LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

DAM ADAPTERIS

KVALIFIKĀCIJAS DARBS

Autors: **Oļģerts Grudulis**

Studenta apliecības Nr: og19006

Vadītājs: Mg. dat. Dace Jansone

RĪGA 2022

ANOTĀCIJA

Darba “DAM adapteris” ietvaros tika izstrādāts risinājums, kas nodrošina ārējo sistēmu sadarbību ar veco Rīgas domes lietvedības sistēmu un jauno dokumentu apstrādes moduļa sistēmu. Adapteris ļauj iegūt vajadzīgo informāciju, kas saistīta ar reģistrētiem dokumentiem, kā arī dod iespējas veidot jaunus dokumentus. Izstrādātā programmatūra palīdz veicināt pāreju no vienas lietvedības sistēmas uz citu vienkāršāku.

Atslēgvārdi: dokumenti, dokumentu vadības sistēma, DVS, Oracle PL/SQL

ABSTRACT

In the “DAM adapter” projects boundaries is the development of a solution, that ensures the interaction of external systems with the old Riga City Council document management system and the new document management system. The adapter allows you to obtain the necessary information related to registered documents, as well as enables you to create new documents. The developed software helps to facilitate the process of transition from one document management system to another.

Keywords: documents, document management system, DMS, Oracle PL/SQL

SATURA RĀDĪTĀJS

APZĪMĒJUMU SARAKSTS

DAM – Dokumentu apstrādes modulis

RD LIS – Rīgas domes lietvedības informācijas sistēma

RD VIS – Rīgas domes Vienotā informācijas sistēma

DVS – Dokumentu vadības sistēma (RD LIS, DAM)

BIS – Būvniecības informācijas sistēma

LIETO – RD VIS lietotāju tiesību un struktūrvienību lietojumprogramma

KAVIS – Klientu attiecību vadības lietojumprogramma

PKIP – Projekts "Pašvaldību klientu informācijas pārvaldības risinājums"

PPS – Programmatūras prasību specifikācija

PPA – Programmatūras projektējuma apraksts

DB – Datubāze

JSON – JavaScript objektu notācija, datu apmaiņas formāts

IEVADS

Nolūks

Dokumenta nolūks ir programmatūras prasības specifikācijas (PPS) un programmatūras projektējuma apraksta (PPA) dokumenta apstrādes moduļa (DAM) adaptera programmatūras aprakstīšanai.

Šīs dokuments ir paredzēts programmatūras pasūtītājam, lai varētu precīzi definēt nepieciešamās prasības pret sistēmu, un izstrādātājiem, lai precīzi definētu izstrādājamās sistēmas funkcijas.

Darbības sfēra

DAM adapteris nodrošinās dokumentu reģistrēšanu un atlasi Rīgas domes Vadības informācijas sistēmā (RD VIS), pēc DVS "Lietvaris" ieviešanas Rīgas domē.

Saistība ar citiem dokumentiem

Dokumenta sastādīšanā izmantots standarts LVS 68:1996 „Programmatūras prasību specifikācijas ceļvedis" [1]. Dokumenta tehniskais noformējums ir izstrādāts saskaņā ar Latvijas Universitātes rīkojumu Nr. 1/38 "Prasības noslēguma darbu (bakalaura, maģistra darbu, diplomdarbu un kvalifikācijas darbu) izstrādāšanai un aizstāvēšanai Latvijas Universitātē" [2]. Programmatūras prasības specifikācija nav paša studenta izstrādātais dokuments, tika izmantots uzņēmuma SIA "ZZ Dats" izstrādātais dokuments "PKIP integrācija ar RD VIS lietojumprogrammas KAVIS apakšsistēmām. PPS v.1.4" [3].

Dokumentu pārskats

Dokuments sastāv no vispārīgā apraksta, kas ietver informāciju par adapteri. Otrā dokumenta daļa iekļauj informāciju par projekta programmatūras prasību specifikāciju: dokumenta adaptera aprakstu, funkcionālās prasības un nefunkcionālās prasības. Šī dokumenta daļa apkopo informāciju programmatūras funkcionalitāti, ievaddatus un izvaddatus. Trešā dokumenta sadaļa ir programmatūras projektējuma apraksts, kur detalizēti tiek aprakstīta adaptera datubāze un daļējs funkciju projektējums. Ceturtajā nodaļa tiek aprakstīta testēšanas dokumentācija, kas ietver informāciju par izvēlētām testēšanas metodēm. Piektā nodaļa ir projekta organizācija, kur aprakstīts, kā tika pārvaldīts projekts, sniegta informācija par konfigurācijas pārvaldību, kvalitātes nodrošināšanu un darbietilpības novērtējumu.

1. VISPĀRĪGS APRAKSTS

1.1. Esošā stāvokļa apraksts

Šobrīd Rīgas dome izmanto lietvedības sistēmu RD LIS un sistēmas iekš RD VIS reģistrē dokumentus RD LIS. 2022.gadā Rīgas dome pāries uz DVS "Lietvaris" lietvedības sistēmu un visām sistēmām būs nepieciešams reģistrēt visus dokumentus DAM. Saistībā ar pāreju, rodas problēma, ka daudzām sistēmām jāpāriet no RD LIS uz DAM.

1.2. Pasūtītājs

DAM adaptera pasūtītājs ir Rīgas dome.

1.3. Produkta perspektīva

Izstrādājamais adapteris ir atkarīgs no DVS "Lietvaris" un RD LIS.

1.4. Darījumprasības

DAM adaptera galvenās funkcijas:

1. Visu dokumentu veidu reģistrāciju, pievienojot korespondentus/adresātus, datnes;
2. Dokumentu savstarpēju sasaisti, kas attēlojas saistīto dokumentu sarakstā (RD LIS sarakste);
3. Virtuālo lietu (administratīvās lietas, piedziņas lietas) reģistrāciju;
4. Rīkojumu un lēmumu grozījumu veidošanu;
5. Kontroles uzdevumu veidošanu.

1.5. Lietvedības dokumentu pieejamība

Lai garantētu gan vēsturisko dokumentu pieejamību, gan jauno dokumentu izveides iespējas, tiek izstrādāts RD LIS-DAM integrācijas dokumentu adapteris (turpmāk – Dokumentu adapteris). Tā primārā funkcija ir atbilstoši iestatījumiem veikt datu izgūšanas un datu saglabāšanas pieprasījumu trasēšanu starp RD LIS un DAM lietojumprogrammām. Šī dokumenta ietvaros tiek aprakstīta Dokumentu adaptera izveides prasības un integrācijas prasības šādām komponentēm:

1. RD LIS integrējamā komponente – Lietvedības dokumentu apstrādes DB pakotne *PK_SERVISI*;
2. DAM integrējamās komponentes:
   2.1. DB procedūras dokumentu reģistrācijai;
   2.2. DB skati dokumentu datu izgūšanai;

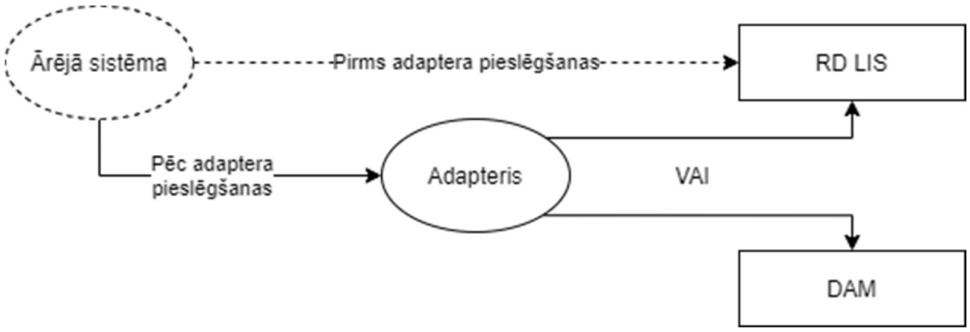
1.6. Vispārējie ierobežojumi

Adaptera procedūru izmantošanai nepieciešams uzstādīt DB sesiju.

2. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA

2.1. Dokumentu adaptera apraksts

Dokumenta adptera mērķis ir nodrošināt iespēju iesaistītajām lietojumprogrammām bez detalizētām zināšanām par dokumentu glabāšanas sistēmu (RD LIS vai DAM) veikt gan vēsturisko dokumentu, gan jauno dokumentu apstrādi (reģistrāciju un izgūšanu) caur vienotu Dokumentu adaptera saskarni.



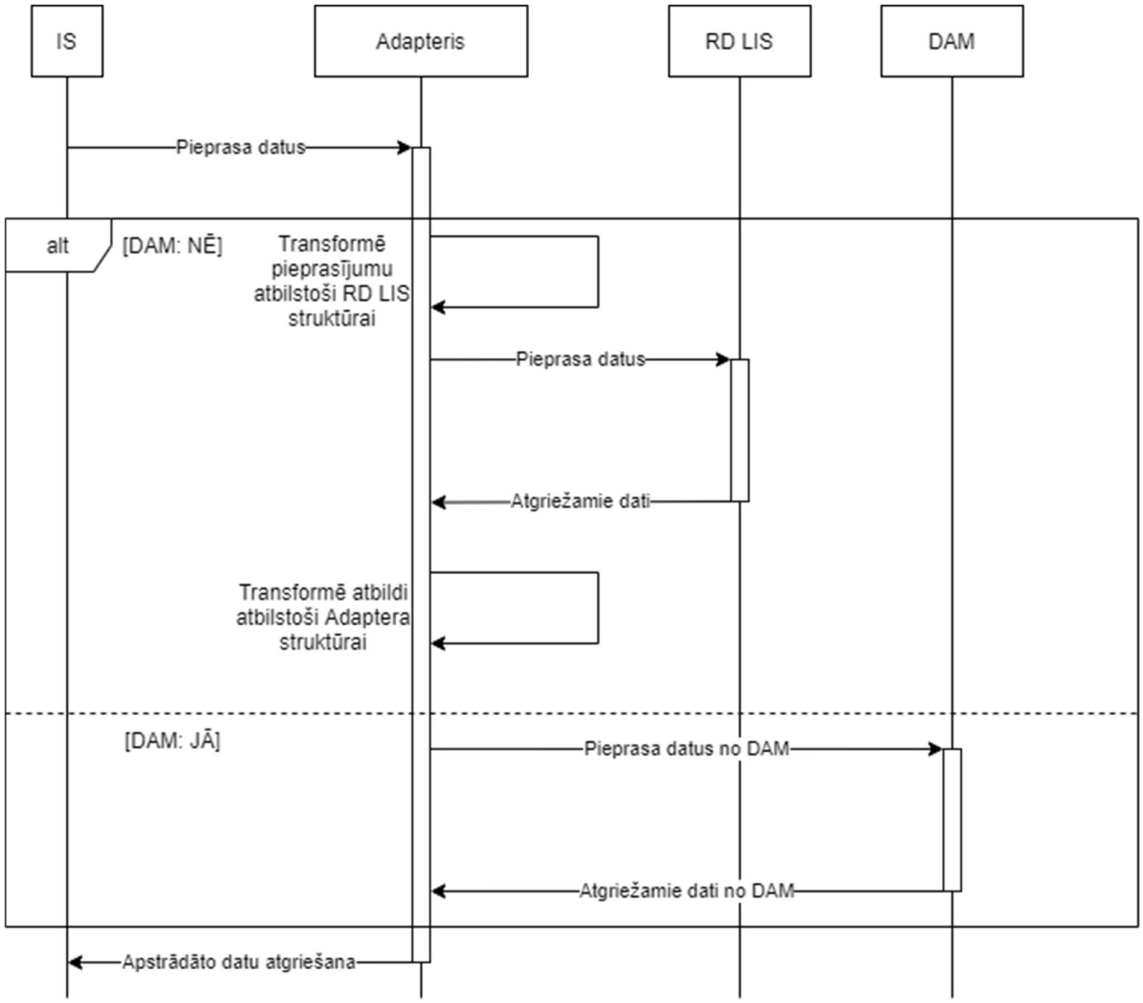
2.1. att. **Dokumenta adaptera darbības princips**

Dokumentu adapteris nosaka, kā var redzēt attēlā 2.1, kura DVS sistēma ir aktīva: RD LIS vai DAM. Uz visām sistēmām, kas izmanto adapteri, attiecas viens slēdža stāvoklis. Aktīvās sistēmas maiņa no RD LIS uz DAM un otrādi ir paredzēta tikai testēšanas posmā testa vidē. Tas ļaus sistēmām veikt pakāpenisku pāreju uz adaptera procedūrām, pārbaudot to darbību RD LIS un DAM. Dokumentu adapteris var tikt uzstādīts produkcijā pirms pārejas no RD LIS uz DAM, un šajā gadījumā aktīvā sistēma būs RD LIS. Pēc tam, kad produkcijā sāks lietot DAM, aktīvā sistēma būs DAM, un adaptera pārslēgšanās uz RD LIS vairāk nenotiks. Pēc pārslēgšanās uz DAM visām ārējām sistēmām ir jāizmanto adaptera procedūras.

2.2. Funkcionālās prasības

RD VIS lietojumprogrammas (ārējās sistēmas) pamazām pieslēdzas pie Adaptera. Līdz pieslēgšanās brīdim ārējās sistēmas izmanto esošo risinājumu integrācijai ar RD LIS.

1. Pieslēdzoties pie adaptera ārējai sistēmai jāizsauc adaptera procedūras un DB skati;
2. Adapteris nosaka, kura sistēma ir aktīva: RD LIS vai DAM, un izsauc aktīvās sistēmas procedūru (skatīt attēlu 2.2);



2.2. att. **Pieprasījuma un atbildes transformācija, pirms pārslēgšanās uz DAM**

3. Adapteris izmanto atbilstības tabulas, lai noteiktu DAM un RD LIS datu atbilstību. Atbilstības tabulas tiks izmantotas šādiem datiem:
   3.1..DAM procedūra – RD LIS procedūra. Ja RD LIS ir vairākas līdzīgas procedūras, tad DAM procedūra atbildīs pēdējai RD LIS procedūrai;
   3.2.Ārējās sistēmas dokumenta veids – RD LIS dokumenta veids;
   3.3.DAM piegādes veida kods – RD LIS piegādes veida identifikators;
   3.4.DAM dokumenta statusa kods – RD LIS dokumenta statusa ID.
4. Ja ārējā sistēma reģistrēs vai saskaņos savus dokumentus DAM, tad tai jāveido savas sistēmas dokumentu saraksts, kam tiks pēc tam veidota integrācija ar DAM. Pēc šī ārējās sistēmas dokumentu veida tiks noteikts vajadzīgais DAM vai RD LIS dokumenta veids;
5. Adapterī tiks veidota integrācijas sasaiste starp ārējās sistēmas dokumenta veidu un DAM, norādot parametrus:

5.1. Modulis – ārējā sistēma;

5.2. Dokuments – arējās sistēmas dokumenta veids;

5.3. Institūcija – ārējās sistēmas padotā institūcija. Ja netiks padota institūcija, tad to varēs izmantot visas institūcijas;

5.4. Dokumenta veids – DAM piesaistītais dokumenta veids. Dokumenta veidam ir piekārtots saskaņošanas nosacījums, kas nosaka plūsmu;

5.5. Dokumenta paveids – DAM dokumenta veida paveids. Paveidam var tikt noteikti savi saskaņošanas iestatījumi, kas nosaka plūsmas. Ja DAM dokumentam ir norādīts paveids, tad tiek izmantots nevis dokumenta veidam, bet paveidam piekārtotais saskaņošanas iestatījums;

5.6. Dokumenta nomenklatūras lietas indekss – DAM lietas indekss, kas tiks piedāvāts, izveidojot no ārējās sistēmas jaunu dokumenta kartīti. Norādīt lietas indeksu varēs tikai tad, ja būs norādīta institūcija;

5.7. Dokumenta jautājums – DAM jautājums, kas tiks piedāvāts, izveidojot no ārējās sistēmas jaunu dokumenta kartīti. Norādīt jautājumu varēs tikai tad, ja būs norādīta institūcija.

6. Izsaucot no ārējās sistēmas dokumenta reģistrāciju, tiks padots ārējās sistēmas dokumenta veids, un DAM dokumenta ieraksts tiks veidots, ņemot vērā izveidoto integrāciju. Ja ārējā sistēma pados citu lietas indeksu vai jautājumu nekā norādīts integrācijā, tad reģistrācijai tiks padots ārējās sistēmas padotais;
7. Ārējās sistēmas izsaucamās Adaptera procedūras un to atbilstību RD LIS skatīt pielikuma 1. tabulā;
8. Ja ārējā sistēmā tiek vienmēr izmantota viena klasifikatora vērtība, kas pašlaik ir ierakstīta kodā, tad vajag šīs RD LIS vērtības identifikatora vietā ierakstīt DAM klasifikatora kodu (DAM klasifikatoriem ir identifikators un kods). Tas var tikt izmantots norādot dokumenta statusu (galīgā atbilde, starpatbilde, uc) un piegādes veidu;
9. Procesā, kad tiek izmantots Adapteris, ārējai sistēmai jāveic šādas darbības:

   9.1. Ja ārējai sistēmai vajag izsaukt procedūru, kur jāpadod DVS dokumenta identifikators, tad ārējā sistēma vispirms izsauc Adaptera procedūru *LIST*, kam kā parametrus padod saglabātos vai lietotāja norādītos dokumenta parametrus: dokumenta numurs un dokumenta datums. Tas jādara tāpēc, ka var būt gadījumi, kad saglabāts ir RD LIS dokumenta identifikators, bet aktīvā sistēma ir DAM, kam ir cits dokumenta identifikators;

9.2. Ja ārējā sistēma procedūrā izmanto klasifikatora vērtības, tad vajag izsaukt adaptera procedūru, kas atgriež klasifikatoru vērtības. Adapteris izsauks aktīvo DVS sistēmu un izgūst šīs sistēmas datus;

9.3. Ārējā sistēma atgriezto dokumenta identifikatoru un dokumenta veidu un atgrieztās klasifikatoru vērtības padod nākošās procedūras izsaukumam, kas nodrošinās, ka adaptera procedūras izsaukumam tiks padoti dati, kas atbilst aktīvajai DVS;

10. Adapteris nosaka aktīvo DVS un veic šādas darbības:

10.1. Ja aktīvā ir DAM sistēma, tad visi padotie parametri tiek padoti, izsaucot DAM procedūru. DAM pēc ārējās sistēmas dokumenta veida identifikatora, atrod DAM – ārējās sistēmas integrācijas tabulā DAM dokumenta veidu, paveidu, lietas indeksu, jautājumu;

10.2. Ja aktīvā sistēma ir RD LIS, tad pirms RD LIS procedūras izsaukuma, adapteris veic šādu datu pielāgošanu:

10.2.1. darbinieka slodzes identifikatoru padod kā vārds, uzvārds, amats;

10.2.2. struktūrvienību padod kā RD LIS struktūrvienības identifikatoru (LIETO ievadītais RD LIS struktūrvienības identifikators);

10.2.3. ja klasifikatora vērtību gadījumā tika padots kods nevis identifikators, tad adapteris pēc koda atrod atbilstības tabulā RD LIS klasifikatora vērtības identifikatoru;

10.2.4. pēc ārējās sistēmas dokumenta veida identifikators tiek atrasts un padots RD LIS dokumenta veida identifikators;

10.2.5. ja tiek izsaukta *LIST* vai *SELECTS*, tad padod parametru *source=6*, kas atlasa visus pieejamos dokumentus (RD LIS atgriež tikai dokumentus, kurus lietotājs var redzēt).

11. Aktīvajā sistēmā tiek veiktas izsauktās procedūras darbības. Rezultāts tiek atgriezts adapterim, kas to atgriež ārējai sistēmai. Ja dati tika saņemti no RD LIS, tad tiek veiktas darbības:

11.1. padod darbinieka slodzes identifikatoru , ja atgriezts darbinieka vārds, uzvārds, amats;

11.2. struktūrvienību padod kā RD VIS struktūrvienības identifikatoru (LIETO ievadītais RD LIS struktūrvienības identifikators);

11.3. dati, kas tiek saņemti no RD LIS kursorā veidā, tiek saglabāti adaptera pagaidu tabulās, no kurām ārējā sistēma izgūst nepieciešamā kursora vērtības.

12. Visi procedūru ievaddatu un izvaddatu apraksti ir pielikuma sadaļā, kur redzami gan RD LIS procedūras dati, gan jaunie adaptera izsaukuma parametri. Šīs nodaļas apakšnodaļās ir detalizēti aprakstītas, katras izsaucamās procedūras izmaiņas, un kā to apstrādās adapteris;
13. Ārējās sistēmas, kas izmanto adapteri, turpmāk izmantos adaptera datu bāzes skatus:
    13.1. Adaptera datu bāzes skatu struktūra var atšķirties no esošajiem RD LIS datiem.
    13.2. Daļa RD LIS skatu tiks apvienoti, piemēram, dokumentu saraksts būs vienots nevis dalīts pa RD LIS kartotēkām
    13.3. Daļa esošo RD LIS skatu ārējām sistēmām būs jāizstāj ar procedūrām, piemēram, jautājumu saraksta izgūšanai.
14. Ārējai sistēmai izsaucot adaptera skatu, adapteris noteiks aktīvo sistēmu:
    14.1. Ja aktīvā sistēma ir DAM, tad atgriezīs DAM skata datus;
    14.2. Ja aktīvā sistēma ir RD LIS, tad atgriezīs RD LIS skata datus. Apvienoto skatu gadījumā Adaptera skatā tiks apvienota vairāku RD LIS skatu informācija.

2.2.1. Ienākošā dokumenta izveide

Identifikators

SAVE_INCOMMING

Ievads

Procedūras mērķis – nodrošināt iespēju reģistrēt ienākošos dokumentus (saņemtos dokumentus, personu iesniegumus).

Ievade

Procedūra saņem ienākošos datus JSON formātā un pārveido tos atbilstoši datu tipiem tabulā 2.1.

2.1. tabula. **SAVE_INCOMMING ievadlauki.**

Lauka nosaukums	Datu tips	Neobligāts/Obligāts
Iesnieguma identifikators ārējā sistēmā	Vesels skaitlis	Obligāts

Lauka nosaukums	Datu tips	Neobligāts/Obligāts
Reģistrējama dokumenta avots	Vesels skaitlis	Obligāts
Ārējās sistēmas dokumenta veida identifikators	Simbolu virkne (no 1 līdz 30 simboli)	Obligāts
Dokumenta anotācija	Simbolu virkne (no 1 līdz 2000 simboli)	Obligāts
Dokumenta saimnieka DAM institūcijas identifikators	Vesels skaitlis	Obligāts
Ienākošā dokumenta pašvaldības adresāti	Masīvs	Obligāts
Ienākošā dokumenta adresāta slodzes identifikators	Vesels skaitlis	Obligāts
Ienākošā dokumenta pašvaldības identifikators	Vesels skaitlis	Neobligāts
Ienākošā dokumenta pašvaldības darbinieka slodzes identifikators	Vesels skaitlis	Neobligāts
Dokumenta reģistrētāja struktūrvienības identifikators	Vesels skaitlis	Daļēji
Dokumenta reģistrētāja slodzes identifikators	Vesels skaitlis	Neobligāts
Jautājuma identifikators	Vesels skaitlis	Neobligāts

Lauka nosaukums	Datu tips	Neobligāts/Obligāts
Lietu nomenklatūras identifikators	Vesels skaitlis	Neobligāts
Lapu skaits	Vesels skaitlis	Neobligāts
Saistītie dokumenti	Masīvs	Neobligāts
Dokumenta atbildes termiņš	Datums un laiks	Neobligāts
Atbildes pazīme	Vesels skaitlis (0 vai 1)	Neobligāts
DAM dokumenta statusa kods	Simbolu virkne (no 1 līdz 7 simboli)	Neobligāts
Korespondenta dokumenta datums	Datums un laiks	Neobligāts
Korespondenta dokumenta numurs	Simbolu virkne (no 1 līdz 22 simboli)	Neobligāts
Saite uz ārējās sistēmas dokumentu	Simbolu virkne (no 1 līdz 500 simboli)	Neobligāts
Korespondentu saraksts	Masīvs	Obligāts
Pievienoto datņu saraksts	Masīvs	Neobligāts

Apstrāde

1. Atšķirības adaptera dokumenta izveides procedūrai (pilnu specifikāciju skatīt pielikuma 2.tabulā) no RD LIS dokumentu izveides procedūrām:
   1.1. Izveidota atsevišķa procedūra ienākošo dokumentu reģistrācijai;
   1.2. Procedūrā ir iekļautas šādas RD LIS procedūras: *SAVE*, *Add_Person*, *Add_Document*.
   1.3. Būtiskākās izmaiņas ievaddatos:
      1.3.1. *Source -> AppID* – sistēma, kas izsauc procedūru;
      1.3.2. *DocTypeID* – Ārējās sistēmas dokumenta veida identifikators nevis DVS dokumenta veida identifikators;

1.3.3. kā struktūrvienības identifikators tiek padots RD VIS struktūras identifikators nevis RD LIS identifikators;

1.3.4. par pašvaldības darbiniekiem padod slodzes identifikatoru un struktūras identifikatoru;

1.3.5. saistītie dokumenti (*RelatedDocID*) un atbildes dokumenti (*ReplayToDocID*) tiek padoti kopējā masīvā (ID, dokumenta numurs, tips). Dokumenta ID atrod izsaucot *LIST*;

1.3.6. dokumenta statuss tiek padots kā DAM kods nevis identifikators. Kodu atrod izsaucot procedūru *Document_Status_List*;

1.3.7. korespondenta dokumenta numurs un datums netiek padots pie korespondenta, bet pie pamatdatiem;

1.3.8. korespondentu dati tiek padoti masīvā, kam struktūra ir līdzīga *Add_Person*;

1.3.9. datnes tiek padotas masīvā, kam struktūra ir līdzīga *Add_Document*;

1.4. Ja ārējās sistēmas kodā tiek ierakstītas noteiktas klasifikatoru vērtības, tad tās jānomaina no RD LIS ID uz DAM klasifikatora vērtībām. To var izmantot klasifikatoriem:

1.4.1. piegādes veids, norādot piegādes veida DAM kodu (atrod izsaucot *Delivery_Type_List*);

1.4.2. dokumenta statuss, norādot dokumenta statusa kodu (atrod izsaucot *Document_Status_List*).

