LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

ROBOTIZĒTU PROCESU AUTOMATIZĀCIJAS RISINĀJUMU PIEGĀDE

BAKALAURA DARBS

Autors: **Elvis Plācis**
Studenta apliecības Nr.: ep15021
Darba vadītājs: M.dat. Andris Bozis

RĪGA 2019

ANOTĀCIJA

Bakalaura darba mērķis ir izanalizēt un pētīt programmatūras piegādes specifiku un tās darbības izolācijas paņēmienus, kā arī pētīt un salīdzināt gan maksas, gan atvērtā pirmkoda robotizētu procesu automatizācijas platformu sniegtās iespējas un piegādes īpatnības.

Bakalaura darbā sākotnēji tiek pētīts programmatūras piegādes process un šī procesa automatizācija, izvērtējot automatizēto risinājumu prasības un ieguvumus. Tālākajā darba gaitā tiek izpētīta piegādātās programmatūras izolācija, izvērtējot iespējamo variantu pozitīvās un negatīvās iezīmes. Turpmākajā darba gaitā tiek pētītas gan atvērtā pirmkoda, gan maksas robotizētu procesu automatizācijas izstrādes platformas, izvērtējot to sniegtās iespējas, kā arī salīdzinot tehniskās prasības un piegādes īpatnības. Gala rezultātā tiek izdarīti secinājumi par programmatūras piegādi un izolāciju robotizētu procesu automatizācijas risinājumu piegādē.

Atslēgas vārdi: programmatūras piegāde, DevOps, programmatūras izolācija, RPA, robotizētu procesu automatizācija

ABSTRACT

DELIVERY OF ROBOTIC PROCESS AUTOMATION SOLUTIONS

Bachelor thesis objective is to analyse and explore software delivery specifics and methods of software isolation, as well as explore and compare paid and open source robotic process automation platforms, their provided opportunities and delivery peculiarities.

At the beginning of bachelor thesis software delivery process and its automation are being researched, meanwhile evaluating automated software delivery solution requirements and benefits. Further into bachelor thesis software isolation is being researched, evaluating potential solution positive and negative aspects. After that in bachelor thesis is continued with open source and paid robotic process automation development platform research, evaluating features of platforms and comparing technical system requirements and delivery peculiarities. At the end of bachelor thesis conclusions are drawn about software delivery and isolation for robotic process automation solution delivery.

Keywords: software delivery, DevOps, software isolation, RPA, robotic process automation

SATURA RĀDĪTĀJS

APZĪMĒJUMU SARAKSTS

1. DevOps – programmatūras izstrādes prakses kopums
2. IT – informācijas tehnoloģija
3. Cgroups – saīsinājums no kontroles grupas, kas ir Linux kodola funkcionalitāte, kura nodrošina procesu grupas resursu lietojumu
4. IPC – starpprocesu saziņa
5. PID – procesa identifikators
6. UTS – UNIX laika dalīšanas nosaukumvieta paredzēta, lai izolētu resursdatora nosaukumu un domēna vārdu
7. RPA – robotizētu procesu automatizācija
8. SQL – strukturēta vaicājumvaloda
9. ElasticSearch – atvērtā pirmkoda klastera meklēšanas un analīzes programma
10. REST API – lietotnes programmas saskarne, kura izmanto HTTP pieprasījums
11. Windows Server – Microsoft izstrādāta serveru operētājsistēma
12. Microsoft SQL Server – Microsoft izstrādāta relāciju datubāžu pārvaldības sistēma
13. VMWare – kompānija, kura darbojas virtualizācijas un mākoņdatošanas programmatūras nodrošināšanā un izstrādē.
14. Citrix XenDesktop – kompānijas "Citrix Systems" izstrādāts darbstaciju virtualizācijas produkts.
15. Oracle VirtualBox – kompānijas "Oracle" izstrādāta atvērtā pirmkoda virtuālo mašīnu izveides, pārvaldības un darbināšanas platforma
16. Microsoft Hyper-V – kompānijas "Microsoft" izstrādāts virtuālo mašīnu pārraugs, kurš darbojas "Windows" operētājsistēmas vidē.
17. Microsoft .NET – kompānijas "Microsoft" izstrādāts programmatūras ietvars.
18. Powershell – kompānijas "Microsoft" izstrādāts uzdevumu automatizācijas un konfigurācijas pārvaldības ietvars.
19. WebDeploy – kompānijas "Microsoft" izstrādāts konfigurācijas un satura sinhronizācijas rīks.
20. Internet Information Services Manager – kompānijas "Microsoft" izstrādāts tīmekļa servera pārvaldības rīks
21. URL Rewrite – kompānijas "Microsoft" izstrādāts tīmekļa adreses pārveidošanas rīks
22. Kibana – "ElasticSearch" datu vizualizācijas programmatūra

23. Redis – atvērtā pirmkoda adresējamā atmiņā strādājoša datu struktūru glabātuve
24. Windows – kompānijas "Microsoft" izstrādāta operētājsistēma
25. Linux – atvērtā pirmkoda operētājsistēmu grupa, kura bāzēta uz Linux kodola.
26. Google Chrome – kompānijas "Google" izstrādāta interneta pārlūkprogramma
27. Microsoft Edge – kompānijas "Microsoft" izstrādāta interneta pārlūkprogramma
28. CPU – centrālais procesors
29. GB – gigabaits, informācijas mērvienība, tas atbilst 1 000 000 000 baitiem.
30. RAM – brīvpiekļuves atmiņa
31. HDD – cietais disks, ierīce, kurā paliekoši var glabāt datus
32. DPI – punkti collā, standarts, kurš tiek izmantots ekrānu izšķirtspējas noteikšanā
33. Subversion – atvērtā pirmkoda versiju pārvaldības sistēma
34. PostgreSQL – atvērtā pirmkoda objektu relāciju datubāzes pārvaldības sistēma
35. Oracle Database – kompānijas "Oracle" izstrādāta relāciju datubāzes pārvaldības sistēma
36. SSL – firmas Netscape Communications izstrādāts protokols drošas un privātas saziņas nodrošināšanai internetā.
37. Internet Explorer – kompānijas "Microsoft" izstrādāta interneta pārlūkprogramma
38. Firefox – atvērtā pirmkoda interneta pārlūkprogramma
39. Visual SVN Server – brīvprogrammatūras serveru pakotne Windows serveriem, paredzēta "Subversion" uzstādīšanai un pārvaldīšanai.
40. Java – kompānijas "Sun Microsystems" izstrādāta objektorientēta programmēšanas valoda.
41. Java SE Runtime – kompānijas "Sun Microsystems" izstrādāta programmatūra, kuras mērķis ir darbināt citu programatūru, kas izmanto "Java".
42. Microsoft Silverlight – kompānijas "Microsoft" izstrādāts bezmaksas interneta pārlūka spraudnis.
43. Adobe Flex – atvērtā pirmkoda lietotņu satvars tīmekļa lietojumprogrammu izstrādei un uzturēšanai.
44. Microsoft MODI – kompānijas "Microsoft" izstrādāta programmatūra, kura ir daļa no "Microsoft Office" un nodrošina skanētu dokumentu transformāciju uz teksta formātu.
45. Transym TOCR – simbolu atpazīšanas programmatūra ar mērķi nodrošināt skanētu tekstu transformāciju uz teksta formātu.
46. Jenkins – atvērtā pirmkoda automatizācijas serveris.
47. XML - paplašināmās marķēšanas valoda
48. RPC – attālās procedūras izsaukums

49. Jython – “Python” programmēšanas valodas implementācija ar mērķi tās darbību nodrošināt “Java” platformā.
50. Python – objektorientēta, augsta līmeņa programmēšanas valoda
51. Apache HTTP – atvērtā pirmkoda tīmekļa serveris
52. Win32 – “Windows” lietojumprogrammas interfeiss 32-bit lietotņu izstrādei
53. WinForms – “Windows Forms” ir grafiska klašu bibliotēka, kas ir daļa no “Microsoft .NET”
54. WPF – “Windows Presentation Foundation “ ir lietotāja saskarnes ietvars
55. FTP - datņu pārsūtīšanas protokols jeb protokols FTP ir TCP/IP steka lietojuma slāņa protokols failu pārsūtīšanai starp datoriem.
56. Java Web Start – kompānijas “Sun Microsystems” izstrādāts ietvars
57. SSH - tīkla protokols, kurš nodrošina drošu datu apmaiņu starp divām tīkla ierīcēm.
58. MongoDB – dokumentu orientēta “NoSQL” datubāze
59. NoSQL – klastera datubāze, kura izmanto dinamiskas shēmas nestrukturizētiem datiem
60. PyMongo – Python rīku kopums ar mērķi nodrošināt darbību ar MongoDB
61. JSON - datu apmaiņas formāts
62. Selenium – pārvietojams ietvars, kurš ir paredzēts tīmekļa lietojumprogrammu testēšanai
63. SOAP - vienkāršais objektpiekļuves protokols
64. CSV – teksta fails, kurš satur datu sarakstu, kuri atdalīti ar atdalītāja vērtību
65. PDF - portatīvā dokumenta formāts ir elektronisku dokumentu datņu formāts.
66. API - lietojumprogrammas saskarne ir iepriekš definētu klašu, procedūru, funkciju, struktūru un konstanšu kopums, kas tiek pasniegts kā pielikums, kuru iespējams izmantot ārējiem programmatūras produktiem.
67. Docker – konteineru platforma
68. Fedora Linux – bezmaksas Linux distribūcija, kas radusies no “Red Hat Linux” distribūcijas.
69. Oracle SQL plus – “Oracle” izstrādāts komandrindas rīks
70. Oracle Weblogic Server – “Oracle” izstrādāts lietotņu serveris
71. Pabot – “RobotFramework” paralēlas izpildes rīks
72. Python Pip – “Python” pakotņu pārvaldnieks
73. Gecko Driver – “Mozilla Firefox” interneta pārlūkprogrammas dzinis
74. CentOS – bezmaksas “Linux” distribūcija.
75. Oracle Linux 7 – kompānijas “Oracle” izstrādāta bezmaksas “Linux” distribūcija.

IEVADS

Bakalaura darba mērķis ir izanalizēt un pētīt programmatūras piegādes specifiku un tās darbības izolācijas paņēmienus, kā arī pētīt un salīdzināt gan maksas, gan atvērtā pirmkoda robotizētu procesu automatizācijas platformu sniegtās iespējas un to piegādes īpatnības.

Bakalaura darba pamatā ir reāla darba nepieciešamība izvēlēties, izstrādāt un piegādāt robotizētu procesu automatizācijas risinājumu. Šī iemesla dēļ tika veikta programmatūras piegādes un izolācijas veidu informācijas atlase, izpēte un analīze, kā arī robotizētu procesu automatizācijas platformu sniegto iespēju un infrastruktūras prasību salīdzinājums, lai spētu izstrādāt un piegādāt labāko iespējamo risinājumu specifiskā klienta sarežģītajos apstākļos un vairāku programmatūras piegādātāju vidē.

Lai sasniegtu šo mērķi, ir jāveic šādi uzdevumi:

- Jāveic izpēte par programmatūras piegādi un šī procesa automatizāciju.
- Jāveic izpēte par programmatūras izolācijas iespējām.
- Jāveic izpēte par robotizētu procesu automatizācijas platformām un to iespējām.
- Jāprecizē klienta specifiskie apstākļi, prasības un ierobežojumi.
- Jāveic praktiskā piegādes risinājuma izstrāde un realizācija.
- Jāizdara secinājumi par veikto izpēti un praktiski piegādāto risinājumu.

Bakalaura darbā sākotnēji tiek veikta izpēte, kas saistīta ar programmatūras piegādi un šī procesa automatizāciju. Pēc tam tiek veikta izpēte par programmatūras izolācijas iespējām un robotizētu procesu automatizācijas risinājumu izstrādes platformām. Tālāk tiek veikta klienta specifiskā risinājuma prasību apkopošana un definēšana, kam seko piegādes risinājuma izstrāde un realizācija, noslēgumā apkopojot veiktās izpētes rezultātus un izdarot secinājumus par robotizētu procesu automatizācijas risinājumu piegādi.

1. PROGRAMMATŪRAS PIEGĀDE

Programmatūras piegādes procesā ietilpst visas aktivitātes, kas nepieciešamas, lai programmatūru piegādātu no izstrādātājiem līdz gala lietotājam. Programmatūras piegādes procesā ietilpst daudzas aktivitātes, kas var būt vai nebūt saistītas savā starpā, nereti tām ir definēta secība, kādā tās ir jāizpilda. [1]

Tādēļ, ka katrs programmatūras gala produkts ir unikāls savā funkcionalitātē un prasībās, kā arī katra gala lietotāja sistēma ir unikāla, šim procesam ir jābūt pielāgotam specifiskajai situācijai un mērķim, lai varētu panākt vēlamo gala rezultātu un veiksmīgu izpildi. [2]

Vienkāršām gala sistēmām piegādes procesa laikā ir pietiekami izveidot specifiskus skriptus, kas izpildītu vēlamās komandas, lai mērķa sistēmā uzstādītu nepieciešamo programmatūru un tās darbībai nepieciešamās atkarības.

Sarežģītākos gadījumos gala lietotāja sistēma parasti atrodas uz citiem fiziskiem serveriem, kā arī izstrādes laikā programmatūra tiek piegādāta no izstrādātāju vidēm uz testēšanas vidēm un tikai pašās programmatūras izstrādes ķēdes beigās atrodas gala lietotāja izmantotā sistēma.

Šī iemesla dēļ ir jāspēj nodrošināt programmatūras piegāde cauri visiem tās izstrādes posmiem, garantējot, ka tā strādā tieši tā, kā paredzēts visās mērķa vidēs.

Programmatūras piegāde sevī ietver:

- Programmatūras arhitektūru.
- Programmatūras izstrādāto risinājumu.
- Aparatūras un infrastruktūras arhitektūru.
- Veiktspējas testēšanu.
- Drošības testēšanu.
- Piegādes procesu un plānotu risinājumu uzlabojumu piegādei.
- Atbilstību likumdošanai. [3].

Sarežģītos DevOps praksē izmantotos nepārtrauktās piegādes gadījumos, kā arī programmatūras kā pakalpojuma sistēmās ir iespējams scenārijs, kad dažādu versiju un konfigurāciju programmatūra tiek darbināta paralēli, lai nodrošinātu vidi ārējiem un iekšējiem klientiem vai, piemēram, pakāpeniski piegādātu programmatūru dažādām mērķa atrašanās vietām. [2]

1.1. DevOps

DevOps ir noteiktas programmatūras piegādes prakses apkopojums, kas ietver procesus starp programmatūras izstrādi un IT komandām, tā mērķis – kompilēt, testēt un piegādāt programmatūru ātrāk un stabilāk.

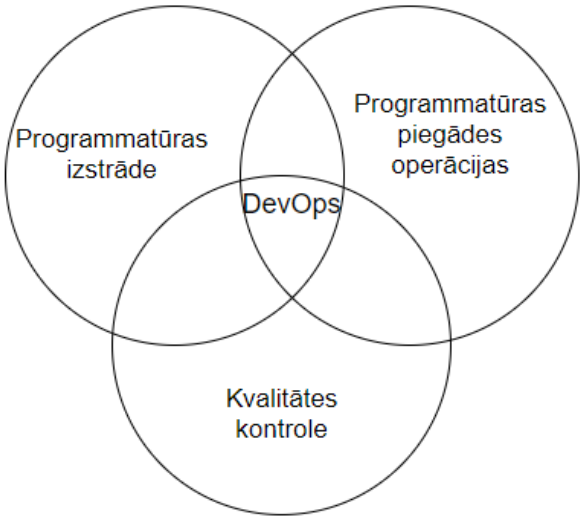
DevOps būtība ir izstrādāta, ievērojot kultūru, kurā komandas, kuras vēsturiski ir strādājušas nodalīti viena no otras, sastrādājas, lai uzlabotu gala produkta piegādi un stabilitāti.

DevOps ideja izveidojās starp 2007. un 2008. gadu, kad IT operāciju un programmatūras izstrādes kopienas nonāca pie secinājuma, ka ir nepieciešama labāka sadarbība, lai novērstu funkcionālas problēmas abu grupu sadarbībā. Problēma radās, jo tradicionālā programmatūras izstrādes modelī viss tika veidots gan funkcionāli, gan organizatoriski nodalīti no cilvēkiem, kuriem programmatūra ir jākompilē un jāatbalsta.

Programmatūras izstrādātājiem un ekspluatācijas profesionāļiem bija savstarpēji nodalīti mērķi un prasības, kā arī efektivitātes rādītāji, pēc kuriem tie tika vērtēti. Nereti tie strādāja vienā uzņēmumā, bet dažādās nodaļās, stāvos vai telpās. Rezultātā tas noveda pie savstarpēji izolētām komandām, kuras katra strādāja pie saviem mērķiem, un pat pie intensīva darba nereti viss rezultējās ar problēmām piegādē un neapmierinātiem klientiem. [4]

1.1.1. DevOps kultūra

DevOps kultūra galvenokārt balstās uz sadarbību, precīzāk, uz daudzfunkcionālu sadarbību. Neviens izstrādes rīks vai automatizācijas risinājums nav derīgs, ja tas netiek izmantots apvienojumā ar vēlmi sadarboties starp iesaistītajām komandām.



