LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

VEIKTSPĒJAS DATU ANALĪZES RĪKS

KVALIFIKĀCIJAS DARBS

Autore: **Ludmila Mačukāne**
Studenta apliecības nr.: lm17052
Darba vadītājs: M.dat. Aleksandrs Dolmatovs

RĪGA 2019

ANOTĀCIJA

„Veiktspējas datu analīzes rīks" ir tīmekļa lietojumprogrammatūra, kuras uzdevums ir atvieglot testēšanas rezultātā iegūto veiktspējas datu pārskatāmību, uztveramību un rezultātu salīdzināmību.

Piekļūstot ārējam serverim, kurā tiek glabāti testēšanas laikā iegūtie dati, "Veiktspējas datu analīzes rīks" ļauj efektīvi vizualizēt un attēlot centrālā procesora (CPU) un brīvpiekļuves atmiņas (RAM) lietojamības testu rezultātus, caur lietotājiem draudzīgu un viegli uztveramu lietotāja interfeisu (UI), atvieglojot tālāko datu analīzes procesu.

Rīks sniedz iespēju vizuāli aplūkot un salīdzināt datus konkrētos laika posmos pēc vēlamajiem parametriem: testa nosaukuma, platformas nosaukuma, platformas versijas, ierīces nosaukuma, programmatūras izstrādes komplekts (SDK) nosaukuma, programmatūras izstrādes komplekta versijas, "job" nosaukuma un būvējuma nosaukuma.

Veiktspējas datu analīzes rīks tika izstrādāts, izmantojot JavaScript, HTML un CSS valodas.

Atslēgvārdi: CPU, RAM, testēšana, veiktspējas datu analīzes rīks, Chart.js.

ABSTRACT

PERFORMACE DATA ANALYSIS TOOL

"Performance data analysis tool" is web-based application which is created in order to facilitate data overview, visibility and comparability of the performance tests results.

Accessing an external server that holds test data, performance data analysis tool allows to effectively visualize and display central processing unit (CPU) usage and random-access memory (RAM) usage test results, through user-friendly and easy-to-understand user interface (UI). The tool provides the opportunity to visually view and compare data over specific time periods according to desired parameters: job name, build number, test name, platform name, platform version, device name, software development kit (SDK) name and SDK version.

Performance data analysis tool was developed using JavaScript, HTML and CSS programming languages.

Keywords: CPU, RAM, testing, performance data analysis tool, Chart.js.

SATURS

APZĪMĒJUMU SARAKTS

Būvējums - (build) programmatūras komponenti, kuri iegūti kompilēšanas rezultātā no atsevišķu avotu versiju kopas. [1]

Chart.js - atvērtā koda JavaScript bibliotēka, kas attēlo datus tīmekļa lietojumprogrammās.

CPU - centrālais procesors.

CSRF - vairāku vietņu pieprasījuma viltošana (tulkots no angļu valodas - cross-site request forgery).

CSS - īpaša stila lapas valoda, ko lieto, lai aprakstītu izskatu iezīmēšanas valodā veidotiem dokumentiem.

Git - versiju kontroles sistēmā.

GitLab - Git repozitoriju pārvaldnieks.

HTML - hiperteksta iezīmēšanas valoda, kas ir izstrādāta tīmekļa lappušu un citas pārlūkprogrammā attēlojamas informācijas glabāšanai.

Integrētā izstrādes vide (IDE) - integrētu rīku komplekts programmatūras izstrādei. Rīki parasti tiek darbināti no viena lietotāja interfeisa un sastāv no kompilatora, redaktora un atkļūdotāja.

JavaScript - skriptu programmēšanas valoda.

Job name – identifikators pēc kura var noteikt pie kuras no testu izpildēm pieder konkrētais tests.

Kompilēt - visu programmas pirmkodu tulkošana no augsta līmeņa valodas uz objekta kodu pirms programmas izpildes.

RAM - brīvpiekļuves atmiņa.

Release candidate (RC) – kandidātlaidiens – Būvējuma versija, kas ir pārbaudīta un ir gatava izlaišanai.

SDK - (software development kit) programmatūras izstrādes komplekts, kas ļauj izveidot lietojumprogrammas konkrētai programmatūras pakotnei, programmatūras ietvaram, datoru sistēmai, operētājsistēmai vai tamlīdzīgai izstrādes platformai.

Test run - testa izpilde.

XSS - Starpvietņu skriptošana jeb datoru drošības ievainojamību paveids.

IEVADS

Savlaicīga un atbilstoša testēšana viennozīmīgi ir neatņemama un ļoti būtiska programmatūras izstrādes daļa. Testēšana palīdz identificēt defektus un atteices programmatūrā, sniedzot informāciju par tālākiem riskiem, kuri varētu rasties, nododot produktu ekspluatācijā. Ne tikai programmatūras testēšana, bet arī iegūto testēšanas datu apstrāde un analīze ir ļoti būtiska. Atbilstoša datu apstrāde būtiski atvieglo testu rezultātu uztveri un izvērtēšanu, palīdzot informāciju sistēmu testētājiem atbilstoši novērtēt iegūtos testu datus un veikt secinājums.

Kvalifikācijas darba mērķis ir izstrādāt "Veiktspējas datu analīzes rīku", kas atvieglotu testēšanas rezultāta iegūto veiktspējas datu pārskatāmību, uztveramību un rezultātu salīdzināmību. Piekļūstot serverim, kurā tiek uzglabāti testēšanas laikā iegūtie dati, rīks sniedz iespēju grafiski attēlot, kāds ir procentuālais centrālā procesora (CPU) noslogojums un brīvpiekļuves atmiņas (RAM) patēriņš megabaitos konkrētā laika posmā. "Veiktspējas datu analīzes rīku" ļauj salīdzināt un analizēt konkrētus veiktspējas datus pēc vēlamajiem parametriem: testa nosaukuma, platformas nosaukuma - iOS, Android, vai kāda no interneta pārlūkprogrammām, platformas versijas , ierīces nosaukuma, programmatūras izstrādes komplekts (SDK) nosaukuma, programmatūras izstrādes komplekta versijas, "job" nosaukuma un būvējuma nosaukuma.

Uzņēmumā veicot informāciju sistēmu testētājas pienākumus, autore kopā ar komandas vadītāju un kolēģiem secināja, ka pašlaik, esošā projekta ietvaros, nav pieejams rīks, kas efektīvi un pārskatāmi spētu attēlot veiktspējas testu datus, tādēļ, balstoties uz kvalitātes nodrošināšanas specialistu vēlmēm, kuri attiecīgajā projektā ir atbildīgi par veiktspējas testu veikšanu, kā arī rezultātu apkopošanu, tika nolemts izstādāt "Veiktspējas datu analīzes rīku", kas viennozīmīgi atvieglotu datu analīzes procesu, kā arī veicinātu autores izpratni par sistēmas izstādi un programmēšanu.

Lai sasniegtu izvirzīto mērķi, tika izvirzīti sekojošie uzdevumi:

1. apgūt JavaScript, Chart.js, Git, rīkus un tehnoloģijas;
2. izstrādāt programmatūras prasību specifikāciju;
3. izstrādāt programmatūras projektējuma aprakstu, balstoties uz izveidoto prasību specifikāciju;
4. izstrādāt rīku;
5. veikt rīka testēšanu un dokumentēt to;
6. ieviest rīku konkrētajā projektā.

Rīks tika izstrādāts pēc ūdenskrituma programmatūras izstrādes dzīves cikla modeļa - sākumā tika nodefinētas vēlamās prasībās un izveidots projektējuma apraksts, un vadoties pēc šī plāna tika veikta rīka izstrāde. Izstrādājot rīku, darba gaitā notika nelielas novirzes no sākotnējā plāna, pielāgojot rīku projekta vajadzībām, kā arī autores pieredzes trūkuma dēļ.

Darbs sastāv no rīka programmatūras prasību specifikācijas, programmatūras projektējuma apraksta, testēšanas dokumentācijās, kā arī no projekta organizācijas, kvalitātes nodrošināšanas, konfigurāciju pārvaldības aprakstiem, darbietilpības novērtēšanas un pielikuma, kurā ir aplūkojami pirmkoda fragmenti, lietotāju saskarnes projektējumi, kā arī izstrādātās sistēmas skati.

1. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA

1.1. Ievads

1.1.1. Nolūks

Šīs programmatūras prasību specifikācijas (PPS) nolūks ir noteikt un apkopot visas prasības, kuras ir jārealizē „Veiktspējas datu analīzes rīka" izstrādē.

Šis dokuments ir savstarpējā vienošanās starp pasūtītāju un izpildītāju, un rīks tiks izstrādāts atbilstoši dokumentā noteiktajām prasībām.

1.1.2. Darbības sfēra

„Veiktspējas datu analīzes rīks" ir testu datu vizualizācijas rīks, veiktspējas testu datu attēlošanai un salīdzināšanai. Tā galvenās funkcijas ir iegūto veiktspējas testu datu attēlošana grafikos un salīdzināšana, atlasot tos pēc noteiktiem kritērijiem – "job" nosaukuma, būvējuma nosaukuma, platformas nosaukuma, ierīces nosaukuma, platformas versijas, SDK nosaukuma un SDK versijas, testa nosaukuma.

Izstrādājamo produktu ir paredzēts izmantot konkrēta projekta ietvaros informācijas tehnoloģiju sfērā, un tā mērķis ir atvieglot testēšanas rezultātā iegūto veiktspējas datu pārskatāmību, uztveramību un rezultātu salīdzināmību. Piemēram, tas var tikt izmantots, lai salīdzinātu dažādas testa izpildes (test run), lai salīdzinātu testus, kuri tika veikti uz dažādām platformām un dažādām platformas versijām, lai salīdzinātu dažādus kandidātlaidienus, kā arī dažādas būvējuma versijas. Nepieciešamības gadījumā rīku var optimizēt un pielāgot, lai attēlotu, salīdzinātu un analizētu ne tikai veiktspējas, bet arī cita veida testa datus.

1.1.3. Saistība ar citiem dokumentiem

Dokuments tika izstrādāts un noformēts, ievērojot standarta LVS 68:1996 „Programmatūras prasību specifikācijas ceļvedis" [2] prasības un rekomendācijas.

1.1.4. Pārskats

Dokuments sastāv no 4 nodaļām:

1. Ievads. Nodaļā tiek aprakstīts dokumenta nolūks, produkta darbības sfēra, dokumenta struktūra kā arī saistība ar citiem dokumentiem.
2. Vispārējais apraksts. Nodaļā tiek īsumā izklāstīta produkta perspektīva, funkcionalitāte, informācija par rīka potenciālajiem lietotājiem, kā arī definēti rīka ierobežojumi, pieņēmumi un atkarības.

3. Funkcionālas prasības. Nodaļā tiek aprakstīta rīka funkcionalitāte, funkciju mērķi, ievaddati, apstrāde, izvaddati un kļūdu paziņojumi.
4. Nefunkcionālas prasības. Nodaļā tiek aprakstītas sistēmas drošības, veiktspējas, uzturamības un izmantojamības prasības.

1.2. Vispārējais apraksts

1.2.1. Produkta perspektīva

Izstrādājamais produkts ir saistīts ar ārēju serveri, kurā, tiek uzglabāti testu dati.

1.2.2. Projekta funkcijas

„Veiktspējas datu analīzes rīkam" tiks nodrošinātas sekojošas funkcijas un metodes:

1. Ienākošo datu apstrādes modulis
   - Ienākošo datu apstrāde
   - Ienākošo datu atbilstības pārbaude
2. Datu filtrēšanas modulis
   - Attiecīgo filtru datu apstrāde
   - Pakāpeniska datu filtrēšana
   - Vairāk kā viena grafika testu datu filtrēšana
3. Datu vizualizācijas modulis
   - Atlasīto filtru datu vizualizēšana
   - Testu komandu nosaukumu attēlošana
   - Līniju, kuras atdala jaunas testu komandu sākšanos attēlošana
   - Testu informācijas attēlošana
   - "Show", "Hide" un "Delete" opcijas
   - Vairāk kā viena grafika attēlošana

1.2.3. Lietotāja raksturiezīmes

Rīku ir paredzēts lietot informāciju tehnoloģijas nozares darbiniekiem - kvalitātes nodrošināšanas speciālistiem un programmatūras izstrādātājiem, konkrēta projekta ietvaros, lai optimizētu testu datu analīzes procesu. Rīka lietotājiem ir jābūt izpratnei par konkrētiem veiktspējas testu datiem, to atbilstību un novirzēm no normas, kā arī izpratnei pēc kādiem parametiem tiks atlasīti testu dati. Rīku ir iespējams lietot lokāli, ievērojot projekta datu drošības prasības - izmantojot integrētu izstrādes vidi un vēlamo pārlūkprogrammu.

Lai izprastu rīka darbības principus, rīka lietotājiem ir vēlams iepazīties ar šo dokumentu.