1.5. Ja Adaptera iestatījumos ir noteikta dokumenta reģistrācija DAM, tad tiks izsaukta DAM procedūra, izmantojot padotos parametrus. DAM tiks izveidots dokumenta ieraksts, kam

1.5.1. dokumenta veids tiks atrasts pēc padotā ārējās sistēmas dokumenta veida adaptera integrācijas tabulā;

1.5.2. dokumentam tiks pievienota informācija par korespondentiem;

1.5.3. pievienotas datnes.

1.6. Ja adaptera iestatījumos ir noteikta dokumenta reģistrācija RD LIS, tad izsaukums tiks transformēts uz RD LIS procedūru *SAVE4*, *AddPerson2*, *AddDocument2* izsaukumu, piemērojot šādu datu apstrādi:

1.6.1. *SAVE4* procedūrā:

1.6.1.1. pēc ārējās sistēmas dokumenta veida ID tiek atrasts un padots RD LIS dokumenta veida ID;

1.6.1.2. struktūrvienības ID padod kā RD LIS struktūrvienības ID (LIETO ievadītais RD LIS struktūrvienības ID);

1.6.1.3. darbinieka slodzes ID tiks padod kā vārds, uzvārds, amats, personas kods;

1.6.1.4. ja klasifikatora vērtību gadījumā tika padots kods nevis ID, tad pēc koda atrod atbilstības tabulā RD LIS klasifikatora vērtības ID;

1.6.1.5. padodamā *source* vērtība tiek atrasta atbilstības tabulā.

1.6.2. *AddPerson2* procedūrā:

1.6.2.1. tiek padota informācija par *SAVE4* reģistrēto dokumentu;

1.6.2.2. darbinieka slodzes ID tiks padod kā vārds, uzvārds, amats;

1.6.2.3. pēc padotā koda tiek atrasts piegādes veida ID;

1.6.2.4. procedūra tiek izsaukta katram korespondentam atsevišķi.

1.6.3. *AddDocument2* procedūrā:

1.6.3.1. tiek padota informācija par *SAVE4* reģistrēto dokumentu;

1.6.3.2. darbinieka slodzes ID tiek padots kā vārds, uzvārds, amats;

1.6.3.3. tiek padota papildus informācija par datni, kas saglabāta repozitorijā;

1.6.3.4. procedūra tiek izsaukta katrai datnei atsevišķi.

2. Aktīvā sistēma atgriezīs adapterim procedūras izvaddatus, kas tiks atgriezti ārējai sistēmai.

Izvade

Nav.

Paziņojumi

Nav.

2.2.2. Datņu pievienošana

Identifikators

ADD_DOCUMENT

Ievads

Nodrošina iespēju pievienot datni iepriekš saglabātam dokumentam. Izmanto gadījumos, kad datne tiek ģenerēta ārējā sistēmā pēc dokumenta reģistrācijas vai tiek pievienota vēlāk.

Ievade

Adapteris saņem ieejas datus no ārējās sistēmas, kurus var aplūkot tabulā 2.2.

2.2. tabula. **ADD_DOCUMENT ievadlauki.**

Lauka nosaukums	Datu tips	Neobligāts/Obligāts
Dokumenta identifikators	Vesels skaitlis	Obligāts
Reģistrētāja struktūrvienības identifikators	Vesels skaitlis	Obligāts
Reģistrētajā slodzes identifikators	Vesels skaitlis	Obligāts
Ierobežotās pieejamības pazīme	Vesels skaitlis	Obligāts
Repozitorija identifikators	Vesels skaitlis	Neobligāts
Dokumenta veida identifikators	Vesels skaitlis	Obligāts

Apstrāde

1. Atšķirības Adaptera datnes pievienošanas procedūrai (pilnu specifikāciju skatīt pielikuma 4.tabulā) no RD LIS datnes pievienošanas procedūrām:
    1.1. Adapteris izmanto tikai vienu procedūru *Add_Document*, bet RD LIS ir *Add_Document*, *Add_Document2*, kas savstarpēji atšķiras ar ieejas parametriem;

1.2. Būtiskākās izmaiņas ievaddatos:

1.2.1. par lietotāju tiek padots lietotāja slodzes ID;

1.2.2. kā struktūrvienības ID tiek padots RD VIS struktūras ID nevis RD LIS ID;

1.2.3. par pievienoto datni tiek padots tikai repozitorija ID un informācija par ierobežotu pieeju. Pārējos datnes metadatus DAM vai Adapteris izgūs no repozitorija.

1.3. Ja procedūra netiek izsaukta uzreiz pēc dokumenta saglabāšanas, tad datu migrācijas dēļ var rasties problēmas ar dokumenta identifikatora vērtību (saglabāts RD LIS ID, bet aktīvā sistēma ir DAM, kam ir cits dokumenta ID). Tāpēc šajos gadījumos ārējai sistēmai vajag izsaukt procedūru *LIST*, lai atrastu pēc saglabātā dokumenta numura, datuma un ID, aktīvās sistēmas dokumenta ID.

1.4. Saglabāts princips, ka ar vienu procedūras izsaukumu var pievienot vienu datni.

2. Ja Adaptera iestatījumos aktīvā sistēma ir DAM, tad tiks izsaukta DAM procedūra, kas pievienos datni DAM dokumentam.
3. Ja Adaptera iestatījumos ir noteikta aktīvā sistēma RD LIS, tad izsaukums tiks transformēts uz RD LIS procedūru *Add_Document* izsaukumu, piemērojot šādu datu apstrādi:

3.1. struktūrvienības ID padod kā RD LIS struktūrvienības ID (LIETO ievadītais RD LIS struktūrvienības ID);

3.2. darbinieka slodzes ID tiek padots kā darbinieka dati;

3.3. tiek padota papildus informācija par datni, kas saglabāta repozitorijā.

Izvade

Nav.

Paziņojumi

Nav.

2.2.3. Dokumenta datnes dzēšana

Identifikators

DELETE_DOCUMENT

Ievads

Nodrošina iespēju nodzēst kļūdaini pievienotu dokumenta datni dokumentam, kas veidots no ārējās sistēmas.

Ievade

Dokumenta datnes dzēšanas ievadlaukus var apskatīt tabulā 2.3.

2.3. tabula. **DELETE_DOCUMENT ievadlauki.**

Lauka nosaukums	Datu tips	Neobligāts/Obligāts
Dokumenta ieraksta identifikators	Vesels skaitlis	Obligāts
Ārējās sistēmas dokumenta veida identifikators	Simbolu virkne (no 1 līdz 30)	Daļēji
Datnes repozitorija identifikators	Vesels skaitlis	-
Datnes nosaukums	Simbolu virkne (no 1 līdz 240)	-

Apstrāde

1. Atšķirības Adaptera dokumenta datnes dzēšanas procedūrai (pilnu specifikāciju skatīt pielikuma 5.tabulā) no RD LIS dokumenta datnes dzēšanas procedūras:
    1.1. Adapteris izmanto procedūru *Delete_Document*, un RD LIS izmanto *Delete_Document*.
    1.2.Būtiskākās izmaiņas ievaddatos:
        1.2.1. dokumenta veida vietā tiek padots ārējās sistēmas dokumenta veids;
    1.3.Ja procedūra netiek izsaukta uzreiz pēc dokumenta saglabāšanas, tad datu migrācijas dēļ var rasties problēmas ar dokumenta identifikatora vērtību (saglabāts RD LIS ID, bet aktīvā sistēma ir DAM, kam ir cits dokumenta ID). Tāpēc šajos gadījumos ārējai sistēmai vajag izsaukt procedūru *LIST*, lai atrastu

pēc saglabātā dokumenta numura, datuma un ID, aktīvās sistēmas dokumenta ID.

2. Ja Adaptera iestatījumos aktīvā sistēma ir DAM, tad tiks izsaukta DAM procedūra, kas dzēš dokumenta datni, ja dokuments būs veidots no ārējās sistēmas.
3. Ja Adaptera iestatījumos aktīvā sistēma ir RD LIS, tad izsaukums tiks transformēts uz RD LIS procedūras *Delete_Document* izsaukumu, piemērojot šādu datu apstrādi:
    3.1. pēc ārējās sistēmas dokumenta veida ID tiek atrasts un padots RD LIS dokumenta veida ID.

Izvade

Nav.

Paziņojumi

Ja datne netiek atrasta DB, tad tiek attēlots paziņojums: "Datni nav iespējams identificēt".

2.2.4. Detalizētās informācijas dokumentu izgūšana

Identifikators

SELECTS

Ievads

Procedūra nodrošina iespēju atlasīt informāciju par konkrētu dokumentu.

Ievade

Ārējā sistēma padod sekojošo informāciju, lai saņemtu datus par konkrētu dokumentu: dokumenta identifikators, dokumenta numurs lietvedībā, pazīme datņu informācijas iegūšanai, lietotāja identifikators un lietotāja struktūrvienības identifikators. Apkopojumu par padotiem parametriem var redzēt tabulā 2.4.

2.4. tabula. **SELECTS ievadlauki.**

Lauka nosaukums	Datu tips	Neobligāts/Obligāts
Identifikators	Vesels skaitlis	Obligāts
Dokumenta numurs lietvedībā	Simbolu virkne (no 1 līdz 35)	Obligāts
Pazīme repozitorija informācijas atgriešanai	Vesels skaitlis (0, 1 vai 2)	Obligāts
Lietotāja slodzes identifikators	Vesels skaitlis	Obligāts
Lietotāja struktūrvienības identifikators	Vesels skaitlis	Obligāts

Apstrāde

1. Atšķirības Adaptera procedūrai detalizētas informācijas iegūšanai par dokumentu (pilnu specifikāciju skatīt pielikuma 6.tabulā) no RD LIS dokumenta datu meklēšanas procedūrām:
    1.1. Adapteris izmanto tikai vienu procedūru *SELECTS*, bet RD LIS ir *SELECTS*, *SELECT2*, *SELECT3*, *SELECT4*¸ *SELECT5*, *SELECT_REZ*, kas savstarpēji atšķiras ar ieejas parametriem;
    1.2.Būtiskākās izmaiņas ievaddatos:
        1.2.1. netiek izmatoti parametri par dokumenta veidu un kartotēku, tāpēc vienmēr vajag padot dokumenta numuru un ID;
        1.2.2. netiek izmantots parametrs *source*, kura vērtība 6, ļāva meklēt starp visiem dokumenta veidiem. DAM atgriezīs tikai tos dokumentus, kas lietotājam būs pieejami DAM;
        1.2.3. par lietotāju tiek padots lietotāja slodzes ID;
        1.2.4. kā struktūrvienības ID tiek padots RD VIS struktūras ID nevis RD LIS ID.
    1.3. Ja dokumenta numurs un ID bija iepriekš saglabāts ārējā sistēmā, tad vajag vispirms izsaukt *LIST*, lai atrastu pēc dokumenta numura, datuma un ID, aktīvās

sistēmas ID. Pirms datu migrācijas tika saglabāts RD LIS ID, bet tagad aktīvā sistēma ir DAM, kam ir cits dokumenta ID.

2. Ja Adaptera iestatījumos aktīvā sistēma ir DAM, tad tiks izsaukta DAM procedūra, kas meklēs DAM dokumentu sarakstā.
3. Ja Adaptera iestatījumos aktīvā sistēma ir RD LIS, tad izsaukums tiks transformēts uz RD LIS procedūras *SELECT5* izsaukumu, piemērojot šādu datu apstrādi:
    3.1. struktūrvienības ID padod kā RD LIS struktūrvienības ID (LIETO ievadītais RD LIS struktūrvienības ID);
    3.2. darbinieka slodzes ID tiek padots kā personas kods;
    3.3. *source=6*;
4. Aktīvā sistēma atgriezīs adapterim procedūras izvaddatus, kas pēc apstrādes tiks atgriezti ārējai sistēmai.
    4.1. Adapteris izgūst datus pagaidu tabulās, kur tos pielāgo atbilstoši definētajai izvaddatu struktūrai. Ārējā sistēma izgūst nepieciešamā kursora datus.
    4.2. Ja dati tiks izgūti no RD LIS, tad tiks veikti šādi pielāgojumi:
        4.2.1. padod darbinieka slodzes ID, ja tika atgriezts darbinieka vārds, uzvārds, amats;
        4.2.2. struktūrvienību padod kā RD VIS struktūrvienības ID (LIETO ievadītais RD LIS struktūrvienības ID);
    4.3. Migrētiem datiem kursors cResponse, kas atgriež aktīvos rezolūcijas izpildītājus, var nebūt datu vai dati var būt nepilnīgi.

Izvade

Pēc datu izgūšanas attiecīgajās pagaidu tabulās un to pielāgošanās, ja tika izgūti no RD LIS, dati tiek atgriezti kursoros, kurus ārējā sistēma iegūst vajadzīgos datus.

Paziņojumi

Nav.

2.2.5. Virtuālās lietas atvienošana

Identifikators

REMOVE_EXECUTIVE_OBJECT

Ievads

Nodrošina iespēju no virtuālās lietas (administratīvās, piedziņas lietas) atvienot dokumentu.

Ievade

Adapteris saņem no ārējās sistēmas ievaddatus, kas tiek attēloti tabulā 2.5.

2.5. tabula. **REMOVE_EXECUTIVE_OBJECT ievadlauki.**

Lauka nosaukums	Datu tips	Neobligāts/Obligāts
Dokumenta identifikators	Vesels skaitlis	Obligāts
Administratīvās lietas identifikators	Vesels skaitlis	Obligāts
Dokumenta veida identifikators	Vesels skaitlis	-

Apstrāde

1. Atšķirības Adaptera procedūrai (pilnu specifikāciju skatīt pielikuma 8.tabulā) no RD LIS procedūras:
    1.1. Adapteris izmanto procedūru *RemoveExecutiveObject*, un RD LIS izmanto *RemoveExecutiveObject*, *RemoveExecutiveObject2*, kam atšķiras ieejas parametri;
    1.2. Izmaiņas ievaddatos nav;
    1.3. Lai noteiktu dokumenta ID, tad vajag izsaukt procedūru *LIST*, kas atgriezīs aktīvās sistēmas dokumenta ID.
2. Ja Adaptera iestatījumos aktīvā sistēma ir DAM, tad tiks izsaukta DAM procedūra.
3. Ja Adaptera iestatījumos aktīvā sistēma ir RD LIS, tad tiks izsaukta RD LIS procedūra *RemoveExecutiveObject2*.

Izvade

Nav.

Paziņojumi

Nav.

2.2.6. Dokumenta saistīto dokumentu saraksti un virtuālās lietas

Identifikators

CORRESPONDENCE_LIST

Ievads

Adaptera procedūra nodrošina iespēju izgūt informāciju par dokumenta saistītajiem dokumentiem un virtuālajām lietām.

Ievade

Adaptera procedūra saņem dokumenta identifikatoru, dokumenta reģistrācijas numuru, dokumenta veida identifikatoru, struktūrvienības identifikatoru, administratīvās lietas identifikatoru un administratīvās lietas reģistrācijas numuru, kas ir redzams tabulā 2.5.

2.6. tabula. **CORRESPONDENCE_LIST ievadlauki.**

Lauka nosaukums	Datu tips	Neobligāts/Obligāts
Dokumenta identifikators	Vesels skaitlis	Obligāts
Dokumenta reģistrācijas numurs	Simbolu virkne (no 1 līdz 35 simboliem)	Neobligāts
Dokumenta veida identifikators	Vesels skaitlis	Neobligāts
Struktūrvienības identifikators	Vesels skaitlis	Obligāts
Administratīvās lietas identifikators	Vesels skaitlis	Neobligāts
Administratīvās lietas reģistrācijas numurs	Simbolu virkne (no 1 līdz 35 simboliem)	Neobligāts

Apstrāde

1. Atšķirības Adaptera procedūrai (pilnu specifikāciju skatīt pielikuma 9.tabulā) no RD LIS procedūras:
    1.1. Adapteris izmanto procedūru *CORRESPONDENCE_LIST*, un RD LIS izmanto *CORRESPONDENCE_LIST*.
    1.2.Izmaiņas ievaddatos:
        1.2.1. netiek padots *source*;
    1.3.DAM nav integrācija ar BIS un netiek saglabāta informācija par BIS lietām.
2. Ja adaptera iestatījumos aktīvā sistēma ir DAM, tad tiks izsaukta DAM procedūra.
3. Ja adaptera iestatījumos aktīvā sistēma ir RD LIS, tad tiks izsaukta RD LIS procedūra *CORRESPONDENCE_LIST*, piemērojot šādu datu apstrādi:
    3.1.*source=6*.
4. Aktīvā sistēma atgriezīs adapterim procedūras izvaddatus, kas pēc apstrādes tiks atgriezti ārējai sistēmai.
    4.1. Adapteris izgūst datus pagaidu tabulās, kur tos pielāgo atbilstoši definētajai izvaddatu struktūrai. Ārējā sistēma izgūst nepieciešamā kursora datus.

Izvade

Ārējā sistēma saņem atgriežamos datus iekš kursoriem: cDoc, cExec un cDocExec.

Paziņojumi

Nav.

2.2.7. Korespondences dokumentu statusi

Identifikators

DOCUMENT_STATUS_LIST

Ievads

Procedūra nodrošina iespēju izgūt institūcijas nomenklatūras lietu sarakstu.

Ievade

Ārējā sistēma nepadod ievaddatus adaptera procedūrai.

Apstrāde

1. Atšķirības Adaptera procedūrai (pilnu specifikāciju skatīt pielikuma 11.tabulā) no RD LIS procedūras:
    1.1. Adapteris izmanto procedūru *Document_Status_List*, un RD LIS izmanto *Document_Status_List*.
    1.2. Ievaddatos izmaiņas nav.
    1.3. Izmaiņas ir izvaddatos, kas papildināti ar DAM kodu, kas jāpadod izsaucot reģistrācijas procedūras.
2. Ja adaptera iestatījumos aktīvā sistēma ir DAM, tad tiks izsaukta DAM procedūra.
3. Ja adaptera iestatījumos aktīvā sistēma ir RD LIS, tad tiks izsaukta RD LIS procedūra *Document_Status_List*.
4. Aktīvā sistēma atgriezīs adapterim procedūras izvaddatus, kas pēc apstrādes tiks atgriezti ārējai sistēmai. Adapteris izgūst datus pagaidu tabulās, kur tos pielāgo atbilstoši definētajai izvaddatu struktūrai. Ārējā sistēma izgūst nepieciešamos kursora datus.

Izvade

Ārējā sistēma saņem datus no adaptera kursora cDocumentStatusList.

Paziņojumi

Nav.

2.2.8. Korespondences jautājumi

Identifikators

SUBJECT_LIST

Ievads

Nodrošina iespēju izgūt institūcijas jautājumu sarakstu.

Ievade

Adaptera procedūra saņem no ārējās sistēmas struktūrvienības identifikatoru (veselais skaitlis).

Apstrāde

1. Atšķirības adaptera procedūrai (pilnu specifikāciju skatīt pielikuma 12.tabulā) no RD LIS procedūras:
    1.1. Adapteris izmanto procedūru *Subject_List*, un RD LIS izmanto *Subject_List*. Ievaddatos izmaiņas nav.
    1.2. Izmaiņas ir izvaddatos, jo DAM nav dalījums fiziskas un juridiskas personas jautājumos.
2. Ja adaptera iestatījumos aktīvā sistēma ir DAM, tad tiks izsaukta DAM procedūra.
3. Ja adaptera iestatījumos ir noteikta aktīvā sistēma RD LIS, tad tiks izsaukta RD LIS procedūra *Subject_List*.
4. Aktīvā sistēma atgriezīs Adapterim procedūras izvaddatus, kas pēc apstrādes tiks atgriezti ārējai sistēmai. Adapteris izgūst datus pagaidu tabulās, kur tos pielāgo atbilstoši definētajai izvaddatu struktūrai. Ārējā sistēma izgūst nepieciešamos kursora datus.

Izvade

Ja tika norādīts struktūrvienības identifikators, tad adapteris atgriezīs kursorā cSubjectList korespondences jautājumu klasifikatoru, kas attiecas uz uzdoto struktūrvienību. Ja struktūrvienība netika norādīta, tad kursorā cSubjectList tiks atgriezti visi jautajumi.

Paziņojumi

Nav.

3. PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS

3.1. Datubāzes projektējums

Dokumentu adapteris ir datubāzes daļa, kas padod datus uz RD LIS vai DAM, izgūst datus no RD LIS vai DAM un apstrādā tos, lai ārējās sistēmas varētu tos izmantot. Adaptera izstrāde notiek Oracle datubāzē izmantojot procedurālo valodu PL/SQL un veidojot DB pagaidu tabulas un kursorus datu glabāšanai un izgūšanai no sistēmām.

3.1.1. Kursoru projektējums

Ārējās sistēmas izsaucot adaptera procedūras iegūst datus no kursoriem, kas tiek padoti adapterim. Kursora izveidošanai nepieciešams izveidot datu tipu, kas atbilst tabulas ieraksta laukiem, un izmantot šo datu tipu kā pamatu veidojot kursora datu tipu – ligzdtabulu (Nested table) jeb masīvu ar tabulas ierakstiem.

3.1.1.1. Kursors "subject_list_tt"

Kursorā "subject_list_tt" tiek glabāti dati par korespondences jautājumiem. Kursors sastāv no jautājuma identifikatora, jautājuma nosaukuma, ieraksta arhivēšanas pazīmes un struktūrvienības identifikatora. Apkopoto informāciju par kursoru var apskatīt tabulā 3.1.

3.1. tabula **subject_list_tt**

Lauka nosaukums	Datu tips	Apraksts
ID	number(10)	Identifikators
Name	varchar2(250)	Nosaukums
Archived	number(1)	Ieraksta arhivēšanas pazīme (0 – nearhivēts, 1 – arhivēts)
DivisionID	number(10)	Struktūrvienības identifikators

3.1.1.2. Kursors "document_status_list_tt"

Kursors "document_status_list_tt" glabā datus par korespondences dokumentu statusiem. Kursorā ir identifikators, dokumenta statusa veids, nosaukums, kārtas numurs, ieraksta arhivēšanas pazīme un DAM statusa kods. Detalizētu informāciju par laukiem un datu tipiem var apskatīt tabulā 3.2.

3.2. tabula **document_status_list_tt**

Lauka nosaukums	Datu tips	Apraksts
ID	number(10)	Identifikators
Type	number(1)	Dokumentu statusu veidi (1- Ienākošo dokumentu statuss, 2 – izejošo dokumentu statuss)
Name	nvarchar2(70)	Nosaukums
ORDER_INDEX	number(10)	Kārtas numurs
Archived	number(1)	Ieraksta arhivēšanas pazīme (0 – nearhivēts, 1 – arhivēts)
Code	varchar2(7)	DAM status kods

3.1.1.3. Kursors "cDoc_list_tt"

Kursors "cDoc_list_tt" satur informāciju par sarkastes dokumentiem, kas iekļauj sevī dokumenta identifikatoru, dokumenta veidu, reģistrācijas datumu un piesaistīto administratīvo lietu sarakstu. Precīza informācija atrodas tabulā 3.3.

3.3. tabula **cDoc_list_tt**

Lauka nosaukums	Datu tips	Apraksts
SubmissionID	number(10)	Dokumenta identifikators
SubmissionDocTypeID	number(10)	Dokumenta veida identifikators
SubmissionDocType	varchar2(100)	Dokumenta veida nosaukums
SubmissionDocNr	nvarchar2(35)	Dokumenta reģistrācijas Nr.
SubmissionDocDate	date	Dokumenta reģistrācijas datums
SubmissionDocAnnotation	nclob	Dokumenta anotācija
SubmissionDocCommentary	nvarchar2(2000)	Dokumenta piezīmes
ExecObjects	varchar2(500)	Piesaistīto administratīvo lietu saraksts (reģistrācijas Nr. atdalīti ar komatu un atstarpi)

Lauka nosaukums	Datu tips	Apraksts
InitDoc	number(1)	Pazīme, vai tas ir sarakstes sākotnējais dokuments (1,0)

3.1.1.4. Kursors "cExec_list_tt"

Kursorā "cExec_list_tt" tiek glabāti dati par sarakstes administratīvām lietām. Kursors sastāv no administratīvās lietas identifikatora, administratīvās lietas veida identifikatora, reģistrācijas numura un datuma, anotācijas, piezīmes, saimnieka struktūrvienības identifikatora un administratīvās lietas statusa. Apkopoto informāciju par kursoru un to lauku datu tipiem var apskatīt tabulā 3.4.