1.1. att. **DevOps izstrādes prakse**

Attēlā 1.1 redzams, ka DevOps ir programmatūras izstrādes prakšu kopums, kas ietver programmatūras izstrādi, informācijas tehniskās operācijas un kvalitātes kontroli, ar mērķi saīsināt sistēmas izstrādes ciklu, vienlaikus piegādājot funkcionalitāti, labojumus un atjauninājumus, strādājot ciešā sadarbībā ar biznesa mērķiem. [4][5][6]

DevOps prakse strādā līdzīgi spējai programmatūras izstrādei, papildus iekļaujot arī programmatūras piegādi. Veidojot uz projektu vai produktu orientētas komandas, iespējams aizstāt komandas, kas bāzējas uz noteiktu funkciju, tā iesaistot kvalitātes kontroles, produktu pārvaldības, dizaina, piegādes, projektu pārvaldības un citas prasmes projekta prasībām.

Strauja pāreja uz projektu bāzētām komandām ir pārāk sarežģīta, taču to ir iespējams darīt pakāpeniski. Izstrādātāju komanda var piesaistīt vēlamos speciālistus no programmatūras piegādes komandas plānošanas procesam un citiem ikdienas darbiem. Programmatūras piegādes komanda var piesaistīt galvenos izstrādātājus, tādējādi veidojot elastīgu un dabīgu veidu, kā sekot līdzi visiem projektiem, idejām un problēmām. Šāds laika patēriņš starp iesaistītajām komandām atmaksājas, jo tiek atvieglota un padarīta efektīvāka programmatūras produkta piegādes pārvaldība un kritisku situāciju analīze.

Krīzes situācijas ir lietderīga metode kā testēt DevOps kultūras efektivitāti. Vai visas iesaistītās puses un klienta atbalsts kopīgi cenšas atrisināt problēmu kā vienota komanda? Vai visi sāk ar spriedumu, ka viņu komanda ir pieņēmusi labāko iespējamo lēmumu ar informāciju un resursiem, kas tiem bija pieejami attiecīgajā brīdī? Vai brīdī, kad saskaramies ar problēmu, galvenais mērķis ir to atrisināt, nevis meklēt atbildīgo? Ja atbilde uz šiem jautājumiem ir pozitīva, tad tas ir labs identifikators, ka komanda funkcionē ar DevOps kultūru tās pamatos. [4]

Veiksmīgi izmantojot DevOps procesus, ir atvērta saskarsme starp komandām un tās regulāri komunicē savā starpā. Visu komandu mērķi ir sinhronizēti un vienoti, kā arī tās spēj pielāgoties situācijai pēc nepieciešamības. Ir vienots uzskats, ka klienta apmierinātība ar saņemto pakalpojumu ir gan projekta pārvaldības atbildība, gan izstrādātāju komandas atbildība. Ir kopīga sapratne, ka DevOps nav vienas komandas atbildība un darbs, bet gan visu iesaistīto pušu darbs.

1.1.2. DevOps prakses ieguvumi

Protams, ar DevOps prakses izmantošanu izmaiņas nenotiek īsā laika posmā, taču šo praksi ir iespējams ieviest soli pa solim, izmantojot mazus un pakāpeniskus soļus gan mazos, gan lielos programmatūras izstrādes projektos.[4]

- DevOps principi balstās uz savstarpēju atbildību, caurredzamību un ātru atbildes sniegšanu, šie noteikumi ir pamatā veiksmīgā DevOps prakses izmantošanā.

- Izmantojot šīs prakses principus, ir nepieciešams sagatavot programmatūru piegādei pēc iespējas ātrāk un efektīvāk, veiksmīgi izmantojot automatizāciju un atsauksmju cikliskumu. Tādējādi uzlabojot kļūdu atrašanas ātrumu un uzturot optimālu izstrādātāju komandas darbu, samazinās nepieciešamība pārslēgties starp dažādiem darbiem. Izmantojot standartizētus rīkus un procesus, komandām ir iespējams palielināt produktivitāti un garantēt programmatūras piegādi ar mazāk apgrūtinājumiem.
- Paātrinot izstrādātā programmatūras produkta atsauksmju iegūšanu, ir iespējams optimāli izmantot komandā pieejamos resursus. Pilnīgi caurskatāma izstrāde un komunikācija atļauj DevOps komandai samazināt dīkstāvi un atrisināt problēmas ātrāk. Gadījumā, ja kritiskas problēmas netiek atrisinātas ātri, cieš klienta apmierinātība ar pakalpojumu, kas rezultātā palielina spiedienu uz izstrādātāju un piegādes komandām. Atvērta komunikācija starp šīm komandām atvieglo problēmu atrašanu un risināšanu, tādējādi ļaujot novērst izveidojušās problēmas daudz efektīvāk.
- Neplānots darbs ir realitāte no jebkuras programmatūras izstrādes komandas ikdienas, taču nereti tieši šis darbs ir tas, kas visvairāk ietekmē komandas produktivitāti. Izstrādājot procesus un veicot strikti definētu darba procesu sastādīšanu, visas iesaistītās komandas var labāk tikt galā ar neplānoto darbu un efektīvi turpināt plānoto darbu. Pārslēgšanās un neplānotā darba prioritāšu kārtošana, strādājot vairākās komandās un sistēmās, nereti ir neefektīva un tālu no reāli darāmā darba. Taču, uzlabojot procesu caurskatāmību un aktīvi iesaistoties komunikācijā, iesaistītās komandas var daudz efektīvāk paredzēt, plānot un savā starpā sadalīt neplānotos darbus.

1.1.3. Automatizācija DevOps praksē

Automatizācijas risinājumu izveidošana samazina atkārtojama darba apjomu un izveido atkārtojamus procesus, tādējādi palīdzot veidot uzticamas sistēmas.

Apkopot, testēt, uzstādīt un pieskatīt automatizētus risinājumus ir standarta sākuma punkts komandām, kurām šādi risinājumi šobrīd neeksistē. Šāda prakse ir vērtīga, jo tā atvieglo visu iesaistīto komandu darbu, un, izveidojot šādu sistēmu, pozitīvu rezultātu iegūst visas iesaistītās puses. Komandas, kuras tikai uzsāk darbu pie automatizācijas, parasti to sāk ar nepārtraukto piegādi. Ar šīs prakses palīdzību visas veiktās programmatūras izmaiņas tiek automatizēti testētas un piegādātas cauri visiem izstrādes soļiem, nereti izmantojot vides, kuras ir bāzētas uz mākoņpakalpojumiem. Kā ir iespējams paredzēt, nepārtraukto piegādi ir sarežģīti un laikietilpīgi ieviest, taču ieguvumi, to izmantojot, ir tā vērti. [4][5]

Datora izpildīti testi ir viennozīmīgāki un uzticamāki nekā cilvēka veiktie. Ar šādu testu palīdzību ir iespējams ātrāk atrast kļūdas programmatūrā un sistēmas drošības caurumus, sniedzot iespēju izstrādātājiem tos ātrāk un efektīvāk novērst. Piegādes komandai ar automatizētu piegādes mehānismu palīdzību cauri visiem izstrādes soļiem tiek samazināts risks un pārsteigumi brīdī, kad programmatūru būs nepieciešams piegādāt klientam.

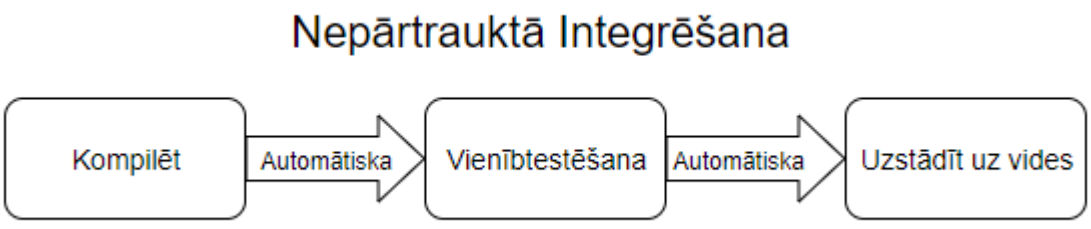
Vienā no pirmajām DevOps praksēm parādījās idejas par to, ka konfigurāciju vajadzētu versionēt līdzīgi kā programmatūras kodu. Izstrādātāji tiecas pēc modulārām un ērti pārvietojamām lietotnēm, jo tās ir uzticamākas un vieglāk uzturamas. Šo ideju ir iespējams paplašināt līdz infrastruktūrai, kura darbina šīs lietotnes un programmatūru kā tādu, neņemot vērā, vai tā tiek darbināta uz mākoņpakalpojumiem vai paša uzņēmuma tīklā.

Taisnība, ka sistēmas vienmēr mainās, taču ir iespējams izveidot priekšstatu par nemainību, izmantojot kodu, kurš kontrolē un nosaka, kā visam ir jāstrādā. Šādā veidā pārvietot programmatūru ir ātrāk un uzticamāk nekā novērst problēmas serveru līmenī, tādējādi samazinot risku. Gan izstrādes, gan piegādes risinājumi var savā darbā ieviest jaunas valodas un tehnoloģijas, izmantojot nodrošinājuma kodu un savā starpā nododot informāciju par izmaiņām. Šādā veidā nesaderības problēmas tiek identificētas uzreiz, nevis piegādes cikla vidū.[6]

Konfigurācija kā kods un nepārtrauktā piegāde nav vienīgie automatizācijas veidi, kurus izmanto DevOps pasaulē, taču tie tika pieminēti, jo tie atvieglo sadarbību un komunikāciju starp programmatūras izstrādi un piegādi. Brīdī, kad DevOps izmanto automatizētu programmatūras piegādi un testēšanu cauri visām vidēm, kuras tiek kontrolētas un nodrošinātas identiski, tiek novērsta problēma, kad programmatūra strādā tikai izstrādātāja vidē.

1.2. Nepārtrauktā integrēšana

Nepārtrauktā integrēšana ir programmatūras izstrādes prakse, kurā izmaiņas tiek sapludinātas galvenajā programmatūras versiju kontroles rīka zarā pēc iespējas biežāk.



1.2. att. **Nepārtrauktās integrēšanas vizuāls piemērs**

Izstrādātās programmatūras izmaiņas tiek automātiski kompilētas un pārbaudītas, izmantojot automātisko testēšanu. Izmantojot nepārtraukto integrēšanu, tiek samazināts risks pēc programmatūras piegādes saskarties ar sapludināšanas problēmām galvenajā programmatūras zarā. Nepārtrauktās integrēšanas ieguvumi ir saistīti ar to, ka uzsvars tiek likts uz automātisku testēšanu pēc mazām izmaiņām programmatūrā, rezultātā ļaujot atrast kļūdas agrākā izstrādes posmā. [7]

Nepārtraukta programmatūras uzstādīšana – programmatūra tiek uzstādīta tiklīdz tiek veiktas izmaiņas, ideālā gadījumā katras uzstādīšanas starpība ir viens izmaiņu kopums.

Automātiskā testēšana – programmatūra vai risinājums, kas pārbauda izstrādāto programmatūru, tādā veidā nodrošinot kvalitāti. Automatizētie testi var izsaukt noteiktas darbības lietotāja saskarnē vai sistēmas servisos.

Šī programmatūras piegādes prakse strādā divās daļās: pirmā daļa pārbauda, vai izstrādātais kods strādā bez tehniskām kļūdām, otrā pārbauda, vai tas strādā tā, kā plānots. Drošākais veids, kā to darīt, ir izmantojot automātisko testēšanu, lai to pārbaudītu visos līmeņos. [7]

Nepārtrauktās integrēšanas prasības:

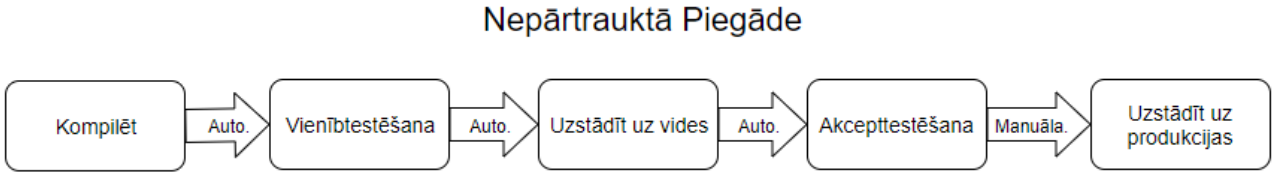
- Nepieciešama automātisko testu izveide katrai jaunajai programmatūras funkcionalitātei, uzlabojumam vai kļūdu labojumam.
- Nepieciešams nepārtrauktās integrēšanas serveris, kas spēj sekot līdzi izmaiņām galvenajā programmatūras versiju kontroles rīka zarā.
- Izstrādātājiem jāsapludina veiktās programmatūras izmaiņas pēc iespējas biežāk.

Nepārtrauktās integrēšanas ieguvumi:

- Samazināts kļūdu skaits programmatūras piegādē uz produkciju, jo izstrādātās programmatūras vai izmaiņu pārbaude notiek agrākā izstrādes stadijā, regulāri izmantojot automātiskos testus.
- Izstrādātās programmatūras apkopošana un piegāde ir atvieglota, jo sapludināšanas problēmas ir novērstas agrāk.
- Izstrādātājiem ir mazāk jāpārslēdzas starp veicamajiem darbiem, jo kļūdas tiek atrastas ātrāk, jo ātrāk tās var novērst.
- Testēšanas izmaksas tiek samazinātas, jo nepārtrauktās integrēšanas serveris var izpildīt automātiskos testus neskaitāmas reizes, samazinot atkārtotu darba apjomu.
- Kvalitātes kontroles komandai ir jāvelta mazāk laiks testējot programmatūru, tādēļ tā var vairāk fokusēties uz svarīgiem uzlabojumiem kvalitātes kontroles procesā.

1.3. Nepārtrauktā piegāde

Nepārtrauktā piegāde ir nākamais solis piegādes automatizācijai no nepārtrauktās integrēšanas, to atšķirība ir tāda, ka programmatūra tiek automātiski piegādāta uz nākamajiem izstrādes soļiem. Šādai programmatūras piegādes metodei ir nepieciešama ne tikai automātiskā testēšana, bet arī programmatūras piegādes automatizācija, kuru ir iespējams izpildīt jebkurā brīdī.



1.3. att. **Nepārtrauktās piegādes vizuāls piemērs**

Teorētiski, izmantojot nepārtraukto piegādi, ir iespējams piegādāt programmatūru jebkurā brīdī, kad klients to pieprasa. Tomēr, lai pēc iespējas efektīvāk izmantotu nepārtraukto piegādi, būtu nepieciešams programmatūru nogādāt produkcijas vidēs pēc iespējas biežāk, šādā veidā piegādājot izstrādātās izmaiņas mazākās daļās un atvieglojot un paātrinot iespējamo kļūdu cēloņu identificēšanu. [8] [9]

Nepārtrauktās piegādes prasības:

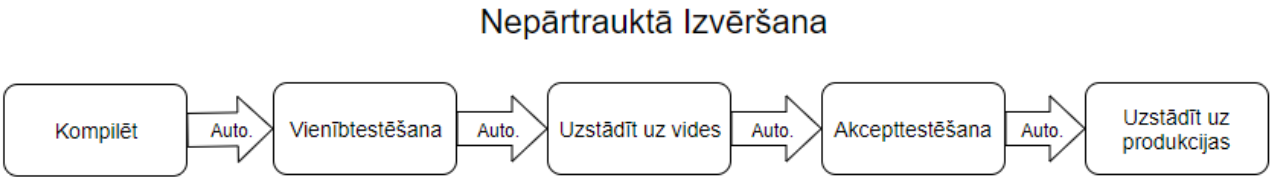
- Ir nepieciešams izstrādāt spēcīgus pamatus nepārtrauktajā integrācijā, jo automātiskajiem testiem ir jāspēj pārbaudīt lielu daļu no programmatūras koda.
- Programmatūras piegādei uz nākamajām vidēm jābūt automatizētai. Tās izsaukšana joprojām var būt manuāla, taču brīdī, kad tā ir uzsākta, tai ir jāspēj strādāt bez cilvēka iesaistes.
- Izstrādātājiem ļoti iespējams būs jāveido specializēta loģika, kas neļautu izmantot programmatūras funkcionalitāti, kas joprojām nav līdz galam pabeigta, lai neietekmētu klientus produkcijas vidē.

Nepārtrauktās piegādes ieguvumi:

- Izstrādātājiem nav nepieciešams patērēt lielu laika apjomu, gatavojoties programmatūras piegādei, jo tā ir automatizēta.
- Programmatūras piegādi ir iespējams veikt mazākās daļās un biežāk, tādējādi paātrinot klienta atsauksmju iegūšanu.
- Ir iespējams viegli un daudz ātrāk piegādāt mazas izmaiņas, rezultātā atvieglojot lēmumu pieņemšanu par šādām izmaiņām.