1.2.4. Vispārējie ierobežojumi

Rīkam ir sekojošie ierobežojumi:

- darbam ar sistēmu nepieciešams dators ar piekļuvi internetam;
- pašlaik rīku ir iespējams lietot tikai lokāli, konkrētā projekta ietvaros;
- lai būtu iespējams lietot veiktspējas datu analīzes rīku, datorā, kurā tiks lietots šis rīks ir jābūt uzinstalētai, kādai no integrētai izstrādes videi (IDE);
- sistēmas darbināšana notiek izmantojot tīmekļa pārlūkprogrammu;
- sistēmas darbībai jābūt atbalstītai uz četrām populārākajām pārlūkprogrammām: Google Chrome, Mozilla Firefox, Safari, Opera.

1.2.5. Pieņēmumi un atkarības

Tiek pieņemts, ka sistēmas lietotāji programmatūras darbināšanai izmantos aktuālas pārlūkprogrammu versijas, jo, izmantojot vecās pārlūkprogrammu versijas, tas var ietekmēt programmatūras darbību un to vizuālo izskatu. Tiek pieņemts, ka līdz ar tīmekļa pārlūkprogrammas vai integrētās izstrādes vides aizvēršanu, kā arī datora restartēšanu vai izslēgšanu, "Veiktspējas datu analīzes rīka" darbība tiek pārtraukta, tādēļ katru reizi pēc rīka darbības pārtraukšanas, tas jāatver izmantojot IDE.

1.3. Funkcionālas prasības

"Veiktspējas datu analīzes rīka" funkcijas pēc to nozīmes tiek sadalītas 3 moduļos:

- ienākošo datu apstrādes modulis - no servera saņemto datu apstrāde;
- datu filtrēšanas modulis - datu filtrēšana pēc konkrētā lietotāja vēlmēm;
- datu vizualizācijas modulis - datu vizualizācija saskaņā ar atlasītajiem datiem.

1.3.1. Ienākošo datu apstrādes modulis

Ienākošo datu apstrādes moduļa funkcijas nodrošina ienākošo datu korektu apstrādi, definējot kādi ienākošie dati būs nepieciešami rīka tālākajai darbībai.

Atkarībā no tā kādi testa dati un to parametri tiek uzglabāti serverī var definēt konkrētus testu parametrus, kuri vēlāk tiks attēloti grafiski. Konkrēti veiktspējas testu gadījumā dati tiks atlasīti pēc šādiem parametriem:

- “test run”, kas sastāv no laukiem “job” un “build”, kas apzīmē pie kuras no pie kuras no testu izpildēm pieder konkrētais tests un kāds ir tā būvējuma numurs;
- “name”, kas apzīmē konkrēto testa nosaukumu;
- “date”, kas apzīmē datumu un laiku, kurā konkrētais tests ticis izpildīts;
- “roles”, kas satur informāciju par lomām, kuras darbojas konkrētajā testā, konkrētāk “role”, kas definē cik lomu ir konkrētajā testā, “actor”, kas definē, kurš aktieris izpildīs kuru lomu, “os”, kas definē uz kuras platformas tiek izpildīta konkrētā testa loma, “device”- ierīce, kas izpilda šo lomu, “version”- konkrētās ierīces versija, “sdk name” - programmatūras izstrādes komplekta nosaukums, “sdk”, kas apzīmē to pie kura no kandidātlaidieniem pieder konkrētā programmatūra;
- “performance”, kas satur informāciju par konkrētā testa veiktspējas datiem, konkrētāk “command name” - testa komandas nosaukumu, “measurement time” - konkrētu laiku, kurā ir notikusi katra no komandām, “cpu usage”- kāds ir bijis centrālā procesora patēriņš pie konkrētā testa soļa, “ mem usage”, kāds ir bijis brīvpiekļuves atmiņas patēriņš pie konkrētā testa soļa.

Visi definētie parametri tiek uzskatīti par obligātām vērtībām un to neesamība tiek uzskatīta par kļūdu, nepieciešamības gadījumā šos parametrus var mainīt un pielāgot atkarībā no ienākošajiem testu datiem.

1.3.1.1. Ienākošo datu apstrādes funkcija

Identifikators P.DataProcess
Apraksts
Funkcija paredzēta, lai pārbaudītu vai no ārējā servera tiek saņemti visi definētie parametri.
Ievads
Obligātie parametri: • test run; • job; • build • name; • date; • roles; • role; • actor;

• os; • device; • version; • sdk name; • sdk; • performance; • command name; • measurement time • cpu usage; • mem usage; Neobligātie parametri: nav
Apstrāde
Pārbauda: • vai visi definētie parametri tiek saņemti no servera; Ja obligātie parametri sakrīt, tālāk tiek pārbaudīts vai to vērtības eksistē un tās ir korektas
Izvade
Atgriež būla tipa vērtību. Ja visi definētie parametri no servera tiek atgriezti, tad tiek atgriezts “true”, ja nē - “false”.
Paziņojumi
Nav

1.3.1.2. Ienākošo parametru vērtību apstrādes funkcija

Identifikators P.DataValueProcess
Apraksts
Funkcija paredzēta, lai pārbaudītu vai visas parametru vērtības eksistē un tās ir korektas.
Ievads
Obligātie parametri: • test run; • job; • build • name; • date; • roles; • role; • actor; • os; • device; • version;

• sdk name; • sdk; • performance; • command name; • measurement time • cpu usage; • mem usage; Neobligātie parametri: nav
Apstrāde
Pārbauda: • vai visiem no servera saņemtajiem parametriem ir definēta vērtība; • vai konkrētās vērtības datu tips tiek definēts korekti (string vai number) Ja obligātie parametri ir korekti, dati tiek nosūtīti uz datu filtrēšanas moduli.
Izvade
Atgriež būla tipa vērtību. Ja visu parametru vērības eksistē, tad tiek atgriezts “true”, ja nē - “false”.
Paziņojumi
Nav

1.3.2. Datu filtrēšanas modulis

Datu filtrēšanas moduļa funkcijas nodrošina attiecīgo filtru datu korektu apstrādi - pakāpenisku datu filtrēšanu izejot no tā kāda vērtība tika atlasīta pirms tam un automātiski pielāgojot pārējās izvēles opcijas. Filtrēšanas modulis arī nodrošinās vairāk kā viena grafika testu datu filtrēšanu. Datu filtrēšanas modulis atlasīs testu datus pēc šādiem parametriem:

- “Job name”, sadaļa, kurā lietotājs varēs redzēt visas pieejamās testu izpildes;
- “Platform”, sadaļa, kurā lietotājs pēc konkrēta “Job name” izvēles varēs redzēt kāds platformas ir pieejamas konkrēti šim “Job Name”;
- “Platform version”, sadaļa, kurā pēc konkrētas platformas izvēles varēs redzēt kādas platformas versijas šai platformai ir pieejamas;
- “ SDK name”, sadaļa, kurā pēc konkrētas platformas versijas izvēles lietotājs varēs izvēlēties tieši kuras programmatūras testu datus viņš vēlas redzēt;
- “SDK version”, sadaļa, kurā pēc “SDK name” izvēles lietotājs varēs izvēlēties, kura kandidātlaidiena testus viņš vēlas redzēt ;

- “Device”, sadaļa kurā pēc “SDK version” izvēles, lietotājam būs redzamas, ierīces uz kurām ir tikuši izpildīti konkrētie testi;
- “Build number”, sadaļa, kur lietotājs pēc konkrētas ierīces izvēles lietotājs varēs redzēt kuru no būvējuma versijām ir pieejama;
- “Test name”, pēdējā testu datu filtrēšanas sadaļa, kurā būs redzami, kādi testi ir pieejami balstoties uz visām iepriekš izvēlētajām opcijām.

Ir iespējams pievienot arī vairākus filtru blokus ar opciju “Add tests”, lai savā starpā varētu salīdzināt vairākus testus, kā arī ar opciju “Delete” šos papildus blokus var dzēst.

1.3.2.1. Filtru vērtību apstrādes funkcija

Identifikators F.DataProcessing
Apraksts
Funkcija paredzēta, lai pārbaudītu vai visas sākotnējas filtru vērtības eksistē un ir pieejamas.
Ievads
Obligātie parametri: • “Job name”; • “Platform”; • “Platform version”; • “SDK name”; • “SDK version”; • “Device”; • “Build number”; • “Test name”; Neobligātie parametri: nav
Apstrāde
Pārbauda: • vai visi definētie parametri eksistē un tiek saņemti ; Ja obligātie parametri sakrīt, tālāk tiek pārbaudīts vai to vērtības eksistē un tās ir korektas
Izvade
Atgriež būla tipa vērtību. Ja filtru vērības eksistē, tad tiek atgriezts “true”, ja nē - “false”.
Paziņojumi
Nav

1.3.2.2. Filtru datu vērtību apstrādes funkcija

Identifikators F.DataValueCheck
Apraksts
Funkcija paredzēta, lai pārbaudītu vai visu filtru datu vērtības ir korektas.
Ievads
Obligātie parametri: • “Job name”; • “Platform”; • “Platform version”; • “SDK name”; • “SDK version”; • “Device”; • “Build number”; • “Test name”; Neobligātie parametri: nav
Apstrāde
Pārbauda: • vai visiem filtru datiem ir definēta vērtība; • vai konkrētās filtru vērtības datu tips tiek definēts korekti (string vai number) Ja obligātie parametri ir korekti, dati tiek nosūtīti uz datu vizualizācijas modulis.
Izvade
Atgriež būla tipa vērtību. Ja visu datu tipu vērtības ir pieejamas un korektas tad tiek atgriezts “true”, ja nē - “false”.
Paziņojumi
Nav

1.3.2.3. Pakāpeniskā datu filtrēšanas funkcija

Identifikators F.DataGradualCheck
Apraksts
Funkcija paredzēta, lai pieejamos datus varētu filtrēt pakāpeniski- atlasot vienu vērtību pēc citas
Ievads
Obligātie parametri: • “Job name”; • “Platform”; • “Platform version”; • “SDK name”; • “SDK version”; • “Device”; • “Build number”; • “Test name”; Neobligātie parametri: nav
Apstrāde
Pārbauda: • vai ir pirms nākamā testa lauka aizpildes tiek aizpildīts iepriekšējais filtra lauks
Izvade
Ja iepriekšējais filtra lauks nav aizpildīts, tad pārējie filtru lauki ir neaktīvi un tos nav iespējams aizpildīt.
Paziņojumi
Nav

1.3.2.4. Filtru datu attēlošanas funkcija

Identifikators F.DataShowCheck
Apraksts
Funkcija paredzēta, lai korekti attēlotu filtru datus pēc tam, kad lietotājs ir pirmo filtra opciju
Ievads
Obligātie parametri: • “Job name”; • “Platform”;

• “Platform version”; • “SDK name”; • “SDK version”; • “Device”; • “Build number”; • “Test name”; Neobligātie parametri: nav
Apstrāde
Pārbauda: • Vai pēc konkrēta filtra lauka aizpildes nākošajā filtra laukā ir redzami korekti dati- vai nākošajā laukā ir redzami tikai tie dati, kuri attiecas uz iepriekšējā lauka izvēli.
Izvade
Ir redzami tikai tie dati, kuru parametrus lietotājs ir atzīmējis iepriekš
Paziņojumi
Nav

1.3.2.5. Filtru bloka pievienošanas funkcija

Identifikators F.AddFilerBlock
Apraksts
Funkcija paredzēta, lai pievienotu vienu vai vairākus filtru blokus
Ievads
Obligātie parametri: • “Job name”; • “Platform”; • “Platform version”; • “SDK name”; • “SDK version”; • “Device”; • “Build number”; • “Test name”; Neobligātie parametri: nav
Apstrāde
Pārbauda: • Vai pēc opcijas “Add test” tiek pievienots vēl viens testu filtru bloks • Vai arī pievienotie testu filtru bloki strādā korekti

Izvade
Pēc opcijas “Add test” izvēles lietotājam tiek attēlots jauns testu filtru bloks ar tādām pašām izvēles opcijām, kā sākotnējā filtru blokā
Paziņojumi
Nav

1.3.2.6. Filtru bloka dzēšanas funkcija

Identifikators F.DeleteFilerBlock
Apraksts
Funkcija paredzēta, lai dzēstu vienu vai vairākus filtru blokus
Ievads
Obligātie parametri: • “Job name”; • “Platform”; • “Platform version”; • “SDK name”; • “SDK version”; • “Device”; • “Build number”; • “Test name”; Neobligātie parametri: nav
Apstrāde
Pārbauda: • Vai pēc opcijas “Delete” tiek dzēsti iepriekš pievienotie filtru bloki.
Izvade
Pēc opcijas “Delete” izvēles lietotāja pievienotie filtru bloki tiek dzēsti
Paziņojumi
Nav

1.3.3. Datu vizualizācijas modulis

Datu vizualizācijas modelis ir cieši saistīts ar datu filtrācijas moduli, tā mērķis ir nodrošināt korektu filtru datu attēlošanu - vizualizējot atlasītos testu datus un attēlojot tos

grafikā. Grafikā tiek attēlots centrālā procesora (CPU usage (%)) un brīvpiekļuves atmiņas (RAM usage (MB)) lietojums atkarībā no konkrētā testa soļa izpildes konkrētā laika posmā. Grafikā tiek vizualizētas līnijas, kuras iezīmē jauna testa soļa sākšanos, kā arī konkrētā testa nosaukums noteiktā laika periodā. Izvēloties konkrētu grafika punktu parādās informācija par komandas nosaukumu, laiku kurā ir tikusi izpildīta šī komanda, kā arī konkrētais centrāla procesora vai brīvpiekļuves atmiņas lietojums šajā punktā.