3.4. tabula **cExec_list_tt**

Lauka nosaukums	Datu tips	Apraksts
ExecObjectID	number(10)	Administratīvās lietas identifikators
ExecObjectDocTypeID	number(10))	Administratīvās lietas veida identifikators
ExecObjectNr	nvarchar2(35)	Administratīvās lietas reģistrācijas Nr.
ExecObjectDate	date	Administratīvās lietas reģistrācijas datums
ExcObjectAnnotation	nclob	Administratīvās lietas anotācija
ExecObjectCommentary	varchar2(2000)	Administratīvās lietas piezīmes
ExecObjectOwnerDivisionID	number(10)	Administratīvās lietas saimnieka struktūrvienības identifikators
ExecObjectStatus	varchar2(250)	Administratīvās lietas statuss

3.1.1.5. Kursors "cDocExec_list_tt"

Kursors "cDocExec_list_tt" glabā datus par administratīvo lietu dokumentiem. Kursors iekļauj sevī dokumenta identifikatoru, dokumenta veida identifikatoru, dokumenta reģistrācijas numuru un reģistrācijas datumu, BIS dokumenta identifikatoru un numuru, administratīvās lietas identifikatoru, lietas veida identifikatoru, reģistrācijas numuru un reģistrācijas datumu. Apkopoto informāciju par kursoru var apskatīt tabulā 3.5.

3.5. tabula **cDocExec_list_tt**

Lauka nosaukums	Datu tips	Apraksts
SubmissionID	number(10)	Dokumenta identifikators
SubmissionDocTypeID	number(10)	Dokumenta veida identifikators
SubmissionDocNr	nvarchar2(35)	Dokumenta reģistrācijas Nr.
SubmissionDocDate	date	Dokumenta reģistrācijas datums
BisSubmissionID	number(10)	BIS dokumenta identifikators
BisSubmissionNr	varchar2(22)	BIS dokumenta Nr.
ExecObjectID	number(10)	Administratīvās lietas identifikators
ExecObjectDocTypeID	number(10))	Administratīvās lietas veida identifikators
ExecObjectNr	nvarchar2(35)	Administratīvās lietas reģistrācijas Nr.
ExecObjectDate	date	Administratīvās lietas reģistrācijas datums

3.1.1.6. Kursori priekš procedūras "SELECTS"

Procedūrā "SELECTS" tiek izmantoti vairāki kursori datu iegūšanai:

1. "cSubmission_list_tt" – dokumenta kartītes pamatdati;
2. "cDocument_list_tt" – dokumentam pievienotās datnes;
3. "cPerson_list_tt" – dati par korespondentu vai adresātiem, un parakstītājiem;
4. "cResponse_list_tt" – ar dokumentu saistītie atbildes dokumenti;
5. "cDivision_list_tt" – dati par darbiniekiem pie kā atrodas dokuments;
6. "cExec_obj_list_tt" – satur datus par dokumentam piesaistītajām virtuālajām lietām;
7. "cNomenclature_list_tt" – dati par dokumentam piesaistītajiem nomenklatūras lietas numuriem un lietā nodošanas datumiem visās struktūrvienībās;
8. "cRevision_list_tt" – dati par rīkojuma vai lēmuma grozījumiem;
9. "cSubject_list_tt" – dati par dokumentam piesaistītajiem jautājumiem visās struktūrvienībās.

Detalizētu informāciju par katru kursoru var apskatīt pielikumu 7.tabulā.

3.1.2. Pagaidu tabulas datu izgūšanai no DAM

Procedūru "SELECTS", kas atlasa detalizētu informāciju par konkrētu dokumentu, vienlaikus izmantos daudzas ārējās sistēmas, kas var radīt problēmas glabājot datus parastajās datubāzes tabulās. Šīs problēmas risināšanai tiek izmantotas globālās pagaidu tabulas, kurās tiek glabāti dati par dokumentu. Galvenās atšķirības no tabulām – dati pagaidu tabulās glabājas līdz transakcijas vai sesijas beigām un ļauj piekļūt tabulām vienlaicīgi daudziem lietotājiem, jo tiek veidota atsevišķa tabula katrai sesijai, kas atrisina tabulas bloķēšanas problēmu.

3.1.3. Pagaidu tabulu apraksti

3.1.3.1. Tabula "DA_SELECTS_DIV_TEMP"

Tabulā "DA_SELECTS_DIV_TEMP" tiks glabāti dati par darbiniekiem pie kā atrodas dokuments. Apkopoto informāciju var apskatīt tabulā 3.6.

3.6. tabula **DA_SELECTS_DIV_TEMP**

Lauka nosaukums	Datu tips	Apraksts
DivisionID	number(10)	Struktūrvienības identifikators pie kā atrodas dokuments
EmployeeID	number(10)	Rezolūcijas izpildītāja slodzes ID
SendDate	date	Datums, kad nosūtīta kustība/rezolūcija
Resolution	varchar2(250)	Rezolūcijas teksts

3.1.3.2. Tabula "DA_SELECTS_DOC_TEMP"

Tabulā "DA_SELECTS_DOC_TEMP" tiek glabāti dati par dokumentam pievienotām datnēm. Precīza informācija par datiem, kas tiek glabāti pagaidu tabulā, atrodas tabulā 3.7.

3.7. tabula **DA_SELECTS_DOC_TEMP**

Lauka nosaukums	Datu tips	Apraksts
ID	number(10)	Ieraksta identifikators DAM
FileName	varchar2(200)	Faila vārds
DocName	nvarchar2(240)	Faila apraksts
RepoId	number(10)	Repozitorija identifikators
EdocFileId	number	Izpakotā edoc bērnu identifikators

Lauka nosaukums	Datu tips	Apraksts
IsSecure	number(1)	Ierobežotas pieejamības pazīme datnei (0 - nav, 1- ir)
SORT_INDEX	number(3)	Datņu kārtas numurs (LP, lēmumi, sēdes)

3.1.3.3. Tabula "DA_SELECTS_EXEC_OBJ_TEMP"

Tabulā "DA_SELECTS_EXEC_OBJ_TEMP" satur datus par dokumentam piesaistītajām virtuālajām (administratīvajām) lietām. Tiek glabāta informācija par lietas numuru, datumu un administratīvās lietas anotāciju. Lauku datu tipus var apskatīt tabulā 3.8.

3.8. tabula **DA_SELECTS_EXEC_OBJ_TEMP**

Lauka nosaukums	Datu tips	Apraksts
ID	number(10)	Administratīvās lietas identifikators
ExeObjectNr	nvarchar2(35)	Administratīvās lietas numurs
ExecObjectDate	date	Administratīvās lietas datums
ExecObjectAnnotation	nvarchar2(1000)	Administratīvās lietas anotācija

3.1.3.4. Tabula "DA_SELECTS_NOM_TEMP"

Tabulā "DA_SELECTS_NOM_TEMP" tiek glabāti dati par dokumentam piesaistītajiem nomenklatūras lietas numuriem un lietā nodošanas datumiem visās struktūrvienībās. Precīzu informāciju var apskatīt tabulā 3.9.

3.9. tabula **DA_SELECTS_NOM_TEMP**

Lauka nosaukums	Datu tips	Apraksts
ID	number(10)	Ieraksta identifikators
DivisionID	number(10)	Struktūrvienības identifikators
NomenclatureID	number(10)	Nomenklatūras numura identifikators
NomenclatureDate	date	Lietā nodošanas datums
NomenclatureCode	varchar2(15)	Nomenklatūras numurs
Year	varchar2(10)	Lietas gads

3.1.3.5. Tabula "DA_SELECTS_PERS_TEMP"

Tabulā "DA_SELECTS_PERS_TEMP" tiek glabāti dati par korespondentu (iesniegumiem) vai adresātiem (izejošiem dokumentiem), un parakstītājiem. Attiecīgu informāciju par pagaidu tabulas laukiem un datu tipiem var redzēt tabulā 3.10.

3.10. tabula **DA_SELECTS_PERS_TEMP**

Lauka nosaukums	Datu tips	Apraksts
ID	number(10)	Ieraksta identifikators DAM
PersonName	varchar2(120)	Dokumenta autors/adresāts
PersonCode	varchar2(11)	Korespondenta/adresāta personas kods vai NMRK
PersonType	number(1)	Pārskaitāmais datu tips: 1 – juridiska persona, 3 – fiziska persona, 4 - pašvaldības darbinieks
Role	number(10)	Personas loma: adresāts, korespondents, parakstītājs
PersonID	number(10)	RD VIS fiziskas vai juridiskas personas identifikators
Address	varchar2(160)	Adrese
AddressID	number(10)	RD VIS adreses identifikators
IsMain	number(1)	Galvenā persona. (1-galvenais, 0-nav) DAM galvenā persona ir tikai parakstītājiem
DivisionID	number(10)	Struktūrvienības identifikators
EmployeeID	number(10)	Darbinieka slodzes ID

3.1.3.6. Tabula "DA_SELECTS_RESP_TEMP"

Tabulā "DA_SELECTS_RESP_TEMP" atrodas informācija par atbildes dokumentiem, kas saistīti ar meklējamo dokumentu. Tabulā tiek ievietoti dati par visiem atbildes dokumentiem, kuriem dokuments norādīts laukā "Atbilde uz" Attiecīgu informāciju var apskatīt tabulā 3.11.

3.11. tabula **DA_SELECTS_RESP_TEMP**

Lauka nosaukums	Datu tips	Apraksts
ID	number(10)	Atbildes dokumenta identifikators
Nr	varchar2(35)	Atbildes dokumenta numurs
RespDate	date	Atbildes dokumenta datums
Annotation	nvarchar2(1000)	Atbildes dokumenta anotācija
Type	varchar2(1)	Vai atbildes dokuments ir starpatbilde (S) vai galīgā atbilde (G)

3.1.3.7. Tabula "DA_SELECTS_REV_TEMP"

Tabulā "DA_SELECTS_REV_TEMP" tiek glabāti dati par rīkojuma vai lēmuma grozījumiem. Tabulā tiek saņemta konkrēta informācija par dokumentiem, ko padotais dokuments groza un kādi dokumenti groza padoto dokumentu. Esošo informāciju var apskatīt tabulā 3.12.

3.12. tabula **DA_SELECTS_REV_TEMP**

Lauka nosaukums	Datu tips	Apraksts
Type	number(1)	Grozījuma tips: 0- padotais dokuments groza, 1- padoto dokumentu groza
RevisDocID	number(10)	Ar grozījumu saistītā dokumenta identifikators
RevisDocNr	nvarchar2(35)	Ar grozījumu saistītā dokumenta numurs
RevisDocDate	date	Ar grozījumu saistītā dokumenta reģistrācijas datums
RevisType	varchar2(250)	Grozījuma veids (klasificēta vērtība)
RevisDate	date	Grozījuma datums
Commentary	nvarchar2(2000)	Grozījuma paskaidrojums

3.1.3.8. Tabula "DA_SELECTS_SUBJ_TEMP"

Tabula "DA_SELECTS_SUBJ_TEMP" glabā saņemtos datus par dokumentam piesaistītajiem jautājumiem visās struktūrvienībās. Attiecīgu informāciju par tabulu var redzēt tabulā 3.13.

3.13. tabula **DA_SELECTS_SUBJ_TEMP**

Lauka nosaukums	Datu tips	Apraksts
DivisionID	number(10)	Struktūrvienības identifikators
SubjectID	number(10)	Jautājuma identifikators
SubjectName	varchar2(250)	Jautājuma nosaukums

3.1.3.9. Tabula "DA_SELECTS_SUBM_TEMP"

Tabulā "DA_SELECTS_SUBM_TEMP" tiek ievietoti dokumenta kartītes pamatdati. Tabula satur informāciju par dokumenta identifikatoru, reģistrācijas numuru, reģistrācijas datumu, dokumenta veidu un cita informācija. Attiecīgus datus var redzēt tabulā 3.9.

3.14. tabula **DA_SELECTS_SUBM_TEMP**

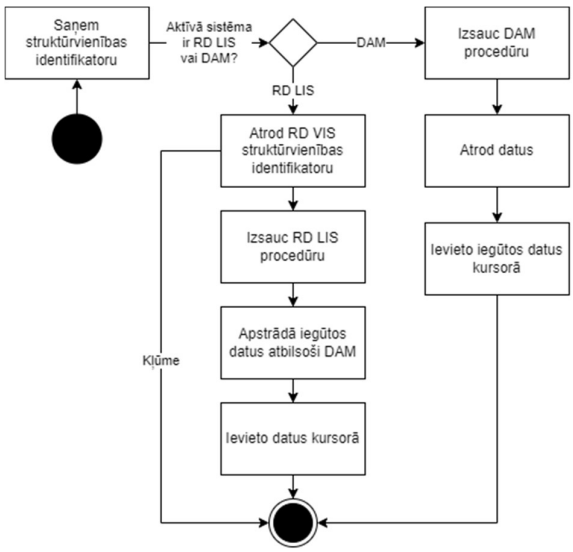
Lauka nosaukums	Datu tips	Apraksts
ID	number(10)	Dokumenta identifikators
Nr	nvarchar2(35)	Dokumenta numurs lietvedībā
DocDate	date	Dokumenta datums
Annotation	nvarchar2(1000)	Īss komentārs par dokumentu
RegDocTypeID	number(10)	Reģistrētā dokumenta veida identifikators
RegDocType	nvarchar2(100)	Reģistrētā dokumenta veida nosaukums
DivisionID	number(10)	Dokumenta struktūrvienības identifikators
ResponseDate	date	Dokumenta atbildes termiņš
NotToAnswer	number(1)	Pazīme, ka nav jāatbild (0- jāatbild, 1-nav jāatbild
Status	number(10)	Dokumenta statuss

Lauka nosaukums	Datu tips	Apraksts
Answered	date	Datums, kad atbildēts
FromDate	date	Spēkā no datums
TillDate	date	Spēkā līdz datums
OriginalNr	varchar2(50)	Ienākošā dokumenta korespondenta oriģinālā dokumenta numurs
OriginalDate	date	Ienākošā dokumenta korespondenta oriģinālā dokumenta datums
Pages	number(10)	Dokumenta identifikators

3.2. Daļējs funkciju projektējuma apraksts

3.2.1. Korespondences jautājumu saraksta iegūšana

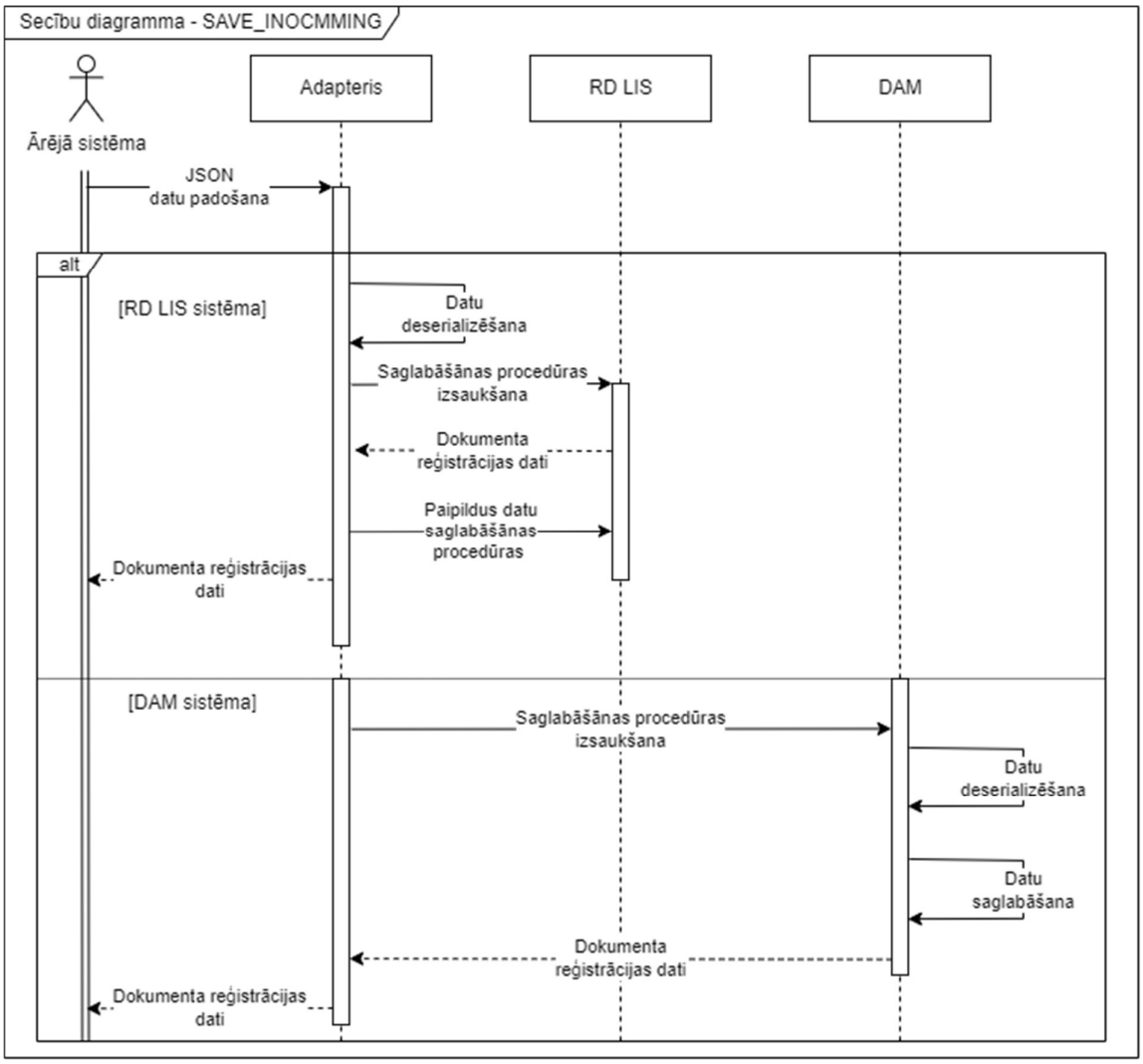
Ārējā sistēma padod dokumentu adapterim struktūrvienības identifikatoru, pēc kura tiks meklēts korespondences jautājumu saraksts. Adapteris pārbauda slēdža stāvokli, lai zinātu kuru sistēmu jāizsauc RD LIS vai DAM. Ja tiek izsaukta RD LIS sistēma, tad atrod pēc padotās DAM struktūrvienības identifikatoru atbilstošu RD LIS identifikatoru. Pēc struktūrvienības atrašanas tiek izsaukta attiecīgā RD LIS procedūra, kas atlasa informāciju un ievieto to RD LIS kursorā. Kad informācija iegūta, tiek veikta datu pārveide atbilstoši subject_list_tt kursoram. Līdzīgi darbojas DAM sistēmas izsaukums, kā var redzēt UML aktivitāšu diagrammā, kuru var apskatīt attēlā 3.1.



3.1. att. **Korespondences jautājumu saraksta iegūšana – UML aktivitāšu diagramma**

3.2.2. Ienākošo dokumentu reģistrācija

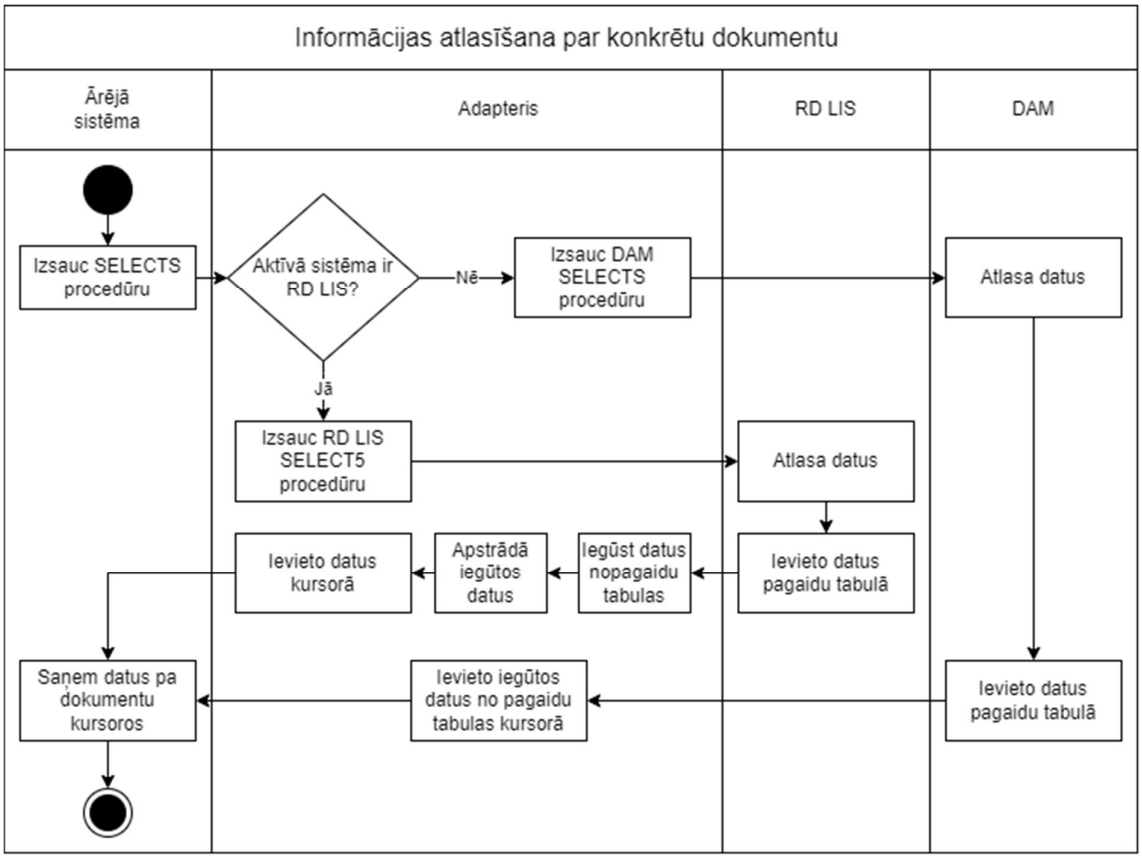
Adapteris saņem no ārējās sistēmas datus JSON formātā un pēc sesijas datiem izsauc attiecīgo zaru: RD LIS vai DAM. Vispirms tiek veikta pārbaude uz JSON struktūras validāciju, pēc kuras izgūst datus dokumenta reģistrācijai. Izsaucot RD LIS procedūru, atrod darbinieka datus pēc viņa identifikatora, kā arī tiek iegūta informācija par saistītiem dokumentiem, jā tāda informācija tika padota. Pēc visu datu iegūšanas izsauc attiecīgās procedūras dokumenta reģistrācijai. Beigās tiek izveidoti JSON formāta dati, kas satur informāciju par ienākošā dokumenta identifikatoru, reģistrācijas numuru un datumu. Izsaucot DAM sistēmu darbības ir līdzīgas, bet JSON deserializēšana notiek jau DAM pusē un tiek izsaukta vajadzīgā procedūra dokumenta reģistrācijai, pēc kuras izpildes tiek iegūti dati par dokumenta identifikatoru, reģistrācijas numuru un datumu, kurus DAM padod atpakaļ adapterim. Ienākošo dokumentu reģistrācijas diagrammu var apskatīt attēlā 3.2.



3.2. att. **Ienākošo dokumentu reģistrācija – secību diagramma**

3.2.3. Informācijas atlasīšana par konkrētu dokumentu

Informācijas atlasīšana RD LIS un DAM notiek līdzīgi: dokumentu adapteris saņem datus par dokumentu, par kuru nepieciešams atrast informāciju, un izsauc attiecīgo RD LIS vai DAM procedūru. Procedūrā "SELECTS" un "SELECT5" tiek atrasti dati no dokumenta pamatkartītes, par saistītām personām, atbildes dokumentiem, struktūrvienībām, virtuālajām (administratīvajām) lietām, nomenklatūrām, dati par rīkojuma vai grozījuma lēmumiem un par piesaistītajiem jautājumiem. Šie dati tiek atrasti un ievietoti globālajās pagaidu tabulās, pie kurām pēc procedūru izpildes pieslēdzas adapteris un ievieto datus attiecīgajos kursoros, lai ārējā sistēma varētu iegūt vajadzīgos sev datus. Diagrammu par informācijas atlasīšanu par konkrētu dokumentu var apskatīt attēlā 3.3.



