LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

PAPLAŠINĀMA IOT PLATFORMA DATU UN NOTIKUMU APKOPOŠANAI

KVALIFIKĀCIJAS DARBS

Autors: Ralfs Sedlenieks

Studenta apliecības Nr.: rs15035

Darba vadītājs: M.dat. Toms Feldmanis

RĪGA 2017

ANOTĀCIJA

Kvalifikācijas darba mērķis ir izveidot paplašināmu platformu, kas ļautu apdrošinātājiem un citiem pakalpojumu sniedzējiem piedāvāt saviem klientiem uz IoT balstītus pakalpojumus, kā arī izmantot no dažādās atrašanās vietās izvietotiem sensoriem iegūtos datus klientu izprašanā un tendenču meklēšanā. Platformā iekļautais klienta bibliotēkas modulis atvieglo dažādu sensoru izmantošanu darbā ar platformu. Uzkrātos datus un notikumus var apskatīt apkopotā veidā tīmekļa vietnē. Sistēma tika veidota, lai tā būtu pēc iespējas vispārīgāka, tomēr darba ietvaros uzsvars tika likts uz sensoriem, ko lietot māju pārraudzībā. Izstrādāto sistēmu paredzēts lietot apdrošināšanas jomā.

Sistēma ir izstrādāta kvalifikācijas darba ietvaros, izmantojot Spring un Apache Camel satvarus.

Atslēgas vārdi: lietu internets, REST API, datu apkopošana, biznesa informācija, integrācija

ABSTRACT

EXTENSIBLE IOT PLATFORM FOR AGGREGATION OF DATA AND EVENTS

The purpose of this qualification work is to develop an extensible platform, which would enable insurers and other service providers to offer their clients services based on IoT. It would also allow them to use data obtained from sensors placed in different locations for customer insight and finding trends. The client library module included in the platform simplifies the use of different manufacturers' sensors for work with the platform. All of the accumulated data and events can be viewed in an aggregated form using a website. The system was developed with the goal of being as general as possible, however, in the scope of this work, focus was placed on sensors for home monitoring. The developed system is intended for use in the insurance sector.

The system was developed as part of a qualification work, using Apache Camel and Spring frameworks.

Keywords: Internet of Things, REST API, data aggregation, business intelligence, integration

SATURS

Apzīmējumu saraksts

- PPS – programmatūras prasību specifikācija
- PPA – programmatūras projektējuma apraksts
- NoSQL – datubāžu veids, kurā netiek izmantotas relācijas un shēmas
- MongoDB – atvērtā pirmkoda dokumentu datubāzes pārvaldības sistēma
- Docker - atvērtā pirmkoda rīks, kas ļauj automatizēt programmatūras uzstādīšanu, izmantojot virtualizāciju un programmatūras konteinerus
- Docker Compose - atvērtā pirmkoda rīks, kas ļauj koordinēt vairāku Docker konteineru darbību
- IoT (Internet of Things) – lietu internets
- HTTP (HyperText Transfer Protocol) – hiperteksta pārsūtīšanas protokols
- REST (Representational State Transfer) – tīmekļa pakalpojumu veids, kurā uzsvars likts uz pareizu HTTP darbību lietojumu atbilstoša veida operācijām
- API (Application Programming Interface) – programmsaskarne
- Elasticsearch – pilnteksta meklētājprogramma
- Kibana – datu vizualizācijas spraudnis Elasticsearch meklētājprogrammai
- API Man – rīks tīmekļa API pārvaldībai
- Throttling – pieprasījumu skaita ierobežošana
- DTO (Data Transfer Object) – datu pārsūtīšanas objekts
- DAO (Data Access Object) – datu pieejas objekts
- ID – identifikators (MongoDB datubāzē – 24 simbolu heksadecimāla virkne)
- IP adrese - skaitliska adrese, kas viennozīmīgi identificē katru datoru internetā un kas izveidota kā četru ar punktiem atdalītu skaitļu virkne

- Reverse proxy - reversais starpniekserveris
- EIP (Enterprise Integration Pattern) - uzņēmumu integrācijas modeļi
- Apache Camel - Java valodā rakstīts integrācijas satvars, balstīts uz EIP modeļiem
- JSON (JavaScript Object Notation) - uz tekstu balstīts datu apmaiņas formāts, kas ir lasāms arī cilvēkiem
- Salt - sāls, kriptogrāfijas kontekstā - nejauši dati, ko izmanto kā papildus ievaddatus jaucējfunkcijā
- Gradle - programmatūras projektu būvēšanas automatizācijas rīks
- Spring - lietotņu izstrādes un vadības inversijas satvars
- BSON (Binary JSON) - uz JSON balstīts datu apmaiņas formāts, kurā dati ir binārā formā. Tiek izmantots MongoDB datubāzē.

Ievads

Mūsdienās aizvien lielāku popularitāti gūst lietu internets jeb Internet of Things (IoT). Lietu interneta galvenais ieguvums ir tas, ka šādā sistēmā savienotas ierīces - sensori, sadzīves tehnika, kameras, aktuatori u.c. - var savā starpā apmainīties ar informāciju par savu apkārtni un stāvokli, kā arī tas, ka šādas sistēmas gandrīz vienmēr var pārvaldīt attālināti. Visplašāk lietu internets tiek pielietots "gudro māju" radīšanā, dažādos veidos padarot ikdienu ērtāku un efektīvāku. Šādi sensoru un iekārtu tīkli, kas spēj pārraudzīt situāciju mājā, ir noderīgi ne tikai pašiem mājas iemītniekiem, bet arī dažādu pakalpojumu sniedzējiem. Šajā darbā uzmanība tiks pievērsta apdrošināšanas jomai.

Pašreizējā stāvoklī apdrošināšanas jomā informācija par biežākajiem nekustamo īpašumu negadījumiem un ar tiem saistītajām apdrošināšanas atlīdzību izmaksām nav pieejama apkopotā veidā. To, par kādiem negadījumiem apdrošinātājs samaksā visvairāk un vislielākās apdrošināšanas atlīdzības, var uzzināt tikai no apdrošināšanas jomā strādājošo pieredzes. Ar šādu pieeju dati par negadījumiem tiek iegūti tikai pēc to iestāšanās un apdrošināšanas atlīdzības izmaksāšanas, kā arī netiek uzkrāti dati, par to, kādi apstākļi mājoklī palielina konkrētu negadījumu risku.

Projekta mērķis ir izveidot prototipu (proof of concept) platformai, kura atvieglotu integrāciju veidošanu ar dažādu ražotāju IoT ierīcēm un sistēmām, kā arī nodrošinātu no sensoriem iegūto datu un notikumu glabāšanu un vizualizāciju. Šī mērķa īstenošana ļautu apdrošinātājiem analizēt tendences un anomālijas apdrošināšanas atlīdzības pieprasījumu statistikā, kas, savukārt, radītu ieguvumu gan apdrošinātājiem (mazāk apdrošināšanas atlīdzības pieprasījumu), gan klientiem (zemākas apdrošināšanas prēmijas).

Lai sasniegtu minēto mērķi, tika izvirzīti šādi uzdevumi:

1. Apgūt sistēmas izstrādē lietojamo rīku (Docker, Gradle, Apache Camel, Spring) izmantošanu
2. Izveidot izstrādājamās sistēmas prasību specifikāciju
3. Balstoties uz izstrādāto sistēmas prasību specifikāciju, izstrādāt sistēmas projektējuma aprakstu
4. Izstrādāt sistēmu
5. Veikt izstrādātās sistēmas vienībtestēšanu un integrācijas testēšanu, sagatavot testēšanas dokumentāciju

Projekta sistēmas izstrāde notika pēc ūdenskrituma modeļa. Sākumā tika izstrādāts plāns sistēmas izstrādei un dokumentēšanai, bet tālāk pēc šī plāna tika izstrādāta sistēma un

tās dokumentācija. Izstrādātāja nelielās pieredzes, kā arī projektā lietoto jauno tehnoloģiju īpatsvara dēļ projekta gaitā izveidojās novirzes no sākotnējā plāna.

Darbs satur izstrādātās sistēmas programmatūras prasību specifikāciju, programmatūras projektējuma aprakstu, testēšanas dokumentāciju un izstrādātās sistēmas pirmkoda fragmentus, kā arī projekta organizācijas, kvalitātes nodrošināšanas, konfigurāciju pārvaldības un darbietilpības novērtējuma aprakstus.

Darbā iekļautā dokumentācija tika izstrādāta, balstoties uz attiecīgajiem valsts standartiem.[1][2] Darba noformēšanā un strukturēšanā tika izmantota Latvijas Universitātes dokumentā "Prasības noslēguma darbu izstrādāšanai un aizstāvēšanai Latvijas Universitātē" atrodamā informācija.

1. Programmatūras prasību specifikācija

1.1. Ievads

1.1.1. Dokumenta nolūks

Šī dokumenta nolūks ir specificēt visas prasības, kuras jārealizē sistēmas izstrādes ietvaros. Šo dokumentu izmantos kā pamatu programmatūras projektējuma izstrādei, produkta izstrādei, kā arī testēšanai. Dokuments paredzēts, lai saskaņotu produkta funkcionalitāti starp izpildītāju un pasūtītāju.

1.1.2. Darbības sfēra

Šajā dokumentā aprakstītās sistēmas mērķis ir pārvaldāmā un kontrolējamā veidā apkopot datus no dažādiem sensoriem un šos datus vizualizēt, lai ļautu labāk saskatīt tendences, anomālijas un problēmu cēloņus. Produkts pēc savas būtības varētu tikt izmantots dažādās sfērās, tomēr šī darba ietvaros uzsvars tiks likts uz apdrošināšanas sfēru. Produktu paredzēts piedāvāt apdrošināšanas organizācijām, lai ļautu tām apkopot datus no klientu mājās izvietotiem sensoriem, lai iegūtu klientu izpratni (customer insight), atrastu biežākās problēmas un negadījumu cēloņus, tādējādi uzlabojot situāciju gan klientiem, gan apdrošinātājiem. Apdrošinātāju ieguvums būtu mazāks negadījumu skaits, kas nozīmē mazākus izdevumus zaudējumu atlīdzināšanā, bet klientu ieguvums - mazākas apdrošināšanas prēmijas.

1.1.3. Saistība ar citiem dokumentiem

Dokuments tika veidots, balstoties uz standartu LVS 68:1996 “Programmatūras prasību specifikācijas ceļvedis”. Šajā dokumentā definētās prasības tiks izmantotas programmatūras projektējuma un testēšanas dokumentācijas izstrādē.

1.1.4. Pārskats

Dokumentā ir piecas nodaļas:

1. Ievads
   Ievadā tiek aprakstīts dokumenta mērķis un darbības sfēra.
2. Vispārējais apraksts
   Šajā nodaļā tiek īsumā aprakstītas produkta funkcijas, lietotāji un to grupas, kā arī dažādi ierobežojumi un pieņēmumi.
3. Funkcionālās prasības
   Šajā nodaļā tiek detalizēti aprakstītas produkta funkcijas, kā arī to ievaddati un darbība.
4. Nefunkcionālās prasības

Šajā nodaļā aprakstītas dažādas prasības, kas attiecas tieši uz produkta darbību, un kuras būtu jāņem vērā produkta projektēšanā un izstrādē.

5. Ārējā saskarne

   Šajā nodaļā aprakstītas sistēmas saskarnes ar lietotāju, kā arī ar citām sistēmām.

1.2 Vispārējais apraksts

1.2.1. Produkta perspektīva

Izstrādājamais produkts paredzēts kā rīks, kura izmantošanai klientiem būs pašiem jāizstrādā integrācija ar konkrētajā gadījumā izmantoto sensoru ražotāja API. Šī iemesla dēļ produkts veidots tā, lai atvieglotu šādu integrāciju veidošanu.

1.2.2. Produkta funkcijas

1. Vārtejas API modulis
   - Organizāciju ierakstu saglabāšana datubāzē
   - Visu organizāciju ierakstu iegūšana no datubāzes
   - Organizācijas atrašana datubāzē pēc ieraksta ID
   - Datubāzē saglabāto organizāciju ID iegūšana
   - Organizācijas ieraksta ID atrašana pēc organizācijas ID API pārvaldības modulī
   - Atrašanās vietu saglabāšana datubāzē
   - Atrašanās vietu iegūšana no datubāzes
   - Datubāzē saglabāto atrašanās vietu ID iegūšana
   - Ierīču saglabāšana datubāzē
   - Ierīču iegūšana no datubāzes
   - Datubāzē saglabāto ierīču ID iegūšana
   - Notikumu saglabāšana datubāzē
   - Notikumu iegūšana no datubāzes
   - Datubāzē saglabāto notikumu ID iegūšana
   - Jēldatu dokumenta saglabāšana datubāzē
   - Ierīču novietojuma klasifikatoru (tags) iegūšana no datubāzes
   - Ierīču novietojuma klasifikatoru (tags) importēšana datubāzē no JSON failiem
   - Datu validācija servera pusē
2. Starpprogrammatūras modulis
   - Datu pieejas objektu izveide

- Datu ielasīšana no datubāzes
- Datu ierakstīšana datubāzē

3. Kartēšanas modulis
    - Datu pārsūtīšanas objektu (DTO) pārveidošana starp sistēmas moduļiem
4. Datu vizualizācijas modulis
    - Datu vizualizāciju izveide
    - Datu vizualizāciju apkopošana informācijas paneļos
5. API pārvaldības modulis
    - Autentifikācija
    - Organizācijas izmantotā datu daudzuma mērīšana
    - Organizācijai pieejamā pieprasījumu skaita ierobežošana (throttling)
    - Lietotāja organizācijas ID iegūšana pēc lietotājvārda
    - Lietotāja reģistrācija
    - Aizmirstas paroles nomaiņa
    - Organizācijas reģistrācija API pārvaldības modulī
    - API reģistrācija API pārvaldības modulī
    - API lietojuma plānu izveidošana
    - API klienta lietotnes reģistrācija
6. Reversā starpniekservera modulis
    - Pieprasījumu nodošana attiecīgajam servisam platformā
7. Klienta bibliotēkas modulis
    - Klienta puses datu validācija
    - Vārtejas API moduļa REST servisu izsaukšana
8. Validācijas modulis
    - Datu pārsūtīšanas objektu validācija
9. REST API dokumentācijas modulis
    - REST servisu dokumentācijas ģenerēšana
    - REST servisu dokumentācijas attēlošana

1.2.3. Lietotāja raksturiezīmes

Kopumā var izdalīt 4 lietotāju grupas, kuras lietos sistēmu.

1. Izstrādātāji, kuri veidos integrācijas ar konkrētiem sensoriem. Tiek paredzēts, ka integrāciju izveidē tiks izmantota platformā iekļautā klienta bibliotēka priekš Apache Camel integrācijas satvara. Šai lietotāju grupai būs nepieciešamas programmēšanas prasmes un zināšanas par uzņēmumu integrācijas modeļiem (EIP).
2. API administratori, kuri izmantos API pārvaldības moduli, lai sekotu līdzi tam, cik daudz katrs klients (organizācija) izmanto platformu (datu apjomi, REST servisu pieprasījumu skaits utml.), kā arī lai uzstādītu attiecīgos ierobežojumus. Šiem lietotājiem būs jāpārzina API pārvaldības principi un rīki, piemēram, API Man.
3. Biznesa analītiķi, kuri izmantos datu vizualizācijas moduli, lai meklētu tendences un anomālijas sensoru dotajos datos. Šiem lietotājiem būs nepieciešamas prasmes darbā ar Kibana vai līdzīgiem datu vizualizācijas rīkiem.
4. Sistēmas uzturētāji, kuri īstenos platformas izvietošanu un uzturēšanu. Šiem cilvēkiem būs nepieciešamas zināšanas par dokumentu datubāžu pārvaldības sistēmām, piemēram, MongoDB, un Docker rīku, kā arī prasmes darbā ar UNIX/Linux sistēmām.

1.2.4. Vispārējie ierobežojumi

Sistēma tiks izstrādāta Java programmēšanas valodā (versija 8). Apstrādājamo datu rakstura dēļ tiks izmantota MongoDB dokumentu datubāze, nevis relāciju datubāze. Sistēmā sagaidāmo jēldatu apjoma dēļ nepieciešams izmantot Elasticsearch datubāzi, kā arī Kibana vizualizācijas spraudni. Lai sasniegtu vēlamo sistēmas veiktspēju, kā reverso starpniekserveri būtu jāizmanto Nginx.

1.2.5. Pieņēmumi un atkarības

Sistēmas darbībai nepieciešams interneta savienojums un sistēmas pieejamība atkarīga no sistēmas izvietošanai izvēlētā mākoņpakalpojumu sniedzēja serveru darbības. Sistēmas uzstādīšanai nepieciešams, lai vidē būtu Docker Toolbox instalācija, kā arī lai vides konfigurācija ļautu darbināt VirtualBox virtuālās mašīnas.

1.3. Funkcionālās prasības

Visām aprakstītajām funkcijām visi ievaddatu lauki ir obligāti, ja nav pateikts citādāk.

Visās funkcijās, izņemot funkciju GAM-SRDD, ar vārdu "datubāze" tiek saprasta platformai piesaistītā dokumentu datubāze. Funkcijas GAM-SRDD aprakstā tiek specifiski pateikts, par kuru datubāzi tiek runāts konkrētajā brīdī.

Tālāk aprakstīto funkciju kļūdu ziņojumi saturēs kļūdas kodu un kļūdas ziņojumu. Kļūdu kodi ļaus aptuveni noteikt, kāpēc radusies kļūda, bet kļūdu ziņojumi sniegs detalizētāku informāciju. Kļūdu kodu nozīmes atrodamas 1.1. tabulā.

1.1.tabula **Kļūdu kodi un to nozīmes**

Kļūdas kods	Kļūdas iemesls
1001	Datu validācijas kļūda
1002	Datubāzes savienojuma kļūda
1003	Klienta bibliotēkas kļūda
1004	Datu objektu kartēšanas kļūda
1005	Datu atlasīšanas kļūda
1006	JSON pārveidojumu kļūda
1007	Autentifikācijas kļūda
1008	Datu vizualizācijas kļūda
1009	API pārvaldības moduļa kļūda
1010	REST dokumentācijas moduļa kļūda

1.3.1. Vārtejas API modulis

1.3.1.1. Organizāciju ierakstu saglabāšana datubāzē

Identifikators	GAM-OCNO
Apraksts	Funkcija nepieciešama, lai saglabātu datubāzē informāciju par organizācijām, kuras lietos sistēmu, kā arī lai nodrošinātu atbilstību starp API pārvaldības modulī autentificētu lietotāju un sistēmā saglabātu organizāciju.
Ievade	Ievaddati tiks iegūti no klienta bibliotēkas moduļa ziņojuma pamatteksta (message body). Lauki: 1. Organizācijas nosaukums – simbolu virkne garumā no 1 līdz

	100 simboliem, drīkst saturēt punktus, atstarpes, domuzīmes, apostrofus, lielos un mazos burtus, ciparus. Pieļaujami arī burti ar diakritiskajām zīmēm 2. Organizācijas identifikators API pārvaldības modulī – patvaļīga simbolu virkne 3. Organizācijas fiziskā adrese – skatīt adrešu lauku nosacījumus 4. Organizācijas juridiskā adrese – skatīt adrešu lauku nosacījumus 5. Organizācijas kontakttālrunis – simbolu virkne, drīkst saturēt '+' un trīs ciparu valsts tālsarunu kodu, obligāti satur 8 ciparu tālruņa numuru. Virknē drīkst būt arī atstarpes starp noteiktām ciparu grupām. Kontakttālruņa piemērs, kurā atstarpes lietotas visās pieļaujamajās vietās: "+371 000 000 00" Abi adrešu lauki satur šādu informāciju: 1. Valsts – simbolu virkne garumā no 1 līdz 60 simboliem, drīkst saturēt domuzīmes, punktus, apostrofus, atstarpes un lielos un mazos burtus 2. Pilsēta – simbolu virkne garumā no 1 līdz 100 simboliem, drīkst saturēt punktus, atstarpes, domuzīmes, apostrofus, lielos un mazos burtus, ciparus, Pieļaujami arī burti ar diakritiskajām zīmēm 3. Ielas nosaukums – simbolu virkne garumā no 1 līdz 100 simboliem, drīkst saturēt punktus, atstarpes, domuzīmes, apostrofus, lielos un mazos burtus, ciparus, Pieļaujami arī burti ar diakritiskajām zīmēm 4. Mājas numurs – vesels skaitlis, kas lielāks par 0
Apstrāde	Ievaddati tiks padoti validācijas funkcijai (GAM-SSDV), kura pārbaudīs vai ievaddatos esošā informācija atbilst norādītajiem ierobežojumiem. Lauki tiek pārbaudīti tādā secībā, kādā tie norādīti šīs funkcijas ievaddatu aprakstā. Ja kāds lauks neatbilst ierobežojumiem, izejošajā ziņojumā tiek ierakstīts 1. kļūdu ziņojums.

	Kļūdu ziņojumā tiek iekļauts tā lauka nosaukums, pie kura radās validācijas kļūda. Ja vairāki lauki neatbilst nosacījumiem, tiek izvadīts nosaukums apstrādes secībā pirmajam kļūdainajam laukam. Ja visi lauki atbilst ierobežojumiem, tad datu objekts tiek padots funkcijai MDW-WDBD. Ja šai funkcijai rodas kāda kļūda, tiek izvadīts 2. vai 3. kļūdu ziņojums (skatīt funkciju MDW-WDBD). Ja kļūdu nav, tad tiek saņemts izveidotā datubāzes ieraksta ID.
Izvade	Izveidotā organizācijas ieraksta ID datubāzē. Ja apstrādes laikā rodas kāda kļūda, tiek izvadīts atbilstošais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1001 Message: Invalid <*lauka nosaukums*>!" 2. "Code: 1002 Message: Database connection error!" 3. “Code: 1004 Message: Source object must not be null!”

1.3.1.2. Visu organizāciju ierakstu iegūšana no datubāzes

Identifikators	GAM-OFAO
Apraksts	Funkcija ļauj no datubāzes iegūt informāciju par visām organizācijām, kuras lieto sistēmu.
Ievade	Funkcijai nav ievaddatu.
Apstrāde	Kad klienta bibliotēkas modulis izsauc šo funkciju, tad tiek izsaukta funkcija MDW-RDBD. Ja izsauktās funkcijas izpildes laikā rodas kāda kļūda, tad tiek izvadīts 1. vai 2. kļūdu ziņojums (skatīt funkciju MDW-RDBD). Ja kļūdu nav, tad tiek iegūts saraksts ar visiem datubāzē glabāto organizāciju objektiem.
Izvade	Saraksts ar visiem datubāzē saglabātajiem organizāciju objektiem. Ja nav atrasts neviens objekts, tiek atgriezts tukšs saraksts. Ja apstrādes laikā rodas kāda kļūda, tiek izvadīts atbilstošais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1002 Message: Database connection error!" 2. “Code: 1004 Message: Source object must not be null!”

1.3.1.3. Organizācijas atrašana datubāzē pēc ieraksta ID

Identifikators	GAM-OFBI
Apraksts	Funkcija ļauj no datubāzes iegūt informāciju par kādu konkrētu

	organizāciju, kura lieto sistēmu.
Ievade	Ievaddati tiek iegūti no klienta bibliotēkas moduļa ziņojuma pamatteksta. Lauki: 1. Organizācijas ID – 24 simbolu heksadecimāla virkne
Apstrāde	Ievaddatos esošais organizācijas ID tiks padots validācijas funkcijai, kura pārbaudīs vai ID atbilst datubāzes ID formātam. Ja ID neatbilst, izejošā ziņojuma pamattekstā tiks ierakstīts 1. kļūdu ziņojums. Ja ID atbilst formātam, tad tiek izsaukta funkcija MDW-RDBD. Ja izsauktās funkcijas izpildes laikā rodas kāda kļūda, tad tiek izvadīts 2. vai 3. kļūdu ziņojums (skatīt funkciju MDW-RDBD). Ja kļūdu nav, tad tiek atgriezts funkcijas rezultāts.
Izvade	Atbilstošais ieraksts datubāzē, ja pēc ID tiek atrasta organizācija. Ja apstrādes laikā rodas kāda kļūda, tiek izvadīts atbilstošais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1001 Message: Invalid ID!" 2. "Code: 1002 Message: Database connection error!" 3. “Code: 1004 Message: Source object must not be null!”