Kad lietotājs ir atlasījis visus vēlamos testu datu, ar funkcijas "Show" palīdzību lietotājs varēs ieraudzīt izvēlētā testa grafiku. Ja lietotājs izvēlēsies opciju "Show" pirms visu filtru lauku aizpildes tiks izvadīts kļūdas paziņojums. Ar funkciju "Hide" ir iespējams paslēpt konkrētos testu grafikus, nepieciešamības gadījuma ar funkciju "Show" tos atkal attēlojot. Ja lietotājam vairs nav nepieciešams redzēt kādu no atvērtajiem testu grafikiem, tos var dzēst ar opciju "Delete"

1.3.3.1. Testa datu vizualizācijas funkcija

Identifikators V.DataVisual
Apraksts
Funkcija paredzēta, lai pārbaudītu vai visas vērtības, kuras nepieciešamas konkrētā testa attēlošanai ir pieejamas.
Ievads
Obligātie parametri: • test run; • job; • build • name; • date; • roles; • role; • actor; • os; • device; • version; • sdk name; • sdk; • performance; • command name; • measurement time • cpu usage;

• mem usage; Neobligātie parametri: nav
Apstrāde
Pārbauda: • vai visas vērtības kuras tika definētas filtrā eksistē;
Izvade
Ja visas izvēlētās vērības eksistē pēc “Show” opcijas izvēles, tiek attēlots konkrētais grafiks, ja kāda no vērtībām neeksistē konsolē tiek izvadīts 1. paziņojums.
Paziņojumi
1. “Cannot read property of undefined”

1.3.3.2. “Show” pogas funkcija

Identifikators V.ShowData
Apraksts
Funkcija paredzēta, lai opcija “Show” varētu attēlot testu datus, kā arī nekorekti aizpildīta filtra gadījumā izvadītu kļūdas paziņojumu
Ievads
Obligātie parametri: • “Job name”; • “Platform”; • “Platform version”; • “SDK name”; • “SDK version”; • “Device”; • “Build number”; • “Test name”; Neobligātie parametri: nav
Apstrāde
Pārbauda: • vai ir pirms grafika attēlošanas ir izpildīti visi priekšnosacījumi- korekti aizpildītas filtra vērtības
Izvade
Ja visi filtra lauki ir aizpildīti, nospiežot pogu “Show” tiek attēlots konkrētais grafiks, ja filtra lauki nav aizpildīti vai ir daļēji aizpildīt, nospiežot pogu “Show” tiek izvadīts 1. paziņojums.

Paziņojumi
1. “No valid filter option given”

1.3.3.3. “Hide” pogas funkcija

Identifikators V.HideData
Apraksts
Funkcija paredzēta, lai paslēptu nevēlamos testu grafikus
Ievads
Obligātie parametri: • “Job name”; • “Platform”; • “Platform version”; • “SDK name”; • “SDK version”; • “Device”; • “Build number”; • “Test name”; Neobligātie parametri: nav
Apstrāde
Pārbauda: • vai ir atvērts vismaz viens testa grafiks
Izvade
Pēc opcijas “Hide” izvēles, konkrētā testa grafiks tiek paslēpts
Paziņojumi
Nav

1.3.3.4. “Delete” pogas funkcija

Identifikators V.DeleteData
Apraksts
Funkcija paredzēta, lai dzēstu nevēlamos testu grafikus
Ievads

Obligātie parametri: • “Job name”; • “Platform”; • “Platform version”; • “SDK name”; • “SDK version”; • “Device”; • “Build number”; • “Test name”; Neobligātie parametri: nav
Apstrāde
Pārbauda: • vai ir atvērts vismaz viens testa grafiks
Izvade
Pēc opcijas “Delete” izvēles, konkrētā testa grafiks tiek dzēsts
Paziņojumi
Nav

1.3.3.5. Testa informācijas attēlošanas funkcija

Identifikators F.ShowInformation
Apraksts
Funkcija paredzēta, lai attēlotu informāciju par attiecīgo testu virs testa grafika
Ievads
Obligātie parametri: • test run; • job; • build • name; • date; • roles; • role; • actor; • os; • device; • version; • sdk name; • sdk;

• performance; • command name; • measurement time • cpu usage; • mem usage; Neobligātie parametri: nav
Apstrāde
Pārbauda: • vai atlasītie grafika parametri ir satopami;
Izvade
Ja visas izvēlētās vērības eksistē tās tiek attēlotas virs grafika, lai sniegtu informāciju par konkrēto grafiku.
Paziņojumi
Nav

1.3.3.6. Testa komandu nosaukuma attēlošanas funkcija

Identifikators F.ShowCommandName
Apraksts
Funkcija paredzēta, lai attēlotu konkrētās testa komandas nosaukumu konkrētā laika posmā
Ievads
Obligātie parametri: • test run; • job; • build • name; • date; • roles; • role; • actor; • os; • device; • version; • sdk name; • sdk; • performance; • command name;

• measurement time • cpu usage; • mem usage; Neobligātie parametri: nav
Apstrāde
Pārbauda: • vai konkrētajā laika posmā ir notikusi kaut viena testa komanda
Izvade
Ja testa komanda ir notikusi to attēlo pretī konkrētajam laikam, kurā komanda notika
Paziņojumi
Nav

1.3.3.7. Jaunas testa komandas sākuma apzīmēšanas funkcija

Identifikators F.NewCommandStarted
Apraksts
Funkcija paredzēta, lai vizuāli attēlotu jaunas komandas sākšanos konkrētā laika posmā
Ievads
Obligātie parametri: • test run; • job; • build • name; • date; • roles; • role; • actor; • os; • device; • version; • sdk name; • sdk; • performance; • command name; • measurement time • cpu usage; • mem usage;

Neobligātie parametri: nav
Apstrāde
Pārbauda: • vai konkrētajā laika posmā ir notikusi testa komandas maiņa
Izvade
Ja testa komandas maiņa ir notikusi, pretī konkrētajam laikam tiek attēlota līnija, kura vizuāli atdala vienu komandu no otras.
Paziņojumi
Nav

1.4. Nefunkcionālās prasības

Tabula 1.4.1. Nefunkcionālo prasību saraksts

Prasības numurs	**Prasības apraksts**
NP.1	Pieejamība - sistēmai jābūt pieejamai visu diennakti, taču, ja ir nepieciešami sistēmas uzlabojumi vai labojumi, sistēmu var atslēgt konkrētā diennakts laikā, kad tā tiek izmantota vismazāk (plkst. 03:00 - plkst. 05:00).
NP.2	Kļūdu pārvaldība – ekspluatācijas laikā jāizmanto kāda no kļūdu pārvaldības sistēmām (Redmine, Jira u.c.), lai efektīvāk un korektāk reģistrētu, apstrādātu un novērstu sistēmas kļūdas.
NP.3	Atkopjamība - atjaunošana notiek, izmantojot iepriekš izveidotās rezerves kopijas.
NP.4	Izmantojamība mērķauditorijai - sistēmai ir intuitīva saskarne, kā rezultātā lietotājiem nevajadzētu rasties grūtībām to izmantojot.
NP.5	Dokumentācija - brīdī, kad programmatūra tiek piegādāta klientam, jābūt pieejamai pilnai sistēmas dokumentācijai.
NP.6	Uzturamība – sistēmas pirmkodam ir jābūt labi un skaidri komentētam (ar atsaucēm uz dokumentā esošajiem attiecīgajiem funkciju punktiem), jābūt iespējai uzglabāt vecākas sistēmas versijas (vismaz 3 versijas pirms pēdējās aktuālās versijas). Nepieciešamības gadījumā būtu iespējams to papildināt ar jaunu funkcionalitāti vai izlabot esošo, minimāli mainot pirmkodu.
NP.7	Pārlūkprogrammu savienojamība - sistēmai jābūt pieejamai vismaz uz četrām populārākajām pārlūkprogrammām: Google

	Chrome, Mozilla Firefox, Safari, Opera.
NP.8	Atbalsts – klientam nepieciešams atbalsts sistēmas uzstādīšanai, konfigurēšanai, uzturēšanai un kļūdu labošanai.
NP.9	Drošība - pašlaik rīks tiek lietots lokāli, un tas ir pieejams konkrētā projekta ietvaros. Nedrīkst pieļaut to, ka, pamainot saites (url) parametrus, var nesankcionēti piekļūt programmas datiem vai tos modificēt. Sistēmai jābūt drošai pret XSS, CSRF drošības problēmām, kā arī jāfiltrē lietotāju ievadītie dati, lai lietotājs nevarētu palaist skriptu ar ievades lauka palīdzību.
NP.10	Datu aizsardzība - testu dati nedrīkst būt brīvi pieejami vai tikt nodoti trešajai pusei. Sistēma ir paredzēta lietošanai tikai konkrētā projekta ietvaros un visa iegūtā informācija var tikt izmantota tikai šī projekta ietvaros.

2. PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS

2.1. Ievads

2.1.1. Nolūks

Šī programmatūras projektējuma apraksta (PPA) nolūks ir noteikt un apkopot visas prasības, pēc kurām tiks projektētas un realizētas "Veiktspējas datu analīzes rīka" funkcijas.

Šis dokuments ir paredzēts dokumentā aprakstītās sistēmas izstrādātājiem.

2.1.2. Darbības sfēra

Izstrādājamais produkts ir "Veiktspējas datu analīzes rīks, kurš, piekļūstot ārējam serverim, kurā tiek glabāti testēšanas laikā iegūtie dati, "Veiktspējas datu analīzes rīks" ļauj efektīvi vizualizēt un attēlot centrālā procesora (CPU) un brīvpiekļuves atmiņas (RAM) lietojamības testu rezultātus, caur viegli uztveramu lietotāja interfeisu (UI), atvieglojot tālāko datu analīzes procesu.

Tā funkcijas ietver iegūto veiktspējas testu datu attēlošanu grafikos un salīdzināšanu, atlasot tos pēc noteiktiem kritērijiem – "job" nosaukuma un būvējuma nosaukuma vai platformas nosaukuma, ierīces nosaukuma, platformas versijas, SDK nosaukuma un SDK versijas, testa nosaukuma. Rīks ļauj salīdzināt gan konkrētas testu kopas vairākus testus, gan dažādu testu kopu vēlamos testus.

Produkta mērķis ir atvieglot testēšanas rezultāta iegūto veiktspējas datu pārskatāmību, uztveramību un testu rezultātu salīdzināmību pēc noteiktiem kritērijiem.

Rīks ir paredzēts izmantošanai konkrētā projekta ietvaros informācijas tehnoloģiju nozarē.

2.1.3. Saistība ar citiem dokumentiem

Dokuments tika izstrādāts un noformēts, ievērojot standarta LVS 72:1996 "Ieteicamā prakse programmatūras projektējuma aprakstīšanai" [3] prasības un rekomendācijas.

2.1.4. Pārskats

Dokuments sastāv no 4 nodaļām:

1. Ievads. Nodaļā tiek aprakstīts dokumenta nolūks, produkta darbības sfēra, dokumenta struktūra, kā arī saistība ar citiem dokumentiem.

2. Dekompozīcijas apraksts. Nodaļā tiek aprakstīti rīka moduļi, to nolūks un funkcijas, kā arī datu dekompozīcijas apraksts, kurā tiek pieminētas rīkā izmantojamās kolekcijas un to entītiju struktūra.
3. Moduļu atkarības. Nodaļā tiek attēlotas moduļu savstarpējās atkarības, izmantojot datu plūsmu diagrammas.
4. Daļējs funkciju un lietotāja saskarņu projektējums. Nodaļa satur lietošanas diagrammu, aktivitāšu diagrammu, kā arī lietotāja saskarņu projektējumus, kuri ļauj labāk izprast rīka darbību.

2.2. Dekompozīcijas apraksts

2.2.1. Moduļu dekompozīcija

Nodaļā tiek aprakstīts sistēmas sadalījums moduļos, katra moduļa nolūks un programmatūras prasību specifikācijā definēto prasību realizācija katrā modulī. Atkarības starp moduļiem ir apskatāmas nodaļā 2.3.

