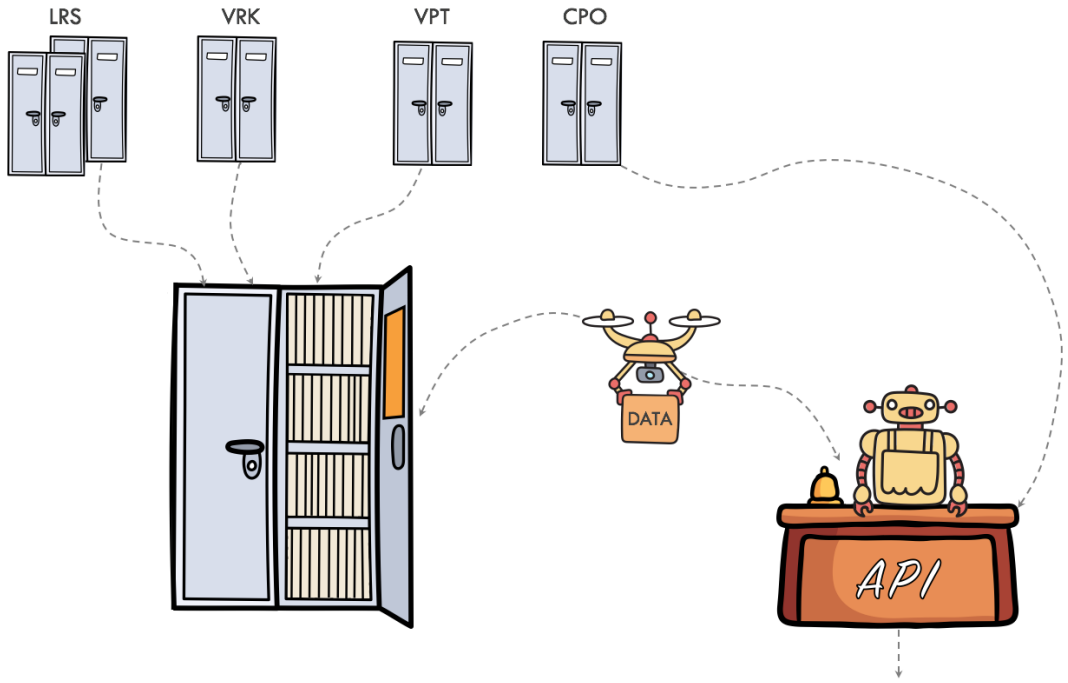
Duomenų atvėrimo vadovas

Lietuvos atvirų duomenų naudotojų bendruomenė

2023-09-23

Vadovas

Šis išsamus duomenų atvėrimo vadovas skirtas organizacijoms atveriančioms duomenis ir atvertų duomenų naudotojams.



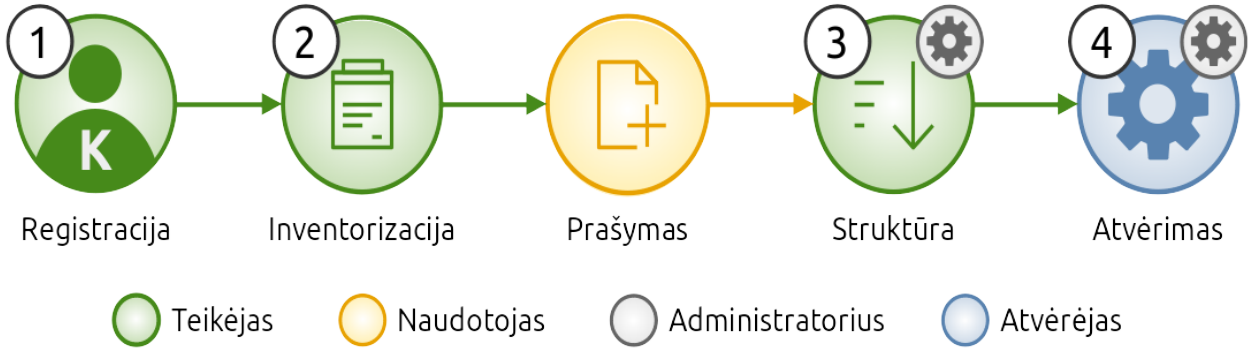
Dokumentacija sudaryta iš šių esminių dalių:

- Informacija **duomenų teikėjams** apie *duomenų atvėrimo*, atvertų duomenų *brandos lygio kėlimo* ir *asmens duomenų tvarkymo* procesą.
- Informacija **duomenų naudotojams**, apie tai, kaip *teikti pageidavimus ir pastabas* dėl duomenų ir kaip *gauti pačius duomenis*.
- *Duomenų struktūros aprašo specifikacija*, kurioje rasite detalią informaciją apie tai kaip rašyti ir skaityti *DSA* lenteles.
- Informacija diegėjams apie tai, kaip diegti ir konfigūruoti *priemones* skirtas darbui su duomenimis ir *DSA* lentelėmis.

SKYRIUS 1

Duomenų teikėjams

Labai apibendrintai, kiekviena įstaiga atverianti duomenis turi atlikti šiuos žingsnius:



žymi žingsnius, kuriems reikalinga IT ekspertų pagalba.

1. Paskirti atvirų duomenų koordinatorių ir užsiregistruoti Portale.
2. Sudaryti *duomenų rinkinių sąrašą* ir publikuoti jį *Portale*.
3. Parengti *duomenų struktūros aprašus* ir paskelbti juos *Portale*.
4. *Atverti duomenis* savarankiškai, su rangovo arba Vyriausybės įgaliotos institucijos pagalba.

Visas duomenų atvėrimo procesas duomenų teikėjams yra kiek įmanoma supaprastintas ir iš esmės susivedantis į duomenų rinkinių sąrašo ir struktūros aprašų parengimą. Visos kitos techninės duomenų nuskaitymo, transformavimo, atvėrimo ir publikavimo veiklos yra arba automatizuotos arba jas atlieka paskirtos atsakingos įstaigos.

Duomenų teikėjams, atveriant duomenis pagalbą teikia šios įstaigos:

Informacinės visuomenės plėtros komitetas (IVPK)

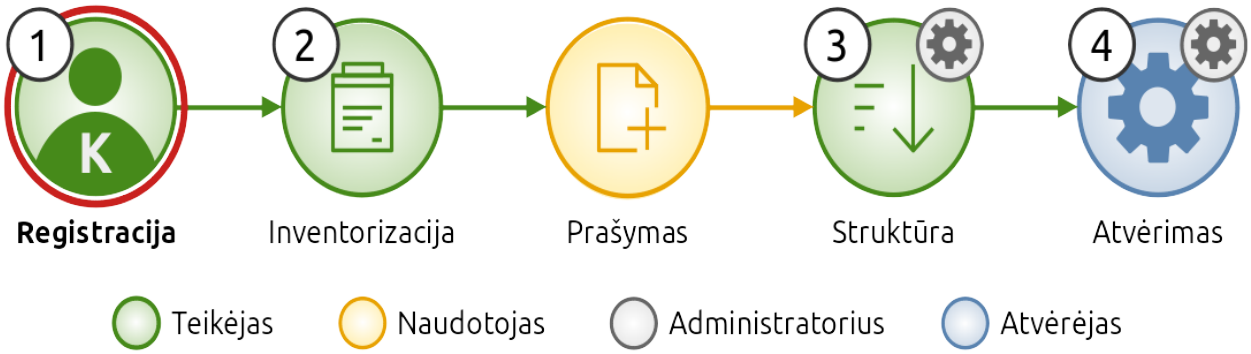
- Prižiūri visą šalies duomenų atvėrimo procesą ir konsultuoja duomenų teikėjus visais klausimais susijusiais su duomenų atvėrimu.
- Prižiūri atvirų duomenų *Katalogą*, kuriame vienoje vietoje skelbiami visi atverti ar planuojami atverti duomenys ir vieno langelio principu priimami prašymai gauti duomenis iš duomenų naudotojų.

- Prižiūri atvirų *duomenų publikavimo Saugyklą*, kurioje duomenys publikuojami laikantis geriausių duomenų teikimo praktikų, suteikiama galimybė duomenis publikuoti aukščiausiu brandos lygiu.
- Kuria ir palaiko *technines priemones* leidžiančias duomenis atverti automatizuotu būdu. Šios priemonės yra laisvai ir nemokamai teikiamos visiems duomenų teikėjams atveriantiems duomenis.
- Vykdo *standartų, protokolų ir techninių dokumentacijų priežiūrą*, kad duomenų atvėrimo procese dalyvaujančios šalys ir naudojami komponentai galėtų susišnekėti tarpusavyje.
- Rengia mokymus ir mokomąją medžiagą tiek atvirų duomenų teikėjams apie duomenų atvėrimą, tiek naudotojams apie atvertų duomenų naudojimą.

Vyriausybės įgaliota institucija (Statistikos departamentas)

- Teikia duomenų atvėrimo paslaugą, naudojant automatizuotas duomenų atvėrimo priemones.

1.1 Koordinatoriaus registravimas



Kiekviena duomenis atverianti įstaiga pirmiausia turi paskirti vieną žmogų atsakingą už duomenų atvėrimo koordinavimą. Šis žmogus bus atsakingas už įstaigos duomenų atvėrimo organizavimą, atsakys į duomenų naudotojų paklausimus pateiktus per atvirų duomenų portalą, išsiaiškins kokius duomenis įstaiga valdo ir kas atsakingas už jų priežiūrą.

Koordinatorius darbui su atvirais duomenimis gali pasitelkti tvarkytojus, kurie būtų atsakingi už atskirų duomenų rinkinių tvarkymą.

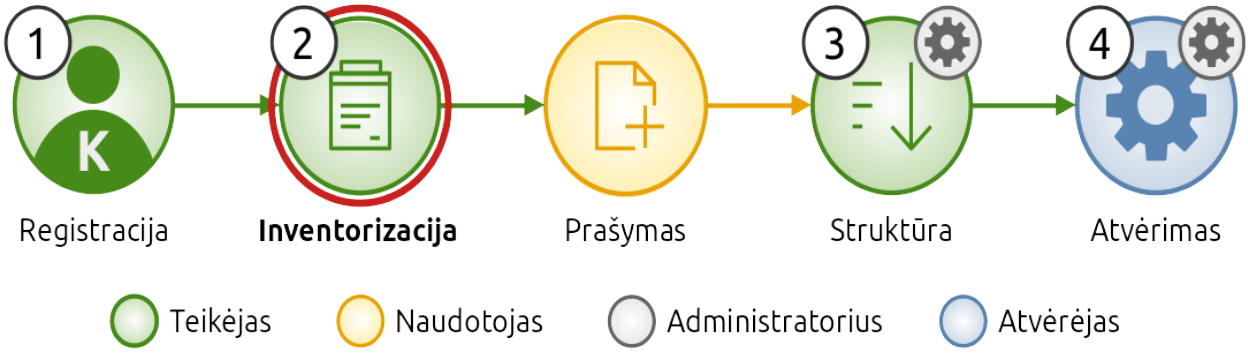
Atvirų duomenų koordinatorius neprivalo turėti techninių duomenų valdymo kompetencijų, tačiau tokių kompetencijų turėjimas būtų privalumas.

Koordinatoriaus paskyrimas įteisinamas į atvirų duomenų portalą pateikianti įstaigos vadovo pasirašytą raštą.

Turint įstaigos vadovo pasirašytą raštą, paskirtasis koordinatorius registruojasi atvirų duomenų portale.

Koordinatorius registruoja valstybės įstaigos ir jų valdomos įmonės. Savo koordinatorius gali registruoti ir privataus sektoriaus atstovai, jei publikuoja atvirus duomenis ir nori, kad jie būtų randami Lietuvos ir Europos atvirų duomenų portaluose.

1.2 Inventorizacija



Metaduomenų rengimas susideda iš dviejų dalių:

- *Duomenų rinkinių sąrašo sudarymas*,
- *Duomenų struktūros aprašų parengimas*.

Duomenų struktūros aprašų parengimas yra pati sudėtingiausia dalis, todėl rekomenduojama pirmiausiai susidaryti rinkinių sąrašus, o po to esant realiam duomenų poreikiui, pereiti prie duomenų struktūros aprašų.

Įstaigos paskirtas koordinatorius apžvelgia įstaigos veiklos nuostatus, valdomas informacines sistema, registrus, jau atvertus duomenis ir sudaro įstaigos valdomų *duomenų rinkinių* sąrašą.

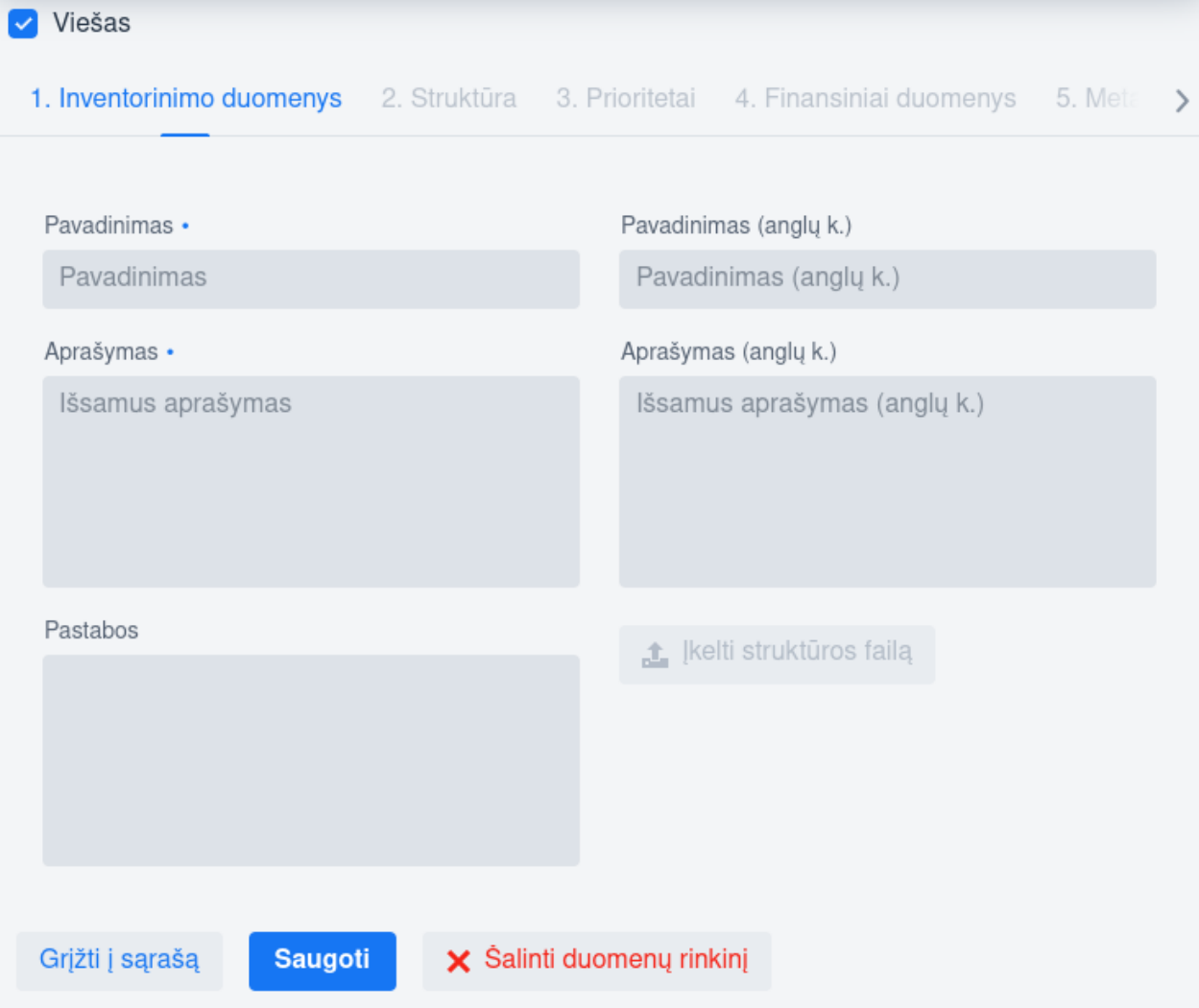
Šiame etape svarbiausiai gerai suprasti kas yra duomenų rinkinys ir distribucija.

Duomenų rinkinys yra grupė duomenų reikalingų tam tikrai organizacijos veiklai vykdyti. Duomenų rinkinys apibrėžia duomenų autorystę ir veiklos pobūdį kurioje naudojami duomenys.

Distribucija yra fizinė duomenų rinkinio išraiška, pavyzdžiui duomenų bazė, skaičiuoklės lentelė, katalogas, kuriame laikomi dokumentai ir pan.

Kadangi organizacijų veikloms reikalingi duomenys dažniausiai saugomi tam tikroje vietoje, tai sudarant duomenų rinkinių sąrašą paprasčiausia apžvelgti resursus naudojamus duomenų saugojimui ir pagal tai įvardinti duomenų rinkinius.

Atvirų duomenų portale, naujas duomenų rinkinys registruojamas užpildžius šią formą:

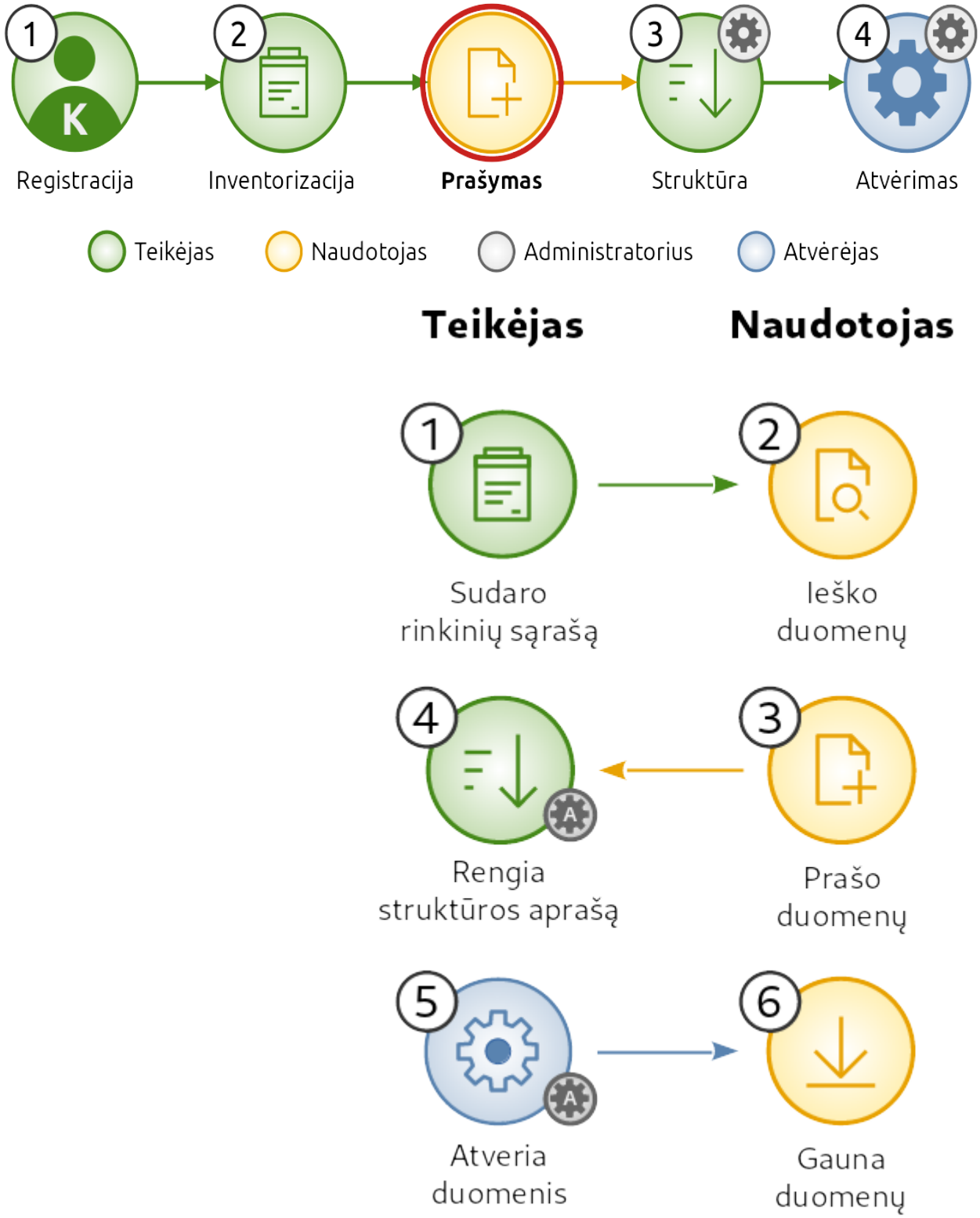


Rekomenduotina rinkinių sąrašus sudaryti tiesiogiai *atvirų duomenų portale*, tačiau yra galimybė parengti rinkinių sąrašo lentelę ir ją vėliau importuoti į *portalą*.

Sudarant rinkinių sąrašus, reikėtų vadovautis principu, kad visi duomenys, kuriems nėra taikomi apribojimai yra atviri.

Inventorizacijos metu, pateikiami dalis tik metaduomenų, kurie yra reikalingi, kad duomenų rinkinius būtų galima surasti atvirų duomenų portale. Kita metaduomenų dalis susijusi su atvertų duomenų periodiškumu, licencija ir naudojimo sąlygomis pateikiama po to, kai yra priimtas sprendimas atverti duomenis.

1.3 Prašymai gauti duomenis



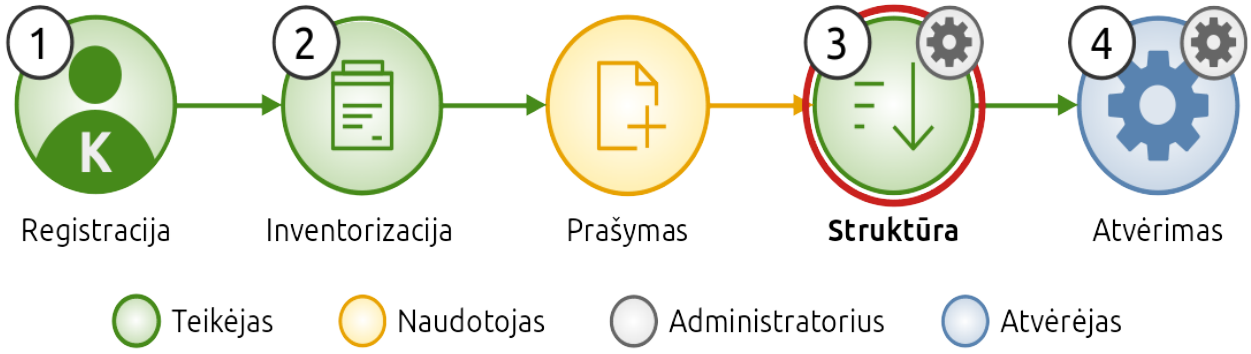
Paskelbus rinkinių sąrašus, kiti žingsniai daromi esant poreikiui atverti duomenis arba savo nuožiūra įvertinus duomenų

paklausos potencialą.