1.3.1.4. Datubāzē saglabāto organizāciju ID iegūšana

Identifikators	GAM-OGAI
Apraksts	Funkcija nepieciešama, lai no datubāzes iegūtu visu saglabāto organizāciju ierakstu identifikatorus, ko tālāk var izmantot, lai saglabātu vai atrastu organizācijai piesaistītas atrašanās vietas.
Ievade	Funkcijai nav ievaddatu.
Apstrāde	Kad klienta bibliotēkas modulis izsauc šo funkciju, tad tiek izsaukta funkcija MDW-RDBD. Ja izsauktās funkcijas izpildes laikā rodas kāda kļūda, tad tiek izvadīts 1. vai 2. kļūdu ziņojums (skatīt funkciju MDW-RDBD). Ja kļūdu nav, tad tiek saņemts saraksts ar visiem datubāzē glabāto organizāciju ierakstu identifikatoriem.
Izvade	Saraksts ar visiem datubāzē saglabāto organizāciju ierakstu identifikatoriem. Ja apstrādes laikā rodas kāda kļūda, tiek izvadīts atbilstošais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1002 Message: Database connection error!"

	2. “Code: 1004 Message: Source object must not be null!”

1.3.1.5. Organizācijas ieraksta ID atrašana pēc organizācijas ID API pārvaldības modulī

Identifikators	GAM-OFBA
Apraksts	Funkcija nepieciešama, lai no datubāzes iegūtu ieraksta ID organizācijai, kuras identifikators API pārvaldības modulī atbilst meklētajam. Funkcija nodrošina arī atbilstību atrašanu starp API pārvaldības modulī autentificēto lietotāju, kurš izsauc šo funkciju un kādu no datubāzē glabātajām organizācijām.
Ievade	Funkcijas ievaddati tiks iegūti no klienta bibliotēkas moduļa ziņojuma iesākuma: 1. Lietotājvārds - ierobežojumus skatīt funkcijā AMM-RNAU
Apstrāde	Kad klienta bibliotēkas modulis izsauc šo funkciju, tiek izdarīts pieprasījums uz API pārvaldības moduļa funkciju AMM-GOBU, lai pēc lietotājvārda atrastu konkrētajam lietotājam piesaistītās organizācijas ID API pārvaldības modulī. tad tiek izsaukta funkcija MDW-RDBD. Ja izsauktās funkcijas izpildes laikā rodas kāda kļūda, tad tiek izvadīts 1. vai 2. kļūdu ziņojums (skatīt funkciju MDW-RDBD). Ja kļūdu nav, tad tiek saņemts meklētās organizācijas ieraksta ID, vai arī teksts “null”.
Izvade	Ja tiek atrasta organizācija, tiek atgriezts šīs organizācijas ID datubāzē. Ja organizācija netiek atrasta, tiek atgriezts teksts “null”. Ja apstrādes laikā rodas kāda kļūda, tiek izvadīts atbilstošais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1002 Message: Database connection error!" 2. “Code: 1004 Message: Source object must not be null!”

1.3.1.6. Atrašanās vietu saglabāšana datubāzē

Identifikators	GAM-LCNL
Apraksts	Funkcija nepieciešama, lai saglabātu datubāzē informāciju par ierīču atrašanās vietām, kas piesaistītas sistēmā saglabātajām organizācijām.
Ievade	Ievaddati tiks iegūti no klienta bibliotēkas moduļa ziņojuma pamatteksta (message body).

	Lauki: 1. Organizācijas ID - 24 simbolu heksadecimāla virkne 2. Atrašanās vietas nosaukums - simbolu virkne garumā no 1 līdz 100 simboliem, drīkst saturēt punktus, atstarpes, domuzīmes, apostrofus, lielos un mazos burtus, ciparus, Pieļaujami arī burti ar diakritiskajām zīmēm 3. Temperatūras skala - 'C' vai 'F' 4. Režīms - skatīt režīma lauka nosacījumus 5. Laika zona - atļauti trīs formāti: a. Trīs lielie burti, piem., "UTC" b. Virkne "Etc/GMT", kam seko '+' vai '-' un stundu nobīde no GMT laika, piem., "Etc/GMT+10" c. Virkne, kas sastāv no reģiona nosaukuma, kam seko '/' un pilsētas nosaukuma, piem., "Europe/Riga" 6. Ģeogrāfiskais platums - decimāldaļskaitlis, slēgtā intervālā no -90.0 līdz +90.0 7. Ģeogrāfiskais garums - decimāldaļskaitlis, slēgtā intervālā no -180.0 līdz +180.0 Režīma lauks satur šādu informāciju: 1. Režīma ID - vesels skaitlis: 0; 1 vai 2 2. Režīma nosaukums - simbolu virkne, atļautās vērtības: a. "HOME" b. "AWAY c. "UNKNOWN""
Apstrāde	Ievaddati tiks padoti validācijas funkcijai (GAM-SSDV), kura pārbaudīs vai ievaddatos esošā informācija atbilst norādītajiem ierobežojumiem. Lauki tiek pārbaudīti tādā secībā, kādā tie norādīti šīs funkcijas ievaddatu aprakstā. Ja kāds lauks neatbilst ierobežojumiem, izejošajā ziņojumā tiek ierakstīts 1. kļūdu ziņojums. Kļūdu ziņojumā tiek iekļauts tā lauka nosaukums, pie kura radās validācijas kļūda. Ja vairāki lauki neatbilst nosacījumiem, tiek izvadīts nosaukums apstrādes secībā pirmajam kļūdainajam laukam. Ja visi lauki atbilst ierobežojumiem, tad datu objekts tiek padots funkcijai MDW-WDBD. Ja šai funkcijai rodas kāda kļūda, tiek izvadīts 2. vai 3. kļūdu ziņojums (skatīt funkciju MDW-WDBD). Ja

	kļūdu nav, tad tiek saņemts izveidotā datubāzes ieraksta ID.
Izvade	Izveidotā atrašanās vietas ieraksta ID datubāzē. Ja apstrādes laikā rodas kāda kļūda, tiek izvadīts atbilstošais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1001 Message: Invalid <*lauka nosaukums*>!" 2. "Code: 1002 Message: Database connection error!" 3. “Code: 1004 Message: Source object must not be null!”

1.3.1.7. Atrašanās vietu iegūšana no datubāzes

Identifikators	GAM-LFBI
Apraksts	Funkcija nepieciešama, lai pēc organizācijas ID atrastu visas organizācijai piesaistītās atrašanās vietas.
Ievade	Ievaddati tiek iegūti no klienta bibliotēkas moduļa ziņojuma pamatteksta. Lauki: 1. Organizācijas ID – 24 simbolu heksadecimāla virkne
Apstrāde	Ievaddatos esošais organizācijas ID tiks padots validācijas funkcijai, kura pārbaudīs vai ID atbilst datubāzes ID formātam. Ja ID neatbilst, izejošā ziņojuma pamattekstā tiks ierakstīts 1. kļūdu ziņojums. Ja ID atbilst formātam, tad tiek izsaukta funkcija MDW-RDBD. Ja izsauktās funkcijas izpildes laikā rodas kāda kļūda, tad tiek izvadīts 2. vai 3. kļūdu ziņojums (skatīt funkciju MDW-RDBD). Ja kļūdu nav, tad tiek iegūts saraksts ar visām attiecīgajai organizācijai piesaistītajām atrašanās vietām.
Izvade	Saraksts ar organizācijai piesaistītajām atrašanās vietām. Ja nav atrasta neviena atrašanās vieta, tiek atgriezts tukšs saraksts. Ja apstrādes laikā rodas kāda kļūda, tiek izvadīts atbilstošais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1001 Message: Invalid ID!" 2. "Code: 1002 Message: Database connection error!" 3. “Code: 1004 Message: Source object must not be null!”

1.3.1.8. Datubāzē saglabāto atrašanās vietu ID iegūšana

Identifikators	GAM-LGAI
Apraksts	Funkcija nepieciešama, lai no datubāzes iegūtu visu saglabāto

	atrašanās vietu ierakstu identifikatorus, ko tālāk var izmantot, lai saglabātu vai atrastu atrašanās vietai piesaistītas ierīces.
Ievade	Funkcijai nav ievaddatu.
Apstrāde	Kad klienta bibliotēkas modulis izsauc šo funkciju, tad tiek izsaukta funkcija MDW-RDBD. Ja izsauktās funkcijas izpildes laikā rodas kāda kļūda, tad tiek izvadīts 1. vai 2. kļūdu ziņojums (skatīt funkciju MDW-RDBD). Ja kļūdu nav, tad tiek iegūts saraksts ar visiem datubāzē glabāto atrašanās vietu ierakstu identifikatoriem.
Izvade	Saraksts ar visiem datubāzē saglabāto atrašanās vietu ierakstu identifikatoriem. Ja apstrādes laikā rodas kāda kļūda, tiek izvadīts atbilstošais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1002 Message: Database connection error!" 2. “Code: 1004 Message: Source object must not be null!”

1.3.1.9. Ierīču saglabāšana datubāzē

Identifikators	GAM-DCND
Apraksts	Funkcija nepieciešama, lai saglabātu datubāzē informāciju par ierīcēm, kas piesaistītas sistēmā saglabātajām ierīču atrašanās vietām.
Ievade	Ievaddati tiks iegūti no klienta bibliotēkas moduļa ziņojuma pamatteksta (message body). Lauki: 1. Atrašanās vietas ID - 24 simbolu heksadecimāla virkne 2. Ierīces ārējais ID - patvaļīga simbolu virkne (formāts atkarīgs no konkrētajā integrācijā lietotajiem sensoriem) 3. Ierīces nosaukums - simbolu virkne garumā no 1 līdz 100 simboliem, drīkst saturēt punktus, atstarpes, domuzīmes, apostrofus, lielos un mazos burtus, ciparus, Pieļaujami arī burti ar diakritiskajām zīmēm 4. Ierīces statuss - simbolu virkne, atļautās vērtības: a. “OK” b. “DOWN” c. “UNKNOWN” 5. Ierīces ražotājs - simbolu virkne garumā no 1 līdz 100 simboliem, drīkst saturēt punktus, atstarpes, domuzīmes,

	apostrofus, lielos un mazos burtus, ciparus, Pieļaujami arī burti ar diakritiskajām zīmēm 6. Ierīces modelis - simbolu virkne garumā no 1 līdz 100 simboliem, drīkst saturēt punktus, atstarpes, domuzīmes, apostrofus, lielos un mazos burtus, ciparus, Pieļaujami arī burti ar diakritiskajām zīmēm 7. Ierīces novietojuma klasifikatori - trīs veseli, nenegatīvi skaitļi
Apstrāde	Ievaddati tiks padoti validācijas funkcijai (GAM-SSDV), kura pārbaudīs vai ievaddatos esošā informācija atbilst norādītajiem ierobežojumiem. Lauki tiek pārbaudīti tādā secībā, kādā tie norādīti šīs funkcijas ievaddatu aprakstā. Ja kāds lauks neatbilst ierobežojumiem, izejošajā ziņojumā tiek ierakstīts 1. kļūdu ziņojums. Kļūdu ziņojumā tiek iekļauts tā lauka nosaukums, pie kura radās validācijas kļūda. Ja vairāki lauki neatbilst nosacījumiem, tiek izvadīts nosaukums apstrādes secībā pirmajam kļūdainajam laukam. Ja visi lauki atbilst ierobežojumiem, tad datu objekts tiek padots funkcijai MDW-WDBD. Ja šai funkcijai rodas kāda kļūda, tiek izvadīts 2. vai 3. kļūdu ziņojums (skatīt funkciju MDW-WDBD). Ja kļūdu nav, tad tiek saņemts izveidotā datubāzes ieraksta ID.
Izvade	Izveidotā ierīces ieraksta ID datubāzē. Ja apstrādes laikā rodas kāda kļūda, tiek izvadīts atbilstošais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1001 Message: Invalid <*lauka nosaukums*>!" 2. "Code: 1002 Message: Database connection error!" 3. “Code: 1004 Message: Source object must not be null!”

1.3.1.10. Ierīču iegūšana no datubāzes

Identifikators	GAM-DFBI
Apraksts	Funkcija nepieciešama, lai pēc atrašanās vietas ID atrastu visas atrašanās vietai piesaistītās ierīces.
Ievade	Ievaddati tiek iegūti no klienta bibliotēkas moduļa ziņojuma pamatteksta. Lauki: 1. Atrašanās vietas ID – 24 simbolu heksadecimāla virkne

Apstrāde	Ievaddatos esošais atrašanās vietas ID tiks padots validācijas funkcijai, kura pārbaudīs vai ID atbilst datubāzes ID formātam. Ja ID neatbilst, izejošā ziņojuma pamattekstā tiks ierakstīts 1. kļūdu ziņojums. Ja ID atbilst formātam, tad tiek izsaukta funkcija MDW-RDBD. Ja izsauktās funkcijas izpildes laikā rodas kāda kļūda, tad tiek izvadīts 2. vai 3. kļūdu ziņojums (skatīt funkciju MDW-RDBD). Ja kļūdu nav, tad tiek iegūts saraksts ar visām attiecīgajai atrašanās vietai piesaistītajām ierīcēm.
Izvade	Saraksts ar atrašanās vietai piesaistītajām ierīcēm. Ja apstrādes laikā rodas kāda kļūda, tiek izvadīts atbilstošais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1001 Message: Invalid ID!" 2. "Code: 1002 Message: Database connection error!" 3. “Code: 1004 Message: Source object must not be null!”

1.3.1.11. Datubāzē saglabāto ierīču ID iegūšana

Identifikators	GAM-DGAI
Apraksts	Funkcija nepieciešama, lai no datubāzes iegūtu visu saglabāto ierīču ierakstu identifikatorus, ko tālāk var izmantot, lai saglabātu vai atrastu ierīcei piesaistītus notikumus.
Ievade	Funkcijai nav ievaddatu.
Apstrāde	Kad klienta bibliotēkas modulis izsauc šo funkciju, tad tiek izsaukta funkcija MDW-RDBD. Ja izsauktās funkcijas izpildes laikā rodas kāda kļūda, tad tiek izvadīts 1. vai 2. kļūdu ziņojums (skatīt funkciju MDW-RDBD). Ja kļūdu nav, tad tiek saņemts saraksts ar visu datubāzē glabāto ierīču identifikatoriem.
Izvade	Saraksts ar visiem datubāzē saglabāto ierīču ierakstu identifikatoriem. Ja apstrādes laikā rodas kāda kļūda, tiek izvadīts atbilstošais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1002 Message: Database connection error!" 2. “Code: 1004 Message: Source object must not be null!”

1.3.1.12. Notikumu saglabāšana datubāzē

Identifikators	GAM-ECNE
Apraksts	Funkcija nepieciešama, lai saglabātu datubāzē informāciju par

	notikumiem, kas piesaistīti sistēmā saglabātajām ierīcēm.
Ievade	Ievaddati tiks iegūti no klienta bibliotēkas moduļa ziņojuma pamatteksta (message body). Lauki: 1. Ierīces ID - 24 simbolu heksadecimāla virkne 2. Datums/laiks - simbolu virkne formātā "yyyy-MM-dd'T'HH:mm:ss.SSS", kur: a. “yyyy”- gads b. “MM” - mēnesis c. “dd” - diena mēnesī d. “HH” - stundas e. “mm” - minūtes f. “ss” - sekundes g. “SSS” - milisekundes 3. Notikuma nosaukums - simbolu virkne garumā no 1 līdz 100 simboliem, drīkst saturēt punktus, atstarpes, domuzīmes, apostrofus, lielos un mazos burtus, ciparus, Pieļaujami arī burti ar diakritiskajām zīmēm 4. Notikuma mērījuma vērtība - decimāldaļskaitlis 5. Notikuma apraksts - simbolu virkne garumā līdz 200 simboliem (ieskaitot) 6. Notikuma režīms - simbolu virkne, atļautās vērtības: a. “HOME” b. “AWAY” c. “UNKNOWN”
Apstrāde	Ievaddati tiks padoti validācijas funkcijai (GAM-SSDV), kura pārbaudīs vai ievaddatos esošā informācija atbilst norādītajiem ierobežojumiem. Lauki tiek pārbaudīti tādā secībā, kādā tie norādīti šīs funkcijas ievaddatu aprakstā. Ja kāds lauks neatbilst ierobežojumiem, izejošajā ziņojumā tiek ierakstīts 1. kļūdu ziņojums. Kļūdu ziņojumā tiek iekļauts tā lauka nosaukums, pie kura radās validācijas kļūda. Ja vairāki lauki neatbilst nosacījumiem, tiek izvadīts nosaukums apstrādes secībā pirmajam kļūdainajam laukam. Ja visi lauki atbilst ierobežojumiem, tad datu objekts tiek padots funkcijai MDW-WDBD. Ja šai funkcijai rodas kāda kļūda, tiek

	izvadīts 2. vai 3. kļūdu ziņojums (skatīt funkciju MDW-WDBD). Ja kļūdu nav, tad tiek saņemts izveidotā datubāzes ieraksta ID. Pēc veiksmīgas notikuma saglabāšanas datubāzē, tiek izsaukta arī funkcija GAM-SRDD, kurai tiek padots notikuma objekts, kas tika saglabāts datubāzē.
Izvade	Izveidotā notikuma ieraksta ID datubāzē, vai arī atbilstošais kļūdu ziņojums, ja apstrādes laikā rodas kāda kļūda.
Kļūdu ziņojumi	1. "Code: 1001 Message: Invalid <*lauka nosaukums*>!" 2. "Code: 1002 Message: Database connection error!" 3. “Code: 1004 Message: Source object must not be null!”

1.3.1.13. Notikumu iegūšana no datubāzes

Identifikators	GAM-EFBI
Apraksts	Funkcija nepieciešama, lai pēc ierīces ID atrastu visus ierīcei piesaistītos notikumus.
Ievade	Ievaddati tiek iegūti no klienta bibliotēkas moduļa ziņojuma pamatteksta. Lauki: 1. Ierīces ID – 24 simbolu heksadecimāla virkne
Apstrāde	Ievaddatos esošais ierīces ID tiks padots validācijas funkcijai, kura pārbaudīs vai ID atbilst datubāzes ID formātam. Ja ID neatbilst, izejošā ziņojuma pamattekstā tiks ierakstīts 1. kļūdu ziņojums. Ja ID atbilst formātam, tad tiek izsaukta funkcija MDW-RDBD. Ja izsauktās funkcijas izpildes laikā rodas kāda kļūda, tad tiek izvadīts 2. vai 3. kļūdu ziņojums (skatīt funkciju MDW-RDBD). Ja kļūdu nav, tad tiek saņemts saraksts ar visiem attiecīgajai ierīcei piesaistītajiem notikumiem.
Izvade	Saraksts ar ierīcei piesaistītajiem notikumiem, vai arī atbilstošais kļūdu ziņojums, ja apstrādes laikā rodas kāda kļūda.
Kļūdu ziņojumi	1. "Code: 1001 Message: Invalid ID!" 2. "Code: 1002 Message: Database connection error!" 3. “Code: 1004 Message: Source object must not be null!”

1.3.1.14. Datubāzē saglabāto notikumu ID iegūšana

Identifikators	GAM-EGAI
Apraksts	Funkcija nepieciešama, lai no datubāzes iegūtu visu saglabāto notikumu ierakstu identifikatorus.
Ievade	Funkcijai nav ievaddatu.
Apstrāde	Kad klienta bibliotēkas modulis izsauc šo funkciju, tiek izsaukta funkcija MDW-RDBD. Ja izsauktās funkcijas izpildes laikā rodas kāda kļūda, tad tiek izvadīts 1. vai 2. kļūdu ziņojums (skatīt funkciju MDW-RDBD). Ja kļūdu nav, tiek saņemts saraksts ar visu datubāzē glabāto
Izvade	Saraksts ar visiem datubāzē saglabāto notikumu ierakstu identifikatoriem. Ja apstrādes laikā rodas kāda kļūda, tiek izvadīts atbilstošais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1002 Message: Database connection error!" 2. “Code: 1004 Message: Source object must not be null!”

1.3.1.15. Jēldatu dokumenta saglabāšana datubāzē

Identifikators	GAM-SRDD
Apraksts	Funkcija paredzēta, lai pie katra jauna notikuma izveides un saglabāšanas dokumentu datubāzē izveidotu jēldatu dokumentu, ko saglabāt jēldatu datubāzē. Šos datus paredzēts izmantot datu vizualizācijas modulī, kā arī dokumentu datubāzes atjaunošanai datu pazaudēšanas gadījumā.
Ievade	Funkcija ievaddatus saņem no funkcijas GAM-ECNE. Ievaddati: 1. Notikuma objekts (laukus un ierobežojumus skatīt funkcijā GAM-ECNE)
Apstrāde	Funkcija vispirms saņem notikuma objektu. No notikuma objekta tiek nolasīts tajā glabātais ierīces ID. Pēc šī ID dokumentu datubāzē tiek meklēta ierīce. Ja ierīci neatrod, tiek izvadīts 1. kļūdu ziņojums. Ja ierīci atrod, no tās objekta tiek nolasīts atrašanās vietas ID. Pēc šī ID dokumentu datubāzē tiek meklēta atrašanās vieta. Ja atrašanās vietu neatrod, tiek izvadīts 2. kļūdu ziņojums. Ja atrašanās vietu atrod, no tās objekta tiek nolasīts organizācijas ID. Pēc šī ID

	dokumentu datubāzē tiek meklēta organizācija. Ja organizāciju neatrod, tiek izvadīts 3. kļūdu ziņojums. Ja organizāciju atrod, tad visi iegūtie objekti tiek ierakstīti jaunā dokumenta objektā. Šo objektu pārveido par JSON simbolu virkni un saglabā jēldatu datubāzē. Ja, izpildot objektu meklēšanu dokumentu datubāzē, neizdodas izveidot savienojumu ar dokumentu datubāzi, tiek izvadīts 4. kļūdu ziņojums. Ja pie izveidotā dokumenta pārveidošanas par JSON simbolu virkni rodas kāda kļūda, tiek izvadīts 5. kļūdu ziņojums.
Izvade	Tiek atgriezts paziņojums par veiksmīgu saglabāšanas operācijas izpildi, vai arī atbilstošais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1005 Message: Device with ID <*ierīces ID*> not found in database" 2. "Code: 1005 Message: Location with ID <*atrašanās vietas ID*> not found in database" 3. "Code: 1005 Message: Organisation with ID <*organizācijas ID*> not found in database" 4. "Code: 1002 Message: Database connection error!" 5. "Code: 1006 Message: Error converting raw document JSON to String."

1.3.1.16. Ierīču novietojuma klasifikatoru (tags) iegūšana no datubāzes

Identifikators	GAM-TGAO
Apraksts	Funkcija paredzēta, lai iegūtu visus datubāzē esošos ierīču novietojuma klasifikatorus.
Ievade	Funkcijai nav ievaddatu.
Apstrāde	Kad šo funkciju izsauc klienta bibliotēkas modulis, tad tiek izsaukta funkcija MDW-RDBD. Ja izsauktās funkcijas izpildes laikā rodas kāda kļūda, tad tiek izvadīts 1. vai 2. kļūdu ziņojums (skatīt funkciju MDW-RDBD). Ja kļūdu nav, tad tiek saņemts saraksts ar visiem atrastajiem klasifikatoriem.
Izvade	Saraksts ar visiem datubāzē saglabātajiem ierīču novietojuma klasifikatoriem. Ja datubāzē netika atrasts neviens klasifikators, tiek atgriezts tukšs saraksts. Ja apstrādes laikā rodas kāda kļūda, tiek

	izvadīts atbilstošais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1002 Message: Database connection error!" 2. “Code: 1004 Message: Source object must not be null!”