2.2.1.1. Ienākošo datu apstrādes modulis

Moduļa nolūks
Nodrošināt ienākošo datu korektu apstrādi, definējot kādi ienākošie dati būs nepieciešami rīka tālākajai darbībai.
Moduļa funkcijas
1. Ienākošo datu apstrāde (P.DataProcess) 2. Ienākošo parametru vērtību apstrāde (P.DataValueProcess)

2.2.1.1. Datu filtrēšanas modulis

Moduļa nolūks
Nodrošināt attiecīgo filtru datu korektu apstrādi - pakāpenisku datu filtrēšanu izejot no tā kāda vērtība tika atlasīta pirms tam un automātiski pielāgojot pārējās izvēles opcijas.
Moduļa funkcijas

1. Filtru vērtību apstrāde (F.DataProcessing)
2. Filtru datu vērtību apstrāde (F.DataValueCheck)
3. Pakāpeniskā datu filtrēšana (F.DataGradualCheck)
4. Filtru datu attēlošanas funkcija (F.DataShowCheck)
5. Filtru bloka pievienošanas funkcija (F.AddFilerBlock)
6. Filtru bloka dzēšanas funkcija (F.DeleteFilerBlock)

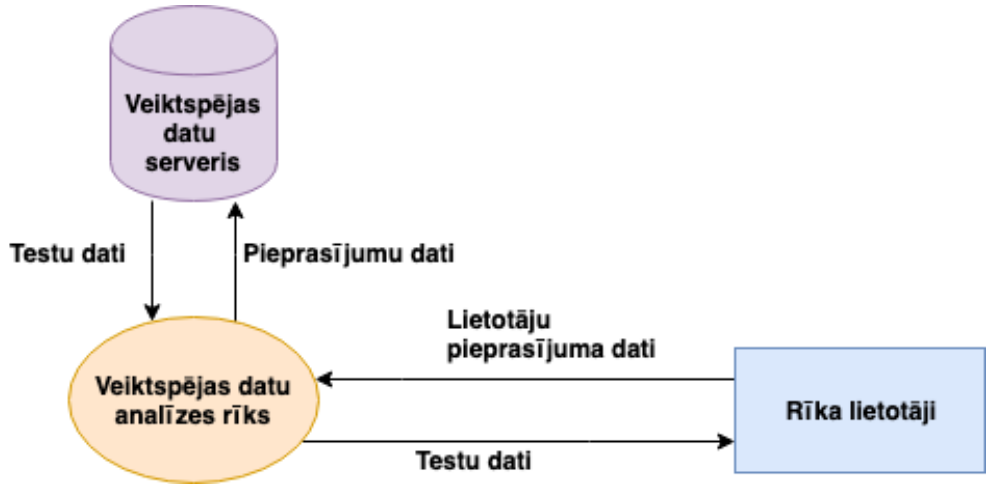
2.2.1.1. Datu vizualizācijas modulis

Moduļa nolūks
Nodrošināt korektu filtru datu attēlošanu - vizualizējot atlasītos testu datus un attēlojot tos grafikā.
Moduļa funkcijas
1. Testa datu vizualizācija (V.DataVisual)
2. “Show” funkcija (V.ShowData)
3. “Hide” pogas funkcija (V.HideData)
4. “Delete” pogas funkcija (V.DeleteData)
5. Testa informācijas attēlošanas (F.ShowInformation)
6. Testa komandu nosaukuma attēlošana (F.ShowCommandName)
7. Jaunas testa komandas sākuma apzīmēšana (F.NewCommandStarted)

2.3. Moduļu atkarības

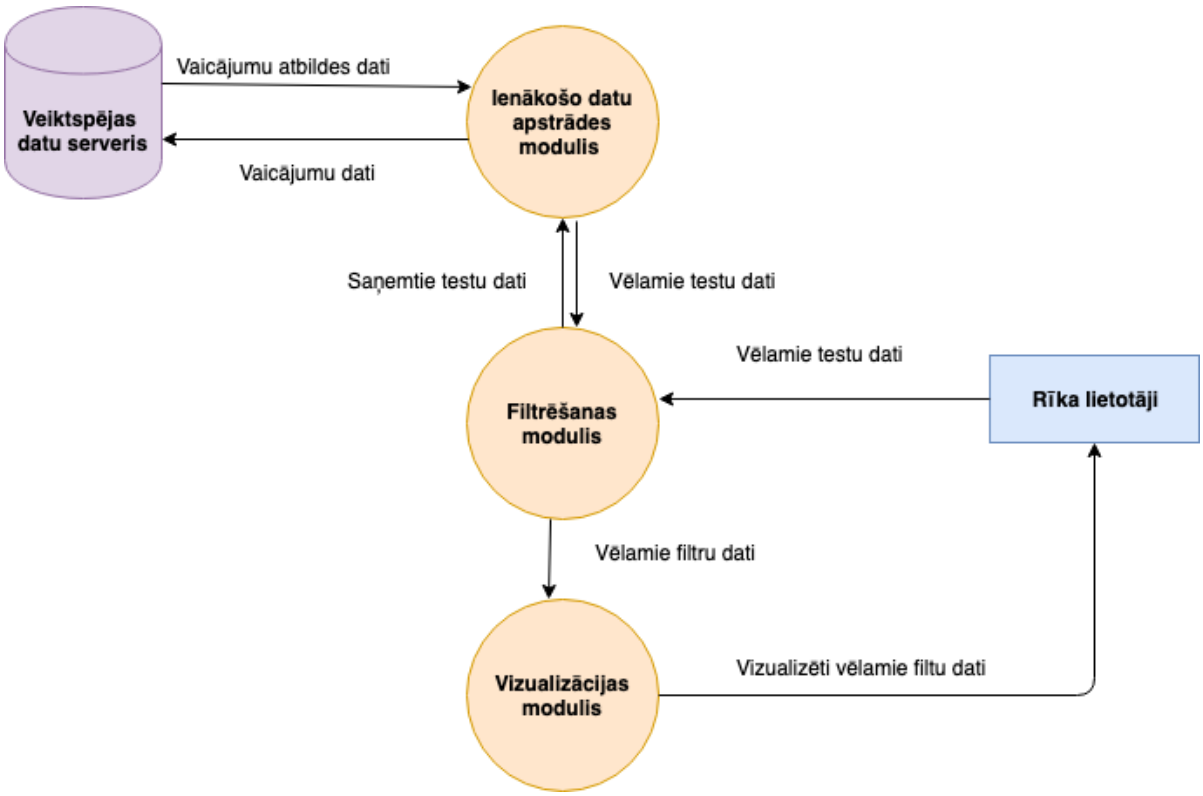
Atkarības starp moduļiem, kā arī rīka saistība ar lietotājiem tiek attēlota, izmantojot datu plūsmu diagrammas.

2.3.1. 0. līmeņa datu plūsmas diagramma



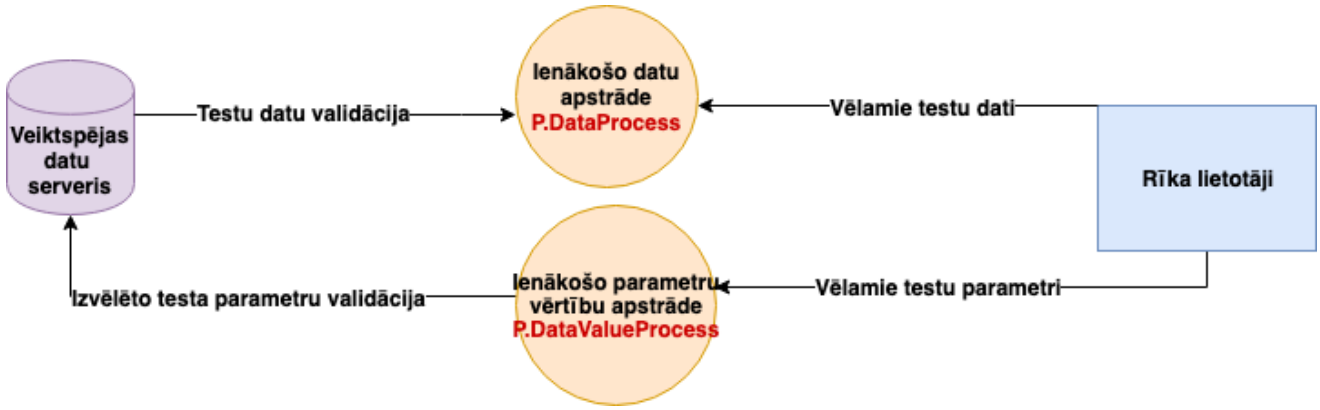
Att. 2.1. **0. līmeņa datu plūsmu diagramma**

2.3.2. 1. līmeņa datu plūsmas diagramma



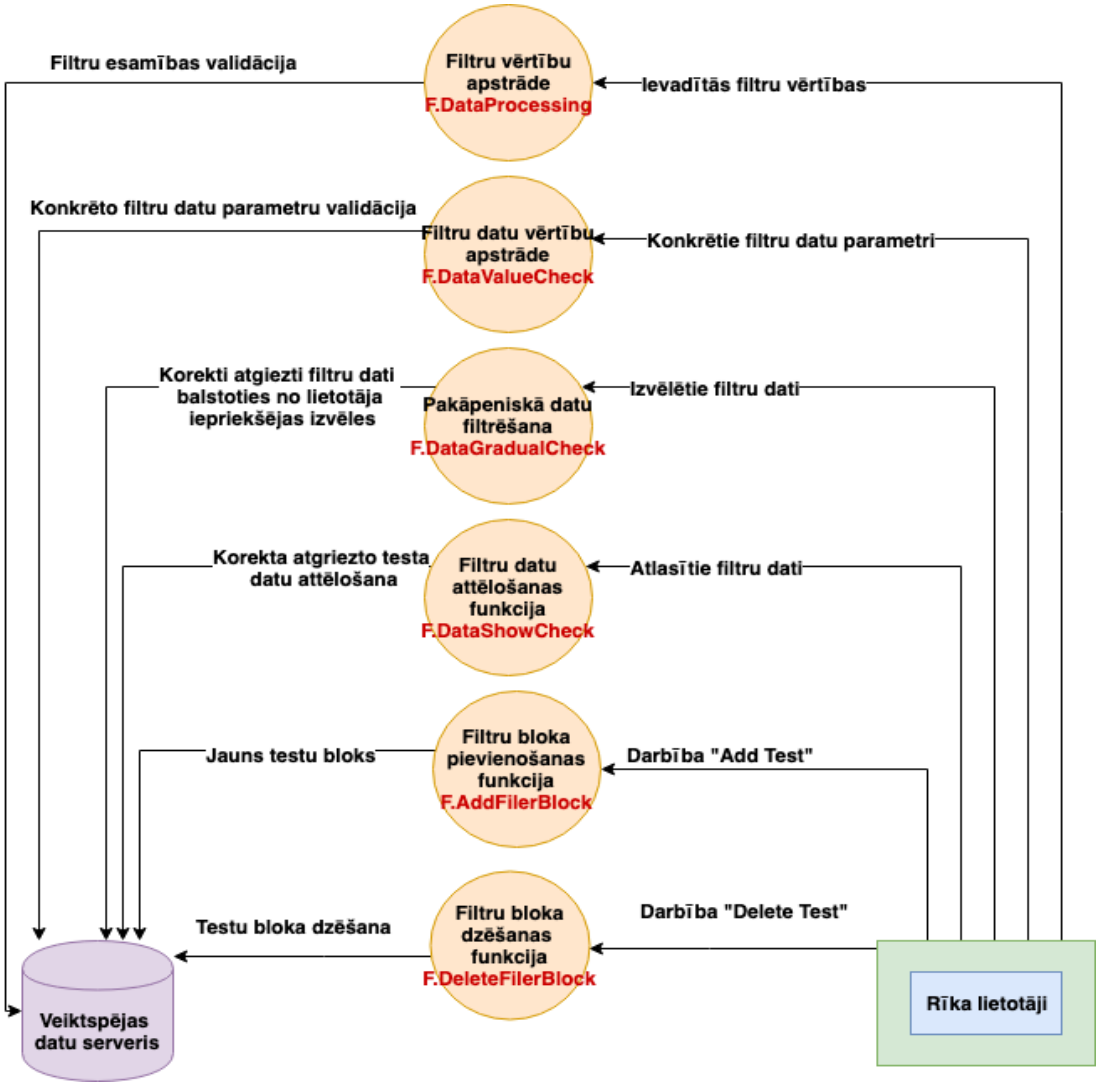
Att. 2.2. **1. līmeņa datu plūsmu diagramma**