Paklausa grįstas duomenų atvėrimas reiškia, kad duomenų tiekėjai iš pradžių parengia atvertinų duomenų rinkinių sąrašus (1), kad juos galėtų rasti duomenų naudotojai (2). Realus duomenų atvėrimo procesas pradedamas tuomet, kai duomenų naudotojai, rade jiems reikalingus duomenų rinkinius pateikia prašymą (3) juos gauti. Teikėjai gavę prašymus atverti duomenis, pradeda duomenų atvėrimo procesą parengdami duomenų struktūros aprašus (4), kurių pagrindu vykdomas duomenų atvėrimas (5). Galiausiai, atvėrus duomenis, naudotojai gauna tai, ko prašė (6).

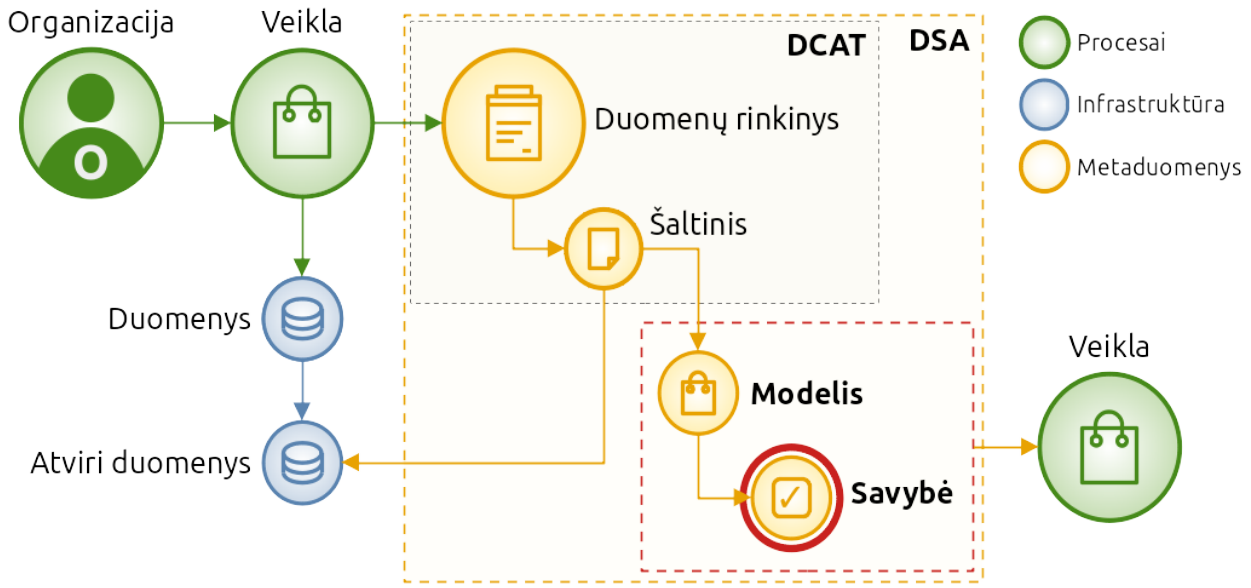
Duomenų teikėjai, savo nuožiūra, įvertinę duomenų paklausos potencialą gali pradėti duomenų atvėrimą ir negavę prašymo.

1.4 Duomenų struktūros aprašas



Duomenų struktūros aprašas rengiamas tada, kai atsiranda prašymas atverti duomenis arba savo nuožiūra įvertinus duomenų paklausos potencialą.

1.4.1 Kas yra duomenų struktūros aprašas?



Duomenų struktūros apraše pateikiama duomenų struktūros išklotinė išvardinant visus duomenų laukus, kurie bus atverti.

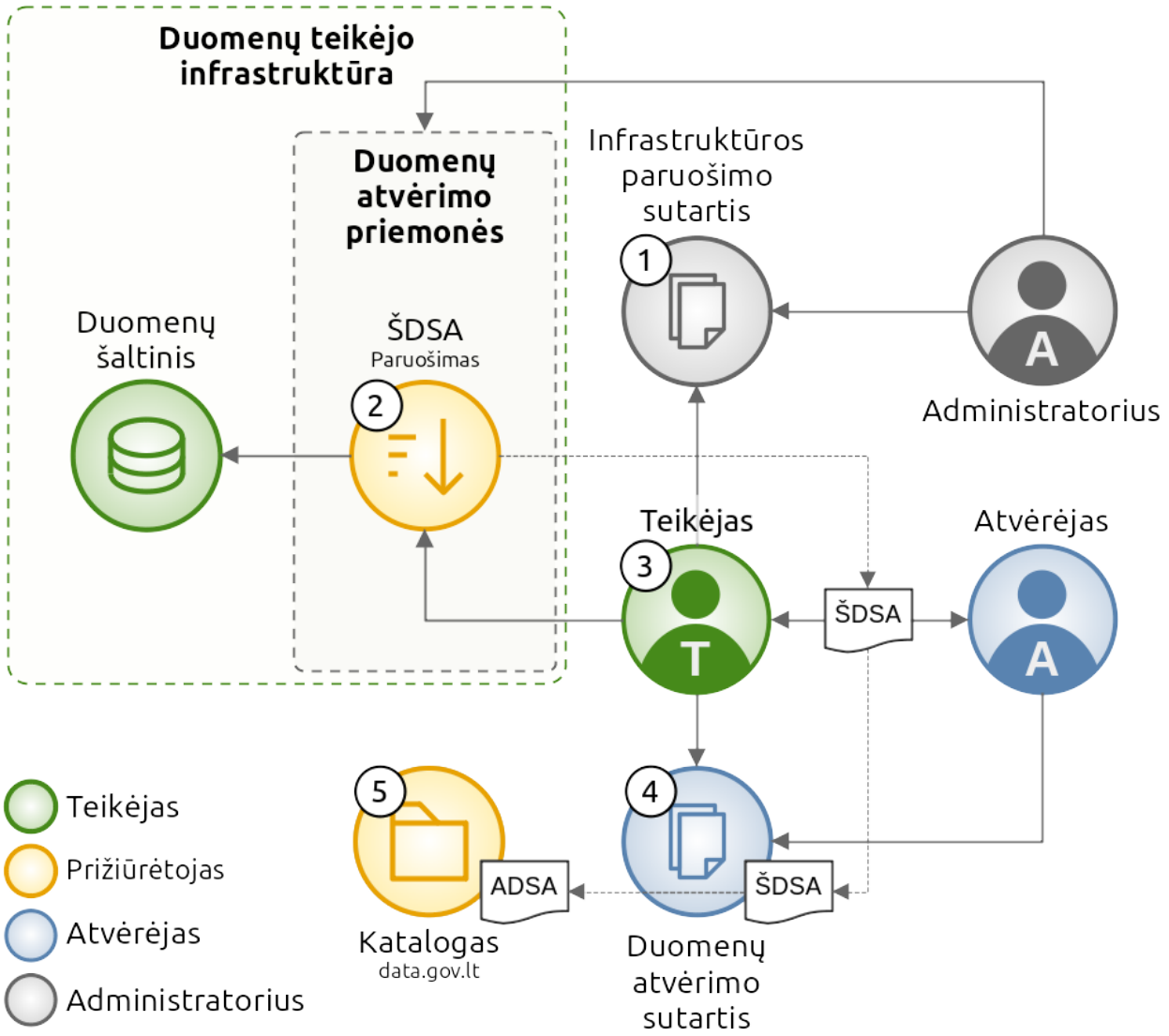
Duomenų struktūros apraše pateikiama pilna duomenų laukų išklotinė.

Duomenų laukai yra skirstomi į modelius. **Modelio** ir **savybės** tiksli prasmė priklauso nuo aprašomo duomenų šaltinio:

Šaltinis	Modelis	Savybė
SQL	Lentelė	Stulpelis
CSV	Lentelė	Stulpelis
XLSX	Lentelė	Stulpelis
JSON	Masyvas	Atributas
XML	Masyvas	Atributas
RDF	Klasė	Savybė

Duomenų struktūros apraše galima aprašyti įvairių duomenų šaltinių turinį vieningu sutartiniu būdu.

1.4.2 Iš kur gauti ŠDSA?



1. *Infrastruktūros paruošimo sutartis*

2. *ŠDSA paruošimas*
3. *ŠDSA suderinimas atvėrimui*
4. *Duomenų atvėrimo sutartis*
5. *ADSA publikavimas*

Infrastruktūros paruošimo sutartis

Atliekant duomenų atvėrimo darbus gali būti reikalingos dvi sutartys. viena yra reikalinga infrastruktūros paruošimui, o kita *duomenų atvėrimui*.

Infrastruktūros parengimo procedūroms dažnai reikalinga pilna prieiga prie duomenų šaltinio ir administratoriaus teisės kompiuteryje ar serveryje, kuriame bus diegiamos duomenų atvėrimo priemonės.

Tuo tarpu duomenų atvėrimo sutartis nereikalauja jokios prieigos prie duomenų šaltinio.

Infrastruktūros paruošimo sutartis yra reikalinga tik tuo atveju, jei Teikėjas neturi jokios duomenų šaltinio priežiūros sutarties ir negali infrastruktūros paruošimo darbų atlikti savarankiškai.

Infrastruktūros paruošimo darbus gali atlikti ir Atvėrėjas.

Apibendrinant, Administratorius atsakingas už infrastruktūros paruošimą gali būti:

- pats Teikėjas, jei turi reikalingas technines kompetencijas,
- esamas Administratorius atliekantis duomenų šaltinio priežiūros ir palaikymo paslaugas,
- Administratorius su kuriuo atskirai sudaroma sutartis tik infrastruktūros duomenų atvėrimui paruošimo prie priežiūros paslaugoms,
- Atvėrėjas, sudarius atskirą sutartį gali teikti Administratoriaus paslaugas.

Infrastruktūros paruošimui reikia atlikti šiuos darbus:

- Kompiuterio ar serverio, kuriame bus diegiamos duomenų atvėrimo priemonės, diegimas ir konfigūravimas.
- Prieigos prie duomenų šaltinio konfigūravimas.
- Priemonių reikalingų duomenų atvėrimui diegimas ir konfigūravimas.
- Nestandartinių duomenų šaltinių transformavimas į standartinius formatus.
- Šaltinio duomenų struktūros aprašo generavimas arba parengimas.
- Duomenų atvėrimo priemonių stebėsena ir palaikymas, atsiradusių trikdžių šalinimas.

Administratorius turėtu naudoti standartines, prižiūrinčios institucijos prižiūrimas duomenų atvėrimo priemones.

Reikalavimai infrastruktūrai

Reikalavimai infrastruktūrai priklauso nuo to, koks variantas labiausiai tinka konkretaus Teikėjo atvejui. Toliau aptarsime kelis galimus variantus, tačiau variantų gali būti ir daugiau.

Atskira virtuali mašina

Šis variantas yra rekomenduojamas.

Teikėjo infrastruktūroje reikia paleisti naują virtualią mašiną, kurioje būtų įdiegta Linux operacinė sistema ir suteikta prieiga prie duomenų šaltinio.

Rekomenduojame naudoti Red Hat, Ubuntu arba Debian Linux distribucijas.

Jei duomenys pateikiami duomenų failuose, tuomet katalogas, kuriame yra saugomi failai, prie virtualios mašinos prijungimas NFS arba SMBFS pagalba. Arba galima failus periodiškai kopijuoti į virtualią mašiną rsync pagalba.

Jei duomenų atvėrimui naudosite VDV IS duomenų jungtį, tuomet, virtualioje mašinoje reikia panašiai tiek pat vietos, kiek užima visi atvėrimui reikalingi duomenys, kadangi VDV IS duomenų jungtis, prieš perduodant duomenis, pasidaro perduodamų duomenų kopiją.

Jei duomenų atvėrimui naudosite standartinę priemonę *Spinta*, tuomet duomenų perdavimui reikalinga tiek vietos, kiek užima visų atveriamų duomenų identifikatoriai. Kiek tiksliai identifikatoriams reikės vietos labai priklauso nuo duomenų šaltinio duomenų.

ŠDSA paruošimas

Administratorius, naudodamasis Prižiūrinčios institucijos patvirtintomis priemonėmis, parengiam duomenų šaltinio struktūros aprašą (ŠDSA).

ŠDSA yra lentelė sudaryta iš 15 stulpelių, kurioje pateikiamas pilnas duomenų šaltinyje esančių duomenų laukų sąrašas su duomenų tipais, ryšiais tarp duomenų objektų ir aprašymais.

Tokią lentelę daugeliu atveju galima sugeneruoti automatiškai naudojant standartines priemones, jei duomenų šaltinis palaikomas. Jei standartinės priemonės duomenų šaltinio nepalaiko, tuomet, Administratorius parengia ŠDSA savaranki6kai.

Tokį pradinį ŠDSA variantą Administratorius perduoda Teikėjui.

Po tam tikro laiko, kai duomenų šaltinio struktūra keičiasi, reikia atnaujinti ir ŠDSA, tačiau išlaikant visus keitimus, kuriuos yra padaręs Teikėjas. Atnaujinant ŠDSA reikia užtikrinti, kad duomenų struktūra, kuri jau buvo publikuota, išliktų nepakitusi. Turi būti užtikrinamas publikuotos duomenų struktūros stabilumas.

Tam tikra apimtimi standartinės priemonės užtikrina ŠDSA atnaujinimą, tačiau sudėtingesniais struktūros pasikeitimo atvejais, gali tekti sugeneruoti naują ŠDSA variantą ir lyginant su anksčiau generuoti ir taisytu variantu palyginti ir atnaujinti rankiniu būdu.

Šaltinio duomenų struktūros aprašas gali būti generuojamas įvairiais būdais, kelis iš jų aptarsime sekančiuose skyreliuose.

Tiesioginis generavimas

Tiesioginis generavimas iš duomenų šaltinio reikalauja tiesioginės prieigos prie duomenų šaltinio. ŠDSA generavimo priemonė jungiasi prie duomenų šaltinio, nuskaito duomenų šaltinio struktūrą ir generuoja ŠDSA.

Šiuo atveju, generavimo priemonė turi turėti pilną prieigą prie duomenų šaltinio ir įprastiniu atveju ją turėtu leisti Administratorius, kuris yra sudaręs infrastruktūros paruošimo sutartį su Teikėju. Generavimas turėtu vykti Teikėjo infrastruktūroje.

Generavimas iš schemos

Jei duomenų šaltinis tai palaiko, galima eksportuoti duomenų šaltinio schemą ir ją perduoti Atvėrėjui, kuris iš schemos parengs ŠDSA.

Šaltinio schema gali būti pateikta SQL DDL ar kitu formatu, kurį palaiko standartinės priemonės.

Šiuo atveju, nereikia diegti jokių papildomų priemonių, tačiau reikalinga Rangovo pagalba eksportuojant duomenų šaltinio schema.

Rankinis paruošimas

Tam tikrais atvejais, kai duomenų šaltinis yra labai nedidelės apimties arba duomenų brandos lygis yra labai žemas, šaltinio duomenų struktūros aprašą galima parengti ir rankiniu būdu, užpildant ŠDSA lentelę.

ŠDSA suderinimas atvėrimui

Turinti paruoštą pradinį ŠDSA variantą, Teikėjas savarankiškai, su Atvėrėjo pagalba parengia ŠDSA atvėrimui.

Ruošiant ŠDSA atvėrimui, nurodoma kurie duomenų laukai bus atveriami, nurodomi filtrai, jei duomenys atveriami ne pilna apimtimi, sutvarkomi kodiniai pavadinimai, kad atitiktų atveriamiems duomenis keliamus reikalavimus, pateikiami trūkstami metaduomenys. Plačiau apie ŠDSA paruošimą atvėrimui skaitykite skyriuje *Duomenų struktūros aprašas*.

Duomenų atvėrimo sutartis

Atvėrimui paruoštas ŠDSA variantas teikiamas derinimui Atvėrėjui. Atvėrėjas patikrina ar ŠDSA paruoštas tinkamai ir informuoja Teikėją apie aptiktas klaidas.

Pasirašant duomenų atvėrimo sutartį, suderintas ŠDSA variantas pateikiamas, kaip sutarties priedas.

Pasirašius sutartį, Teikėjas perduoda Atvėrėjui Katalogo API raktą, kad Atvėrėjas galėtų automatiškai atnaujinti atveriamo duomenų rinkinio metaduomenis.

ADSA publikavimas

Atvėrėjas ŠDSA pagrindu generuoja ADSA variantą, kuriame pašalinami visi atveriamo duomenų šaltinio metaduomenys ir paliekama tik ta dalis, kuri skirta publikavimui. Atvėrėjas publikuoja ADSA Kataloge per *Katalogo partnerių API*.

Publikavus ADSA Kataloge, ADSA taip pat perduodamas ir į atvirų duomenų Saugyklą, ko pasekoje Saugykla paruošiama duomenų priėmimui, kurie atitinka ADSA pateiktus metaduomenis.

Kataloge užtikrinama, kad įkeltas ADSA neturi struktūros pakeitimų, kurie nėra suderinami su prie6 tai publikuota ADSA versija, atlieka pilną metaduomenų patikrinimą.

1.4.3 Kaip pildyti ŠDSA?

Duomenų struktūros aprašo rengimas susideda iš tokių žingsnių:

1. Duomenų šaltinio administratorius pateikia šaltinio *duomenų struktūros išklotinę (ŠDSA)*.
2. Duomenų srities ekspertai su duomenų šaltinio administratoriaus pagalba pateikia trūkstamus metaduomenis duomenų struktūros aprašo lentelėje.

Jei pirminio duomenų struktūros aprašo varianto sugeneruoti iš duomenų šaltinio neįmanoma, pavyzdžiui, jei duomenys yra labai žemo brandos lygio, tuomet duomenų struktūros aprašas pildomas nuo nulio naudojant `aprašo lentelės šabloną`.

Duomenų struktūros aprašas yra lentelė susidedanti iš 15 stulpelių, kuriuose aprašoma duomenų struktūra. Tarkime, turint tokius duomenis:

ŠALIS			
ID	KODAS	ŽEMYNAS	ŠALIS
1	lt	eu	Lietuva
2	lv	eu	Latvija
3	ee	eu	Estija

Duomenų struktūra aukšiau pateiktiems duomenims atrodys taip:

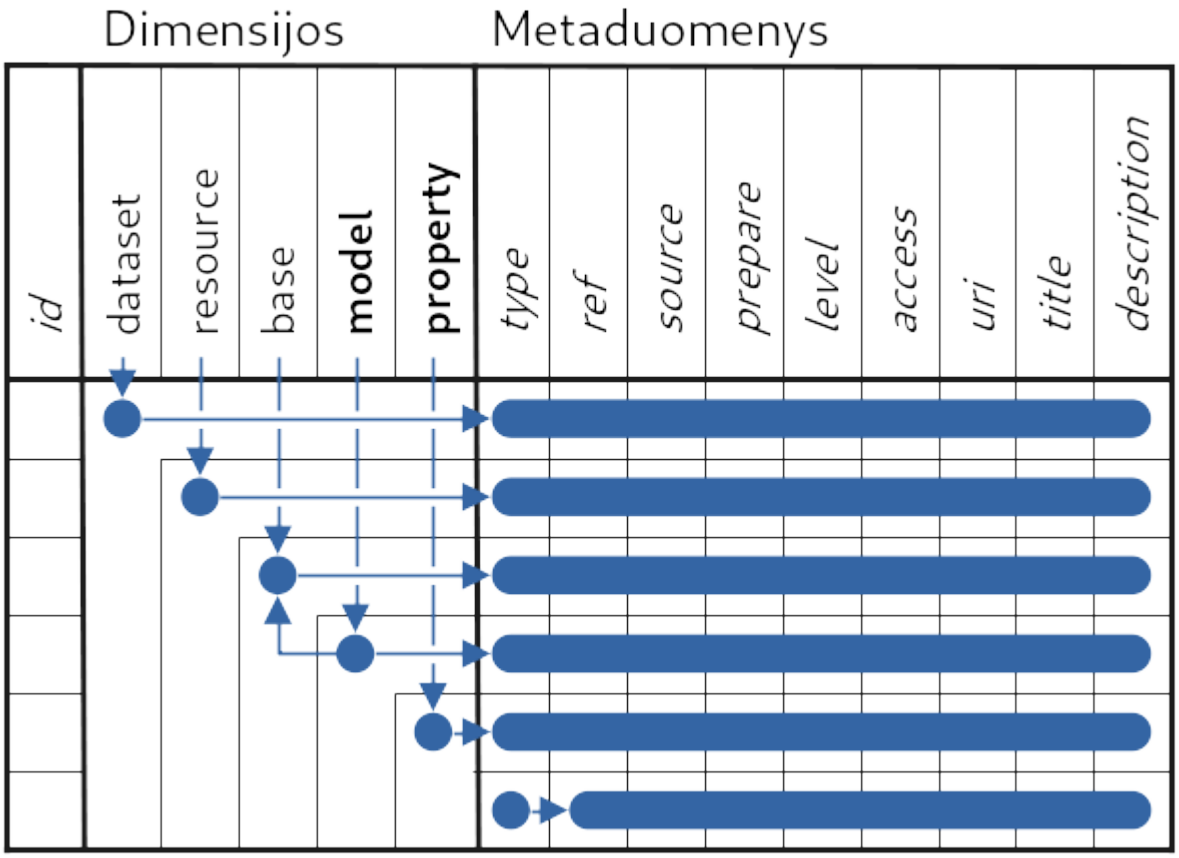
1 Table : Duomenų struktūros aprašas

id	d	r	b	m	property	type	ref	source	prepare	level	access	uri	title	description
	datasets/example/countries													
		salys				sql		sqlite://						
				Country			id	ŠALIS	continent="eu"					
					id	integer		ID		4	private			
					code	string		KODAS		2	open			
					continent	string		ŽEMYNAS		2	private			
					name	string		ŠALIS		2	open			

Pastaba: Siekiant padidinti duomenų struktūros aprašo lentelės skaitomumą, kai kurie stulpelių pavadinimai yra sutrumpinti:

d - dataset - duomenų rinkinio kodinis pavadinimas.
r - resource - duomenų šaltinio kodinis pavadinimas.
b - base - modelio bazės kodinis pavadinimas.
m - model - modelio kodinis pavadinimas.

Duomenų struktūros aprašo lentelė susideda iš *5 dimensijų* (dataset, resource, base, model, property) ir *9 metaduomenų stulpelių*, kurių prasmė priklauso nuo vienos iš 5 dimensijų.



Plačiau apie tai, ką reiškia kiekvienas stulpelis galite skaityti skyriuje *Lentelės struktūra*.