1.4. Nepārtrauktā izvēršana

Nepārtrauktā izvēršana ir nākamā automatizācijas pakāpe pēc nepārtrauktās piegādes. Izmantojot šo programmatūras piegādes metodi, izstrādātā programmatūra, kura ir veiksmīgi izpildījusi visus automātiskos testus visās izstrādes stadijās, tiek arī automātiski piegādāta uz produkcijas vides klientam. [9]



1.3. att. **Nepārtrauktās izvēršanas vizuāls piemērs**

Nepārtrauktās izvēršanas darbība notiek pilnīgi automātiski, bez cilvēka iesaistes līdz brīdim, kad kāds no izpildītajiem automātiskajiem testiem atgriež kļūdu, tajā brīdī šīs izmaiņas tiek atgrieztas iepriekšējā stāvoklī un tiek apturēta to piegāde klientam.

Nepārtrauktās izvēršanas ieguvumi galvenokārt ir saistīti ar piegādes ātruma palielināšanu, šādā veidā paātrinot arī klienta sniegtās atsauksmes par izstrādāto programmatūru. Šāda piegādes veida izmantošana arī samazina izstrādātāju stresu piegādes dienā un ļauj vairāk koncentrēties uz programmatūras izstrādāšanu, jo gala rezultāta piegāde notiek pilnīgi automātiski. [8]

Nepārtrauktās izvēršanas prasības:

- Automātisko testu kvalitātei ir jābūt ļoti augstā līmenī, jo to kvalitāte noteiks arī piegādātās programmatūras kvalitāti kopumā.
- Programmatūras dokumentācijas izstrādes un uzturēšanas procesam ir jānotiek vismaz vienādā ātrumā kā programmatūras piegādei.
- Programmatūras funkcionalitātes ierobežošana ir obligāti nepieciešama, lai izvairītos no nepabeigtas vai priekšlaicīgi piegādātas programmatūras funkcionalitātes darbības produkcijas vidē.

Nepārtrauktās izvēršanas ieguvumi:

- Programmatūras izstrāde notiek daudz straujāk, jo nav nepieciešams to apturēt, lai veiktu sagatavošanas darbus piegādei uz produkcijas vides. Programmatūras piegāde notiek pilnīgi automātiski pie katrām veiktajām izmaiņām.

- Programmatūras piegādes ir mazāk riskantas un vieglāk labojamas, jo piegāde notiek mazākās daļās, tādējādi ir daudz mazāka ietekme uz produkcijas vidi un vieglāk identificēt iespējamās problēmas.
- Klientam ir iespējams redzēt jaunu vai uzlabotu funkcionalitāti daudz biežāk, kā arī regulāri ir iespējams redzēt kvalitātes uzlabojumus.

1.5. Konfigurācija kā kods

Konfigurācija kā kods ir process, lai pārvaldītu programmatūras risinājuma konfigurācijas datus. Konfigurācija kā kods ir formāla konfigurācijas migrācija starp vidēm, kurā tiek izmantota versiju kontroles sistēma. [10]

Izmantojot konfigurāciju kā kodu, nepieciešamā konfigurācija tiek migrēta reizē ar piegādes procesā piegādāto programmatūras kodu. Gadījumā, kad ir nepieciešams veikt konfigurācijas izmaiņas, tās tiek uztvertas tieši tāpat kā koda izmaiņas. [11]

Konfigurācijas kā koda ieguvumi:

- Automatizācija un standartizācija – tā kā konfigurācija ir rakstīta koda formātā, ir iespējams izmantot visas labākās programmatūras izstrādes prakses, lai to optimizētu, piemēram, izveidot atkārtoti izmantojamus definīciju plānus, parametrus, izmantot cikliskas darbības, lai izveidotu dažādas konfigurācijas daļas.
- Versiju kontrole izmaiņām – ar tās palīdzību ir iespējams redzēt, kas un kad ir veicis izmaiņas attiecīgajā failā. Šādā veidā ir arī iespējams izolēt izmaiņas, kuras šobrīd ir izstrādes procesā, paralēli izmantojot esošās citās vidēs.
- Droša pāreja no testēšanas uz produkciju – izmantojot konfigurāciju kā kodu, ir iespējams veiktās izmaiņas no sākuma droši nogādāt uz testēšanas vidēm un pārbaudīt to darbību. Tieši tādā pašā veidā tās ir iespējams uzstādīt arī produkcijas vidē, rezultātā kontrolējot gala rezultātu.
- Vides sinhronizācija – izmantojot konfigurāciju kā kodu, ir iespējams kontrolēt to, lai uz visām izstrādes laikā izmantotajām vidēm atrastos vēlamā konfigurācija.

1.6. Infrastruktūra kā kods

Infrastruktūra kā kods ir infrastruktūras pārvaldības aprakstošs modelis, kuram izmanto tādu pašu versiju kontroli, kā DevOps komanda izmanto pirmkodam. Izmantojot šādu principu, ir iespējams izveidot identisku vidi katru reizi, kad šis kods tiek izmantots. Šāda prakse ir viena no galvenajām, kuru izmanto DevOps praksē, to apvienojot ar nepārtrauktu piegādi. [12]

Infrastruktūra kā kods atrisina problēmu, kad programmatūras izstrādes laikā kods tiek piegādāts uz dažādām vidēm, šādā veidā atrisinot iespējamās atšķirības starp tām. Bez šīs prakses izmantošanas, komandai ir jāuztur vienoti iestatījumi katrai videi, kurā tiek piegādāta programmatūra. Laika gaitā katra no vidēm izstrādes ciklā kļūst unikāla, tai eksistē unikāla konfigurācija, kuru nevar atkārtot automatizēti.

Šādā veidā atšķirīgas vides izstrādes laikā noved pie kļūdām programmatūras piegādes laikā. Šādas atšķirīgas vides ir grūti uzturēt un pārvaldīt, jo tās prasa daudz manuālu procesu, kuriem ir grūti sekot līdz un tie var novest pie kļūdām.

Infrastruktūras kā koda ieguvumi ir tādi, ka, pirms programmatūras uzstādīšanas uz mērķa vides, šī vide tiek konfigurēta un sakārtota vienādi, neatkarīgi no tā, kādā stāvoklī tā bija pirms tam. Šī iespēja sasniegta, izmantojot automātisku konfigurācijas uzstādīšanu uz mērķa vai jaunas vides izveidošanu no jauna. [12]

Izmantojot infrastruktūru kā kodu, komanda veic izmaiņas vides aprakstā un versiju kontroles sistēmas uzturētā konfigurācijas modelī, kas parasti ir labi dokumentēts koda formāts. Veicot programmatūras automatizētu piegādi, šis modelis tiek izpildīts uz mērķa vides. Izmaiņu nepieciešamības gadījumā, tās tiek veiktas pirmkodā, nevis mērķa vidē.

Infrastruktūra kā kods atļauj testēt izstrādāto programmatūru vidēs, kas ir vienādas ar produkcijas vidi agrākā programmatūras izstrādes cikla stadijā. DevOps komandai pēc pieprasījuma ir nepieciešams nodrošināt vairākas uzticamas testa vides. Infrastruktūru aprakstītu kā kodu ir arī iespējams pārbaudīt un testēt, lai novērtu bieži sastopamas piegādes problēmas. Kā arī gadījumā, ja tiek izmantoti mākoņpakalpojumi, tos ir iespējams izveidot, kontrolēt un nodzēst, izmantojot infrastruktūru kā kodu. [13] [12]

Komandas, kuras izmanto infrastruktūru kā kodu, var ātri piegādāt daudzas stabilas vides, šādā veidā izvairoties no manuālas vides konfigurēšanas, garantējot vienādu vēlamo konfigurāciju. Šāda prakse ļauj atbrīvoties no iespējamām infrastruktūras konfigurācijas atšķirībām un trūkstošām programmatūras atkarībām.

2. PROGRAMMATŪRAS IZOLĀCIJA

Programmatūras piegādes laikā ir nozīmīgi zināt, kas atradīsies uz fiziskās mērķa sistēmas, lai varētu nodrošināt piegādātās programmatūras darbību, neietekmējot citu procesu un produktu darbību. Piemēram, gadījumā, kad uz vienas fiziskās ierīces atrodas vairāk kā viens gala produkts, ir jāspēj nodrošināt, lai visi programmatūras produkti spētu pilnvērtīgi un paredzami veikt savu darbību.

2.1. Konteineri

Konteineru izveide darbojas abstraktā operētājsistēmas līmeni, kas sniedz iespēju individuālai, modulārai un strikti definētai programmatūras funkcionalitātei darboties neatkarīgi. Rezultātā vairāki šādā veidā izolēti konteineri spēj strādāt dinamiski, izmantojot vienus un tos pašus fiziskos sistēmas resursus. [14]

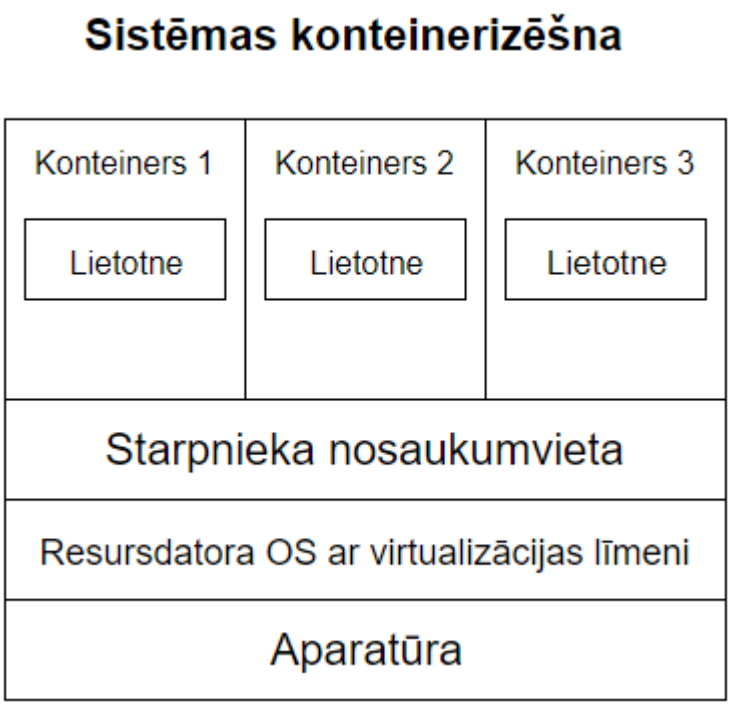
2.1.1. Vispārējais apraksts

Konteiners ir standartizēta vienība, kurā ir apvienota vēlamā programmatūra ar tās darbībai nepieciešamajām atkarībām, rezultātā lietotne ir ērti pārvietojama un efektīvi spēj strādāt dažādās sistēmās. Vispārīgā skaidrojumā, konteiners ir izolēta procesu grupa, kurai ir limitēta piekļuve resursdatora failu sistēmai un procesiem. [15]

Konteinera risinājumi nodrošina septiņu dažādu resursu izolāciju katrai tā instancei:

1. Cgroups: izolē saknes mapi.
2. IPC: izolē starpprocesu komunikāciju.
3. Tīkls: izolē tīkla segmentu.
4. Montēšanas punkts: izolē montēšanas punktu.
5. PID: izolē procesu identifikatorus.
6. Lietotāji: izolē lietotājus un lietotāju grupas.
7. UTS: izolē resursdatora nosaukumus un domēna vārdus.

Šādā veidā tiek nodrošināts tas, ka resursdatora līmenī konteiners ir process, bet konteinera līmenī tas darbojas kā atsevišķa sistēma. Tādējādi ir iespējams izolēt noteiktu lietotni ar visām tās darbībai nepieciešamajām atkarībām. [16]

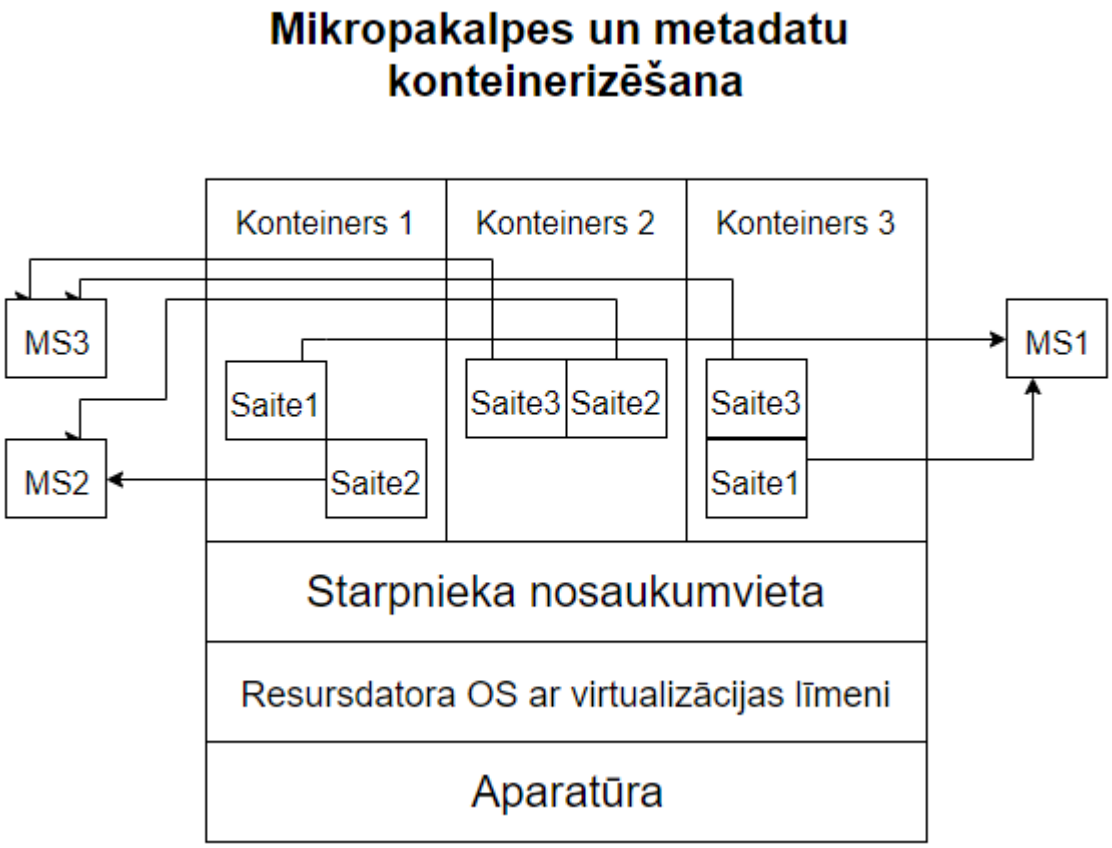


2.1. att. **Sistēmas konteinerizēšanas vizuāls piemērs**

Attēlā 2.1 ir iespējams redzēt to, ka ar konteineru palīdzību izstrādātājiem nav nepieciešams veidot programmatūru dažādās sistēmās, lai nodrošinātu darbībai nepieciešamos resursus. Pilnvērtīgu lietotnes daļu vai visu lietotni ir iespējams izpildīt tai specifiski izveidotā izolētā vidē, neietekmējot citas sistēmas daļas vai programmatūru, tā novēršot iespējamos konfliktus, kas būtu saistīti ar bibliotēkām vai lietotnēm.

Brīdī, kad lietotnei, kura ievietota konteinerī, ir nepieciešama specifiska funkcionalitāte vai bibliotēka, kura nav pieejama resursdatora operētājsistēmā, tā spēj nosūtīt vēlamo darbības pieprasījumu uz koda fragmentu vai bibliotēku, kura atrodas pašā konteinerā. [14]

Konteineru izmantošana programmatūras piegādē ir īpaši noderīga, izstrādājot, uzstādot un testējot modernas lietotnes un mikropakalpju procesus, kas spēj darboties izolētās izpildes vidēs, uz vienas fiziskās sistēmas.



2.2. att. **Mikropakalpes konteineru vizuāls piemērs**

Attēlā 2.2 redzams ka mikropakalpes izveido vienu funkcijas elementu, kas sniedz iespēju to izsaukt ārējam avotam. Šādi risinājumi ir noderīgi funkcionalitātei, kura ir nepieciešama vairākām sistēmām un izvietojama konteinera risinājumos, tādā veidā izvairoties no individuālas funkcionalitātes izstrādes katrā vēlamajā sistēmā. Ieguvumi, izmantojot šādu mikropakalpju konteineru tehnoloģiju, kura atrodas, piemēram, vienotā mākoņkrātuvē, ir galvenokārt dinamiska fizisko resursu sadalīšana, samazinot iespējamību sasniegt resursu limitus. [14] [15]

Praktiskais ieguvums no konteinera risinājumu izmantošanas ir tāds, ka šādā programmatūras darbības un izolācijas veidā ir iespējams nodrošināt vairāku lietotņu darbību uz viena resursdatora ar pseidoizolētas vides palīdzību, nodrošinot katrai lietotnei tās darbībai nepieciešamās atkarības, tādējādi efektīvāk izmantojot pieejamos resursus. Konteinera risinājumi darbojas ar to pašu operētājsistēmu kā resursdators. [14] [16]

2.1.2. Konteineru izmantošanas ieguvumi

- Samazināts resursu patēriņš – konteineru izmantošana prasa mazāku fiziskā servera resursu apjomu, jo, atšķirībā no virtuālās mašīnas, tā sevī neietver operētājsistēmu. [17]

- Atvieglota pārvietošana – programmatūra, kura tiek darbināta konteinerī, ir viegli pārvietojama uz dažādām operētājsistēmām un aparatūras platformām.
- Augstāka stabilitāte – DevOps komanda var būt pārliecināta, ka programmatūra, kura darbojas konteinerī, strādās identiski neatkarīgi no tā, kur tā tiks uzstādīta.
- Augstāka efektivitāte – konteineru izmantošana ļauj piegādāt programmatūru un tās uzlabojumus lielā ātrumā.
- Efektīvāka izstrāde – konteineru izmantošana pilnīgi atbalsta spējās programmatūras izstrādes metodes un DevOps praksi, rezultātā paātrinot programmatūras piegādi un testēšanu visā programmatūras piegādes ciklā.