1.3.1.17. Ierīču novietojuma klasifikatoru (tags) importēšana datubāzē no JSON failiem

Identifikators	GAM-TIFJ
Apraksts	Funkcija paredzēta, lai datubāzē jau pie sistēmas palaišanas ievietotu ierīču novietojuma klasifikatorus.
Ievade	Funkcija izmantos visus konkrētā mapē ievietotos JSON failus. Katrā JSON failā katra rindiņa ir viens datubāzes ieraksts. Katrā rindiņā jābūt šādai informācijai: 1. Ieraksta ID - pozitīvs nenulles skaitlis, unikāls starp visiem JSON failu ierakstiem 2. Klasifikatora nosaukums - patvaļīga simbolu virkne
Apstrāde	Šī funkcija tiek izsaukta sistēmas palaišanas procesā un tā apstrādā visus konkrētā mapē atrodamos JSON failus. Katra faila katru rindiņu funkcija mēģina ierakstīt datubāzē kā ierakstu. Apturot un palaižot sistēmu atkārtoti, datubāzē tiek pievienoti ieraksti ar jauniem un unikāliem ID, kā arī atjaunināti ieraksti, kuru ID jau ir atrodami datubāzē. Gadījumā, ja neizdodas izveidot savienojumu ar datubāzi, tiek izvadīts 1. kļūdu ziņojums.
Izvade	Ja apstrādes laikā rodas kāda kļūda, tiek izvadīts atbilstošais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1002 Message: Database connection error!"

1.3.1.18. Datu validācija servera pusē

Identifikators	GAM-SSDV
Apraksts	Funkcija nepieciešama, lai veiktu datu pārsūtīšanas objektu validāciju servera pusē.
Ievade	Funkcijas ievaddati tiek saņemti no klienta bibliotēkas moduļa ziņojuma pamatteksta.
Apstrāde	Validējamais objekts tiek padots datu validācijas moduļa funkcijai DVM-VDTO.
Izvade	Tiek atgriezts validācijas rezultāts kā vērtība “true”, ja objekts atbilst

	visiem validācijas nosacījumiem. Lauki tiek pārbaudīti tādā secībā, kādā tie norādīti attiecīgā objekta izveidošanas funkcijas ievaddatu aprakstā. Ja kāds lauks neatbilst ierobežojumiem, izejošajā ziņojumā tiek ierakstīts 1. kļūdu ziņojums. Kļūdu ziņojumā tiek iekļauts tā lauka nosaukums, pie kura radās validācijas kļūda. Ja vairāki lauki neatbilst nosacījumiem, tiek izvadīts nosaukums apstrādes secībā pirmajam kļūdainajam laukam.
Kļūdu ziņojumi	1. "Code: 1001 Message: Invalid <*lauka nosaukums*>!"

1.3.2. Starpprogrammatūras modulis

1.3.2.1. Datu pieejas objektu izveide

Identifikators	MDW-CDAO
Apraksts	Funkcija nepieciešama, lai datubāzē varētu izpildīt no vārtejas API moduļa nākošos pieprasījumus.
Ievade	Funkcija ievaddatos saņem datu objektus no funkcijām MDW-RDBD un MDW-WDBD.
Apstrāde	Saņemtie datu objekti tiek padoti funkcijai DMM-MDTO. Ja minētajā funkcijā rodas kļūda, tiek atgriezts 1. kļūdu ziņojums. Ja kļūdu nav, tiek saņemts pārveidotais datu objekts, ko atgriež izsaucošajai funkcijai.
Izvade	Datu pieejas objekts vai arī attiecīgais kļūdas ziņojums.
Kļūdu ziņojumi	1. “Code: 1004 Message: Source object must not be null!”

1.3.2.2. Datu ielasīšana no datubāzes

Identifikators	MDW-RDBD
Apraksts	Funkcija nepieciešama, lai no datubāzes iegūtu vārtejas API moduļa funkciju pieprasītos datus.
Ievade	Funkcija ievaddatus saņem no vārtejas API moduļa funkcijām. Ievaddatos ir izpildāmā datubāzes operācija. Ja tiek pieprasīta meklēšana pēc ID, tad ievaddatos jābūt arī identifikatoram, pēc kura tiks izdarīta meklēšana.
Apstrāde	Izsaucot funkciju, pēc ievaddatiem tiek izveidots pieprasījums uz datubāzi ar attiecīgo vaicājumu. No datubāzes iegūtie dati tiek padoti

	funkcijai MDW-CDAO. Ja neizdodas izveidot savienojumu ar datubāzi, tiek izvadīts 1. kļūdu ziņojums. Ja minētajā funkcijā rodas kļūda, tiek atgriezts 2. kļūdas ziņojums. Ja kļūdu nav, tad datu pieejas objekti tiek atgriezti izsaucošajai vārtejas API moduļa funkcijai.
Izvade	No datubāzes iegūtie dati vai arī attiecīgais kļūdas ziņojums.
Kļūdu ziņojumi	1. "Code: 1002 Message: Database connection error!" 2. "Code: 1004 Message: Source object must not be null!"

1.3.2.3. Datu ierakstīšana datubāzē

Identifikators	MDW-WDBD
Apraksts	Funkcija nepieciešama, lai datubāzē varētu saglabāt no vārtejas API moduļa funkcijām saņemtos datu objektus.
Ievade	Funkcija saņem saglabājamo datu objektu no izsaucošās vārtejas API moduļa funkcijas.
Apstrāde	Saņemtais datu objekts tiek padots funkcijai MDW-CDAO. Ja funkcijā rodas kļūda, tad tiek izvadīts 1. kļūdu ziņojums. Ja kļūdu nav, tad uz datubāzi tiek nosūtīts pieprasījums kopā ar saglabājamo datu objektu. Ja neizdodas izveidot savienojumu ar datubāzi, tad tiek izvadīts 2. kļūdu ziņojums. Ja operācija ir veiksmīga, tiek atgriezts izveidotā datubāzes ieraksta ID.
Izvade	Izveidotā datubāzes ieraksta ID vai arī attiecīgais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1004 Message: Source object must not be null!" 2. "Code: 1002 Message: Database connection error!"

1.3.3. Kartēšanas modulis

1.3.3.1. Datu pārsūtīšanas objektu (DTO) pārveidošana

Identifikators	DMM-MDTO
Apraksts	Funkcija nodrošina vārtejas API moduļa datu pārsūtīšanas objektu pārveidošanu par starpprogrammatūras moduļa objektiem un otrādi, lai nodrošinātu vārtejas API moduļa un starpprogrammatūras moduļa savstarpējo saziņu.
Ievade	Funkcija ievaddatus iegūst no starpprogrammatūras moduļa funkcijas MDW-CDAO. Tā pieņem šādus datu objektus:

	1. Organizācija - laukus un to ierobežojumus skatīt funkcijā GAM-OCNO 2. Atrašanās vieta - laukus un to ierobežojumus skatīt funkcijā GAM-LCNL 3. Ierīce - laukus un to ierobežojumus skatīt funkcijā GAM-DCND 4. Notikums - laukus un to ierobežojumus skatīt funkcijā GAM-ECNE 5. ID objekts - 24 simbolus gara heksadecimāla virkne
Apstrāde	Organizāciju, atrašanās vietu, ierīču un notikumu objektiem tiek kartēti lauki, kuru nosaukumi sakrīt. Starpprogrammatūras moduļa objektiem tiek pievienots lauks "ID", kas ir paša objekta identifikators datubāzē. Kartēšanā uz vārtejas API moduļa datu objektiem šis lauks tiek ignorēts. Ja kartēšana notiek no starpprogrammatūras moduļa organizācijas, atrašanās vietas, ierīces vai notikuma objekta uz ID objektu, tad ID objektā tiek ierakstīta attiecīgā objekta "ID" lauka vērtība. Ja kartēšanas avota objekts ir "null", tad tiek izvadīts 1. kļūdu ziņojums.
Izvade	Funkcija atgriež kartēšanas rezultātā izveidoto objektu vai attiecīgo kļūdas ziņojumu.
Kļūdu ziņojumi	1. "Code: 1004 Message: Source object must not be null!"

1.3.4. Datu vizualizācijas modulis

1.3.4.1. Datu vizualizāciju izveide

Identifikators	DVS-DDGM
Apraksts	Funkcija nepieciešama, lai jēldatu datubāzē saglabātos notikumu datus varētu attēlot grafiskā formā, tādējādi atvieglojot datu analīzi.
Ievade	Funkcijas ievaddati tiks iegūti caur lietotāja saskarni. 1. Vizualizācijas tips - izvēle starp šādiem tipiem: a. Līnijas grafiks b. Stabiņu diagramma c. Datu attēlojums kartē d. Sektoru diagramma Funkcija paredz arī iespēju rediģēt jau saglabātas vizualizācijas. Šādā

	gadījumā lietotājs var izvēlēties vizualizāciju no saglabāto vizualizāciju saraksta.
Apstrāde	Gadījumā, kad lietotājs veido jaunu vizualizāciju, tad, kad ir izvēlēts vizualizācijas tips, lietotājs var vizualizāciju konfigurēt, izvēloties, kurus datu laukus iekļaut vizualizācijā, kā šos datus agregēt, kā arī konfigurēt asu nosaukumus, mērogus. Izveidoto vizualizāciju saglabājot, tai jāpiešķir nosaukums. Gadījumā ja lietotājs rediģē jau saglabātu vizualizāciju, prasības ir tādas pašas, kā veidojot jaunu vizualizāciju, tikai lietotājam nav iespējas mainīt vizualizācijas tipu. Abos gadījumos, ja lietotājs mēģina saglabāt kādā veidā kļūdaini konfigurētu vizualizāciju, tiek izvadīts 1. kļūdas ziņojums un saglabāšana netiek veikta.
Izvade	Paziņojums par veiksmīgu vizualizācijas saglabāšanu, vai arī attiecīgais kļūdas ziņojums.
Kļūdu ziņojumi	1. "Code: 1008 Message: Errors in visualisation"

1.3.4.2. Datu vizualizāciju apkopošana informācijas paneļos

Identifikators	DVS-DVDB
Apraksts	Funkcija nepieciešama, lai varētu veidot informācijas paneļus (dashboards), kuros apkopotas vairākas datu vizualizācijas.
Ievade	Funkcijas ievaddatus iegūst caur lietotāja saskarni. 1. Vizualizāciju novietojums - lietotājs to var konfigurēt, izmantojot "vilkt un nomest" saskarni (drag and drop) Funkcija paredz arī iespēju rediģēt jau esošu informācijas paneli. Šādā gadījumā lietotājam jāizvēlas informācijas panelis no saglabāto informācijas paneļu saraksta.
Apstrāde	Gan veidojot jaunu informācijas paneli, gan rediģējot jau iepriekš saglabātu informācijas paneli, lietotājs var panelim pievienot vizualizācijas no saglabāto vizualizāciju saraksta. Panelim pievienotās vizualizācijas var mērogot, velkot aiz vizualizācijas stūra, vai arī pārvietot, novietojot kursoru vizualizācijas vidū un velkot. Pārvietojumu apstiprināt var, atlaižot peles taustiņu. Saglabājot veiktās izmaiņas, ir jānorāda vizualizācijas nosaukums.

Izvade	Paziņojums par veiksmīgu informācijas paneļa izmaiņu saglabāšanu.
Kļūdu ziņojumi	Nav.

1.3.5. API pārvaldības modulis

1.3.5.1. Autentifikācija

Identifikators	AMM-AUTH
Apraksts	Funkcija nodrošina ka platformas servisiem piekļūt var tikai autentificēti lietotāji.
Ievade	Funkcijai ievaddati tiek padoti vai nu ziņojuma iesākumā vai arī caur lietotāja saskarni. Funkcijai nepieciešami šādi dati: 1. Lietotājvārds 2. Parole Abu lauku ierobežojumus skatīt funkcijā AMM-RNAU.
Apstrāde	Ievadītie dati tiek salīdzināti ar API pārvaldības moduļa datubāzes ierakstiem. Ja ievadītie dati atbilst datubāzē atrastajiem, tad lietotājs tiek autentificēts sistēmā. Ja dati neatbilst, tiek izvadīts 1. kļūdas ziņojums.
Izvade	Veiksmīgas autentifikācijas gadījumā lietotājs tiek novirzīts uz API pārvaldības moduļa saskarnes sākumlapu. Citos gadījumos tiek izvadīts attiecīgais kļūdu ziņojums.
Kļūdu ziņojumi	1. “Code: 1007 Message: Incorrect username or password!”

1.3.5.2. Organizācijas izmantotā datu daudzuma ierobežošana

Identifikators	AMM-LODQ
Apraksts	Funkcija nepieciešama, lai varētu realizēt funkcijā AMM-RNAP aprakstīto API lietojuma plānu nosacījumus par izmantotā datu apjoma ierobežošanu.
Ievade	Funkcijas ievaddati ir ienākošie API pieprasījumi.
Apstrāde	Funkcija nolasa API pieprasījuma ziņojuma pamattekstu un nosaka tā izmēru. No pieprasījuma iesākuma tiek iegūta organizācijas API piekļuves atslēga. Pēc tās API pārvaldības moduļa datubāzē tiek atrasts, cik lielu datu apjomu organizācija jau ir izmantojusi. Ja pieskaitot saņemtā ziņojuma pamatteksta izmēru, tas pārsniedz plānā

	noteikto limitu, ziņojums netiek padots vārtejas API modulim un tiek atgriezts 1. kļūdu ziņojums. Ja limits netiek pārsniegts, tiek atjaunināts organizācijas izmantotā datu apjoma ieraksts datubāzē un pieprasījums tiek padots uz attiecīgo vārtejas API moduļa REST servisu.
Izvade	Tiek atgriezta izsauktā REST servisa atbilde, vai arī attiecīgais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1009 Message: Organisation data quantity limit reached!"

1.3.5.3. Organizācijai pieejamā pieprasījumu skaita ierobežošana

Identifikators	AMM-TOAR
Apraksts	Funkcija nepieciešama, lai varētu realizēt funkcijā AMM-RNAP aprakstīto API lietojuma plānu nosacījumus par pieejamo pieprasījumu skaita ierobežošanu (throttling).
Ievade	Funkcijas ievaddati ir ienākošie API pieprasījumi.
Apstrāde	No pieprasījuma iesākuma tiek iegūta organizācijas API piekļuves atslēga, pēc kuras tiek atrasts organizācijas jau izdarīto pieprasījumu skaits. Šis skaitlis tiek palielināts par viens, un ja tas pārsniedz API lietojuma plānā noteikto pieprasījumu skaitu, pieprasījuma pamattekstā tiek ierakstīts 1. kļūdu ziņojums. Ja skaits nav pārsniegts, tad pieprasījums tiek padots attiecīgajam vārtejas API REST servisam.
Izvade	Izsauktā vārtejas API REST servisa atbilde vai arī attiecīgais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1009 Message: Organisation request limit reached!"

1.3.5.4. Lietotāja organizācijas ID iegūšana pēc lietotājvārda

Identifikators	AMM-GOBU
Apraksts	Funkcija nepieciešama, lai pēc autentificētā lietotāja lietotājvārda iegūtu atbilstošās organizācijas ID, ko izmantot, lai vārtejas API modulī pēc šī ID atrastu datubāzē atbilstošo organizācijas ierakstu.
Ievade	Funkcijai nepieciešamos ievaddatus padod kā pieprasījuma vaicājuma parametrus (query parameters):

	1. Lietotājvārds - ierobežojumus skatīt funkcijā AMM-RNAU
Apstrāde	Kad funkcija tiek izsaukta, pēc padotā lietotājvārda API pārvaldības moduļa datubāzē tiek meklēts lietotājs. Ja lietotājs tiek atrasts, tad tiek meklēta šim lietotājam atbilstošā organizācija. Situācija, ka lietotājs netiek atrasts, nevar iestāties, jo funkcija pieejama tikai autentificētiem lietotājiem. Neautentificēta piekļuves mēģinājuma gadījumā tiek atgriezts 1. kļūdu ziņojums.
Izvade	Ja meklētajam lietotājam atbilst kāda organizācija, tad tiek atgriezts šīs organizācijas ID API pārvaldības modulī. Ja lietotājam neatbilst neviena organizācija, tad tiek atgriezts tukšs saraksts. Kļūdas gadījumā tiek atgriezts attiecīgais kļūdu ziņojums.
Kļūdu ziņojumi	1. “Code: 1009 Message: Unauthorized access!”

1.3.5.5. Lietotāja reģistrācija

Identifikators	AMM-RNAU
Apraksts	Funkcija nepieciešama, lai API pārvaldības modulī piereģistrētu jaunus lietotājus.
Ievade	Funkcija datus iegūst no ievadformas saskarnē. Nepieciešami šādi dati: 1. Lietotāja lietotājvārds - simbolu virkne garumā no 3 līdz 20 simboliem, drīkst saturēt burtus, ciparus un atstarpes 2. Lietotāja vārds - simbolu virkne garumā no 3 līdz 20 simboliem, drīkst saturēt burtus un atstarpes 3. Lietotāja uzvārds - simbolu virkne garumā no 3 līdz 30 simboliem, drīkst saturēt burtus un atstarpes 4. Lietotāja e-pasta adrese - simbolu virkne, kas atbilst e-pasta adreses formātam 5. Lietotāja parole - simbolu virkne garumā no 8 līdz 40 simboliem, drīkst saturēt burtus, ciparus un simbolus, obligāti jāsatur vismaz viens lielais burts, vismaz viens cipars un vismaz viens simbols 6. Parole atkārtoti - simbolu virknei jāsakrīt ar paroles laukā ievadīto
Apstrāde	Pēc datu ievadīšanas lietotājam ir jāspiež poga “Register”. Ja kāds no

	ievaddatu laukiem neatbilst nosacījumiem, kļūdainais lauks tiek vizuāli izcelts, tā norādot uz kļūdu. Ja visi lauki atbilst nosacījumiem, lietotājs tiek saglabāts API pārvaldības moduļa datubāzē, kā arī uzreiz autentificēts sistēmā un novirzīts uz API pārvaldības moduļa saskarnes sākumlapu. Ja API pārvaldības moduļa datubāzē netiek atrasts lietotāja ievadītajai informācijai atbilstošs ieraksts, autentifikācija nenotiek un tiek izvadīts 1. kļūdu ziņojums.
Izvade	Lietotājam tiek atgriezts API pārvaldības moduļa sākumlapas saturs vai arī attiecīgais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1009 Message: Invalid username and/or password!"

1.3.5.6. Aizmirstas paroles nomaiņa

Identifikators	AMM-RFPW
Apraksts	Funkcija nepieciešama, lai lietotājiem būtu iespēja nomainīt paroli tās aizmiršanas gadījumā.
Ievade	Funkcijas ievaddatus iegūs no ievadformas lietotāja saskarnē. 1. Lietotāja e-pasts - e-pasta adreses formātam atbilstoša simbolu virkne
Apstrāde	Izvēloties nomainīt aizmirsto paroli, lietotājs tiek novirzīts uz ievadformas skatu. Pēc e-pasta adreses ievadīšanas ievadformā, lietotājam jāspiež pogu "Nosūtīt". Ja e-pasta adrese neatbilst formātam, tiek izvadīts 1. kļūdu ziņojums un ievadformas lauks tiek vizuāli izcelts. Ja e-pasta adrese atbilst nosacījumiem, tad uz ekrāna parādās paziņojums par paroles nomaiņas instrukciju nosūtīšanu uz norādīto e-pasta adresi. Ja instrukciju nosūtīšanā rodas kļūda, tiek izvadīts 2. kļūdu ziņojums. Veiksmīgas instrukciju nosūtīšanas gadījumā, lietotāja e-pastā būs atrodama ziņu ar saiti, uz kuras uzspiežot, lietotājs tiek novirzīts uz tīmekļā lapu, kurā jāievada jauna parole un tās apstiprinājums. Ja jaunā parole neatbilst paroļu nosacījumiem (skat. funkciju AMM-RNAU), tiek izvadīts 3. kļūdu ziņojums, bet, ja nesakrīt vērtības jaunās paroles un paroles apstiprinājuma laukos, tiek izvadīts 4. kļūdu paziņojums. Ja kļūdu nav, tiek atjaunināts lietotāja ieraksts API pārvaldības moduļa datubāzē, lietotājs tiek autentificēts sistēmā un novirzīts uz API

	pārvaldības moduļa sākumlapu.
Izvade	Lietotājam tiek atgriezts API pārvaldības moduļa sākumlapas saturs, vai arī attiecīgais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1009 Message: Invalid email address!" 2. "Code: 1009 Message: Failed to send instructions!" 3. "Code: 1009 Message: Password does not meet requirements! (8 to 40 symbols, at least one capital letter, at least one digit and at least one symbol)" 4. "Code: 1009 Message: Passwords do not match!"

1.3.5.7. Organizācijas reģistrācija API pārvaldības modulī

Identifikators	AMM-ROAM
Apraksts	Funkcija nepieciešama, lai API pārvaldības modulī piereģistrētu jaunu organizāciju, kā arī izveidotu organizācijas ierakstu datubāzē.
Ievade	Funkcijas ievaddati tiek iegūti caur lietotāja saskarnes ievadformu. Nepieciešamos ievaddatus un to ierobežojumus skatīt funkcijā GAM-OCNO.
Apstrāde	Kad lietotājs ir ievadījis nepieciešamos ievaddatus, ir jāspiež saglabāšanas poga. Ja organizācijas nosaukums sakrīt ar kādu sistēmā jau esošu organizāciju, tiek atgriezts 1. kļūdu ziņojums. Ievadītie dati tiek padoti funkcijai GAM-OCNO. Ja kāds no laukiem neatbilst nosacījumiem, tiek atgriezts 2. kļūdu ziņojums. Ja kļūdu nav, tad dokumentu datubāzē tiks izveidots organizācijas ieraksts, kā arī organizācija tiks reģistrēta API pārvaldības moduļa datubāzē, bet lietotājs tiks novirzīts uz organizācijas sākumlapu API pārvaldības modulī.
Izvade	Organizācijas sākumlapas saturs vai arī attiecīgais kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1009 Message: Organisation with name <*organizācijas nosaukums*> already exists!" 2. "Code: 1001 Message: Invalid <*lauka nosaukums*>!"

1.3.5.8. API reģistrācija API pārvaldības modulī

Identifikators	AMM-RNAM
Apraksts	Funkcija nepieciešama, lai piereģistrētu jaunu pārvaldāmu API

	servisu.
Ievade	Funkcijas ievaddati tiek iegūti no ievadformas. 1. API servisa pieejas adrese - URL simbolu virkne ar protokolu, domēna vārdu vai IP adresi, portu un ceļu 2. API servisa nosaukums API pārvaldības modulī - patvaļīga simbolu virkne 3. Izmantojamais API lietojuma plāns - izvēle no API pārvaldības moduļa datubāzē saglabātajiem API lietojuma plāniem, vai arī izvēle padarīt servisu publisku (bez ierobežojumiem)
Apstrāde	Kad ir ievadīti visi ievaddati, jāspiež saglabāšanas poga. Ja ievadītā API servisa pieejas adrese nesatur kādu no norādītajām sastāvdaļām, tiek izvadīts 1. kļūdu ziņojums. Ja kļūdu nav, API serviss tiek reģistrēts API pārvaldības moduļa datubāzē un lietotājs tiek novirzīts uz API pārvaldības moduļa sākumlapu.
Izvade	API pārvaldības moduļa sākumlapas saturs vai arī attiecīgais kļūdu ziņojums.
Kļūdu ziņojumi	1. “Code: 1009 Message: API service URL is invalid!”