ŠDSA lentelėje reikia pateikti tokius duomenis:

id	dataset	resource	base	model	property	type	ref	source	prepare	level	access	uri	title	description
A	1													
A		2				2		2						
A														
A				A			A	A	5			3	7	7
A					A	A	A	A		6	4	3	7	7

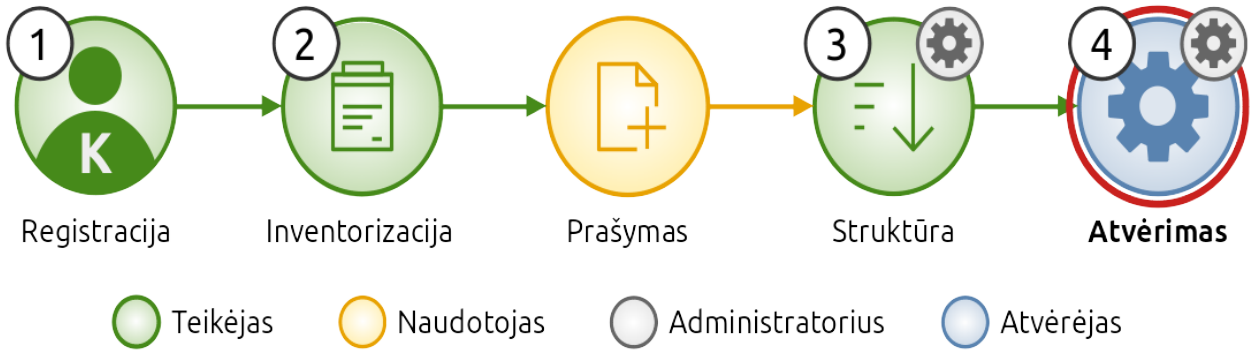
A Pildo kompiuteris ○ Pildo žmogus

1. *Duomenų rinkiniui* suteikti *kodinį pavadinimą*.
2. Pateikti duomenų šaltinio pavadinimą, *tipą ir adresą*.
3. Užpildyti `uri` stulpelį, nurodant kuriose vietose yra *asmens duomenys*.
4. Užpildyti `property.access`, nurodant duomenų *prieigos lygį*.
5. Užpildyti `model.prepare`, jei duomenys atveriami ne pilna apimtimi ir reikia juos *filtruoti*.
6. `property.level` stulpelyje nurodyti esamą duomenų laukų *brandos lygį*.
7. Užpildyti `title` ir `description` stulpelius pateikiant `model` ir `property` pavadinimus ir aprašymus.

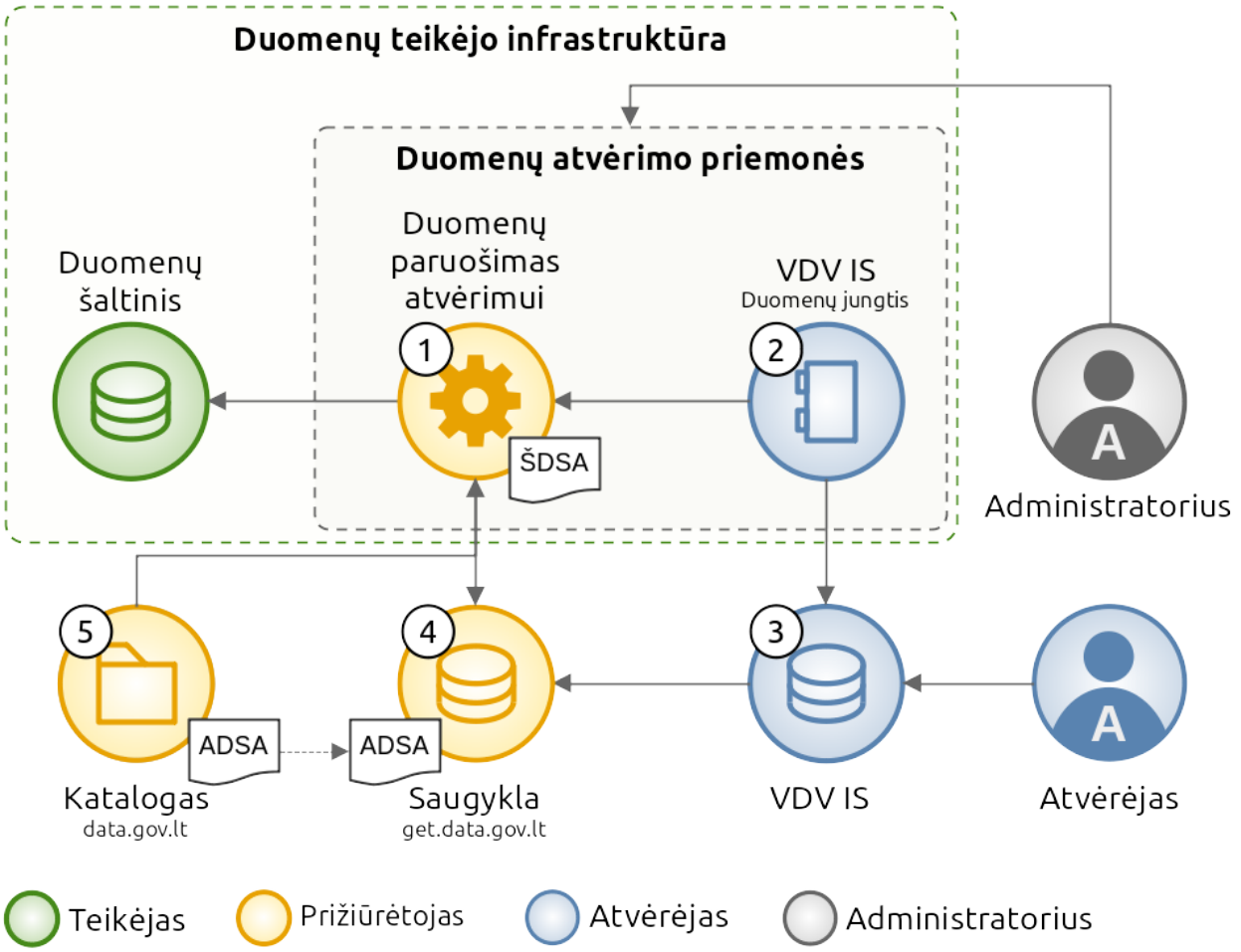
Galiausiai, toks duomenų struktūros aprašas gali būti naudojamas *automatizuotam duomenų atvėrimui ir publikavimui* arba naudojamas kaip sutarties priedas, jei įstaiga duomenis atveria su rangovo ar Vyriausybės paskirtos įstaigos pagalba.

Jei įstaiga jau yra atvėrusi duomenis ir juos publikuoja savo infrastruktūroje, tuomet į atvirų duomenų portalą turi būti įkeliamas, ne *ADSA*, o *ŠDSA*, kuriame aprašyti įstaigos infrastruktūroje publikuojami duomenys.

1.5 Duomenų atvėrimas



Duomenų atvėrimo žingsnį atlieka ne Teikėjas, o Atvėrėjas - Vyriausybės įgaliota įstaiga, kuri atsakinga už duomenų atvėrimą.



Sekantys žingsniai yra *anksčiau aptartų žingsnių* tęsinys.

1. *Duomenų paruošimas atvėrimui*
2. *Duomenų perdavimas atvėrimui*
3. *Duomenų kokybės ir konfidencialumo užtikrinimas*
4. *Duomenų publikavimas*
5. *Metaduomenų atnaujinimas*

Priklausomai nuo situacijos, ne visi šie žingsniai yra privalomi, detaliau apie išimtis yra paaiškinta prie kiekvieno žingsnio.

1.5.1 Duomenų paruošimas atvėrimui

Administratorius sudaręs infrastruktūros paruošimo sutartį, Duomenų teikėjo infrastruktūroje įdiegia ir sukonfigūruoja duomenų paruošimo atvėrimui priemones.

Atvėrimas per VDV IS

Jei duomenys atveriami per VDV IS, tuomet duomenų paruošimas reikalingas tik tais atvejais, kai VDV IS duomenų jungtis nepalaiko duomenų šaltinio.

Jei VDV IS duomenų jungtis nepalaiko duomenų formato, tuomet duomenys transformuojami ir pateikiami VDV IS duomenų jungčiai naudojant standartines priemones.

Atvėrimas naudojant standartines priemones

Jei duomenys atveriame ne per VDV IS, tuomet galima naudoti Standartines priemones, kurių pagalba duomenys bus perduodami tiesiogiai į atvirų duomenų Saugyklą.

`resource.type` skyriuje galite rasti pilną palaikomų formatų sąrašą.

Pastaba: Net jei duomenų struktūros apraše formatas yra palaikomas, tačiau standartinės priemonės, gali nepalaikyti visų specifikacijoje aprašytų formatų.

Jei Standartinės priemonės nepalaiko Duomenų šaltinio formato, tuomet Administratorius turi transformuoti duomenis į tokį formatą, kurį palaiko Standartinės priemonės.

Nepalaikomų duomenų formatų paruošimas

Jei standartinės priemonės nepalaiko duomenų šaltinio formato, tuomet duomenų paruošimui reikia naudoti konkrečiam duomenų šaltiniui pritaikytą specializuotą sprendimą.

Duomenys turi būti transformuojami į vieną iš formatų, kurį palaiko standartinės priemonės.

Atvėrimas savo infrastruktūroje

Jei duomenys publikuojami Duomenų teikėjo infrastruktūroje, tuomet Atvirų duomenų Kataloge, kiekvieną kartą atnaujinus publikuojamus duomenis, reikia atnaujinti ir *publikuojamų duomenų metaduomenis*.

1.5.2 Duomenų perdavimas atvėrimui

Atveriant duomenis per VDV IS, Administratorius, Teikėjo infrastruktūroje įdiegia ir sukonfigūruoja VDV IS duomenų jungtį. VDV IS duomenų jungtis kopijuoja duomenis į VDV IS, apibrėžtus duomenų atvėrimo sutartyje, duomenų struktūros priede.

Toks duomenų perdavimas vyksta nustatytu periodiškumu.

1.5.3 Duomenų kokybės ir konfidencialumo užtikrinimas

Perkėlus duomenis iš Teikėjo infrastruktūros į VDV IS, naudojantis VDV IS funkcionalumu užtikrinama duomenų kokybė, atliekamas nuasmeninimas ir užtikrinamas neatvertinų duomenų konfidencialumas.

Pilnai paruošti atvėrimui duomenys perduodami publikavimui į atvirų duomenų Saugyklą.

1.5.4 Duomenų publikavimas

Atvirų duomenų Saugykla iš Katalogo gauna ADSA, tokiu būdu Saugykla yra paruošiama duomenų priėmimui, kurie atitinka ADSA.

Saugykla, duomenis gali gauti vienu iš šių būdų:

- jei duomenys atveriami per VDV IS, tuomet duomenis į Saugyklą PUSH būdu perduoda VDV IS,
- jei duomenys į Saugyklą perduodami tiesiogiai, tuomet Saugykla duomenis priima tiesiogiai iš Duomenų teikėjo infrastruktūros PUSH būdu, tokiu būdu duomenys gali būdu perduodami realiu laiku,
- jei duomenys publikuojami Teikėjo infrastruktūroje arba Kataloge, tuomet Saugykla naudojantis ADSA metaduomenis išorėje publikuojamus duomenis importuoja PULL būdu, nurodytu periodiškumu.

Saugykla užtikrina, kad perduodami duomenys atitinka ADSA, duomenys atitinka nurodytus duomenų tipus.

Į Saugyklą patenkantys duomenys saugomi vidinėje bazėje iš kurios publikuojami per API, įvairiais formatais, realiu laiku, suteikiama galimybė atsisiųsti duomenis tiek po vieną įrašą, tiek visus vienu kartu.

1.5.5 Metaduomenų atnaujinimas

Pasirašius sutartį, Teikėjas perduoda Katalogo API raktą Atvėrėjui, kuris automatiškai per Katalogo Partnerių API atnaujina atvertų duomenų metaduomenis ir po kiekvieno duomenų atnaujinimo siunčia pranešimą per API į Katalogą apie tai, kad duomenys buvo sėkmingai atnaujinti.

Tokiu būdu Katalogas gali patikrinti, kad duomenys atnaujinami nustatytu periodiškumu.

Plačiau apie metaduomenų atnaujinimą galite skaityti *Metaduomenų atnaujinimas* skyriuje ir Katalogo Partnerių API dokumentacijoje.

1.5.6 Struktūros pasikeitimų valdymas

Siekiant užtikrinti atvertų duomenų kokybę, pasikeitimai duomenų struktūroje turi būti daromi aiškiai apibrėžtu ir kontroliuojamu būdu.

Atvirų duomenų naudotojai, užsiregistravę portale, turi galimybę gauti pranešimus apie planuojamus duomenų struktūros pasikeitimus. Prieš darant pakeitimus, kurie yra nesuderinami su anksčiau publikuota duomenų struktūra (toliau Nesuderinami pakeitimai), duomenų naudotojai turi būti informuojami prieš 6 mėnesius, kad turėtų pakankamai laiko atsinaujinti savo turimas integracijas.

Nesuderinami pakeitimai yra:

- anksčiau publikuoto duomenų lauko, modelio ar vardų erdvės pavadinimo keitimas,
- anksčiau publikuoto duomenų lauko, modelio ar vardų erdvės panaikinimas,
- anksčiau publikuoto duomenų lauko tipo keitimas,
- anksčiau publikuoto duomenų lauko reikšmių mastelio, vienetų ar formato keitimas, nekeičiant duomenų tipo.

Vis šie, su anksčiau publikuota struktūros versija nesuderinami pakeitimai, galo būti daromi tik įspėjus apie planuojamus pakeitimus, prieš šešis mėnesius. Pakeitimai gali būti atliekami praėjus 6 mėnesiams nuo įspėjimo.

Naujų vardų erdvių, modelių ar duomenų laukų įtraukimas yra su ankstesne struktūros versija suderinamas pakeitimas, nereikalaujantis išankstinio įspėjimo.

Duomenų naudotojų informavimas apie struktūros ar duomenų pasikeitimus darotas atvirų duomenų Kataloge, pateikiant naują duomenų struktūros versiją, kurioje pažymėta, kas keičiasi, nurodant datą 6 mėnesius į priekį. Įkėlus tokį duomenų struktūros aprašą, Kataloge užsiregistravę duomenų naudotojai bus automatiška informuojami apie planuojamus struktūros pasikeitimus, nurodant data į priekį, kada jie įsigalios.

Apie struktūros pasikeitimų žymėjimą skaitykite skyriuje *Struktūros keitimas*.

SKYRIUS 2

Duomenų naudotojams

ADK svetainėje duomenų naudotojai gali teikti pasiūlymus dėl jiems reikalingų duomenų. Galima apytiksliai įvardinti pageidaujamo duomenų rinkinio pavadinimą ir aprašymą, tačiau galima duomenų poreikį įvardinti labai tiksliai, poreikio formoje įkeliant *DSA* lentelę CSV formatu.

Iš jau publikuojamų *ADSA* lentelių, galima atsirinkti dominančius duomenis ir susirašyti juos į savo poreikio *DSA* lentelę.

Kaip pavyzdį galime panagrinėti skyrelyje *Duomenų teikėjams* naudotą *ADSA* lentelę, kuri atrodo taip:

1 Table : Planuojamų atverti duomenų struktūros aprašas (ADSA)

id	d	r	b	m	property	type	ref	source	prepare	level	access	uri	title	description
6	datasets/example/countries						1							
1		salys				sql								
2				Country			_id							
4					code	string				2	open			
5					name	string				2	open			

Tokią lentelę galima atsisiųsti ir atsidaryti su bet kuria skaičiuoklės programa ir ją papildyti trūkstamais metaduomenimis, kurie jums yra reikalingi.

Pavyzdžiui tokia papildyta poreikio *ADSA* lentelė gali atrodyti taip:

2 Table : Pageidaujamų duomenų struktūros aprašas (ADSA)

id	d	r	b	m	property	type	ref	source	prepare	level	access	uri	title	description
8						prefix	esco					http://data.europa.eu/esco/model#		
6	datasets/example/countries						1							
1		salys				sql								
2				Country			_id							
4					code	string				5	open	esco:isoCountryCodeA2		
5					name	string				2	open			
9						comment								Kokia kalba pateiktas šalies kodas?
6					continent	string				3	open			
7					population	string				3	open			

Šioje lentelėje, buvo pateikti tokie pageidavimai dėl duomenų:

- Pasiūlyta padidinti `code` savybės brandos lygį nuo 2 iki 5, pateikiant šios savybės sąsają su `esco:isoCountryCodeA2` išoriniu žodynu, kuriame įvardinta, kad šalies kodas turi atitikti ISO 3166 šalių kodų standartą.

  Papildomai buvo įtrauktas naujas URI prefiksas 8-oje eilutėje.

- Pateiktas paklausimas `name` savybei įtraukiant papildomą *`comment`* dimensiją, apie tai, kokia kalba pateikti šalių pavadinimai.

- Pasiūlyta įtraukti dvi naujas `country` savybes, `continent` ir `population`, nurodant, kad pageidaujamas šių savybių brandos lygis turėtu būti ne mažesnis, nei 3.

Duomenų naudotojai turi galimybę ne tik teikti pageidavimus, bet ir prisidėti prie atvertų duomenų brandos lygio kėlimo. Pageidavimo formoje įkelti *DSA* gali būti papildyti pakeitimais *didinančiais duomenų brandos lygį*.

SKYRIUS 3

Brandos lygio kėlimas

Brandos lygio kėlimas dažniausiai yra susijęs su metaduomenų tvarkymu, tačiau kartais tai gali būti susiję ir su pačių duomenų tvarkymu.

Atveriant duomenis, rekomenduojama pirmiausia publikuoti *ADSA*, o tada pradėti duomenų brandos lygio kėlimą, atsižvelgiant į gautus atsiliepimus iš duomenų naudotojų.

3.1 Nekaupiami duomenys

`level` = 0.

Trūkstami duomenys yra žemiausio brandos lygio, `level` stulpelyje žymimi skaičiumi `0`.

Toks nulinis brandos lygis suteikiamas duomenims, kurių dar nėra, bet yra poreikis juos turėti. Tokiu atveju, galima parengti neegzistuojančių, bet reikalingų duomenų aprašą, pažymint `level` skaičiumi `0`.

Gali būti, kad trūksta viso *duomenų rinkinio*, arba vieno *duomenų modelio* arba vienos *savybės*.

3.2 Duomenys be aiškios struktūros

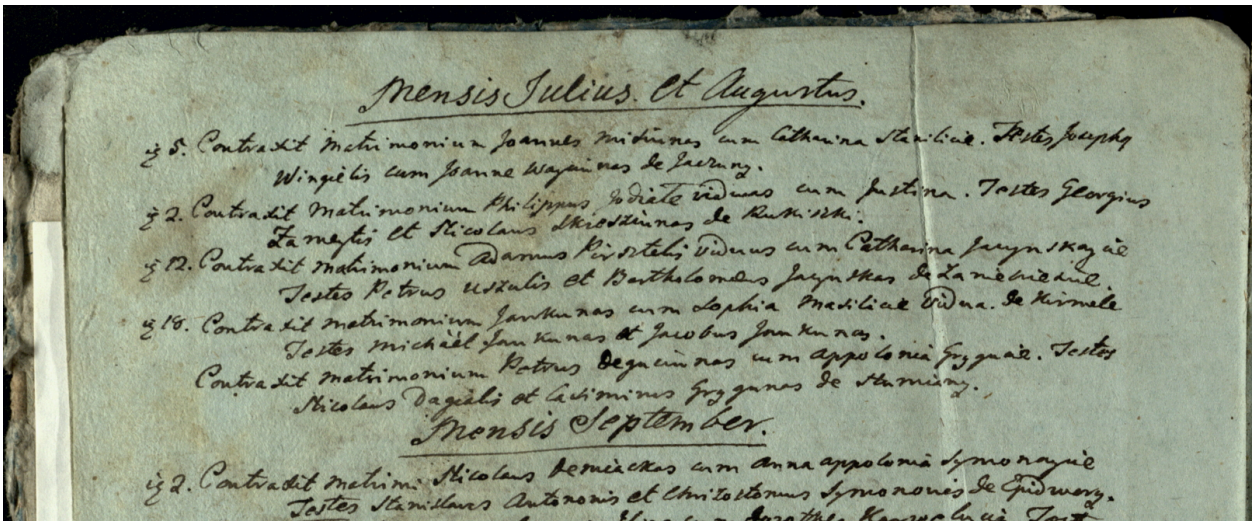
`level` = 1.

Duomenys, kurie kaupiami bet kokia forma, kurioje yra matoma tam tikra duomenų struktūra, tačiau patys duomenys pateikti taip, kad matomos struktūros nuskaityti galimybės nėra, tada `level` stulpelyje nurodome 1. Tokiu būdu pažymėdami, kad duomenys yra, juose aiškiai matoma tam tikra struktūra, bet jų nuskaityti galimybės nėra.

Nestruktūruotų duomenų inventorizacija yra svarbi, kadangi tai leidžia matyti pilnesnį viso duomenų ūkio vaizdą, leidžia užpildyti trūkstamų duomenų skyles.

Inventorizuojant nestruktūruotus duomenis, pirmiausia reikia surasti tam tikrą pasikartojančią struktūrą ir ją aprašyti.

Kaip pavyzdį galima galime imti skaitmenintus RKB metrikus.



Konkrečiai šiame pavyzdyje pateikti santuokos metrikų įrašai, tokių skaitmenintų paveikslėlių yra ištisos knygos ir visose knygose pateikiami gimimo, santuokos ir mirties įrašai, turintys labai aiškią struktūrą.

Deja, nuskaityti duomenis iš tokių paveiksliukų galimybės nėra, arba tokių duomenų nuskaitymas yra ypač sunkiai įgyvendinimas, tačiau galime pateikti tokių duomenų struktūros aprašą, kuris atrodys taip:

id	d	r	b	m	property	type	ref	source	level
	datasets/gov/rkb/metrikai								
		epaveldas							
				Lapas					
					paveikslas	image			1
				Asmuo					
					vardas	string			1
					pavarde	string			1
				Ivykis					
					tipas	string			1
					asmuo	ref	asmuo		1
					data	date			1
					lapas	ref	lapas		1

Paruošus, kad ir labai primityvų inventorizacijos lentelės variantą, galima toliau su ja dirbti, sieti su kitais modeliais, gauti paklausos rodiklius i duomenų naudotojo, tobulinti duomenų modelį, dokumentuoti duomenų laukus.

Tai, kad tokie duomenys dalyvauja bendroje apskaitoje, reiškia, kad galima matyti, kiek potencialių projektų galėtų įdarbinti šiuos duomenis ir kokią naudą tai galėtų atnešti.

3.3 Nestandartinio formato duomenys

`level` = 2.

Jei duomenys turi aiškią struktūrą ir pateikta struktūra nesunkiai nuskaitoma, tačiau formatas nėra standartinis, tada tokiems duomenims suteikiama `level` reikšmė 2.

Kalbant apie nestandartinius formatus, turima mintyje, ne tik duomenų šaltinio formatą, bet ir reikšmių formatą.

3.4 Duomenys be metaduomenų

level = 3.

Jei duomenys yra struktūruoti, sturktūra lengvai nuskaitoma ir nuskaitoma standartinėmis priemonėmis, tada tokiems duomenims suteikiama *level* reikšmė 3.

3.5 Duomenys be identifikatorių

level = 4.

3.5.1 Objektų identifikavimas

Kadangi atvirų duomenų saugykloje duomenys turėtų būti saugomi normalizuotoje formoje, susiejat lenteles tarpusavyje ryšiais, labai svarbu tinkamai identifikuoti objektus.

Tarkim, jei turime tokius duomenis:

COUNTRIES	
code	country
lt	Lietuva
lv	Latvija
ee	Estija

Šioje lentelėje nėra pirminio rakto, todėl inventorizacijos lentelėje, `model` eilėtės `ref` stulpelis yra tuščias:

id	d	r	b	m	property	type	ref	source	level
	datasets/gov/dc/countries								
		sql							
				Countries				COUNTRIES	
					code	string		code	2
					country	string		country	2

Tam, kad lentelę būtų galima sieti su kitomis lentelėmis reikia turėti patikimą identifikatorių. Šiuo atveju, galima daryti prielaidą, kad laukas `code` unikaliai identifikuoja `countries` modelio įrašus, todėl `model` ielutės `ref` stulpeliui galima priskirti `code` reikšmę taip pakeliand modelio brandos lygį iki 4.

id	d	r	b	m	property	type	ref	source	level
	datasets/gov/dc/countries								
		sql							
				Countries			code	COUNTRIES	
					code	string		code	4
					country	string		country	4

Šiuo atveju, laukas `code` yra šalies kodas, kuris unikaliai identifikuoja objektą. Todėl galima šį lauką naudoti, kaip unikaliai identifikuojančią šalies reikšmę.