3.3. att. **Informācijas atlasīšana par konkrētu dokumentu – peldceliņu diagramma**

4. TESTĒŠANAS DOKUMENTĀCIJA

4.1. Testēšanas apraksts

Projekta ietvaros adaptera izstrādes laikā tika rakstīti vienību testi, kas nodrošināja dokumenta adaptera darbību pārbaudi. Vienību testu izstrāde palīdzēja izstrādes laikā atrast kļūdas izstrādātajā kodā. Vienību testiem tika izveidota atsevišķa datubāzes pakotne, kur atrodas visi testpiemēri, kurus var palaist vai nu atsevišķi, vai nu visus pēc kārtas. Pēc izmaiņu pieprasījumiem un koda rediģēšanas vienību testi tika laisti atkārtoti, lai pārbaudītu izmaiņas kodā.

4.2. Manuālā testēšana

Adaptera izstrādes laikā izstrādātajiem bija izveidotas atsevišķas datubāzes pakotnes, kur tika izstrādātas procedūras. Izstrādes laikā tika aktīvi izmantoti iebūvētie programmas PL/SQL Developer testēšanas rīki, kas palīdzēja atrast kļūdas kodā un veikt atkļūdošanu.

4.3. Vienībtestēšana

Adapteris strādā ar divām sistēmām RD LIS un DAM, tāpēc nepieciešams pārbaudīt izstrādāto procedūru darbību abās sistēmās. Adaptera procedūras tika izstrādātas ar slēdža stāvokļa noteikšanu, kas deva iespēju izstrādāt vienu vienību testu, kuram var padot dažādas slēdža stāvokļa vērtības. Šī pieeja atvieglo vienību testu rakstīšanu un padara kodu labāk lasāmu, kas ietekmē adaptera kvalitāti.

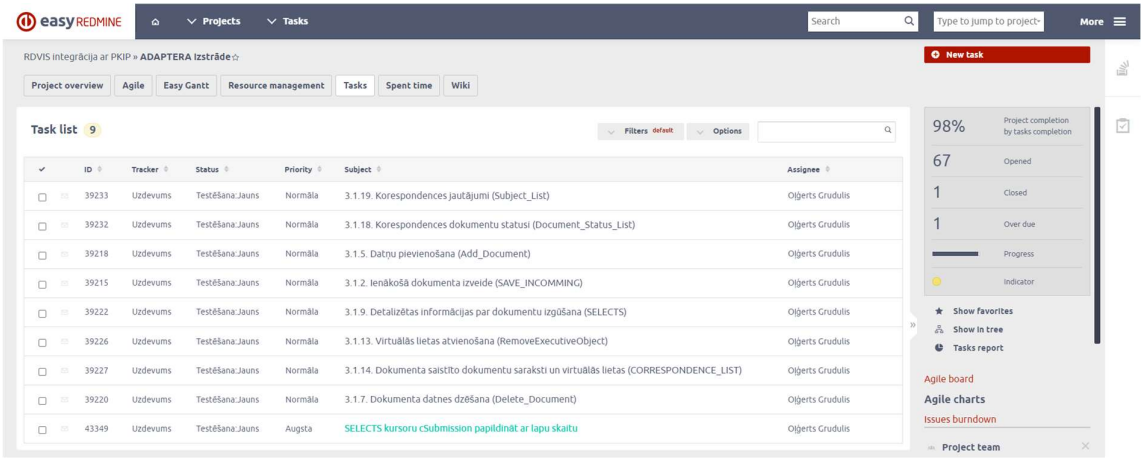
Visi vienību testi tika izstrādāti pēc pieejas, ka tiek veidots viens tests, kur pēc padotiem parametriem tiek noteikts slēdža stāvoklis: 0 – izmantot RD LIS sistēmu, 1 – izmantot DAM sistēmu. Vienību testa rezultātā tiek atgriezta būla tipa vērtība – true vai false. Pēc veiksmīgas testa izpildes tiek attēlota informācija, kura tiek meklēta vai izvada paziņojumu par veiksmīgu pievienošanu vai dzēšanu. Kļūdas gadījumā tiek attēlota informācija, kur atrodas kļūda un kādi iemesli. Katrai uzrakstītai procedūrai tika veidots viens vienību tests un vismaz divas funkcijas, kas izsauc doto testu, pārbaudei katrā sistēmā. Vienību testu piemērus var apskatīt pielikuma sadaļā.

5. PROJEKTA ORGANIZĀCIJA

Adaptera izstrāde ir PKIP projekta daļa, kurš tiek izstrādāts pēc hibrīdās izstrādes pieejas: ūdenskrituma un spējās izstrādes pieejām, jo lielākā daļa prasību tika definēta dokumentācijā iepriekš, bet daļa prasību tika definēta adaptera izstrādes laikā, kā arī pēc kolēģu pieprasījumiem no citām komandām, kas izstrādā ārējās sistēmas. Projekta izstrādes laikā notika komandas iknedēļas sapulces, kur tika apspriests padarītais darbs, plāni uz tālāko darbu un uzdoti jautājumi, ja ir kādas neskaidrības.

5.1. Projekta pārvaldība

Projekta izstrādes uzdevumi reģistrēja izmantojot projektu pārvaldības rīku Redmine, ko var redzēt attēlā 5.1., kur tika ievietota informācija par izstrādi un veiktas piezīmes projekta vadītājiem par neatbilstībām sākotnējā PPS.



5.1. att. **Projekta uzdevumu reģistrēšanas vietne**

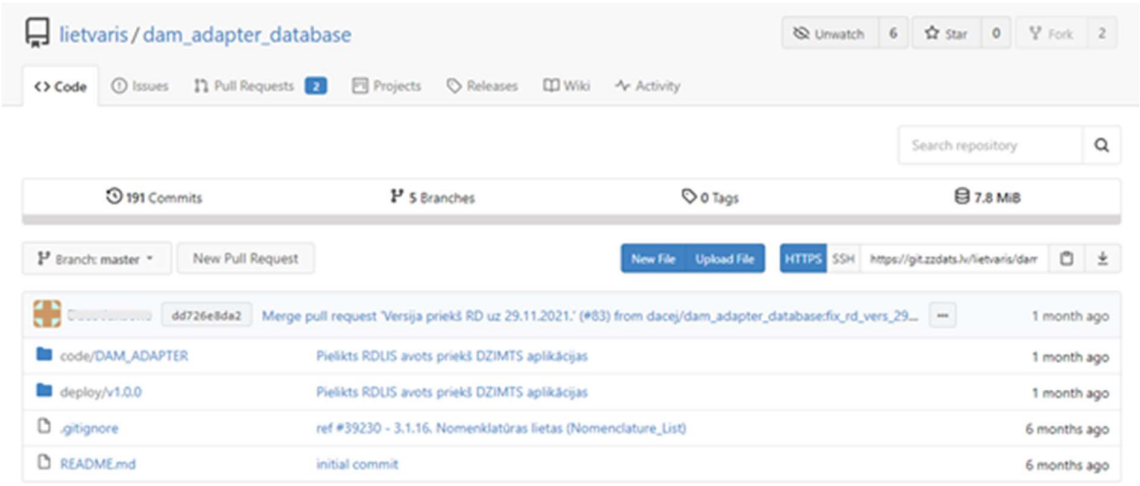
DAM adaptera izstrādē piedalījās daudzas uzņēmuma komandas. No DVS "Lietvaris" komandas, kas ir atbildīgi par DAM adaptera izstrādi, bija iesaistīti vecākais programmētājs, programmētājs, projekta pārvaldnieks un divi projekta vadītāji.

Pirms tika sākta adaptera izstrāde bija izveidota dokumentācija, pēc kuras programmētāji izstrādāja produktu. Izstrādes laikā programmētāji piefiksēja pie uzdevumiem projekta pārvaldības rīkā Redmine atšķirības vai neatbilstības izveidotajā dokumentācijā, kas tika labotas.

Adaptera izstrādes laikā tika izmantota Oracle PL/SQL valoda un koda izstrādei izmantoti rīki bija PL/SQL Developer 14 un Notepad++.

5.2. Konfigurācijas pārvaldība

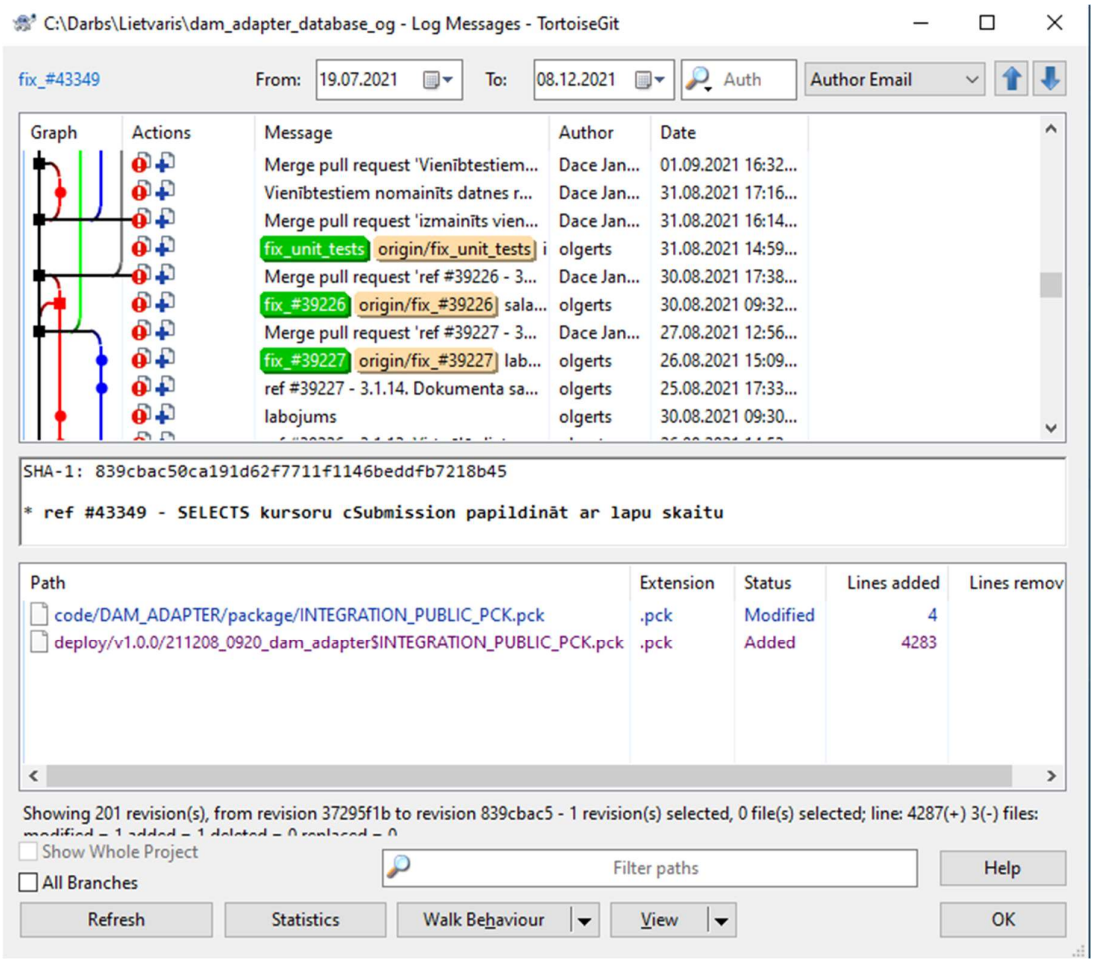
Projekta konfigurācijas pārvaldībai bija izmantots versiju kontroles rīks Git un SIA "ZZ Dats" serveris, kur atrodas vairāki uzņēmuma projekti. DAM adaptera projektam tika izmantoti divi repozitoriji: 'lietvaris/dam_adapter_database', kas satur adaptera datubāzes programmkodu un redzams attēlā 5.2., un "lietvaris/database", kas satur DVS "Lietvaris" datubāzes programmkodu.



5.2. att. **Uzņēmuma Git serveris – repozitorijs "lietvaris/dam_adapter_database"**

Izstrādes laikā darba autors izmantoja papildus divus rīkus konfigurāciju pārvaldībai – Git (attēls 5.3.) un TortoiseGit (attēls 5.4), kas nodrošināja repozitorija klonēšanu uz darbstacijas un izmaiņu ielādēšanu repozitorijā. Ar GUI rīku TortoiseGit tika nodrošināta iespēja pārvaldīt tikai savas izmaiņas datubāzes programmkodā, jo adaptera izstrāde notika vienota datubāzē.

5.3. att. **Git komandrindas saskarne**



5.4. att. **TortoiseGit grafiskā lietotāja saskarne**

Pēc katras procedūras izstrādes daba autors augšupielādēja programmkodu uz uzņēmuma Git serveri attiecīgajā repozitorijā, kur vecākais programmētais veica koda apskatu un vai nu apstiprināja izmaiņas, vai nu noraidīja un pieprasīja izmaiņas.

5.3. Kvalitātes nodrošināšana

Nodrošināt kvalitatīvu projekta izstrādi veicināja šādas darbības:

1. Pirmkods tika izstrādāts atbilstoši uzņēmuma vadlīnijām [6];
2. Tika izmantots konfigurācijas pārvaldes rīks Git, kas nodrošināja kvalitatīvu koda izstrādi, apskatot to vecākajam programmētājam pirms ielādēšanas Git repozitorijā;
3. Ir izveidoti vienību testi, kas nodrošina procedūru un funkciju darbības pārbaudi;
4. Pirmkods ir komentēts un paskaidro koda darbību;
5. Dokumentācijas rakstīšanas laikā tika izmantoti Akadēmiskās terminu datubāzes svešvārdu tulkojumi [7].

5.4. Darbietilpības novērtējums

Darbietilpības novērtējums tika veikts aprunājoties ar pieredzējušo programmētāju un atbilstoši uzstādītajiem paredzamajiem laikiem, ko uzlika projekta vadītāji. Novērtējums sadalīts dažās daļās: projekta iepazīšana, dokumentācijas izstrāde un adaptera izstrāde.

5.1. tabula **Darbietilpības novērtējuma tabula**

Darbs	**Darbietilpības aprēķins stundās**		
	Pesimistiskais	**Reālistiskais**	**Optimistiskais**
Iepazīšanās ar projektu	50	40	20
Iepazīšanās ar RD LIS un DAM sistēmām	40	20	10
Dokumentācijas izstrāde			
PPS	60	40	20
PPA	80	60	40
Testēšana	60	30	10
Adaptera izstrāde			
SAVE_INCOMMING	60	32	20
ADD_DOCUMENT	45	30	20
DELETE_DOCUMENT	30	20	10
SELECT	130	112	70
RemoveExecutiveObject	45	26	15
CORRESPONDENCE_LIST	100	76	45
DOCUMENT_STATUS_LIST	20	6	3
SUBJECT_LIST	20	6	3
Kopā	736	498	286

Darbietilpība tika aprēķināta pēc formulas: Darbietilpība = (Pesimistiskais + 4* Reālistiskais + Optimistiskais)/ 6. Aprēķinātā darbietilpība sanāca ap 502 stundām, kas ir 3.1 personmēnesis. Reāli patērētais laiks iznāca ap 430 stundām, kas atbilst 2.7 personmēnešiem.

REZULTĀTI

Kvalifikācijas darba rezultātā tika izveidots dokumentu adapteris, kas dod iespēju ārējām sistēmām vienkāršāk pāriet no vecās Rīgas domes lietvedības sistēmas uz jaunu. Izveidotā programmatūra ļauj saņemt informāciju par jau reģistrētiem dokumentiem RD LIS un DAM sistēmās, kā arī veidot jaunus dokumentus ārējām sistēmām un reģistrēt tos lietvedības sistēmās.

SECINĀJUMI

Kvalifikācijas darba izstrādes laika tika iegūtas un attīstītas daudzas zināšanas. Tika apgūta Oracle datubāzes sistēma, programmēšanas valoda Oracle PL/SQL, attīstītas prasmes konfigurāciju pārvaldē un komandu darbā. Iepazīšanās ar lielo projektu deva iespēju vairāk uzzināt par biznesa procesu un spējās izstrādes metodi.

IZMANTOTĀ LITERATŪRA UN AVOTI

[1] Latvijas Valsts Standarts LVS 68:1996 "Programmatūras prasību specifikācijas ceļvedis."

[2] Latvijas Universitātes rīkojums Nr. 1/38 "Prasības noslēguma darbu (bakalaura, maģistra darbu, diplomdarbu un kvalifikācijas darbu) izstrādāšanai un aizstāvēšanai Latvijas Universitātē."

[3] SIA "ZZ Dats" programmatūras prasību specifikācija "PKIP integrācija ar RD VIS lietojumprogrammas KAVIS apakšsistēmām. PPS v.1.4"

[4] SIA "ZZ Dats", AS "Emergn" dokumentācija "RD LIS un RD VIS integrācija"

[5] Oracle PL/SQL dokumentācija [tiešsaiste] – [atsauce 07.01.2022.]

Pieejams: https://docs.oracle.com/

[6] Oracle programmētāja rokasgrāmata [tiešsaiste] – [atsauce 07.01.2022.]

Pieejams autentificējoties: https://wiki.zzdats.lv/wiki/25/oracle-programmētāja-rokasgrāmata-papildināts-ar-pareizu-sekvenču-veidošanu

[7] Akadēmiskā terminu bāze [tiešsaiste] – [atsauce 07.01.2022.]

Pieejams: https://termini.gov.lv/

PIELIKUMI

Atbilstība starp Adaptera – RD LIS – DAM procedūrām

Pielikuma 1. tabula **Atbilstību tabula RD LIS un DAM procedūrām**

Darbība	**RD LIS procedūra**	**Adaptera procedūra, kas jāizsauc ārējai sistēmai**	**Adapteris izsauc RD LIS vai KAVIS procedūru**	**Adapteris izsauc DAM procedūru**
Ienākošā dokumenta reģistrācija	SAVE, SAVE2, SAVE3, SAVE4	SAVE_INCOMMING	Sadala atsevišķās procedūrās: SAVE4 Add_Person2 Add_Document2	SAVE_INCOMMING
Datņu pievienošana	Add_Document, Add_Document2	Add_Document	Add_Document	Add_Document
Datņu dzēšana	Delete_Document	Delete_Document	Delete_Document	Delete_Document
Administratīvās/ piedziņas lietas atvienošana	RemoveExecutiveObject, RemoveExecutiveObject2	RemoveExecutiveObject,	RemoveExecutiveObject2	RemoveExecutiveObject,
Detalizētas informācija par dokumentu izgūšana	SELECTS, SELECT2, SELECT_REZ, SELECT3, SELECT4, SELECT5	SELECTS	SELECTS5	SELECTS

Darbība	RD LIS procedūra	Adaptera procedūra, kas jāizsauc ārējai sistēmai	Adapteris izsauc RD LIS vai KAVIS procedūru	Adapteris izsauc DAM procedūru
Informācija par sarakstēm un administratīvām lietām	CORRESPONDENCE_LIST	CORRESPONDENCE_LIST	CORRESPONDENCE_LIST	CORRESPONDENCE_LIST
Korespondences jautājumi	Subject_List	Subject_List	Subject_List	Subject_List
Korespondences dokumentu statusi	Document_Status_List	Document_Status_List	Document_Status_List	Document_Status_List

Ievaddatu un izvaddatu atbilstības tabulas Adaptera procedūrām

Pielikuma 2. tabula **Ievaddatu atbilstību tabula SAVE_INCOMMING procedūrai**

RD LIS				**DAM**				
RD LIS nosaukums	**Garums**	**Tips**	**Obligāts**	**DAM nosaukums**	**Garums**	**Tips**	**Obligāts**	**Apraksts**
ExternalID	10	NUMBER	jā	ExternalID	10	NUMBER	jā	Iesnieguma identifikators ārējā sistēmā
SourceID	10	NUMBER	jā	AppID	10	NUMBER	jā	Reģistrējamā dokumenta avots
DocTypeID	10	NUMBER	jā	DocTypeCode	30	VARCHAR2	jā	Ārējās sistēmas dokumenta veida identifikators
Annotation	1000	VARCHAR2	jā	Annotation	2000	VARCHAR2	jā	Dokumenta anotācija (nosaukums, īss komentārs)
OwnerDivisionID	10	NUMBER	jā	OwnerDivisionID	10	NUMBER	jā	Dokumenta saimnieka DAM institūcijas identifikators – nosaka reģistrācijas Nr.
OfficDivisionID	10	NUMBER	jā	Offic		string	jā	Masīvs. Ienākošā dokumenta pašvaldības adresāts, var būt vairāki adresāti Padod struktūru (OfficDivisionID), darbinieka slodzes ID (OfficeEmployeeID)

RD LIS				DAM				
RD LIS nosaukums	**Garums**	**Tips**	**Obligāts**	**DAM nosaukums**	**Garums**	**Tips**	**Obligāts**	**Apraksts**
OfficEmployeeName	50	VARCHAR2	nē	OfficeEmployeeID	10	NUMBER	jā	Ienākošā dokumenta adresāts slodzes ID
RespDivisionID	10	NUMBER	nē	RespDivisionID	10	NUMBER	nē	Ienākošā dokumenta pašvaldības darbinieks
RespEmployeeName	50	VARCHAR2	nē	RespEmployeeID	10	NUMBER	nē	Ienākošā dokumenta pašvaldības darbinieka slodzes ID
RegDivisionID	10	NUMBER	daļēji	RegDivisionID	10	NUMBER	daļēji	Dokumenta reģistrētāja struktūrvienība
RegEmployeeName	50	VARCHAR2	nē	RegEmployeeID	10	NUMBER	nē	Dokumenta reģistrētāja slodzes ID
SubjectID	10	NUMBER	nē	SubjectID	10	NUMBER	nē	Jautājuma identifikators
NomenclatureID	10	NUMBER	nē	NomenclatureID	10	NUMBER	nē	Reģistrētāja struktūrvienības lietu nomenklatūras identifikators
Pages	10	NUMBER	jā	Pages	10	NUMBER	nē	Lapu skaits
RelatedDocID	10	NUMBER	nē	RelatedDoc		STRING	nē	Saistīto dokumentu masīvs, padod dokumenta ID, numuru un tipu (0-saistītais dokuments, 1-atbilde uz)
ReplayToDocID	10	NUMBER	nē					
ResponseDate		DATE	nē	ResponseDate		DATE	nē	Dokumenta atbildes termiņš

RD LIS				DAM				
RD LIS nosaukums	**Garums**	**Tips**	**Obligāts**	**DAM nosaukums**	**Garums**	**Tips**	**Obligāts**	**Apraksts**
NotToAnswer	1	NUMBER	nē	NotToAnswer	1	NUMBER	nē	Pazīme, ka nav jāatbild (0 – jāatbild, 1- nav jāatbild)
Status	10	NUMBER	nē	StatusCode	7	VARCHAR2	nē	DAM dokumenta statusa kods (Atrod ar metodi document_statuss_list)
				SubmissionDate		DATE	nē	Korespondenta dokumenta datums
				SubmissionNr	22	VARCHAR2	nē	Korespondenta dokumenta numurs
				Externallinks		URL		Saite uz ārējās sistēmas dokumentu, ko var izmantot, lai atvērtu ārējo sistēmu
				Correspondents			jā	Masīvs. Korespondentu saraksts, kas atbilst AddPersons struktūrai
				Files			nē	Pievienoto datņu saraksts. Masīvs ar struktūru (Galvenais (Primary), repozitorija ID (RepositoryID), IPI (IsSecure))

Pielikuma 3. tabula **Izvaddatu atbilstību tabula SAVE_INCOMMING procedūrai**

RD LIS			**DAM**			
RD LIS nosaukums	**Garums**	**Tips**	**DAM nosaukums**	**Garums**	**Tips**	**Apraksts**
ID	10	Number	DocID	10	NUMBER	Izveidotā dokumenta identifikators
Nr	22	VARCHAR2	DocNr	22	VARCHAR2	Dokumenta reģistrācijas numurs
RegDate		DATE	RegDate		DATE	Dokumenta reģistrēšanas datums

Pielikuma 4. tabula **Ievaddatu atbilstību tabula ADD_DOCUMENT procedūrai**

RD LIS				**DAM**				
RD LIS nosaukums	**Garums**	**Tips**	**Obligāts**	**DAM Nosaukums**	**Garums**	**Tips**	**Obligāts**	**Apraksts**
Submission_ID	10	NUMBER	jā	SubmissionID	10	NUMBER	jā	Dokumenta ID. Atrod ar LIST, ja izsaukums nenotiek vienlaicīgi ar reģistrāciju.
Division_ID	10	NUMBER	jā	EmployeeDivision_ID	10	NUMBER	jā	Reģistrētāja struktūrvienība
Employee_Name	52	VARCHAR2	jā	EmployeeID	10	NUMBER	jā	Reģistrētāja slodzes ID
Is_Secure		NUMBER	jā	Is_Secure		NUMBER	jā	IPI (0-nav, 1-IPI)

RD LIS				DAM				
RD LIS nosaukums	**Garums**	**Tips**	**Obligāts**	**DAM Nosaukums**	**Garums**	**Tips**	**Obligāts**	**Apraksts**
Edoc_Id	10	NUMBER	nē	Repo_Id	10	NUMBER	nē	Repozitorija ID
Doc_Type_ID	10	NUMBER	jā	Doc_Type_ID	10	NUMBER	jā	Dokumenta veida identifikators