2.1.3. Konteineru izmantošanas trūkumi

Operētājsistēmas dalīšana starp vairākiem konteineriem to agrīnā stadijā radīja problēmas. Kods, kura izpildei bija nepieciešams piešķirt resursdatora operētājsistēmas līmeņa tiesības, atver to potenciāliem riskiem un ļaunprātīgai izmantošanai, piešķirot tiesības un iespējas uzbrukumam uz šīs sistēmas esošajām lietotnēm, rezultātā sagraujot visas konteinera lietotnes, kas atradās uz resursdatora. [18]

Senākās konteineru platformas ievietoja virtuālajās mašīnās, lai novērstu šīs drošības problēmas, taču šāda rīcība savā ziņā iznīcina konteineru izmantošanas ieguvumus. Modernās konteineru vidēs šie drošības riski ir apzināti un vismaz daļēji novērsti. [18]

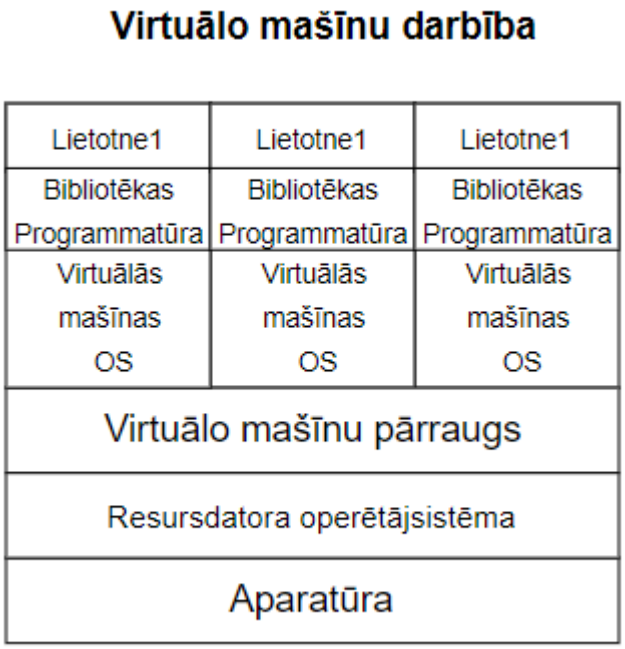
2.2. Virtuālās mašīnas

Virtuālās mašīnas var raksturot kā programmatūru, ar kuras funkcionalitātes sniegtajām iespējām tiek simulēta fiziska aparatūra vai sistēma.

2.2.1. Vispārējais apraksts

Virtuālā mašīna darbojas uz virtuālās mašīnas pārrauga, kas simulē fiziskās aparatūras funkcionalitāti ar programmatūras palīdzību. Virtuālās mašīnās ietilpst visi nepieciešamie elementi, lai varētu darbināt lietotnes, ieskaitot atmiņu, krātuvi, tīklošanu utt. Tajā arī var ietilpt lietotnei nepieciešamās bibliotēkas, tikmēr pati virtuālās mašīnas operētājsistēma vadīta un izpildīta, izmantojot virtuālās mašīnas pārraugu. [14]

Virtuālie aparatūras resursi ir savienoti kopā un piešķirti noteiktās lietotnes darbībai virtuālajā mašīnā. Šāds abstrakcijas līmenis ir izveidots, lai savstarpēji nošķirtu lietotni no fiziskās aparatūras un infrastruktūras.[16] Rezultātā fiziskā aparatūra var tikt mainīta, uzlabota vai paplašināta, neietekmējot virtuālajā mašīnā darbinātās lietotnes darbību un veiktspēju.



2.1. att. **Virtuālās mašīnas darbības vizuāls piemērs**

Attēlā 2.1 ir iespējams redzēt, ka virtuālā mašīna darbosies kā izolēta darbstacija, un ar šādas metodes palīdzību uz vienas fiziskās aparatūras ir iespējams nesaistīti darbināt pilnīgi izolētas virtuālās mašīnas ar dažādu slodzi un funkcionalitāti. Virtuālo mašīnu darbība galvenokārt ir prasīga pret fiziskās aparatūras resursiem, tādēļ tās neļauj vienai lietotnes funkcionalitātei darboties izolētā virtualizētā vidē, ja vienīgi gadījumā, kad cita virtuālā mašīna ir lietota cita modulāra lietotnes elementa izpildei. Ja lietotni ir nepieciešams pārvietot uz citu atrašanās vietu vai fizisko aparatūru, tad tā ir jāpārvieto kopā ar visu operētājsistēmu. [19]

2.2.2. Virtuālo mašīnu izmantošanas ieguvumi

Virtuālās mašīnas ļauj darbināt vairākas savstarpēji neatkarīgas vides, kuras spēj vienlaicīgi eksistēt, izmantojot vienu fizisko serveri. Šāda pieeja ļauj ievērojami palielināt fizisko serveru resursu lietojumu un rezultātā samazināt to skaitu, kas nepieciešams, lai veiktu noteiktu darba apjomu, samazinot sistēmas uzturēšanas izmaksas, elektrības patēriņu un fizisko serveru dzesēšanas prasības. [20]

Darba mobilitāte, ņemot vērā to, ka operētājsistēmas un lietotnes, kuras ir uzstādītas tieši uz fiziskā servera, ir piesaistītas šī servera specifiskajiem iestatījumiem un aparatūrai, ļauj izmantot abstraktu līmeni resursu dalīšanā neatkarīgi no fiziskajiem resursiem un aparatūras. Šādā veidā ir iespējams pārvietot virtuālās mašīnas no viena fiziskā servera uz citu, pārsvarā ar ļoti maz vai bez

traucējumiem pašā virtuālajā mašīnā. Vienīgās virtuālo mašīnu pārvietošanas prasības ir attiecināmas uz virtuālo mašīnu pārrauga saderības un adekvāta fiziskā mērķa servera resursu apjoma. [14][16][19]

Virtuālās mašīnas ļauj ātri dublēt to saturu. Tās savā būtībā ir kods un datu instance, kas darbojas piešķirtajā atmiņā. Tādēļ tās ir iespējams viegli dublēt kā diska failu, tajā ietverot pilnīgi visu – sākot no operētājsistēmas līdz datiem un lietotnēm. Ir iespējams veikt šo virtuālo mašīnu regulāru rezerves kopiju saglabāšanu, nodrošinot iespēju atjaunot to agrākā stāvoklī.

2.2.3. Virtuālo mašīnu izmantošanas trūkumi

Virtuālo mašīnu darbībai rezervētie fiziskie resursi pilnīgi tiek izmantoti reti, rezultātā atlikušie neizmantotie resursi nevar tikt izmantoti citiem procesiem, kas ir cieši saistīts ar nepieciešamo resursu plānošanu un sadalīšanu starp vairākām virtuālajām mašīnām. Šis noved pie neprecīzas plānošanas un lielas resursu izšķērdēšanas, lai gan virtualizācijas mērķis ir izstrādāt specifisku un optimizētu fizisko resursu izmantošanu un sadalīšanu datu centros.[16]

Virtuālās mašīnas ievieš jaunu risku uz veiktspējas ierobežojumu, it īpaši uz tīkla vai atmiņas daļām, kā arī tie patērē vairāk atmiņas, atšķirībā no procesu darbināšana bez virtualizācijas. Šādu risinājumu uzstādīšana nav iespējama publiskajās mākoņkrātuvēs, jo tai ir nepieciešama datortehnikas līmeņa virtualizācijas atbalsts, kurš nav pieejams šādās publiskajās mākoņkrātuvēs.[14][20]

2.3. Fiziskais serveris

Programmatūras uzstādīšanai uz serveriem bez virtualizācijas ir savi ieguvumi, lai arī šī pieeja ir smagnēja, to nav iemesla labot, ja tā nav problēma. [21]

2.3.2. Fizisku serveru izmantošanas ieguvumi

Programmatūras uzstādīšana uz fiziska servera sniedz augstāku uzticamību un drošību pret sistēmas līmeņa problēmām, ņemot vērā to, ka virtuāli serveri joprojām ir atkarīgi no fiziskās ierīces, uz kuras tie tiek darbināti. Ja fizisks serveris saskaras ar mehāniskām problēmām un tam nav veiktas darbības, kuras nodrošinātu augstāku sistēmas uzticamību, tad rezultātā var ciest vairāki virtuālie serveri, kas savukārt var novērst pie apjomīgām finansiālām sekām un ilgu sistēmas dīkstāves laiku.

Programmatūras uzstādīšana uz fiziska servera samazina veiktspējas kritumu, jo, izmantojot virtuālas sistēmas, tās joprojām ir saistītas ar fizisko resursu līmeni, tos dalot uz viena servera. Ņemot vērā to, ka aizvien vairāk virtuāli serveri ir izvietoti uz fiziskajiem serveriem, tiek izveidota

situācija, kad paaugstinās risks uz apjomīgām veiktspējas problēmām uz visām virtuālajām sistēmām, rezultātā negatīvi ietekmējot servisu orientētas vai reāla laika biznesa darbības. [21]

Daļā no lietotnēm nav atbalstītas dalītās vides, kas rezultātā var novest pie programmatūras licences noteikumu pārkāpuma. Kā arī atkarībā no izvēlētās operētājsistēmas ir nepieciešams pārliecināties, ka ir izvēlēta pareizā licence vēlamajai videi, kas var rezultēties ar lielām izmaksām atkarībā no tā, vai programmatūra ir uzstādīta uz fiziskas vides vai vairākiem virtuāliem serveriem.

Izmantojot fizisku serveru infrastruktūru, programmatūras piegādē tiek samazināta sarežģītība, jo šādu serveru uzturēšana ir laikietilpīga, taču nav kompleksa. Noteikta programmatūra var pieprasīt specifisku aparatūras jaudu tās optimālai darbībai, šādā gadījumā atbilstošāka ir fizisku serveru izmantošana.

2.3.3. Fizisku serveru izmantošanas trūkumi

Izmaksu samazinājums – ņemot vērā to, ka fizisku serveru resursi maksā naudu, tad ir izšķērdīgi tērēt lieki, uzturot lielāku serveru apjomu, kuru pilnvērtīgi neizmanto. Ar virtualizācijas palīdzību fiziskie serveru resursi tiek sadalīti vairākās virtuālajās vidēs, rezultātā efektīvāk izmantojot pieejamos resursus un samazinot fizisko serveru skaitu. Šādā veidā ir iespējams elastīgi pielāgoties biznesa vajadzībām ar mazākām vai bez papildus izmaksām par fiziskajiem serveru resursiem. [21]

Laika patēriņš – fizisko serveru uzturēšana un pārvaldība ir laikietilpīgs process, sagatavotas virtuālās vides, kuras ir iespējams pārvietot un startēt pēc nepieciešamības, ir daudz ātrāk sagatavojamas biznesa nepieciešamībai. Šāda pieeja samazina patērēto laiku uz programmatūras un konfigurācijas uzstādīšanu un pārlikšanu, tīkla konfigurāciju, rezerves kopiju sagatavošanu un visu pārējo, kas ir ietverts serveru līmenī. [21]

Atvieglota testēšana – strādājot elastīgā vidē, ir iespējams testēt jaunas konfigurācijas izmaiņas vai lietotnes efektīvāk, ņemot vērā to, ka testēšanas laikā nav nepieciešami papildus resursi. Ir iespējams testēt jaunu aparatūru un programmatūru, sniedzot vēlamo informāciju un uzlabojumus biznesam. Dalītu resursu gadījumā ir iespējams piešķirt specifisku virtuālo servera vidi testētājiem, lai tie varētu testēt, konfigurēt un pārkonfigurēt dažādus iespējamos scenārijus un beigās tos ērti atslēgt, šādā veidā samazinot pavadīto laiku un sarežģījumus testēšanas laikā. [22]

Ātrāka atjaunošana – gluži tāpat kā testētājiem, ir iespējams startēt serveri, kuru tie ir konfigurējuši vēlamajam mērķim, ir iespējams noteikt vēlamo programmatūru un tīkla iestatījumus iepriekš, tādējādi sistēmas, kuras tiks izmantotas tālākās izstrādes stadijās, ir iespējams veidot jau pēc pārbaudīta un vienlīdzīga procesa. Šīs vides ir iespējams darbināt paralēli, tādējādi nodrošinot

iespējamību uz virtuālo serveru vienlīdzību konfigurācijas un iestatījumu līmenī. Šādas vides ir iespējams sagatavot kā rezerves kopijas, gadījumā, ja galvenajā sistēmā notiek kritiska problēma, tādējādi izmantojot virtuālas vides šādiem scenārijiem risku novēršanai. [22]

3. ROBOTIZĒTU PROCESU AUTOMATIZĀCIJA

Robotizētu procesu automatizācija ir tehnoloģija, kura vēsturiski cēlusies no biznesa procesu automatizācijas tehnoloģijas. Šīs tehnoloģijas risinājumu mērķis ir, izmantojot programmatūras sniegtās iespējas, pilnīgi vai daļēji automatizēt un simulēt reālu cilvēku darbību noteikto biznesa mērķu un pienākumu izpildē. [23]

Šo risinājumu iespējas ļauj interpretēt un strādāt līdzīgi īstam darbiniekam, izmantojot komunikāciju arī starp vairākām sistēmām. Tādējādi ar robotizētu procesu automatizācijas risinājumu palīdzību ir iespējams izpildīt atkārtojamus un uz striktiem likumiem balstītus uzdevumus.

Galvenais robotizētu procesu automatizācijas ieguvums ir virtuāls darbinieks, kas nekad neguļ, nekļūdās un ilgtermiņā izmaksā mazāk kā reāls darbinieks, rezultātā biznesam samazinot ilgtermiņa izmaksas darbinieku skaita dēļ, kā arī novēršot risku uz cilvēciskām kļūdām.

Robotizētu procesu automatizācijas risinājumu tehniskā darbība ir gluži kā virtuālam robotam, katram no tiem ir nepieciešama sava virtuālā darbstacija. Izmantojot virtuālas perifērijas ierīces un ekrānu, šis robots ir spējīgs strādāt uz mērķa biznesa sistēmas. [24]

Viens no plusiem šādam risinājumam ir tas, ka virtuālo robotu skaitu ir salīdzinoši viegli un ērti mainīt, tādējādi piedāvājot mērogojamību atkarībā no biznesa prasībām un darba apjoma.

Negatīvā puse šādiem risinājumiem galvenokārt ir saistīta ar to, ka robotizētu procesu automatizācijas risinājumi izmanto grafisko lietotāja saskarni veidos, kuri, to izstrādājot, iespējams nemaz nebija paredzēti, tādējādi izveidojot papildus darbu un sarežģītību brīžos, kad tajā tiek veiktas izmaiņas. [25]

4. RPA PLATFORMAS UN TO TEHNISKĀS PRASĪBAS

Šajā nodaļā tiks apskatītas četras robotizētu procesu automatizācijas izstrādes platformas, precīzāk apskatot divus tirgū populārus maksas risinājums un divus atvērtā koda risinājums, galvenokārt pievēršot uzmanību šo izstrādes platformu sniegtajām iespējām, atbalstam, kā arī programmatūras un tehniskajām prasībām, kuras ir kritiskas gala risinājumu piegādē.

4.1. "UiPath" platforma

"UiPath" ir viena no industriju vadošajām un skaitliski populārākā robotizētu procesu automatizācijas risinājumu izstrādes platforma. Šī platforma spēj piedāvāt gan robotizētu procesu automatizāciju jeb robotus bez lietotāja iesaistes, gan robotizētu darbstaciju automatizāciju jeb robotus, kuriem nepieciešama lietotāju iesaiste, katram no šiem produktiem ir nepieciešamas atšķirīgas licences un to cenas atšķiras. [23] [26]

"UiPath" robotizētu procesu automatizācija spēj automatizēt visus atkārtojamos uzdevumus, kuri tiek izpildīti datorā, tai skaitā ietverot arī vecāku sistēmu integrācijas, datu ievadi, ekrāna apstrādi, testēšanu, migrāciju utt. [26]

"UiPath" platforma ir izstrādāta tā, lai to varētu izmantot dažādu izmēru biznesiem. Biznesiem, kuri to informācijas sistēmās izmanto lielu apjomu ar manuālu cilvēku darbu, "UiPath" piedāvā uzticamas tehnoloģijas, kas spēj sniegt inovatīvus risinājumus un paaugstināt efektivitāti un rezultātā arī peļņu.