1.3.5.9. API lietojuma plānu izveidošana

Identifikators	AMM-RNAP
Apraksts	Funkcija nepieciešama, lai izveidotu jaunus API lietojuma plānus, ar kuriem ierobežot API lietojumu.
Ievade	Funkcija ievaddatus iegūst no lietotāja saskarnes ievadformas: 1. Plāna nosaukums - patvaļīga simbolu virkne, organizācijas ietvaros unikāls 2. Plāna apraksts - neobligāts, patvaļīga simbolu virkne 3. Plāna nosacījumi - var pievienot vairākus nosacījumus, izvēle: a. Ierobežot pieprasījumu skaitu laika periodā (minūtē, stundā, diennaktī, mēnesī vai gadā) b. Ierobežot pārsūtīto datu apjomu laika periodā (minūtē, stundā, diennaktī, mēnesī vai gadā) c. Liegt piekļuvi pie plānu lietojošiem API

	neautentificētiem lietotājiem
Apstrāde	Kad lietotājs ir aizpildījis visus ievadformas laukus, ir jāspiež saglabāšanas poga. Ja lietotāja organizācijas ietvaros jau eksistē plāns ar tādu pašu nosaukumu, kā ievadformā ievadītais, tad tiek izvadīts 1. kļūdas ziņojums. Ja kļūdu nav, tad tiek veikta plāna saglabāšana API pārvaldības moduļa datubāzē.
Izvade	Paziņojums par veiksmīgu plāna saglabāšanu vai arī attiecīgais kļūdas ziņojums.
Kļūdu ziņojumi	1. "Code: 1009 Message: Plan with name <*plāna nosaukums*> already exists in organisation!"

1.3.5.10. API klienta lietotnes reģistrācija

Identifikators	AMM-RNAC
Apraksts	Funkcija nepieciešama, lai reģistrētu API izmantojošās klientu lietotnes (klientu integrācijas ar platformu), kas ir piesaistītas kādai organizācijai sistēmā.
Ievade	Funkcijas ievaddati tiek iegūti no ievadformas. 1. Klienta lietotnes nosaukums - patvaļīga simbolu virkne 2. Izmantojamais API - izvēle no API pārvaldības modulī reģistrētajiem API servisiem 3. Izmantojamais API lietojuma plāns - izvēle no API pārvaldības modulī reģistrētajiem API servisiem, vai arī izvēle "public"
Apstrāde	Kad visi dati ir ievadīti, lietotājam jāspiež saglabāšanas poga. Ja klienta lietotnes nosaukums sakrīt ar kādu jau reģistrētu klienta lietotni, tiek izvadīts 1. kļūdu ziņojums. Ja kļūdu nav, klienta lietotne tiek reģistrēta API pārvaldības moduļa datubāzē un piesaistīta dotajā brīdī autentificētā lietotāja organizācijai. Lietotājs tiek novirzīts uz klienta lietotnes sākumlapu API pārvaldības modulī. Ja pie lietojuma plāna tika izvēlēta jebkura opcija, izņemot "public", tad lietotnei tiek uzģenerēta API lietojuma atslēga, kuru jāpievieno visiem API pieprasījumiem no šīs lietotnes.
Izvade	Klienta lietotnes sākumlapas saturs API pārvaldības modulī vai arī attiecīgais kļūdu ziņojums.

Kļūdu ziņojumi	1. “Code: 1009 Message: Client app with name <*lietotnes nosaukums*> already exists!”

1.3.6. Reversā starpniekservera modulis

1.3.6.1. Pieprasījumu nodošana attiecīgajam servisam platformā

Identifikators	RPM-RTTS
Apraksts	Funkcija nepieciešama, lai nodrošinātu piekļuvi sistēmas funkcijām caur reverso starpniekserveri, kā arī lai visiem sistēmas moduļiem varētu izmantot vienu publisko IP adresi.
Ievade	Funkcija ievaddatus saņem no HTTP pieprasījumiem. 1. Pieprasījumā norādītais domēna vārds 2. Pieprasījumā iekļautie ziņojuma iesākuma (message header) lauki - patvaļīgi atslēgu - vērtību pāri 3. Pieprasījumā iekļautais ziņojuma pamatteksts - patvaļīga simbolu virkne
Apstrāde	Pēc pieprasījumā iekļautā domēna vārda, tiek noteikts, kuram platformas servisam pieprasījums būtu jāpadod. Starpniekserveris uz attiecīgo servisu padod arī ziņojuma iesākuma vērtības, kā arī ziņojuma pamattekstu. Pēc tam, kad ziņojums ir ticis apstrādāts sistēmā, starpniekserveris atgriež atbildi. Ja no iekšējā servisa iegūtā atbilde ir ar HTTP kodiem 2** vai 3**, tad lietotājam tiek atgriezta neizmainīta iekšējā servisa atbilde. Ja pieprasītajā servisā rodas iekšēja kļūda, un tā atbilde satur HTTP kodus 5**, tad starpniekserveris izveido atbildi, kas satur 1. kļūdu ziņojumu.
Izvade	Izsauktā servisa atbilde uz pieprasījumu vai arī attiecīgais kļūdu ziņojums.
Kļūdu ziņojumi	1. “500 Internal server error!”

1.3.7. Klienta bibliotēkas modulis

1.3.7.1. Klienta puses datu validācija

Identifikators	CLM-CSDV
Apraksts	Funkcija nepieciešama, lai veiktu datu objektu validāciju klienta pusē, tādejādi samazinot vārtejas API un datubāzes noslodzi.

Ievade	Objektu, kam jāveic validācija, funkcija saņem no klientu integrāciju pieprasījumiem.
Apstrāde	Saņemto objektu funkcija padod datu validācijas moduļa funkcijai DVM-VDTO.
Izvade	Tiek atgriezts validācijas rezultāts kā vērtība “true”, ja objekts atbilst visiem validācijas nosacījumiem, vai arī 1. kļūdu ziņojums, ja kaut viens objekta lauks neatbilst validācijas nosacījumiem.
Kļūdu ziņojumi	1. "Code: 1001 Message: Invalid <*lauka nosaukums*>!"

1.3.7.2. Vārtejas API moduļa REST servisu izsaukšana

Identifikators	CLM-GAEC
Apraksts	Funkcija nepieciešama, lai platformu lietojošo izstrādātāju veidotās integrācijas varētu piekļūt vārtejas API moduļa REST servisiem.
Ievade	Funkcijas ievaddati ir simbolu virkne formā: “phoenix:*P1*?basePath=*P2*&method=*P3*”, kur ● P1 - patvaļīgs nosaukums ● P2 - vārtejas API servera IP adrese/domēna vārds un ports ● P3 - izsaucamā REST galapunkta nosaukums Derīgi REST servisu nosaukumi: 1. “ping” - savienojuma testēšanai 2. “tags” - izsauc funkciju GAM-TGAO 3. “organisationByID” - izsauc funkciju GAM-OFBI 4. “organisationAll” - izsauc funkciju GAM-OFAO 5. “organisationIDs” - izsauc funkciju GAM-OGAI 6. “organisationCreate” - izsauc funkciju GAM-OCNO 7. “organisationApimanID””- izsauc funkciju GAM-OFBA 8. “locationsByOrgID” - izsauc funkciju GAM-LFBI 9. “locationIDs” - izsauc funkciju GAM-LGAI 10. “locationCreate” - izsauc funkciju GAM-LCNL 11. “devicesByLocID” - izsauc funkciju GAM-DFBI 12. “deviceIDs” - izsauc funkciju GAM-DGAI 13. “deviceCreate” - izsauc funkciju GAM-DCND 14. “eventsByDevID” - izsauc funkciju GAM-EFBI

	15. “eventIDs” - izsauc funkciju -GAM-EGAI 16. “eventCreate” - izsauc funkciju GAM-ECNE Izsaucot kādu no minētajām funkcijām, ziņojuma pamattekstā ir papildus jāpadod konkrētajai funkcijai nepieciešamie ievaddati, ja tādi ir. Šādu ievaddatu ierobežojumus skatīt pie attiecīgas funkcijas apraksta.
Apstrāde	Izsaucot funkciju, izmantojot ievaddatu informāciju, tiek izveidots pieprasījums vārtejas API modulim. Ja tiek padots nederīgs REST servisa nosaukums, tiek izvadīts 1. kļūdas ziņojums. Citādi tiek atgriezti attiecīgās funkcijas izvaddati.
Izvade	Izsauktās vārtejas API moduļa funkcijas izvaddati vai attiecīgais kļūdas ziņojums.
Kļūdu ziņojumi	1. “Code: 1003 Message: Invalid method name!”

1.3.8. Datu validācijas modulis

1.3.8.1. Datu pārsūtīšanas objektu validācija

Identifikators	DVM-VDTO
Apraksts	Funkcija nodrošina to, ka datubāzē tiks saglabāti tikai objekti ar korektām lauku vērtībām.
Ievade	Servera puses validācijā objekts, kura validācija jāveic, tiek saņemts no vārtejas API moduļa funkcijas GAM-SSDV. Klienta puses validācijā objekts, kura validācija jāveic, tiek saņemts no klienta bibliotēkas moduļa funkcijas CLM-CSDV.
Apstrāde	Funkcijai padotā objekta lauki tiek pārbaudīti pēc nosacījumiem, kas minēti vārtejas API moduļa funkcijās: 1. Organizāciju objektiem - GAM-OCNO 2. Atrašanās vietu objektiem - GAM-LCNL 3. Ierīču objektiem - GAM-DCND 4. Notikumu objektiem - GAM-ECNE Ja kaut viens objekta lauks neatbilst nosacījumiem, objekts tiek uzskatīts par nederīgu. Ja funkcijai tiek padots kāda objekta ID datubāzē, tad šim ID ir jābūt 24 simbolus garai heksadecimālai virknei.
Izvade	Ja visi objekta lauki atbilst nosacījumiem, tiek atgriezta vērtība

	"true". Ja kaut viens lauks neatbilst nosacījumiem, tiek atgriezts 1. kļūdu ziņojums.
Kļūdu ziņojumi	1. "Code: 1001 Message: Invalid <*lauka nosaukums*>!"

1.3.9. REST API dokumentācijas modulis

1.3.9.1. REST servisu dokumentācijas ģenerēšana

Identifikators	RDM-RSDG
Apraksts	Funkcija izmanto vārtejas API modulī definētos REST servisus, lai ģenerētu šo servisu dokumentāciju.
Ievade	Funkcijas ievaddati ir vārtejas API pirmkodā iekļautās anotācijas.
Apstrāde	Vārtejas API moduļa būvēšanas laikā, izmantojot pirmkodā iekļautās anotācijas, tiek ģenerēts JSON fails, kurā ir iekļauta informācija par katra vārtejas API moduļa REST servisa izsaukšanas ceļu (path), ievaddatiem, ja tādi ir, un to formātu, kā arī atgrieztajiem datiem un to formātu. Ja ģenerējot dokumentāciju, rodas kāda kļūda, tiek izvadīts 1. kļūdas ziņojums.
Izvade	JSON fails, kas satur visu vārtejas API REST servisu un tajos izmantoto objektu definīcijas vai arī attiecīgais kļūdas ziņojums.
Kļūdu ziņojumi	1. "Code: 1010 Message: Failed to build JSON specification"

1.3.9.2. REST servisu dokumentācijas attēlošana

Identifikators	RDM-RSDD
Apraksts	Funkcija izmanto no vārtejas API modulī definētajiem REST servisiem ģenerēto dokumentāciju, lai attēlotu šo dokumentāciju lietotāja saskarnē.
Ievade	Funkcijas ievaddati: 1. Funkcijā RDM-RSDG ģenerētais JSON fails
Apstrāde	Izmantojot dokumentāciju JSON failā, tiek izveidota caur tīmekļa pārlūku pieejama dokumentācijas mājaslapa, kurā iekļauti visi vārtejas API definētie REST servisi, kā arī tajos izmantoto datu objektu definīcijas. Izveidotajā lapā jābūt iespējai arī izmēģināt katra servisa darbību. Ja veidojot dokumentācijas attēlojumu, ģenerēto JSON failu nevar nolasīt vai iegūt, tad tiek izvadīts 1. kļūdu

	ziņojums.
Izvade	Tiek atgriezta izveidotā dokumentācijas tīmekļa lapa, vai arī attiecīgais kļūdas ziņojums.
Kļūdu ziņojumi	1. "Code: 1010 Message: Failed to read JSON specification"

1.4. Nefunkcionālās prasības

1.4.1. Drošības prasības

API pārvaldības moduļa datubāzē glabātās autentifikācijai nepieciešamās paroļu jaucējvērtības (hash values) tiks veidotas, izmantojot PBKDF2WithHMAC-SHA1 algoritmu un "sāli" (skatīt sadaļu "Apzīmējumu saraksts"). Lietotāju parolēm jābūt vismaz 8 simbolus garām un jāsatur vismaz vienu lielo burtu, vismaz vienu ciparu un vismaz vienu simbolu.

Piekļuvei sistēmas funkcijām jānotiek caur reverso starpniekserveri, lai sistēmas iekšējā tīkla topoloģija nebūtu redzama no ārpuses.

Datu drošībai tiek izvirzīta prasība, ka sistēmā ir jābūt iespējai atjaunot galvenās dokumentu datubāzes saturu datu pazaudēšanas gadījumā.

1.4.2. Veiktspējas prasības

Sistēmai jābūt mērogojamai, lai neparedzēti lielas slodzes gadījumos mazāk kā stundas laikā varētu atgūt nemainīgu pakalpojumu kvalitāti. Tā kā izstrādājamā sistēma ir prototips, tad konkrētākas veiktspējas prasības netiek izvirzītas.

1.4.3. Uzturamības prasības

Sistēmai jābūt būvētai modulāri, lai nepieciešamības gadījumā jaunu funkcionalitāti varētu realizēt atsevišķā sistēmas modulī.

1.4.4. Apjoma prasības

Aparatūrai, uz kuras tiks darbināta sistēma, būtu nepieciešami vismaz 2 GB operatīvās atmiņas, kā arī jābūt iespējai paplašināt fizisko atmiņu.

1.5. Ārējā saskarne

1.5.1. Lietotāja saskarne

Tā kā sistēmas lietotāji būs dažādu jomu pārstāvji, kur katrai lietotāju grupai savā darbā ir svarīga cita informācija, kā arī tāpēc, ka sistēma ir prototips, netiek izvirzīta prasība pēc vienota lietotāja saskarnes stila. Tas ļautu vajadzības gadījumā sistēmā izmantot konfigurējamus atvērtā pirmkoda risinājumus, kuri apmierina izvirzītās sistēmas prasības.

Visām lietotāju saskarnēm jābūt pieejamām, izmantojot tīmekļa pārlūkprogrammas.

1.5.2. Aparatūras saskarne

Tā kā sistēmu plānots izvietot uz mākoņpakalpojumu platformām, tad netiek izvirzītas prasības par konkrētas aparatūras lietošanu. Sistēmai jābūt izvietojamai uz vairāk nekā vienas mākoņpakalpojumu platformas.

1.5.3. Programmatūras saskarne

Sistēmai jābūt savietojamai ar vairākās programmatūras valodās rakstītām klientu integrācijām, pieņemot, ka šīs integrācijas izmanto sistēmā iekļauto klienta bibliotēku. Netiek garantēta sistēmas korekta sadarbība ar integrācijām, kurās izmantotas paštaisītas klientu bibliotēkas.

1.5.4. Sakaru saskarne

Platformas vārtejas API moduļa REST servisi izmantos HTTP protokolu datu apmaiņā. HTTP protokols tiks izmantots arī saziņā ar sistēmas datubāzēm, kā arī platformas saziņā ar klientu integrācijām.

2. Programmatūras projektējuma apraksts

2.1. Ievads

2.1.1. Dokumenta nolūks

Šajā dokumentā aprakstīts izstrādājamās sistēmas projektējums, pēc kura tiks realizētas šīs sistēmas prasību specifikācijā iekļautās funkcijas. Dokuments ir paredzēts tajā aprakstītās sistēmas izstrādātājiem.

2.1.2. Darbības sfēra

Izstrādājamā produkta nosaukums ir "Paplašināma IoT platforma datu un notikumu apkopošanai". Produkta mērķis ir apkopot datus no dažādiem sensoriem un šos datus vizualizēt, lai ļautu labāk saskatīt tendences, anomālijas un dažādu problēmu cēloņus. Produktu paredzēts piedāvāt apdrošināšanas organizācijām un tas ļautu tām apkopot datus no klientu mājās izvietotiem sensoriem, lai iegūtu klientu izpratni (customer insight), atrastu biežākās problēmas un to cēloņus, tādējādi uzlabojot situāciju gan klientiem, gan apdrošinātājiem.

2.1.3. Saistība ar citiem dokumentiem

Dokuments veidots, balstoties uz standartu LVS 72:1996 "Ieteicamā prakse programmatūras projektējuma aprakstīšanai", kā arī uz PPS izvirzītajām programmatūras prasībām.

2.2. Dekompozīcijas apraksts

2.2.1. Moduļu dekompozīcija

Šajā sadaļā tiek aprakstīts sistēmas dalījums moduļos, katra moduļa nolūks, kā arī tas, kuras no PPS definētajām funkcionālajām prasībām tiek realizētas katrā modulī. Atkarības starp šeit aprakstītajiem sistēmas moduļiem var apskatīt 2.3.1. nodaļā "Starpmoduļu un starpprocesu atkarības".

2.2.1.1. Datu validācijas modulis

Moduļa nolūks	Modulis paredzēts klienta un servera puses validācijas funkciju īstenošanai, lai samazinātu tīkla noslodzi, kā arī lai nodrošinātu, ka datubāzē tiek saglabāti tikai derīgi objekti.
Moduļa funkcijas	1. Datu pārsūtīšanas objektu validācija (1.3.8.1)

2.2.1.2. Vārtejas API modulis

Moduļa nolūks	Modulis paredzēts sistēmai nepieciešamo REST API servisu realizācijai. Tas ļauj klientu integrācijām veikt darbības datubāzē ar organizāciju, atrašanās vietu, ierīču un notikumu objektiem, kā arī iegūt datubāzē glabātos ierīču novietojuma klasifikatorus. Modulis nodrošina arī datu dublēšanu jēldatu datubāzē, kā arī datu validāciju servera pusē.
Moduļa funkcijas	1. Organizāciju ierakstu saglabāšana datubāzē (1.3.1.1) 2. Visu organizāciju ierakstu iegūšana no datubāzes (1.3.1.2) 3. Organizācijas atrašana datubāzē pēc ieraksta ID (1.3.1.3) 4. Datubāzē saglabāto organizāciju ID iegūšana (1.3.1.4) 5. Organizācijas ieraksta ID atrašana pēc organizācijas ID API pārvaldības modulī (1.3.1.5) 6. Atrašanās vietu saglabāšana datubāzē (1.3.1.6) 7. Atrašanās vietu iegūšana no datubāzes (1.3.1.7) 8. Datubāzē saglabāto atrašanās vietu ID iegūšana (1.3.1.8) 9. Ierīču saglabāšana datubāzē (1.3.1.9) 10. Ierīču iegūšana no datubāzes (1.3.1.10) 11. Datubāzē saglabāto ierīču ID iegūšana (1.3.1.11) 12. Notikumu saglabāšana datubāzē (1.3.1.12) 13. Notikumu iegūšana no datubāzes (1.3.1.13) 14. Datubāzē saglabāto notikumu ID iegūšana (1.3.1.14) 15. Jēldatu dokumenta saglabāšana datubāzē (1.3.1.15) 16. Ierīču novietojuma klasifikatoru (tags) iegūšana no datubāzes (1.3.1.16) 17. Ierīču novietojuma klasifikatoru (tags) importēšana datubāzē no JSON failiem (1.3.1.17) 18. Datu validācija servera pusē (1.3.1.18)

2.2.1.3. Klienta bibliotēkas modulis

Moduļa nolūks	Modulis paredzēts, lai atvieglotu klientu integrāciju izstrādi. Tas satur HTTP klientu, kas pielāgots, lai sadarbotos ar vārtejas API piedāvātajiem REST API. Modulis paredzēts lietošanai ar Apache

	Camel integrācijas satvaru. Šis modulis atbild arī par datu validāciju klienta pusē.
Moduļa funkcijas	1. Klienta puses datu validācija (1.3.7.1) 2. Vārtejas API moduļa REST servisu izsaukšana (1.3.7.2)

2.2.1.4. Starpprogrammatūras modulis

Moduļa nolūks	Modulis paredzēts, lai nodrošinātu vārtejas API moduļa datu apmaiņu ar dokumentu un jēldatu datubāzēm.
Moduļa funkcijas	1. Datu pieejas objektu izveide (1.3.2.1) 2. Datu ielasīšana no datubāzes (1.3.2.2) 3. Datu ierakstīšana datubāzē (1.3.2.3)

2.2.1.5. Kartēšanas modulis

Moduļa nolūks	Modulis nepieciešams, lai veiktu vārtejas API moduļa datu objektu pārveidošanu par starpprogrammatūras moduļa datu objektiem, un otrādi.
Moduļa funkcijas	1. Datu pārsūtīšanas objektu (DTO) pārveidošana starp sistēmas moduļiem (1.3.3.1)

2.2.1.6. API pārvaldības modulis

Moduļa nolūks	Modulis paredzēts, lai padarītu vārtejas API piedāvātos REST API servisus pārvaldāmus, atļautu tikai autentificētu lietotāju pieeju šiem servisiem, kā arī lai uzskaitītu katras klientu organizācijas API lietojumu.
Moduļa funkcijas	1. Autentifikācija (1.3.5.1) 2. Organizācijas izmantotā datu daudzuma mērīšana (1.3.5.2) 3. Organizācijai pieejamā pieprasījumu skaita ierobežošana (throttling) (1.3.5.3) 4. Lietotāja organizācijas ID iegūšana pēc lietotājvārda (1.3.5.4)

	5. Lietotāja reģistrācija (1.3.5.5) 6. Aizmirstas paroles nomaiņa (1.3.5.6) 7. Organizācijas reģistrācija API pārvaldības modulī (1.3.5.7) 8. API reģistrācija API pārvaldības modulī (1.3.5.8) 9. API lietojuma plānu izveidošana (1.3.5.9) 10. API klienta lietotnes reģistrācija (1.3.5.10)

2.2.1.7. Reversā starpniekservera modulis

Moduļa nolūks	Modulis paredzēts, lai uzlabotu sistēmas drošību, noslēpjot tās iekšējo topoloģiju no publiski pieejamās tīmekļa daļas, kā arī lai nodrošinātu ienākošo pieprasījumu maršrutēšanu uz pieprasītajiem platformas servisiem.
Moduļa funkcijas	1. Pieprasījumu nodošana attiecīgajam servisam platformā (1.3.6.1)

2.2.1.8. REST API dokumentācijas ģenerēšanas modulis

Moduļa nolūks	Modulis paredzēts, lai no vārtejas API moduļa pirmkoda ģenerētu modulī iekļauto REST servisu dokumentāciju un to attēlotu tīmekļa saskarnē.
Moduļa funkcijas	1. REST servisu dokumentācijas ģenerēšana (1.3.9.1) 2. REST servisu dokumentācijas attēlošana (1.3.9.2)

2.2.1.9. Datu vizualizācijas modulis

Moduļa nolūks	Modulis paredzēts, lai platformā būtu iespēja lietotājiem veidot platformas jēldatu datubāzē saglabāto datu vizualizācijas, kā arī grupēt šīs vizualizācijas informācijas paneļos.
Moduļa funkcijas	1. Datu vizualizāciju izveide (1.3.4.1) 2. Datu vizualizāciju apkopošana informācijas paneļos (1.3.4.2)

2.2.2. Datu dekompozīcija

Platformai kopumā ir 3 datubāzes - dokumentu datubāze, jēldatu datubāze un API pārvaldības datubāze. Dokumentu datubāzes kolekciju aprakstus var atrast 2.1. tabulā, jēldatu datubāzes tabulu aprakstus - 2.2. tabulā. Jēldatu datubāzē tiks glabāti arī datu vizualizācijas modulī izveidotie informācijas paneļi un vizualizācijas, bet kolekciju, kurā šie objekti tiks glabāti izveido pati sistēma, tādēļ šī kolekcija netiks aprakstīta. Platformas ierīcēm paredzētie novietojuma klasifikatori, kas datubāzē tiks importēti pie sistēmas palaišanas, tiks glabāti JSON failos, jo sistēmas darbības laikā nav paredzēts, ka tie mainīsies. API pārvaldības moduļa datubāze šeit netiks aprakstīta, jo ir paredzēts API pārvaldības moduļa prasību realizācijai izmantot konfigurējamu atvērtā pirmkoda sistēmu API Man, kurā ir iekļauta moduļa entītiju glabāšanai paredzēta datubāze.