Pielikuma 5. tabula **Ievaddatu atbilstību tabula DELETE_DOCUMENT procedūrai**

RD LIS				DAM				
RD LIS nosaukums	**Garums**	**Tips**	**Obligāts**	**DAM nosaukums**	**Garums**	**Tips**	**Obligāts**	**Apraksts**
SubmissionID	10	NUMBER	daļēji	SubmissionID	10	NUMBER	jā	Dokumenta ieraksta ID
DocTypeID	10	NUMBER	daļēji	DocTypeCode	30	VARCHAR2	daļēji	Ārējās sistēmas dokumenta veida ID
FileID	10	NUMBER	daļēji	RepoID	10	NUMBER		Datnes repozitorija ID
FileName	240	VARCHAR2	daļēji	FileName	240	VARCHAR2		Datnes vārds

Pielikuma 6. tabula **Ievaddatu atbilstību tabula SELECTS procedūrai**

RD LIS				**DAM**				
RD LIS nosaukums	**Garums**	**Tips**	**Obligāts**	**DAM nosaukums**	**Garums**	**Tips**	**Obligāts**	**Apraksts**
ID	10	NUMBER	1/2	ID	10	NUMBER	jā	Identifikators (atrod ar LIST)
Nr	22	VARCHAR2	1/2	Nr	35	VARCHAR2	jā	Dokumenta numurs lietvedībā
ReturnBinDoc	1	NUMBER	jā	ReturnBinDoc	1	NUMBER	jā	Pazīme vai dokuments tiek atgriezts ar informāciju par repozitorija dokumentiem vai bez (0 – bez, 1 – ar pievienotajiem dokumentiem, kam pieejamība nav ierobežota vai lietotājam ir tiesības atvērt ierobežotās pieejamības dokumentu, 2 – visi pievienotie dokumenti, neanalizējot IPI)
UserPK	11	VARCHAR2	jā	EmployeeID	10	NUMBER	jā	Procedūru inicializējušā lietotāja slodzes ID
UserDivisionID	10	NUMBER	jā	EmployeeDivisionID	10	NUMBER	jā	Procedūru inicializējušā lietotāja struktūrvienības identifikators

Pielikuma 7. tabula **Izvaddatu atbilstību tabula SELECTS procedūrai**

RD LIS			**DAM**			
RD LIS nosaukums	**Garums**	**Tips**	**DAM nosaukums**	**Garums**	**Tips**	**Apraksts**
cSubmission – Dokumenta kartītes pamatdati						
ID	10	NUMBER	ID	10	NUMBER	Dokumenta identifikators
Nr	22	VARCHAR2	Nr	35	NVARCHAR2	Dokumenta numurs lietvedībā
DocDate		DATE	DocDate		DATE	Dokumenta datums
Annotation	1000	VARCHAR2	Annotation	1000	NVARCHAR2	Īss komentārs par dokumentu
RegDocTypeID	10	NUMBER	RegDocTypeID	10	NUMBER	Reģistrētā dokumenta veida identifikators
RegDocType	50	VARCHAR2	RegDocType	100	NVARCHAR2	Reģistrētā dokumenta veida nosaukums
DivisionID	10	NUMBER	DivisionID	10	NUMBER	Dokumenta struktūrvienības identifikators
ResponseDate		DATE	ResponseDate		DATE	Dokumenta atbildes termiņš
NotToAnswer	1	NUMBER	NotToAnswer	1	NUMBER	Pazīme, ka nav jāatbild (0 – jāatbild, 1 – nav jāatbild)
Status	10	NUMBER	Status	10	NUMBER	Dokumenta statuss. Atgriež statusa id, kur vērtību atrod no procedūras DOCUMENT_STATUS_LIST
Answered		DATE	Answered		DATE	Datums, kad atbildēts
FromDate		DATE	FromDate		DATE	Spēkā no datums

TillDate		DATE	TillDate		DATE	Spēkā līdz datums
			OriginalNr	50	VARCHAR2	Ienākošā dokumenta korespondenta oriģinālā dokumenta numurs
			OriginalDate		DATE	Ienākošā dokumenta korespondenta oriģinālā dokumenta datums
cDocument – Dokumentam pievienotās datnes. Padod datus, ja ieejas parametrs ReturnBinDoc ir 1 vai 2						
ID	10	NUMBER	ID	10	NUMBER	Ieraksta identifikators DAM (Document_files.id)
FileName	50	VARCHAR2	FileName	50	VARCHAR2	Faila vārds
DocName	240	VARCHAR2	DocName	240	NVARCHAR2	Faila apraksts
Edocid	10	NUMBER	Repoid	10	NUMBER	Repozitorija id
EdocFileId		NUMBER	EdocFileId		NUMBER	Izpakotā edoc bērnu id
IsSecure	1	NUMBER	IsSecure	1	NUMBER	Ierobežotas pieejamības pazīme datnei (0 – nav, 1 – ir)
KartasNr	3	NUMBER	SORT_INDEX	3	NUMBER	Datņu kārtas numurs (LP, lēmumi, sēdes)
DocBDate		DATE	DocBDate		DATE	Datnes pievienošanas vai pēdējais labošanas datums
cPerson – Dati par korespondentu (iesniegumiem) vai adresātu (izejošiem dokumentiem), un parakstītājiem						
ID	10	NUMBER	ID	10	NUMBER	Ieraksta identifikators DAM Document_persons.id
PersonName	120	VARCHAR2	PersonName	120	VARCHAR2	Dokumenta autors/adresāts

PersonCode	11	VARCHAR2	PersonCode	11	VARCHAR2	Korespondenta/adresāta personas kods vai NMRK
PersonType	1	NUMBER	PersonType	1	NUMBER	Pārskaitāmais datu tips: 1 – juridiska persona, 3 – fiziska persona, 4 – pašvaldības darbinieks
			Role	10	Number	Personas loma: adresāts, korespondents, parakstītājs
PersonID	10	NUMBER	PersonID	10	NUMBER	RD VIS fiziskas vai juridiskas personas identifikators
Address	160	VARCHAR2	Address	160	VARCHAR2	Adrese
AddressID	10	NUMBER	AddressID	10	NUMBER	RD VIS adreses identifikators
IsMain	1	NUMBER	IsMain	1	NUMBER	Galvenā persona. (1 – galvenais, 0 – nav) DAM galvenā persona ir tikai parakstītājiem
DivisionID	10	NUMBER	DivisionID	10	NUMBER	Struktūrvienības identifikators
JobTitle	50	VARCHAR2	EmployeeID	10	Number	Darbinieka slodzes ID
cResponse – Ar dokumentu saistītie atbildes dokumenti (dokumenta numurs norādīts laukā „Atbilde uz")						
ID	10	NUMBER	ID	10	NUMBER	Atbildes dokumenta identifikators
Nr	22	VARCHAR2	Nr	35	VARCHAR2	Atbildes dokumenta numurs
RespDate		DATE	RespDate		DATE	Atbildes dokumenta datums
Annotation	1000	VARCHAR2	Annotation	1000	NVARCHAR2	Atbildes dokumenta anotācija

Type	1	VARCHAR2	Type	1	VARCHAR2	Vai atbildes dokuments ir starpatbilde (S) vai galīgā atbilde (G)
cDivision – Dati par darbiniekiem pie kā atrodas dokuments. Rezolūcijas izpildītājs, kura rezolūcijai nav izpildes pazīme un nav izveidota pakārtotā rezolūcija						
DivisionID	10	NUMBER	DivisionID	10	NUMBER	Struktūrvienības identifikators pie kā atrodas dokuments
JobTitle	50	VARCHAR2	EmployeeID	10	Number	Rezolūcijas izpildītāja slodzes ID
SendDate		DATE	SendDate		DATE	Datums, kad nosūtīta kustība/rezolūcija
Resolution	250	VARCHAR2	Resolution	250	VARCHAR2	Rezolūcijas teksts
cExecObject – Satur datus par dokumentam piesaistītajām virtuālajām (administratīvajām) lietām						
ID	10	NUMBER	ID	10	NUMBER	Administratīvās lietas identifikators
ExecObjectNr	22	VARCHAR2	ExecObjectNr	35	NVARCHAR2	Administratīvās lietas numurs
ExecObjectDate		DATE	ExecObjectDate		DATE	Administratīvās lietas datums
ExecObjectAnnotation	1000	VARCHAR2	ExecObjectAnnotation	1000	NVARCHAR2	Administratīvās lietas anotācija
cNomenclature – Dati par dokumentam piesaistītajiem nomenklatūras lietas numuriem un lietā nodošanas datumiem visās struktūrvienībās						
ID	10	NUMBER	ID	10	NUMBER	Ieraksta identifikators
DivisionID	10	NUMBER	DivisionID	10	NUMBER	Struktūrvienības identifikators
NomenclatureID	10	NUMBER	NomenclatureID	10	NUMBER	Nomenklatūras numura identifikators
NomenclatureDate		DATE	NomenclatureDate		DATE	Lietā nodošanas datums

NomenclatureCode	15	NUMBER	NomenclatureCode	15	NUMBER	Nomenklatūras numurs
Year	4	NUMBER	Year	4	NUMBER	Lietas gads.
cRevision – Dati par rīkojuma vai lēmuma grozījumiem, konkrēti dati par dokumentiem, ko padotais dokuments groza un kādi dokumenti groza padoto dokumentu						
SubmissionID	10	NUMBER				Padotā dokumenta identifikators
SubmissionNr	22	VARCHAR2				Padotā dokumenta numurs
Type	1	NUMBER	Type	1	NUMBER	Grozījuma tips: 0 – padotais dokuments groza, 1 – padoto dokumentu groza
RevisDocID	10	NUMBER	RevisDocID	10	NUMBER	Ar grozījumu saistītā dokumenta identifikators
RevisDocNr	22	VARCHAR2	RevisDocNr	35	NVARCHAR2	Ar grozījumu saistītā dokumenta numurs
RevisDocDate		DATE	RevisDocDate		DATE	Ar grozījumu saistītā dokumenta reģistrācijas datums
RevisType	250	VARCHAR2	RevisType	250	VARCHAR2	Grozījuma veids (klasificēta vērtība)
RevisDate		DATE	RevisDate		DATE	Grozījuma datums
Commentary	2000	VARCHAR2	Commentary	2000	NVARCHAR2	Grozījuma paskaidrojums
cSubject – Dati par dokumentam piesaistītajiem jautājumiem visās struktūrvienībās						
DivisionID	10	NUMBER	DivisionID	10	NUMBER	Struktūrvienības identifikators
SubjectID	10	NUMBER	SubjectID	10	NUMBER	Jautājuma identifikators
SubjectName	250	VARCHAR2	SubjectName	50	VARCHAR2	Jautājuma nosaukums

Pielikuma 8. tabula **Ievaddatu atbilstību tabula RemoveExecutiveObject procedūrai**

RD LIS				**DAM**				
RD LIS nosaukums	**Garums**	**Tips**	**Obligāts**	**DAM nosaukums**	**Garums**	**Tips**	**Obligāts**	**Apraksts**
Submission_ID	10	NUMBER		SubmissionID	10	NUMBER	jā	Dokumenta identifikators (vienmēr atrod ar LIST)
ExecobjectID	10	NUMBER	jā	ExecObjectID	10	NUMBER	jā	Administratīvās lietas id, kam piesaistīts dokuments
SubmissionDocTypeId	10	NUMBER	3/4	SubmissionDocTypeId	10	NUMBER		Dokumenta veida ID

Pielikuma 9. tabula **Ievaddatu atbilstību tabula CORRESPONDENCE_LIST procedūrai**

RD LIS				**DAM**				
RD LIS nosaukums	**Garums**	**Tips**	**Obligāts**	**DAM nosaukums**	**Garums**	**Tips**	**Obligāts**	**Apraksts**
ID	10	NUMBER	Daļēji	ID	10	NUMBER	jā	DAM dokumenta identifikators
Nr	22	VARCHAR2	Daļēji	Nr	35	NVARCHAR2	nē	DAM dokumenta reģistrācijas Nr.
RegDocTypeID	10	NUMBER	Daļēji	RegDocTypeID	10	NUMBER	nē	DAM dokumenta veida identifikators
DivisionID	10	NUMBER	Daļēji	DivisionID	10	NUMBER	jā	Saraksti identificējošās struktūrvienības identifikators

RD LIS				DAM				
RD LIS nosaukums	**Garums**	**Tips**	**Obligāts**	**DAM nosaukums**	**Garums**	**Tips**	**Obligāts**	**Apraksts**
ExecObjectID	10	NUMBER	Daļēji	ExecObjectID	10	NUMBER	nē	DAM administratīvās lietas identifikators
ExecObjectNr	22	VARCHAR2	Daļēji	ExecObjectNr	35	VARCHAR2	nē	Administratīvās lietas reģistrācijas Nr.

Pielikuma 10. tabula **Izvaddatu atbilstību tabula CORRESPONDENCE_LIST procedūrai**

RD LIS			DAM			
RD LIS nosaukums	**Garums**	**Tips**	**DAM nosaukums**	**Garums**	**Tips**	**Apraksts**
cDoc – Dati par sarakstes dokumentiem						
SubmissionID	10	NUMBER	SubmissionID	10	NUMBER	Dokumenta identifikators
SubmissionDocTypeID	10	NUMBER	SubmissionDocTypeID	10	NUMBER	Dokumenta veida identifikators
SubmissionDocType	100	VARCHAR2	SubmissionDocType	100	VARCHAR2	Dokumenta veida nosaukums
SubmissionDocNr	22	VARCHAR2	SubmissionDocNr	35	NVARCHAR2	Dokumenta reģistrācijas Nr.
SubmissionDocDate		DATE	SubmissionDocDate		DATE	Dokumenta reģistrācijas datums
SubmissionDocAnnotation	1000	VARCHAR2	SubmissionDocAnnotation	2000	NVARCHAR2	Dokumenta anotācija
SubmissionDocCommentary	1000	VARCHAR2	SubmissionDocCommentary	2000	NVARCHAR2	Dokumenta piezīmes

RD LIS			DAM			
RD LIS nosaukums	**Garums**	**Tips**	**DAM nosaukums**	**Garums**	**Tips**	**Apraksts**
ExecObjects	500	VARCHAR2	ExecObjects	500	VARCHAR2	Masīvs Piesaistīto administratīvo lietu saraksts (reģistrācijas Nr. atdalīti ar komatu un atstarpi)
InitDoc	1	NUMBER	InitDoc	1	NUMBER	Pazīme, vai tas ir sarakstes sākotnējais dokuments (1,0)
BisSubmissionID	10	NUMBER				BIS dokumenta identifikators
BisSubmissionNr	22	VARCHAR2				BIS dokumenta Nr.
cExec – dati par sarakstes administratīvām lietām.						
ExecObjectID	10	NUMBER	ExecObjectID	10	NUMBER	Administratīvās lietas identifikators
ExecObjectDocTypeID	10	NUMBER	ExecObjectDocTypeID	10	NUMBER	Administratīvās lietas veida identifikators
ExecObjectNr	22	VARCHAR2	ExecObjectNr	35	NVARCHAR2	Administratīvās lietas reģistrācijas Nr.
ExecObjectDate		DATE	ExecObjectDate		DATE	Administratīvās lietas reģistrācijas datums
ExecObjectAnnotation	1000	VARCHAR2	ExecObjectAnnotation	2000	VARCHAR2	Administratīvās lietas anotācija
ExecObjectCommentary	1000	VARCHAR2	ExecObjectCommentary	2000	VARCHAR2	Administratīvās lietas piezīmes
ExecObjectOwnerDivisionID	10	NUMBER	ExecObjectOwnerDivisionID	10	NUMBER	Administratīvās lietas saimnieka struktūrvienības identifikators
ExecObjectStatus	250	VARCHAR2	ExecObjectStatus	250	VARCHAR2	Administratīvās lietas statuss
BisExecObjectID	10	NUMBER				Saistītās BIS lietas identifikators

RD LIS			DAM			
RD LIS nosaukums	**Garums**	**Tips**	**DAM nosaukums**	**Garums**	**Tips**	**Apraksts**
BisExecObjectNr	22	VARCHAR2				Saistītās BIS lietas Nr.
cDocExec – Dati par administratīvo lietu dokumentiem						
SubmissionID	10	NUMBER	SubmissionID	10	NUMBER	Dokumenta identifikators
SubmissionDocTypeID	10	NUMBER	SubmissionDocTypeID	10	NUMBER	Dokumenta veida identifikators
SubmissionNr	22	VARCHAR2	SubmissionNr	35	NVARCHAR2	Dokumenta reģistrācijas Nr.
SubmissionDate		DATE	SubmissionDate		DATE	Dokumenta reģistrācijas datums
BisSubmissionID	10	NUMBER	BisSubmissionID	10	NUMBER	BIS dokumenta identifikators
BisSubmissionNr	22	VARCHAR2	BisSubmissionNr	22	VARCHAR2	BIS dokumenta Nr.
ExecObjectID	10	NUMBER	ExecObjectID	10	NUMBER	Administratīvās lietas identifikators
ExecObjectDocTypeID	10	NUMBER	ExecObjectDocTypeID	10	NUMBER	Administratīvās lietas veida identifikators
ExecObjectNr	22	VARCHAR2	ExecObjectNr	22	VARCHAR2	Administratīvās lietas reģistrācijas Nr.
ExecObjectDate		DATE	ExecObjectDate		DATE	Administratīvās lietas reģistrācijas datums

Pielikuma 11. tabula **Izvaddatu atbilstību tabula DOCUMENT_STATUS_LIST procedūrai**

RD LIS			**DAM**			
RD LIS nosaukums	**Garums**	**Tips**	**DAM nosaukums**	**Garums**	**Tips**	**Apraksts**
ID	10	NUMBER	ID	10	NUMBER	Identifikators
Type	1	NUMBER	Type	1	NUMBER	Dokumentu statusu veidi Ienākošo dokumentu statuss – 1, izejošo dokumentu statuss – 2
Nosaukums	250	VARCHAR2	Name	70	NVARCHAR2	Nosaukums
Kartas_nr	10	NUMBER	ORDER_INDEX	10	NUMBER	Kārtas numurs
Arhivets	1	NUMBER	Archived	1	NUMBER	Ieraksta arhivēšanas pazīme (0 – nearhivēts, 1 – arhivēts)
			Code	7	Varchar2	DAM statusa kods

Pielikuma 12. tabula **Izvaddatu atbilstību tabula SUBJECT_LIST procedūrai**

RD LIS			**DAM**			
Nosaukums	**Garums**	**Tips**	**DAM nosaukums**	**Garums**	**Tips**	**Apraksts**
ID	10	NUMBER	ID	10	NUMBER	Identifikators
Name	250	VARCHAR2	Name	250	VARCHAR2	Nosaukums
Archived	1	NUMBER	Archived	1	NUMBER	Ieraksta arhivēšanas pazīme (0 – Nearhivēts, 1 – Arhivēts)
DivisionID	10	NUMBER	DivisionID	10	NUMBER	Struktūrvienības identifikators

Programmatūras pirmkoda fragmenti

Kursoru datu tipu izveide

```
  TYPE document_status_list_t IS RECORD
  (
    ID            NUMBER(10),      --Identifikators
    Type          NUMBER(1),       --Dokumentu statusu veidi: Ienākošo dokumentu
statuss - 1, izejošo dokumentu statuss - 2
    Name          NVARCHAR2(70),   --Nosaukums
    ORDER_INDEX   NUMBER(10),      --Kārtas numurs
    Archived      NUMBER(1),       --Ieraksta arhivēšanas pazīme (0 - nearhivēts, 1
- arhivēts)
    Code          VARCHAR2(7)      --DAM status kods
  );
  TYPE document_status_list_tt IS TABLE OF document_status_list_t;

  TYPE subject_list_t IS RECORD
  (
    ID            NUMBER(10),      -- Identifikators
    Name          VARCHAR2(250),   -- Nosaukums
    Archived      NUMBER(1),       -- Ieraksta arhivēšanas pazīme (0 - nearhivēts, 1
- arhivēts)
    DivisionID    NUMBER(10)       -- Struktūrvienības identifikātors
  );
  TYPE subject_list_tt IS TABLE OF subject_list_t;

  TYPE cSubmission_list_t IS RECORD
  (
    ID            NUMBER(10),        -- Dokumenta identifikators
    Nr            NVARCHAR2(35),     -- Dokumenta numurs lietvedībā
    DocDate       DATE,              -- Dokumenta datums
    Annotation    NCLOB,             -- Īss komentārs par dokumentu
    RegDocTypeID  NUMBER(10),        -- Reģistrētā dokumenta veida identifikators
    RegDocType    NVARCHAR2(100),    -- Reģistrētā dokumenta veida nosaukums
    DivisionID    NUMBER(10),        -- Dokumenta struktūrvienības identifikators
    ResponseDate  DATE,              -- Dokumenta atbildes termiņš
    NotToAnswer   NUMBER(1),         -- Pazīme, ka nav jāatbild (0- jāatbild, 1-nav
jāatbild)
    Status        NUMBER(10),        -- Dokumenta statuss. Atgriež statusa id, kur
vērtību atrod no procedūras DOCUMENT_STATUS_LIST
    Answered      DATE,              -- Datums, kad atbildēts
    FromDate      DATE,              -- Spēkā no datums
    TillDate      DATE,              -- Spēkā līdz datums
    OriginalNr    VARCHAR2(50),      -- Ienākošā dokumenta korespondenta oriģinālā
dokumenta numurs
    OriginalDate  DATE,              -- Ienākošā dokumenta korespondenta oriģinālā
dokumenta datums
    Pages         NUMBER(10)         -- Lapu skaits
  );
  TYPE cSubmission_list_tt IS TABLE OF cSubmission_list_t;

  TYPE cDocument_list_t IS RECORD
  (
    ID          NUMBER(10),       -- Ieraksta identifikators DAM (Document_files.id)
    FileName    VARCHAR2(200),     -- Faila vārds
    DocName     NVARCHAR2(240),   -- Faila apraksts
    Repoid      NUMBER(10),       -- Repozitorija id
    EdocFileId  NUMBER,           -- Izpakotā edoc bērnu id
    IsSecure    NUMBER(1),        -- Ierobežotas pieejamības pazīme datnei (0 - nav,
1- ir)
    SORT_INDEX  NUMBER(3),        -- Datņu kārtas numurs (LP, lēmumi, sēdes)
    DocBDate    DATE              -- Datnes pievienošanas vai pēdējais labošanas
datums
  );
  TYPE cDocument_list_tt IS TABLE OF cDocument_list_t;

  TYPE cPerson_list_t IS RECORD
  (
    ID          NUMBER(10),      -- Ieraksta identifikators DAM Document_persons.id
```

```
    PersonName  VARCHAR2(120),  -- Dokumenta autors/adresāts
    PersonCode  VARCHAR2(11),   -- Korespondenta/adresāta personas kods vai NMRK
    PersonType  NUMBER(1),      -- Pārskaitāmais datu tips: (1 - juridiska persona,
3 - fiziska persona, 4 - pašvaldības darbinieks)
    Role        NUMBER(10),     -- Personas loma: adresāts, korespondents,
parakstītājs
    PersonID    NUMBER(10),     -- RD VIS fiziskas vai juridiskas personas
identifikators
    Address     VARCHAR2(160),  -- Adrese
    AddressID   NUMBER(10),     -- RD VIS adreses identifikators
    IsMain      NUMBER(1),      -- Galvenā persona. (1-galvenais, 0-nav) DAM
galvenā persona ir tikai parakstītājiem
    DivisionID  NUMBER(10),     -- Struktūrvienības identifikators
    EmployeeID  NUMBER(10)      -- Darbinieka slodzes ID
  );
  TYPE cPerson_list_tt IS TABLE OF cPerson_list_t;

  TYPE cResponse_list_t IS RECORD
  (
    ID          NUMBER(10),    -- atbildes dokumenta identifikators
    Nr          VARCHAR2(35),  -- Atbildes dokumenta numurs
    RespDate    DATE,          -- Atbildes dokumenta datums
    Annotation  NCLOB,         -- Atbildes dokumenta anotācija
    Type        VARCHAR2(1)    -- Vai atbildes dokuments ir starpatbilde (S) vai
galīgā atbilde (G)
  );
  TYPE cResponse_list_tt IS TABLE OF cResponse_list_t;

  TYPE cDivision_list_t IS RECORD
  (
    DivisionID NUMBER(10),     -- Struktūrvienības identifikators pie kā atrodas
dokuments
    EmployeeID NUMBER(10),     -- Rezolūcijas izpildītāja slodzes ID
    SendDate   DATE,           -- Datums, kad nosūtīta kustība/rezolūcija
    Resolution VARCHAR2(1000)  -- Rezolūcijas teksts
  );
  TYPE cDivision_list_tt IS TABLE OF cDivision_list_t;

  TYPE cExec_obj_list_t IS RECORD
  (
    ID                   NUMBER(10),      -- Administratīvās lietas identifikators
    ExecObjectNr         NVARCHAR2(35),   -- Administratīvās lietas numurs
    ExecObjectDate       DATE,            -- Administratīvās lietas datums
    ExecObjectAnnotation NCLOB            -- Administratīvās lietas anotācija
  );
  TYPE cExec_obj_list_tt IS TABLE OF cExec_obj_list_t;

  TYPE cNomentclature_list_t IS RECORD
  (
    ID              NUMBER(10),     -- Ieraksta identifikators
    DivisionID      NUMBER(10),     -- Struktūrvienības identifikators
    NomenclatureID  NUMBER(10),     -- Nomenklatūras numura identifikators
    NomenclatureDate DATE,          -- Lietā nodošanas datums
    NomenclatureCode VARCHAR2(100), -- Nomenklatūras numurs
    Year            VARCHAR2(10)    -- Lietas gads.
  );
  TYPE cNomenclature_list_tt IS TABLE OF cNomentclature_list_t;

  TYPE cRevision_list_t IS RECORD
  (
    Type          NUMBER(1),       -- Grozījuma tips: 0- padotais dokuments groza,
1- padoto dokumentu groza
    RevisDocID    NUMBER(10),      -- Ar grozījumu saistītā dokumenta
identifikators
    RevisDocNr    NVARCHAR2(35),   -- Ar grozījumu saistītā dokumenta numurs
    RevisDocDate  DATE,            -- Ar grozījumu saistītā dokumenta reģistrācijas
datums
    RevisType     VARCHAR(250),    -- Grozījuma veids (klasificēta vērtība)
    RevisDate     DATE,            -- Grozījuma datums
    Commentary    NVARCHAR2(2000)  -- Grozījuma paskaidrojums
  );
```

```
  TYPE cRevision_list_tt IS TABLE OF cRevision_list_t;

  TYPE cSubject_list_t IS RECORD
  (
    DivisionID  NUMBER(10),  -- Struktūrvienības identifikators
    SubjectID   NUMBER(10),  -- Jautājuma identifikators
    SubjectName VARCHAR2(250)  -- Jautājuma nosaukums
  );
  TYPE cSubject_list_tt IS TABLE OF cSubject_list_t;
```