Dažnai pasitaiko, kad neužtenka vieno lauko norint unikaliai identifikuoti objektą, tokiu atveju, galima pateikti kelis laukus `ref` stulpelyje, atskiriant juos kableliu.

Po pertvarkymų taip pat reikėtų nepamiršti atnaujinti `level` stulpelio reikšmių, nurodant pasikeitusį brandos lygį. Kadangi atsirado galimybė identifikuoti modelio objektus, `code` laukui suteikėme 4 brandos lygį. Atitinkamai, pakeliam ir kitų laukų brandos lygį, kadangi įsitikinome, kad automatiškai suteiktas `string` tipas yra teisingas, kas leidžia suteikti 3 brandos lygį, tačiau taip pat įsitikinome, kad nei vienas iš laukų nėra ryšio su kita lentele laukas, todėl galime suteikti 4 brandos lygį.

Nei vienam iš šių laukų negalima suteikti 5 brandos lygio, kadangi `base` eilutė yra tuščia.

3.5.2 Objektai be identifikatoriaus

Duomenų šaltinis ne visada leidžia unikaliai identifikuoti objektą. Pavyzdžiui, jei turime tokią šaltinio lentelę:

VILLAGES	
name	population
Gudeliai	28
Gudeliai	27
Gudeliai	19

Lentelė objektas yra kaimo gyvenvietė, tačiau nėra jokio kaimo gyvenvietės unikalaus identifikatoriaus. Lietuvoje gali būti daug gyvenviečių tokiu pačiu pavadinimu, ką ir matome lentelėje. Jungti gyvenvietės pavadinimo su gyventojų skaičiumi taip pat negalime, nes gyventojų skaičius gali sutapti su pavadinimu, be to gyventojų skaičius nuolat kinta.

Šiuo atveju neturim jokios išeities ir vienintelis būdas pakelti šio rinkinio brandos lygį, keičiant originalų duomenų šaltinį. Susidūrėme su nepakankamų duomenų atveju.

Galutinė inventorizacijos lentelė turėtų atrodyti taip:

id	d	r	b	m	property	type	ref	source	level
	datasets/gov/dc/villages								
		sql							
				Villages				VILLAGES	
					name	string		name	4
					population	string		population	4

`name` ir `population` laukams suteikėme 4 brandos lygį, kadangi šie laukai nėra `ref` tipo. Tačiau bendro modelio brandos lygio skaičiavime, šių laukų brandos lygis bus nuleistas iki 3, kadangi modelis neturi identifikatoriaus, todėl nė vienas laukas išskyrus `ref` tipo laukus, negali turėti didesnio brandos lygio nei 4.

Inventorizacijos lentelėse, kiekvieno lauko brandos lygį galima žymėti individualiai. Net jei modelis neturi identifikatoriaus, tačiau tam tikras laukas nėra `ref` tipo ir to lauko duomenys tvarkingi ir atitinka lauko duomenų tipą, lauko pavadinimai naudoja manifesto žodyno pavadinimus, tada tam laukui galima suteikti 5 brandos lygį. Tačiau reikia atkreipti dėmesį, kad bendro brandos lygio skaičiavimuose, šio lauko brandos lygis gali būti sumažintas, jei modelis neatitinka tam tikrų kriterijų, pavyzdžiui jei modelis neturi unikalaus identifikatoriaus.

3.5.3 Ryšiai tarp lentelių

Labai svarbu atveriant duomenis nepamesti ryšių tarp lentelių. Turint veikiančius ryšius tarp lentelių atsiranda galimybė duomenis jungti tarpusavyje, o tai yra labai svarbu.

Tarkime, duomenų šaltinyje yra tokios dvi lentelės:

COUNTRIES		
id	code	country
1	lt	Lietuva
2	lv	Latvija
3	ee	Estija

CITIES		
id	country	city
1	1	Vilnius
2	1	Kaunas
3	1	Klaipėda

Iš šių lentelių gauname tokią inventorizacijos lentelę:

id	d	r	b	m	property	type	ref	source	level
	datasets/gov/dc/countries								
		sql							
				Countries			id	COUNTRIES	
					id	integer		id	4
					code	string		code	4
					country	string		country	4
				Cities			id	CITIES	
					id	integer		id	4
					country	ref	countries	country	4
					city	string		city	4

Kaip matome ryšys tarp lentelių buvo aptiktas automatiškai, kadangi tokia informacija yra pateikta duomenų bazės schemoje. Tačiau gali pasitaikyti atvejai, kad ryšiai tarp lentelių nėra aprašyti duomenų bazės schemoje, tokiais atvejais, ryšius reikia aprašyti rankiniu būdu.

Norint nurodyti ryšį su kita lentele, reikia lauko `type` stulpelyje nurodyti `ref`, o `ref` stulpelyje nurodyti kitos lentelės pavadinimą iš `model` stulpelio.

Ryšiai tarp lentelių gali būti nurodomi tik vieno duomenų rinkinio resurso ribose.

Laukai naudojami ryšiams tarp lentelių automatiškai nustatomi pagal rodomo modelio `ref` reikšmes. Pavyzdžiui šiuo atveju modelio `countries` eilutės `ref` reikšmė yra `id`, todėl modelio `cities` savybė `country` automatiškai siejama su `id` lauku. Tačiau galima laukus, nurodyti ir rankiniu būdu taip: `countries[id]`.

Atveriant duomenis, vidinės duomenų bazės identifikatoriai nėra perkeliami. Visi identifikatoriai generuojami naujai, kad neatskleisti vidinės duomenų bazės detalių.

Jei šaltinio lentelės yra susietos naudojant daugiau nei vieną lauką, `source` stulpelyje galima nurodyti kelis laukus, atskiriant juos kableliu. Arba `property` eilutės `ref` stulpelyje galima nurodyti kelis laukus taip `countries[id, code]`.

3.5.4 Sudėtiniai identifikatoriai

Dažnai pasitaiko, kad informacinių objektų negalima identifikuoti kurios nors vienos savybės pagalba. Tokiais atvejais, tenka pasitelkti sudėtinius identifikatorius, kur vienas informacinis objektas identifikuojamas kelių savybių pagalba.

Kaip pavyzdį galime panagrinėti šį duomenų šaltinį

CITIES	
COUNTRY	CITY
Lietuva	Vilnius
Lietuva	Kaunas
Latvija	Ryga

STREETS			
ID	COUNTRY	CITY	STREET
1	Lietuva	Vilnius	Gedimino pr.
2	Lietuva	Vilnius	Vilniaus g.
3	Lietuva	Vilnius	Konstitucijos pr.

Čia matome, kad `STREETS` lentelė siejasi su `CITIES` lentele, tačiau sąsajai tarp lentelių neužtenka vieno lauko. Norinti unikaliai identifikuoti `CITIES` *objektą* būtina naudoti dvi `country` ir `city` *savybes*.

Tokią duomenų struktūrą galima aprašyti taip:

id	d	r	b	m	property	type	ref	source	prepare	level	access
1	datasets/gov/dc/countries										
2		db				sql					
3				City			id	CITIES			
4					id	array			country, name	4	private
5					country	string		COUNTRY		3	open
6					name	string		CITY		3	open
7				Street			id	STREET			
8					id	integer		ID		4	private
9					country	string		COUNTRY		3	open
10					city_name	string		CITY		3	private
11					city	ref	city		country, city_name	4	open
12					name	string		STREET		3	open

Tam, kad `city` lentelei aprašyti kompozicinį raktą, 4-oje eilutėje buvo įtraukta nauja savybė `id`, kuri tiesioginio analogo pirminiame duomenų šaltinyje neturi, todėl šios savybės *`property.source`* yra tuščias, tačiau šios savybės reikšmė gaunama *`property.prepare`* pagalba, kur nurodyta, kad reikšmė gaunama apjungiant `country` ir `name` *savybes*.

Analogiška situacija ir su `street` *modeliu*.

`street.city_name` *`property.access`* pažymėtas `private`, kadangi miesto pavadinimas yra perteklinė informacija. Miesto pavadinimą galima gauti apjungiant `city` ir `street` *modelius*.

3.5.5 Globalūs identifikatoriai

Dažniausiai nėra didelių problemų su lokaliais, vieno duomenų rinkinio ribose naudojamais identifikatoriais. Objektus galima jungti tarpusavyje, tačiau tik vieno duomenų rinkinio ribose.

Atsiveria žymiai didesnės galimybės, jei objektus galima jungti ir už vieno rinkinio ribų, su visais kitais, visuose kituose rinkiniuose esančiais objektais.

Kad tai veiktų, naudojami globalūs objektų identifikatoriai. Iliustruosiu, kaip visa tai veikia pavyzdžiu. Tarkime turime tokią lentelę viename duomenų rinkinyje:

COUNTRIES		
id	code	country
1	ltu	Lithuania
2	lva	Latvia
3	est	Estonia

Ir kitą lentelę, kitame duomenų rinkinyje:

SALYS		
id	kodas	salis
9	lt	Lietuva
8	lv	Latvija
7	ee	Estija

Abu duomenų rinkiniais valdomi skirtingose įstaigose, nors abu rinkiniai apie tą patį šalies objektą, tačiau vidiniai identifikatoriai skirtingi, žodynas taip pat skirtingas ir net patys duomenys yra skirtingi. Iš esmės nėra galimybės šių duomenų sujungti tarpusavyje.

Tačiau mums pasisekė, nes yra dar trečias duomenų šaltinis su šalių kodais:

CODES	
A2	A3
lt	ltu
lv	lva
ee	est

Pasitelkus šį trečiąjį duomenų šaltinį sujungti visas lenteles pasidaro įmanoma.

Galutinė, pilnai sutvarkyta visų trijų duomenų rinkinių inventorizacijos lentelė atrodytų taip:

id	d	r	b	m	property	source	type	ref	level
	datasets/gov/dp1/countries								
		sql							
			/place/Country					a3code	
				Countries		COUNTRIES		id	
					id	id	integer		3
					a3code	code	string		2
					name.en	country	text		2
	datasets/gov/dp2/countries								
		sql							
			/place/Country					a2code	
				Salys		SALYS		id	
					id	id	integer		5
					a2code	kodas	string		5
					name.lt	salis	text		5
	datasets/gov/dp3/countries								
		sql							
			/place/Country					a3code	
				Codes		CODES		a3code	
					a2code	A2	string		5
					a3code	A3	string		5

Žodyno lentelė turėtų atrodyti taip:

id	m	property	type
	place/Country		
		a2code	string
		a3code	string
		name	text

Duomenų atvėrimo metu, visi inventorizuoti duomenų rinkiniai bus siejami su žodyno modeliais pasitelkiant identifikatorių nurodytą *base.ref* stulpelyje. Jei duomenų rinkinio modelis neturi tokio lauko, tada susiejimas nebus daromas ir viso modelio brandos lygis nukris iki 4 brandos lygio.

Duomenų atvėrimo metu atskirų duomenų rinkinių duomenys bus saugomi atskirai, kadangi jie gali turėti laukų ne iš žodyno. Iš visų duomenų rinkinių bus kuriami ir globalūs, nuo konkretaus duomenų rinkinio nepriklausomi žodynų objektai.

Konkrečiai šiuo atveju `place/country` žodyno lentelė atvėrus duomenis atrodys taip:

place/country				
id	a2code	a3code	name.en	name.lt
1	lt	ltu	Lithuania	Lietuva
2	lv	lva	Latvia	Latvija
3	ee	est	Estonia	Estija

Kaip matote, iš pirmo žvilgsnio atrodė, kad dviejų duomenų rinkinių neįmanoma sujungti tarpusavyje, tačiau prijungus dar daugiau duomenų rinkinių, kaip kokia dėlionė iš mažų detalių susidėliojo pilna ir išsami modelio `place/country` lentelė.

3.6 Duomenys apibrėžti nestandartiniu žodynu

level = 5.

3.6.1 Normalizavimas

Duomenų normalizavimas iš esmės yra duomenų pasikartojimo mažinimas. Štai pavyzdys, kaip atrodo denormalizuoti duomenys:

https://example.com/1/miestai.csv	
šalis	miestas
Lietuva	Vilnius
Lietuva	Kaunas
Lietuva	Klaipėda

Kaip matote, stulpelyje `šalis` daug kartų pakartota reikšmė `Lietuva`. Toks duomenų pasikartojimas kelia daug problemų, jei duomenys keičiasi arba tas pats objektas gali būti pavadintas keliais skirtingais pavadinimais. Tarkime, turime duomenis iš kito tiekėjo, kurie atrodo taip:

https://example.com/2/miestai.csv	
šalis	miestas
Lietuvos respublika	Šiauliai
Lietuvos respublika	Panevėžys

Matome, kad šioje vietoje ta pati šalis pavadinta skirtingai.

Tokias besikeičiančių ir pasikartojančių duomenų problemas padeda spręsti unikalūs identifikatoriai arba pirminiai raktai.

Norint normalizuoti duomenis, mūsų lentelę reikia išskaidyti į dvi atskiras lenteles:

Šalys

id	šalis
1	Lietuva

Miestai

id	šalis	miestas
1	1	Vilnius
2	1	Kaunas
3	1	Klaipėda
4	1	Šiauliai
5	1	Panevėžys

Turinti tokią normalizuotą duomenų bazę, galima nesunkiai keisti šalies pavadinimą, galima į šalies lentelę įtraukti daugiau atributų ir visa tai užtenka padaryti vienoje vietoje, kadangi šalies pirminis raktas niekada nesikeičia.

Pirminius duomenis visada rekomenduojama saugoti normalizuotoje formoje, o denormalizuotos duomenų bazės kuriamos normalizuotos duomenų bazės pagrindu, jei norima atlikti duomenų analizę išvengiant skirtingų lentelių jungimo kainos.

Duomenų struktūrų aprašai turėtų būti kiek įmanoma normalizuoti. Jei pirminis duomenų šaltinis yra denormalizuotas, duomenų aprašuose nesunkiai galima atlikti normalizavimą, su sąlyga, jei įmanoma unikaliai identifikuoti objektus.

Mūsų aprašytą miestų pavyzdį normalizuoti galima šio duomenų struktūros aprašo pagalba:

d	r	b	m	property	type	ref	source	prepare
datasets/example/norm								
					enum	country	Lietuvos respublika	"Lietuva"
	miestai				csv		https://example.com/{}.csv	
			Country				1/miestai	
				name	string		šalis	
			City				1/miestai	
				name	string		miestas	
				country	ref	country	šalis	
		Country			proxy	name		
			Country2			name	2/miestai	
				name			šalis	choose(self, self, country)
		City			proxy	name		
			City2			name	2/miestai	
				name	string		miestas	
				country	ref	country	šalis	choose(self, self, country)

Iš šio pavyzdžio matome, kad miestų duomenys iš pirmojo šaltinio `miestai1` skaitomi du kartus ir paskirstomi dviejose lentelėse. Pirmą kartą skaitome tik šalis, generuojant pirminį raktą iš šalies pavadinimo, antrą kartą skaitome tik miestus ir prijungiame šalį panaudojant šalies pirminį raktą.

Antram duomenų šaltiniui darome tą patį, tik normalizuojame šalies pavadinimus panaudodami *Klasifikatoriai* reikšmių normalizavimo sąrašą.

Abiejų duomenų šaltinių modeliai turi vieną `country` bazę ir vieną `city` bazę. O kadangi *`base.type`* yra `proxy`, tai duomenų saugykloje, bus saugoma tik viena lentelė, bendra abiem šaltiniams. Šaltinių duomenys š bazines lenteles bus apjungiami sutapatinant objektus, pagal miesto ir šalies pavadinimus.

Galutiniame rezultate gauname tokias lenteles:

datasets/example/norm/country	
_id	pavadinimas
1	Lietuva

datasets/example/norm/city		
_id	šalis	miestas
1	1	Vilnius
2	1	Kaunas
3	1	Klaipėda
4	1	Šiauliai
5	1	Panevėžys

3.6.2 Lentelių apjungimas

Kartais yra poreikis, skirtingas šaltinio lenteles apjungti į vieną. Pavyzdžiui:

APSKRITYS	
id	pavadinimas
1	Vilniaus
2	Kauno
3	Klaipėdos

SAVIVALDYBES		
id	apskritis	pavadinimas
1	1	Vilniaus miesto
2	1	Vilniaus rajono
3	1	Trakų rajono

Kadangi skirtingos šalis naudoja skirtingus administracinius suskirstymus, tai mes norime normalizuoti šias lenteles, ir padaryti iš jų vieną administracijų lentelė.

Tarkime, apskrities administracinis vienetas bus žymimas skaičiumi `1`, o savivaldybės skaičiumi `2`. Turime dvi konstantas administraciniam vienetui.

Mūsų pradinė inventorizacijos lentelė atrodys taip:

id	d	r	b	m	property	type	ref	source	level
	datasets/gov/dc/administracijos								
		sql							
				Apskritys			id	APSKRITYS	
					id	integer		id	4
					pavadinimas	string		pavadinimas	2
				Savivaldybes			id	SAVIVALDYBES	
					id	integer		id	4
					apskritis	ref	apskritys	apskritis	4
					pavadinimas	string		pavadinimas	2

Mums reikia pertvarkyti inventorizacijos lentelę taip, kad gautume tokį duomenų pavidalą:

ADMINISTRACIJOS			
id	priklauso	lygis	pavadinimas
1	NULL	1	Vilniaus
2	NULL	1	Kauno
3	NULL	1	Klaipėdos
4	1	2	Vilniaus miesto
5	1	2	Vilniaus rajono
6	1	2	Trakų rajono

Kad tai gautume, mums reikia atlikti tokius pakeitimus:

- Primiausiai, apsirašome naują modelį `administracijos`, kadangi galutiniame rezultate norime turėti viską vienoje lentelėje.
- Tada nurodome, kad `apskritys` ir `savivaldybes` yra modelio `administracijos` dalis. Tai reiškia, kad galiausiai duomenys iš `apskritys` ir `savivaldybes` bus apjungti į vieną modelį `administracijos`.

- Keičiame lauko `savivaldybes.apskritis` pavadinimą į `priklauso`, kad lauko pavadinimas sutaptu su `administracijos.priklauso`.

  Kai du modeliai siejamie per `base` lauką, apjungtieji modeliai tampa vieno modelio dalimi ir turi tokias pačias savybes, kaip ir bazinis modelis. Šiuo atveju bazinis modelis yra `administracijos`.

- Paskutinis pakeitimas, tiek apskritims, tiek savivaldybėms pridėti `lygis` savybę nurodant konstantas `1` ir `2`.

Po pertvarkymų, mūsų inventorizacijos lentelė turėtų atrodyti taip:

id	d	r	b	m	property	type	ref	source	level
	datasets/gov/dc/administracijos								
		sql							
				Administracijos					
					priklauso	ref	administracijos		
					lygis	integer			
					pavadinimas	string			
			Administracijos			proxy			
				Apskritys			id	APSKRITYS	
					id	integer		id	4
					lygis	integer		1	4
					pavadinimas	string		pavadinimas	4
				Savivaldybes			id	SAVIVALDYBES	
					id	integer		id	4
					priklauso	ref	apskritys	apskritis	4
					lygis	integer		2	4
					pavadinimas	string		pavadinimas	4

`administracijos` modelis neturi `level` reikšmių, taip yra todėl, kad `administracijos` modelis yra išvestinis ir neturi tiesioginio šaltinio, o duomenų brandos lygis nurodomas duomenų laukams kurie tiesiogiai gaunami iš tam tikro duomenų šaltinio.

Kadangi `base` `administracijos` eilutėje `ref` stulpelio yra reikšmė, tai susiejimas bus daromas pagal vidinį modelio identifikatorių. Tai reiškia, kad modeliai `apskritys` ir `savivaldybes` nepersidengs.

`base` `administracijos` eilutėje `type` sulpelio reikšmė `proxy` reiškia, kad modeliai `apskritys` ir `savivaldybes` jokių duomenų nesaugos, o veiks kaip perlaidos režimu ir duomenis rašys tik į `administracijos` modelį.

3.6.3 Lentelės skaidymas

Prieš tai aptarėme kaip apjungti kelias lenteles į vieną modelį. O dabar aptarsime, kaip daryti atvirkštinį procesą, kaip skaidyti vieną lentelę į kelis modelius.

Tarkime turime tokią lentelę:

ADMINISTRACIJOS			
id	priklauso	lygis	pavadinimas
1	NULL	1	Vilniaus
2	NULL	1	Kauno
3	NULL	1	Klaipėdos
4	1	2	Vilniaus miesto
5	1	2	Vilniaus rajono
6	1	2	Trakų rajono

Norime šią lentelę suskaidyti į dvi atskiras lenteles. Įrašai, kurių `lygis` reikšmė yra `1` turėtų keliauti į apskričių modelį, o įrašai, kurių `lygis` reikšmė yra `2` turėtų keliauti į savivaldybių modelį.

Pirminė inventorizacijos lentelė atrodo taip:

id	d	r	b	m	property	type	ref	source	level
	datasets/gov/dc/administracijos								
		sql							
				Administracijos			id	ADMINISTRACIJOS	
					id	integer		id	4
					priklauso	ref	administracijos	priklauso	4
					lygis	integer		lygis	2
					pavadinimas	string		pavadinimas	2

Tam, kad suskaidyti vienos lentelės duomenis į kelis skirtingus modelius, mums reikia panaudoti filtrus lentelės lygmenyje. Metaduomenys lentelės lygmenyje taikomi tada, kai `property` reikšmė yra tuščia.

`source` stulpelyje galima nurodyti užklausą duomenims filtruoti. Duomenų filtras pateikiamas tarp `[]` skliaustelių.

Šiuo atveju, mums reikia filtruoti duomenis pagal stulpelio `lygis` reikšmes.

Galutinė inventorizacijos lentelė, po pertvarkymų atrodo taip:

id	d	r	b	m	property	type	ref	source	prepare	level
	datasets/gov/dc/administracijos									
		sql								
				Apskritys			id	ADMINISTRACIJOS	lygis=1	
					id	integer		id		4
					pavadinimas	string		pavadinimas		4
				Savivaldybes			id	ADMINISTRACIJOS	lygis=2	
					id	integer		id		4
					apskritis	ref	apskritys	priklauso		4
					pavadinimas	string		pavadinimas		4

3.6.4 Vieningo žodyno naudojimas

Tam, kad iš pirminio duomenų chaoso padaryti aukščiausio brandos lygio atvirus duomenis, būtina išversti `model` ir `property` stulpelių pavadinimus į pavadinimus iš vieningo žodyno.

Kaip pavyzdį galime imti tokius duomenis:

COUNTRIES		
id	code	country
1	lt	Lietuva
2	lv	Latvija
3	ee	Estija

Šiuose duomenyse yra šalių kodai ir pavadinimai. Kadangi, tai gan dažnai naudojami duomenys, tikėtina, kad skirtinguose duomenų šaltiniuose panaši lentelė ir jos laukai turės kitokius pavadinimus.

Tam, kad suvienodinti pavadinimus, mums reikia pasitelkti vieningą žodyną.