2.1.tabula **Dokumentu datubāzes kolekcijas un to apraksti**

Kolekcija	Apraksts
organisations	Glabā sistēmā reģistrēto organizāciju informāciju
locations	Glabā sistēmā reģistrētajām organizācijām piesaistīto atrašanās vietu informāciju
devices	Glabā sistēmā reģistrētajām atrašanās vietām piesaistīto ierīču (sensoru) informāciju
events	Glabā sistēmā reģistrētajām ierīcēm piesaistīto notikumu (piem., mērījumu izmaiņu) informāciju.
tags	Glabā sistēmā definētos ierīču novietojuma klasifikatorus

2.2.tabula **Jēldatu datubāzes indeksi un to apraksti**

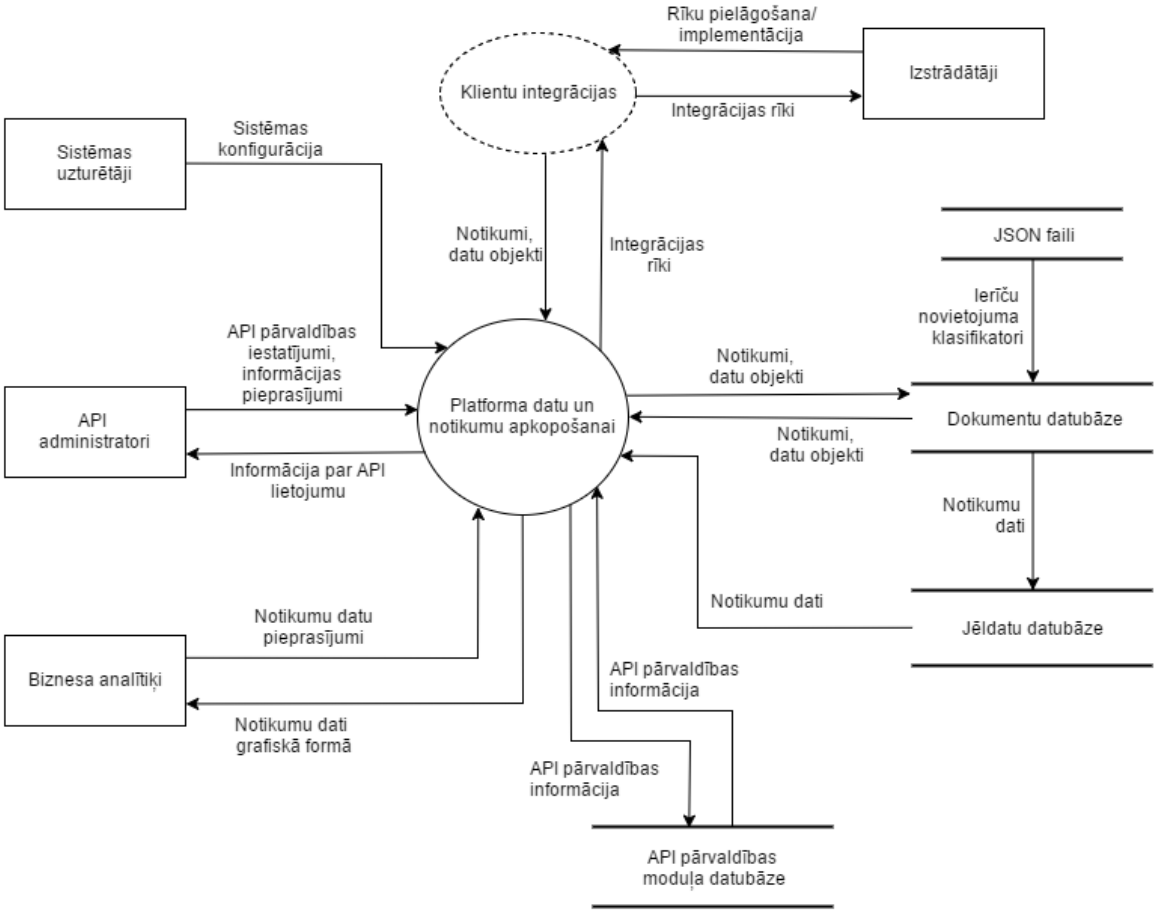
Indekss	Apraksts
organisation/rawdata	Glabā jēldatu dokumentus, kuri tiek izveidoti pie katra notikuma saglabāšanas dokumentu datubāzē un satur informāciju par notikumu, ierīci, atrašanās vietu un organizāciju

2.3. Atkarību apraksts

2.3.1. Starpmoduļu un starpprocesu atkarības

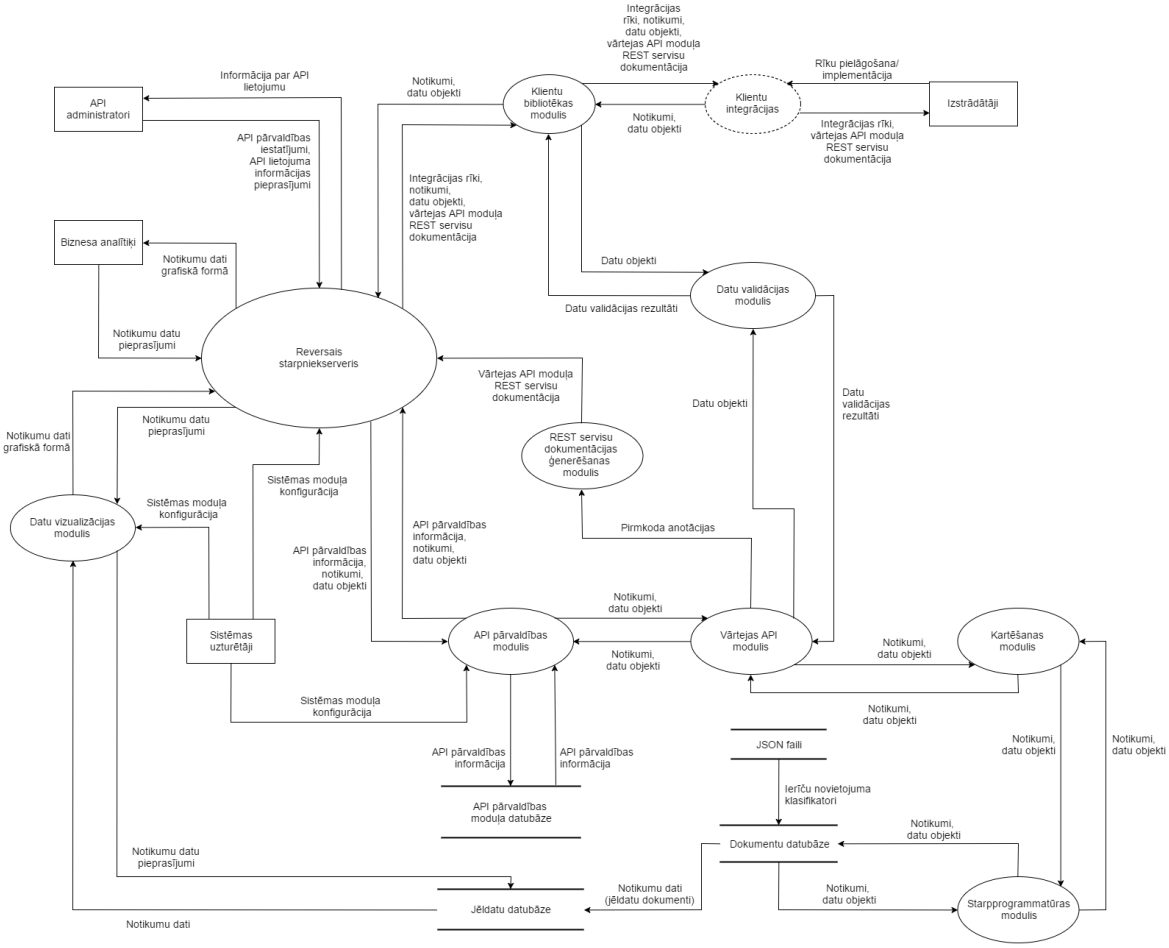
Šajā sadaļā ir iekļautas sistēmas datu plūsmu diagrammas (DPD). 0. līmeņa diagrammā (2.1. att.) redzams, kādi dati pārvietosies starp platformu, lietotājiem, ārējām sistēmām un platformas datubāzēm. 1. līmeņa diagrammā (2.2. att.) tiek izvērsts sistēmas dalījums moduļos, bet 2. līmeņa diagrammās tiek attēlota katra moduļa iekšējā darbība.

2.3.1.1. Sistēmas 0. līmeņa datu plūsmu diagramma



2.1.att. **Platformas 0. līmeņa DPD**

2.3.1.2. Sistēmas 1. līmeņa datu plūsmu diagramma

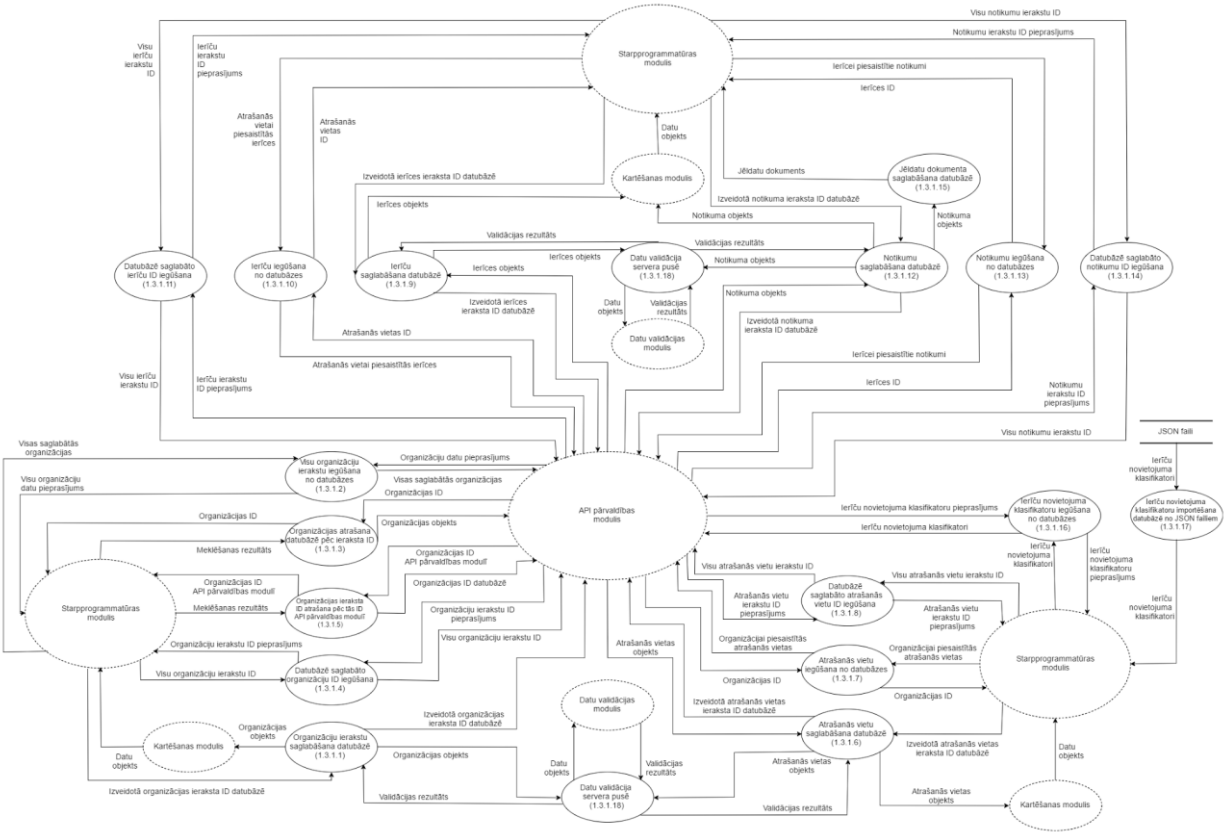


2.2.att. **Platformas 1. līmeņa DPD**

2.3.1.3. Datu validācijas moduļa 2. līmeņa datu plūsmu diagramma

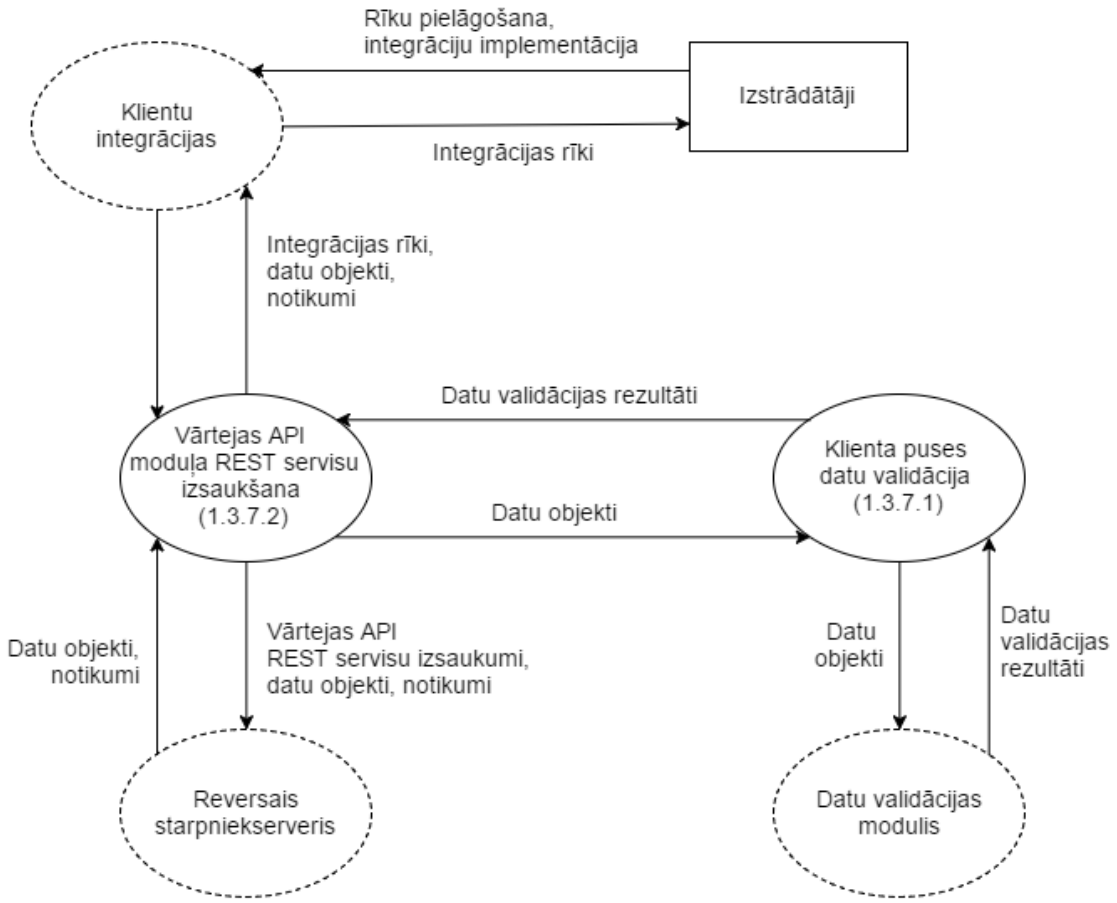
2.3.att. **Datu validācijas moduļa 2. līmeņa DPD**

2.3.1.4. Vārtejas API moduļa 2. līmeņa datu plūsmu diagramma



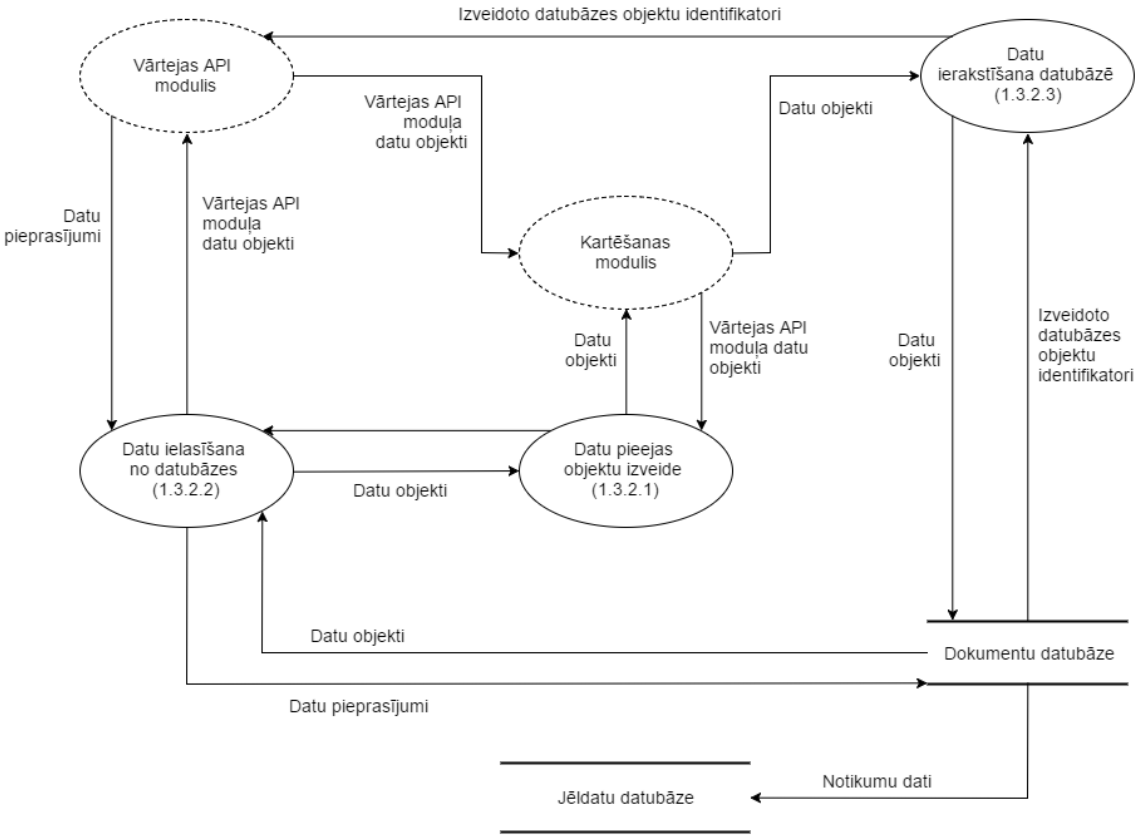
2.4.att. **Vārtejas API moduļa 2. līmeņa DPD**

2.3.1.5. Klienta bibliotēkas moduļa 2. līmeņa datu plūsmu diagramma



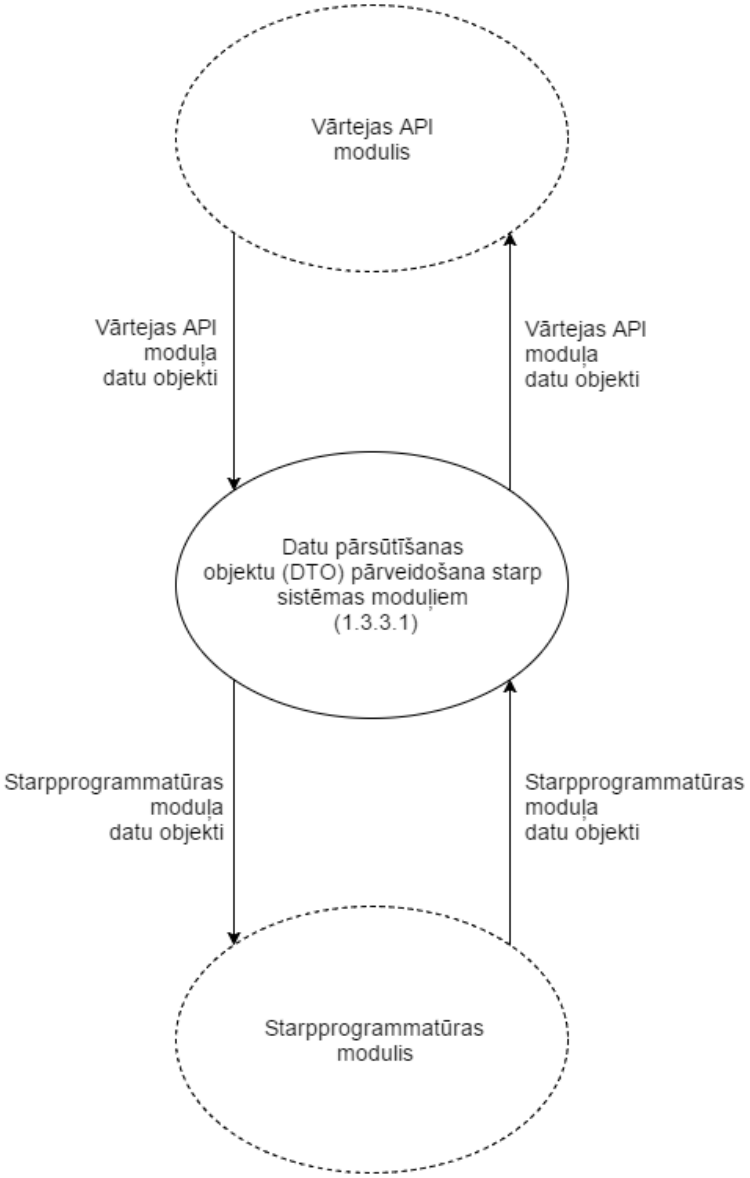
2.5.att. **Klienta bibliotēkas moduļa 2. līmeņa DPD**

2.3.1.6. Starpprogrammatūras moduļa 2. līmeņa datu plūsmu diagramma



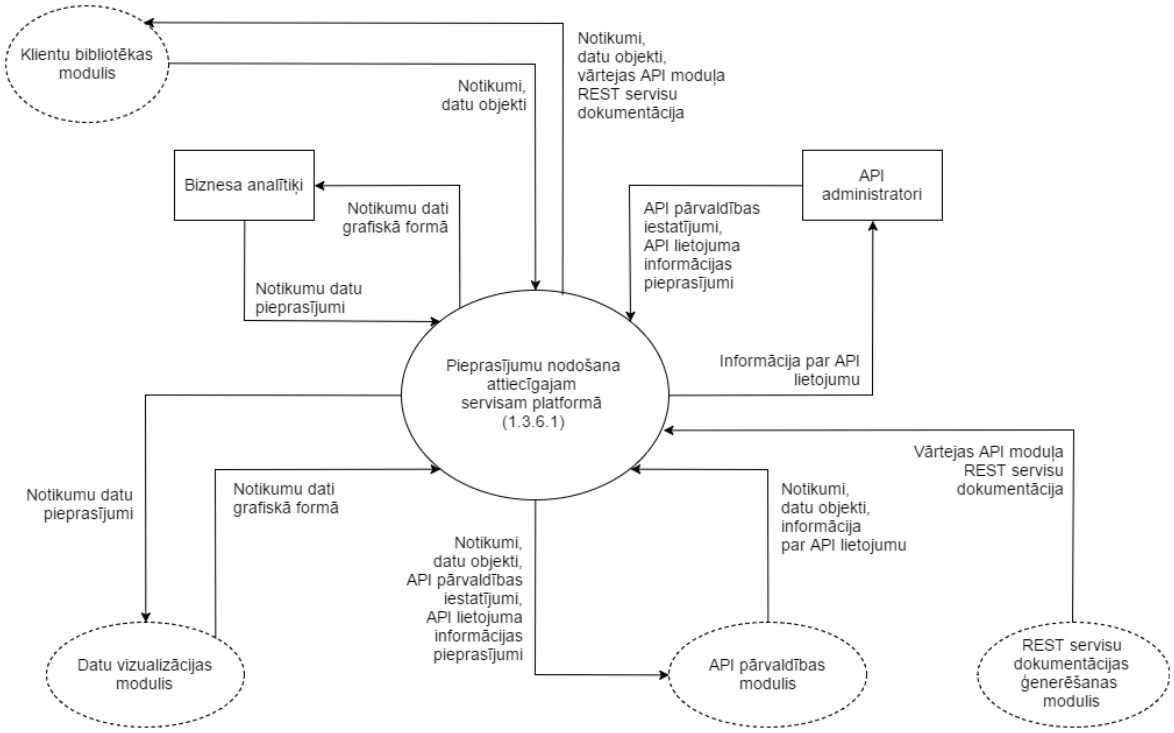
2.6.att. **Starpprogrammatūras moduļa 2. līmeņa DPD**