Šī platforma palīdz biznesam izmantot robotus, lai efektīvi automatizētu atkārtojamus un manuālus darbus, kuri ir bāzēti uz definētiem noteikumiem, šādā veidā ļaujot biznesam integrēt robotus bez izmaiņām vecākās sistēmās un ar mazākām izmaksām kā cilvēku darbaspēka alternatīvas. "UiPath" roboti ir apmācāmi, un tie sistēmas grafisko vidi redz tieši tāpat kā cilvēki. Tie sniedz iespēju automatizēti atvieglot darbu un sadarboties ar darbiniekiem, nepārtraukti ziņojot par progresu. To darbība notiek uz darbinieku darbstacijām vai speciāli šim mērķim paredzētiem serveriem, tie spēj strādāt bez apstājas, šādā veidā uzlabojot produktivitāti visam procesam kopumā. [26]

"UiPath" platforma ir pieejama gan maksas, gan bezmaksas versijā. Bezmaksas versija ir pieejama studentiem un izstrādātājiem ar mērķi sniegt iespēju šo platformu apgūt. Maksas versija ir paredzēta uzņēmumiem tās izmantošanai, tās licencēšanas izmaksas ir atkarīgas no izvēlētajiem pakalpojumiem un robotizētu procesu automatizācijas robotu skaita. [26] Šīs platformas

bezmaksas versijas atbalsts ir galvenokārt izmantojot forumu un izstrādātāju kopienu komunikāciju, taču maksas versijai ir ražotāja sniegts atbalsts problēmu risināšanai.

Lai uz šīs platformas izstrādāti robotizētu procesu automatizācijas risinājumi spētu darboties, tiem ir nepieciešama specifiski programmatūras moduļi:

- "UiPath Orchestrator" – sinhronizācijas vadības modulis.
- "UiPath Robot" – izpildes modulis.

4.1.1. "UiPath Orchestrator" modulis

Tā ir tīmekļa lietotne, kas sniedz iespēju pārvaldīt un kontrolēt "UiPath Robot" robotu izveidi, pārraudzību, resursu izmantošanas kontroli mērķa videi. Tas darbojas kā integrācijas punkts ārējiem risinājumiem un lietotnēm. [27]

Galvenās vadības moduļa sniegtās iespējas:

- Pārvaldība – izveidot un uzturēt saikni starp robotiem un mērķa lietotni.
- Piegāde – nodrošina kontroli pār to, lai roboti izpildei izmantotu pareizo pakotnes versiju
- Konfigurācija – uztur un piegādā robota vides un procesu konfigurāciju
- Izpildes kontrole – pārvalda robotu izpildes secību
- Pārraudzība – kontrolē robotu identifikācijas datus un lietotāja tiesības
- Reģistrēšana – glabā un indeksē žurnāla ierakstus SQL datubāzē un/vai "ElasticSearch" (atkarībā no konfigurācijas un arhitektūras)
- Savstarpējā savienojamība – darbojas kā centralizēts komunikācijas punkts starp ārējiem risinājumiem vai lietotnēm.

Sinhronizācijas vadības moduļa darbība ir loģiski nodalāma trīs daļās:

- Izklāsta.
- Tīmekļa pakalpes.
- Patstāvīguma.

Izklāsta daļa atbild par vizuālo noformējumu servera platformai. Tās ietvarā lietotājs var veikt vairākas darbības grafiskā vidē: izveidot robotu grupas, piesaistīt noteiktas pakotnes vēlamajām grupām, analizēt žurnāla ierakstu informāciju par robotiem vai to grupām, kā arī iedarbināt vai apturēt robotu darbību.

Tīmekļa pakalpes modulī, izmantojot "REST API", spēj nodot tālāk vēlamo konfigurāciju, kontrolēt komunikāciju, žurnāla ierakstu saņemšanu, pakotnes piegādi un izpildes secības kontroli mērķa robotiem. [28]

Patstāvīguma modulis sastāv no SQL servera un indeksēšanas servera. Indeksēšanas servera mērķis ir glabāt un indeksēt žurnāla informāciju, kura ir saņemta no robotiem. SQL servera mērķis ir glabāt robotu un to grupu saistīto procesu, lietotāju un secības informāciju, kura ir pieejama izklāsta modulī, pārvaldīt izpildes secības ierakstus un glabāt žurnāla ierakstus, kuri ir saņemti no robotiem. [28]

"UiPath Orchestrator" moduļa tehniskās prasības:

4.1. tabula

"UiPath" platformas "UiPath Orchestrator" moduļa programmatūras prasības

	Servera prasības
Operētājsistēma	Windows Server 2008 R2 Windows Server 2012 R2 Windows Server 2016
Datubāze	SQL Server 2008 R2 SQL Server 2012 SQL Server 2014 SQL Server 2016 SQL Server 2017
Virtualizācijas platformas	VMWare Citrix XenDesktop – 7.6+ Oracle VirtualBox – 5.0+ Microsoft Hyper-V
Microsoft .NET satvars	Vismaz 4.6.1 versija
Powershell	4.0+ versija
WebDeploy	3.5+ 64bit versija
Internet Information Services Manager	7.5+ versija
URL Rewrite	2.0+ versija
ElasticSearch	2.3 – 6.x versija
Kibana	4.0 – 6.6.1 versija

Redis	3.0.504 versija (uz Windows) 3.0.7 – 4.0.11 (uz Linux)
Google Chrome	50+ versija
Microsoft Edge	20+ versija

[29]

"UiPath Orchestrator" moduļa tehniskā nodrošinājuma prasības:

4.2. tabula

"UiPath" platformas "UiPath Orchestrator" moduļa tehniskā nodrošinājuma prasības nepieciešamajam tīmekļa izklāsta lietotnes serverim atkarībā no robotu skaita

	<20	<50	<100	<200	<250
CPU (kodoli)	4	4	4	4	4
RAM (GB)	4	4	4	4	4
HDD (GB)	100	100	150	200	250

[30]

4.3. tabula

"UiPath" platformas "UiPath Orchestrator" moduļa tehniskā nodrošinājuma prasības nepieciešamajam SQL serverim atkarībā no robotu skaita

	<20	<50	<100	<150	<250
CPU (kodoli)	4	4	4	8	8
RAM (GB)	8	8	8	8	16
HDD (GB)	100	200	300	400 (SSD)	400 (SSD)

[30]

4.4. tabula

"UiPath" platformas "UiPath Orchestrator" moduļa tehniskā nodrošinājuma prasības nepieciešamajam "Elasticsearch" serverim atkarībā no robotu skaita

	<20	<50	<100	<200	<250
CPU (kodoli)	4	4	4	4	4
RAM (GB)	4	4	8	12	12
HDD (GB)	100	100	150	200	300

[30]

4.1.2. "UiPath Robot" modulis

Tas ir izpildes aģents, kurš ļauj darbināt šīs platformas izstrādātos risinājumus. Šo risinājumu ir iespējams uzstādīt divos veidos, kā servisu sistēmas līmenī vai lietotāja līmenī. Neatkarīgi no izvēlētā uzstādīšanas veida tā darbību var kontrolēt, izmantojot platformas piedāvāto sinhronizācijas vadības moduli. Robotiem ir nepieciešams savienojums ar platformas piedāvāto sinhronizācijas vadības moduli vai arī tiem ir jābūt licencētiem uz uzstādītās mērķa sistēmas individuāli. [37] [38]

Robots ir sadalīts vairākās daļās, kur katra daļa atbild par noteiktu uzdevumu automatizācijas procesa izpildē.[37][38] Šīs daļas ir:

- Pārvaldīts robota serviss – pārvalda un uzrauga servisa darbību, un strādā kā starpnieks starp izpildes sistēmu un sinhronizācijas vadības moduli. Pārvalda robota pierakstīšanās informāciju, darbina konsoles lietotni izpildes sistēmā.
- Lietotāja robota serviss – pārvalda un uzrauga servisa darbību, un strādā kā starpnieks starp izpildes sistēmu un sinhronizācijas vadības moduli. Pārvalda robota pierakstīšanās informāciju. Darbina lietotni automātiski izpildes sistēmas lietotāja daļā.
- Izpildītājs – izpilda vēlamo robota darbu esošajā sesijā. Seko līdzi monitora DPI iestatījumiem.
- Aģents – lietotne, kura ataino pieejamos robota darbus. Sniedz iespēju pieprasīt darbu uzsākšanu vai apturēšanu un izmainīt to iestatījumus. Darbojas kā servisa klients.
- Komandrinda – konsoles lietotne, kura var pieprasīt darbu uzsākšanu un gaidīt to rezultātu. Darbojas kā servisa klients.

Atkarībā no izvēlētās licences veida tiek noteiktas robota sniegtās iespējas:

- Uzraugāms – darbojas uz tās pašas darbstacijas, uz kuras strādā lietotājs, ar mērķi atvieglot lietotāja ikdienas darbus. Parasti darbība tiek uzsākta pēc specifiskas lietotāja darbības. Šādu robota tipu nav iespējams kontrolēt no sinhronizācijas vadības moduļa, jo tas nespēj darboties zem aizslēgta ekrāna.
- Bez uzraudzības – darbojas bez lietotāja iesaistes virtuālā vidē, šādā veidā automātiski darbinot jebkādu nepieciešamo procesu skaitu. Papildus uzraugāma robota iespējām šis robots spēj arī izpildīt darbības attālināti, uzraudzīt un kontrolēt izpildes secību un laiku.

"UiPath Robot" moduļa tehniskās prasības:

4.5. tabula

"UiPath" platformas "UiPath Robot" moduļa programmatūras prasības

	Servera prasības	Darbstacijas prasības
Operētājsistēma	Windows Server 2008 R2 Windows Server 2012 R2 Windows Server 2016	Windows 7 Windows 8.1 Windows 10
Microsoft .NET satvars	Vismaz 4.6.1 versija	Vismaz 4.6.1 versija

[39]

4.6. tabula

"UiPath" platformas "UiPath Robot" moduļa tehniskā nodrošinājuma prasības

	Minimālās prasības	Rekomendētās prasības
CPU	2x 1.8 Ghz 32-bit (x86)	4x 2.4Ghz 64-bit (x64)
RAM	4 GB	8GB

[40]

4.1.3. Risinājumu piegāde izmantojot "UiPath" platformu

Izstrādātos risinājumus ir iespējams publicēt jeb nosūtīt uz "UiPath Orchestrator", kurš darbojas, kā šīs platformas risinājumu sinhronizācijas vadības modulis. [28][38] Tas attiecīgi gala risinājumus piegādās tālāk robotiem izpildei un veiks šo robotu kontroli. Gadījumā, ja platformas piedāvātais sinhronizācijas modulis nav pieejams, piemēram, uzraugāma robota gadījumā, vēlamo risinājumu ir nepieciešams piegādāt tieši lietotāja segmentā uz mērķa darbstacijas. Šīs platformas

risinājumu izmantošanai ir plašas prasības uz nepieciešamo programmatūru un infrastruktūras resursiem. [28]

Ņemot vērā programmatūras izstrādes laikā izmantotās vides, ir iespējams infrastruktūras sagatavošanas procesu nodrošināt automatizēti, izmantojot "Infrastruktūra kā kods" pieeju, taču šādas automatizācijas sagatavošana būtu laikietilpīgs process, ņemot vērā iesaistīto vienību skaitu.

Tehniskā izstrādātā risinājuma piegāde neatkarīgi no tā, vai tiek izmantoti uzraugāmi vai bez uzraudzības roboti, ir salīdzinoši vienkārši automatizējama platforma sniegto iespēju dēļ, rezultātā pēc infrastruktūras izveides ir iespējams izveidot un izmantot DevOps praksē pielietoto nepārtrauktās izstrādes praksi.

4.2. "Automation Anywhere" platforma

"Automation Anywhere" ir elastīga, kognitīva un integrēta robotizētu procesu automatizācijas platforma, kura sniedz iespēju izmantot gudrus un mērogojamus robotus. Ar šo robotu palīdzību ir iespējams uzlabot uzņēmuma produktivitāti. [31]

Bizness var izveidot konfigurējamus robotus, kuriem pēc tam ir iespējams uzdot vēlamos darbus, un kontrolēt to darbību pēc ieskatiem. Tie darbojas kā digitāli izstrādāts darbaspēks, kuriem ir iespējams iemācīt vēlamās darbības, un pēc tam tie spēs to darīt automātiski. Šai robotizētu procesu automatizācijas izstrādes platformai lietotāju atbalsts ir nodrošināts no paša platformas ražotāja un sniegtās dokumentācijas.

Roboti, kuri ir izstrādāti, izmantojot šo platformu, var sazināties arī ar vairākām biznesa lietotnēm tāpat kā to darītu cilvēks. Šos robotus ir iespējams arī dublēt, tādējādi sniedzot iespēju tiem veikt vienu un to pašu uzdevumu dažādos biznesa procesos. [31]

Pilnīga automatizācija – "Automation Anywhere" piedāvā kompānijām izveidot gudrus robotus, kuri var veikt darbības dažādās biznesa daļās. Tie var tikt izveidoti un piegādāti gan gala lietotāja vidē un strādāt ar procesiem, kuri ir bāzēti uz biznesa noteikumiem, kā arī darboties serveru pusē kā pilnīgs robotizētu procesu automatizācijas risinājums.

Gudri roboti – izstrādātāji var izveidot automatizēšanas robotus, izmantojot "Automation Anywhere" platformu, kuriem ir iespējams iemācīt, kā rīkoties ar piešķirtajiem uzdevumiem, rezultātā nodrošinot to neatkarīgu darbību. Šādā veidā piedāvājot risinājumu, kurš pieprasītu minimālu lietotāja kontroli un pārraudzīšanu to darbībai, jo izstrādātāji var būt droši par to, ka roboti sekos tiem sniegtajiem izpildes soļiem, lai pabeigtu saņemto uzdevumu. [32]

Efektivitātes ieskats – "Automation Anywhere" piedāvā kompānijām novērtēt, kā to robotizētu procesu automatizācijas roboti palīdz biznesa procesos. Platformas piedāvātajā

programmatūrā ietilpst funkcionalitāte, kura sniedz analītisku informāciju un atskaites saistītas ar robotu veiktspēju un biznesa iespaidu. Šāda funkcionalitāte ļauj noteikt, kurās biznesa daļās ir iespējami uzlabojumi.

Vienkārša programmatūra – lietotājiem draudzīga programmatūra, kuru ir iespējams izmantot arī biznesa uzņēmumu darbiniekiem bez iepriekšējas IT apmācības. Tā sniedz iespēju būt produktīviem un uzstādīt robotus pēc nepieciešamības, ērti un atvieglojot IT nodaļas vai piegādātāja slodzi. [31]

Augsta līmeņa drošība – "Automation Anywhere" ir nodrošināta ar banku sektora drošības protokoliem, kas sniedz biznesam apstiprinājumu tam, ka robotu apstrādātā informācija ir droši glabāta. [32]

Lai uz šīs platformas izstrādāti robotizētu procesu automatizācijas risinājumi spētu darboties tiem ir nepieciešama specifiski programmatūras moduļi:

- "Enterprise Control Room" – robotu kontroles, pārvaldības un uzstādīšanas modulis.
- "Bot Runner" – robotu izpildes modulis.
- "Enterprise Client" – komunikācijas modulis.

4.2.1. "Enterprise Control Room" modulis

"Enterprise Control Room" modulis no "Automation Anywhere" platformas ir "Windows" operētājsistēmas serveru bāzēta pārvaldības platforma, kura sniedz vienotu skatu uz visu automatizācijas vidi un platformu, sniedzot centralizētu administrācijas saskarni. [33]

Ar šī moduļa palīdzību ir iespējams ieplānot, uzstādīt, izpildīt, pārvaldīt un kontrolēt organizācijas robotu darbības. Iebūvētā versiju kontroles funkcionalitāte nodrošina vairāku lietotāju sadarbību. [33] [34]

Moduļa administratoram ir iespējams definēt pielāgotas lomas, piešķirt tiesības, kuras darbojas visā moduļa apmērā, tajās ietverot moduļa objektus un funkcijas. Administratoram ir iespējams piešķirt tiesības arī lietotāju pārvaldībai un licencēšanai. Šis modulis piedāvā arī pierakstīšanās informācijas glabāšanu, robotu izpildes plāna sagatavošanu un kontroli, kā arī audita žurnāla ierakstu apkopošanu un glabāšanu. "Automation Anywhere" modulis piedāvā arī vairāku pakāpju autentifikāciju, piedāvājot dažādas, industrijā populāras metodes, kā arī pašā modulī iekšēji kontrolētu un glabātu pieslēgšanās informācijas apstrādi. [33]

"Enterprise Control Room" moduļa sniegtās iespējas:

- Apgrieztais starpnieks – tas nodrošina klienta uzsāktu saziņu ar vairāk kā vienu mērķa serveri.