Adaptera procedūru pirmkods

DOCUMENT_TYPE_LIST procedūra

```
-- Procedūra atgriež dokumentu veidu sarakstu.
PROCEDURE DOCUMENT_TYPE_LIST(po_DocumentTypeList OUT document_types_tt) IS
  v_switch_value NUMBER(1);
  v_list_dam SYS_REFCURSOR;
  v_row document_type_t;
BEGIN
  v_switch_value := U_SETTINGS_PCK.GetSwitchValue;

  -- Pielogo procedūras izsaukumus
  U_LOG_PCK.LogData(pi_level        => WEBLOG_PCK.c_info,
                    pi_description  => 'Procedūras izsaukums:
INTEGRATION_PUBLIC_PCK.Document_Type_List',
                    pi_data1        => 'Slēdža vērtība: '||v_switch_value,
                    pi_info2        =>
'INTEGRATION_PUBLIC_PCK.Document_Type_List');

  IF v_switch_value = const_pck.c_switch_rdlis THEN -- RDLIS
    PK_SERVISI_WRAP_TBL.REGISTERED_DOCUMENT_TYPE_VISI;

    SELECT t.ID, t.NOSAUKUMS, t.KARTOTEKA_NOS, NULL
    BULK COLLECT INTO po_DocumentTypeList
    FROM REG_DOC_TYPE_LIST_CUR_TEMP@RDLIS t;
  ELSE -- DAM
    API_DAM_ADAPTER_PCK.Document_Type_List(po_list => v_list_dam);

    LOOP
      FETCH v_list_dam
      INTO v_row.ID, v_row.Name, v_row.CardFileName, v_row.CardFileCode;
      EXIT WHEN v_list_dam%NOTFOUND;
      BEGIN
        po_DocumentTypeList.extend;
        po_DocumentTypeList(po_DocumentTypeList.count) := v_row;
      EXCEPTION
        WHEN OTHERS THEN
          po_DocumentTypeList := document_types_tt(v_row);
      END;
    END LOOP;
    CLOSE v_list_dam;
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    -- Pielogo kļūdu
    U_LOG_PCK.LogData(pi_level        => WEBLOG_PCK.c_error,
                      pi_description  => 'Notikusi kļūda, izsaucot
INTEGRATION_PUBLIC_PCK.Document_Type_List',
                      pi_data1        => 'Slēdža vērtība: '||v_switch_value,
                      pi_stack        => dbms_utility.format_error_stack || chr(10)
|| DBMS_UTILITY.FORMAT_ERROR_BACKTRACE,
                      pi_info2        =>
'INTEGRATION_PUBLIC_PCK.Document_Type_List');

    -- Un rais'o tālāk
    RAISE;
END DOCUMENT_TYPE_LIST;
```

```
--Procedūra atgriež korespondences jautājumu sarakstu.
PROCEDURE SUBJECT_LIST(
  pi_DivisionID IN NUMBER,            -- Struktūrvienības identifikators
  po_SubjectList OUT subject_list_tt  -- Atgriežamie dati
) IS
  null_value_exc EXCEPTION;  --Izņēmums apstrādei, kad netiek atrasts
struktūrvienības id

  v_switch_value NUMBER(1);
  v_rdlis_list PK_SERVISI_WRAP_PCK.subjects_tt;
  v_dam_list SYS_REFCURSOR;
  v_row subject_list_t;
  v_rdlis_division_id CL_DIVISIONS.LISST%TYPE;
  v_dam_division_id CL_DIVISIONS.STVID%TYPE;

  FUNCTION GetInputValuesForLog RETURN NVARCHAR2 IS
  BEGIN
    RETURN 'Slēdža vērtība: '||v_switch_value||'
            pi_DivisionID: '||pi_DivisionID;
  END GetInputValuesForLog;
BEGIN
  v_switch_value := U_SETTINGS_PCK.GetSwitchValue;

  U_LOG_PCK.LogData(pi_level            => WEBLOG_PCK.c_info,
                    pi_description      => 'Procedūras izsaukums:
INTEGRATION_PUBLIC_PCK.SUBJECT_LIST',
                    pi_data1            => GetInputValuesForLog(),
                    pi_info2            => 'INTEGRATION_PUBLIC_PCK.SUBJECT_LIST');

  IF v_switch_value = const_pck.c_switch_rdlis THEN --RDLIS

    BEGIN
      IF pi_DivisionID IS NOT NULL THEN
        v_rdlis_division_id := GET_LISST_ID_FROM_DIVISION_ID(pi_division_id =>
pi_DivisionID);
        --Pārbaude pēc RDLIS struktūrvienības identifikātora meklēšanas vai ir tāda
struktūrvienība
        IF v_rdlis_division_id IS NULL THEN
          RAISE null_value_exc;
        END IF;
      END IF;
    END;

    PK_SERVISI_WRAP_PCK.SUBJECT_LIST_TT(pi_division_id => v_rdlis_division_id,
po_list => v_rdlis_list);

    po_SubjectList := subject_list_tt();
    FOR i IN 1 .. v_rdlis_list.count LOOP
      --Pārveidojam RDLIS struktūrvienības identifikātoru par DAM struktūrvienības
identifikātoru
      v_dam_division_id := GET_DIVISION_ID_FROM_LISST_ID(pi_lisst_id =>
v_rdlis_list(i).divID);

      po_SubjectList.extend();
      po_SubjectList(po_SubjectList.count) := subject_list_t(v_rdlis_list(i).id,
                                                             v_rdlis_list(i).name,

v_rdlis_list(i).archived,
                                                             v_dam_division_id);
    END LOOP;
  ELSE  --DAM
    API_DAM_ADAPTER_PCK.Subject_List(pi_DivisionID => pi_DivisionID, po_list =>
v_dam_list);

    LOOP
      FETCH v_dam_list
      INTO v_row.ID, v_row.Name, v_row.Archived, v_row.DivisionID;
      EXIT WHEN v_dam_list%NOTFOUND;
```

```
      BEGIN

        --Tā kā DAM ir atribūts Active, kuram ir vērtības: 0 - dzēsts, 1 - aktīvs;
        --un arhivēšanas pazīmes vērtības ir: 0 - nearhivēts, 1 - arhivēts,
        --tādēļ saņemtajiem datiem tiek veikta pārbaude vai vērtība ir aktīva
        IF v_row.Archived = 0 THEN  --Ja ir dzēsts, tad ieliek vērtību, ka ir
arhivēts
          v_row.Archived := 1;
        ELSE  --Ja ir aktīvs, tad ieliek vērtību, ka nav arhivēts
          v_row.Archived := 0;
        END IF;

        po_SubjectList.extend();
        po_SubjectList(po_SubjectList.count) := v_row;
      EXCEPTION
        WHEN OTHERS THEN
          po_SubjectList := subject_list_tt(v_row);
      END;

    END LOOP;
  END IF;

EXCEPTION
  WHEN null_value_exc THEN
    U_LOG_PCK.LogData(pi_level            => WEBLOG_PCK.c_error,
                      pi_description      => 'Nav atrasta struktūrbienība',
                      pi_data1            => GetInputValuesForLog(),
                      pi_info2            =>
'INTEGRATION_PUBLIC_PCK.SUBJECT_LIST');
    RAISE;
  WHEN OTHERS THEN
    --Pielogo kļūdu
    U_LOG_PCK.LogData(pi_level            => WEBLOG_PCK.c_error,
                      pi_description      => 'Notikusi kļūda izsaucot
INTEGRATION_PUBLIC_PCK.SUBJECT_LIST',
                      pi_data1            => GetInputValuesForLog(),
                      pi_stack            => dbms_utility.format_error_stack ||
chr(10) || DBMS_UTILITY.FORMAT_ERROR_BACKTRACE,
                      pi_info2            =>
'INTEGRATION_PUBLIC_PCK.SUBJECT_LIST');
    RAISE;
END SUBJECT_LIST;
```

SAVE_INCOMMING procedūra

```
-- Mērķis: Reģistrēt no ārējās sistēmas ienākošo dokumentu (saņemto dokumentu,
personas iesniegumu).
PROCEDURE SAVE_INCOMMING(
  pi_Data IN NCLOB,   -- Ienākošie dati
  po_Data OUT NCLOB   -- Atgriežamie dati
) IS
  v_switch_value NUMBER(1);

  v_RegEmployeeName     V_EMPLOYEES.FIRST_NAME%TYPE;
  v_RegEmployeeLastName V_EMPLOYEES.LAST_NAME%TYPE;
  v_RegJobTitle         V_EMPLOYEES.POSITION%TYPE;
  v_RegPK               V_EMPLOYEES.CODE%TYPE;

  v_json_obj json_object_t; -- galvenais JSON
  v_out_json json_object_t; -- izejošais JSON

  v_ExternalID        NUMBER(10);     -- Iesnieguma identifikators ārējā sistēmā
  v_AppID             NUMBER(10);     -- Reģistrējamā dokumenta avots
  v_DocTypeCode       VARCHAR2(30);   -- Ārējās sistēmas dokumenta veida
identifikators
  v_Annotation        VARCHAR2(2000); -- Dokumenta anotācija (nosaukums, īss
komentārs)
  v_OwnerDivisionID   NUMBER(10);     -- Dokumenta saimnieka DAM institūcijas
identifikators - nosaka reģistrācijas Nr
```

```
  v_Offic             json_array_t;    -- Masīvs. Ienākošā dokumenta pašvaldības
adresāts, var būt vairāki adresāti. Padod struktūru (OfficDivisionID), darbinieka
slodzes ID (OfficeEmployeeID)
  v_RegDivisionID     NUMBER(10);      -- Dokumenta reģistrētājs
  v_RegEmployeeID     NUMBER(10);      -- Dokumenta reģistrētāja slodzes ID
  v_SubjectID         NUMBER(10);      -- Jautājuma identifikators
  v_NomenclatureID    NUMBER(10);      -- Reģistrētāja struktūrvienības lietu
nomenklatūras identifikators
  v_Pages             NUMBER(10);      -- Lapu skaits
  v_RelatedDoc        json_array_t;    -- Saistīto dokumentu masīvs, padod dokumenta
ID, numuru un tipu (0- saistītais dokuments, 1-atbilde uz)
  v_ResponseDate      DATE;            -- Dokumenta atbildes termiņš
  v_NotToanswer       NUMBER(1);       -- Pazīme, ka nav jāatbild (0- jāatbild, 1-
nav jāatbild)
  v_Status_code       VARCHAR2(7);     -- DAM dokumenta statusa kods (Atrod ar
metodi document_statuss_list)

  v_RelatedDocID      NUMBER(10);
  v_RelatedDocNr      VARCHAR2(22);
  v_ReplyToDocID      NUMBER(10);
  v_ReplyToDocNr      VARCHAR2(22);

  v_OfficEmployeeName     V_EMPLOYEES.FIRST_NAME%TYPE;
  v_OfficEmployeeLastName V_EMPLOYEES.LAST_NAME%TYPE;
  v_OfficJobTitle         V_EMPLOYEES.POSITION%TYPE;
  v_OfficPK               V_EMPLOYEES.CODE%TYPE;

  v_file_name         NVARCHAR2(200);
  v_file_description  NVARCHAR2(1000);
  v_file_date         DATE;
  v_file_author       NVARCHAR2(200);
  v_file_number       NVARCHAR2(20);
  v_file_doctype_code NVARCHAR2(100);
  v_file_metadata     SYS_REFCURSOR;

  v_Correspondents json_array_t;
  v_Files          json_array_t;

  v_DocID    NUMBER(10);      -- Izveidotā dokumenta identifikators
  v_DocNR    VARCHAR2(22);    -- Dokumenta reģistrācijas numurs
  v_RegDate  DATE;            -- Dokumenta reģistrēšanas datums

  FUNCTION GetInputValuesForLog RETURN NVARCHAR2 IS
  BEGIN
    RETURN 'Slēdža vērtība: '||v_switch_value||'
            pi_Data: '||pi_Data;
  END GetInputValuesForLog;

BEGIN
  v_switch_value := U_SETTINGS_PCK.GetSwitchValue;

  U_LOG_PCK.LogData(pi_level            => WEBLOG_PCK.c_info,
                    pi_description      => 'Procedūras izsaukums:
INTEGRATION_PUBLICK_PCK.SAVE_INCOMMING',
                    pi_data1            => GetInputValuesForLog(),
                    pi_info2            =>
'INTEGRATION_PUBLICK_PCK.SAVE_INCOMMING');

  IF v_switch_value = const_pck.c_switch_rdlis THEN -- RDLIS
    -- JSON struktūras validācija
    BEGIN
      v_json_obj := JSON_OBJECT_T(pi_Data);
    EXCEPTION
      WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20001, 'Padots nekorekts JSON');
    END;
    -- iegūstam datus no JSON
    BEGIN
      v_ExternalID := v_json_obj.get_Number('ExternalID');
```

```
      v_AppID := v_json_obj.get_Number('AppID');

      v_DocTypeCode := v_json_obj.get_String('DocTypeCode');

      v_Annotation := v_json_obj.get_String('Annotation');

      v_OwnerDivisionID := v_json_obj.get_Number('OwnerDivisionID');

      v_Offic := v_json_obj.get_Array('Offic');

      v_RegDivisionID := v_json_obj.get_Number('RegDivisionID');

      v_RegEmployeeID := v_json_obj.get_Number('RegEmployeeID');

      v_SubjectID := v_json_obj.get_Number('SubjectID');

      v_NomenclatureID := v_json_obj.get_Number('NomenclatureID');

      v_Pages := v_json_obj.get_Number('Pages');

      v_RelatedDoc := v_json_obj.get_Array('RelatedDoc');

      v_ResponseDate := v_json_obj.get_Date('ResponseDate');

      v_NotToanswer := v_json_obj.get_Number('NotToAnswer');

      v_Status_code := v_json_obj.get_String('StatusCode');

      v_Correspondents := v_json_obj.get_Array('Correspondents');

      v_Files := v_json_obj.get_Array('Files');
    EXCEPTION
      WHEN OTHERS THEN
        RAISE;
    END;

    U_UTILITIES_PCK.GetEmployeeInfoById(pi_employee_id        =>
TREAT(v_Offic.get(0) AS JSON_OBJECT_T).get_Number('OfficeEmployeeID'),
                                        po_employee_name      =>
v_OfficEmployeeName,
                                        po_employee_last_name =>
v_OfficEmployeeLastName,
                                        po_employee_position  => v_OfficJobTitle,
                                        po_employee_code      => v_OfficPK);

    U_UTILITIES_PCK.GetEmployeeInfoById(pi_employee_id        => v_RegEmployeeID,
                                        po_employee_name      => v_RegEmployeeName,
                                        po_employee_last_name =>
v_RegEmployeeLastName,
                                        po_employee_position  => v_RegJobTitle,
                                        po_employee_code      => v_RegPK);
    IF v_RelatedDoc IS NOT NULL THEN
      FOR i IN 0 .. v_RelatedDoc.Get_Size - 1 LOOP
        IF TREAT(v_RelatedDoc.get(i) AS JSON_OBJECT_T).get_Number('DocType') = 0
THEN
          v_RelatedDocID := TREAT(v_RelatedDoc.get(i) AS
JSON_OBJECT_T).get_Number('DocumentID');
          v_RelatedDocNr := TREAT(v_RelatedDoc.get(i) AS
JSON_OBJECT_T).get_String('DocumentNr');
          EXIT;
        END IF;
      END LOOP;

      FOR i IN 0 .. v_RelatedDoc.Get_Size - 1 LOOP
        IF TREAT(v_RelatedDoc.get(i) AS JSON_OBJECT_T).get_Number('DocType') = 1
THEN
          v_ReplyToDocID := TREAT(v_RelatedDoc.get(i) AS
JSON_OBJECT_T).get_Number('DocumentID');
          v_ReplyToDocNr := TREAT(v_RelatedDoc.get(i) AS
JSON_OBJECT_T).get_String('DocumentNr');
```

```
          EXIT;
        END IF;
      END LOOP;
    END IF;

    -- SAVE4
    PK_SERVISI_WRAP_PCK.Save4(pi_ExternalId            => v_ExternalID,
                              pi_Source                =>
U_UTILITIES_PCK.GetRdlisSourceByAppId(v_AppID),
                              pi_DocTypeID             =>
U_UTILITIES_PCK.GetRdlisDocTypeIdByExtCode(v_DocTypeCode),
                              pi_Date                  => null,
                              pi_BufferLess            => 1,
                              pi_Annotation            => v_Annotation,
                              pi_OwnerDivisionID       =>
GET_LISST_ID_FROM_DIVISION_ID(v_OwnerDivisionID),
                              pi_OfficDivisionID       =>
GET_LISST_ID_FROM_DIVISION_ID(TREAT(v_Offic.get(0) AS
JSON_OBJECT_T).get_Number('OfficDivisionID')),
                              pi_OfficEmployeeName     => v_OfficEmployeeName,
                              pi_OfficEmployeeLastName => v_OfficEmployeeLastName,
                              pi_OfficJobTitle         => v_OfficJobTitle,
                              pi_OfficPK               => v_OfficPK,
                              pi_RespDivisionID        => null,
                              pi_RespEmployeeName      => null,
                              pi_RespEmployeeLastName  => null,
                              pi_RespJobTitle          => null,
                              pi_RespPK                => null,
                              pi_RegDivisionID         =>
GET_LISST_ID_FROM_DIVISION_ID(v_RegDivisionID),
                              pi_RegEmployeeName       => v_RegEmployeeName,
                              pi_RegEmployeeLastName   => v_RegEmployeeLastName,
                              pi_RegJobTitle           => v_RegJobTitle,
                              pi_RegPK                 => v_RegPK,
                              pi_PrepDivisionID        => null,
                              pi_PrepEmployeeName      => null,
                              pi_PrepEmployeeLastName  => null,
                              pi_PrepJobTitle          => null,
                              pi_PrepPK                => null,
                              pi_SubjectID             => v_SubjectID,
                              pi_NomenclatureID        => v_NomenclatureID,
                              pi_Pages                 => v_Pages,
                              pi_RelatedDocID          => v_RelatedDocID,
                              pi_ReplayToDocID         => v_ReplyToDocID,
                              pi_ResponseDate          => v_ResponseDate,
                              pi_NotToAnswer           => v_NotToanswer,
                              pi_Status                =>
U_UTILITIES_PCK.GetRdlisStatusIdByDamCode(v_Status_code, 1),
                              pi_DocFlowID             => null,
                              pi_Commentary            => null,
                              pi_RelatedDocNr          => v_RelatedDocNr,
                              pi_ReplayToDocNr         => v_ReplyToDocNr,
                              po_ID                    => v_DocID,
                              po_Nr                    => v_DocNR,
                              po_RegDate               => v_RegDate);

    IF v_Offic.Get_Size > 1 THEN
      FOR i IN 1 .. v_Offic.Get_Size - 1 LOOP
        U_UTILITIES_PCK.GetEmployeeInfoById(pi_employee_id       =>
TREAT(v_Offic.get(i) AS JSON_OBJECT_T).get_Number('OfficeEmployeeID'),
                                            po_employee_name     =>
v_OfficEmployeeName,
                                            po_employee_last_name =>
v_OfficEmployeeLastName,
                                            po_employee_position  =>
v_OfficJobTitle,
                                            po_employee_code     => v_OfficPK);
        -- Add_Person2
        PK_SERVISI_WRAP_PCK.Add_Person2(pi_SubmissionID      => v_DocID,
```

```
                                        pi_PersonType        => 4,  -- Pašvaldības
darbnieks
                                        pi_PersonName        => NULL,
                                        pi_PersonFirstName   => NULL,
                                        pi_PersonCode        => v_OfficPK,
                                        pi_DivisionID        =>
GET_LISST_ID_FROM_DIVISION_ID(TREAT(v_Offic.get(i) AS
JSON_OBJECT_T).get_Number('OfficDivisionID')),
                                        pi_OfficialName      =>
v_OfficEmployeeLastName,
                                        pi_OfficialFirstName =>
v_OfficEmployeeName,
                                        pi_OfficialPosition  => v_OfficJobTitle,
                                        pi_PersonID          => NULL,
                                        pi_StreetAddress     => NULL,
                                        pi_City              => NULL,
                                        pi_AddressID         => NULL,
                                        pi_PostalCode        => NULL,
                                        pi_EMail             => NULL,
                                        pi_PhoneNumber       => NULL,
                                        pi_DeliveryTypeID    => NULL,
                                        pi_SubmissionDate    => v_RegDate,
                                        pi_SubmissionNr      => v_DocNR,
                                        pi_DocFlowID         => NULL);
      END LOOP;
    END IF;

    IF v_Correspondents IS NOT NULL AND v_Correspondents.Get_Size > 0 THEN
      FOR i IN 0 .. v_Correspondents.get_Size - 1 LOOP
        U_UTILITIES_PCK.GetEmployeeInfoById(pi_employee_id        =>
TREAT(v_Correspondents.get(i) AS JSON_OBJECT_T).get_Number('EmployeeID'),
                                            po_employee_name      =>
v_OfficEmployeeName,
                                            po_employee_last_name =>
v_OfficEmployeeLastName,
                                            po_employee_position  =>
v_OfficJobTitle,
                                            po_employee_code      => v_OfficPK);

        -- Add_Person2
        PK_SERVISI_WRAP_PCK.Add_Person2(pi_SubmissionID      => v_DocID,
                                        pi_PersonType        =>
TREAT(v_Correspondents.get(i) AS JSON_OBJECT_T).get_Number('PersonType'),
                                        pi_PersonName        =>
TREAT(v_Correspondents.get(i) AS JSON_OBJECT_T).get_String('PersonName'),
                                        pi_PersonFirstName   =>
TREAT(v_Correspondents.get(i) AS JSON_OBJECT_T).get_String('PersonFirstName'),
                                        pi_PersonCode        =>
TREAT(v_Correspondents.get(i) AS JSON_OBJECT_T).get_String('PersonCode'),
                                        pi_DivisionID        =>
GET_LISST_ID_FROM_DIVISION_ID(TREAT(v_Correspondents.get(i) AS
JSON_OBJECT_T).get_Number('DivisionID')),
                                        pi_OfficialName      =>
v_OfficEmployeeLastName,
                                        pi_OfficialFirstName =>
v_OfficEmployeeName,
                                        pi_OfficialPosition  => v_OfficJobTitle,
                                        pi_PersonID          =>
TREAT(v_Correspondents.get(i) AS JSON_OBJECT_T).get_Number('PersonID'),
                                        pi_StreetAddress     =>
TREAT(v_Correspondents.get(i) AS JSON_OBJECT_T).get_String('StreetAddress'),
                                        pi_City              =>
TREAT(v_Correspondents.get(i) AS JSON_OBJECT_T).get_String('City'),
                                        pi_AddressID         =>
TREAT(v_Correspondents.get(i) AS JSON_OBJECT_T).get_Number('AddressID'),
                                        pi_PostalCode        =>
TREAT(v_Correspondents.get(i) AS JSON_OBJECT_T).get_String('PostalCode'),
                                        pi_EMail             =>
TREAT(v_Correspondents.get(i) AS JSON_OBJECT_T).get_String('EMail'),
```

```
                                        pi_PhoneNumber       =>
TREAT(v_Correspondents.get(i) AS JSON_OBJECT_T).get_String('PhoneNumber'),
                                        pi_DeliveryTypeID    =>
U_UTILITIES_PCK.GetRdlisDelTypeIdByDamCode(TREAT(v_Correspondents.get(i) AS
JSON_OBJECT_T).get_String('DeliveryTypeCode')),
                                        pi_SubmissionDate    => v_RegDate,
                                        pi_SubmissionNr      => v_DocNR,
                                        pi_DocFlowID         => NULL);
      END LOOP;
    END IF;

    IF v_Files IS NOT NULL AND v_Files.Get_Size > 0 THEN
      FOR i IN 0 .. v_Files.Get_Size - 1 LOOP
        REPO_APP_DETAILS_PCK.DocumentDetailsShort(pi_document_id          =>
TREAT(v_Files.get(i) AS JSON_OBJECT_T).get_Number('RepositoryID'),
                                                  po_document_name        =>
v_file_name,
                                                  po_document_date        =>
v_file_date,
                                                  po_document_number      =>
v_file_number,
                                                  po_document_author      =>
v_file_author,
                                                  po_document_description =>
v_file_description,
                                                  po_document_docTypeCode =>
v_file_doctype_code,
                                                  po_document_metadata    =>
v_file_metadata);

        -- Add_Document2
        PK_SERVISI_WRAP_PCK.Add_Document3(pi_Submission_ID      => v_DocID,
                                        pi_Division_ID        =>
GET_LISST_ID_FROM_DIVISION_ID(v_RegDivisionID),
                                        pi_Employee_Name      =>
v_RegEmployeeName,
                                        pi_Employee_Last_Name =>
v_RegEmployeeLastName,
                                        pi_Job_Title          => v_RegJobTitle,
                                        pi_File_Name          => v_file_name,
                                        pi_Doc_Name           =>
v_file_description,
                                        pi_Is_Secure          =>
TREAT(v_Files.get(i) AS JSON_OBJECT_T).get_Number('IsSecure'),
                                        pi_Edoc_ID            =>
TREAT(v_Files.get(i) AS JSON_OBJECT_T).get_Number('RepositoryID'),
                                        pi_Edoc_ID_Is_Edoc    => CASE WHEN
REPO_APP_DETAILS_PCK.GetDocumentStatus(TREAT(v_Files.get(i) AS
JSON_OBJECT_T).get_Number('RepositoryID')) = 1 THEN 1 ELSE 0 END,
                                        pi_DocTypeID          =>
U_UTILITIES_PCK.GetRdlisDocTypeIdByExtCode(v_DocTypeCode),
                                        pi_UserPK             => v_RegPK,
                                        pi_Secure_Date        => NULL,
                                        pi_IPIGroundID        => NULL,
                                        pi_Signed             => NULL,
                                        pi_KartasNr           => NULL);
      END LOOP;
    END IF;

    v_out_json := json_object_t();
    v_out_json.put('DocID', v_DocID);
    v_out_json.put('DocNr', v_DocNR);
    v_out_json.put('RegDate', TO_CHAR(v_RegDate, CONST_PCK.c_date_format));

    po_Data := v_out_json.to_clob;

  ELSE -- DAM
    API_DAM_ADAPTER_PCK.SAVE_INCOMMING(pi_Data => pi_Data, po_Data => po_Data);
  END IF;
EXCEPTION
```

```
  WHEN OTHERS THEN
    -- Pielogo kļūdu
    U_LOG_PCK.LogData(pi_level        => WEBLOG_PCK.c_error,
                      pi_description  => 'Notikusi kļūda, izsaucot
INTEGRATION_PUBLIC_PCK.SAVE_INCOMMING',
                      pi_data1        => GetInputValuesForLog(),
                      pi_stack        => dbms_utility.format_error_stack || chr(10)
|| DBMS_UTILITY.FORMAT_ERROR_BACKTRACE,
                      pi_info2        => 'INTEGRATION_PUBLIC_PCK.SAVE_INCOMMING');

    RAISE;
END SAVE_INCOMMING;
```