2.3.1.7. Kartēšanas moduļa 2. līmeņa datu plūsmu diagramma



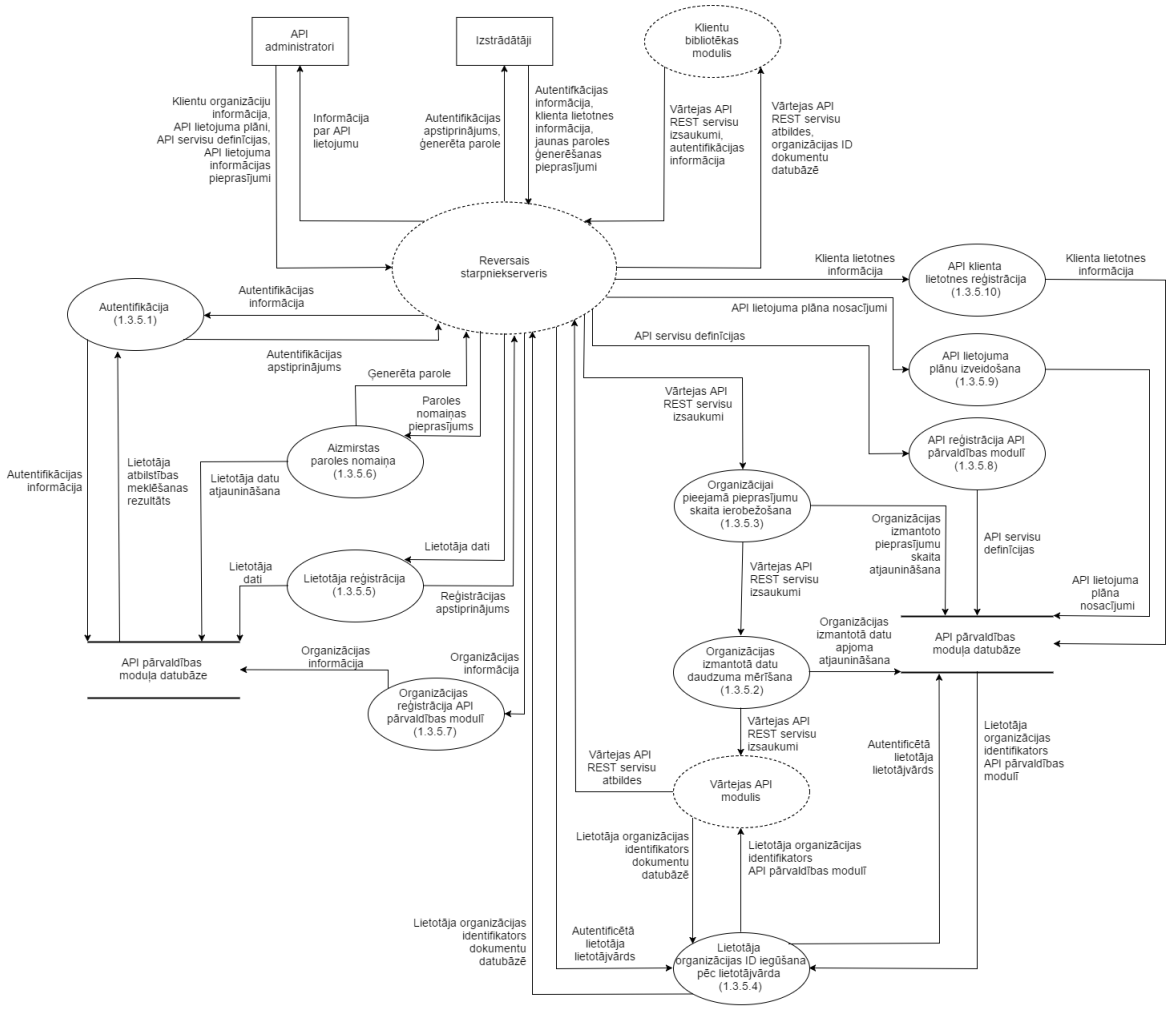
2.7.att. **Kartēšanas moduļa 2. līmeņa DPD**

2.3.1.8. Reversā starpniekservera moduļa 2. līmeņa datu plūsmu diagramma



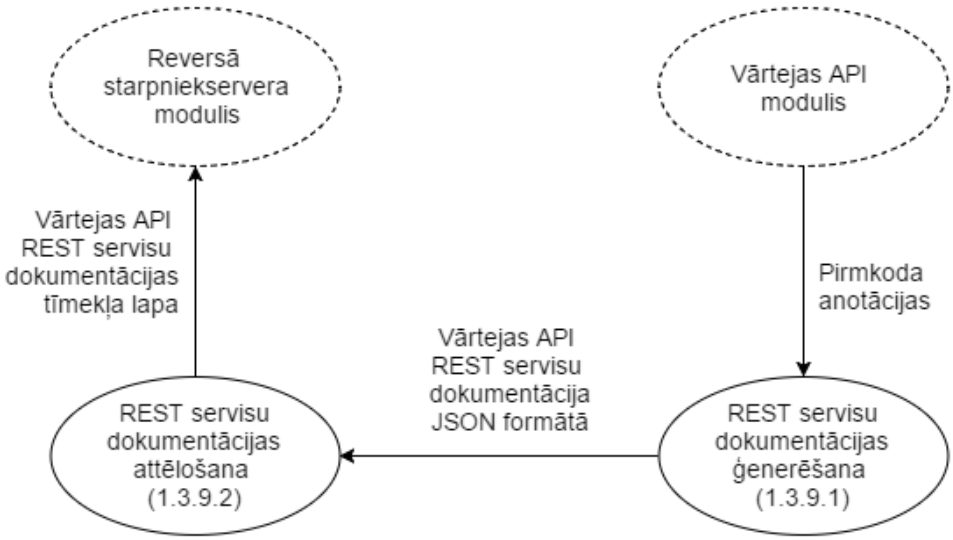
2.8.att. **Reversā starpniekservera moduļa 2. līmeņa DPD**

2.3.1.9. API pārvaldības moduļa 2. līmeņa datu plūsmu diagramma



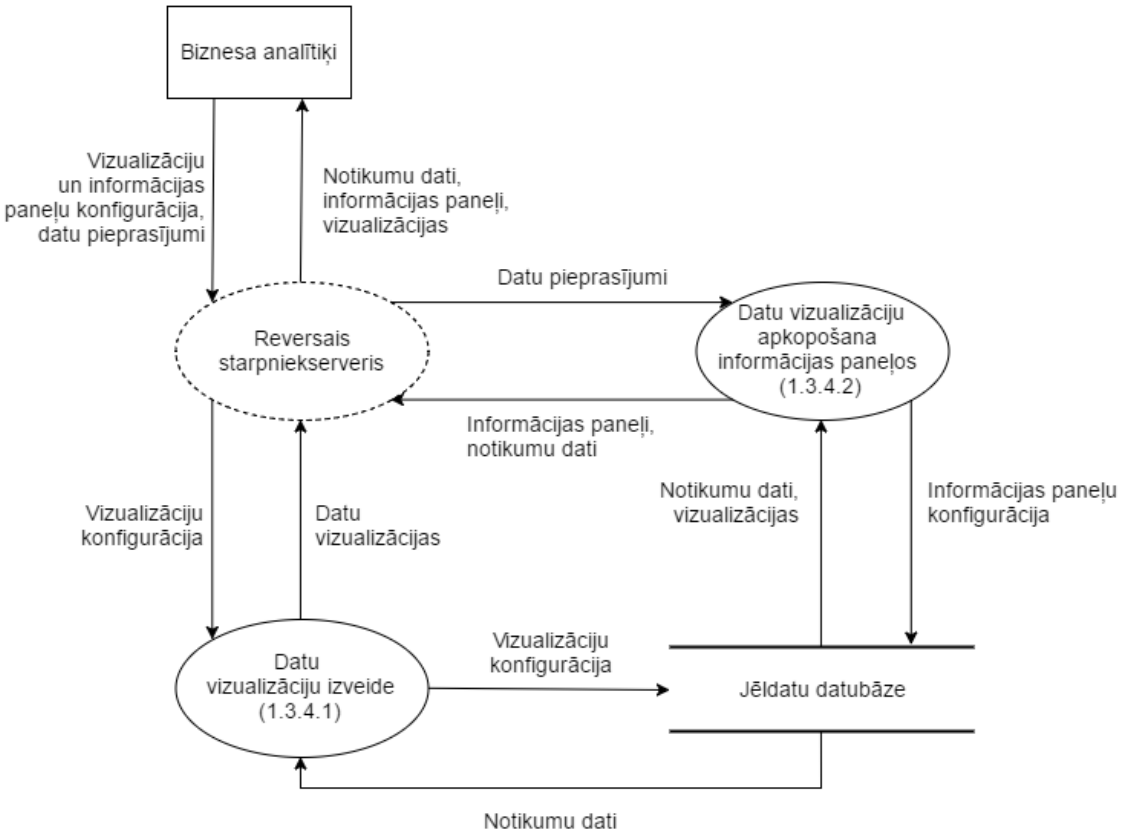
2.9.att. **API pārvaldības moduļa 2. līmeņa DPD**

2.3.1.10. REST servisu dokumentācijas ģenerēšanas moduļa 2. līmeņa datu plūsmu diagramma



2.10.att. **REST servisu dokumentācijas ģenerēšanas moduļa 2. līmeņa DPD**

2.3.1.11. Datu vizualizācijas moduļa 2. līmeņa datu plūsmu diagramma

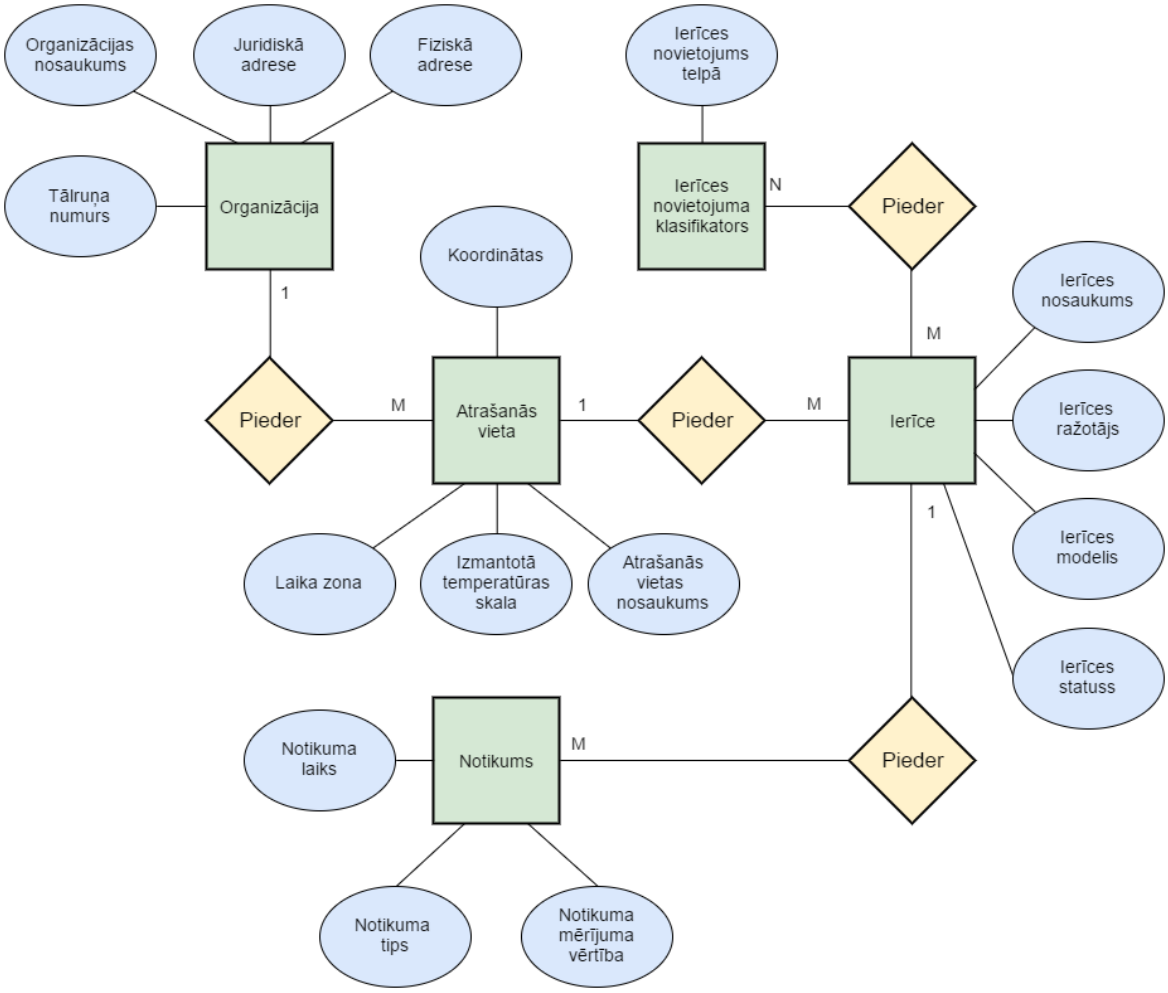


2.11.att. **Datu vizualizācijas moduļa 2. līmeņa DPD**

2.3.2. Datu atkarības

Jēldatu datubāzē starp glabātajiem dokumentiem nepastāv loģiskas atkarības. Katrs dokuments šajā datubāzē ir neatkarīga datu vienība.

Tā kā dokumentu datubāzē nav mehānismu, lai ieviestu datubāzes shēmu, tad datu integritāte tiek nodrošināta ar klienta un servera puses validācijas funkcijām. Starp dokumentu datubāzē glabātajām entītijām pastāvošās atkarības ir attēlotas 2.12. attēlā redzamajā ER diagrammā. Detalizētāku platformas entītiju aprakstu skatīt 2.4.1. nodaļā "Datu detalizēts projektējums".



2.12.att. **Dokumentu datubāzes konceptuālais ER modelis**

2.4. Detalizēts projektējums

2.4.1. Datu detalizēts projektējums

Dokumentu datubāzes kolekciju aprakstā kā datu tipi tiks izmantoti BSON tipi, jo šādā formātā dati tiek glabāti datubāzē. Lauki, kuri satur strukturētus objektus, tiks aprakstīti formā "*<lauka nosaukums>.<objekta iekšējā lauka nosaukums>*".

2.4.1.1. Kolekcija "organisations"

Šī dokumentu datubāzes kolekcija satur informāciju par platformā reģistrētām klientu organizācijām.

2.3.tabula **Kolekcijas "organisations" lauki un to tipi**

Lauks	**Datu tips**	**Atslēga**	**Apraksts**
_id	ObjectID	PK	MongoDB ģenerēts ieraksta identifikators
name	String		Organizācijas nosaukums
apiOrgId	String		Organizācijas identifikators API pārvaldības modulī
legalAddress.country	String		Organizācijas juridiskās adreses valsts
legalAddress.city	String		Organizācijas juridiskās adreses pilsēta
legalAddress.street	String		Organizācijas juridiskās adreses iela
legalAddress.number	32-bit integer		Organizācijas juridiskās adreses numurs
physicalAddress.country	String		Organizācijas fiziskās adreses valsts
physicalAddress.city	String		Organizācijas fiziskās adreses pilsēta
physicalAddress.street	String		Organizācijas fiziskās adreses iela
physicalAddress.number	32-bit integer		Organizācijas fiziskās adreses numurs
telephone	String		Organizācijas kontakttālrunis

2.4.1.2. Kolekcija “locations”

Šī dokumentu datubāzes kolekcija satur informāciju par atrašanās vietām, kuras ir piesaistītas platformā reģistrētajām organizācijām.

2.4.tabula **Kolekcijas “locations” lauki un to tipi**

Lauks	**Datu tips**	**Atslēga**	**Apraksts**
_id	ObjectID	PK	MongoDB ģenerēts ieraksta identifikators
organisationId	String	FK	Atrašanās vietai piesaistītās organizācijas ID
locationName	String		Atrašanās vietas nosaukums
temperatureScale	String		Atrašanās vietas izmantotā temperatūras skala (‘C’ - Celsija skala, ‘F’ - Fārenheita skala)
mode.modeId	64-bit integer		Atrašanās vietas režīma identifikators (0, 1 vai 2)
mode.modeName	String		Atrašanās vietas režīma nosaukums (“HOME”, “AWAY”, “UNKNOWN”)
timeZone	String		Atrašanās vietas izmantotā laika zona
latitude	Double		Atrašanās vietas ģeogrāfiskais platums
longitude	Double		Atrašanās vietas ģeogrāfiskais garums

2.4.1.3. Kolekcija “devices”

Šī dokumentu datubāzes kolekcija satur informāciju par ierīcēm, kuras ir piesaistītas platformā reģistrētajām atrašanās vietām.

2.5.tabula **Kolekcijas “devices” lauki un to tipi**

Lauks	**Datu tips**	**Atslēga**	**Apraksts**
_id	ObjectID	PK	MongoDB ģenerēts ieraksta identifikators
locationId	String	FK	Ierīcei piesaistītās atrašanās vietas identifikators

externalId	String		Ierīces ražotāja API izmantotais ierīces identifikators
deviceName	String		Ierīces nosaukums
status	String		Ierīces statuss (“OK”, “DOWN”, “UNKNOWN”)
manufacturer	String		Ierīces ražotājs
model	String		Ierīces modelis
tagIds	Array		Saraksts ar ierīces novietojuma klasifikatoru identifikatoriem, satur tieši 3 identifikatorus

2.4.1.4. Kolekcija “events”

Šī dokumentu datubāzes kolekcija satur informāciju par notikumiem, kuri ir piesaistīti platformā reģistrētajām ierīcēm.

2.6.tabula **Kolekcijas “events” lauki un to tipi**

Lauks	**Datu tips**	**Atslēga**	**Apraksts**
_id	ObjectID	PK	MongoDB ģenerēts ieraksta identifikators
deviceId	String	FK	Notikumam piesaistītās ierīces ID
date	Date		Notikuma laiks un datums
eventName	String		Notikuma nosaukums, paredzēts lai noteiktu notikuma tipu (temperatūra, mitrums utml.)
eventValue	Double		Notikuma mērījuma vērtība
description	String		Notikuma apraksts, sniedz plašāku informāciju par notikumu
eventMode	String		Notikuma režīms (“HOME”, “AWAY”, “UNKNOWN”)

2.4.1.5. Kolekcija “tags”

Šī dokumentu datubāzes kolekcija satur informāciju par ierīču novietojuma klasifikatoriem.

2.7.tabula **Kolekcijas “tags” lauki un to tipi**

Lauks	**Datu tips**	**Atslēga**	**Apraksts**
_id	64-bit integer	PK	Klasifikatora identifikators, atšķirībā no citām entītijām, klasifikatoru identifikatorus norāda JSON failos, no kuriem klasifikatori tiek importēti
tagName	String		Klasifikatora nosaukums (norāda uz ierīces atrašanās vietu telpā)

2.4.1.6. Indekss “organisation/rawdata”

Šis jēldatu datubāzes indekss satur sistēmā saglabātos jēldatu dokumentus. Katrs dokuments satur informāciju par notikumu, ierīci, atrašanās vietu un organizāciju. Indeksā faktiski tiek apkopoti visās dokumentu datubāzes kolekcijās atrodamie lauki. Lauki, kuru nosaukumi sākas ar apakšsvītru, ir dokumenta metainformācijas lauki.

2.8.tabula **Indeksa “organisation/rawdata” lauki un to tipi**

Lauks	**Datu tips**	**Apraksts**
_id	text	Dokumenta identifikators (PK)
_index	text	Indeksa nosaukums, kurā atrodas dokuments (tabulas ekvivalents Elasticsearch datubāzē)
_type	text	Dokumenta tips, kas ir sīkāks sadalījums indeksa ietvaros
organisation.name	text	Organizācijas nosaukums
organisation.apiOrgId	text	Organizācijas identifikators API pārvaldības modulī
organisation.legalAddress.country	text	Organizācijas juridiskās adreses valsts
organisation.legalAddress.city	text	Organizācijas juridiskās adreses pilsēta
organisation.legalAddress.street	text	Organizācijas juridiskās adreses iela

organisation.legalAddress.number	long	Organizācijas juridiskās adreses numurs
organisation.physicalAddress.country	text	Organizācijas fiziskās adreses valsts
organisation.physicalAddress.city	text	Organizācijas fiziskās adreses pilsēta
organisation.physicalAddress.street	text	Organizācijas fiziskās adreses iela
organisation.physicalAddress.number	long	Organizācijas fiziskās adreses numurs
organisation.telephone	text	Organizācijas kontakttālrunis
location.organisationId	text	Atrašanās vietai piesaistītās organizācijas ID
location.locationName	text	Atrašanās vietas nosaukums
location.temperatureScale	text	Atrašanās vietas izmantotā temperatūras skala (‘C’ - Celsija skala, ‘F’ - Fārenheita skala)
location.mode.modeId	long	Atrašanās vietas režīma identifikators (0, 1 vai 2)
location.mode.modeName	text	Atrašanās vietas režīma nosaukums (“HOME”, “AWAY”, “UNKNOWN”)
location.timeZone	text	Atrašanās vietas izmantotā laika zona
location.latitude	float	Atrašanās vietas ģeogrāfiskais platums
location.longitude	float	Atrašanās vietas ģeogrāfiskais garums
device.locationId	text	Ierīcei piesaistītās atrašanās vietas identifikators
device.externalId	text	Ierīces ražotāja API izmantotais ierīces identifikators
device.deviceName	text	Ierīces nosaukums
device.status	text	Ierīces statuss (“OK”, “DOWN”, “UNKNOWN”)
device.manufacturer	text	Ierīces ražotājs
device.model	text	Ierīces modelis

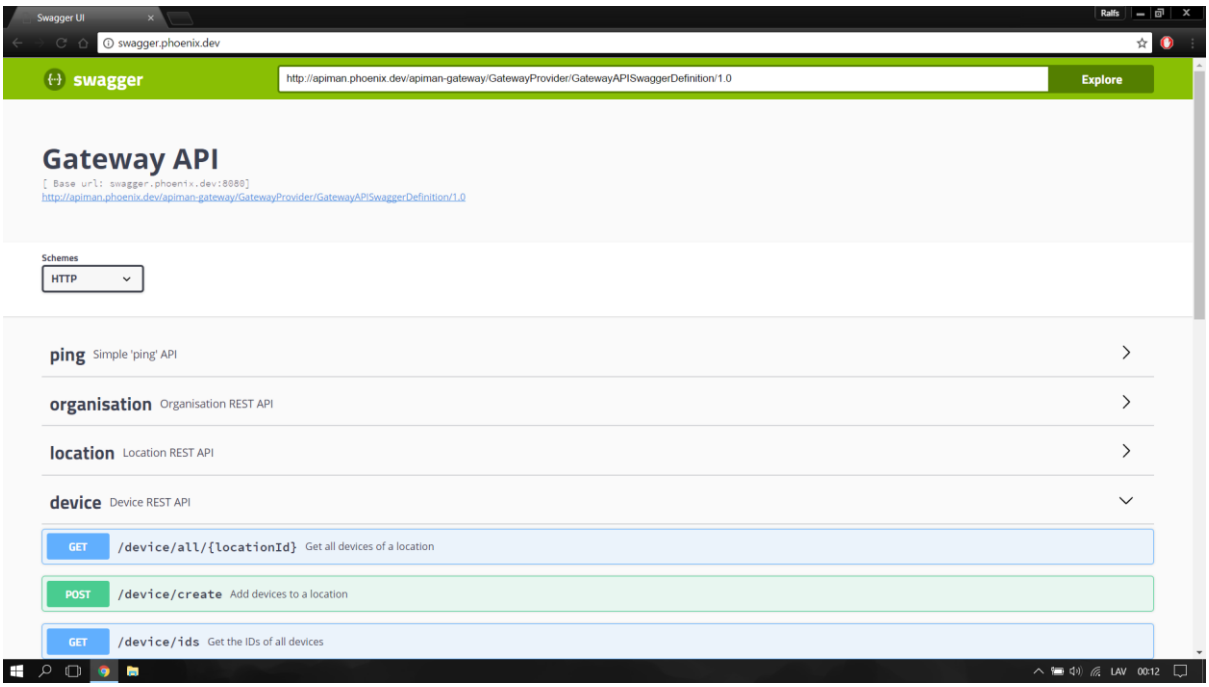
device.tagIds	long	Saraksts ar ierīces novietojuma klasifikatoru identifikatoriem, satur tieši 3 identifikatorus
event.deviceId	text	Notikumam piesaistītās ierīces ID
event.date	date	Notikuma laiks un datums
event.eventName	text	Notikuma nosaukums, paredzēts lai noteiktu notikuma tipu (temperatūra, mitrums utml.)
event.eventValue	float	Notikuma mērījuma vērtība
event.description	text	Notikuma apraksts, sniedz plašāku informāciju par notikumu
event.eventMode	text	Notikuma režīms (“HOME”, “AWAY”, “UNKNOWN”)

2.4.2. Lietotāja saskarnes apraksts

Tā kā PPS netika izvirzītas konkrētas prasības attiecībā uz konkrētām lietotāja saskarnes vizuālā dizaina īpašībām, moduļi, kuriem nepieciešama lietotāja saskarne, drīkst tikt realizēti ar konfigurējamiem atvērtā pirmkoda risinājumiem. Tālāk tiks aprakstītas projektējumā izvēlēto risinājumu lietotāju saskarnes.