- Licencēšanas kontrole.
- "Control room" servisi.
- "Control Room" saziņa starp šo platformas moduli un SQL vai Oracle datubāzes serveri.
- "Control room" kešdarbe ar "Subversion" versiju pārvaldības serveri.
- "Elasticsearch" iekšējā darbība un saziņa ar ārēju "PostgreSQL" datubāzi lietotnes metadatu glabāšanai.

"Enterprise Control Room" moduļa tehniskās prasības:

4.7. tabula

"Automation Anywhere" platformas "Enterprise Control Room" moduļa programatūras prasības

	Servera prasības
Operētājsistēma	Windows Server 2016 Windows Server 2012 R2
Microsoft .NET satvars	Vismaz 4.6 versija
Datubāze	Microsoft SQL Server 2017 Microsoft SQL Server 2016 Microsoft SQL Server 2014 SP1 Microsoft SQL Server 2012 Oracle Database 12.1.0.2 Azure PostgreSQL 9.5.14 (SSL ieslēgts)
Interneta pārlūkprogrammas	Google Chrome versija 57 vai jaunāks Internet Explorer versija 10 vai 11 Firefox versija 52 vai jaunāka
Versiju kontroles integrācija (ja ieslēgta)	Subversion 1.9.7 (Visual SVN Server 3.6.x) Subversion 1.8.15 (Visual SVN Server 3.3.x) Subversion 1.7.2 (Visual SVN Server 2.5.2)

[35]

4.8. tabula

“Automation Anywhere” platformas “Enterprise Control Room” moduļa tehniskā nodrošinājuma prasības

	Minimālās prasības	Rekomendētās prasības
CPU	8 kodoli	8 kodoli
RAM	16 GB	32 GB
HDD	500 GB	500 GB

[35]

4.2.2. “Bot Runner” modulis

Šis modulis, kuru sniedz “Automation Anywhere” platforma, fiziski darbina izstrādātos robotizētu procesu automatizācijas risinājumus. Ir iespējams darbināt vairākus šādus moduļus uz vienas fiziskās sistēmas paralēli, šai fiziskajai sistēmai tikai nepieciešama licence katram darbinātajam “Bot Runner” modulim. Šis modulis sniedz žurnāla informāciju un statusa ziņojumu “Enterprise Control Room” modulim. [36]

Tas piedāvā noteiktas lietotāja saskarnes un fona servisus, kuri darbojas bez lietotāja iesaistes. [36]

Lietotāja saskarnes:

- Notikumu pārraugs
- Atskaites
- Robotu izpildītājs

Fona servisi:

- Automātiska identificēšanās
- “Enterprise Control Room” komunikācija
- Vietējs plānotājs robotu darbībai

4.2.3. “Enterprise Client” modulis

Izmantojot šo Automation Anywhere moduli, “Control Room” modulis spēj komunicēt ar “Bot Runner” moduli . Šis robotizētu procesu automatizācijas platformas modulis ir paredzēts un pielāgots biznesa lietotājiem, taču ietver arī sarežģītāku funkcionalitāti izstrādātājiem un administratoriem. [41]

Šī moduļa funkcionalitāte ietver:

- Robotu uzdevumu vai procesu veidošanu, ierakstīšanu, izpildīšanu un modificēšanu.
- Automatizētu uzdevumu izveidi.
- Darbībā esošu automatizētu uzdevumu pagaidu vai pilnīgu apturēšanu.
- Robotu darba plāna izveidi.
- Iespēju redzēt sistēmas žurnāla ierakstus.
- Komunikāciju ar "Control Room" moduli.

"Enterprise Client" moduļa tehniskās prasības:

4.9. tabula

"Automation Anywhere" platformas "Enterprise Client" moduļa programmatūras prasības

	Servera prasības	Darbstacijas prasības
Operētājsistēma	Windows Server 2016 Windows Server 2012 R2 Windows Server 2012 Windows Server 2008 R2	Windows 10 Windows 8.1/8 Windows 7 SP1
Microsoft .NET satvars	4.6 vai 4.6.1 versija	4.6 vai 4.6.1 versija
Java	Java SE Runtime 1.8.0 Java SE Runtime 1.7.0 Java SE Runtime 1.6.0	Java SE Runtime 1.8.0 Java SE Runtime 1.7.0 Java SE Runtime 1.6.0
Microsoft Silverlight	5.1.x versija	5.1.x versija
Adobe Flex	24 versija	24 versija
Microsoft MODI	12.0 versija	12.0 versija
Transym TOCR	5.0 versija	5.0 versija
Interneta pārlūkprogrammas	Google Chrome 49+ versija Internet Explorer 10 vai 11	Google Chrome 49+ versija Internet Explorer 10 vai 11 Microsoft Edge (tikai uz Windows 10)

[42]

4.10. tabula

"Automation Anywhere" platformas "Enterprise Client" moduļa tehniskā nodrošinājuma prasības

	Minimālās prasības	Rekomendētās prasības
CPU	4 kodoli (1.2-1.5 GHz)	4 vai vairāk kodoli (3.5 GHz)
RAM	6GB	8GB
HDD	300 MB (instalācijai)	300 MB (instalācijai) kā arī 30-50 GB (Ilgtermiņa darbībai)

[42]

4.2.3. Risinājumu piegāde, izmantojot "Automation Anywhere" platformu

Izstrādāto risinājumu piegāde, izmantojot "Automation Anywhere" platformu, notiek divās daļās, kur viena no tām ietver infrastruktūras prasības un atkarības un otra pašu robotizētu procesu automatizācijas robotu risinājumu. [43]

Pirmā daļa ietver nepieciešamās infrastruktūras izveidi un kontroli visā izstrādes posmā. Ņemot vērā to, ka ir nepieciešams liels skaits ar atkarībām platformas risinājumu darbības nodrošināšanai, šis process sākotnēji ir sarežģīts. Šādā situācijā ir iespējams visas nepieciešamās infrastruktūras prasības piegādāt ar DevOps praksē pielietoto "Infrastruktūra kā kods" praksi, nodrošinot identisku darbības vidi visos izstrādes posmos. [43] [44]

Otrā daļa sevī ietver izstrādāto robotizētu procesu automatizācijas robotu piegādi cauri visam izstrādes posmam, kura, atšķirībā no infrastruktūras, ir daudz vienkāršāka platformas sniegto iespēju dēļ. [43] [44] Ņemot vērā to, ka roboti tiek kontrolēti un piegādāti tālāk caur "Control Room" moduli, ir nepieciešams tos apvienot vienā kopumā un piegādāt nākamajam izstrādes posmam. Gadījumā, ja tiek izmantoti roboti, kuru darbība notiek gala lietotāju sistēmās, tie ir jāpiegādā līdz tām, jo "Control Room" modulis nespēj tos pārvaldīt.

4.3. "RobotFramework" platforma

"RobotFramework" ir vispārējs atvērtā pirmkoda automatizācijas ietvars, tas ir paredzēts akcepttestēšanai, akcepttestu bāzētai izstrādei un robotizētu procesu automatizācijai.

Ņemot vērā to, ka šī platforma ir atvērta un ērti paplašināma, to ir iespējams integrēt ar jebkuru rīku, lai spētu piegādāt un izveidot elastīgus un spēcīgus robotizētu procesu automatizācijas risinājumus. [45]

Šī ietvara pamati ir vispārīga automatizācijas platforma, kura ir populāra testēšanas automatizācijā un tiek izmantota industriju vadošajās programmatūras izstrādes kompānijās. Ņemot vērā tās popularitāti, tā ir aktīvi atbalstīta un nepārtraukti uzlabota. Šīs platformas atbalsts kopumā ir balstīts uz plašo kopienu kura to izmanto, taču detalizētu jautājumu vai specifisku problēmu gadījumā ir nepieciešams vērsties pie risinājuma izstrādātājiem.

Šīs platformas risinājumu paplašināšana ir iespējama dēļ tās modulārās arhitektūras, kura bāzējas uz paplašinājumu bibliotēkām. Tādēļ, ka šī platforma ir atvērtā pirmkoda, to ir iespējams izmantot bez licencēšanas izmaksām un ir iespējams darbināt jebkādu robotu skaitu paralēli. Vairāku robotu pārvaldības funkcionalitāti lielāka apjoma projektos ir iespējams izveidot, izmantojot piedāvāto "Jenkins" spraudni. [46]

"RobotFramework" robotizētu procesu automatizācijas risinājumus ir iespējams izstrādāt un paplašināt brīvi un pielāgoti klienta vēlmēm. Tos ir iespējams darbināt ikvienā mērķa sistēmā un cik bieži nepieciešams.

4.3.1. "RobotFramework" platformas bibliotēkas

"RobotFramework" robotizētu procesu automatizācijas risinājumu darbība un funkcionalitāte ir tieši atkarīga no izstrādes laikā izmantotajām paplašinājumu bibliotēkām un rīkiem. Šīs bibliotēkas iedalās divās daļās – standarta jeb iekļautas pamata pakotnē un ir izstrādātas reizē ar pašu satvaru un ārējās, kuras ir izstrādātas specifikas funkcionalitātes nodrošināšanai, ko nespēj sniegt pamata bibliotēkas. [47]

4.11. tabula

"RobotFramework" platformas pamata bibliotēkas

Bibliotēka	Sniegtā funkcionalitāte
Builtin	Sniedz vispārīgu atslēgas vārdu kopumu risinājuma darbībai
OperatingSystem	Sniedz iespēju darboties dažādās opērētājsistēmās un izpildīt tajās vēlamos uzdevumus
String	Bibliotēka, lai ģenerētu, modificētu un pārbaudītu virknes mainīgos
Process	Bibliotēka, kura nodrošina procesu darbību mērķa sistēmā
Dialogs	Nodrošina iespēja apturēt robota darbību izpildes laikā, lai saņemtu lietotāja ievadītu informāciju

Remote	Specifiska bibliotēka, kas nodrošina komunikāciju starp "RobotFramework" un citām bibliotēkām. Citas bibliotēkas var strādāt fiziski citā sistēmā, taču tām ir jāatbalsta XML-RPC protokols.
Telnet	Sniedz funkcionalitāti pieslēgties Telnet serverim un izpildīt tajā komandas
DateTime	Šī bibliotēka nodrošina datuma un laika pārveidošanu vēlamajā formātā
Collections	Nodrošina atslēgas vārdus, lai strādātu ar Python sarakstiem un vārdnīcām
Screenshot	Nodrošina atslēgas vārdus, lai veiktu sistēmas ekrānšāviņus
XML	Bibliotēka, ar kuras palīdzību var ģenerēt, modificēt un pārbaudīt XML failus.

[47]

4.12. tabula

"RobotFramework" platformas ārējās bibliotēkas

Bibliotēka	Sniegtā funkcionalitāte
Achive Library	Bibliotēka, kas nodrošina darbības ar zip un tar arhīviem
Database Library (Java)	Bibliotēkas, kas ir bāzēta uz Java, lai nodrošinātu darbības saistībā ar datubāzi. Šo bibliotēku var izmantot ar Jython
Database Library (Python)	Bibliotēka, kas nodrošina darbības ar datubāzi un darbojas ar jebkuru Python interpretātoru ieskaitot Jython.
Diff Library	Nodrošina salīdzināšanu starp diviem faileim
HTTP Bibliotēka (Requests)	Nodrošina HTTP līmeņa darbības, izmantojot iekšējus pieprasījumus
HttpRequestLibrary (Java)	Bibliotēka, kura nodrošina darbības ar Apache HTTP klientu
ImageHorizonLibrary	Starp platformu bibliotēka bāzēta uz Python, kura nodrošina grafiskās lietotāja saskarnes automatizāciju, izmantojot attēlu atpazīšanu
WhiteLibrary	Bibliotēka, kura nodrošina Windows grafiskās lietotāja saskarnes automatizāciju, atbalstot Win32, WinForms un WPF tehnoloģijas

FTP library	Bibliotēka, kura nodrošina FTP serveru izmantošanu
RemoteSwingLibrary	Nodrošina iespēju pieslēgties Java procesiem, izmantojot SwingBibliotēku, piemēram, Java Web Start lietotnēm.
SSHLibrary	Sniedz iespēju izpildīt komandas uz attālinātas mašīnas, izmantojot SSH pieslēgumu, sniedz iespēju arī izmantot SFTP failu pārvietošanu.
MongoDB library	Nodrošina saziņu ar MongoDB, izmantojot PyMongo
RESTinstance	Nodrošina iespēju izmantot HTTP JSON APIs
SeleniumLibrary	Nodrošina pārlūka darbības, izmantojot Selenium rīkus iekšēji
SudsLibrary	Bibliotēka, kura nodrošina darbības uz SOAP bāzētiem servisiem.
TFTPLibrary	Bibliotēka, kura nodrošina saziņu, izmantojot triviālo failu pārsūtīšanas protokolu.

[47]

4.3.2. Risinājumu piegāde izmantojot "RobotFramework" platformu

Izstrādāto risinājumu piegāde, izmantojot "RobotFramework" platformu, ir specifiska, ņemot vērā to, ka šī platforma darbojas līdzīgi lietotnei un nodrošina pārvaldību un sinhronizāciju, izmantojot ārēju rīku spraudņus.

Šīs platformas risinājumu piegāde ietvertu nepieciešamās atkarības attiecīgajam risinājumam un izstrādāto robotu kodu, to ir iespējams piegādāt, izmantojot DevOps praksē populāro programmatūras izolācijas veidu konteineri, tā garantējot "RobotFramework" risinājumu darbību visās izstrādes stadijās un atvieglojot versiju kontroli un konfigurāciju. Šajā gadījumā ir jāņem vērā konteineru risinājumu ierobežojumi un darbības princips.

4.4. "Kantu" platforma

"Kantu" ir robotizētu procesu automatizācijas platforma, kura sniedz iespēju izmantot gan tīmekļa automatizāciju, gan darbstacijas automatizāciju. Izmantojot šīs platformas "UI Vision" funkcionalitāti, ir iespējams izstrādāt risinājumus, kuri strādā pietuvināti reālam cilvēkam. [49]

Šīs atvērtā pirmkoda platformas risinājumu izmantošana izceļas ar augsto drošību. Lielākā daļa no platformas sniegtajām iespējām tiek izpildīta uz fiziskās sistēmas, kura to darbina, taču

funkcionalitāte, kas balstās uz ārēju pakalpojumu, ir pieejama tikai tad, ja lietotājs to specifiski pieprasa. Šī platforma ir bāzēta uz atvērtā pirmkoda pamatu, taču tai ir pieejama arī biznesam paredzēta maksas versija, kas sevī ietver klientu atbalstu un pilnīgi lokālu darbību mērķa sistēmā. Bezmaksas versijai ir pieejams atbalsts tikai no publiskā foruma. [48] [50]

4.4.1. "UI Vision" modulis

"Kantu" robotizētu procesu automatizācijas platforma darbojas, izmantojot attēla un teksta atpazīšanu. Izmantojot šīs tehnoloģijas, tās vēlāk procesu izpildē nolasa pieejamo informāciju ekrānā un meklē procesa solī definēto informāciju, tādējādi veicot automatizēto darbu.

Šīs platformas risinājumi sevī ietver "Selenium IDE" svarīgākās komandas, papildus sniedzot iespēju rakstīt un lasīt CSV failus, veikt vizuālu salīdzināšanu, failu lejupielādes automatizēšanu un PDF failu testēšanu. Šī moduļa ietvarā ir iekļauts arī API komandrindas paplašinājums, kurš ļauj to integrēt citos eksistējošos procesos un darboties dažādās programmēšanas valodās. [51]

Šī platformas moduļa funkcionalitāti ir iespējams paplašināt, izmantojot piedāvātās ražotāja paplašinājuma lietotnes. [51]

- "RealUser Simulation XModule" – šī lietotne paplašina platformas sniegtās iespējas, ļaujot simulēt peles kursora piespiešanu, objektu pārvietošanu un informācijas ievadīšanu, izmantojot klaviatūru.
- "File Access XModule" – šī lietotne paplašina platformas sniegtās iespējas, sniedzot tiešu pieeju failiem, rezultātā failu lasīšana un rakstīšana ir veicama tieši mērķa sistēmā.

4.4.2. Risinājumu piegāde, izmantojot "Kantu" platformu

"Kantu" robotizētu procesu automatizācijas risinājumu piegāde ir atkarīga no pielāgotā risinājuma un automatizēto procesu izpildītāju skaita. Ņemot vērā platformas izpildes specifiku, kura notiek bez sinhronizācijas vadības, visi procesi darbojas gala lietotāju sistēmās. [51]

Šīs platformas risinājumu piegādei ir nepieciešamas divas daļas, kur pirmā daļa ir nepieciešamo atkarību piegādāšana uz mērķa sistēmas, bet otrā – izstrādāto robotizētu procesu automatizācijas risinājumu novietošana uz mērķa sistēmas. [48] [50] [51]

Ņemot vērā to, ka šīs platformas risinājumiem nav nepieciešama specifiska infrastruktūra ārpus bibliotēkām un moduļiem, tās darbību ir iespējams piegādāt, izmantojot konteineru risinājumu, atvieglojot atkārtotas piegādes procesu un piegādi uz vairākām darbstacijām, garantējot vienotu konfigurāciju un izstrādāto risinājumu versijas.