Žodynų sudarymas, yra gan sudėtingas darbas, todėl, jei tik yra galimybė reikėtų remtis egzistuojančiais žodynais. Egzistuojančius žodynus galima rasti LOV svetainėje, WikiData dažniausiai taip pat būna labai naudingas.

Tačiau nebūtina tiksliai atkartoti tai, kas pateikiama žodynuose, nes dažnai žodynai yra labai bendro pobūdžio ir ne viską apimantys. Todėl sudarant žodynus yra laisvė

Vieningas žodyno *DSA* atrodo taip:

id	m	property	type	ref	uri	title
			prefix	esco	http://data.europa.eu/esco/model#	
			prefix	og	http://ogp.me/ns#	
	place/Country				esco:Country	Šalis
		code	string		esco:isoCountryCodeA2	ISO 3166-1 A2 kodas
		name	string		og:country-name	Pavadinimas

Toliau, įprastai aprašome duomenų šaltinį ir įtraukiame *base* dimensiją, kurios pagalba duomenis nukreipiame į standartų vardų erdvę.

id	d	r	b	m	property	type	ref	source	level
1	datasets/gov/dc/countries								
2		sql							
3			/place/Country				code		
4				Countries			id	COUNTRIES	
5					id	integer		id	3
6					code	string		code	3
7					name	string		country	3

Duomenų rinkinių modeliai siejami su žodynu *base* stulpelyje pateikiant susiejamo modelio pavadinimą iš standartų vardų erdvės. Tada atitinkamai reikia pakeisti `property` reikšmes, kad jos atitiktų *base* modelio pavadinimus.

Dar vienas svarbus momentas yra `code` reikšmė *base.source* stulpelyje, 3-ioje eilutėje. Ši reikšmė nurodo kaip `datasets/gov/dc/countries/countries` modelio *objektai* turi būti identifikuojami `place/country` modelyje. Šiuo atveju nurodyta, kad objektų siejimas turi būti daromas per `code` lauką. Toks objektų susiejimas leidžia turėti vienodus identifikatorius visiems duomenų rinkiniams kurie yra `place/country` modelio dalis.

SKYRIUS 4

Asmens duomenys

4.1 Nuasmeninimas

Duomenų laukų reikšmių nuasmeninimas atliekamas *property.prepare* stulpelio pagalba.

randomize(*n*)
: Reikšmės keičiamos parenkant atsitiktinę vertę ±n intervale nuo tikrosios vertės.

permutate()
: Atsitiktine tvarka sumaišomos duomenų reikšmės.

hash()
: Taikyti numatytą maišos funkciją.

hash(*name*)
: Taikyti konkrečią name maišos funkciją.

sample(*n*)
: Atsitiktine tvarka atrenkama n procentų žodžių naudojamų tekste.

group(*n*)
: Pakeičia originalias reikšmes į intervalų grupes taip, kad į vieną intervalą patektų ne mažiau nei n reikšmių. Jei viena konkreti reikšmė pasikartoja daugiau nei n kartų, tada intervalas nekuriamas, reikšmė paliekama tokia kokia yra šaltinyje.

4.2 Asmenų identifikuojantys duomenys

Asmenų identifikuojančios savybės turi būti pažymėtos `property.ref` stulpelyje. Rekomenduojame naudoti tokius reikšmes, kadangi jos gali būti naudojamos nuasmeninimo procese:

type	ref	uri	title
prefix	person	https://www.w3.org/ns/person#	
	pii	https://data.gov.lt/pii/	
model		person:Person	Fizinis asmuo
string		person:patronymicName	Tėvavardis
string		person:birthName	Pilnas vardas
string		person:placeOfBirth	Vieta kurioje gimė
string		person:placeOfDeath	Vieta kurioje mirė
string		person:countryOfBirth	Šalis kurioje gimė
string		person:countryOfDeath	Šalis kurioje mirė
string		person:citizenship	Tautybė
string		person:residency	Vieta kurioje gyvena
string		pii:name	Vardas ir/arba pavardė
date		pii:dob	Gimimo data
string		pii:phone	Telefono numeris
string		pii:email	El. pašto adresas
string		pii:pid	Asmens kodas
string		pii:id	Asmens identifikatorius
string		pii:address	Asmens adresas
string		pii:age	Amžius
string		pii:sex	Lytis
string		pii:other	Kita šiame sąraše nepateikta asmenų identifikuojanti informacija

4.3 Viešieji asmenys duomenys

Jokie asmens duomenys negali būti teikiami, kaip atviri duomenys, tačiau gali būti teikiami pakartotiniam naudojimui laikantis *BDAR* ir *kitų duomenų valdymo* reikalavimų.

Siekiant užtikrinti viešąjį interesą, tam tikri viešųjų asmenų duomenys gali būti teikiami pakartotiniam naudojimui, tačiau ribojant duomenų naudojimo tikslą ir laikantis visų reikalavimų taikomų asmens duomenims.

Šiame skyriuje aptariama, kaip gali būti teikiami viešųjų asmenų duomenys.

4.3.1 Asmens duomenų identifikavimas

DSA lentelėje, duomenys kuriuos galima viešinti, tačiau jų naudojimui taikomi papildomi apribojimai, `access` stulpelyje turi būti pažymėti `public` reikšme (toliau vadinami `public` duomenimis).

Viename *modelyje* negali būti sumaišyti asmens ir kiti duomenys. Pavyzdžiui jei vienoje lentelėje galima rasti tiek fizinių, tiek juridinių asmenų duomenis, tada, fizinių ir juridinių asmenų duomenys turi būti išskaidyti į atskirus duomenų *modelius*, iš kurių vienas gali būti teikiamas, kaip atviri duomenys, o kitas su `public` prieigos teis.

`public` žyme galima žymėti viešus asmenis, kurių duomenų viešinimas yra būtinas siekiant užtikrinti viešąjį interesą. Privačių asmenų ar kiti konfidencialūs duomenys turi būti žymimi griežtesnėmis `protected` arba `private` žymėmis.

4.3.2 Duomenų naudotojų autorizavimas

`public` duomenys nėra teikiami, kaip atviri duomenys. Duomenų naudotojai, pageidaujantys gauti `public` duomenis, privalo save identifikuoti. Tada tokie duomenų naudotojai užregistruojami ir jiems išduodamas naudotojo identifikavimo kodas ir slaptažodis.

Registruoti naudotojai gali kreiptis į duomenų saugyklą su prašymu išduoti *autorizacijos raktą*.

Duomenų naudotojai vykdydami užklausas duomenims gauti, `public` duomenų atveju yra nukreipiami į savitarnos puslapį, kuriame gali susipažinti su pageidaujamų duomenų naudojimo sąlygomis. Susipažinę su sąlygomis ir patvirtinę, kad su sąlygomis sutinka, gauna prieigą prie duomenų.

Skirtingi `public` duomenų rinkiniai gali turėti skirtingas naudojimo sąlygas, su kuriomis susipažinti ir su jomis sutikti reikia atskirai. Tačiau visus `public` duomenis, su kurių naudojimo sąlygomis sutiko, duomenų naudotojas gauna vienu ir tuo pačiu prieigos raktu.

Duomenų naudotojo registracija yra ilgalaikė, išduotas autorizacijos raktas yra trumpalaikis, galiojantis kelias minutes ar kelias valandas. Duomenų naudotojo sutikimai su sąlygomis yra ilgalaikiai, tačiau priklausomai nuo duomenų rinkinio, gali būti terminuoti.

4.3.3 Duomenų naudotojų įsipareigojimai

Duomenų naudotojas, gavęs prieigos raktą, įsipareigoja laikytis duomenų naudojimo sąlygų ir įgyvendinti priemones asmens duomenų šalinimui iš savo duomenų saugyklos. Duomenų šalinimui, duomenų naudotojas privalo teikti sutartinį API prieigos tašką *aprašytą šiame vadove*. Privaloma įgyvendinti tik *wipe* operaciją.

Jei keičiasi viešų asmens duomenų naudojimo reglamentavimas ar pats viešųjų asmens duomenų subjektas atšaukia sutikimą naudoti savo duomenis arba baigiasi terminas, kurio metu buvo galima naudoti duomenis arba duomenų naudotojas nesilaiko duomenų naudojimo sąlygų, tada duomenų tiekėjas vykdo *wipe* užklausą, duomenų naudotojo duomenų saugykloje, taip nurodant, kad duomenų naudotojas privalo visiškai pašalini arba nuasmeninti nurodyto asmens subjekto arba visus asmens duomenis.

Įvykdžius *wipe*, duomenų tiekėjas, kelis kartus tikrina ar duomenys tikrai ištrinti vykdydamas *getone* užklausą.

Taip pat, duomenų tiekėjas gali vykdyti *getone* užklausą, jei viešų asmens duomenų subjektas prašo eksportuoti visus savo duomenis.

Dėl minėtų priežasčių, duomenų naudotojas įsipareigojai įgyvendinti *getone* ir *wipe* operacijas savo duomenų saugykloje ir suteikti *prieigą* prie savo saugyklos duomenų tiekėjui su *getone* ir *wipe* teisėmis iš tiekėjo gautiems duomenims.

Jei duomenų naudotojas nesilaiko duomenų naudojimo taisyklių, tuomet duomenų tiekėjas gali nutraukti asmens duomenų tiekimą ir papildomai vykdys eilę *wipe* užklausų, kad pašalintu asmens duomenis duomenų naudotojo pusėje.

Duomenų naudotojas, naudojantis asmens duomenis, tampa asmens duomenų valdytoju ir prisiima visą su tuo susijusią atsakomybę, įsipareigoja laikytis visų *BDAR* reikalavimų.

4.3.4 Subjektų savitarna

Asmens duomenų subjektams yra prieinama savitarnos sritis, kurioje subjektai gali matyti kokie jų duomenys saugomi saugykloje, kam, kokiu pagrindu ir kokiu tikslu duomenys teikiami, gali atšaukti sutikimą teikti duomenis, gali eksportuoti visus savo duomenis.

SKYRIUS 5

Duomenų šaltiniai

5.1 SQL

Tarkime turime PostgreSQL duomenų bazę, kurioje yra dvi lentelės:

SALIS		
ID	KODAS	PAVAD
100	lt	Lietuva
101	lv	Latvija
102	ee	Estija

MIESTAS		
ID	SALIS_ID	PAVAD
204	100	Vilnius
205	100	Kaunas
206	100	Klaipėda
207	101	Ryga
208	102	Talinas

Tarkime, kad mes norime atverti tik Lietuvos duomenis, ignoruojant kitų šalių duomenis.

Duomenų aprašas atrodys taip:

id	d	r	b	m	property	type	ref	source	prepare	access
	datasets/example/sql									
		db				sql		postgresql://user@host/dbname		
				country			id	SALIS	code="lt"	
					id	integer		ID		private
					code	string		KODAS		open
					name	string		PAVAD		open
				city			id	MIESTAS	country.code="lt"	
					id	integer		ID		private
					country	ref	country	SALIS_ID		open
					name	string		PAVAD		open

Reliacinės duomenų bazės (RDBVS) yra populiariausias duomenų saugojimo būdas. Tačiau taip pat RDBVS struktūra yra lengviausiai aprašoma, didelė dalis metaduomenų saugoma pačioje RDBVS, todėl nesunkiai galima generuoti gan išsamias *DSA* lenteles automatiškai.

Atkreipkite dėmesį, kad `id` *savybės* pažymėtos `private` prieigos žyme. Taip rekomenduojama daryti tam, kad nebūtų atskleisti vidiniai duomenų bazės identifikatoriai. Dažniausiai tokie identifikatoriai yra naudingi tik duomenų bazės viduje ir išorėje neturi naudos. Be to, tam tikromis situacijomis, pasinaudojant vidiniais identifikatoriais galima atskleisti konfidencialius duomenis.

Kadangi išsikėlėme reikalavimą atverti tik Lietuvos duomenis, `model.prepare` stulpelyje nurodėme atveriamų duomenų filtrus. Dažniausiai tokie filtrai naudojami asmens duomenų filtravimui arba skaidant lenteles į kelis atskirus *medelius*.

Atlikus visas *DSA* lentelėje aprašytas transformacijas gausime tokias duomenų lenteles:

datasets/example/sql/country		
_id	code	name
1	lt	Lietuva
2	lv	Latvija
3	ee	Estija

datasets/example/sql/city		
_id	country	name
3	1	Vilnius
4	1	Kaunas
5	1	Klaipėda

Taip pat skaitykite: *Duomenų atranka*, *SQL*.

5.2 CSV

Tarkime turime tokius duomenis CSV formatu. CSV failuose reikšmės atskirtos ne įprastiniu `,` simboliu, o `;` simboliu.

https://example.com/salys.csv		
ID	KODAS	PAVADINIMAS
100	lt	Lietuva
101	lv	Latvija
102	ee	Estija

https://example.com/miestai.csv		
ID	ŠALIS	PAVADINIMAS
204	100	Vilnius
205	100	Kaunas
206	100	Klaipėda
207	101	Ryga
208	102	Talinas

Duomenų aprašas atrodys taip:

id	d	r	b	m	property	type	ref	source	prepare	access
1	datasets/example/csv									
2		salys				csv		https://example.com/{}.csv	tabular(sep: ";")	
3				Country			id	salys	code="lt"	
4					id	integer		ID		private
5					code	string		KODAS		open
6					name	string		PAVADINIMAS		open
7				City			id	miestai	country.code="lt"	
8					id	integer		ID		private
9					country	ref	Country	ŠALIS		open
10					name	string		PAVADINIMAS		open

CSV duomenų resursas, pavadinimu `salys` nurodo iš kur skaityti duomenis ir kokiu formatu. Nurodant adresą iki CSV failo `https://example.com/{}.csv` naudojama vietos žymė `{}`, kuri pakeičiama modelio šaltinio pavadinimu, kuris nurodytas `model.source` stulpelyje.

Prieš skaitant duomenis, `tabular.sep()` nurodo, kad CSV faile naudojamas nestandartinis reikšmių skirtukas, kabliataškis.

Visa kita aprašoma lygiai taip pat, kaip ir SQL atveju.

5.3 ZIP

Tais atvejais, kai duomenys pateikiami failais, o failai pateikiami tam tikruose failų konteineriuose, pavyzdžiui ZIP archyvuose, tuomet aprašomi du skirtingi šaltiniai, kurie yra susiję vienas su kitu.

Pavyzdžiui turint analogišką pavyzdį, kaip ir su CSV failais, tik jei CSV failai būtų patalpinti ZIP archyve, duomenų struktūros aprašas atrodytų taip:

d	r	b	m	property	type	ref	source
datasets/example/zip							
	archyvas				zip		https://example.com/data.zip
	salys				csv	archyvas	{}.csv
			Country			id	salys
				id	integer		ID
				code	string		KODAS
				name	string		PAVADINIMAS

Šiame pavyzdyje matome, kad atsirado naujas resursas pavadinimu `archyvas` rodantis į ZIP archyvo failą. Tuo tarpu CSV resursas pavadinimu `salys`, *`resource.ref`* stulpelyje, rodo, kad CSV failai yra `archyvas` resurso sudėtyje.

5.4 JSON

Tarkime JSON atveju turime API kuris atrodo taip:

```
https://example.com/salys/
```

```
{
   "šalys": [
      {"id": 100, "kodas": "lt", "šalis": "Lietuva"},
      {"id": 101, "kodas": "lv", "šalis": "Latvija"},
      {"id": 102, "kodas": "ee", "šalis": "Estija"}
   ]
}
```

```
https://example.com/miestai/lt
```

```
{
   "miestai": [
      {"id": 204, "miestas": "Vilnius"},
      {"id": 205, "miestas": "Kaunas"},
      {"id": 206, "miestas": "Klaipėda"}
   ]
}
```

```
https://example.com/miestai/lv
```

```
{
   "miestai": [
      {"id": 207, "miestas": "Ryga"}
   ]
}
```

```
https://example.com/miestai/ee
```

```
{
   "miestai": [
      {"id": 208, "miestas": "Talinas"}
   ]
}
```

Tokio API duomenų struktūrą galima aprašyti sekančios *DSA* lentelės pagalba:

id	d	r	b	m	property	type	ref	source	prepare	access
1	datasets/example/json									
2		api				json		https://example.com/{}/		
3		salys				json	api	salys		
4				Country			id	šalys		
5					id	integer		id		private
6					code	string		kodas		open
7					name	string		šalis		open
8		miestai				json		miestai/{country.code}		
9						param	country	Country	select()	
10				City			id	miestai		
11					id	integer		id		private
12					country	ref	Country		param("country").id	open
13					name	string		miestas		open

Šį kartą turime reikalą su dinaminiu API, kuris neleidžia gauti visų miestų vienos užklausos pagalba. Norint gauti visus miestus, pirmiausia gauti visų šalių kodus, o tada turint šalies kodą, galima gauti tos šalies miestų duomenis.

Kad užduotis nebūtų per daug lengva, šį kartą aprašome visų šalių duomenis, ne tik Lietuvos.

`model.source` stulpelyje nurodyti JSON atributų pavadinimai, iš kurių skaitomi duomenys.

8-oje eilutėje, `miestai` `resource` kontekste įtrauktas *Parametrai* pavadinimu `country`, kuris generuoja parametrus, skaitant duomenis iš 3-ioje eilutėje aprašyto `Country` *modelio*. Tokiu būdu gauname visų šalių sąrašą ir 7-oje eilutėje `resource.source` galime nurodyti URI su šalies kodu, gautu iš `country` *Parametrai*.

11-oje eilutėje, `country` reikšmę gauname iš `country` parametro, kadangi miesto duomenyse, nei miesto kodo, nei `id` nėra.

Galiausiai gauname tokius duomenis:

datasets/example/json/country		
_id	code	name
1	lt	Lietuva
2	lv	Latvija
3	ee	Estija

datasets/example/json/city		
_id	country	name
3	1	Vilnius
4	1	Kaunas
5	1	Klaipėda
6	2	Ryga
7	3	Talinas

5.5 XML

Tarkime turime XML failą, kuris pasiekiamas adresu `https://example.com/countries.xml`, failo turinys yra toks:

```
<root>
    <country id="100" code="lt" name="Lietuva">
        <city id="204" name="Vilnius" />
        <city id="205" name="Kaunas" />
        <city id="206" name="Klaipėda" />
    </country>
    <country id="101" code="lv" name="Latvija">
        <city id="207" name="Ryga" />
    </country>
    <country id="102" code="ee" name="Estija">
        <city id="208" name="Talinas" />
    </country>
</root>
```

Šio XML failo *DSA* atrodys taip:

id	d	r	b	m	property	type	ref	source	prepare	access
1	datasets/example/xml									
2		api				xml		https://example.com/{}.xml		
3		countries				xml	api	countries		
4				country			id	/root/country		
5					id	integer		@id		private
6					code	string		@code		open
7					name	string		@name		open
8				city			id	/root/country/city		
9					id	integer		@id		private
10					country	ref	country	parent::country/@id		open
11					name	string		@name		open

Šiuo atveju, visi duomenys pateikti viename XML faile, todėl aprašomas tik vienas *resource*. *model.source* ir *property.source* stulpelyje pateikiamas XPath reikšmė, kuri, jei *prepare* neužpildytas, vykdoma su `xml.xpath()` funkcija.

Galutiniame rezultate gauname tokius duomenis:

datasets/example/xml/country		
_id	code	name
1	lt	Lietuva
2	lv	Latvija
3	ee	Estija

datasets/example/xml/city		
_id	country	name
3	1	Vilnius
4	1	Kaunas
5	1	Klaipėda
6	2	Ryga
7	3	Talinas

5.6 XLSX

Tarkime yra XLSX failas, patalpintas adresu `https://example.com/SALYS.XLSX`, kuriame yra tokios dvi lentelės:

ŠALYS	
KODAS	PAVADINIMAS
lt	Lietuva
lv	Latvija
ee	Estija

MIESTAI	
ŠALIS	PAVADINIMAS
lt	Vilnius
lt	Kaunas
lt	Klaipėda
lv	Ryga
ee	Talinas

Duomenų aprašas atrodys taip:

id	d	r	b	m	property	type	ref	source	prepare	access
	datasets/example/sql									
		lentele				xlsx		https://example.com/SALYS.XLSX		
				country			code	ŠALYS		
					code	string		KODAS		open
					name	string		PAVADINIMAS		open
				city			id	MIESTAI		
					id	array			country, name	private
					country	ref	country	ŠALIS		open
					name	string		PAVADINIMAS		open

Šiuo atveju, turime problemą, kad lentelėje nėra pateikti aiškūs identifikatoriai. Šalių atveju, kaip identifikatorių galima naudoti `KODAS` stulpelį, tačiau miestų atveju, darant prielaidą, kad skirtingose šalyse gali būti miestai tokiais pačiai pavadinimais, pirminį raktą formuojame iš šalies kodo ir miesto pavadinimo, tam įtraukiame naują `id` stulpelį, kuris kuriamas iš `country` ir `name` reikšmių.

Galutiniame rezultate gauname tokius duomenis.

datasets/example/xml/country		
_id	code	name
1	lt	Lietuva
2	lv	Latvija
3	ee	Estija

datasets/example/xml/city		
_id	country	name
3	1	Vilnius
4	1	Kaunas
5	1	Klaipėda
6	2	Ryga
7	3	Talinas

5.7 Spinta

Paskutinis pavyzdys atliekant transformaciją tos pačios duomenų saugyklos viduje. Visi duomenys aukščiau aprašytuose pavyzdžiuose bus apjungiami ir perkelti į standartų vardų erdvę. Tokiu būdu, turėsime vieną aiškią duomenų struktūrą, visiems iki šilo aprašytiems duomenų šaltiniams.

Tokia transformacijų *DSA* atrodo taip:

d	r	b	m	property	type	ref	source
geo					ns		
			Country				
				code	string		
				name	string		
			City				
				country	ref	Country	
				name	string		
transformations/geo							
	data				spinta		https://example.com/
		/geo/Country			proxy	code	
			Country				/datasets/example/{source}
					param	source	sql
							csv
							json
							xml
							xlsx
				code	string		code
				name	string		name
		/geo/City			proxy	country, name	
			City				
				country	ref	Country	country
				name	string		name

Pirmiausiai apibrėžiame `geo` standarto duomenų struktūrą, toliau nurodome duomenų šaltinį `spinta`, kurio

resource.source sutampa su saugyklos adresu.

`source` parametrui priskiriame sąrašą visų iki šiol aprašytų duomenų rinkinių ir šio parametro pagalba skaitome visų šaltinių duomenis ir *base.type* `proxy` pagalba siunčiame visus juos į `geo` vardų erdvę.

base.ref stulpelyje nurodome, kaip bus identifikuojami *objektai*, kad neatsirastu dublikatų.

Galutiniame rezultate, gausime tokius duomenis:

geo/country		
_id	code	name
1	lt	Lietuva
2	lv	Latvija
3	ee	Estija

geo/city		
_id	country	name
3	1	Vilnius
4	1	Kaunas
5	1	Klaipėda
6	2	Ryga
7	3	Talinas

SKYRIUS 6

Priemonės

Siekiant užtikrinti sklandų ir kiek įmanoma paprastesnį duomenų atvėrimą, pateikiame ne tik metodinę medžiagą, kaip atverti duomenis, tačiau taip pat pateikiame ir įrankius, kurie padeda automatizuoti daugelį su duomenų atvėrimu susijusių veiklų.

Siūlome tokias priemones:

Katalogas *Atvirų duomenų Katalogas* pasiekiamas adresu data.gov.lt. Kataloge, vienoje vietoje galima rasti informacija apie atvertus duomenų rinkinius.