2.3.3. 2. līmeņa ienākošo datu apstrādes moduļa datu plūsmu diagramma



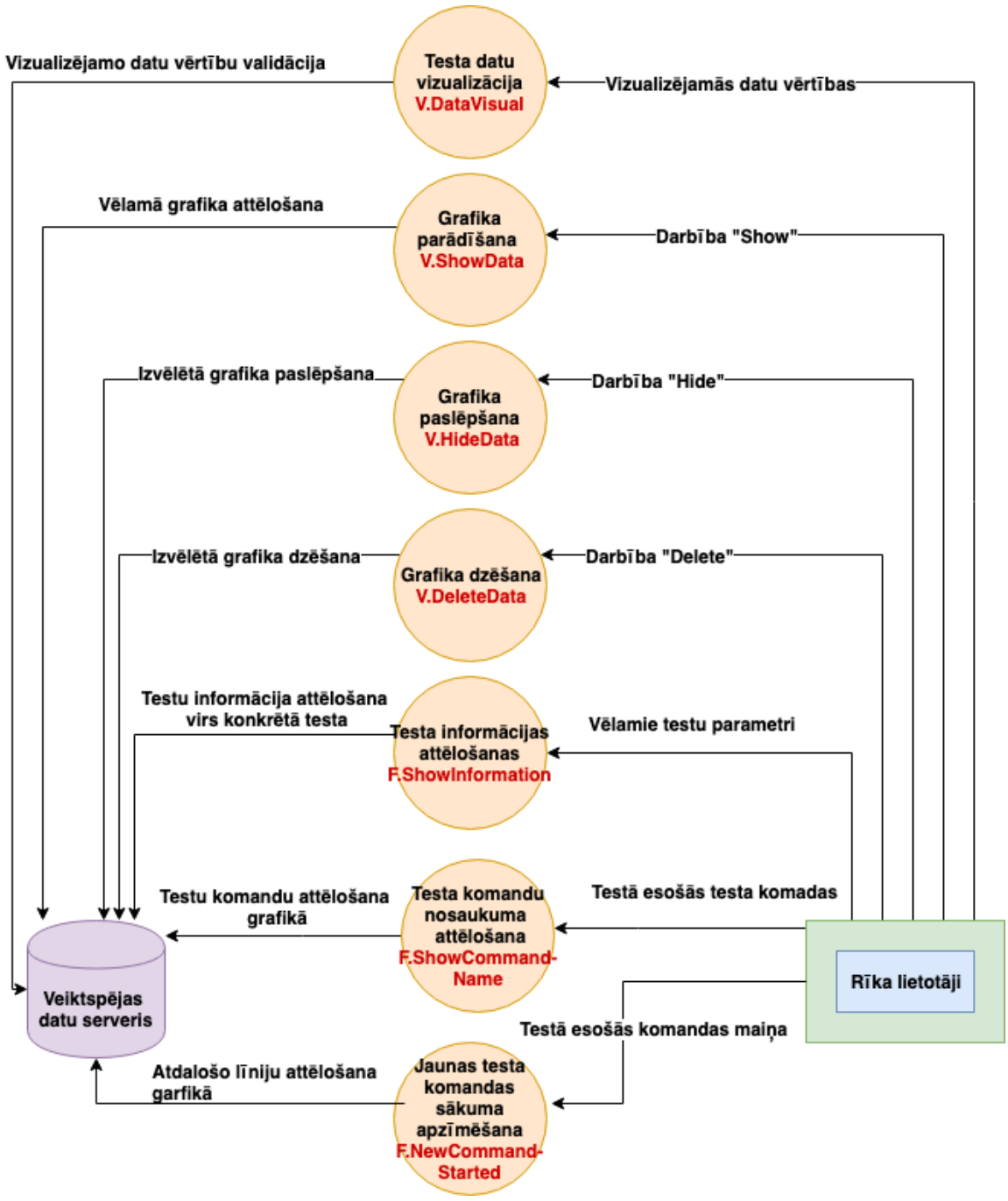
Att. 2.3. **2. līmeņa ienākošo datu apstrādes moduļa datu plūsmu diagramma**

2.3.4. 2. līmeņa datu filtrēšanas moduļa datu plūsmu diagramma



Att. 2.4. **2. līmeņa datu filtrēšanas moduļa datu plūsmu diagramma**

2.3.5. 2. līmeņa datu vizualizācijas moduļa datu plūsmu diagramma



Att. 2.5. **2. līmeņa līmeņa datu vizualizācijas moduļa datu plūsmu diagramma**

2.4. Daļējs funkciju un lietotāja saskarņu projektējums.

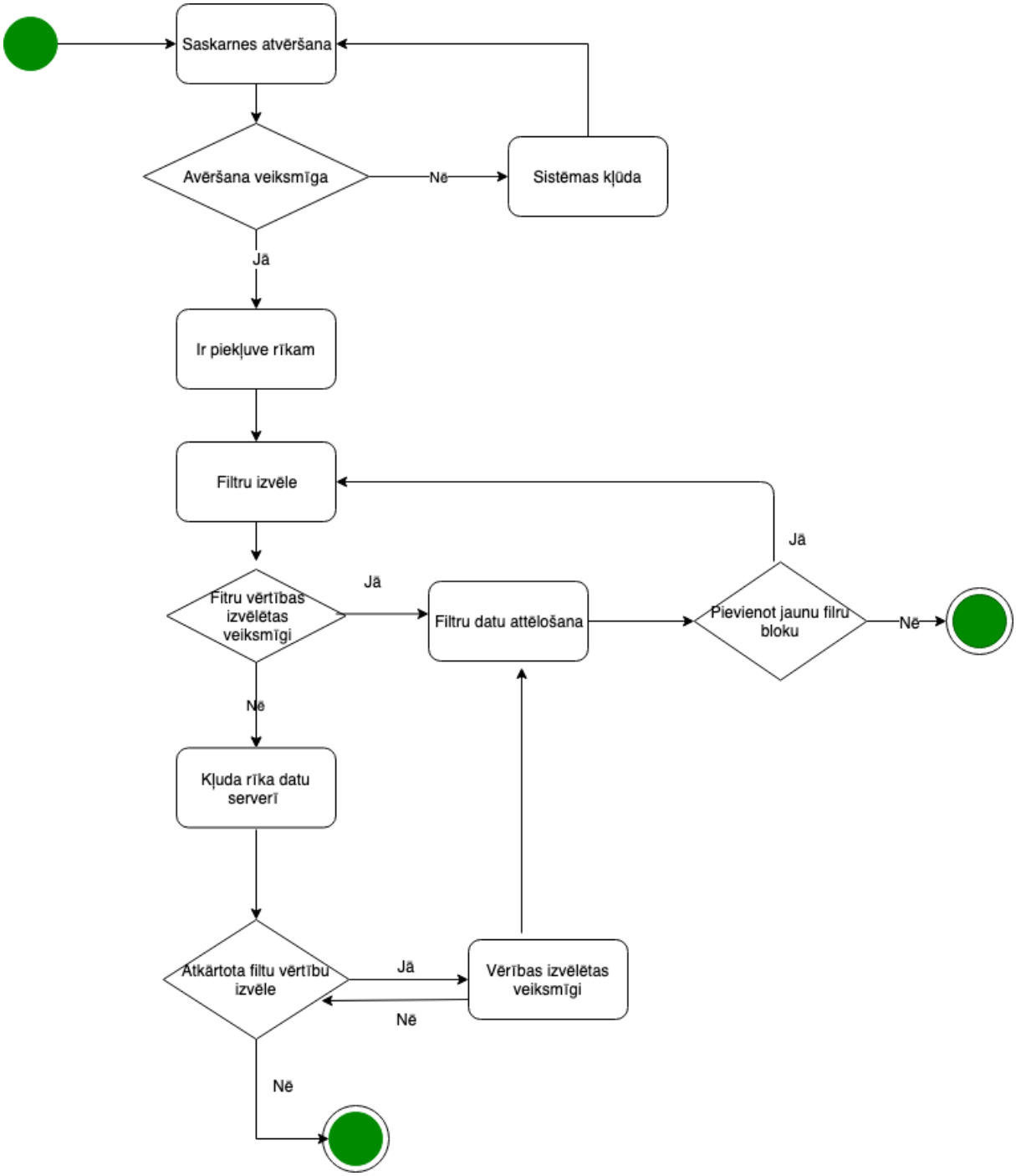
2.4.1. Lietošanas piemēru diagramma

Lai attēlotu "Veiktspējas datu analīzes rīka" lietotāju pamatdarbības, tika izveidota lietošanas diagramma. Tā attēlo kādas darbības ar rīku ir iespējams veikt tā lietotājiem.

Att. 2.6 **Lietošanas piemēru diagramma**

2.4.2. Aktivitāšu diagramma

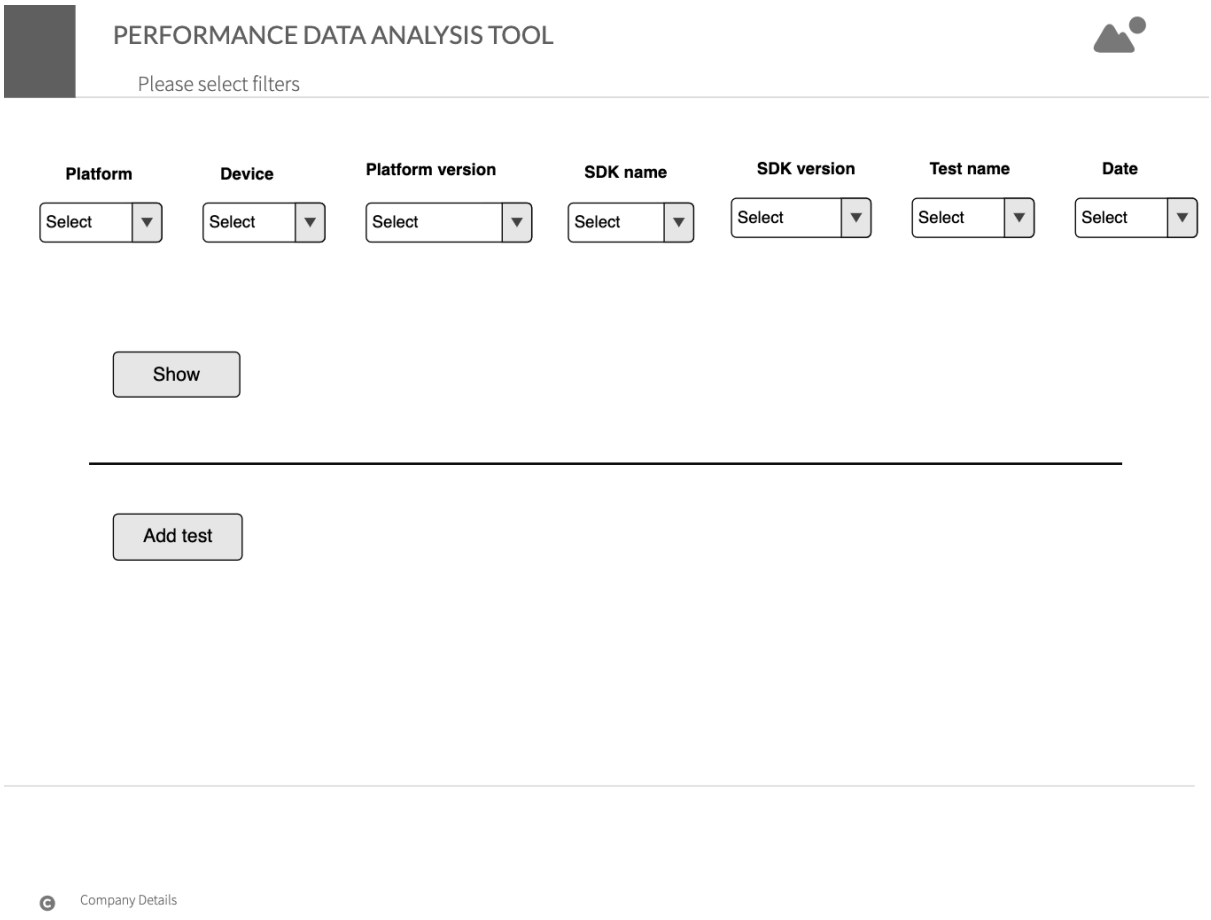
Lai attēlotu "Veiktspējas datu analīzes rīka" lietošanas soļu secību, tika izveidota aktivitāšu diagramma. Tā attēlo lietotāju aktivitāšu secību.



