS rakendusliidese põhine ressursilink ehk viide kirjega seotud keskkonna identiteedile. A la *api.testlio.com/builds/v3/collections/1/builds/1*
created_at	Ei	Datetime	MySql vaikepikkus	CURRENT_TIMESTAMP	Kirje loomise hetke ajatempel
created_by	Ei	Integer	11		Võõrvõti EBDB.WP_USERS tabeli id väljale ehk viide kirje uuendaja identiteedile
deleted_at	Ei	Datetime	MySql vaikepikkus	0000-00-00 00:00:00	Kirje kustutamise ajatempel
deleted_by	Jah	Integer	11	NULL	Võõrvõti EBDB.WP_USERS tabeli id väljale ehk viide kirje kustutaja identiteedile

Tabel 14. BUILD.COLLECTIONS atribuudid

Atribuut	Nullitav	Tüüp	Pikkus	Vaikeväärtus	Selgitus
id	Ei	Integer	11		Primaarvõti ehk keskkondade kogumi identifikaator
created_at	Ei	Datetime	MySql vaikepikkus	CURRENT_TIMESTAMP	Kirje loomise hetke ajatempel
created_by	Ei	Varchar	255		EBDB.WP_USERS rakendusliidese põhine ressursilink ehk viide kirje looja identiteedile. A la *api.testlio.com/user/v1/users/1*
deleted_at	Jah	Datetime	MySql vaikepikkus	NULL	Kirje kustutamise ajatempel
deleted_by	Jah	Varchar	255	NULL	Võõrvõti EBDB.WP_USERS tabeli id väljale ehk viide kirje kustutaja identiteedile. A la *api.testlio.com/user/v1/users/1*

Tabel 15. BUILD.BUILDS atribuudid

Atribuut	Nullitav	Tüüp	Pikkus	Vaikeväärtus	Selgitus
id	Ei	Integer	11		Primaarvõti ehk keskkonna identifikaator
collection_id	Ei	Integer	11		Võõrvõti BUILD.COLLECTIONS tabeli id väljale ehk viide kirjega seotud keskkondade kogumi identiteedile
name	Ei	Varchar	255		Keskkonna nimi

Atribuut	Nullitav	Tüüp	Pikkus	Vaikeväärtus	Selgitus
version	Jah	Varchar	255	NULL	Keskkonna versioon. Tavapärane praktika on semantilise versioneerimise (*semantic versioning*) [89] kasutamine. A la *3.2.1*
tag	Jah	Varchar	255	NULL	Keskkonna märge. Vabatekstiline märge keskkondade paremaks struktureerimiseks. A la *regression*
notes	Jah	Text	Mysql vaikepikkus	NULL	Kliendi reliisimärkmed (*release notes*)
created_at	Ei	Datetime	MySql vaikepikkus	CURRENT_TIMESTAMP	Kirje loomise hetke ajatempel
created_by	Ei	Varchar	255		EBDB.WP_USERS rakendusliidese põhine ressursilink ehk viide kirje looja identiteedile. A la *api.testlio.com/user/v1/users/1*
updated_at	Ei	Datetime	MySql vaikepikkus	CURRENT_TIMESTAMP	Iga kirje uuendamise hetke ajatempel, mh ka kirje loomine
updated_by	Jah	Varchar	255	NULL	EBDB.WP_USERS rakendusliidese põhine ressursilink ehk viide kirje uuendaja identiteedile. A la *api.testlio.com/user/v1/users/1*
deleted_at	Jah	Datetime	MySql vaikepikkus	NULL	Kirje kustutamise ajatempel
deleted_by	Jah	Varchar	255	NULL	EBDB.WP_USERS rakendusliidese põhine ressursilink ehk viide kirje kustutaja identiteedile. A la *api.testlio.com/user/v1/users/1*

Tabel 16. BUILD.FILES atribuudid

Atribuut	Nullitav	Tüüp	Pikkus	Vaikeväärtus	Selgitus
id	Ei	Integer	11		Primaarvõti ehk faili identifikaator
build_id	Ei	Integer	11		Võõrvõti BUILD.BUILDS tabeli id väljale ehk viide kirjega seotud keskkonna identiteedile
file_type_id	Ei	Integer	11		Võõrvõti BUILD.FILE_TYPES tabeli id väljale ehk viide kirjega seotud failitüübi identiteedile
url	Ei	Varchar	255		Veebilink faili allalaadimiseks. Link võib viidata kas kliendi veebikeskkonnale või Testlio pilvepõhisele repositooriumile
created_at	Ei	Datetime	MySql vaikepikkus	CURRENT_TIMESTAMP	Kirje loomise hetke ajatempel
created_by	Ei	Varchar	255		EBDB.WP_USERS rakendusliidese põhine ressursilink ehk viide kirje looja identiteedile. A la *api.testlio.com/user/v1/users/1*
Atribuut	Nullitav	Tüüp	Pikkus	Vaikeväärtus	Selgitus
deleted_at	Jah	Datetime	MySql vaikepikkus	NULL	Kirje kustutamise ajatempel
deleted_by	Jah	Varchar	255	NULL	Võõrvõti EBDB.WP_USERS tabeli id väljale ehk viide kirje kustutaja identiteedile. A la *api.testlio.com/user/v1/users/1*