2.4.2.1. REST servisu dokumentācijas tīmekļa saskarne

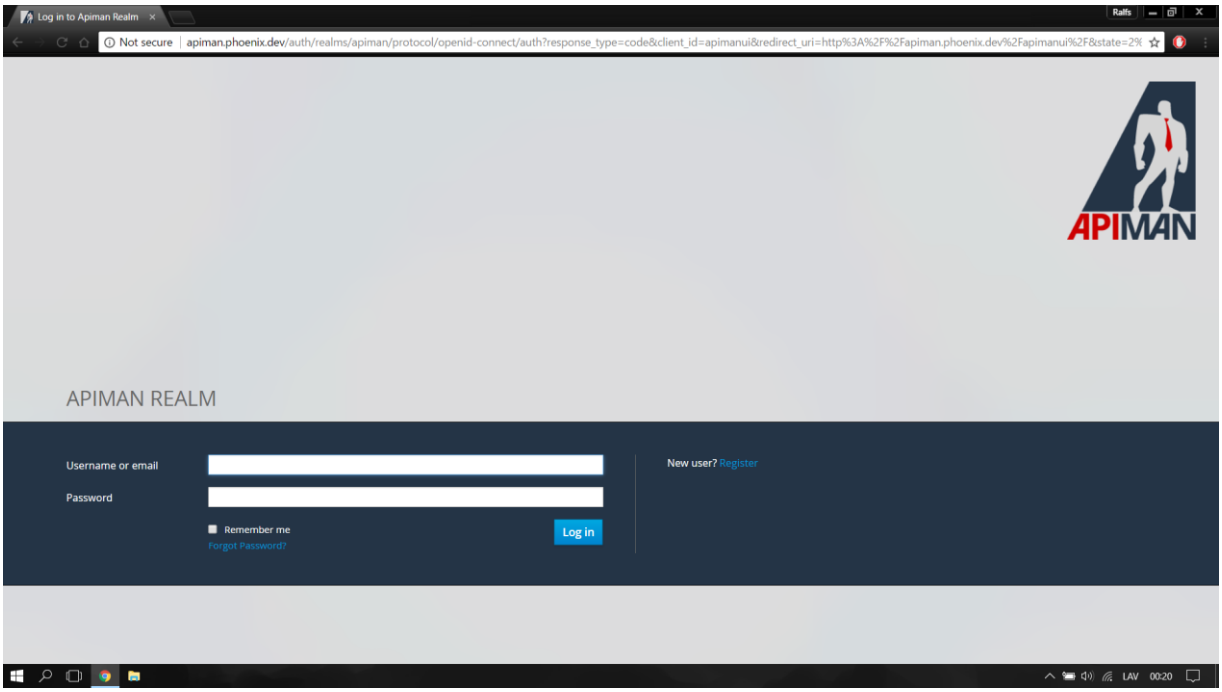
REST servisu dokumentācijas ģenerēšanas moduļa prasību realizācijai izvēlēts rīks “Swagger UI”. Attēlā 2.13. redzamā saskarne tiek izveidota no moduļa ģenerētās vārtejas API moduļa REST servisu dokumentācijas JSON formātā. Tajā visi REST servisi ir sagrupēti un ir iespējams izmēģināt katra servisa darbību.



2.13.att. **REST servisu dokumentācijas ģenerēšanas moduļa saskarne**

2.4.2.2. API pārvaldības moduļa tīmekļa saskarne

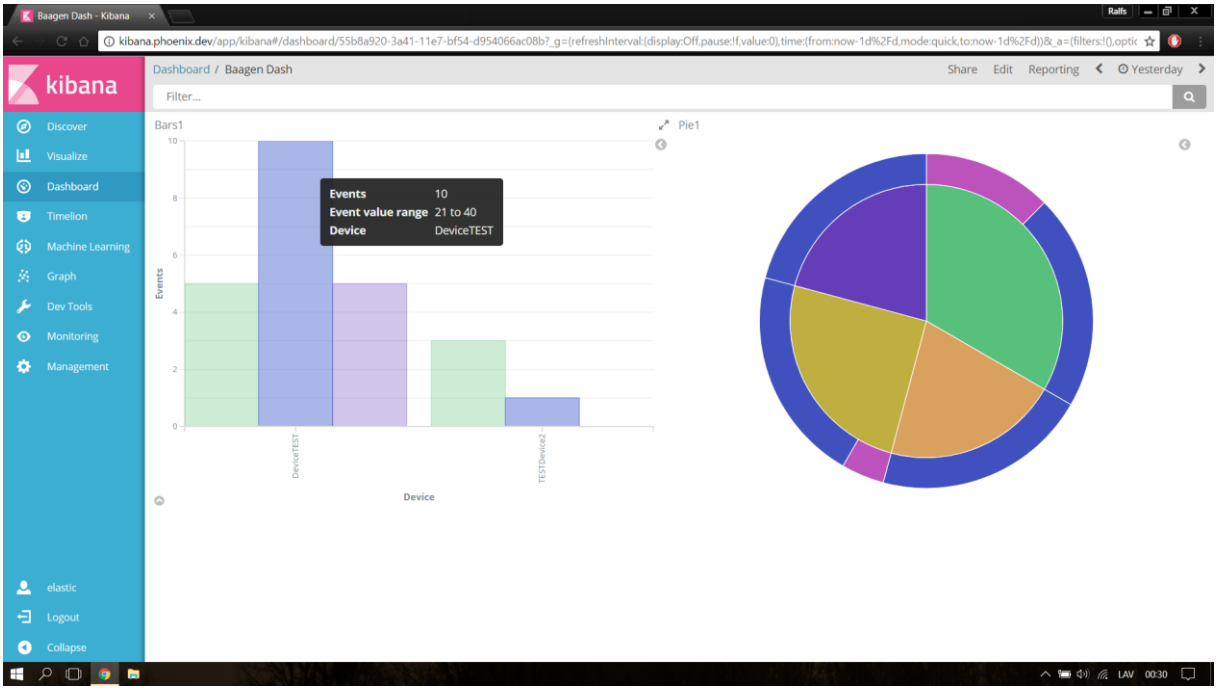
API pārvaldības moduļa prasību realizācijai izvēlēts rīks "API Man". Attēlā 2.14. redzama rīka autentifikācijas saskarne. Tajā ir divi ievadlauki (lietotājvārdam un parolei), kā arī saites uz lietotāja reģistrācijas un aizmirstas paroles nomaiņas saskarnēm. Rīkā ir arī iespēja šīs saskarnes vizuālo tēlu pielāgot konkrētai sistēmai, tomēr tas neietilpa šī projekta ietvaros.



2.14.att. **API pārvaldības moduļa autentifikācijas saskarne**

2.4.2.3. Datu vizualizācijas moduļa tīmekļa saskarne

Datu vizualizācijas moduļa prasību realizācijai izvēlēts rīks “Kibana”, kas ir paredzēts lietošanai kopā ar “Elasticsearch” datubāzi/pilnteksta meklētājprogrammu. 2.15. attēlā ir redzama informācijas paneļa saskarne. Attēlā redzamajā informācijas panelī ir izvietotas divas vizualizācijas. Izvēloties opciju “Edit” augšējā labajā pusē, ir iespējams panelim pievienot vai no tā noņemt vizualizācijas, kā arī pārkārtot panelī jau esošās. Lai veidotu informācijas paneli, vispirms ir jāizveido un jāsaglabā datu vizualizācijas. Tas ir izdarāms, izvēloties opciju “Visualize” kreisajā izvēlņu joslā.



2.15.att. **Datu vizualizācijas moduļa informācijas paneļa saskarne**

3. Testēšanas dokumentācija

3.1. Plāns

Platformas testēšanai jānotiek, izmantojot "melnās kastes" principu. Testēšanas procesā obligāti ir jāveic vismaz šādas darbības:

- Datu validācijas moduļa testēšana ar automatizētiem vienībtestiem
- Klienta bibliotēkas moduļa testēšana ar automatizētiem integrācijas testiem
- Vārtejas API moduļa testēšana ar manuāliem testiem

Testu izstrādei jānotiek paralēli vai uzreiz pēc attiecīgā moduļa izstrādes. Automatizētajiem testiem ir jāizpildās kā daļai no projekta automatizētās būvēšanas procesa.

3.2. Žurnāls

3.2.1. Datu validācijas moduļa testēšana

Datu validācijas modulis tika testēts, izmantojot JUnit vienībtestēšanas satvaru. Šī moduļa testi tiek veikti projekta būvēšanas procesā.

3.1.tabula **Funkcijas "Datu pārsūtīšanas objektu validācija" (1.3.8.1) vienībtesti**

Nr.	Testa apraksts	Sagaidāmais rezultāts	Testa rezultāts
1.	Testā tiek pārbaudīta datubāzes identifikatoru validācija, funkcijai padodot vienu korektu un vienu nekorektu vērtību.	Korektā vērtība tiks atpazīta kā korekta, bet nekorektajai vērtībai tiks konstatēta neatbilstība ierobežojumiem.	Atbilst sagaidāmajam rezultātam.
2.	Testā tiek pārbaudīta tālruņa numuru validācija, padodot funkcijai vairākas korektas un nekorektas vērtības. Starp izmantotajām vērtībām ir tālruņa numuri ar un bez atstarpēm, ar un bez valstu kodiem.	Visi korektie tālruņa numuri, kuri atbilst dažādiem formātiem, tiks atpazīti kā korekti, bet dažādiem nekorektiem tālruņa numuriem tiks konstatēta neatbilstība ierobežojumiem.	Funkcija par korektu atzina tālruņa numuru ar nekorektu valsts kodu. Tika izlabota funkcijā veiktā formāta pārbaude.
3.	Testā tiek pārbaudīta ielu numuru validācija, padodot funkcijai vienu korektu un vienu nekorektu vērtību.	Korektā vērtība tiks atpazīta kā korekta, bet nekorektajai vērtībai tiks konstatēta neatbilstība ierobežojumiem.	Atbilst sagaidāmajam rezultātam.
4.	Testā tiek pārbaudīta valstu nosaukumu validācija, padodot funkcijai vienu korektu un vairākas dažādu iemeslu dēļ nekorektas vērtības.	Korektais valsts nosaukums tiks atpazīts kā korekts, bet visās nekorektajās vērtībās tiks konstatēta neatbilstība ierobežojumiem.	Atbilst sagaidāmajam rezultātam.

5.	Testā tiek pārbaudīta notikumu aprakstu validācija, padodot funkcijai vienu korektu aprakstu, un divus garuma nosacījumiem neatbilstošus aprakstus.	Korektais apraksts tiks atpazīts kā korekts, bet ierobežojumiem neatbilstošie apraksti tiks atpazīti kā nekorekti.	Atbilst sagaidāmajam rezultātam.
6.	Testā tiek pārbaudīta ierīču statusu validācija, padodot funkcijai visas 3 korektās vērtības un vienu nekorektu vērtību.	Visas korektās vērtības tiks atpazītas kā korektas un tiks konstatēta nekorektās vērtības neatbilstība ierobežojumiem.	Atbilst sagaidāmajam rezultātam.
7.	Testā tiek pārbaudīta ģeogrāfiskā platuma validācija, padodot funkcijai korektas vērtības (gan pozitīvas, gan negatīvas) un nekorektas vērtības.	Gan pozitīvās, gan negatīvās korektās vērtības tiks atpazītas kā korektas, bet nekorektajām vērtībām tiks konstatēta neatbilstība ierobežojumiem.	Atbilst sagaidāmajam rezultātam.
8.	Testā tiek pārbaudīta ģeogrāfiskā garuma validācija, padodot funkcijai korektas vērtības (gan pozitīvas, gan negatīvas) un nekorektas vērtības.	Gan pozitīvās, gan negatīvās korektās vērtības tiks atpazītas kā korektas, bet nekorektajām vērtībām tiks konstatēta neatbilstība ierobežojumiem.	Atbilst sagaidāmajam rezultātam.
9.	Testā tiek pārbaudīta atrašanās vietu un notikumu režīmu nosaukumu validācija, padodot funkcijai visas trīs korektās vērtības, kā arī nekorektas vērtības.	Visas korektās vērtības tiks atpazītas kā korektas un tiks konstatēta nekorekto vērtību neatbilstība ierobežojumiem.	Atbilst sagaidāmajam rezultātam.
10.	Testā tiek pārbaudīta atrašanās vietu un notikumu režīmu identifikatoru validācija, padodot funkcijai visas trīs korektās vērtības, kā arī nekorektas vērtības.	Visas korektās vērtības tiks atpazītas kā korektas un tiks konstatēta nekorekto vērtību neatbilstība ierobežojumiem.	Funkcija kā korektu atzina vērtību, kas bija veidota, savienojot visas trīs pieļaujamās vērtības. Tika izlabota funkcijā veiktā vērtības pārbaude.
11.	Testā tika pārbaudīta nosaukumu validācija. Funkcijai tika padotas korektas vērtības, tajā skaitā vērtības, kas satur burtus ar diakritiskajām zīmēm, kā arī garuma vai iekļauto simbolu dēļ nekorektas vērtības.	Visas korektās vērtības tiks atpazītas kā korektas, bet nekorektajām vērtībām tiks konstatēta neatbilstība ierobežojumiem.	Atbilst sagaidāmajam rezultātam.

12.	Testā tika pārbaudīta ierīču novietojuma klasifikatoru validācija. Funkcija tika pārbaudīta gan ar korektām vērtībām, gan nekorektām (sarakstiem, kas satur negatīvus skaitļus vai satur vairāk nekā 3 elementus).	Visi korektie novietojuma klasifikatoru saraksti tiks atpazīti kā korekti, bet nekorektajiem sarakstiem tiks konstatēta neatbilstība ierobežojumiem.	Atbilst sagaidāmajam rezultātam.
13.	Testā tika pārbaudīta temperatūras skalu validācija, padodot funkcijai gan abas korektās vērtības, gan nekorektas vērtības.	Abas korektās vērtības tiks atpazītas kā korektas, bet nekorektajām vērtībām tiks konstatēta neatbilstība ierobežojumiem.	Atbilst sagaidāmajam rezultātam.
14.	Testā tika pārbaudīta laika zonu validācija, izmantojot gan dažādus korektus formātus, gan dažādu iemeslu dēļ nekorektas vērtības.	Vērtības, kas atbilst korektajiem formātiem, tiks atpazītas kā korektas, bet nekorektām vērtībām tiks konstatēta neatbilstība ierobežojumiem.	Atbilst sagaidāmajam rezultātam.

3.2.2. Klienta bibliotēkas moduļa testēšana

Klienta bibliotēkas moduļa testos tika pārbaudīta moduļa iekšējā darbība, gan arī moduļa sadarbība ar vārtejas API un starpprogrammatūras moduļiem, kā arī platformas datubāzēm. Testi tika veikti, izmantojot integrācijas satvarā Apache Camel iekļautās testu bibliotēkas. Šī moduļa testi tiek veikti projekta būvēšanas procesā.

3.2.tabula **Klienta bibliotēkas moduļa integrācijas testi**

Nr.	**Testa apraksts**	**Sagaidāmais rezultāts**	**Testa rezultāts**
1.	Testā tika pārbaudīta vārtejas API testa servisa izsaukšana no klienta bibliotēkas moduļa.	Tika sagaidīts, ka izsaukuma atbildē būs teksts “pong”.	Atbilst sagaidāmajam rezultātam.
2.	Tika pārbaudīta klienta bibliotēkas moduļa reakcija, kad izsaukumā ir nekorekts izsaucamās funkcijas nosaukums.	Tika sagaidīts, ka uz vārtejas API netiks izdarīts izsaukums un tā vietā tiks atgriezts kļūdas ziņojums.	Atbilst sagaidāmajam rezultātam.
3.	Testā tika pārbaudīta ierīču novietojuma klasifikatoru iegūšanas funkcijas izsaukšana no klienta bibliotēkas moduļa.	Tika sagaidīts, ka funkcijas izsaukuma atbildē būs visi datubāzē saglabātie ierīču novietojuma klasifikatori.	Atbilst sagaidāmajam rezultātam.
4.	Testā tika pārbaudītas	Tika sagaidīts, ka, izveidojot	Atbilst

	organizācijas saglabāšanas un organizācijas atrašanas pēc ID funkcijas. Vispirms tika izveidots jauns organizācijas ieraksts datubāzē un pēc atgrieztā ID tika izdarīta meklēšana.	organizācijas ierakstu, tiks atgriezts ieraksta ID, kā arī, ja pēc šī ID izdara meklēšanu, tad tiks atgriezts saglabātais organizācijas ieraksts.	sagaidāmajam rezultātam.
5.	Testā tika pārbaudīta visu organizāciju ierakstu iegūšana no datubāzes, izmantojot klienta bibliotēkas moduļa funkcijas.	Tika sagaidīts, ka izsaukuma atbildē būs visi datubāzē glabātie organizāciju ieraksti.	Atbilst sagaidāmajam rezultātam.
6.	Testā tika pārbaudīta visu organizāciju ID iegūšana no datubāzes, izmantojot klienta bibliotēkas moduļa funkcijas.	Tika sagaidīts, ka izsaukuma atbildē būs visu datubāzē glabāto organizāciju ierakstu identifikatori.	Atbilst sagaidāmajam rezultātam.
7.	Testā tika pārbaudītas atrašanās vietas saglabāšanas un atrašanas pēc piesaistītās organizācijas ID funkcijas. Vispirms tika izveidots jauns atrašanās vietas ieraksts datubāzē un pēc 4. testā izveidotā ieraksta ID tika izdarīta meklēšana.	Tika sagaidīts, ka, izveidojot atrašanās vietas ierakstu, tiks atgriezts ieraksta ID, kā arī, ja pēc 4. testā izveidotā organizācijas ID izdara piesaistīto atrašanās vietu meklēšanu, tad tiks atgriezts saglabātais atrašanās vietas ieraksts.	Atbilst sagaidāmajam rezultātam.
8.	Testā tika pārbaudīta visu atrašanās vietu ID iegūšana no datubāzes, izmantojot klienta bibliotēkas moduļa funkcijas.	Tika sagaidīts, ka izsaukuma atbildē būs visu datubāzē glabāto atrašanās vietu ierakstu identifikatori.	Atbilst sagaidāmajam rezultātam.
9.	Testā tika pārbaudītas ierīces saglabāšanas un atrašanas pēc piesaistītās atrašanās vietas ID funkcijas. Vispirms tika izveidots jauns ierīces ieraksts datubāzē un pēc 7. testā izveidotā ieraksta ID tika izdarīta meklēšana.	Tika sagaidīts, ka, izveidojot ierīces ierakstu, tiks atgriezts ieraksta ID, kā arī, ja pēc 7. testā izveidotā atrašanās vietas ieraksta ID izdara piesaistīto ierīču meklēšanu, tad tiks atgriezts saglabātais ierīces ieraksts.	Atbilst sagaidāmajam rezultātam.
10.	Testā tika pārbaudīta visu ierīču ID iegūšana no datubāzes, izmantojot klienta bibliotēkas moduļa funkcijas.	Tika sagaidīts, ka izsaukuma atbildē būs visu datubāzē glabāto ierīču ierakstu identifikatori.	Atbilst sagaidāmajam rezultātam.

11.	Testā tika pārbaudītas notikumu saglabāšanas un atrašanas pēc piesaistītās ierīces ID funkcijas. Vispirms tika izveidots jauns notikuma ieraksts datubāzē un pēc 9. testā izveidotā ieraksta ID tika izdarīta meklēšana.	Tika sagaidīts, ka, izveidojot notikuma ierakstu, tiks atgriezts ieraksta ID, kā arī, ja pēc 9. testā izveidotā ierīces ieraksta ID izdara piesaistīto notikumu meklēšanu, tad tiks atgriezts saglabātais notikuma ieraksts. Papildus tika sagaidīts, ka tiks izveidots jēldatu dokumenta ieraksts platformas Elasticsearch datubāzē.	Atbilst sagaidāmajam rezultātam.
12.	Testā tika pārbaudīta visu notikumu ID iegūšana no datubāzes, izmantojot klienta bibliotēkas moduļa funkcijas.	Tika sagaidīts, ka izsaukuma atbildē būs visu datubāzē glabāto notikumu ierakstu identifikatori.	Atbilst sagaidāmajam rezultātam.

3.2.3. Vārtejas API moduļa testēšana

Vārtejas API moduļa testos tika pārbaudīta modulī realizēto REST servisu darbība, atbilstība dokumentācijai, kā arī reakcija uz nekorektiem ievaddatiem. Šī moduļa testi tika veikti manuāli, izmantojot REST servisu dokumentācijas ģenerēšanas moduļa izveidoto tīmekļa saskarni.