4.5. Robotizētu procesu automatizācijas izstrādes platformu salīdzinājums

Bakalaura darba izstrādes laikā izpētītās maksas un bezmaksas robotizētu procesu automatizācijas izstrādes platformas atšķiras savā starpā. Salīdzinot abas maksas platformas savā starpā, ir iespējams novērot, ka šīm platformām ir salīdzinoši līdzīgas programmatūras un tehniskās prasības pret sistēmas resursiem, abas platformas spēj sniegt detalizētu informāciju par nepieciešamajiem serveru resursiem un paredzētu noslodzi.

Šīs platformas savstarpēji atšķiras ar piedāvāto iespēju klāstu, taču risinājumu piegādes metodes, ārpus infrastruktūras piegādes, ir ļoti līdzīgas. Taču abu platformu risinājumus var ērti un strauji piegādāt, bet izstrādes laikā ir iespējams izmantot manuālu infrastruktūras izveidi ar visām no tā izrietošām sekām, vai arī šo procesu automatizēt, izmantojot "Infrastruktūru kā kods" pieeju.

Bezmaksas robotizētu procesu automatizācijas izstrādes platformas ir īpaši mazā skaitā, galvenokārt šīs platformas ir agrīnās izstrādes stadijās vai to atbalsts ir zudis pavisam. Abas izpētītās bezmaksas platformas atšķiras no apskatītajām maksas platformām, jo nespēj skaidri paredzēt nepieciešamo sistēmas resursu apjomu.

"RobotFramework" platforma spēj sniegt salīdzinoši lielāku funkcionalitātes klāstu un šī klāsta vieglu paplašināšanu, izmantojot ārējās bibliotēkas. Šīs platformas risinājumus var izvietot gan uz serveriem, gan darbstacijām. To piegāde ir iespējama vienkāršoti, izmantojot konteineru programmatūras izolācijas sniegtās iespējas, tādējādi nodrošinot risinājumu darbību.

"Kantu" izstrādes platforma izceļas ar teksta atpazīšanas funkcionalitāti, kas ir pieejama arī apskatītajās maksas platformās, taču šī platforma nespēj nodrošināt darbību serveru vidē.

5. PRAKTISKAIS RISINĀJUMS

Sākotnēji, pirms vēlamā risinājuma izstrādes, klients apkopoja un precizēja savas vēlamā robotizētu procesu automatizācijas risinājuma darbības specifiku, vēlmes un ierobežojumus.

Klienta definētās infrastruktūras un risinājuma prasības:

- Robotizētu procesu automatizācijas platforma: "RobotFramework".
- Darbības nodrošināšana uz servera un lietotāja darbstacijas:
  - Servera operētājsistēma: "Oracle Linux 7".
  - Darbstacijas operētājsistēma: "Windows 10 Enterprise".
- Tīmekļa ierobežojumi: Atsevišķi nodalīti tīmekļa segmenti lietotāju un serveru pusē, lietotāju segmentam ir piekļuve tikai mērķa lietotnei.
- Darbības nodrošināšanai jāizmanto tikai eksistējošā infrastruktūra un sistēmas resursi.
- Robotizētu procesu automatizācijas risinājumam jāspēj strādāt ar biznesa lietotnes daļām:
  - "Oracle 12c R1" datubāzi.
  - "Oracle Weblogic Server 10.3.0.6" darbinātu lietotni un integrācijām.
- Robotizētu procesu automatizācijas risinājuma piegādei jāizmanto esošais piegādes process.

Izstrādātāju komanda robotizētu procesu automatizācijas risinājuma izveidē papildināja klienta sniegto ierobežojumu specifiku un dokumentēja risinājuma darbībai nepieciešamās papildus bibliotēkas un sistēmas atkarības.

Izstrādes laikā apzinātās un precizētās papildus prasības infrastruktūrai un risinājuma darbībai:

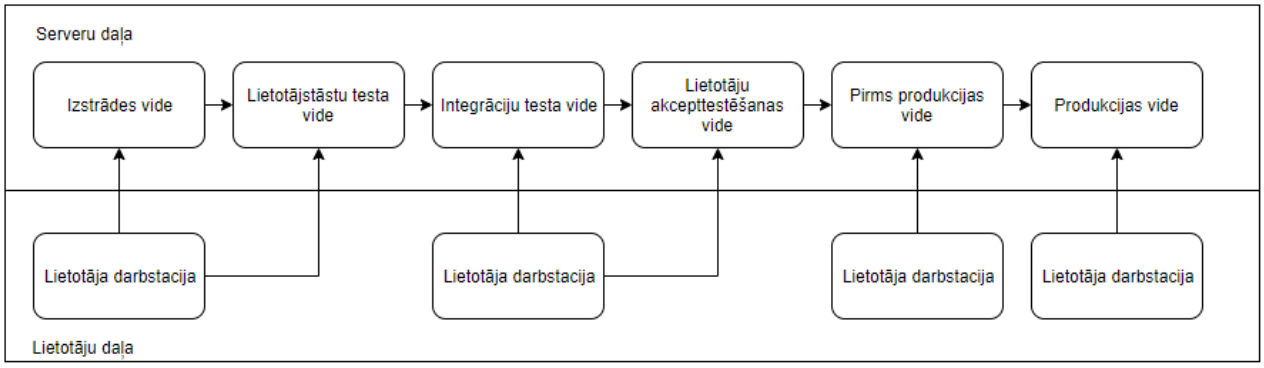
- Darbības nodrošināšana uz serveriem un lietotāja darbstacijām:
  - Serveru operētājsistēmas: "CentOS", "Oracle Linux 7".
  - Darbstacijas operētājsistēmas: "Windows 10 Enterprise", "Fedora Linux".
- Risinājuma darbībai nepieciešamās bibliotēkas:
  - RobotFramework – robotizēta procesu automatizācijas risinājuma pamata darbībai
  - Pabot – risinājuma paralēlas izpildes nodrošināšanai.
  - Database Library – darbību veikšanai datubāzē.
  - Selenium Library – paplašinātas risinājuma funkcionalitātes nodrošināšanai.
  - cx_Oracle – "Oracle" datubāzes pieslēguma izveidei, izmantojot "Oracle SQL plus".
- Risinājuma darbībai nepieciešamās sistēmas atkarības:
  - Chrome Driver – "Google Chrome" interneta pārlūkprogrammas darbības nodrošināšanai.

- Firefox – “Mozilla Firefox” interneta pārlūkprogrammas darbības nodrošināšanai.
- Python pip – nepieciešamo “Python” bibliotēku ērtai iegūšanai.
- “Xauth” – autorizācijas nodrošināšanai.
- “Xvfb” – darbstacijas grafiskās lietotāja saskarnes simulēšanai darbojoties bez grafiskās saskarnes vidē.
- “Oracle SQL plus” – “cx_Oracle” bibliotēkas darbības nodrošināšanai.
- “Gecko Driver” – “Selenium Library” bibliotēkas darbības nodrošināšanai.

5.1. Programmatūras piegādes process

Robotizētu procesu automatizācijas risinājuma piegādei ir jāizmanto jau eksistējošais piegādes process, tādējādi jāizmanto arī esošais piegādes cikls, modelis un vides. Esošais piegādes cikls ir nodalīts divās daļās – serveru daļā un lietotāju darbstaciju daļā.

Kopumā visas vides ir nodalītas trīs dažādos tīklos, kuriem nav iespējams savienojums savā starpā. Attēlā 5.1 redzamā esošās infrastruktūras darbība, programmatūras testēšana un piegāde ir nodalīta dažādiem piegādātājiem un vairākās vidēs.



5.1.att. **Esošā piegādes procesa infrastruktūras vizuāls atainojums**

Katram piegādātājam ir atšķirīga atbildība, pieejamība vidēm un pienākumi:

Piegādātājs X:

- Izstrāde - programmatūras izstrāde, robotizētu procesu automatizācijas izstrāde, automatizēto testu izstrāde.
- Testēšana - vienībtestēšana, lietotājstāstu testēšana un integrācijas testēšana.
- Piegāde - risinājumu piegāde no izstrādes vides līdz integrācijas testa videi. Risinājumu piegādes atbalsts bez tiešas piekļuves no lietotāju akcepttestēšanas vides līdz produkcijas videi.

- Infrastruktūra - izstrādes vides un lietotājstāstu testa vides uzstādīšana un uzturēšana. Atbalsts un rekomendācijas infrastruktūras uzturēšanai no integrācijas testa vides līdz lietotāju akcepttestēšanas videi, bez tiešas pieejas mērķa vidēm.
- Pieejamība - lietotāja darbstaciju daļas un serveru daļas pieejamība no izstrādes vides līdz lietotāju akcepttestēšanas videi.

Piegādātājs Y:

- Izstrāde - automatizēto testu izstrāde.
- Testēšana - integrācijas testēšana, lietotāju akcepttestēšana, pirms produkcijas vides pārbaude .
- Pieejamība - lietotāju darbstacijas pieejamība no integrācijas testa vides līdz pirms produkcijas videi.

Piegādātājs Z:

- Piegāde - risinājumu piegāde no lietotāju akcepttestēšanas līdz produkcijas videi.
- Infrastruktūra - vides sagatavošana un uzturēšana no integrācijas testa vides līdz produkcijas videi.
- Pieejamība - serveru daļas pieejamība no integrācijas testu vides līdz produkcijas videi.

Klients:

- Pieejamība - serveru daļas pieejamība pirms produkcijas un produkcijas vidēs. Lietotāju darbstacijas pieejamība no integrācijas testa vides līdz produkcijas videi.

5.2. Risinājuma piegāde

Izstrādātā risinājuma piegāde tika nodalīta divās daļās atkarībā no uzstādīšanas mērķa sistēmas:

- Piegāde risinājuma darbībai uz serveriem, izmantojot konteineri.
- Piegādes risinājums darbībai uz darbstacijas, izmantojot virtuālo mašīnu ar konteineri.

Piegādes risinājuma nodalīšana atkarībā no mērķa sistēmas tika izvēlēta, jo risinājumam ir jāspēj strādāt gan uz serveriem, gan uz biznesa lietotāja darbstacijas. Šāda prasība piegādes risinājuma izveidi sarežģī dažādo operētājsistēmu un esošās infrastruktūras limitu dēļ.

“RobotFramework” risinājuma darbības nodrošināšanai tika izveidots konteineris, ar kura sniegto funkcionalitāti un īpašībām ir iespējams izolēt risinājuma darbībai nepieciešamās bibliotēkas un sistēmas atkarības, novēršot risku uz citu risinājumu vai programmatūras darbības traucējumiem.

5.2.1. Piegādes risinājums darbībai uz serveriem

Risinājumu piegāde uz serveriem tika veikta, izmantojot uzstādīšanas instrukcijas un iepriekš izveidotu arhīvu, kurš satur konteinera izveides failu, izstrādātos robotizētu procesu automatizācijas risinājumus un daļu no sistēmas atkarību instalācijai nepieciešamajiem failiem. Šādā veidā piegāde tika atrisināta visā programmatūras piegādes ciklā.

Izstrādātā konteinera risinājuma maskētā pirmkodā, kas atrodams 1. pielikumā, ir redzams, kā tika izveidots konteiners "RobotFramework" risinājuma izpildei un atrisinātas nepieciešamās bibliotēku un sistēmas atkarības.

Izveidotais konteiners balstās uz "Fedora Linux" kodolu. Ņemot vērā konteinera risinājuma darbību un īpašības tiek sasniegts nepieciešamais programmatūras risinājuma izolācijas līmenis esošajā infrastruktūrā. Visas robotizētu procesu automatizācijas risinājumu bibliotēkas un sistēmas atkarības ir ietvertas konteinera līmenī, nodrošinot risinājuma darbību, taču izolējot tās no pārējās infrastruktūras, tādējādi novēršot risku radīt traucējumus citām lietotnēm un procesiem, kuri darbojas uz vienotās infrastruktūras.

Atkarības tiek iegūtas ar iespēju tās lejupielādēt, izmantojot tīmekļa savienojumu, izņemot "Oracle SQL plus" darbībai nepieciešamās sistēmas atkarības. Šīs atkarības uzstādīšanai nepieciešamās instalācijas tiek iekļautas arhīvā un konteinera risinājuma izstrādes brīdī tajā kopētas, uzstādītas un konfigurētas. Konteinera iekšienē tika konfigurētas visas nepieciešamās tiesības un konteinera izveides koda beigās tika iestatīts robotizētu procesu automatizācijas risinājumu darbības izpildes skripts.

5.2.2. Piegādes risinājums darbībai uz darbstacijām

Risinājumu piegāde uz darbstacijām tika veikta, izmantojot uzstādīšanas un lietošanas instrukcijas un iepriekš izveidotu "Oracle VirtualBox" virtuālo mašīnu, kura sevī ietver uzstādītu "Docker" konteineru platformas risinājumu robotizētu procesu automatizācijas darbības nodrošināšanai.

Risinājumu piegādei uz darbstacijām tiek izmantots tāds pats risinājums, kā tos piegādā uz serveriem, taču tas tiek ietverts papildus virtuālajā mašīnā, rezultātā piegādei notiekot saīsinātā programmatūras piegādes ciklā.

Risinājuma piegādei uz darbstacijām izveidotā virtuālā mašīna darbojās darbstacijās, kurās uzstādīts "Fedora Linux", šādā veidā nodrošinot jau izstrādātā risinājuma atkārtotu izmantošanu un darbības nodrošināšanu uz Windows darbstacijas bez risinājuma izmaiņām.

REZULTĀTI

Bakalaura darba izstrādes gaitā tika izpētītas programmatūras piegādes procesa mērķis un metodes, kā arī šī procesa automatizācijas iespējas, prasības un ieguvumi. Darbā tika veikta programmatūras darbības izolācijas veidu izpēte, šādas prakses izmantošanas ieguvumu, sarežģījumu un iespēju izpēte.

Papildus tika izpētītas vairākas maksas un atvērtā pirmkoda robotizētu procesu automatizācijas izstrādes platformas, padziļināti pētot uz šādām platformām izstrādātu risinājumu darbībai nepieciešamo infrastruktūru un pašu risinājumu piegādes veidus.

Praktiski veicot robotizētu procesu automatizācijas risinājuma piegādi reālam klientam specifiskos, vairāku iesaistīto piegādātāju apstākļos, bija iespējams pārbaudīt veiktās izpētes rezultātus un sarežģījumus.

Pēc klienta prasību precizēšanas un izveidotā risinājuma izstrādes, bija iespējams izmantot vēlamo izolācijas līmeni risinājuma darbības nodrošināšanai, izveidojot efektīvu risinājumu klienta specifiskajā situācijā un atverot iespējas nākotnē automatizēt visu robotizētu procesu automatizācijas risinājuma piegādi, izmantojot esošo risinājumu par pamatu.

SECINĀJUMI

Pēc bakalaura darba laikā veiktās programmatūras piegādes analīzes un izpētes varu secināt, ka šī procesa automatizācija ir efektīvs veids, kā saīsināt programmatūras piegādes laiku un stabilitāti cauri visam izstrādes un piegādes ciklam, taču šis process pieprasa sarežģītu, laikietilpīgu un detalizētu automatizācijas izveidi, kas ir pielāgota specifiskās programmatūras un klienta vajadzībām.

Programmatūras izolācijas veidu izpētes laikā tika secināts, ka šādu metožu izmantošana uzlabo fizisko serveru resursu optimālu izmantošanu un atvieglo programmatūras izstrādes laikā izveidoto risinājumu pārvietošanu citā atrašanās vietā, taču programmatūras izolācijas metožu izmantošana ievieš papildus sarežģītību izstrādes procesā un sevī ietver potenciālus drošības riskus nepārdomāta risinājuma gadījumā.

Robotizētu procesu automatizācijas izstrādes platformas izpētes un salīdzināšanas laikā tika secināts, ka kopumā visas platformas cenšas nodrošināt visu klientam nepieciešamo un vēlamo tehnoloģiju un iespēju kopumu, taču reizē ar funkcionalitātes un pārskatāmības uzlabojumiem tiek paaugstinātas prasības pret nepieciešamo infrastruktūru un fiziskajiem resursiem.

Maksas platformās vienkopus ir viss nepieciešamais daudzpusīgu risinājumu izstrādē un tiek sniegts platformas ražotāja atbalsts problēmu gadījumā, taču vienlaikus šādām platformām ir salīdzinoši labi precizētas, bet specifiskas prasības nepieciešamajai programmatūrai un pieejamajiem sistēmas resursiem.