Saugykla *Atvirų duomenų Saugykla* pasiekiama adresu get.data.gov.lt. Saugykloje yra galimybė talpinti pačius duomenis, kurie bus publikuojami laikantis aukščiausių duomenų publikavimo standartų suteikiant galimybę duomenis gauti įvairiais formatais ar naudotis funkcionalia duomenų mašininio skaitymo sąsaja (*API*).

Spinta *Spinta* yra įvairių priemonių rinkinys skirtas duomenų atvėrimo automatizavimui. Atveriant duomenis užtenka parengti tik *duomenų struktūros aprašą*, o visos kitos duomenų atvėrimo veiklos Spintos priemonių pagalba atliekamas automatizuotai.

SKYRIUS 7

Katalogas

Organizacijos koordinatorių ir tvarkytojų aplinka yra skirta:

- duomenų rinkinių įkėlimui, įvertinimui ir atvėrimui;
- ataskaitų sudarymui.

7.1 Terminai ir sąvokos

„Laukas“ Kabutėmis „“ žymimas duomenų įvedimo laukas, kur tarp kabučių rašomas lauko pavadinimas, matomas aprašomame lange.

„Tekstinis laukas“ Vieta, kur sistemos naudotojas gali suvesti duomenis arba duomenys yra vaizduojami.

[Mygtukas] Dialogo lango arba lango mygtukai (taip pat vadinami komandiniais mygtukais) tekste yra žymimi kvadratiniais skliausteliais []. Tarp skliaustelių yra rašomas mygtuko pavadinimas. Sistemoje atvaizduoti kaip keturkampiai mygtukai ir aktyvios nuorodos (pabrauktas tekstas).

API (angl. Application Programming Interface) Programos valdymo sąsaja įgalina automatizuotai atlikti duomenų rinkinių kėlimą ir valdymą naudojant organizacijos programinę įrangą.

Lango mygtukas Mygtukas, kurio veiksmas įtakoja visus lango duomenis.

Įrašo mygtukas Mygtukas, kurio veiksmas įtakoja vieno įrašo duomenis.

IRS Rinkiniai iš Informacinių Rinkinių Sistemos (IRS)

„Meniu punktas” Kabutėmis „” yra žymimas meniu punktas. Tarp kabučių rašomas meniu punkto pavadinimas.

Žymimasis langelis Kvadrato formos figūra, kurios dešinėje rašomas tekstas. Aprašymas atspindi galimą veiksmą. Figūra parodo, ar yra pasirinkta nurodyta reikšmė, ar ne. Naudotojas gali keisti pasirinkimo langelio reikšmę pele pažymint arba panaikinant požymį langelyje.

Pagrindinis meniu Pagrindiniame meniu yra pristatomos pagrindinės sistemos funkcijos. Pagrindinio meniu punktai yra pasirenkami pelės kairiojo klavišo spustelėjimu pažymint juos.

Pasirinktas meniu punktas Norėdamas pasirinkti meniu punktą, sistemos naudotojas turi spragtelėti ant jo kairiu pelės klavišu.

Saugus slaptažodis Bent aštuonių simbolių ilgio; sudarytas iš raidžių, skaičių ir specialių simbolių (tokių kaip „@“, „#“ ar pan.).; skiriasi nuo ankstesnio.

VIISP Valstybės Informacinių Išteklių Sąveikumo Platforma, prieiname per Elektroninius Valdžios Vartus

El. Vartai Elektroninių valdžios vartų puslapis

Įstaiga Organizacija / institucija, vykdanti nustatytas veiklas

Sąrašo rūšiavimas Sąrašą galima rikiuoti pagal bet kurį iš stulpelių: spauskite pasirinkto stulpelio pavadinimą arba .

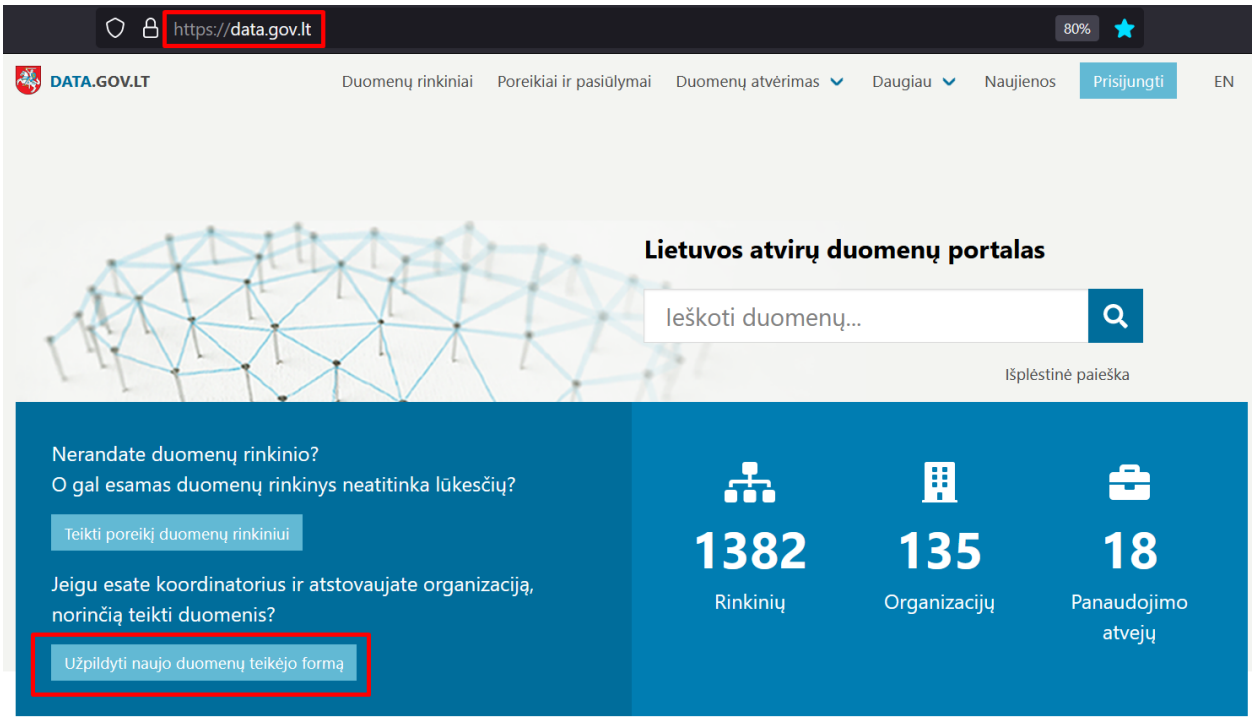
7.2 Koordinatoriaus registravimas

Portale reikia registruoti savo instituciją ir vieną atstovaujantį koordinatorių.

Koordinatorius paskiria atvirų duomenų rinkinių tvarkytojus.

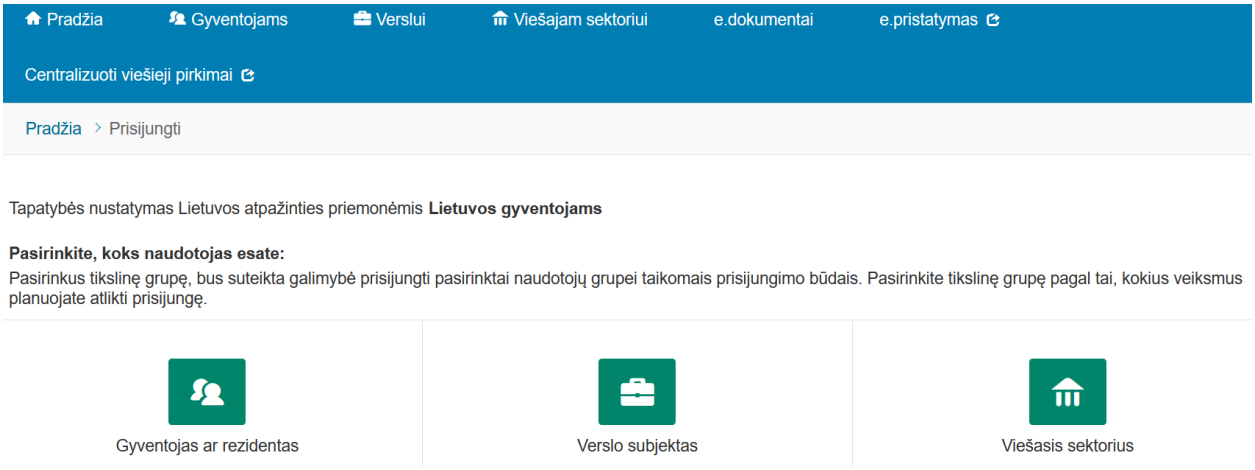
Prieš pradedant registraciją, siūlome turėti vadovo parašu patvirtintą koordinatoriaus skyrimo raštą.

1. Naršyklėje atsidarykite Portalo puslapį https://data.gov.lt.
2. Pagrindiniame lange spauskite **[Užpildyti naujo duomenų teikėjo formą]**:



1 pav. Portalo pagrindinio lango viršutinis fragmentas

Būsite nukreipti į El. Vartų puslapį, kur pateikiami prisijungimo būdai:



2 pav. El.Vartų prisijungimo puslapio fragmentas

Jeigu koordinatoriumi yra skiriamas:

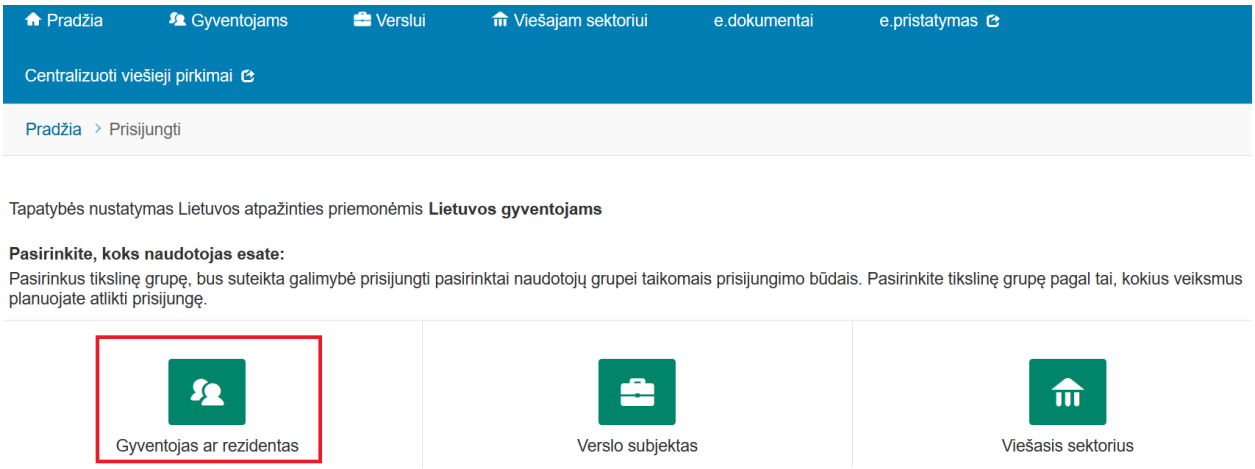
- Valstybės tarnautojas: Registruokitės pasirinkę „Gyventojas“ arba „Viešasis sektorius“
- Darbuotojas, dirbantis pagal darbo sutartį: Registruokitės pasirinkę „Gyventojas“.

Siūlome registruotis pasirinkus „Gyventojas“.

7.2.1 Registracija pasirinkus „Gyventojas“

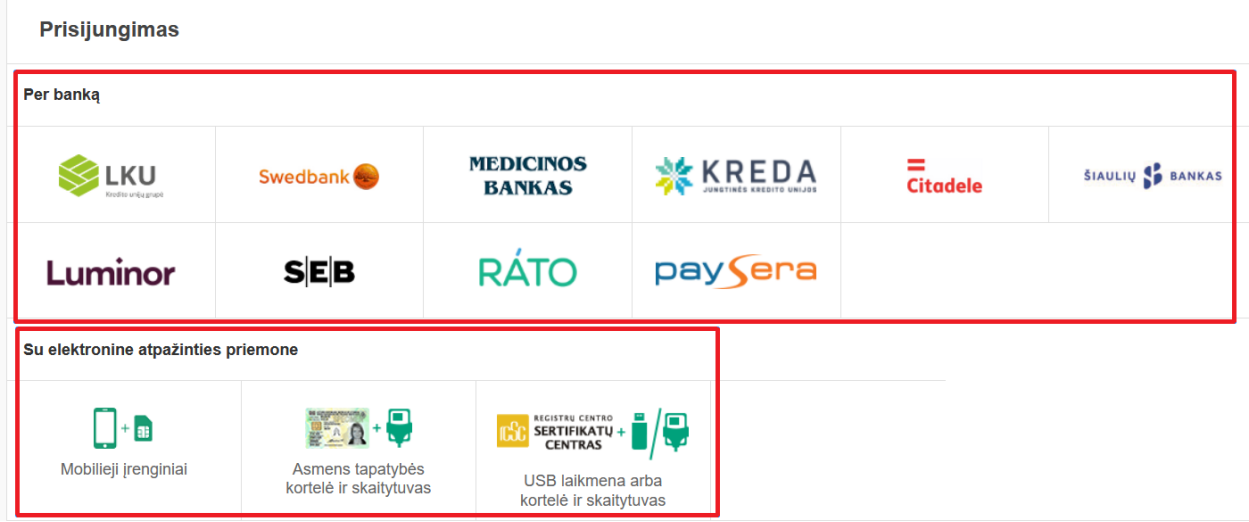
El.valdžios vartų puslapyje:

1. Paspauskite „Gyventojas ar rezidentas“.



3 pav. El.Vartų naudotojo pasirinkimo lango fragmentas, kai pasirinkta „Gyventojas ar rezidentas“

2. Atsidariusiame lange pasirinkite registraciją „Per banką“ arba „Su elektroninės atpažinties priemone“.

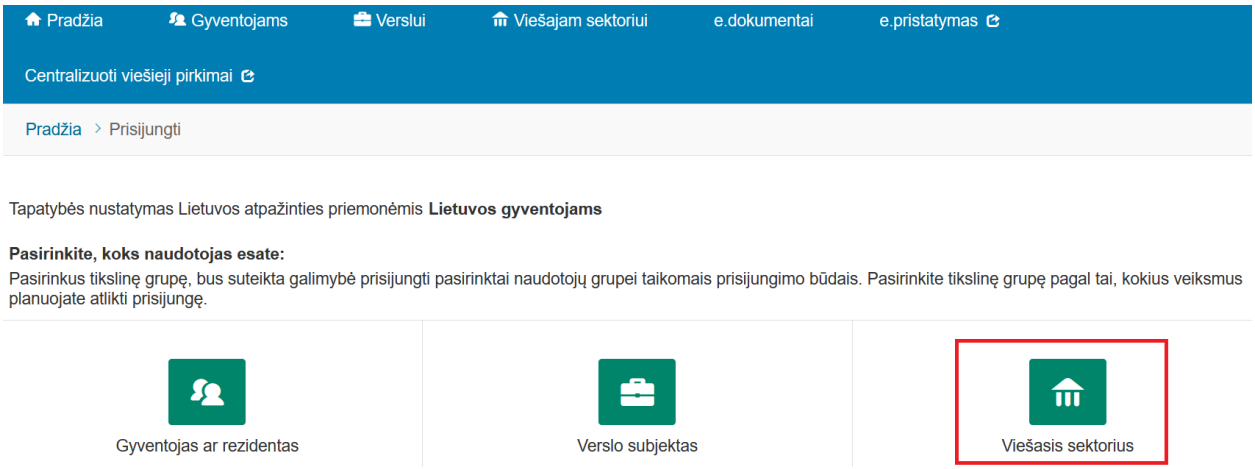


4 pav. Lango "Prisijungimas" fragmentas

3. *Siūlome rinktis variantą „Per banką“:* paspauskite savo banko ikoną, suvesti el. bankininkystės prisijungimo duomenis.

7.2.2 Registracija pasirinkus „Viešasis sektorius“

1. Paspauskite „Viešasis sektorius“.

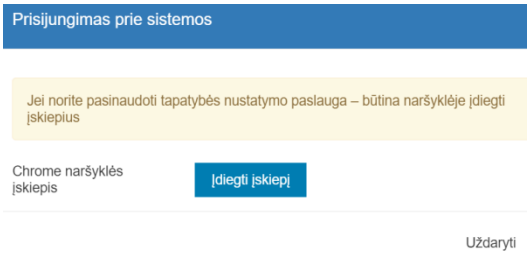


5 pav. El.Vartų naudotojo pasirinkimo lango fragmentas, kai pasirinkta "Viešasis sektorius"

2. Atsidariusiame lange spauskite „Valstybės tarnautojo pažymėjimas ir skaitytuvas“.

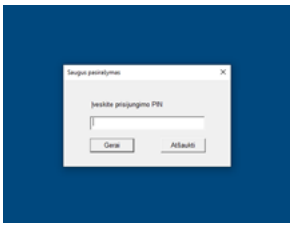
6 pav. Prisijungimo priemonės nustatymo lango fragmentas

3. Jei reikia – įdiekite įskiepį naršyklei:



7 pav. Prisijungimo priemonės įskiepio diegimo pranešimo langas

4. Suveskite reikiamus identifikavimo duomenis:



8 pav. Pasirašymo su PIN lango fragmentas

7.2.3 Identifikavus tapatybę

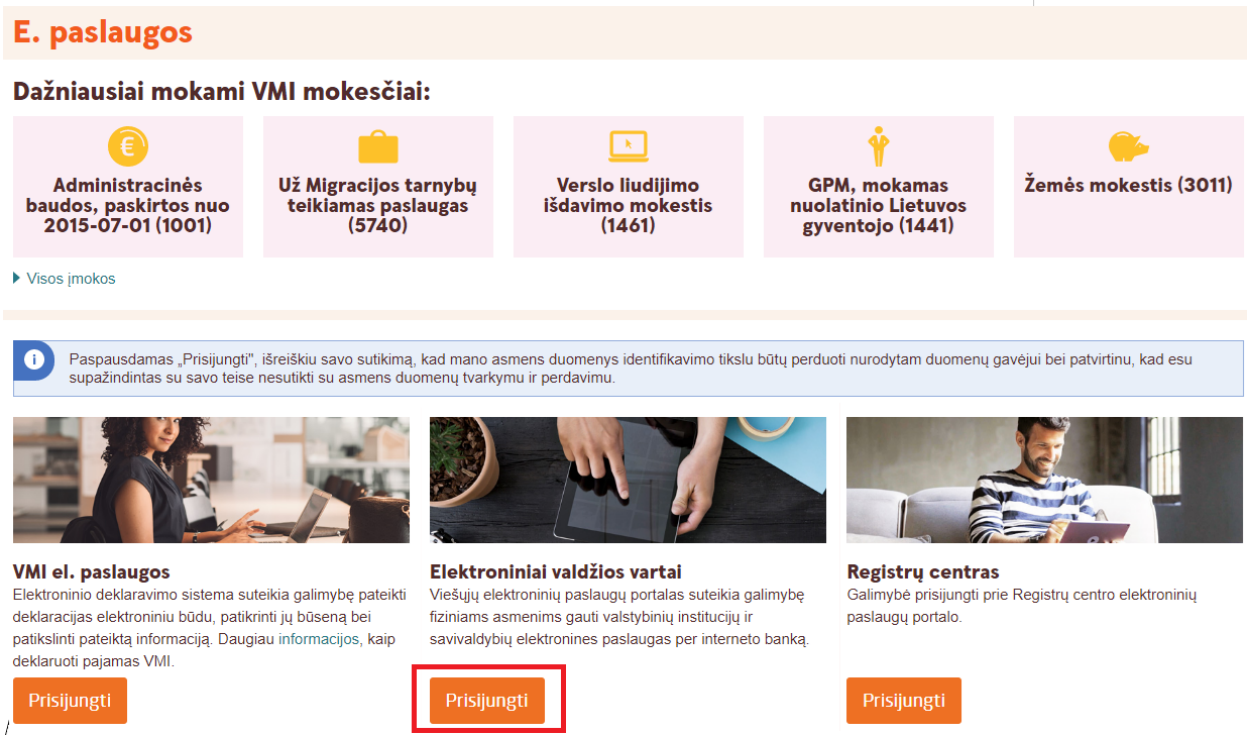
Tiek registruojantis kaip „Gyventojas“, tiek kaip „Viešasis sektorius“, po tapatybės nustatymo būsite nukreipti į puslapį „E. paslaugos“ arba į langą „Duomenų teikimas į Atvirų duomenų portalą“.