3.3.tabula **Vārtejas API moduļa manuālie testi**

Nr.	**Testa apraksts**	**Sagaidāmais rezultāts**	**Testa rezultāts**
1.	Testā tika pārbaudīts vārtejas API moduļa testa serviss “ping”.	Tiek sagaidīts, ka izsaukuma atbildē būs teksts “pong”.	Atbilst sagaidāmajam rezultātam.
2.	Testā tika pārbaudīts vārtejas API moduļa REST serviss visu saglabāto organizāciju ID iegūšanai.	Tiek sagaidīts, ka izsaukuma atbildē būs saraksts ar visiem datubāzē glabāto organizāciju identifikatoriem.	Atbilst sagaidāmajam rezultātam.
3.	Testā tika pārbaudīts vārtejas API moduļa REST serviss organizācijas atrašanai pēc tās ID, izmēģinot gan korektas, gan nekorektas ID vērtības.	Tiek sagaidīts, ka, saņemot korektu ID, funkcija atgriezīs atrasto organizāciju, bet, saņemot nekorektu ID, izvadīs kļūdas ziņojumu.	Atbilst sagaidāmajam rezultātam.
4.	Testā tika pārbaudīts vārtejas API moduļa REST serviss visu saglabāto organizāciju ierakstu iegūšanai.	Tiek sagaidīts, ka izsaukuma atbildē būs saraksts ar visām datubāzē glabātajām organizācijām.	Atbilst sagaidāmajam rezultātam.
5.	Testā tika pārbaudīts vārtejas API moduļa REST	Tiek sagaidīts, ka pie korektu ievaddatu ievadīšanas tiks	Atbilst sagaidāmajam

	serviss organizācijas saglabāšanai datubāzē, izmēģinot gan korektas, gan nekorektas organizācijas lauku vērtības.	izveidots organizācijas ieraksts datubāzē un atbildē tiks saņemts izveidotā ieraksta ID. Nekorektu ievaddatu gadījumā jāsaņem kļūdas ziņojums, kurā minēts, kurš lauks neatbilst nosacījumiem.	rezultātam.
6.	Testā tika pārbaudīts vārtejas API moduļa REST serviss atrašanās vietas meklēšanai pēc organizācijas ID, izmēģinot gan korektas, gan nekorektas ID vērtības.	Tiek sagaidīts, ka, saņemot korektu ID, funkcija atgriezīs organizācijai piesaistītās atrašanās vietas, bet, saņemot nekorektu ID, izvadīs kļūdas ziņojumu.	Atbilst sagaidāmajam rezultātam.
7.	Testā tika pārbaudīts vārtejas API moduļa REST serviss atrašanās vietas saglabāšanai datubāzē, izmēģinot gan korektas, gan nekorektas atrašanās vietas lauku vērtības.	Tiek sagaidīts, ka pie korektu ievaddatu ievadīšanas tiks izveidots atrašanās vietas ieraksts datubāzē un atbildē tiks saņemts izveidotā ieraksta ID. Nekorektu ievaddatu gadījumā jāsaņem kļūdas ziņojums, kurā minēts, kurš lauks neatbilst nosacījumiem.	Atbilst sagaidāmajam rezultātam.
8.	Testā tika pārbaudīts vārtejas API moduļa REST serviss visu saglabāto atrašanās vietu ID iegūšanai.	Tiek sagaidīts, ka izsaukuma atbildē būs saraksts ar visu datubāzē glabāto atrašanās vietu identifikatoriem.	Atbilst sagaidāmajam rezultātam.
9.	Testā tika pārbaudīts vārtejas API moduļa REST serviss ierīces meklēšanai pēc atrašanās vietas ID, izmēģinot gan korektas, gan nekorektas ID vērtības.	Tiek sagaidīts, ka, saņemot korektu ID, funkcija atgriezīs atrašanās vietai piesaistītās ierīces, bet, saņemot nekorektu ID, izvadīs kļūdas ziņojumu.	Atbilst sagaidāmajam rezultātam.
10.	Testā tika pārbaudīts vārtejas API moduļa REST serviss ierīces saglabāšanai datubāzē, izmēģinot gan korektas, gan nekorektas ierīces lauku vērtības.	Tiek sagaidīts, ka pie korektu ievaddatu ievadīšanas tiks izveidots ierīces ieraksts datubāzē un atbildē tiks saņemts izveidotā ieraksta ID. Nekorektu ievaddatu gadījumā jāsaņem kļūdas ziņojums, kurā minēts, kurš lauks neatbilst nosacījumiem.	Atbilst sagaidāmajam rezultātam.
11.	Testā tika pārbaudīts vārtejas API moduļa REST serviss visu saglabāto ierīču ID iegūšanai.	Tiek sagaidīts, ka izsaukuma atbildē būs saraksts ar visu datubāzē glabāto ierīču identifikatoriem.	Atbilst sagaidāmajam rezultātam.
12.	Testā tika pārbaudīts vārtejas API moduļa REST	Tiek sagaidīts, ka, saņemot korektu ID, funkcija atgriezīs ierīcei	Atbilst sagaidāmajam

	serviss notikuma meklēšanai pēc ierīces ID, izmēģinot gan korektas, gan nekorektas ID vērtības.	piesaistītos notikumus, bet, saņemot nekorektu ID, izvadīs kļūdas ziņojumu.	rezultātam.
13.	Testā tika pārbaudīts vārtejas API moduļa REST serviss notikuma saglabāšanai datubāzē, izmēģinot gan korektas, gan nekorektas notikuma lauku vērtības.	Tiek sagaidīts, ka pie korektu ievaddatu ievadīšanas tiks izveidots notikuma ieraksts datubāzē un atbildē tiks saņemts izveidotā ieraksta ID. Nekorektu ievaddatu gadījumā jāsaņem kļūdas ziņojums, kurā minēts, kurš lauks neatbilst nosacījumiem. Funkcijai būtu arī jāizveido ieraksts jēldatu datubāzē.	Pēc notikuma saglabāšanas dokumentu datubāzē, netika izveidots ieraksts jēldatu datubāzē. Kļūda tika izlabota attiecīgajā funkcijā.
14.	Testā tika pārbaudīts vārtejas API moduļa REST serviss visu saglabāto notikumu ID iegūšanai.	Tiek sagaidīts, ka izsaukuma atbildē būs saraksts ar visu datubāzē glabāto notikumu identifikatoriem.	Atbilst sagaidāmajam rezultātam.
15.	Testā tika pārbaudīts vārtejas API moduļa REST serviss visu saglabāto ierīču novietojuma klasifikatoru iegūšanai.	Tiek sagaidīts, ka izsaukuma atbildē būs saraksts ar visiem datubāzē glabātajiem ierīču novietojuma klasifikatoriem.	Atbilst sagaidāmajam rezultātam.

3.3. Rezultātu kopsavilkums

Visi testēšanas laikā atrastie defekti tika novērsti, veicot nepieciešamos labojumus konkrētajās funkcijās. Lielākoties defektu cēlonis bija neuzmanības kļūdas. Laika trūkuma dēļ tika veikta tikai testēšanas plānā norādīto moduļu testēšana. Šie konkrētie moduļi tika izvēlēti, jo tie ir kritiski sistēmas darbībai. Tā kā platforma paredzēta kā prototips un prasībās netika iekļautas konkrētas veiktspējas prasības, veiktspējas testēšana netika veikta.

4. Projekta organizācija

Platformas izstrāde tika plānota atbilstoši ūdenskrituma modelim. Šī pieeja tika izvēlēta autora nelielās pieredzes dēļ. Reāli izstrādes gaitā radās nelielas novirzes no šī modeļa, jo atsevišķas prasības tika mainītas pēc izstrādes sākuma.

Pirms projekta izstrādes plānošanas, vispirms tika apgūtas uzņēmumā plašāk izmantotās tehnoloģijas, papildinot funkcionalitāti kādam iesāktam projektam. Pēc tam tika definētas platformas īstenojamās prasības. Tad tika veikta izstrādājamās sistēmas uzbūves plānošana. Sistēmas izstrāde tika sākta ar vārtejas API moduli un MongoDB dokumentu datubāzes konfigurāciju. Veidojot vārtejas API moduli, tam tika pievienota Dozer bibliotēka, ar kuras palīdzību tika īstenotas kartēšanas moduļa prasības. Tad platformai tika pievienots un konfigurēts REST servisu dokumentācijas ģenerēšanas modulis, kura realizācijai tika izvēlēts rīks Swagger UI. Paralēli šo moduļu izstrādei tika uzsākta arī dokumentācijas izveide. Tālāk tika sākta starpprogrammatūras moduļa izstrāde. Mēģinot īstenot definētās prasības, tika izlemts, ka nepieciešams veikt labojumus vārtejas API un starpprogrammatūras moduļos īstenotajā datu apstrādē. Tad tika uzsākta klienta bibliotēkas moduļa izstrāde, kā arī modulim paredzēto testu izstrāde. Sākotnēji datu validācija tika īstenota kā daļa no klienta bibliotēkas moduļa, tomēr vēlāk tā tika īstenota atsevišķā modulī, kuru izmanto gan klienta bibliotēkas modulis, gan vārtejas API modulis. Šim modulim tika izstrādāti arī vienībtesti. Pēc klienta bibliotēkas moduļa izstrādes tika izveidota arī piemēra integrācija ar platformu, lai pārbaudītu klienta bibliotēkas darbību un veiktu nepieciešamos uzlabojumus. Šī piemēra integrācija gan nav uztverama kā daļa no platformas. Tālāk tika veikta API pārvaldības moduļa prasību īstenošanai izvēlētā rīka API Man apguve un konfigurācija. Visbeidzot platformai tika pievienota Elasticsearch datubāze jēldatu dokumentu glabāšanai, kā arī Kibana spraudnis Elasticsearch datubāzei, ar kuru tika īstenotas datu vizualizācijas moduļa prasības.

Projekta pamatprasību formulēšana notika, darba autoram sadarbojoties ar projekta vadītāju. Konsultācijas ar projekta vadītāju tika izmantotas arī platformas izstrādes gaitā, lai ietaupītu laiku īpaši sarežģītu problēmu risināšanā. Produkta izstrādi, testēšanu, kā arī dokumentācijas veidošanu veica darba autors.

Projekta izstrādē tika izmantota Eclipse izstrādes vide. Pirmkoda un platformas vides konfigurācijas tika pārvaldītas, izmantojot rīkus Git un Docker Compose.

5. Kvalitātes nodrošināšana

Projekta realizācijas gaitā tika ievēroti vairāki pasākumi, lai nodrošinātu gala produkta kvalitāti:

1. Platformas izstrādes gaitā jauna funkcionalitāte tika izstrādāta atsevišķos repozitorija zaros, kurus sapludināja ar repozitorija pamatzaru tikai pēc funkcionalitātes pabeigšanas
2. Izmantotajā konfigurāciju pārvaldības rīkā GitLab izmaiņu iekļaušana repozitorija pamatzarā notika, izmantojot sapludināšanas pieprasījumus (merge requests). Lai pieprasījumu varētu īstenot, vispirms projekta vadītajam bija jāveic koda apskate un jādod apstiprinājums, ka izmaiņas drīkst iekļaut pamata repozitorijā
3. Sistēma tika būvēta, lai to varētu darbināt, izmantojot Docker Compose rīku, kas nodrošina konstantu programmas izpildes vidi, neatkarīgi no pamata iekārtas vides
4. Programmkoda izstrādē tika ievērots vienots stils
5. Programmkods tika organizēts modulāri, cenšoties panākt programmkoda moduļu atbilstību PPS minētajiem moduļiem
6. Projekta dokumentācija tika izstrādāta, balstoties uz valsts standartiem

6. Konfigurācijas pārvaldība

Projekta programmkoda konfigurāciju pārvaldībai tikai izmantota programma Git, bet programmkoda repozitorijs tika glabāts GitLab sistēmā. Saglabājot repozitorijā kārtējās izmaiņas, tām tika pievienoti komentāri par tajās paveikto. Gadījumos, kad jaunas funkcionalitātes izstrāde tika uzsākta vēl pirms iepriekšējās funkcionalitātes zara sapludināšanas ar repozitorija pamatzaru, radušies sapludināšanas konflikti tika risināti lokāli, sapludinot repozitorija pamatzaru ar lokālo iepriekšējās funkcionalitātes zaru. Pēc konfliktu manuālas atrisināšanas, izmaiņas tika saglabātas repozitorijā, tika veikta sapludināšana ar pamatzaru, bet pēc tam tika veikta repozitorija pamatzara sapludināšana ar lokālo jaunās funkcionalitātes zaru.

Sistēmas vides konfigurācija tika pārvaldīta, izmantojot Docker Compose rīku. Šī iemesla dēļ platformā ir modulis atsevišķu sistēmas daļu konteinerizācijas failiem (Dockerfile), kuri satur instrukcijas katras sistēmas daļas vides izveidošanai. Docker Compose rīka konfigurācija tiek glabāta vienā failā (docker-compose.yml), kurā tiek definēts, kā sistēmas daļas jeb konteineri savā starpā sadarbosies. Gan šis fails, gan konteineru instrukciju faili tiek uzturēti repozitorijā kopā ar platformas pirmkodu.

7. Darbietilpības novērtējums

Projekta plānošana tika veikta projekta sākumā, izveidojot plānu ar uzdevumiem, kas aptuveni atbilda platformas moduļiem. Plāns tika izstrādāts, konsultējoties ar pieredzējušākiem izstrādātājiem uzņēmumā un projektu vadītajiem. Plānā paredzētais projekta izstrādes laiks bija 3 mēneši un 10 dienas. Līdzīgs novērtējums tika iegūts, veicot salīdzināšanu ar sarežģītības ziņā līdzīgu projektu, kas uzņēmumā tika realizēts jau agrāk. Faktiskā darba izpilde aizņēma par 15 dienām vairāk.

Ņemot vērā izstrādātāja nelielo pieredzi, kā arī faktu, ka projektā izmantotas vairākas gan izstrādātājam, gan uzņēmumam jaunas tehnoloģijas, testēšanas plāna izveidē tika identificētas platformas kritiskākās daļas, kuras būtu obligāti jānotestē. Plānā tika paredzēta arī iespēja atsevišķu moduļu realizācijai izmantot konfigurējamus atvērtā pirmkoda risinājumus, kas arī tika izmantota. Izstrādes laikā tika mēģināts iekļauties sākotnējā plānā, tomēr, par spīti visiem plānā iekļautajiem laika taupīšanas variantiem, radās zināmas novirzes no plāna. Galvenās platformas daļas, kuru izstrāde vai konfigurācija aizņēma vairāk laika kā plānots:

1. API pārvaldības modulis - izvēlētais konfigurējamais atvērtā pirmkoda risinājums API Man sevī iekļāva netriviālu arhitektūru, kas prasīja laiku, lai to izprastu
2. Vārtejas API moduļa REST servisi - tā kā šī sistēmas daļa tika izstrādāta pirmā, tās izstrāde aizņēma ilgāku laiku, jo autoram vēl nebija pietiekamas pieredzes darbā ar Apache Camel un Spring satvariem
3. Klienta bibliotēkas moduļa integrācijas komponente - šīs sistēmas daļas izstrāde prasīja padziļinātu Apache Camel satvara jēdzienu un principu izpratni
4. Reversā starpniekservera modulis - radās neparedzētas problēmas mēģinot panākt pareizu moduļa sadarbību ar API pārvaldības moduli

Secinājumi

Izstrādājot darbu, tika izveidots specifikācijai atbilstošs platformas prototips, kura funkcionalitāti papildinot un uzlabojot, varētu iegūt perspektīvu produktu. Izstrādājot šo prototipu, tika secināts, ka ir iespējams apvienot lietu internetu, REST tīmekļa servisus un NoSQL datubāzes, lai iegūtu apdrošināšanas jomā pielietojamu produktu.

Projekta laikā tika gūta pieredze un zināšanas darbā ar vairākām jaunām tehnoloģijām. Padziļināti tika apgūts Apache Camel integrāciju satvars, kā arī Spring vadības inversijas satvars. Labā līmenī tika apgūti Docker un Docker Compose konteinerizācijas rīki, kā arī dažu platformas moduļu realizācijai izvēlētie konfigurējamie atvērtā pirmkoda rīki: API Man, Swagger UI un Kibana. Tika gūta arī praktiska pieredze Nginx servera konfigurācijā. Tā kā lielākajā daļā platformas moduļu tika izmantota viena vai vairākas jaunas tehnoloģijas, tad visas izstrādes gaitā nozīmīgu laika daudzumu aizņēma šo tehnoloģiju dokumentācijas lasīšana. Platformas izstrādes procesā tika gūta arī atziņa, ka integrācijas ir starp sarežģītākajiem programmu veidiem no izstrādātāja puses.

Izmantotā literatūra

1. LVS 68:1996 “Programmatūras prasību specifikācijas ceļvedis”
2. LVS 72:1996 “Ieteicamā prakse programmatūras projektējuma aprakstīšanai”
3. MongoDB Documentation [tiešsaiste] https://docs.mongodb.com/
4. Apiman User Guide [tiešsaiste] https://apiman.gitbooks.io/apiman-user-guide/
5. Elasticsearch reference [tiešsaiste] https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html
6. Kibana User Guide [tiešsaiste] https://www.elastic.co/guide/en/kibana/current/index.html

Programmkoda fragmenti

1. Vārtejas API moduļa klase “RouteBuilder”

```
/*
   RouteBuilder class defines and configures
   routes for Gateway API REST services
 */

@Component
public class RouteBuilder extends SpringRouteBuilder {

   /*
      Field contains application properties
      defined in application.properties file
    */

   @Autowired
   private ApplicationProperties properties;

   // Function configures the REST API routes

   @Override
   public void configure() throws Exception {

   // Configuration for all REST services

      restConfiguration().enableCORS(true)
      .bindingMode(RestBindingMode.json)
      .jsonDataFormat("json-jackson")
      .dataFormatProperty("prettyPrint", "true")
      .apiContextPath("/swagger.json")
      .apiProperty("host", properties.getRestHost() + ":" +
properties.getRestPort())
      .apiProperty("api.title", "Gateway API")
      .apiProperty("cors", "true");

   /*
      REST endpoints for the "ping" API
    */

      rest("/ping")
      .description("Simple 'ping' API")
      .produces("text/plain")
         .get()
            .description("For checking if gateway-api is working")
            .outType(String.class)
            .route().setBody().simple("pong").endRest()
            .responseMessage().code(200)
            .message("Returns 'pong'")
            .endResponseMessage();

   /*
      REST endpoints for the "Organisation" API
    */

      rest("/organisation")
      .description("Organisation REST API")
      .produces("application/json")
         .get("/all")
            .description("Get all organisations")
            .outTypeList(RestOrganisation.class)
            .responseMessage().code(200)
```

```
            .message("All organisations from database")
            .endResponseMessage()
            .to("bean:documentService?method=getAllOrganisations")
         .get("/{id}")
            .description("Get organisation by id")
            .param().name("id")
            .type(RestParamType.path)
            .description("Id of organisation")
            .dataType("string")
            .endParam()
            .outType(RestOrganisation.class)
            .responseMessage().code(200)
            .message("Organisation with the given ID")
            .endResponseMessage()
            .to("bean:documentService?method=getOrganisationById")
         .post("/create")
            .description("Insert an organisation")
            .param().name("body")
            .type(RestParamType.body)
            .description("Body containing organisation info")
            .endParam()
            .type(RestOrganisation.class)
            .outType(RestIdWrapper.class)
            .responseMessage().code(200)
            .message("Returns the ID of created organisation object")
            .endResponseMessage()
            .to("bean:documentService?method=createOrganisation")
         .get("/ids")
            .description("Get the IDs of all organisations")
            .outTypeList(RestIdWrapper.class)
            .responseMessage().code(200)
            .message("All organisation IDs from database")
            .endResponseMessage()
            .to("bean:documentService?method=getOrganisationIds")
         .get("/apimanID")
            .description("Get user organisation record ID in database")
            .outType(RestIdWrapper.class)
            .responseMessage().code(200)
            .message("User organisation's ID in the database")
            .endResponseMessage()
            .to("direct:apiman");

   /*
      REST endpoints for the "Location" API
    */

      rest("/location")
      .description("Location REST API")
      .produces("application/json")
         .get("/all/{organisationId}")
            .description("Get all locations of an organisation")
            .param().name("organisationId")
            .type(RestParamType.path)
            .description("Id of organisation")
            .dataType("string")
            .endParam()
            .outTypeList(RestLocation.class)
            .responseMessage().code(200)
            .message("All locations from database with the given
organisation ID")
            .endResponseMessage()
            .to("bean:documentService?method=getLocations")
         .post("/create")
            .description("Add locations to an organisation")
```

```
                .param().name("body")
                .type(RestParamType.body)
                .description("Body containing location to add")
                .endParam()
                .type(RestLocation.class)
                .outType(RestIdWrapper.class)
                .responseMessage().code(200)
                .message("Returns the ID of created location object")
                .endResponseMessage()
                .to("bean:documentService?method=createLocation")
            .get("/ids")
                .description("Get the IDs of all locations")
                .outTypeList(RestIdWrapper.class)
                .responseMessage().code(200)
                .message("All location IDs from database")
                .endResponseMessage()
                .to("bean:documentService?method=getLocationIds");

        /*
           REST endpoints for the "Device" API
         */

           rest("/device")
           .description("Device REST API")
           .produces("application/json")
                 .get("/all/{locationId}")
                    .description("Get all devices of a location")
                    .param().name("locationId")
                    .type(RestParamType.path)
                    .description("Id of location")
                    .dataType("string")
                    .endParam()
                    .outTypeList(RestDevice.class)
                    .responseMessage().code(200)
                    .message("All devices from database with the given location
ID")
                    .endResponseMessage()
                    .to("bean:documentService?method=getDevices")

                 .post("/create")
                    .description("Add devices to a location")
                    .param().name("body")
                    .type(RestParamType.body)
                    .description("Body containing device to add")
                    .endParam()
                    .type(RestDevice.class)
                    .outType(RestIdWrapper.class)
                    .responseMessage().code(200)
                    .message("Returns the ID of created device object")
                    .endResponseMessage()
                    .to("bean:documentService?method=createDevice")
                 .get("/ids")
                    .description("Get the IDs of all devices")
                    .outTypeList(RestIdWrapper.class)
                    .responseMessage().code(200)
                    .message("All device IDs from database")
                    .endResponseMessage()
                    .to("bean:documentService?method=getDeviceIds");

        /*
           REST endpoints for the "Tag" API
         */

           rest("/tags")
```

```
        .description("Tag REST API")
        .produces("application/json")
            .get("/all")
                .description("Get a list of all tags and their IDs in the
database")
                .outTypeList(RestTag.class)
                .responseMessage().code(200)
                .message("All tags from database")
                .endResponseMessage()
                .to("bean:documentService?method=getTags");

    /*
        REST endpoints for the "Event" API
     */

        rest("/event")
        .description("Event REST API")
        .produces("application/json")
            .get("/all/{deviceId}")
                .description("Get all events of a device")
                .param().name("deviceId")
                .type(RestParamType.path)
                .description("Id of device")
                .dataType("string")
                .endParam()
                .outTypeList(RestEvent.class)
                .responseMessage().code(200)
                .message("All events from database with the given device ID")
                .endResponseMessage()
                .to("bean:documentService?method=getEvents")
            .post("/create")
                .description("Add events to a device")
                .param().name("body")
                .type(RestParamType.body)
                .description("Body containing event to add")
                .endParam()
                .type(RestEvent.class)
                .outType(RestIdWrapper.class)
                .responseMessage().code(200)
                .message("Returns the ID of created event object")
                .endResponseMessage()
                .to("bean:documentService?method=createEvent")
            .get("/ids")
                .description("Get the IDs of all events")
                .outTypeList(RestIdWrapper.class)
                .responseMessage().code(200)
                .message("All event IDs from database")
                .endResponseMessage()
                .to("bean:documentService?method=getEventIds");
    }
}
```

2. Vārtejas API moduļa darbības vides konfigurācijas fails (Dockerfile)

```
FROM java:openjdk-8-jdk

MAINTAINER ralfs.sedlenieks@galeoconsulting.com

ENV UID=30101
ENV GATEWAY_WORK=/opt/gateway-dir
ENV JAVA_OPTS=-Xmx128M

RUN useradd -r -u ${UID} -m -c "gateway api role account" -d
${GATEWAY_WORK} -s /bin/false gateway-api-usr

COPY ./phoenix-iot_gateway-api-*.jar ${GATEWAY_WORK}/app.jar
RUN chown gateway-api-usr -R ${GATEWAY_WORK}

USER gateway-api-usr

EXPOSE 8080

CMD java ${JAVA_OPTS} ${APP_ARGS} -jar ${GATEWAY_WORK}/app.jar
```

Kvalifikācijas darbs „***Paplašināma IoT platforma datu un notikumu apkopošanai***” 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: ***Ralfs Sedlenieks*** **_____ ___.05.2017**.

Rekomendēju darbu aizstāvēšanai

Darba vadītājs: ***M.dat., Toms Feldmanis*** **_____ ___.05.2017.**

Recenzents: ***M.dat., Jānis Baiža***

Darbs iesniegts Datorikas fakultātē 29.05.2017.

Dekāna pilnvarotā persona:

docente, Dr.dat. Darja Solodovņikova _____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē

___.06.2017. prot. Nr. _____

Komisijas sekretārs(-e): _____