Att. 2.7 **Aktivitāšu diagramma**

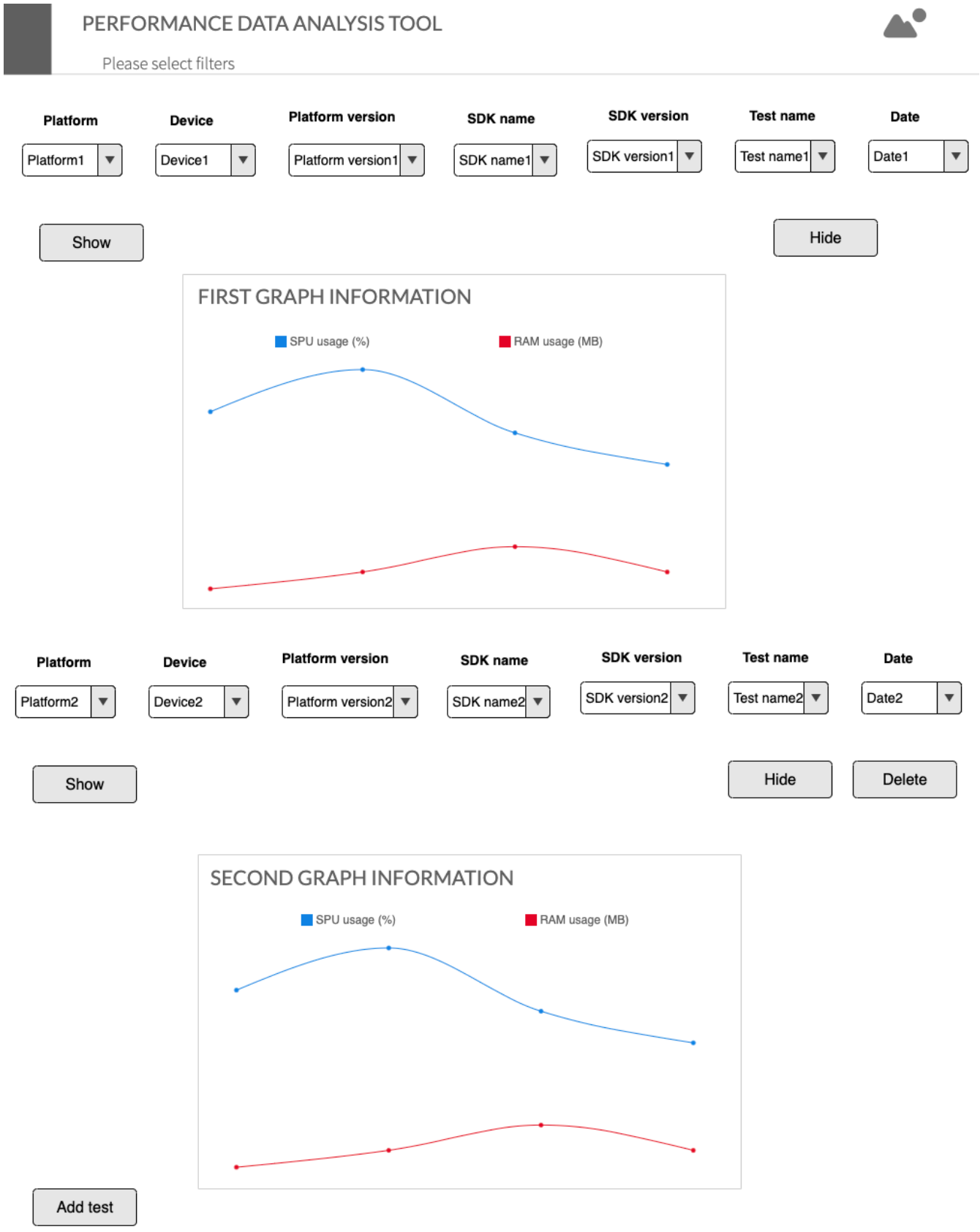
2.4.3 Daļējs lietotāja saskarņu projektējums

"Veiktspējas datu analīzes rīka" sākuma saskarnē ir redzamas septiņas filtru opcijas: "Platform", "Device", "Platform version", "SDK name", "SDK version", "Test name" un "Date" pēc kurām lietotājs var izvēlēties konkrētus testu datus, un nospiežot pogu "Show" lietotājs redzēs izvēlētā testa grafiku. Ja lietotājs vēlas salīdzināt vairāk kā vienu testa grafiku ar pogu "Add test" lietotājs var pievienot vairākus testa filtra blokus.



Att. 2.8 **"Veiktspējas datu analīzes rīka" sākuma saskarne**

"Veiktspējas datu analīzes rīka" grafiku salīdzināšanas saskarnē ir vizuāli redzami pēc konkrētiem parametriem atlasītie veiktspējas testu grafiki. Ar "Hide" pogas palīdzību iespējams paslēpt jebkuru no atvērtajiem testu grafikiem, vēlāk to atkal atverot nospiežot pogu "Show". Ja lietotājam kāds no atvērtajiem grafikiem vairs nav nepieciešams, tos var dzēst nospiežot pogu "Delete". Ja lietotājs vēlas pievienot vēl kādu testa grafiku to var izdarīt nospiežot pogu "Add test", un filtru sadaļā atlasot sev vēlamos parametrus.



Att. 2.8 **"Veiktspējas datu analīzes rīka" grafiku salīdzināšanas saskarne**

3. TESTĒŠANAS DOKUMENTĀCIJA

3.1. Testēšanas apraksts

"Veiktspējas datu analīzes rīka" testēšana tika veikta, izmantojot "melnās kastes" metodi, konkrētāk lietošanas piemēru testēšana, t.i., testējot vai lietotājs tiešām var veikt visas plānotās darbības ar konkrēto sistēmu. Lai pēc iespējas plašāk pārbaudītu visas sistēmas funkcijas, tikai pielietota uz pieredzi balstīta testēšanas tehnika- pētnieciskā testēšana. [4]

Lai pārbaudītu rīka funkcionalitāti un darbību, tika izmantotas šādas testēšanas metodes: datu filtrēšanas moduļa manuāla testēšana, datu vizualizācijas moduļa manuāla testēšana, izmantojot tīmekļa pārlūkprogrammas Google Chrome, Mozilla Firefox, Safari, Opera.

Kopš pirmās rīka funkcionalitātes ieviešanas rīks tiek testēts manuāli, lai pārliecinātos, ka tas funkcionē atbilstoši noteiktajām prasībām. Testi tika dokumentēti testu pārvaldības programmatūrā TestRail.

3.2. Testpiemēri

Liela testpiemēru skaita dēļ, šajā nodaļā tiks atspoguļoti tikai nozīmīgākie testpiemēri, kuri tika izmantoti rīka funkciju testēšanai.

3.2.1. Datu filtrēšanas un datu vizualizācijas moduļu testēšana

Testa nosaukums
"Show" pogas izvēle, pirms filtru izvēles
Priekšnosacījumi
Lietotājs ir atvēris Veiktspējas datu analīzes rīka" sākuma skatu.
Testa soļi
1. Lietotājs neizvēlas ne vienu no filtra opcijām. 2. Lietotājs klikšķina uz pogas "Show".
Sagaidāmais rezultāts
Lietotājs redz paziņojumu "No valid filter option given".
Esošais rezultāts

+
Testa nosaukums
Nepilnīga filtru izvēle
Priekšnosacījumi
Lietotājs ir atvēris Veiktspējas datu analīzes rīka” sākuma skatu.
Testa soļi
1. Lietotājs izvēlas tikai pirmo filtra opciju “Job name”. 2. Lietotājs klikšķina uz pogas “Show”. 3. Lietotājs atkārto testu pakāpeniski liekot klāt karu nākošo filtru līdz sadaļai “Build name”.
Sagaidāmais rezultāts
Lietotājs redz paziņojumu “No valid filter option given”.
Esošais rezultāts
+

Testa nosaukums
Pilnīga filtru lauku aizpildīšana
Priekšnosacījumi
Lietotājs ir atvēris Veiktspējas datu analīzes rīka” sākuma skatu.
Testa soļi
1. Lietotājs aizpilda visus filtra laukus sākot no lauka “Job name” līdz laukam “Test name” ar sev vēlamajiem parametriem. 2. Lietotājs klikšķina uz pogas “Show”.
Sagaidāmais rezultāts
Lietotājam tiek attēlots attiecīgais grafiks ar pieprasītajiem testa datiem.
Esošais rezultāts
+

	Testa nosaukums
	Testu lauku pārbaude pēc “Job name 1” parametra
	Priekšnosacījumi
	Lietotājs ir atvēris Veiktspējas datu analīzes rīka” sākuma skatu.
	Testa soļi
1. 2. 3. 4.	Lietotājs laukā “Job name” izvēlas 1. lauku “Job1”. Lietotājs laukā “Platform” var izvēlēties naākošo parametru “Javasript”. Lietotājs laukā “Platform version” var izvēlēties nākošo parametru “2.2.1.1.1.1”. Lietotājs laukā “SDK name” var izvēlēties nākošo parametru “Name1”.
	Sagaidāmais rezultāts
	Lietotājs no filtru saraksta var izvēlēties vēlamo “SDK name” parametru.
	Esošais rezultāts
	“SDK name” sarakstā neatrodas neviens parametrs. Komentārs- kļūda filtrēšanas funkcijā, kļūda tika novērsta.

	Testa nosaukums
	Ierīces atbilstība platformas versijai
	Priekšnosacījumi
	Lietotājs ir atvēris Veiktspējas datu analīzes rīka” sākuma skatu.
	Testa soļi
1. 2. 3. 4. 5. 6.	Lietotājs laukā “Job name” izvēlas 1. lauku “Job2”. Lietotājs laukā “Platform” var izvēlēties nākošo parametru “Android”. Lietotājs laukā “Platform version” var izvēlēties nākošo parametru “9”. Lietotājs laukā “SDK name” var izvēlēties nākošo parametru “Name2”. Lietotājs laukā “SDK version” var izvēlēties nākošo parametru “3.0.0-rc2”. Lietotājs laukā “Device” var izvēlēties nākošo parametru “Samsung S8”.
	Sagaidāmais rezultāts
	Lietotājs laukā “Build name” var izvēlēties nākošo parametru.
	Esošais rezultāts

Laukā “Build name” nav redzams neviens parametrs, kuru būtu iespējams izvēlēties. Komentārs- netika validēta ierīces atbilstība konkrētajai platformas versijai, ja vienā testā tiek lietotas dažādas ierīces ar dažādām operētājsistēmas versijām, kļūda tikt novērsta.

	Testa nosaukums
	Platformas atbilstība platformas versijai
	Priekšnosacījumi
	Lietotājs ir atvēris Veiktspējas datu analīzes rīka” sākuma skatu.
	Testa soļi
1. 2. 3. 4. 5.	Lietotājs laukā “Job name” izvēlas 1. lauku “Job2”. Lietotājs laukā “Platform” var izvēlēties nākošo parametru “iOS”. Lietotājs laukā “Platform version” var izvēlēties nākošo parametru “12.1”. Lietotājs laukā “SDK name” var izvēlēties nākošo parametru “Name1”. Lietotāji spēj korekti izvēlēties arī pārējos laukus.
	Sagaidāmais rezultāts
	Platformas versija atbilst konkrētajai platformai un grafiks tiek attēlots atbilstoši izvēlētajām vērtībībām.
	Esošais rezultāts
+	

	Testa nosaukums
	Grafika apraksts sakrīt ar ievadītajām filtru vērtībām
	Priekšnosacījumi
	Lietotājs ir atvēris Veiktspējas datu analīzes rīka” sākuma skatu.
	Testa soļi
1. 2. 3.	Lietotājs aizpilda visus filtra laukus sākot no lauka “Job name” līdz laukam “Test name” ar sev vēlamajiem parametriem. Lietotājs klikšķina uz pogos “Show”. Lietotājs redz vēlamo grafiku un konkrētā grafika informāciju virs tā.
	Sagaidāmais rezultāts

Informācija kāda tiek attēlota virs grafika sakrīt ar izvēlēto informāciju.
Esošais rezultāts
+

Testa nosaukums
Konkrētu grafika punktu vērtību attēlošana
Priekšnosacījumi
Lietotājs ir atvēris Veiktspējas datu analīzes rīka" sākuma skatu.
Testa soļi
1. Lietotājs aizpilda visus filtra laukus sākot no lauka "Job name" līdz laukam "Test name" ar sev vēlamajiem parametriem. 2. Lietotājs klikšķina uz pogas "Show". 3. Lietotājs redz vēlamo grafiku un konkrētas punktu vērtības, kuras tiek attēlotas grafikā. 4. Lietotājs, ar datorpeles palīdzību pārbauda konkrēto punktu vērtības.
Sagaidāmais rezultāts
Konkrētas grafika punktu vērtības tiek attēlotas korekti.
Esošais rezultāts
+

Testa nosaukums
Komandu atdalošās līnijas attēlošana
Priekšnosacījumi
Lietotājs ir atvēris Veiktspējas datu analīzes rīka" sākuma skatu.
Testa soļi
1. Lietotājs aizpilda visus filtra laukus sākot no lauka "Job name" līdz laukam "Test name" ar sev vēlamajiem parametriem. 2. Lietotājs klikšķina uz pogas "Show". 3. Lietotājam ir redzams grafiks ar līnijām, kuras iezīmē konkrēti kurā laika sākas jauna komanda.