DELETE_DOCUMENT procedūra

```
-- Mērķis: Nodrošināt iespēju nodzēst kļūdaini pievienotu dokumenta datni
dokumentam, kas veidots no ārējās sistēmas.
PROCEDURE DELETE_DOCUMENT(
  pi_SubmissionID IN NUMBER,    -- Dokumenta ieraksta ID
  pi_DocTypeCode  IN VARCHAR2, -- Ārējās sistēmas dokumenta veida kods
  pi_RepoID       IN NUMBER,    -- Datnes repozitorija ID
  pi_FileName     IN VARCHAR2   -- Datnes vārds
) IS
  v_switch_value NUMBER(1);

  v_file_id NUMBER(10);

  v_doc_type_id NUMBER(10);

  FUNCTION GetInputValuesForLog RETURN NVARCHAR2 IS
  BEGIN
    RETURN 'Slēdža vērtība: '||v_switch_value||'
            pi_SubmissionID: '||pi_SubmissionID||'
            pi_DocTypeCode: '||pi_DocTypeCode||'
            pi_RepoID: '||pi_RepoID||'
            pi_FileName'||pi_FileName;
  END GetInputValuesForLog;

BEGIN
  v_switch_value := U_SETTINGS_PCK.GetSwitchValue;

  U_LOG_PCK.LogData(pi_level            => WEBLOG_PCK.c_info,
                    pi_description      => 'Procedūras izsaukums:
INTEGRATION_PUBLICK_PCK.Delete_Document',
                    pi_data1            => GetInputValuesForLog(),
                    pi_info2            =>
'INTEGRATION_PUBLICK_PCK.Delete_Document');

  IF v_switch_value = const_pck.c_switch_rdlis THEN -- RDLIS

    v_doc_type_id := U_UTILITIES_PCK.GetRdlisDocTypeIdByExtCode(pi_doctype_code =>
pi_DocTypeCode);

    -- Atrod datnes ID
    PK_SERVISI_WRAP_TBL.SELECTS(p_id             => pi_SubmissionID,
                                p_nr             => NULL,
                                p_ReturnBinDoc   => 2,
                                p_RegDocKartoteka => NULL,
                                p_RegDocTypeId   => v_doc_type_id,
                                p_Source         => const_pck.c_rdlis_source_all);

    SELECT doc.ID INTO v_file_id
    FROM SELECTS_CUR_DOC_TEMP@rdlis doc
    WHERE doc.EDOCID = pi_RepoID
      AND doc.FileName = pi_FileName
      AND ROWNUM = 1;

    -- Izsauc datnes dzēšanu
    PK_SERVISI_WRAP_PCK.Delete_Document(pi_SubmissionID => pi_SubmissionID,
                                        pi_DocTypeID    => v_doc_type_id,
```

```
                                        pi_SubmissionNr => NULL,
                                        pi_FileID       => v_file_id,
                                        pi_FileName     => pi_FileName);

  ELSE -- DAM
    API_DAM_ADAPTER_PCK.Delete_Document(pi_SubmissionID => pi_SubmissionID,
pi_RepoID => pi_RepoID);
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    -- Pielogo kļūdu
    U_LOG_PCK.LogData(pi_level        => WEBLOG_PCK.c_error,
                      pi_description  => 'Notikusi kļūda, izsaucot
INTEGRATION_PUBLIC_PCK.Delete_Document',
                      pi_data1        => GetInputValuesForLog(),
                      pi_stack        => dbms_utility.format_error_stack || chr(10)
|| DBMS_UTILITY.FORMAT_ERROR_BACKTRACE,
                      pi_info2        => 'INTEGRATION_PUBLIC_PCK.Delete_Document');

    RAISE;
END DELETE_DOCUMENT;
```

DAM sistēmas procedūras priekš adaptera

SUBJECT_LIST procedūra DAM sistēmā

```
--Procedūra atgriež jautājumu klasifikatoru sarakstu.
procedure Subject_List(
  pi_DivisionID IN NUMBER,     --Struktūrvienības identifikators
  po_list OUT SYS_REFCURSOR    --Jautājumu klasifikatoru saraksts
) is
begin
  open po_list for
  select clt.ID, clt.Name, clt.Active as Archived, clt.Division_ID
         from cl_themes clt
         where (pi_DivisionID is null or clt.division_id = pi_DivisionID)
         and clt.theme_type = const_pck.theme_type_document;
end Subject_List;
```

ADD_DOCUMENT procedūra DAM sistēmā

```
-- Procedūra pievieno datni iepriekš saglabātam dokumentam
PROCEDURE Add_Document(
  pi_SubmissionID           IN NUMBER,     -- Dokumenta ID
  pi_Is_Secure              IN NUMBER,     -- IPI (0-nav, 1-IPI)
  pi_Repo_Id                IN NUMBER,     -- Repozitorija ID
  pi_SkipFillTags           IN NUMBER DEFAULT 0 -- Pazīme par tagu aizpildīšanu (1
- nevajag aizpildīt)
) IS
  v_file_description REPOSITORY_FILES.DOCU_DESCRIPTION%TYPE;
  v_file_id DOCUMENT_FILES.ID%TYPE;
  v_errors errorlist;
  v_call call_t;
  v_card_file_code CL_CARD_FILES.code%type;
  v_require_signing DOCUMENT_FILES.Require_Signing%type;
BEGIN
  v_call := CALL_T('API_DAM_ADAPTER_PCK', 'Add_Document',
                   PRM_TT(PRM_T('pi_SubmissionID', pi_SubmissionID),
                   PRM_T('pi_Is_Secure', pi_Is_Secure),
                   PRM_T('pi_Repo_Id', pi_Repo_Id)));

  select repo.docu_description
         into v_file_description
         from repository_files repo
         where pi_Repo_Id = repo.docu_id;

  I_DOCUMENT_PCK.GetDocumentCardFileCode(pi_document_id    => pi_SubmissionID,
                                         po_card_file_code => v_card_file_code,
```

```
                                        pio_errorList     => v_errors);

  I_LIETVARIS_API_PCK.ApiCheckErrors(pi_errors => v_errors, pi_call => v_call);

  -- Saņemtajiem dokumentiem un notikumiem pievieno kā informatīvo datni, pārējo
kartotēku dokumentiem - kā parakstāmo datni
  if v_card_file_code in (CONST_PCK.CardFileIncomingDoc, CONST_PCK.CardFileEvent)
then
    v_require_signing := 0;
  else
    v_require_signing := 1;
  end if;

  I_DOCUMENT_FILES_PCK.AddDocumentFile(pi_documentId      => pi_SubmissionID,
                                       pi_repositoryId    => pi_Repo_Id,
                                       pi_description     => v_file_description,
                                       pi_limitedAccess   => pi_Is_Secure,
                                       pi_requiredSigning => v_require_signing,
                                       po_fileId          => v_file_id,
                                       pio_errorList      => v_errors);

  I_LIETVARIS_API_PCK.ApiCheckErrors(pi_errors => v_errors, pi_call => v_call);

  -- Aizpilda tagus
  if pi_SkipFillTags <> 1 then
    I_TAGS_PCK.FillTagsForDocument(pi_document_id => pi_SubmissionID, po_errors =>
v_errors);
    I_LIETVARIS_API_PCK.ApiCheckErrors(pi_errors => v_errors, pi_call => v_call);
  end if;
END Add_Document;
```

SELECTS procedūra DAM sistēmā

```
-- Mērķis: Atlasīt detalizētu informāciju par vienu dokumentu
PROCEDURE SELECTS(
  pi_ID                 IN NUMBER,    -- Identifikators
  pi_Nr                 IN VARCHAR2, -- Dokumenta numurs lietvedībā
  pi_ReturnBinDoc       IN NUMBER     -- Pazīme vai dokuments tiek atgriezts ar
informāciju par repozitorija dokumentiem vai bez
                                      -- (0- bez, 1- ar pievienotajiem dokumentiem,
kam pieejamība nav ierobežota vai lietotājam ir tiesības atvērt ierobežotās
pieejamības dokumentu,
                                      -- 2- visi pievienotie dokumenti, neanalizējot
IPI)
) IS
  v_call call_t;

  v_statuss_name cl_document_statuss.name%type;
  v_statuss_code cl_document_statuss.code%type;

  v_errors errorlist;
BEGIN
  v_call := CALL_T('API_DAM_ADAPTER_PCK', 'SELECTS',
                   PRM_TT(PRM_T('pi_ID', pi_ID),
                   PRM_T('pi_Nr', pi_Nr),
                   PRM_T('pi_ReturnBinDoc', pi_ReturnBinDoc)));

  -- Ja dokuments ar tādu ID neeksistē rais'o kļūdu
  IF I_CLASSIFIER_PCK.CheckDocumentId(pi_id => pi_ID, pio_errors => v_errors) IS
NULL THEN
    raise_application_error(-20001, 'Dokuments neeksistē!');
  END IF;

  -- Izdzēš vērtības no pagaidu tabulas
  delete from da_selects_subm_temp;
  -- Submission
  -- Atrodam dokumenta statusa kodu, pēc kura saņemsim satusa id
  I_DOCUMENT_PCK.GetDocumentStatuss(pi_document_id           => pi_ID,
                                    po_document_statuss_name => v_statuss_name,
                                    po_document_statuss_code => v_statuss_code,
```

```
                                    pio_errorList           => v_errors);

  I_LIETVARIS_API_PCK.ApiCheckErrors(pi_errors => v_errors, pi_call => v_call);

  insert into da_selects_subm_temp(id, nr, docdate, annotation, regdoctypeid,
regdoctype, divisionid, responsedate,
                                   nottoanswer, status, answered, fromdate,
tilldate, originalnr, originaldate, pages)
  select doc.ID, doc.regnumber as Nr, doc.regdate as DocDate, doc.annotation,
doc.doctype_id as RegDocTypeID,
       t.name as RegDocType, doc.Division_Id, doc.Due_Date as ResponseDate,
       (CASE
         WHEN doc.regdate = doc.execution_date THEN 1
         ELSE 0
       END) as NotToAnswer,
       (select ds.id from cl_document_statuss ds where ds.code = v_statuss_code) as
Status,
       doc.Execution_Date as Answered, doc.Valid_From as FromDate, doc.Valid_Till
as TillDate,
       doc.Correspondent_Doc_No as OriginalNr, doc.Correspondent_Doc_Date as
OriginalDate,
       doc.Page_Count as Pages
  from documents doc
  join cl_document_types t
    on doc.doctype_id = t.id
  where doc.id = pi_ID
    and doc.regnumber = pi_Nr
    and doc.active = 1;

  -- Izdzēš vērtības no pagaidu tabulas
  delete from da_selects_doc_temp;
  if pi_ReturnBinDoc = 1 or pi_ReturnBinDoc = 2 then
    -- Document
    insert into da_selects_doc_temp(id, filename, docname, repoid, edocfileid,
issecure, sort_index, docbdate)
    select doc.ID, repo.docu_name as FileName, doc.Name as DocName,
       (CASE
         WHEN doc.parent_file_id IS NOT NULL THEN (
           select df.repository_id
           from document_files df
           where df.id = doc.parent_file_id
         )
         ELSE doc.repository_id
       END) as Repoid,
       (CASE
         WHEN doc.parent_file_id IS NOT NULL THEN doc.repository_id
         ELSE null
       END) as EdocFileId,
       doc.Limited_Access as IsSecure, doc.Order_Index as Sort_Index,
CAST(doc.Last_Stamp as DATE) as DocBDate
    from document_files doc
    join repository_files repo
      on repo.docu_id = doc.repository_id
    where doc.document_id = pi_ID
      and doc.active = 1
      and (pi_ReturnBinDoc = 2 or
I_DOCUMENT_PCK.ValidateDocumentOpenRightsNoEr(pi_document_file_id => doc.id) = 1);
  end if;

  -- Izdzēš vērtības no pagaidu tabulas
  delete from da_selects_pers_temp;
  -- Person
  insert into da_selects_pers_temp(id, personname, personcode, persontype, role,
personid, address, addressid,
                                   ismain, divisionid, employeeid)
  select pers.ID, trim(pers.First_Name||' '||pers.Last_Name) as PersonName,
pers.Code as PersonCode,
       (CASE
         WHEN pers.employee_id IS NOT NULL THEN const_pck.c_rdlis_pers_official --
pašvaldības darbnieks
```

```
         WHEN pers.client_id IS NOT NULL AND
              pt.code = const_pck.c_person_code THEN const_pck.c_rdlis_pers_phys --
fiziska persona
         WHEN pt.code = CONST_PCK.c_foreigner_code THEN CONST_PCK.c_pers_foreigner
-- nerezidents
         ELSE const_pck.c_rdlis_pers_client -- juridiska persona
       END) as PersonType,
       pers.Type as Role,
       (CASE
         WHEN pers.employee_id IS NOT NULL OR (pers.client_id IS NOT NULL AND
              pt.code = const_pck.c_person_code) THEN
                cli.person_id
         ELSE cli.company_id
       END) as PersonID, pers.Address, pers.Address_Id,
       pers.Isprimary as IsMain, pers.Employee_Division_ID as DivisionID,
pers.Employee_Id
  from document_persons pers
  left join clients cli
    on cli.id = pers.client_id
  left join cl_person_types pt
    on cli.perstype_id = pt.id
  where pers.document_id = pi_ID
    and pers.active = 1;

  -- Izdzēš vērtības no pagaidu tabulas
  delete from da_selects_resp_temp;
  -- Response
  insert into da_selects_resp_temp(id, nr, respdate, annotation, type)
  select dl.ID, doc.regnumber as Nr, doc.Execution_Date as RespDate,
doc.Annotation, dl.Type
  from document_links dl
  join documents doc
    on doc.id = dl.linked_document_id
  where dl.document_id = pi_ID
    and dl.type = const_pck.Document_link_answer
    and dl.active = 1;

  -- Izdzēš vērtības no pagaidu tabulas
  delete from da_selects_div_temp;
  -- Division
  insert into da_selects_div_temp(divisionid, employeeid, senddate, resolution)
  select res.DivisionId, res.employeeid, res.Rez_Datums as SendDate, res.Txt_Teksts
as Resolution
  from resolutionlist res
  join documents doc
    on doc.flow_document_id = res.flowDocumentId
  where doc.ID = pi_ID
    and res.Finished = 0
    -- projektus neatlasa
    and res.ResolutionActionId is not null
    and res.ResolutionActionId not in (CONST_PCK.ResActStateResProjSaved,
CONST_PCK.ResActStateManagerChanged)
    and not exists (
      select 1 from resolutionlist rl
        where rl.parentResolutionId = res.Rez_Id
    );

  -- Izdzēš vērtības no pagaidu tabulas
  delete from da_selects_exec_obj_temp;
  --ExecObject
  insert into da_selects_exec_obj_temp(id, execobjectnr, execobjectdate,
execobjectannotation)
  select vd.ID, doc.regnumber as ExecObjectNr, doc.Regdate as ExecObjectDate,
doc.Annotation as ExecObjectAnnotation
  from virtual_folder_documents vd
  join documents doc
    on doc.ID = vd.virtual_folder_id
  where vd.linked_document_id = pi_ID
    and vd.active = 1;
```

```
  -- Izdzēš vērtības no pagaidu tabulas
  delete from da_selects_nom_temp;
  -- Nomenclature
  insert into da_selects_nom_temp(id, divisionid, nomenclatureid, nomenclaturedate,
nomenclaturecode, year)
  select dn.id, nom.division_id as DivisionID, dn.Nomenclature_Id, dn.Case_Date as
NomenclatureDate,
       nom.code as NomenclatureCode, dn.Case_Year as Year
  from document_nomenclatures dn
  join cl_nomenclatures nom
    on dn.nomenclature_id = nom.id
  where dn.document_id = pi_ID
    and dn.active = 1;

  -- Izdzēš vērtības no pagaidu tabulas
  delete from da_selects_rev_temp;
  -- Revision
  insert into da_selects_rev_temp(type, revisdocid, revisdocnr, revisdocdate,
revistype, revisdate, commentary)
  select -- padotais dokuments groza
    0 as Type, doc.id as RevisDocID, doc.regnumber as RevisDocNr, doc.regdate as
RevisDocDate,
    cat.name as RevisType, da.amendment_date as RevisDate, da.explanation as
Commentary
  from document_amendments da
  join documents doc
    on doc.id = da.linked_document_id
  join cl_amendment_types cat
    on da.cl_amendment_type_id = cat.id
  where da.linked_document_id = pi_ID
    and da.active = 1
  union
  select -- padoto dokumentu groza
    1 as Type, doc.id as RevisDocID, doc.regnumber as RevisDocNr, doc.regdate as
RevisDocDate,
    cat.name as RevisType, da.amendment_date as RevisDate, da.explanation as
Commentary
  from document_amendments da
  join documents doc
    on doc.id = da.document_id
  join cl_amendment_types cat
    on da.cl_amendment_type_id = cat.id
  where da.document_id = pi_ID
    and da.active = 1;

  -- Izdzēš vērtības no pagaidu tabulas
  delete from da_selects_subj_temp;
  -- Subject
  insert into da_selects_subj_temp(divisionid, subjectid, subjectname)
  select doc.division_id, dt.theme_id as SubjectID, ct.name as SubjectName
  from document_themes dt
  join documents doc
    on doc.id = dt.document_id
  join cl_themes ct
    on ct.id = dt.theme_id
  where dt.document_id = pi_ID
    and dt.active = 1;

END SELECTS;
```

```
-- Mērķis: Reģistrēt no ārējās sistēmas ienākošo dokumentu (saņemto dokumentu,
personas iesniegumu).
PROCEDURE SAVE_INCOMMING(
  pi_Data IN NCLOB,  -- Ienākošie dati
  po_Data OUT NCLOB  -- Atgriežamie dati
) IS
  v_json json;
  v_sub json;
  v_jlist json_list;
  v_found NUMBER(1);
  v_errors errorlist;
  v_call CALL_T;

  TYPE offic_list_t IS RECORD (OfficDivisionID NUMBER(10), OfficeEmployeeID
NUMBER(10));
  TYPE offic_list_tt IS TABLE OF offic_list_t;

  v_OfficDivID NUMBER(10);
  v_OfficeEmpID NUMBER(10);

  v_ExternalID NUMBER(10);
  v_AppID cl_document_integration.outside_app_id%type;
  v_DocTypeCode VARCHAR2(30);
  v_Annotation VARCHAR2(2000);
  v_OwnerDivisionID NUMBER(10);
  v_Offic offic_list_tt;  -- Masīvs
  v_RegDivisionID NUMBER(10);
  v_RegEmployeeID NUMBER(10);
  v_SubjectID cl_document_integration.lietvaris_theme_id%type;
  v_NomenclatureID cl_document_integration.lietvaris_nomenclature_id%type;
  v_Pages NUMBER(10);
  v_RelatedDoc related_documents_tt;  -- Masīvs
  v_ResponseDate DATE;
  v_NotToAnswer NUMBER(1);
  v_Status_code VARCHAR2(7);
  v_SubmissionDate DATE;
  v_SubmissionNr VARCHAR2(22);
  v_ExternalLinks VARCHAR2(500);
  v_Addressees addressees_tt;
  v_Correspondents addressees_tt;  -- Masīvs
  v_Files files_tt;  -- Masīvs

  v_DocID NUMBER(10);
  v_DocNR VARCHAR2(22);
  v_RegDate DATE;
BEGIN
  v_call := CALL_T('API_DAM_ADAPTER_PCK', 'SAVE_INCOMMING', PRM_TT(PRM_T('pi_Data',
pi_Data)));
  -- Pielogo izsaukumu
  u_log_pck.LogDebug(v_call);

  -- JSON struktūras validācija
  BEGIN
    v_json := json(pi_Data);
  EXCEPTION
    WHEN OTHERS THEN
      u_error_pck.AddErrorToList(v_errors, null, 'err:invalid:json');
      RAISE;
  END;

  -- JSON deserializācija
  BEGIN
    v_ExternalID := u_json_pck.GetNumberProperty(v_json, 'ExternalID', v_errors);

    v_AppID := u_json_pck.GetNumberProperty(v_json, 'AppID', v_errors);

    v_DocTypeCode := u_json_pck.GetStringProperty(v_json, 'DocTypeCode', v_errors);
```

```
    v_Annotation := u_json_pck.GetStringProperty(v_json, 'Annotation', v_errors);

    v_OwnerDivisionID := u_json_pck.GetNumberProperty(v_json, 'OwnerDivisionID',
v_errors);

    u_json_pck.GetJsonList(v_json, 'Offic', v_jlist, v_found);
    IF v_jlist.count > 0 THEN
      v_Offic := offic_list_tt();
      FOR i IN 1 .. v_jlist.count LOOP
        v_sub := json(v_jlist.get(i));

        v_Offic.extend();
        v_Offic(i).OfficDivisionID := u_json_pck.GetNumberProperty(v_sub,
'OfficDivisionID', v_errors);
        v_Offic(i).OfficeEmployeeID := u_json_pck.GetNumberProperty(v_sub,
'OfficeEmployeeID', v_errors);
      END LOOP;

      IF v_Offic.count > 0 THEN
        v_OfficeEmpID := v_Offic(1).OfficeEmployeeID;
        v_OfficDivID := v_Offic(1).OfficDivisionID;
      ELSE
        v_OfficeEmpID := NULL;
        v_OfficDivID := NULL;
      END IF;
    END IF;

    v_RegDivisionID := u_json_pck.GetNumberProperty(v_json, 'RegDivisionID',
v_errors);

    v_RegEmployeeID := u_json_pck.GetNumberProperty(v_json, 'RegEmployeeID',
v_errors);

    v_SubjectID := u_json_pck.GetNumberProperty(v_json, 'SubjectID', v_errors);

    v_NomenclatureID := u_json_pck.GetNumberProperty(v_json, 'NomenclatureID',
v_errors);

    v_Pages := u_json_pck.GetNumberProperty(v_json, 'Pages', v_errors);

    u_json_pck.GetJsonList(v_json, 'RelatedDoc', v_jlist, v_found);
    IF v_jlist.count > 0 THEN
      v_RelatedDoc := related_documents_tt();
      FOR i IN 1 .. v_jlist.count LOOP
        v_sub := json(v_jlist.get(i));

        v_RelatedDoc.extend();
        v_RelatedDoc(i).DocumentID := u_json_pck.GetNumberProperty(v_sub,
'DocumentID', v_errors);
        v_RelatedDoc(i).DocumentNr := u_json_pck.GetStringProperty(v_sub,
'DocumentNr', v_errors);
        v_RelatedDoc(i).DocType := u_json_pck.GetNumberProperty(v_sub, 'DocType',
v_errors);
      END LOOP;
    END IF;

    v_ResponseDate := u_json_pck.GetDateProperty(v_json, 'ResponseDate', v_errors);

    v_NotToAnswer := u_json_pck.GetNumberProperty(v_json, 'NotToAnswer', v_errors);

    v_Status_code := u_json_pck.GetStringProperty(v_json, 'StatusCode', v_errors);

    v_SubmissionDate := nvl(u_json_pck.GetDateProperty(v_json, 'SubmissionDate',
v_errors), trunc(sysdate));

    v_SubmissionNr := u_json_pck.GetNumberProperty(v_json, 'SubmissionNr',
v_errors);

    v_ExternalLinks := u_json_pck.GetStringProperty(v_json, 'ExternalLinks',
v_errors);
```

```
    -- Pievienojam pārējos Offic adresātus pie saraksta
    v_Addressees := addressees_tt();
    IF v_Offic.count > 1 THEN
      FOR i IN 2 .. v_Offic.count LOOP
        v_Addressees.extend();
        v_Addressees(v_Addressees.count).DivisionID := v_Offic(i).OfficDivisionID;
        v_Addressees(v_Addressees.count).EmployeeID := v_Offic(i).OfficeEmployeeID;
      END LOOP;
    END IF;

    -- Pievienojam korespondentus
    v_Correspondents := addressees_tt();
    u_json_pck.GetJsonList(v_json, 'Correspondents', v_jlist, v_found);
    IF v_jlist.count > 0 THEN
      FOR i IN 1 .. v_jlist.count LOOP
        v_sub := json(v_jlist.get(i));

        v_Correspondents.extend();
        JsonToAddresseess(pi_json => v_sub, po_row =>
v_Correspondents(v_Correspondents.last), po_errors => v_errors);
      END LOOP;
    END IF;

    u_json_pck.GetJsonList(v_json, 'Files', v_jlist, v_found);
    IF v_jlist.count > 0 THEN
      v_Files := files_tt();
      FOR i IN 1 .. v_jlist.count LOOP
        v_sub := json(v_jlist.get(i));

        v_Files.extend();
        v_Files(i).RepositoryID := u_json_pck.GetNumberProperty(v_sub,
'RepositoryID', v_errors);
        v_Files(i).IsSecure := u_json_pck.GetNumberProperty(v_sub, 'IsSecure',
v_errors);
        v_Files(i).Primary := u_json_pck.GetNumberProperty(v_sub, 'Primary',
v_errors);
      END LOOP;
    END IF;

  END;

  -- Pārbauda, vai nav kļūdas
  I_LIETVARIS_API_PCK.ApiCheckErrors(v_errors, v_call);

  INNER_SAVE_DOC(pi_ExternalID       => v_ExternalID,
                 pi_AppID            => v_AppID,
                 pi_DocTypeCode      => v_DocTypeCode,
                 pi_Movement         => const_pck.c_movement_incoming,
                 pi_BufferLess       => 1,
                 pi_RegDate          => NULL,
                 pi_ProjectNr        => NULL,
                 pi_Annotation       => v_Annotation,
                 pi_OwnerDivisionID  => v_OwnerDivisionID,
                 pi_OfficDivisionID  => v_OfficDivID,
                 pi_OfficEmployeeID  => v_OfficeEmpID,
                 pi_RespDivisionID   => null,
                 pi_RespEmployeeID   => null,
                 pi_RegDivisionID    => v_RegDivisionID,
                 pi_RegEmployeeID    => v_RegEmployeeID,
                 pi_PrepDivisionID   => null,
                 pi_PrepEmployeeID   => null,
                 pi_SubjectID        => v_SubjectID,
                 pi_NomenclatureID   => v_NomenclatureID,
                 pi_Pages            => v_Pages,
                 pi_Notes            => null,
                 pi_RelatedDoc       => v_RelatedDoc,
                 pi_ResponseDate     => v_ResponseDate,
                 pi_NotToAnswer      => v_NotToAnswer,
                 pi_StatusCode       => v_Status_code,
```

```
                  pi_FromDate         => null,
                  pi_TillDate         => null,
                  pi_Type             => null,
                  pi_ContrAdditType   => NULL,
                  pi_MainContractID   => NULL,
                  pi_Addressees       => v_Addressees,
                  pi_Correspondents   => v_Correspondents,
                  pi_Files            => v_Files,
                  pi_ExternalLink     => v_ExternalLinks,
                  pi_VirtFoldStatusID => null,
                  pi_SubmissionDate   => v_SubmissionDate,
                  pi_SubmissionNr     => v_SubmissionNr,
                  po_DocID            => v_DocID,
                  po_DocNr            => v_DocNR,
                  po_RegDate          => v_RegDate);

  v_json := json();
  u_json_pck.SetIntProperty(v_json, 'DocID', v_DocID);
  u_json_pck.SetStringProperty(v_json, 'DocNr', v_DocNR);
  u_json_pck.SetDateProperty(v_json, 'RegDate', v_RegDate);

  u_json_pck.writeToClob(v_json, po_Data);

END SAVE_INCOMMING;
```