Turpretim bezmaksas platformas sevī ietver samazinātu pieejamās funkcionalitātes apjomu un sarežģītāku risinājumu izstrādi, kā arī pieejamais atbalsts problēmu gadījumā galvenokārt tiek risināts, izmantojot platformas kopienu vai novirzīts risinājuma izstrādātājiem. Šādas platformas nespēj sniegt detalizētas prasības pret nepieciešamajiem serveru resursiem, taču spēj nodrošināt mazāku nepieciešamās programmatūras apjomu, tādējādi atvieglojot risinājumu piegādi. Šādas platformas galvenokārt strādā, izmantojot atvērtā pirmkoda risinājumus funkcionalitātes paplašināšanai. Robotizētu procesu automatizācijas risinājumu piegāde ir tieši atkarīga no izvēlētās izstrādes platformas, mērķa sistēmas un klienta vēlmēm un prasībām, šī iemesla dēļ katrs risinājums ir jāveido un jāpiegādā individuāli klienta apstākļiem.

Bakalaura darba izstrādes laikā autors nostiprināja un ieguva jaunas zināšanas programmatūras piegādē un šī procesa automatizācijā, programmatūras izolācijā un kopumā uzlaboja izpratni par robotizētu procesu automatizācijas risinājumu sniegtajām iespējām, piegādi un tās specifiku.

IZMANTOTĀ LITERATŪRA

1. “Distribution” [tiešsaiste]. [Atsauce 24.04.2019]. Pieejams internetā: https://www.techopedia.com/definition/3263/distribution
2. “A word about software distribution methods” [tiešsaiste]. [Atsauce 24.04.2019]. Pieejams internetā: https://theosperiment.wordpress.com/2011/05/01/a-word-about-software-distribution-methods/
3. “What is Software Delivery?” [tiešsaiste]. [Atsauce 24.04.2019]. Pieejams internetā: https://wbsimms.com/what-is-software-delivery/
4. “What is DevOps?” [tiešsaiste]. [Atsauce 25.04.2019]. Pieejams internetā: https://www.atlassian.com/devops
5. “DevOps Engineer Roles and Responsibilities” [tiešsaiste]. [Atsauce 25.04.2019]. Pieejams internetā: https://www.bmc.com/blogs/devops-engineer-roles-and-responsibilities/
6. “DevOps Team Roles And Responsibilities” [tiešsaiste]. [Atsauce 25.04.2019]. Pieejams internetā: https://hackernoon.com/devops-team-roles-and-responsibilities-6571cfb56843
7. “What is continuous integration?” [tiešsaiste]. [Atsauce 27.04.2019]. Pieejams internetā: https://www.atlassian.com/continuous-delivery/continuous-integration
8. “Continuous integration vs. continuous delivery vs. continuous deployment” [tiešsaiste]. [Atsauce 27.04.2019]. Pieejams internetā: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment
9. “What are the differences between continuous integration, continuous delivery, and continuous deployment?” [tiešsaiste]. [Atsauce 27.04.2019]. Pieejams internetā: https://www.plutora.com/blog/continuous-integration-continuous-delivery-continuous-deployment
10. “Configuration as Code: Everything You Need to Know” [tiešsaiste]. [Atsauce 29.04.2019]. Pieejams internetā: https://rollout.io/blog/configuration-as-code-everything-need-know/
11. “What is configuration as code?” [tiešsaiste]. [Atsauce 29.04.2019]. Pieejams internetā: https://confluence.atlassian.com/bamboo/what-is-configuration-as-code-894743909.html
12. “What is Infrastructure as Code?” [tiešsaiste]. [Atsauce 29.04.2019]. Pieejams internetā: https://docs.microsoft.com/en-us/azure/devops/learn/what-is-infrastructure-as-code

13. “What is Infrastructure as Code? IaC Explained” [tiešsaiste]. [Atsauce 29.04.2019]. Pieejams internetā: https://www.bmc.com/blogs/infrastructure-as-code/
14. “Containers vs Virtual Machines: What’s The Difference?” [tiešsaiste]. [Atsauce 01.05.2019]. Pieejams internetā: https://www.bmc.com/blogs/containers-vs-virtual-machines/
15. “What is a Container?” [tiešsaiste]. [Atsauce 01.05.2019]. Pieejams internetā: https://www.docker.com/resources/what-container
16. “What is the difference between a process, a container, and a VM?” [tiešsaiste]. [Atsauce 01.05.2019]. Pieejams internetā: https://medium.com/@jessgreb01/what-is-the-difference-between-a-process-a-container-and-a-vm-f36ba0f8a8f7
17. “What Are Containers?” [tiešsaiste]. [Atsauce 01.05.2019]. Pieejams internetā: https://www.netapp.com/us/info/what-are-containers.aspx
18. “What are containers, and how do they work?” [tiešsaiste]. [Atsauce 01.05.2019]. Pieejams internetā: https://searchitoperations.techtarget.com/tip/What-are-containers-and-how-do-they-work
19. “The Power of Hypervisor-Based Containers” [tiešsaiste]. [Atsauce 01.05.2019]. Pieejams internetā: https://thenewstack.io/hypervisors-container-era/
20. “Understanding Citrix VDI: XenDesktop and VDI-in-a-Box” [tiešsaiste]. [Atsauce 02.05.2019]. Pieejams internetā: https://searchservervirtualization.techtarget.com/definition/virtual-machine
21. “Pros and cons of physical servers” [tiešsaiste]. [Atsauce 02.05.2019]. Pieejams internetā: https://www.datacenterdynamics.com/opinions/pros-and-cons-of-physical-servers/
22. “Physical Servers vs Virtual Servers for Business” [tiešsaiste]. [Atsauce 03.05.2019]. Pieejams internetā: https://www.optimalnetworks.com/2015/09/04/physical-servers-vs-virtual-servers-business/
23. “Robotic Process Automation” [tiešsaiste]. [Atsauce 03.05.2019]. Pieejams internetā: https://www.uipath.com/rpa/robotic-process-automation
24. “What is Robotic Process Automation?” [tiešsaiste]. [Atsauce 07.05.2019]. Pieejams internetā: https://www.aiim.org/What-is-Robotic-Process-Automation
25. “What is RPA? A revolution in business process automation”[tiešsaiste]. [Atsauce 07.05.2019]. Pieejams internetā: https://www.cio.com/article/3236451/what-is-rpa-robotic-process-automation-explained.html

26. “UiPath REVIEW”[tiešsaiste]. [Atsauce 08.05.2019]. Pieejams internetā: https://reviews.financesonline.com/p/uipath/
27. “UiPath Orchestrator Guide Introduction” [tiešsaiste]. [Atsauce 08.05.2019]. Pieejams internetā: https://orchestrator.uipath.com/docs/introduction
28. “UiPath Orchestrator Guide About Physical Deployment” [tiešsaiste]. [Atsauce 08.05.2019]. Pieejams internetā: https://orchestrator.uipath.com/docs/about-physical-deployment
29. “UiPath Orchestrator Guide Software Requirements” [tiešsaiste]. [Atsauce 09.05.2019]. Pieejams internetā: https://orchestrator.uipath.com/docs/software-requirements
30. “UiPath Orchestrator Guide Hardware Requirements” [tiešsaiste]. [Atsauce 09.05.2019]. Pieejams internetā: https://orchestrator.uipath.com/docs/hardware-requirements-orchestrator
31. “Automation Anywhere Enterprise REVIEW” [tiešsaiste]. [Atsauce 09.05.2019]. Pieejams internetā: https://reviews.financesonline.com/p/automation-anywhere-enterprise/
32. “AUTOMATION ANYWHERE ENTERPRISE” [tiešsaiste]. [Atsauce 09.05.2019]. Pieejams internetā: https://www.automationanywhere.com/products/enterprise
33. “Enterprise Control Room Overview” [tiešsaiste]. [Atsauce 10.05.2019]. Pieejams internetā: https://docs.automationanywhere.com/bundle/enterprise-v11.3/page/topics/control-room/getting-started/control-room-overview.html
34. “Enterprise Control Room overview” [tiešsaiste]. [Atsauce 11.05.2019]. Pieejams internetā: https://docs.automationanywhere.com/bundle/enterprise-v11.3/page/topics/aae-architecture-implementation/control-room-overview.html
35. “Enterprise Control Room prerequisites” [tiešsaiste]. [Atsauce 11.05.2019]. Pieejams internetā: https://docs.automationanywhere.com/bundle/enterprise-v11.3/page/topics/control-room/install/prerequisites-control-room.html
36. “Automation Anywhere Tutorial” [tiešsaiste]. [Atsauce 12.05.2019]. Pieejams internetā: https://www.guru99.com/automation-anywhere-tutorial.html
37. “UiPath Robot Guide Introduction” [tiešsaiste]. [Atsauce 12.05.2019]. Pieejams internetā: https://robot.uipath.com/docs/introduction
38. “UiPath Robot Guide About Robot Deployment Types” [tiešsaiste]. [Atsauce 13.05.2019]. Pieejams internetā: https://robot.uipath.com/docs/about-robot-deployment-types

39. “UiPath Robot Guide Software Requirements” [tiešsaiste]. [Atsauce 14.05.2019]. Pieejams internetā: https://robot.uipath.com/docs/software-requirements
40. “UiPath Robot Guide Hardware Requirements” [tiešsaiste]. [Atsauce 14.05.2019]. Pieejams internetā: https://robot.uipath.com/docs/hardware-requirements
41. “Deployed components” [tiešsaiste]. [Atsauce 14.05.2019]. Pieejams internetā: https://docs.automationanywhere.com/bundle/enterprise-v11.3/page/topics/aae-architecture-implementation/components-datacenter-deploy.html
42. “Enterprise client prerequisites” [tiešsaiste]. [Atsauce 14.05.2019]. Pieejams internetā: https://docs.automationanywhere.com/bundle/enterprise-v11.3/page/topics/aae-client/install/enterprise-prerequisites.html#Zj0vY2F0ZWdvcnkvaW5zdGFsbD9wPUluc3RhbGw=
43. “Bot Lifecycle Management – Bring Calm to Your Bot Development Chaos” [tiešsaiste]. [Atsauce 14.05.2019]. Pieejams internetā: https://www.automationanywhere.com/blog/changing-the-world-with-automation/bot-lifecycle-management-bring-calm-to-your-bot-development-chaos
44. “Automation Anywhere architecture” [tiešsaiste]. [Atsauce 14.05.2019]. Pieejams internetā: https://docs.automationanywhere.com/bundle/enterprise-v11.3/page/topics/aae-architecture-implementation/architecture-overview.html
45. “Robot Framework INTRODUCTION” [tiešsaiste]. [Atsauce 15.05.2019]. Pieejams internetā: https://robotframework.org/
46. “ROBOT FRAMEWORK RPA” [tiešsaiste]. [Atsauce 15.05.2019]. Pieejams internetā: https://robotframework.org/rpa/
47. “Robot Framework LIBRARIES” [tiešsaiste]. [Atsauce 15.05.2019]. Pieejams internetā: https://robotframework.org/#libraries
48. “Kantu XModules Pricing” [tiešsaiste]. [Atsauce 17.05.2019]. Pieejams internetā: https://ui.vision/kantu/x/pricing#ee
49. “UI Vision for Desktop Automation” [tiešsaiste]. [Atsauce 17.05.2019]. Pieejams internetā: https://ui.vision/
50. “UI.Vision Kantu” [tiešsaiste]. [Atsauce 17.05.2019]. Pieejams internetā: https://ui.vision/kantu
51. “Kantu XModules” [tiešsaiste]. [Atsauce 17.05.2019]. Pieejams internetā: https://ui.vision/kantu/x

PIELIKUMI

1. Pielikums. “RobotFramework” platformas risinājuma darbības konteiners

```
FROM fedora:29
MAINTAINER Elvis Placis
LABEL RPA RobotFramework in Docker

# Setup volumes for input and output
VOLUME /opt/robotframework/reports
VOLUME /opt/robotframework/tests

ENV SCREEN_COLOUR_DEPTH 24
ENV SCREEN_HEIGHT 1080
ENV SCREEN_WIDTH 1920

# Set number of threads for parallel execution
# By default, no parallelisation
ENV ROBOT_THREADS 1

# Dependency versions
ENV CHROMIUM_VERSION 71.0.*
ENV FIREFOX_VERSION 65.0*
ENV GECKO_DRIVER_VERSION v0.22.0
ENV PABOT_VERSION 0.51
ENV PYTHON_PIP_VERSION 18.0*
ENV ROBOT_FRAMEWORK_VERSION 3.1.1
ENV SELENIUM_LIBRARY_VERSION 3.3.1
ENV XVFB_VERSION 1.20.*
ENV CX_ORACLE_VERSION 7.1
ENV DATABASE_LIBRARY_VERSION 1.2

# Install system dependencies
RUN dnf upgrade -y \
  && dnf install -y \
    chromedriver-$CHROMIUM_VERSION \
    chromium-$CHROMIUM_VERSION \
    firefox-$FIREFOX_VERSION \
    python2-pip-$PYTHON_PIP_VERSION \
    xauth \
    xorg-x11-server-Xvfb-$XVFB_VERSION \
    which \
    wget \
  && dnf clean all

# Install Database dependencies
COPY oracle-instantclient12.1-basic-12.1.0.2.0-1.x86_64.rpm /opt/robotframework/bin/dbdriver/
COPY oracle-instantclient12.1-devel-12.1.0.2.0-1.x86_64.rpm /opt/robotframework/bin/dbdriver/
COPY oracle-instantclient12.1-sqlplus-12.1.0.2.0-1.x86_64.rpm /opt/robotframework/bin/dbdriver/

RUN chmod -R 7777 /opt/robotframework/*
RUN dnf install -y /opt/robotframework/bin/dbdriver/oracle-instantclient12.1-basic-12.1.0.2.0-1.x86_64.rpm
RUN dnf install -y /opt/robotframework/bin/dbdriver/oracle-instantclient12.1-devel-12.1.0.2.0-1.x86_64.rpm
RUN dnf install -y /opt/robotframework/bin/dbdriver/oracle-instantclient12.1-sqlplus-12.1.0.2.0-1.x86_64.rpm
RUN dnf install -y libnsl

ENV LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib/
ENV ORACLE_HOME=/usr/lib/oracle/12.1/client64

RUN echo export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib/ >> /etc/bashrc
RUN echo export ORACLE_HOME=/usr/lib/oracle/12.1/client64 >> /etc/bashrc
```

```
RUN echo export PATH=/opt/robotframework/bin:/opt/robotframework/drivers:/usr/lib/oracle/12.1/client64/bin:$PATH >> /etc/bashrc
RUN ln -s libclntsh.so.12.1 libclntsh.so
RUN echo /usr/lib/oracle/12.1/client64/lib > /etc/ld.so.conf.d/oracle-instantclient12.1.conf
RUN chmod -R 7777 /usr/lib/oracle/*
#RUN ln -s $ORACLE_HOME/libclntsh.so.12.1 libclntsh.so
#ENV PATH "/usr/lib/oracle/12.1/client64/bin:$PATH"

# Install Robot Framework and Required Libraries
RUN pip install \
  robotframework==$ROBOT_FRAMEWORK_VERSION \
  robotframework-pabot==$PABOT_VERSION \
  robotframework-seleniumlibrary==$SELENIUM_LIBRARY_VERSION \
  robotframework-databaselibrary==$DATABASE_LIBRARY_VERSION \
  cx_Oracle==$CX_ORACLE_VERSION

# Download Gecko drivers directly from the GitHub repository
RUN wget -q "https://github.com/mozilla/geckodriver/releases/download/$GECKO_DRIVER_VERSION/geckodriver-$GECKO_DRIVER_VERSION-linux64.tar.gz" \
      && tar xzf geckodriver-$GECKO_DRIVER_VERSION-linux64.tar.gz \
      && mkdir -p /opt/robotframework/drivers/ \
      && mv geckodriver /opt/robotframework/drivers/geckodriver \
      && rm geckodriver-$GECKO_DRIVER_VERSION-linux64.tar.gz

# Prepare binaries to be executed
COPY chromedriver.sh /opt/robotframework/bin/chromedriver
COPY chromium-browser.sh /opt/robotframework/bin/chromium-browser
COPY run-tests-in-virtual-screen.sh /opt/robotframework/bin/

RUN mv /usr/lib64/chromium-browser/chromium-browser /usr/lib64/chromium-browser/chromium-browser-original \
  && ln -sfv /opt/robotframework/bin/chromium-browser /usr/lib64/chromium-browser/chromium-browser

RUN chmod -R 7777 /opt/robotframework/*
# Update system path
ENV PATH=$PATH:/opt/robotframework/bin:/opt/robotframework/drivers:/usr/lib/oracle/12.1/client64/bin
# Execute robot script
CMD ["run-rpa-virtual-screen.sh"]
```

Bakalaura darbs „Robotizētu procesu automatizācijas risinājumu piegāde” izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: _____ Elvis Plācis

Rekomendēju/nerekomendēju darbu aizstāvēšanai *(nederīgo svītro vadītājs)*
Vadītājs: Apdrošināšanas risinājumu nodaļas vadītājs M.dat. Andris Bozis
_____ __.05.2019.

Recenzents: prof., Dr.dat. Jānis Zuters

Darbs iesniegts Datorikas fakultātē __.05.2019.
Dekāna pilnvarotā persona: vecākā metodiķe Ārija Sproģe _____

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē
____.06.2019. prot. Nr. _____
Komisijas sekretārs(-e): _____