	Sagaidāmais rezultāts
	Līnijas tiek attēlotas korekti- tieši tajā brīdī, sākas testa komanda, kā arī ir redzami komandu nosaukumi vertikālā stāvoklī.
	Esošais rezultāts
+	

	Testa nosaukums
	Vairāk kā divu testa lomu attēlošana
	Priekšnosacījumi
	Lietotājs ir atvēris Veiktspējas datu analīzes rīka" sākuma skatu.
	Testa soļi
1. 2.	Lietotājs izvēlas attēlot grafiku ar vairāk nekā divām testa lomām . Lietotājs klikšķina uz pogas "Show".
	Sagaidāmais rezultāts
	Lietotājs redz korektu lomu un testa vērtību attēlojumu.
	Esošais rezultāts
+	

	Testa nosaukums
	Testa bloka pievienošana
	Priekšnosacījumi
	1. Lietotājs ir atvēris Veiktspējas datu analīzes rīka" sākuma skatu, kurā ir redzami pieejamie filtri. 2. Lietotājs ir izvēlējies pirmo testu un redz korektu pirmā testa grafiku.
	Testa soļi
1. 2. 3.	Lietotājs klikšķina uz pogas "Add test" . Lietotājs redz jaunus filtra laukus ar iespēju izvēlēties otru testu. Lietotājs izvēlas otrā testa vērtības un nospiež pogu "Show".

	Sagaidāmais rezultāts
	Lietotājs korekti redz abu testu grafikus un to vērtības.
	Esošais rezultāts
+	

	Testa nosaukums
	Testa bloka dzēšana
	Priekšnosacījumi
	1. Lietotājs ir atvēris Veiktspējas datu analīzes rīka" sākuma skatu, kurā ir redzami pieejamie filtri. 2. Lietotājs ir izvēlējies pirmo testu un redz korektu pirmā testa grafiku.
	Testa soļi
1. 2.	Lietotājs pievieno jaunu testa bloku ar opciju "Add test". Neaizpildot testa bloku ar parametriem lietotājs izvēlas opciju "Delete".
	Sagaidāmais rezultāts
	Pievienotais testa bloks tiek dzēsts.
	Esošais rezultāts
+	

	Testa nosaukums
	Vairāk kā divu testu korekts attēlojums
	Priekšnosacījumi
	1. Lietotājs ir atvēris Veiktspējas datu analīzes rīka" sākuma skatu, kurā ir redzami pieejamie filtri. 2. Lietotājs ir izvēlējies pirmo testu un redz korektu pirmā testa grafiku.
	Testa soļi
1. 2.	Lietotājs klikšķina uz pogas "Add test" un izvēlas otrā grafika parametrus. Lietotājs ar pogu "Add test" pievieno sev vēlamo grafiku skaitu.

	Sagaidāmais rezultāts
	Lietotājs korekti redz visus testu grafikus un to vērtība.
	Esošais rezultāts
+	

	Testa nosaukums
	Izvēlētā grafika paslēpšana
	Priekšnosacījumi
	Lietotājs ir atvēris Veiktspējas datu analīzes rīka" sākuma skatu, kurā ir redzami pieejamie filtri.
	Testa soļi
1. 2. 3.	Lietotājs aizpilda visus filtra laukus sākot no lauka "Job name" līdz laukam "Test name" ar sev vēlamajiem parametriem. Lietotājs klikšķina uz pogas "Show". Lietotājs izvēlas paslēpt konkrēto testu izvēloties pogu "Hide".
	Sagaidāmais rezultāts
	Lietotājs vairs neredz konkrētā testa grafiku, bet ar opciju "Show" tas atkal kļūst redzams.
	Esošais rezultāts
+	

	Testa nosaukums
	Izvēlētā grafika dzēšana
	Priekšnosacījumi
1. 2.	Lietotājs ir atvēris Veiktspējas datu analīzes rīka" sākuma skatu, kurā ir redzami pieejamie filtri. Lietotājs ir izvēlējies pirmo testu un redz korektu pirmā testa grafiku.
	Testa soļi

1. 2.	Lietotājs pievieno vēl dažus testa grafikus ar opciju “Add test”. Lietotājs izvēlas dzēst pievienotos testa grafikus ar opciju “Delete”.
	Sagaidāmais rezultāts
	Lietotājs izvēlētie grafiki tiek dzēsti
	Esošais rezultāts
+	

3.3. Rezultātu kopsavilkums

Testēšanas laikā atrastās kļūdas tika novērstas, veicot vairākus uzlabojumus pirmkodā. Pēc kļūdu atrašanas un likvidēšanas testēšana tika veikta atkārtoti, lai pārliecinātos, ka ieviestais risinājums ir stabils un efektīvs un novērstās testu kļūdās vairs neatkārtojas. Pārsvarā kļūdas radās pieredzes trūkuma un neuzmanības dēļ.

4. PROJEKTA ORGANIZĀCIJA

Rīka izstrādē tika plānots pieturēties pie ūdenskrituma programmatūras izstrādes dzīves cikla modeļa. Sākot "Veiktspējas datu analīzes rīka" izstrādi, tika definētas prasības tā izstrādei, bet rīka izstrādes gaitā radās novirzes no izvēlētā modeļa, tā kā izstrādes procesa laikā tika definētas dažas jaunas prasības. Projekta sākumā tika izveidota rīka programmatūras prasību specifikācija un projektējuma apraksts.

Tā kā, sākot darbu pie projekta, autorei nebija pieredzes darbā ar JavaScript, kā arī īpaši šai darba specifikai domāto JavaScript bibliotēku Chart.js izstrāde aizņēma vairāk laika nekā tika plānots sākotnēji (sk. nodaļu 7). Sākumā tika nokonfigurēta izstrādei nepieciešamā integrētā izstrādes vide - uzinstalēti visi nepieciešamie satvari, JavaScript pakotņu pārvaldnieki, uzlikta un nokonfigurēta izstrādes vide, pievienotas JavaScript bibliotēkas. Tika apgūti JavaScript valodas pamati un iegūtas iemaņas darbā ar grafiku izstrādes bibliotēku Chart.js, kā arī iegūts priekšstats par kvalitātes nodrošināšanas nozares aktualitātēm un specifiku.

Rīka testēšana norisinājās vienlaikus ar izstrādi: funkcijas un metodes tika pārbaudītas uzreiz pēc to koda uzrakstīšanas, lai noskaidrotu, vai tās darbojas atbilstoši iepriekš definētajai prasību specifikācijai un vai ir nepieciešami to uzlabojumi. Dotā metode palīdzēja ātrāk un efektīvāk atkļūdot izstrādāto rīku. Izstrādes beigās rīks tika testēts vēlreiz, lai pārliecinātos vai iepriekš atrastās kļūdas neatkārtojas.

Projekta pamatprasību un rīka funkciju definēšana notika, konsultējoties ar darba vadītāju un kvalitātes nodrošināšanas speciālistiem, kuri vēlāk šo rīku izmantos. Produkta izstrāde, testēšana un dokumentēšana tika veikta patstāvīgi.

Projekta izstrādē tika izmantota WebStorm izstrādes vide, pirmkoda uzturēšanai tika izmantota versiju kontroles sistēma Git.

Rīka testēšanā tika izmantotas pārlūkprogrammas Google Chrome (73.0.3683.86 (64-bit)), Mozilla Firefox (67.0 (64-bit)), Safari (12.1) un Opera. (60.0.3255.83).

5. KVALITĀTES NODROŠINĀŠANA

Lai nodrošinātu izstrādājamā produkta kvalitāti, tika veikti šādi pasākumi:

- rīka prasību specifikācija un projektējuma apraksts tika noformēti atbilstoši valsts standartiem (sk. nodaļas 1.1.3. un 2.1.3.);
- rīka pirmkodā tika ievērots vienots programmēšanas stils;
- rīka pirmkods ir komentēts vietās, kur tas ir nepieciešams, kā arī iekļauts reģistrētājs, lai būtu vienkāršāk sekot līdzi procesu un funkciju izpildei un atkļūdot rīku;
- pirmkods tika strukturēts modulāri;
- izstrādājot jaunu funkcionalitāti, tika izveidoti jauni repozitorija zari, lai nekaitētu jau izstrādātai rīka funkcionalitātei. Zari tika sapludināti ar pamatzaru tikai tad, kad funkcionalitātes izstrāde bija pabeigta un kad rīks spēja iziet visus testpiemērus;
- pirms jaunās funkcionalitātes repozitorijas zaru sapludināšanas ar pamatzaru tika veikta koda apskate; sapludināšana notika tikai pēc projekta vadītāja apstiprinājuma.

6. KONFIGURĀCIJAS PĀRVALDĪBA

Izstrādājamā “Veiktspējas datu analīzes rīka” pirmkoda konfigurāciju pārvaldībai tika izmantots versiju kontroles rīks Git, un tā rīka repozitorijs tika glabāts Git repozitoriju pārvaldniekā GitLab.

Izstrādes procesā tika izmantots Git Flow zarošanas princips. Git repozitorijā tika uzturēts zars master, uz kura glabājās strādājoša un vadītāja akceptēta rīka versija. Ja bija nepieciešams veikt kādas izmaiņas, tika veidots jauns zars, kurā tiek veiktas konkrētās izmaiņas. Pirms zara nosūtīšanas uz GitLab repozitoriju, ir jāpārliecinās, ka rīka esošā funkcionalitāte strādā veiksmīgi.

Pēc izmaiņu veikšanas zars tiek saglabāts GitLab repozitorijā. Pirms jaunā zara sapludināšanas ar master zaru tiek veikta koda apskate. Sapludināšana tika veikta tikai tad, ja projekta vadītājs apstiprina izmaiņas.

7. DARBIETILPĪBAS NOVĒRTĒŠANA

Ņemot vērā to, ka autorei pirms rīka izstrādes nebija pieredzes darbā ar pielietotajām tehnoloģijām, bija salīdzinoši grūti precīzi novērtēt projekta darbietilpību, un tādējādi rīka izstrāde autorei aizņēma vairāk laika nekā tika plānots.

Darbietilpības novērtēšanai tika izmantota trīskāršā metode: projekta izstrādes posmi tiek novērtēti no pesimistiskā, reālistiskā un optimistiskā skatu punkta, un kopējā darbietilpība tiek aprēķināta, izmantojot formulu:

$$\frac{\mathrm{S(optimistiskais)} + 4\mathrm{S(reālistiskais)} + \mathrm{S(pesimistiskais)}}{6}$$

Par dienu tiek uzskatīta pilnās slodzes darbinieka darba diena - tās ilgums ir 8 stundas.

Tabula 7.1. **Darbietilpības novērtēšanas tabula**

	Pesimistiskais novērtējums (dienas)	Reālistiskais novērtējums (dienas)	Optimistiskais novērtējums (dienas)
Iepazīšanās ar projektā izmantojamām tehnoloģijām	15	12	7
Konsultācijas ar darba vadītāju	5	3	2
Programmatūras prasību specifikācijas noformēšana	7	5	3
Programmatūras projektējuma apraksta noformēšana	7	5	3
Ienākošo datu apstrādes modulis	4	3	1
Datu filtrēšanas modulis	20	15	7
Datu vizualizācijas modulis	15	10	7
Testpiemēru izveide	15	10	5

Testēšanas dokumentācijas noformēšana un rezultātu apkopošana	5	3	1
Kvalifikācijas darba noformēšana	6	2	1
Summā:	**99**	**68**	**34**

Darbietilpība = (99 + (83 * 4) + 34)/6 = 77,5 dienas jeb 3, 875 personmēneši.

Darbietilpības novērtējums pēc trīskāršās metodes ir vienāds ar 3, 875 personmēnešiem, tātad projekta apjoms atbilst kvalifikācijas darba izstrādes prasībām.

SECINĀJUMI

Izstrādājot darbu, tika izveidots funkcionējošs rīks, kurš atbilst prasību specifikācijai un projektējuma aprakstam. Tas tika izstrādāts ar mērķi atvieglot testēšanas rezultāta iegūto veiktspējas datu pārskatāmību, uztveramību un rezultātu salīdzināmību, konkrēta uzņēmuma projekta ietvaros. Rīks tiks aktīvi pielietots uzņēmuma ikdienā, optimizējot testu datu analīzes procesu. Pēc lietotāju ieteikumiem rīks nākotnē var tikt uzlabots, papildinot tā funkcionalitāti, kā arī nodrošinot plašāku testu datu veidu analīzi.

Rīka izstrādē tika apgūta JavaScript valoda, īpaši tās bibliotēka Chart.js, kura ļāva izmantot JavaScript valodu, lai vizualizētu vēlamos testu datus.