DELETE_DOCUMENT procedūra DAM sistēmā

```
-- Mērķis: Nodrošināt iespēju nodzēst kļūdaini pievienotu dokumenta datni
dokumentam, kas veidots no ārējās sistēmas.
PROCEDURE Delete_Document(
  pi_SubmissionID IN NUMBER,  -- Dokumenta ieraksta ID
  pi_RepoID       IN NUMBER   -- Datnes repozitorija ID
) IS
  v_call CALL_T;
  v_errors errorlist;

  v_file_id document_files.id%type;
BEGIN
  v_call := CALL_T('API_DAM_ADAPTER', 'Delete_Document',
PRM_TT(PRM_T('pi_SubmissionID', pi_SubmissionID), PRM_T('pi_RepoID', pi_RepoID)));
  -- Pielogo izsaukumu
  u_log_pck.LogDebug(v_call);

  begin
    select df.id into v_file_id
    from document_files df
    where df.document_id = pi_SubmissionID
      and df.repository_id = pi_RepoID
      and df.active = 1
      and rownum = 1; -- ja ir vairākas vienādas datnes, tad izdzēš tikai vienu
   exception
     when no_data_found then
       raise_application_error(-20001, 'Datni nav iespējams identificēt');
   end;

  I_DOCUMENT_FILES_PCK.DeleteDocumentFile(pi_documentFileId => v_file_id,
                                          pi_documentId     => pi_SubmissionID,
                                          pio_warning       => v_errors);

  -- Pārbauda, vai nav kļūdas
  I_LIETVARIS_API_PCK.ApiCheckErrors(v_errors, v_call);

END Delete_Document;
```

DOCUMENT_STATUS_LIST vienību tests

```
-- Pārbauda dokumentu statusu saraksta atlasīšanu
function Document_Status_List(pi_switch_value in number default null)
return boolean is
  v_hasRecords boolean := false;
  v_resultList INTEGRATION_PUBLIC_PCK.document_status_list_tt;
begin
  if pi_switch_value is not null then
    u_settings_pck.SetSwitchValue(pi_switch_value);
  end if;

  INTEGRATION_PUBLIC_PCK.Document_Status_List(po_DocumentStatusList =>
v_resultList);

  for v_row in (select * from table(v_resultList)) loop
    v_hasRecords := true;
    dbms_output.put_line(v_row.name);
  end loop;

  return v_hasRecords;
exception
  when others then
    dbms_output.put_line(dbms_utility.format_error_stack || chr(10) ||
DBMS_UTILITY.FORMAT_ERROR_BACKTRACE);
    return false;
end;

-- Pārbauda dokumentu statusu saraksta atlasīšanu RDLIS
function Document_Status_List_rdlis return boolean is
begin
  return Document_Status_List(pi_switch_value => const_pck.c_switch_rdlis);
end;

-- Pārbauda dokumentu statusu saraksta atlasīšanu DAM
function Document_Status_List_dam return boolean is
begin
  return Document_Status_List(pi_switch_value => const_pck.c_switch_dam);
end;
```

SUBJECT_LIST vienību tests

```
-- Pārbauda korespondences jautājumu saraksta atlasīšanu
function Subject_List(
  pi_switch_value in number default null,
  pi_division_id in number default null  -- null - atlasīt visus
jautājumus, struktūrvienības id - atlasīt strukturvienības jautājumus
) return boolean is
  v_hasRecords boolean := false;
  v_resultList INTEGRATION_PUBLIC_PCK.subject_list_tt;
begin
  if pi_switch_value is not null then
    u_settings_pck.SetSwitchValue(pi_switch_value);
  end if;

  INTEGRATION_PUBLIC_PCK.Subject_List(pi_DivisionID => pi_division_id,
po_SubjectList => v_resultList);

  for v_row in (select * from table(v_resultList)) loop
    v_hasRecords := true;
    dbms_output.put_line(v_row.name);
```

```
  end loop;

  return v_hasRecords;
exception
  when others then
    dbms_output.put_line(dbms_utility.format_error_stack || chr(10) ||
DBMS_UTILITY.FORMAT_ERROR_BACKTRACE);
    return false;
end Subject_List;

-- Pārbauda visu korespondences jautājumu atlasīšanu RDLIS
function Subject_List_all_rdlis return boolean is
begin
  return Subject_List(pi_switch_value => const_pck.c_switch_rdlis);
end Subject_List_all_rdlis;

-- Pārbauda korespondences jautājumu atlasīsanu konkrētai struktūrvienībai
RDLIS
function Subject_List_div_rdlis return boolean is
begin
  return Subject_List(pi_switch_value => const_pck.c_switch_rdlis,
pi_division_id => 1);
end Subject_List_div_rdlis;

-- Pārbauda visu korespondences jautajumu atlasīšanu DAM
function Subject_List_all_dam return boolean is
begin
  return Subject_List(pi_switch_value => const_pck.c_switch_dam);
end Subject_List_all_dam;

-- Pārbauda korespondences jautājumu atlasīsanu konkrētai struktūrvienībai
DAM
function Subject_List_div_dam return boolean is
begin
  return Subject_List(pi_switch_value => const_pck.c_switch_dam,
pi_division_id => 1);
end Subject_List_div_dam;
```

SELECTS vienību tests

```
-- Pārbauda dokumenta detalizēto atlasīšanu
function Selects(
  pi_switch_value in number default null,
  pi_submission_id in number default null,
  pi_regnumber in varchar2 default null
) return boolean is
  v_session session_pck.t_session;
  v_doc_id  number(10);
  v_doc_nr  varchar2(35);
  v_doctype_id  number(10);

  v_Submission INTEGRATION_PUBLIC_PCK.cSubmission_list_tt;
  v_Document INTEGRATION_PUBLIC_PCK.cDocument_list_tt;
  v_Person INTEGRATION_PUBLIC_PCK.cPerson_list_tt;
  v_Response INTEGRATION_PUBLIC_PCK.cResponse_list_tt;
  v_Division INTEGRATION_PUBLIC_PCK.cDivision_list_tt;
  v_ExecObj INTEGRATION_PUBLIC_PCK.cExec_obj_list_tt;
  v_Nomenclature INTEGRATION_PUBLIC_PCK.cNomenclature_list_tt;
  v_Revision INTEGRATION_PUBLIC_PCK.cRevision_list_tt;
  v_Subject INTEGRATION_PUBLIC_PCK.cSubject_list_tt;
begin
  v_session := session_pck.GetSessionData_noExc();
  if pi_switch_value is not null then
```

```
    u_settings_pck.SetSwitchValue(pi_switch_value);
  end if;

  GetOneSubmissionID(pi_id         => pi_submission_id,
                     pi_regnumber  => pi_regnumber,
                     po_id         => v_doc_id,
                     po_regnumber  => v_doc_nr,
                     po_doctype_id => v_doctype_id);

  INTEGRATION_PUBLIC_PCK.SELECTS(pi_ID                => v_doc_id,
                                 pi_Nr                => v_doc_nr,
                                 pi_ReturnBinDoc      => 2,
                                 pi_EmployeeID        =>
v_session.slodze_id,
                                 pi_EmployeeDivisionID => v_session.stv_id,
                                 po_Submission        => v_Submission,
                                 po_Document          => v_Document,
                                 po_Person            => v_Person,
                                 po_Response          => v_Response,
                                 po_Division          => v_Division,
                                 po_ExecObject        => v_ExecObj,
                                 po_Nomenclature      => v_Nomenclature,
                                 po_Revision          => v_Revision,
                                 po_Subject           => v_Subject);

  if v_Submission.count = 1 then
    -- Submission dati
    dbms_output.put_line('--- Dokumenta kartītes pamatdati ---');
    FOR i IN 1..v_Submission.count LOOP
      dbms_output.put_line('ID: '||v_Submission(i).ID||'| Nr:
'||v_Submission(i).Nr||'| Annotation: '||v_Submission(i).Annotation||'|
Status: '||v_Submission(i).Status);
    END LOOP;

    -- Document dati
    dbms_output.put_line('--- Dokumentam pievienotās datnes ---');
    FOR i IN 1..v_Document.count LOOP
      dbms_output.put_line('ID: '||v_Document(i).ID||'| FileName:
'||v_Document(i).FileName||'| DocName: '||v_Document(i).DocName);
    END LOOP;

    -- Person dati
    dbms_output.put_line('--- Dati par korespondentu vai adresātu, un
parakstītājiem ---');
    FOR i IN 1..v_Person.count LOOP
      dbms_output.put_line('Name: '||v_Person(i).PersonName||'| PersonID:
'||v_Person(i).PersonID||'| EmployeID: '||v_Person(i).EmployeeID);
    END LOOP;

    -- Response dati
    dbms_output.put_line('--- Ar dokumentu saistītie atbildes dokumenti ---
');
    FOR i IN 1..v_Response.count LOOP
      dbms_output.put_line('ID: '||v_Response(i).ID||'| Nr:
'||v_Response(i).Nr||'| Annotation: '||v_Response(i).Annotation);
    END LOOP;

    -- Division dati
    dbms_output.put_line('--- Dati par darbiniekiem pie kā atrodas
dokuments ---');
    FOR i IN 1..v_Division.count LOOP
```

```
      dbms_output.put_line('DivisionID: '||v_Division(i).DivisionID||'|
EmployeeID: '||v_Division(i).EmployeeID||'| Resolution:
'||v_Division(i).Resolution);
    END LOOP;

    -- ExecObject dati
    dbms_output.put_line('--- Dati par dokumentam piesaistītajām
virtuālajām ---');
    FOR i IN 1..v_ExecObj.count LOOP
      dbms_output.put_line('ID: '||v_ExecObj(i).ID||'| Nr:
'||v_ExecObj(i).ExecObjectNr);
    END LOOP;

    -- Nomenclature dati
    dbms_output.put_line('--- Dati par dokumentam piesaistītajiem
nomenklatūras lietas numuriem un lietā nodošanas datumiem visās
struktūrvienībās ---');
    FOR i IN 1..v_Nomenclature.count LOOP
      dbms_output.put_line('ID: '||v_Nomenclature(i).ID||'| NomenclatureID:
'||v_Nomenclature(i).NomenclatureID||'| NomenclatureCode:
'||v_Nomenclature(i).NomenclatureCode);
    END LOOP;

    -- Revision dati
    dbms_output.put_line('--- Dati par rīkojuma vai lēmuma grozījumiem ---
');
    FOR i IN 1..v_Revision.count LOOP
      dbms_output.put_line('RevisDocID: '||v_Revision(i).RevisDocID);
    END LOOP;

    -- Subject dati
    dbms_output.put_line('--- Dati par dokumentam piesaistītajiem
jautājumiem visās struktūrvienībās ---');
    FOR i IN 1..v_Subject.count LOOP
      dbms_output.put_line('DivisionID: '||v_Subject(i).DivisionID||'|
SubjectID: '||v_Subject(i).SubjectID||'| Annotation:
'||v_Subject(i).SubjectName);
    END LOOP;

    return true;
  else
    return false;
  end if;
exception
  when others then
    dbms_output.put_line(dbms_utility.format_error_stack || chr(10) ||
DBMS_UTILITY.FORMAT_ERROR_BACKTRACE);
    return false;
end Selects;

-- Pārbauda dokumenta detalizēto atlasīšanu RDLIS
function Selects_rdlis return boolean is
begin
  return Selects(pi_switch_value => const_pck.c_switch_rdlis);
end Selects_rdlis;

-- Pārbauda dokumenta detalizēto atlasīšanu DAM
function Selects_dam return boolean is
begin
  return Selects(pi_switch_value => const_pck.c_switch_dam);
end Selects_dam;
```

```
-- Pārbauda dokumenta atvienošanu no administratīvās lietas
function RemoveExecutiveObject(
  pi_switch_value in number default null,
  pi_submission_id in number default null,
  pi_regnumber in varchar2 default null,
  pi_exec_submission_id in number default null,
  pi_exec_regnumber in varchar2 default null
) return boolean is
  v_session session_pck.t_session;

  v_doc_id  number(10);
  v_doc_nr  varchar2(35);
  v_doctype_id  number(10);
  v_vl_doc_id number(10);
  v_vl_doc_nr varchar2(35);

  v_List INTEGRATION_PUBLIC_PCK.list_tt;

  v_data_in NCLOB;
  v_data_out NCLOB;
  v_json JSON_OBJECT_T;
  v_json_array json_array_t;
  v_sub_json json_object_t;
begin
  v_session := session_pck.GetSessionData_noExc();
  if pi_switch_value is not null then
    u_settings_pck.SetSwitchValue(pi_switch_value);
  end if;

  if pi_submission_id is null and pi_regnumber is null then
    -- Dokumenta izveidošana administratīvai lietai, kuru pēc pievienošanas
atvienos
    v_json := json_object_t();
    v_json.put('ExternalID', -1);
    v_json.put('AppID', c_test_app_id);
    v_json.put('DocTypeCode', c_test_doctype_code);
    v_json.put('BufferLess', 1);
    v_json.put('Annotation', 'Testa dokuments no DAM adaptera (JSON)
'||systimestamp);
    v_json.put('OwnerDivisionID', 1); -- Rīgas dome
    v_json.put('OfficDivisionID', v_session.stv_id);
    v_json.put('OfficEmployeeID', v_session.slodze_id);
    v_json.put('RespDivisionID', v_session.stv_id);
    v_json.put('RespEmployeeID', v_session.slodze_id);
    v_json.put('RegDivisionID', v_session.stv_id);
    v_json.put('RegEmployeeID', v_session.slodze_id);
    v_json.put('PrepDivisionID', v_session.stv_id);
    v_json.put('PrepEmployeeID', v_session.slodze_id);
    v_json.put_null('SubjectID');
    v_json.put_null('NomenclatureID');
    v_json.put_null('Pages');
    v_json.put_null('RelatedDoc');
    v_json.put('ResponseDate', to_char(sysdate, const_pck.c_date_format));
    v_json.put_null('NotToAnswer');
    v_json.put_null('StatusCode');
    v_json.put_null('FromDate');
    v_json.put_null('TillDate');
    v_json.put_null('Type');
    v_json.put_null('Addressees');
    v_json.put_null('ExternalLink');
    v_json.put_null('FlowOnlySave');
```

```
    v_json.put_null('Coordinators');
    v_json.put('Callback', 'DAM_ADAPTER.DUMMY_PCK.TEST');

    v_json_array := json_array_t();
    v_sub_json := json_object_t.parse('{"RepositoryID":
'||c_test_repo_id||', "Primary": 1, "IsSecure": 0}');
    v_json_array.append(v_sub_json);
    v_json.put('Files', v_json_array);

    v_data_in := v_json.to_clob();
    INTEGRATION_PUBLIC_PCK.SAVE_OUTGOING(pi_Data => v_data_in, po_Data =>
v_data_out);

    v_json := json_object_t.parse(v_data_out);
    v_doc_id := v_json.get_number('DocID');
    v_doc_nr := v_json.get_string('DocNr');
  else
    v_doc_id := pi_submission_id;
    v_doc_nr := pi_regnumber;
  end if;

  INTEGRATION_PUBLIC_PCK.LIST(pi_ID                      => null,
                              pi_Nr                      => v_doc_nr,
                              pi_FromDate                => null,
                              pi_TillDate                => null,
                              pi_PersonID                => null,
                              pi_PersonName              => null,
                              pi_PersonCode              => null,
                              pi_Address                 => null,
                              pi_Annotation              => null,
                              pi_DivisionID              => null,
                              pi_RegDocTypeID            => null,
                              pi_CardfileCode            => null,
                              pi_DocFlowDivisionID       => null,
                              pi_DocFlowDateFrom         => null,
                              pi_DocFlowDateTo           => null,
                              pi_NomenclatureID          => null,
                              pi_DocFlowHLDivisionId     => null,
                              pi_FromDocBDate            => null,
                              pi_TillDocBDate            => null,
                              pi_DocChangeFromDate       => null,
                              pi_DocChangeTillDate       => null,
                              pi_HLDivisionID            => null,
                              pi_IntegratedDivisionID    => null,
                              pi_IntegratedHLDivisionID => null,
                              pi_EmployeeID              =>
v_session.slodze_id,
                              pi_EmployeeDivisionID      =>
v_session.stv_id,
                              po_List                    => v_List);

  SELECT t.REGDOCTYPEID INTO v_doctype_id
  FROM TABLE(v_List) t;

  if pi_exec_submission_id is null and pi_exec_regnumber is null then
    INTEGRATION_PUBLIC_PCK.Create_Executive_Object(pi_SubmissionID
=> v_doc_id,
                                                   pi_Annotation          =>
'Testa virtuālā lieta no DAM adaptera dokumentu
atvienošanai'||systimestamp,
                                                   pi_InitDivisionID      =>
1,
```

```
                                                    pi_InitEmployeeID      =>
v_session.slodze_id,
                                                    pi_RegDivisionID       =>
1,
                                                    pi_RegEmployeeID       =>
v_session.slodze_id,
                                                    pio_ExecObjectID       =>
v_vl_doc_id,
                                                    pio_ExecObjectNumber   =>
v_vl_doc_nr,
                                                    pi_ExecObjectType      =>
1,
                                                    pi_SubmissionDocTypeId =>
v_doctype_id);
  else
    v_vl_doc_id := pi_exec_submission_id;
    v_vl_doc_nr := pi_exec_regnumber;
  end if;

  INTEGRATION_PUBLIC_PCK.Remove_Executive_Object(pi_SubmissionID        =>
v_doc_id,
                                                 pi_ExecObjectID        =>
v_vl_doc_id,
                                                 pi_SubmissionDocTypeID =>
v_doctype_id);
  -- ja nav nekādu kļūdu pēc dzēšanas, tad atgriež TRUE
  dbms_output.put_line('Dokuments '||v_doc_id||' ('||v_doc_nr||') atvienots
no administratīvās lietas');
  return true;

exception
  when others then
    dbms_output.put_line(dbms_utility.format_error_stack || chr(10) ||
DBMS_UTILITY.FORMAT_ERROR_BACKTRACE);
    return false;
end RemoveExecutiveObject;

-- Pārbauda dokumenta atvienošanu no administratīvās lietas RDLIS
function RemoveExecutiveObject_rdlis return boolean is
begin
  return RemoveExecutiveObject(pi_switch_value =>
const_pck.c_switch_rdlis);
end RemoveExecutiveObject_rdlis;

-- Pārbauda dokumenta atvienošanu no administratīvās lietas DAM
function RemoveExecutiveObject_dam return boolean is
begin
  return RemoveExecutiveObject(pi_switch_value => const_pck.c_switch_dam);
end RemoveExecutiveObject_dam;
```

Kvalifikācijas darbs „***DAM adapteris***” 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: ***Oļģerts Grudulis*** _____ **___.01.2022**.

Rekomendēju darbu aizstāvēšanai

Darba vadītāja: ***Mg. dat. Dace Jansone*** _____ **___.01.2022.**

Recenzents: ***Filips Jeļisejevs***

Darbs iesniegts 10.01.2022.

Kvalifikācijas darbu pārbaudījumu komisijas sekretārs: _____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē

___.01.2022. prot. Nr. _____

Komisijas sekretārs(-e): _____