Priklausomai pagal atpažinimo pobūdį, atlikite veiksmus:

- *Nuo***Varianto (A)**, *jeigu buvote nukreipti į puslapį „E-paslaugos“,*
- *Nuo***Varianto (B)**, *jeigu buvote nukreipti „Duomenų teikimas į Atvirų duomenų portalą“.*

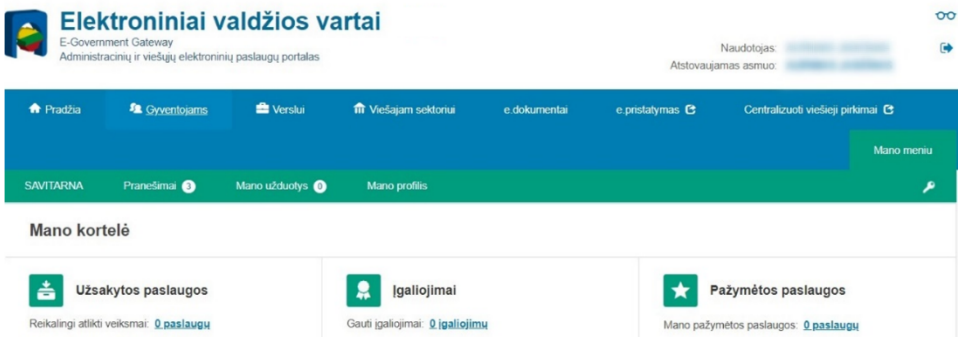
Variantas A:

1. Jeigu buvote nukreipti į E. paslaugos puslapį, spauskite „**Prisijungti**“.



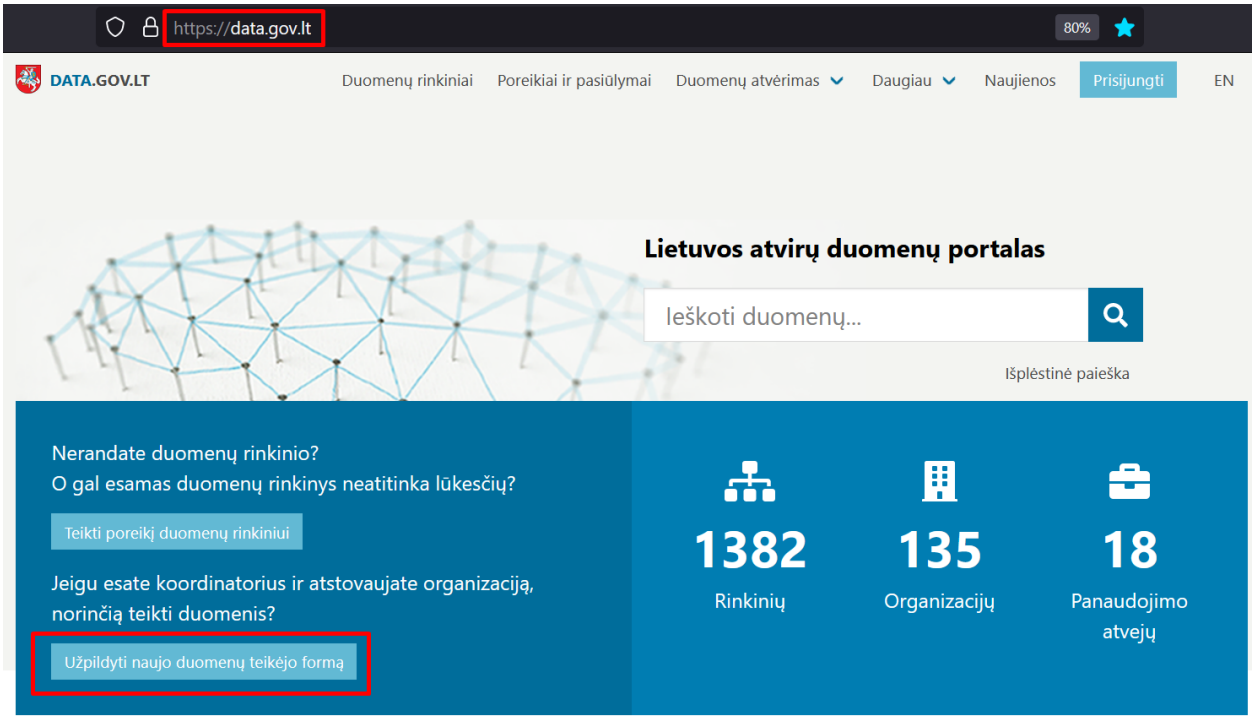
9 pav. Prisijungimo per elektroninį banką langas

Būsite nukreipti į El.Vartų puslapį „Mano kortelė“:



10 pav. El.Vartų puslapio „Mano kortelė“ lango fragmentas

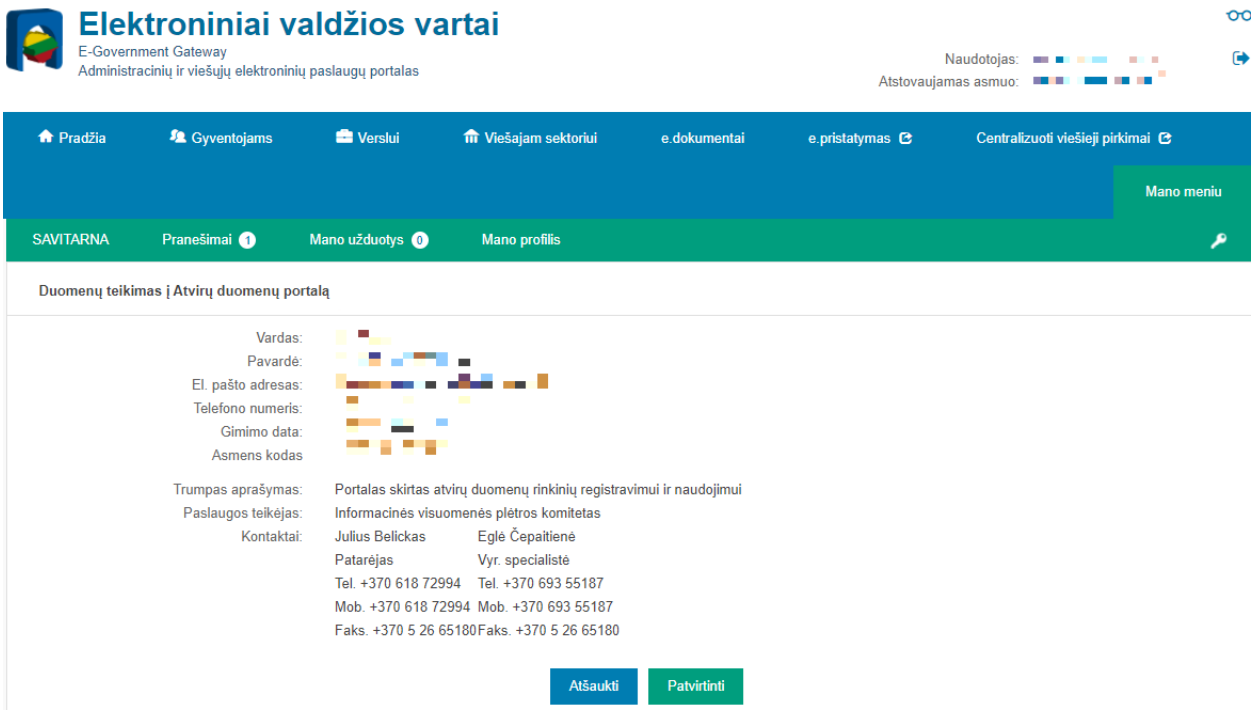
2. Naujame naršyklės lange atsidarykite Portalo puslapį https://data.gov.lt.
3. Pakartotinai spauskite **[Užpildyti naujo duomenų teikėjo formą]**:



11 pav. Portalo pradinio lango fragmentas

Variantas B:

Atsidarys El.Vartų langas „Duomenų teikimas į Atvirų duomenų portalą“: čia automatiškai surenkami ir atvaizduojami registruojamo koordinatoriaus duomenys: vardas, pavardė, kontaktai.



12 pav. El.Vartų langas „Duomenų teikimas į Atvirų duomenų portalą“

> **Atsisakyti patvirtini duomenis:** spauskite **[Atšaukti]**.

> **Patvirtinti informaciją:** spauskite **[Patvirtinti]**.

4. Atsidarys „**Duomenų teikėjo registracijos**“ forma:

13 pav. Duomenų teikėjo registracijos lango fragmentas

5. Pateikite *organizacijos atvirų duomenų koordinatoriaus* informaciją (* – privaloma):

- **Organizaciją** * : pasirinkite iš sąrašo;

  Jeigu organizacijos nėra sąraše, spauskite **[Pridėti organizaciją]**;

- **El. pašto adresą** *;
- **Telefoną** *;
- Įkelkite institucijos vadovo (ar įgalioto darbuotojo) pasirašytą **koordinatoriaus paskyrimo raštą** *.

  Rašto formatas: skaitmeninė kopija (.pdf), el. parašu pasirašytas (.adoc), arba e-parašu pasirašyto dokumento nuorašas (.pdf).

  Rašto pavyzdį rasite po nuoroda „orgKoordinatoriausPaskyrimoRastas.doc“

6. Spauskite **[Registruoti duomenų teikėją]** formos apačioje.

Atsakymo žinutę gausite per keletą dienų nurodytu el. paštu.

Jei registracija – patvirtinta:

- *Žinutėje nurodomi koordinatorius prisijungimo duomenys – el. paštas ir slaptažodis;*
- *Dabar prie portalo galite prisijungti kaip koordinatorius!*

Jei registracija – atmesta:

- *Žinutėje nurodoma atmetimo priežastis, pvz., „raštas nėra tinkamai užpildytas“ ar „trūksta parašo“.*

7.2.4 Prisijungimas prie sistemos

1. Naršyklėje atidarykite puslapį https://data.gov.lt/login:

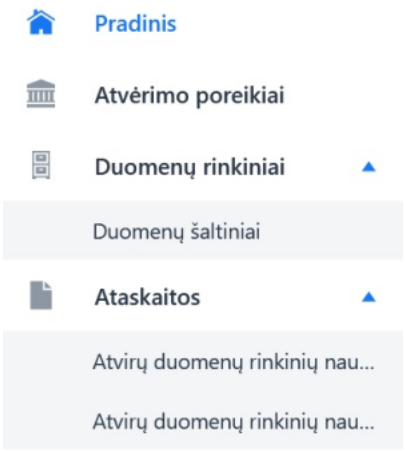
14 pav. Prisijungimo langas

2. Įveskite **[El. paštas]** ir **[Slaptažodis]**;
3. Spauskite **[Prisijungti]**.

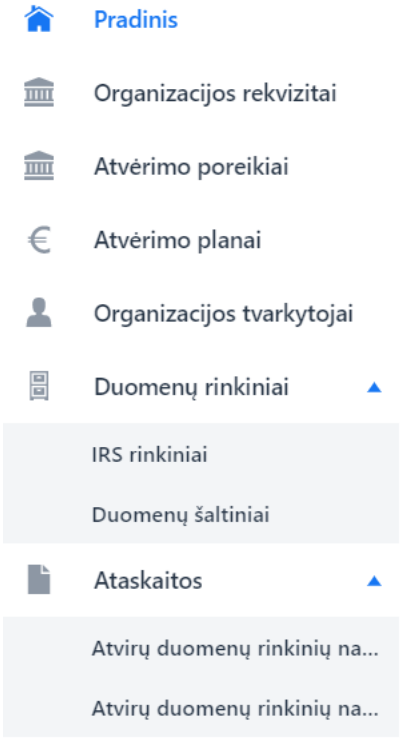
7.3 Pagrindinis meniu

Pagrindinis meniu matomas prisijungus prie sistemos ir leidžia pasiekti aplinkos atributus:

- **Pagrindinis:** pirmasis puslapis rodomas po prisijungimo;
- **Atvėrimo poreikiai:** atvėrimo poreikių sąrašas;
- **Atvėrimo planai:** metiniai duomenų rinkinių atvėrimo planai;
- **Organizacijos rekvizitai:** duomenys apie pačią organizaciją;
- **Organizacijos tvarkytojai:** priskirtus tvarkytojų duomenys;
- **Duomenų rinkiniai:** duomenų rinkinių informacija [Meniu];
- **Ataskaitos:** ataskaitų ruošimo langas [Meniu];
- *Kiti skyriai, reikalingi Koordinatorių arba Tvarkytojų darbui.*



15 pav. Pagrindinis meniu Tvarkytojams



16 pav. Pagrindinis meniu Koordinatoriams

> **Išskleisti meniu:** Pelyte užveskite ant meniu juostos, o tada **> Išskleisti**.

> **Suskleisti meniu:** Spauskite **< Suskleisti** meniu apačioje.

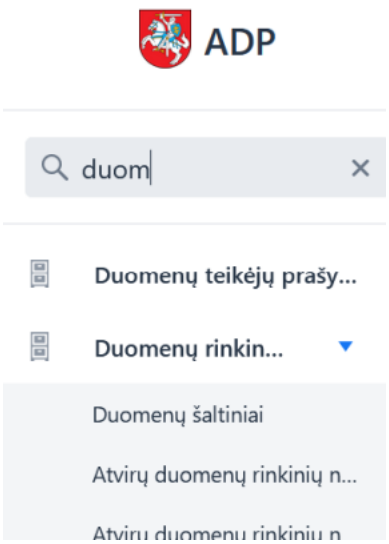
> **Išskleisti meniu lauką:** Spauskite [ ].

> **Suskleisti meniu lauką:** Spauskite [ ].

7.4 Paieška

Kairėje ekrano pusėje, meniu viršuje – tekstinis laukas skirtas meniu laukų paieškai.

1. Į tekstinį lauką įveskite pavadinimą, pilną arba fragmentą;
2. Spauskite **[Enter]**.

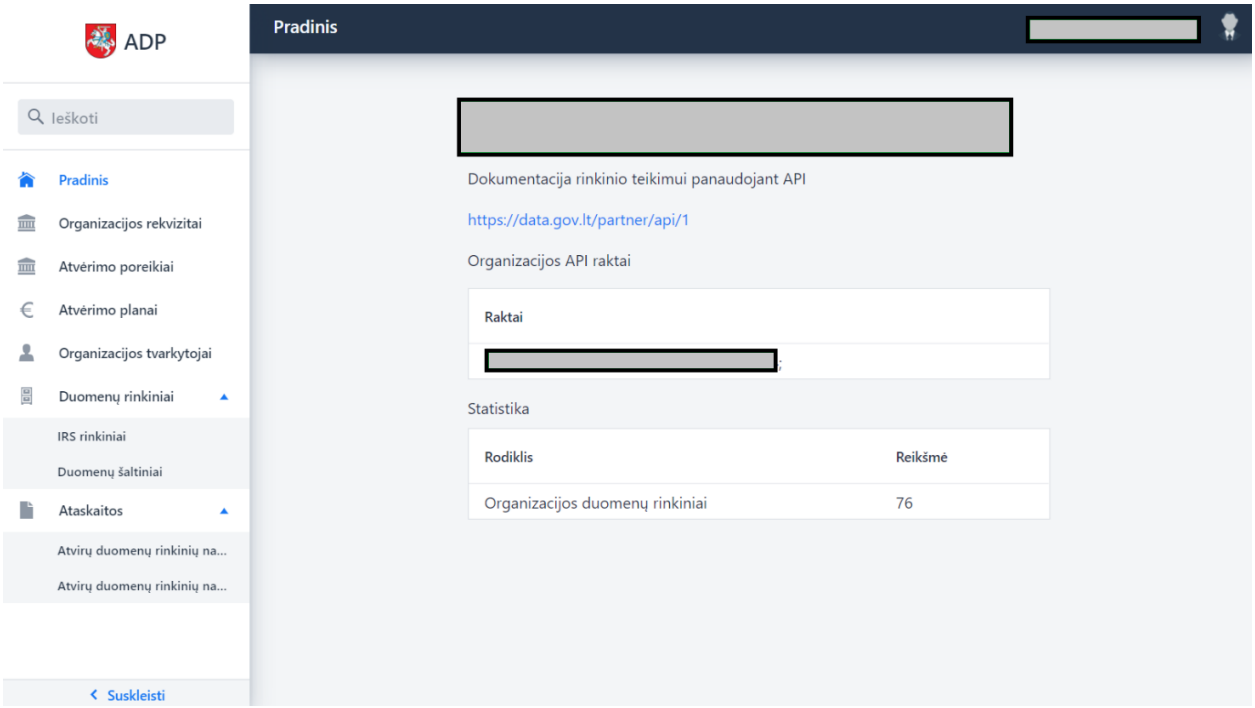


17 pav. Pagrindinio meniu laukų paieškos rezultatų pavyzdys

3. Paiešką atitinkantys variantai bus atvaizduoti meniu lauke.
4. Vėl norėdami matyti pilną meniu, spauskite ×.

Pradinis langas matomas:

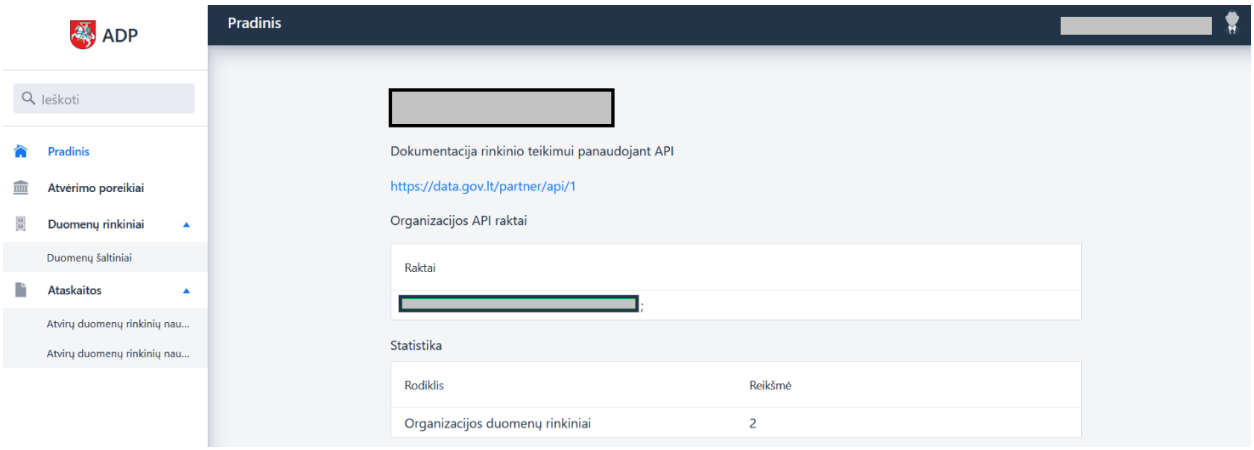
- *prisijungus prie paskyros,*
- *pagrindiniame meniu paspaudus punktą* **„Pradinis“**.



18 pav. Organizacijos koordinatoriaus aplinkos pradinio lango fragmentas

Lange pateikiama lentelė, kurioje pateikta informacija apie organizacijos statistiką:

- **Organizacijos:** skaičius organizacijų, Portale pateikusių duomenų rinkinius;
- **Organizacijos duomenų rinkiniai:** skaičius rinkinių Portale, priskirtų organizacijai;
- **Distribucijos**: į Portalą įkeltų duomenų rinkinių pateikimo distribucijos failų skaičius;
- **Portalo naudotojai:** Portalo registruotų naudotojų paskyrų skaičius.

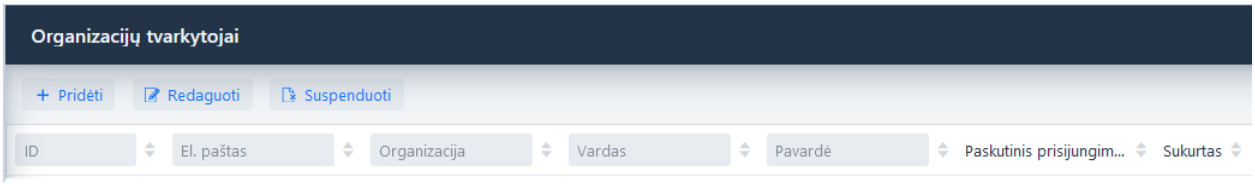


19 pav. Organizacijos tvarkytojo aplinkos pradinio lango fragmentas

7.5 Duomenų rinkinių tvarkytojai

Šis skyrius skirtas tik Koordinatoriams.

1. Pagrindiniame meniu išskleiskite „Naudotojai“ ir pasirinkite **„Organizacijų tvarkytojai“**:



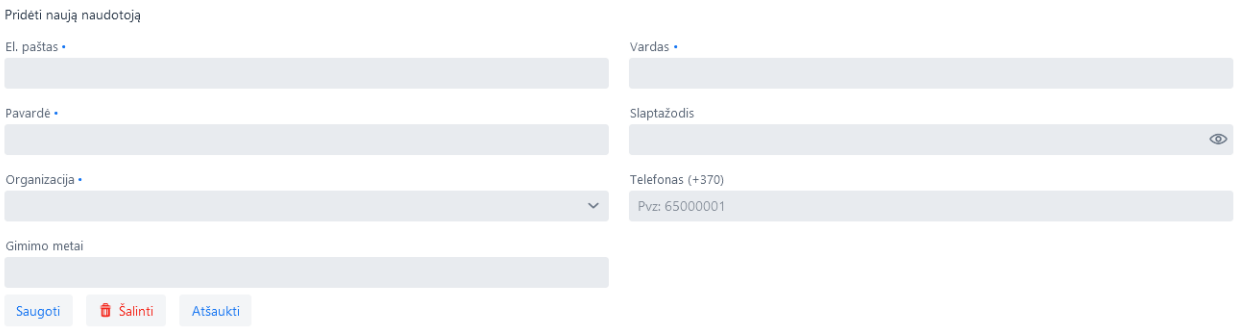
20 pav. Organizacijos tvarkytojų paskyrų sąrašo lango meniu

Sąrašo duomenis:

- **ID:** unikalus paskyros identifikatorius sistemoje. Galima filtruoti ir rikiuoti.
- **El. paštas:** el. paštas, kuriuo registruotos paskyros. Galima filtruoti ir rikiuoti.
- **Organizacija:** teikia duomenis. Galima filtruoti ir rikiuoti.
- **Vardas, Pavardė:** paskyrų naudotojų vardų ir pavardžių stulpeliai. Galima filtruoti ir rikiuoti.
- **Paskutinis prisijungimas:** Data ir laikas, kada naudotojas buvo paskutinį kartą prisijungęs prie savo paskyros. Galima rikiuoti.
- **Sukurtas:** paskyros sukūrimo data ir laikas. Galima rikiuoti.

> Sukurti naują savo organizacijos AD tvarkytojo paskyrą:

1. Spauskite **[+ Pridėti]** ir užpildykite registracijos langą ir išsaugokite pakeitimus:



21 pav. Langas pridėti naują organizacijos tvarkytoją

> Redaguoti paskyras:

1. Sąraše pažymėkite reikiamą paskyrą ir lango meniu spauskite **[Redaguoti]**:

22 pav. Langas pridėti naują organizacijos tvarkytoją

> Panaikinti paskyrą:

- Sąraše pažymėkite paskyrą ir spauskite **[Suspenduoti]**.

  Atsidaro papildomas langas, kuriame pasirinkite naudotoją, kuriam bus priskirti duomenys:

23 pav. Langas pašalinti pasirinktą paskyrą

> Filtruoti naudotojų sąrašą:

1. Pasirinkite lauką iš sąrašo;
2. Filtruoti galima pagal kelis laukus vienu metu;
3. Rikiuoti galima vienu metu tik pagal vieną kurį nors stulpelį.

7.6 Organizacijos rekvizitais

Šis skyrius skirtas tik Koordinatoriams.

> Redaguoti savo organizacijos pateikiamus aplinkoje duomenis:

1. Pagrindiniame meniu pasirinkite **„Organizacijos“**.
2. Pasirinkite organizaciją, kurios informaciją norėsite redaguoti. Spauskite **[Redaguoti]**.
3. Rekvizitų lange redaguokite reikiamus duomenis.

4. Įsitikinę, kad tinkamai užpildėte/redagavote laukus, spauskite [✓ Saugoti] lango apačioje.

5. Norėdami ištrinti pasirinktą organizaciją, spauskite [✕ Šalinti] lango apačioje.

24 pav. Rekvizitų redagavimo langas

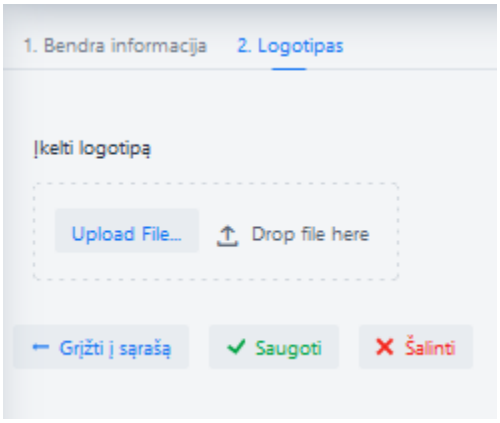
Skiltyje **„1. Bendra informacija“** galima redaguoti laukus:

- **Pavadinimas:** pilnas organizacijos pavadinimas;
- **Įmonės kodas:** juridinio asmens kodas;
- **El. pašto adresas:** kontaktinis organizacijos el. pašto adresas;
- **Adresas:** registruotas organizacijos adresas;
- **Telefono numeris:** kontaktinis organizacijos telefono numeris;

- **Tinklalapis:** nuoroda į organizacijos oficialią svetainę;
- **Regionas:** iš sąrašo pasirenkamas apskrities, kurioje registruota organizacija, pavadinimas;
- **Savivaldybė:** iš sąrašo pasirenkamas savivaldybės, kuriai priklauso organizacija, pilnas pavadinimas;
- **Ministrų valdymo sritis:** ministerija, su kuria susijusi organizacija.

Spauskite [ ] lauko dešinėje pusėje ir išplėsite sąrašą.