Tā kā autorei projektā izmantojamās tehnoloģijas pirms izstrādes uzsākšanas nebija pazīstamas, daudz laika tika patērēts to apgūšanai un dokumentācijas pētīšanai. Vislielākās grūtības sagādāja datu filtrēšanas un datu vizualizācijas moduļu izstrāde, jo uz tiem balstās izstrādātais rīks un to funkcionalitāte ir īpaši svarīga.

IZMANTOTIE AVOTI

[1] https://www.microsoft.com/en-us/language/Search?&searchTerm=build&langID=477&Source=true&productid=0I
(Pēdējo reizi skatīts: 27.05.2019)

[2] LVS 68:1996 Programmatūras prasību specifikācijas ceļvedis

[3] LVS 72:1996 Ieteicamā prakse programmatūras projektējuma aprakstīšanai

[4] ISTQB Standard Glossary of Terms used in Software Testing v2.2, prot. Nr. 459

1. pielikums. Rīka pirmkoda fragments

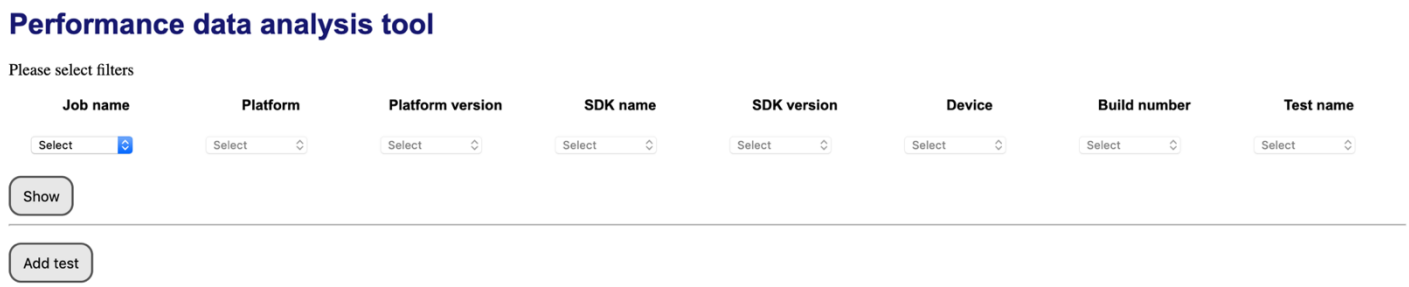
```
// Drawing graph from performance data

console.log(testrun);

function draw(performanceData, chartIndex) {
    Chart.defaults.LineWithLine = Chart.defaults.line;
    Chart.controllers.LineWithLine = Chart.controllers.line.extend({
        draw: function(ease) {
            Chart.controllers.line.prototype.draw.call(this, ease);

            if (this.chart.tooltip._active && this.chart.tooltip._active.length)
{
                var activePoint = this.chart.tooltip._active[0],
                    ctx = this.chart.ctx,
                    x = activePoint.tooltipPosition().x,
                    topY = this.chart.scales['y-axis-0'].top,
                    bottomY = this.chart.scales['y-axis-0'].bottom;

                // draw line
                ctx.save();
                ctx.beginPath();
                ctx.moveTo(x, topY);
                ctx.lineTo(x, bottomY);
                ctx.lineWidth = 2;
                ctx.strokeStyle = '#07C';
                ctx.stroke();
                ctx.restore();
            }
        }
    });
    console.log('Performance data in draw()');
    console.log(performanceData);

    const config = {  //Defeniton of a graph features
        type: 'line',
        data: {
            labels: performanceData.timestamp,

            datasets: [{
                label: 'CPU usage (%)',
                yAxisID: 'CPU usage',
                data: performanceData.cpu,
                borderWidth: '1.8',
                backgroundColor: 'rgba(54,162,235,0.2)',
                borderColor: 'rgba(54,162,235,1)',
                //fill: false,
                commandName: performanceData.commands

            },
                {
                    label: 'RAM usage (MB)',
                    yAxisID: 'RAM usage',
                    data: performanceData.mem,
                    borderWidth: '1.8',
                    backgroundColor: 'rgba(255,99,132,0.2)',
                    borderColor: 'rgba(255,99,132,1)',
                    //fill: false

                }]
        },
        //Define test command name context and placement in the graph
        options: {
```

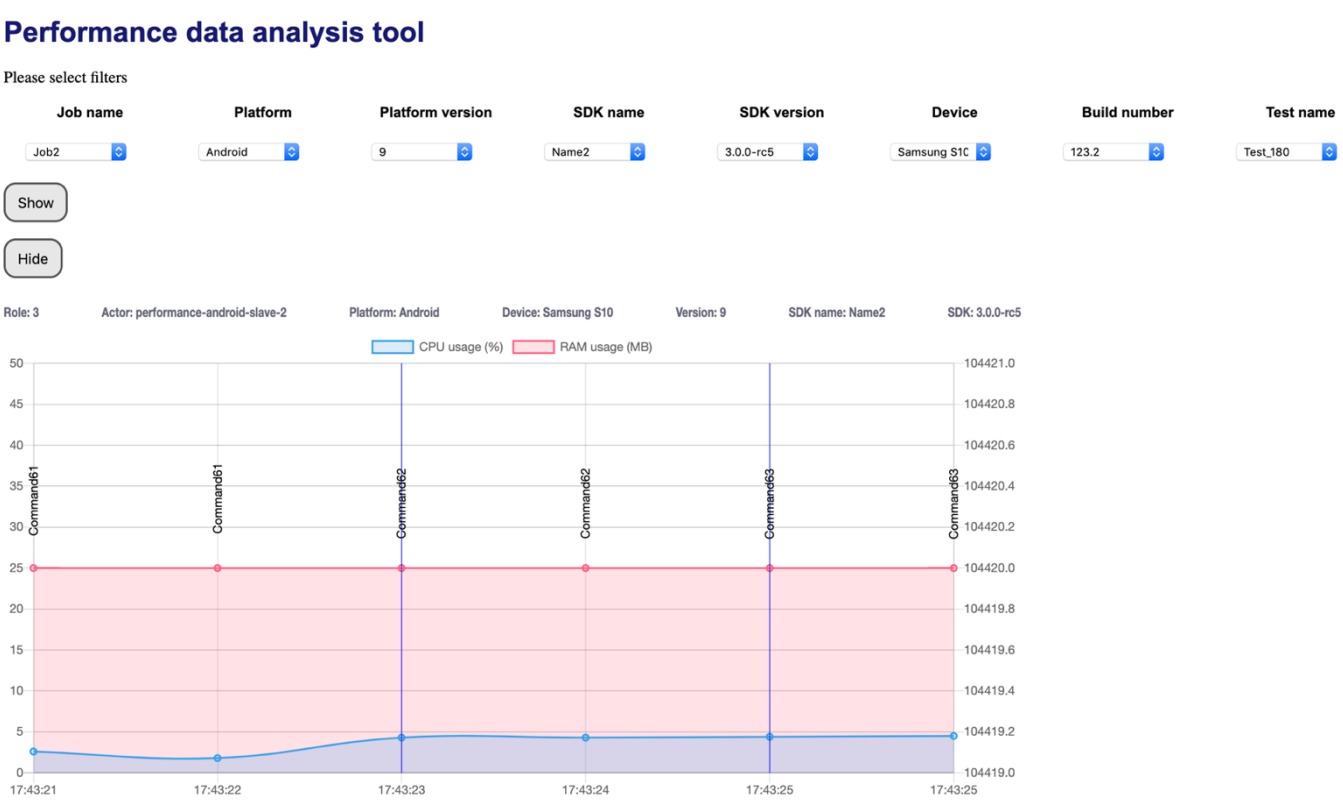
```
plugins: {
    datalabels: {
        formatter: function(value, context) {
            if (context.dataset.yAxisID === "CPU usage") {
                var index = context.dataIndex;
                // console.log(context);
                return context.dataset.commandName[index];
            } else {
                return '';
            }
        },

        offset: function(context) {
            if (context.dataset.yAxisID === "CPU usage") {
                let max = context.chart.scales['CPU usage'].max;
                let min = context.chart.scales['CPU usage'].min;
                let middle = (max + min) / 2;
                let index = context.dataIndex;
                let objectValue = context.dataset.data[index];
                let objectName = context.dataset.commandName[index];
                let objectLength = objectName.length;
                let offset = Math.abs(middle - objectValue);
                return offset *  10 - objectLength * 2.1
            }


        },

        align: function(context) {
            if (context.dataset.yAxisID === "CPU usage") {
                let max = context.chart.scales['CPU usage'].max;
                let min = context.chart.scales['CPU usage'].min;
                let middle = (max + min) / 2;
                let index = context.dataIndex;
                let objectValue = context.dataset.data[index];

                let offset = middle - objectValue;

                if (offset > 0) {
                    return 'top'
                } else if (offset < 0) {
                    return 'bottom'
                } else {
                    return 'center'
                }

            }
        },
        color: 'black',
        anchor: 'end',
        textStrokeWidth: 0,
        rotation: -90
    }
},
annotation: {
    annotations: performanceData.annotations
},
title: {
    display: true,
    fontSize: 13,
    fontColor:'#667',
    position: 'top',
    text:
        `Role: ${performanceData.roleNr}
        Actor: ${performanceData.actor}
        Platform: ${performanceData.os}
```

```
                    Device: ${performanceData.device}
                    Version: ${performanceData.version}
                    SDK name: ${performanceData.sdkName}
                    SDK: ${performanceData.sdk}`

            },
            scales: {
                yAxes: [{
                    id: 'RAM usage',
                    type: 'linear',
                    position: 'right',
                }, {
                    id: 'CPU usage',
                    type: 'linear',
                    position: 'left',
                    ticks: {
                        max: 50,
                        min: 0
                    }
                }]
            },
            tooltips: {
                callbacks: {
                    title: function(tooltipItem) { //Displaying the value of a
specific graph point
                        return `Command name:
${performanceData.commands[tooltipItem[0].index]}, Time:
${tooltipItem[0].xLabel}`;
                    }
                }
            }
        }
    };

    const holder = document.getElementById('chart-holder' + chartIndex);
    console.log('Fetched holder container location by id:', holder);
    const newChart = document.createElement('canvas');
    newChart.id = `chart${performanceData.roleNr}`;
    console.log('Create chart canvas element:', newChart);
    holder.append(newChart);

    const ctx = newChart.getContext("2d");
    const chart = new Chart(ctx, config);
}

async function parseRoleData(roleObj) { // Displaying the information about the
test role
    const perfData = {
        roleNr: null,
        actor: null,
        os: null,
        device: null,
        version: null,
        sdkName: null,
        sdk: null,
        commands: [],
        cpu: [],
        mem: [],
        timestamp: [],
        annotations: []
    };


    //Converts time from timestamp to hours minutes and seconds
    const parseTime = (timestamp) => {
        let date = new Date(timestamp);
```

```
        let hours = date.getHours();
        let minutes = date.getMinutes();
        let seconds = date.getSeconds();
        return `${hours}:${minutes}:${seconds}`;
    };

    console.log(roleObj);

    perfData.roleNr = roleObj.role;
    perfData.actor = roleObj.actor;
    perfData.os = roleObj.os;
    perfData.device = roleObj.device;
    perfData.version = roleObj.version;
    perfData.sdkName = roleObj.sdkName;
    perfData.sdk = roleObj.sdk;

    let previousCommand = null;

    // Displaying lines which divides the beginnig of the new command

    for(let command of roleObj.performance) {
        if(command.commandName !== previousCommand && previousCommand !== null)
{
            const annotation = {
                type: "line",
                mode: "vertical",
                scaleID: "x-axis-0",
                value: parseTime(command.data[0].measurementTime),
                borderColor: "blue",
                borderWidth: 0.7,

                label: {
                    content: `${command.commandName}`,
                    enabled: false,
                    position: "top",
                    yPadding: 0,
                    yAdjust: 9,
                    xAdjust: 9,



                }
            };
            perfData.annotations.push(annotation);
        }

        for(let metric of command.data) {
            perfData.commands.push(command.commandName);
            perfData.cpu.push(metric.cpuUsage);
            perfData.mem.push(metric.memUsage);
            perfData.timestamp.push(parseTime(metric.measurementTime));

        }
        previousCommand = command.commandName;
    }

    return perfData;
}
```

2. pielikums. Rīka lietotāju saskarnes





Kvalifikācijas darbs „***Veiktspējas datu analīzes rīks***” izstrādāts Latvijas Universitātes Datorikas fakultātē.

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

Autors: ***Ludmila Mačukāne*** _____ **___.05.2019**.

Rekomendēju darbu aizstāvēšanai
Darba vadītājs: ***M.dat Aleksandrs Dolmatovs*** _____ **___.05.2019.**

Recenzents: ***Uldis Karlovs-Karlovskis***

Darbs iesniegts 27.05.2019.
Kvalifikācijas darbu pārbaudījumu komisijas sekretāre: *Darja Solodovņikova*_____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē
___.06.2019. prot. Nr. _____

Komisijas sekretārs(-e): _____