Lisa 3 – Avaliku rakendusliidese integratsioonijuhis

Avaliku rakendusliidese integratsioonijuhis on klientidega jagatav dokument, mille eesmärk on luua arusaam rakendusliidese kasutusest. Dokument on loodud lõputööle tuginedes ning on inglise keelne.

Testlio Build Upload API

for external Partners Version 1.0

Testlio Build Upload API allows partners to upload build files and related metadata directly to Testlio Build Service for more convenient and faster handling for builds Testlio needs to test and make available for testers.

Changelog

Date	Change
November 29, 2019	Added support to send URL as build
June 3, 2019	New optional field "*notes*" added to POST request for accepting release notes.

Uploading build

Testlio accepts two different types of builds:

- ● Actual build files (ipa, apk, zip etc)
- ● "Web builds" - URLs

To upload a build to Testlio, you'll need to send a POST request of type *multipart/form-data* or *application/json* depending if you are sending an actual file or URL. The request should contain the file you would like to upload or Web URL, as well as the metadata for creating a build.

URL (where to make a post request)

● URL: https://api.testlio.com/build/v3/collections/{collectionId}/builds ● Arguments:

○ collectionId (integer) - id of a collection

In first stage Testlio will provide list of available collections depending on setup agreed between client and Testlio.

Authorization

Authentication to the API is performed via JWT. Token is provided to client by Testlio. Add your token in request header as following:

Authorization: Bearer [CLIENT_AUTH_TOKEN]

Parameters

file (file, required) - only if sending actual file
A file to upload. The file should follow the specifications of RFC 2388 (which defines file transfers for the multipart/form-data protocol).

url (string, required, maximum length 255 characters) - only if sending Web URL instead of file
Web URL. Use in if testers need to test websites etc.

name (string, required, maximum length 255 characters)
Name of a build. This is what is stored in Testlio systems in order to identify specific build.

tag (string, optional, maximum length 255 characters)
Comma separated list of tags or label client wants to send to Testlio.

version (string, optional, maximum length 255 characters) Version of the build. Used to specify build.

notes (string, optional, maximum length 5000 characters, remaining content will be truncated after that)
Release notes as text.

isResigned (boolean, optional)
If build sent to Testlio is resigned or not. Non-resigned builds will we resigned internally by Testlio in order to distribute them amongst Testlio testers.

Difference between sending file and URL

	Build file	URL
Request content type	multipart/form-data	application/json
file	required	do not send
url	do not send	required
name	required	optional
tag	optional	optional
version	optional	optional
notes	optional	optional
isResigned	optional	do not send

Response

Testlio Build API responds always with . Successful request to build service will return status code along with the build's metadata once the build upload has been fully completed:

201 Created

```
HTTP/1.1 201
Content-Type: application/json
{
  "id": 1,
  "name": "Testing build",
  "version": null,
  "tag": null,
  “notes”: null

}
```

Example - build file

Following is the cURL example request made against Testlio Build API to upload a build file with successful response:

POST https://api.testlio.com/build/v3/collections/123/builds

```
curl --request POST https://api.testlio.com/build/v3/collections/123/builds
\
  -H "Authorization: Bearer [CLIENT_AUTH_TOKEN]" \
  -F name="iOS build 2.1" \
  -F version=”2.1”
  -F file="@/path/to/a/file.ipa"
```

Response

```
HTTP/1.1 201
Content-Type: application/json

{
"id": 23,
"name": "iOS build 2.1", "version": “2.1”, "tag": null,
“notes”: null

}
```

Example - web URL

Following is the cURL example request made against Testlio Build API to upload a build file with successful response:

POST https://api.testlio.com/build/v3/collections/123/builds

```
curl --request POST https://api.testlio.com/build/v3/collections/123/builds
\
  -H "Authorization: Bearer [CLIENT_AUTH_TOKEN]" \
  -H 'Content-Type: application/json' \
  -d $'{
  "tag": "random-tag",
```

```
  "url": "http://www.testlio.com"

}'
```

Response

```
HTTP/1.1 201
Content-Type: application/json
{
  "id": 23,
  "url": "http://www.testlio.com",
  "version": null,
  "Tag": ”random-tag”,
  “notes”: null

}
```