Skiltyje **„2. Logotipas“** galite įkelti naują arba pakeisti įkeltą organizacijos logotipą:



25 pav. Rekvizitų redagavimo langas

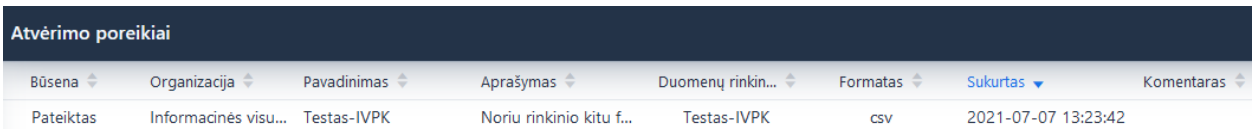
7.7 Prašymai gauti duomenis

Šis skyrius skirtas tik Koordinatoriams.

Portalo naudotojams pateikus arba vyr. koordinatoriui priskyrus poreikį Jūsų organizacijai, gausite pranešimą apie naują poreikį į Jūsų paskyrai užregistruotą el. paštą.

7.7.1 Poreikių sąrašo peržiūra

1. Pagrindiniame menu pasirinkite **„Atvėrimo poreikiai“**.



26 pav. Poreikių sąrašo langas

Bendrame organizacijai pateiktų duomenų atvėrimo poreikių sąraše – pagrindinė poreikių informacija:

- **Būsena:** organizacijai pateikto poreikio atvėrimo būsena. Galimos reikšmės:
  - „Pateiktas“, jei organizacijos vardu dar nebuvo atsakyta į poreikį;

  - „Planuojama atverti“, jei poreikis patvirtintas;
  - „Nenumatytas atverti“, jei poreikis atmestas;
- **Organizacija:** kuriai pateiktas duomenų atvėrimo poreikis;
- **Pavadinimas:** duomenų rinkinio, kuriam pateiktas poreikis atverti, pavadinimas;
- **Aprašymas:** trumpas duomenų rinkinio, kuriam pateiktas poreikis atverti, aprašymas;
- **Duomenų rinkinio ID:** duomenų rinkinio unikalus identifikatorius sistemoje.

  Pateikiamas, jei poreikis skirtas jau atvertą rinkinį atnaujinti;
- **Formatas:** pageidaujamas duomenų pateikimo formatas;
- **Sukurtas:** Data ir laikas, kada poreikis buvo sukurtas viešame Portale;
- **Komentaras:** koordinatoriaus/ tvarkytojo atsakymas su atvėrimo poreikio priežastimi.

Įsijungus puslapį, sąrašas pateikiamas automatiškai surikiuotas pagal sukūrimo datą, pateikiant naujausios datos poreikius viršuje.

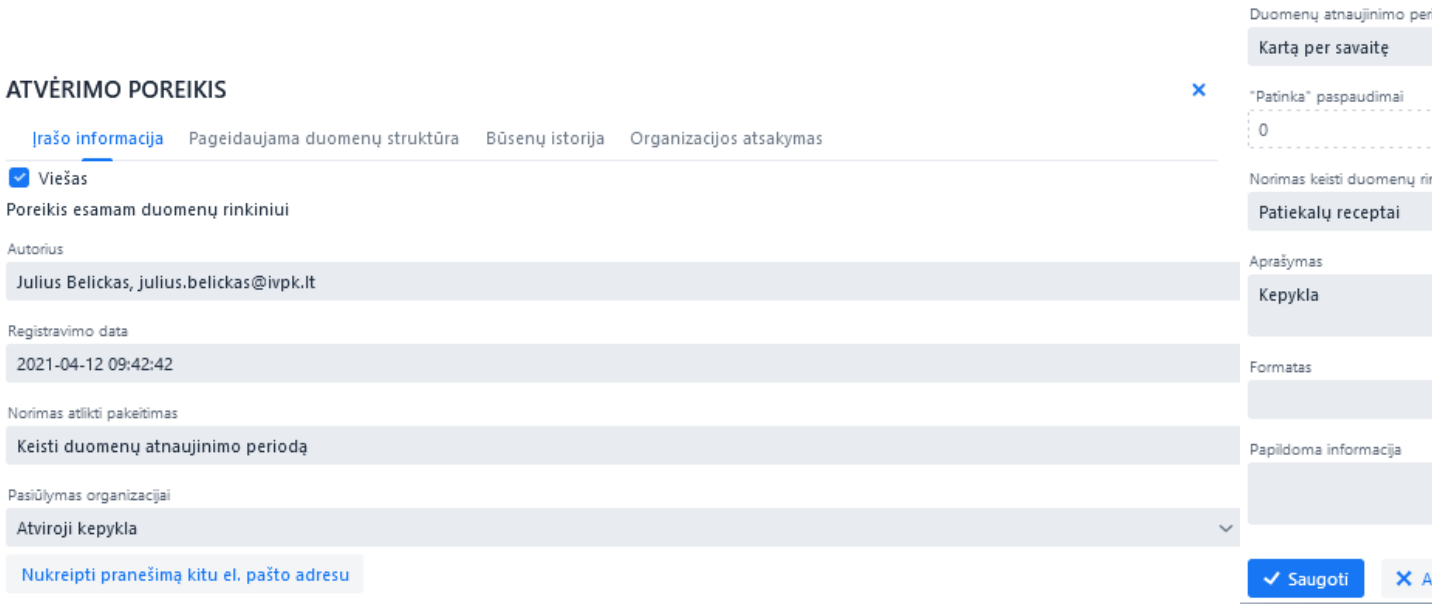
7.7.2 Atvėrimo poreikio peržiūra

Su sąraše pateiktais poreikiais galite:

- *Peržiūrėti išsamesnę informaciją;*
- *Redaguoti kai kuriuos poreikių duomenis;*
- *Atsakyti į poreikį.*

> Peržiūrėti išsamesnę pateikto poreikio informaciją:

1. Sąraše pasirinkite poreikį ir du kartus paspauskite ant poreikio.
2. Atverto lango viršuje pateiktas meniu, kuriuo naudojantis pasiekiamos atvėrimo poreikio skiltys. Melsvai paryškinta ta skiltis, kurioje tuo metu esate. Numatytoji atidaryti skiltis yra **„Įrašo informacija“**:



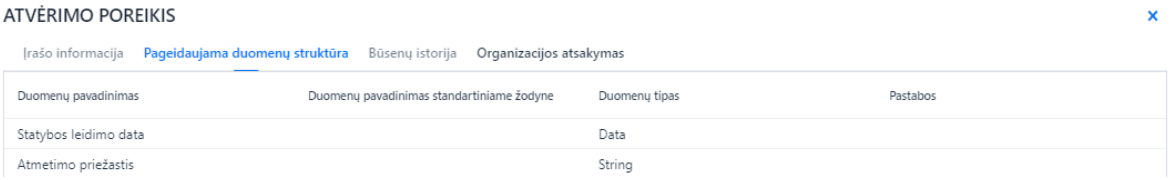
27 pav. Atvėrimo poreikio lango meniu

Šiame lange pateikta pagrindinė poreikio informacija, pateikta poreikio teikėjo:

- **Autorius**: poreikio teikėjo vardas, pavardė ir el. paštas;
- **Registravimo data**: data ir laikas, kada poreikis buvo pateiktas;
- **Norimas atlikti pakeitimas:** įvardinama, koks konkrečiai pakeitimas turėtų būti įvykdytas esamam duomenų rinkiniui. Poreikio teikėjas gali pasirinkti nuo vieno iki trijų pakeitimų iš pateikto sąrašo;
- **Pasiūlymas organizacijai:** poreikio teikėjas gali pateikti pasiūlymą organizacijai;
- **Duomenų atnaujinimo periodiškumas:** nurodomas periodas, kas kiek laiko turi būti atnaujinti rinkinyje pateikti duomenys;
- **Patinka paspaudimai:** neredaguotinas laukas, kuriame nurodyta, kiek viešos aplinkos naudotojų paspaudė „Patinka“ prie poreikio. Į „Patinka“ paspaudimų skaičių atsižvelgiama vertinant poreikio prioritetą;
- **Norimas keisti duomenų rinkinys:** pavadinimas jau esamo duomenų rinkinio, kuriam atnaujinti teikiamas poreikis;
- **Aprašymas:** duomenų rinkinio ir poreikio aprašymas. Pildo *poreikio teikėjas*;
- **Formatas:** kokias formatais atveriamas duomenų rinkinys. Laukas neredaguotinas. Pildo *poreikio teikėjas*;
- **Papildoma informacija:** papildoma informacija apie duomenų rinkinį ar poreikį. Pildo *poreikio teikėjas*.

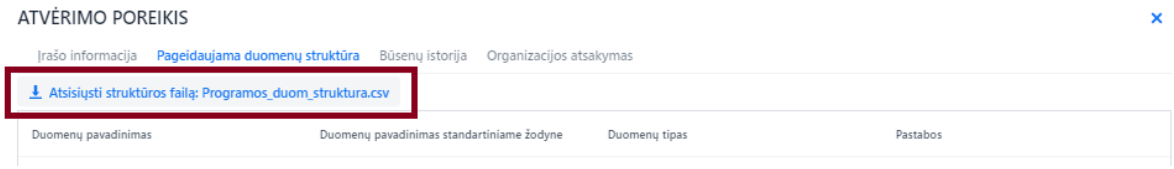
> Peržiūrėti poreikio teikėjo pateiktą struktūrą:

1. Pagrindiniame meniu pasirinkite **„Atvėrimo poreikiai“;** du kartus paspauskite ant pasirinkto poreikio ir atsidarytų papildomas meniu „Atvėrim poreikis“, pasirinkite „**Pageidaujama duomenų struktūra**“:



28 pav. Pageidaujamos duomenų rinkinio struktūros fragmentas

2. Pageidaujama duomenų struktūra aprašoma šiuose laukuose (*: privalomi laukai):
   - **Duomenų pavadinimas * :** Lauko, kuriame pateikiamas tam tikro tipo duomuo, pavadinimas;
   - **Duomenų pavadinimas standartiniame žodyne:** duomens pavadinimas, atitinkantis lietuvių kalbos taisykles.
   - **Duomenų tipas * :** duomenų tipas, įrašomas anglų kalba.
   - **Pastabos:** laisvai įvedamos pastabos apie duomenis.
3. Duomenų teikėjas duomenų rinkinio struktūrą gali įkelti kaip Excel failą: atveju lango viršuje patiekiama nuoroda parsisiųsti failą.



29 pav. Reikiamos duomenų struktūros failo atsisiuntimo nuoroda poreikio lange

SVARBU: pakeisti pageidaujamą duomenų struktūrą gali tik poreikio teikėjas.

> Peržiūrėti atvėrimo poreikio būsenų pasikeitimus:

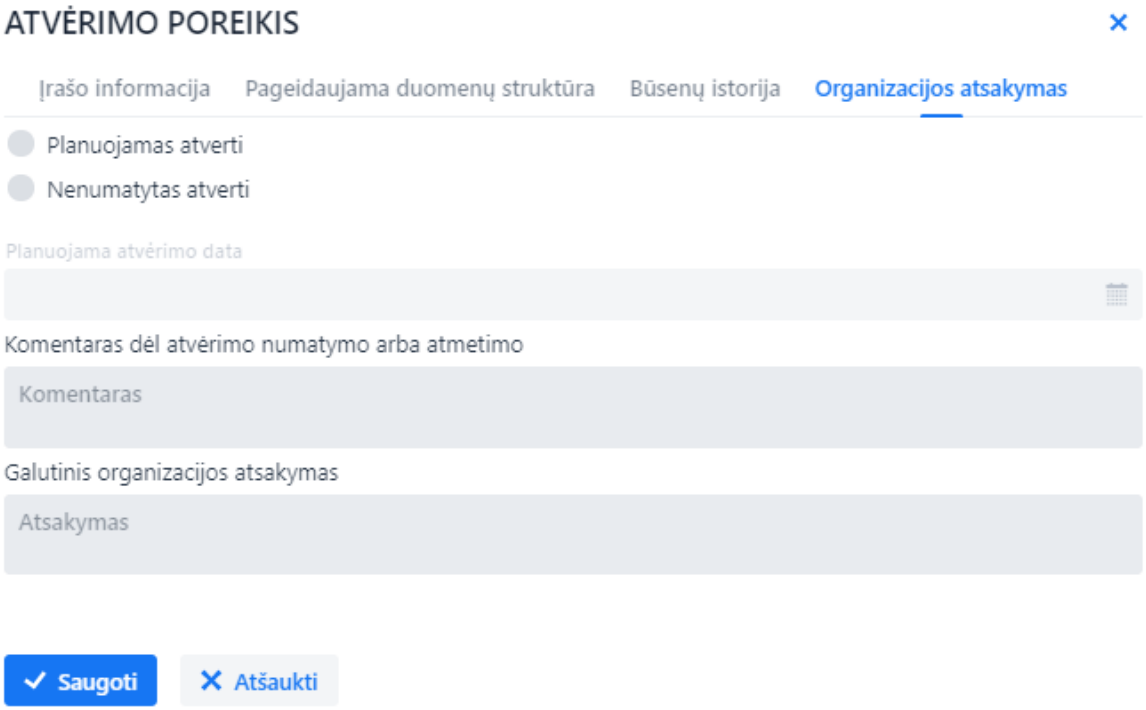
1. Pagrindiniame meniu pasirinkite **„Atvėrimo poreikiai“;** du kartus paspauskite ant pasirinkto poreikio, kad atsidarytų papildomas meniu „Atvėrimo poreikis“, kur pasirinkite „**Būsenų istorija**“.

30 pav. Atvėrimo poreikio lango skiltis „Būsenų istorija“

2. Sąraše atvaizduojami duomenys:
   - **Sukurtas:** Data ir laikas, kada buvo atliktas veiksmas
   - **Būsena:** Veiksmo, kuris buvo atliktas, pavadinimas. Galimos reikšmės
     - „Sukurtas“
     - „Redaguotas“
     - „Patvirtintas“
     - „Atmestas“.
   - **Autorius:** paskyros savininko, atlikusio veiksmus, vardas ir pavardė.
   - **Komentaras:** Komentaras, kuris gali būti paliekamas atsakant į poreikį.

7.7.3 Atsakymas į pateiktą atvėrimo poreikį

1. Pagrindiniame meniu pasirinkite **„Atvėrimo poreikiai“;** du kartus paspauskite ant pasirinkto poreikio, kad atsidarytų papildomas meniu „Atvėrimo poreikis“, kur pasirinkite „**Organizacijos atsakymas**“.



31 pav. Atvėrimo poreikio lango skilties „Organizacijos atsakymas“ pavyzdys

2. Užpildykite atsakymo formos laukus:
   - Pasirinkite atsakymą pažymint vieną iš atsakymo laukų:
     - „Planuojamas atverti“, jei poreikis priimamas ir rinkinys planuojamas atverti.
     - „Nenumatytas atverti“, jei tuo metu duomenų rinkinys nenumatytas atverti.
   - Jei rinkinys planuojamas atverti, lauke **„Planuojama atvėrimo data“** įveskite numatytą rinkinio atvėrimo datą formatu „mm/dd/mmmm“ arba pasirinkę kalendoriuje, spausdami **[ ]** dešinėje.
   - Tekstiniame lauke **„Komentaras dėl atvėrimo numatymo arba atmetimo“** įrašykite komentarą dėl atsakymo. Ši informacija bus pateikta viešame portale bei matoma atvėrimo poreikių sąrašo lange.
   - Tekstiniame lauke **„Galutinis organizacijos atsakymas“** įrašykite atsakymą dėl atvėrimo ir priežastį. Ši informacija bus pateikta poreikio teikėjui.

Spauskite mygtuką **[Saugoti]**, norėdami išsaugoti atsakymą: atsakymas į poreikį ir jo priežastis bus automatiškai nusiųsta poreikio teikėjui.

Poreikio patvirtinimo būsena po atsakymo gali būti pakeista pagal poreikį.

7.8 Duomenų rinkiniai

Duomenys į Portalą keliami dviem būdais:

1. Kuriant naują duomenų rinkinį;

   Plačiau: *Naujo duomenų rinkinio sukūrimas ir inventorinimas*;

2. Importuoti pagal šabloną užpildytą Excel failą;

   Plačiau: *Duomenų šablono atsisiuntimas*.

> Importavus duomenų rinkinį:

1. Koreguokite informaciją.

   Plačiau: *Duomenų distribucijos tvarkymas*;

2. Pereikite prie veiksmų su rinkiniu.

   Plačiau: *Duomenų rinkinio struktūros sukūrimas*.

> Išsaugoti įvestą informaciją:

1. Atitinkamoje kortelėje spauskite **[Saugoti]**.

> Atšaukti įvestų pakeitimų saugojimą:

1. Jei pakeitimų atsisakote, spauskite **[Grįžti į sąrašą]**.

7.8.1 Duomenų rinkinių sąrašo peržiūra

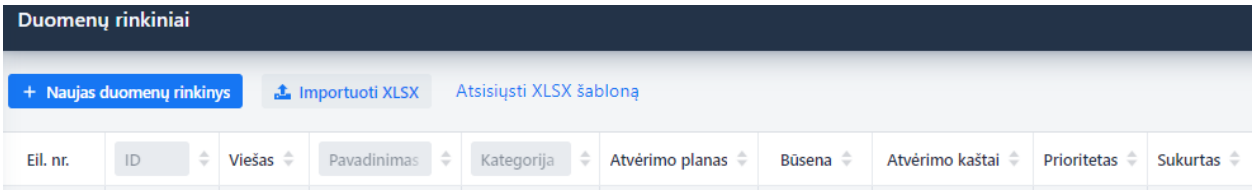
Organizacijos pačios kuria ir pildo joms priklausančius duomenų rinkinius.

Jei duomenų rinkinys perkeliamas iš vienos organizacijos kitai, pirmoji netenka prieigos prie jo.

Duomenų rinkinių sąrašo lange galite:

- *kurti naujus duomenų rinkinius;*
- *atsisiųsti duomenų pateikimo šabloną;*
- *importuoti duomenis;*
- *peržiūrėti sukurtų duomenų rinkinių informaciją.*

1. Pagrindiniame meniu spauskite **[Duomenų rinkiniai]**



32 pav. Duomenų rinkinių sąrašo langas

2. Sąraše pateikiama pagrindinė duomenų rinkinių informacija:

- **Eil. Nr.**: duomenų rinkinio eilės numeris duomenų rinkinių sąraše;
- **ID:** duomenų rinkinio unikalus identifikatorius sistemoje. Galite filtruoti sąrašą: įveskite skaitinę reikšmę;

- **Viešas:** duomenų rinkinio vaizdavimo viešoje aplinkoje būsena. Jei pateiktas varnelė [], rinkinys – pateiktas viešoje aplinkoje. Jei lauke pateiktas tuščias apskritimas [], rinkinys matomas tik administracinėje aplinkoje;
- **Pavadinimas:** duomenų rinkinio pavadinimas; galite filtruoti;
- **Kategorija:** duomenų rinkiniui priskirta kategorija. galite filtruoti ;
- **Atvėrimo planas:** metai, į kurių planą įtrauktas rinkinys;
- **Būsena:** duomenų rinkinio kūrimo ir pildymo būsena;
- **Atvėrimo kaštai:** Suma eurais, reikalinga atverti duomenų rinkinį;
- **Prioritetas:** duomenų rinkinio prioriteto skaitinė reikšmė;
- **Sukurtas:** data ir laikas, kada duomenų rinkinys buvo sukurtas.

> Filtruoti rinkinių sąrašą:

1. Įveskite lauko vertę arba jo fragmentą į lauką pasirinkto stulpelio pavadinime;

> Rikiuoti sąrašą pagal stulpelį:

1. Prie pasirinkto stulpelio pavadinimo spauskite  .

> Keisti stulpelius vietomis:

1. Pasirinkite stulpelį ir pelyte jį nutempkite iki reikiamos pozicijos.

7.8.2 Duomenų šablono atsisiuntimas

Duomenų rinkinys, įkeliamas kaip failas, privalo atitikti šabloną.

1. Pagrindiniame meniu spauskite **[Duomenų rinkiniai]**.
2. Duomenų rinkinių lange spauskite mygtuką **[**Atsisiųsti XLSX šabloną**]**.
3. Į kompiuterį taip parsisiųsite Excel failą.
4. Pagal gautą šabloną sudarykite (arba užpildykite gautą) duomenų failą, kurį galėsite įkelti į ADP.

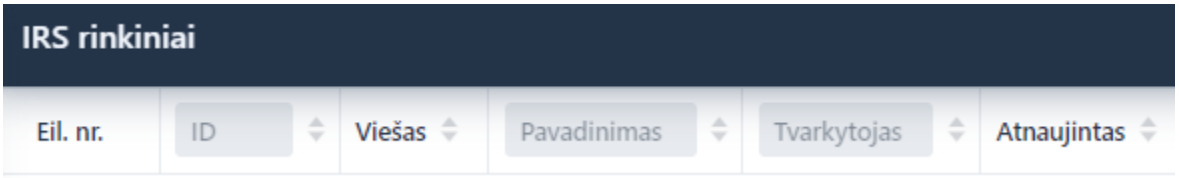
7.8.3 Duomenų rinkinio importavimas

1. Pagrindiniame meniu pasirinkite **„Duomenų rinkiniai“**;
2. Duomenų rinkinių sąrašo lange paspauskite **[Importuoti XLSX]**;

   Importuojant XLSX failą, įkeliamas tik aprašas. Inventorinimo, prioritetų, finansiniai ir meta- duomenys nėra įkeliami. Rinkinio metaduomenys įkeliami atskirai, po aprašymo failo įkėlimo.

3. Įkelkite failą iš kompiuterio sekdami įkėlimo lango nuorodas.

7.8.4 IRS rinkiniai



59 pav. IRS rinkinių sąrašo lango tvarkytojams fragmentas

IRS rinkinių sąraše galima peržiūrėti tokią bendrą informaciją:

- **Eil. nr.:** rinkinio numeris sąraše, sugeneruotas portale automatiškai;
- **ID:** unikalus rinkinio identifikatorius sistemoje, sugeneruotas automatiškai importuojant rinkinį.

  Galima filtruoti (skaitinis);

- **Viešas:** rinkinio viešumo portalo viešoje aplinkoje būsena. Laukas žymimas varnele [ ], jei rinkinys – viešas;
- **Pavadinimas:** pilnas duomenų rinkinio pavadinimas.

  Galima filtruoti (tekstinis);

- **Tvarkytojas:** Organizacijos tvarkytojo, atsakingo už rinkinį, vardas ir pavardė.

  Galima filtruoti (tekstinis);

  Gali būti tuščias;

- **Atnaujintas:** data ir laikas, kada rinkinys buvo paskutinį kartą atnaujintas.

> Redaguoti IRS rinkinį:

1. Paspauskite ant IRS rinkinio, kurį norite peržiūrėti/ tikslinti/ sutvarkyti eilutės:

60 pav. IRS rinkinių sąrašo langas

SVARBU: IRS rinkinį reikia pašalinti, jeigu duomenų rinkinys nebėra aktualus, arba jau yra įvesta naujas panašus duomenų rinkinys. IRS rinkinius galima redaguoti. Tikslinant IRS rinkinius galioja tos pačios taisyklės, kaip ir naujo duomenų rinkinio įvedimo atveju.

Tikslinant IRS rinkinius rekomenduotina pakeisti „5. Metaduomenų“ lauko „Katalogas“ informaciją: iš „Duomenys iš IRS“ į „Lietuvos“. Atlikus šį pakeitimą duomenų rinkinys bus matomas tik bendrame „Duomenų rinkiniai“ sąraše, o „IRS rinkiniai“ sąraše šio duomenų rinkinio nebus.

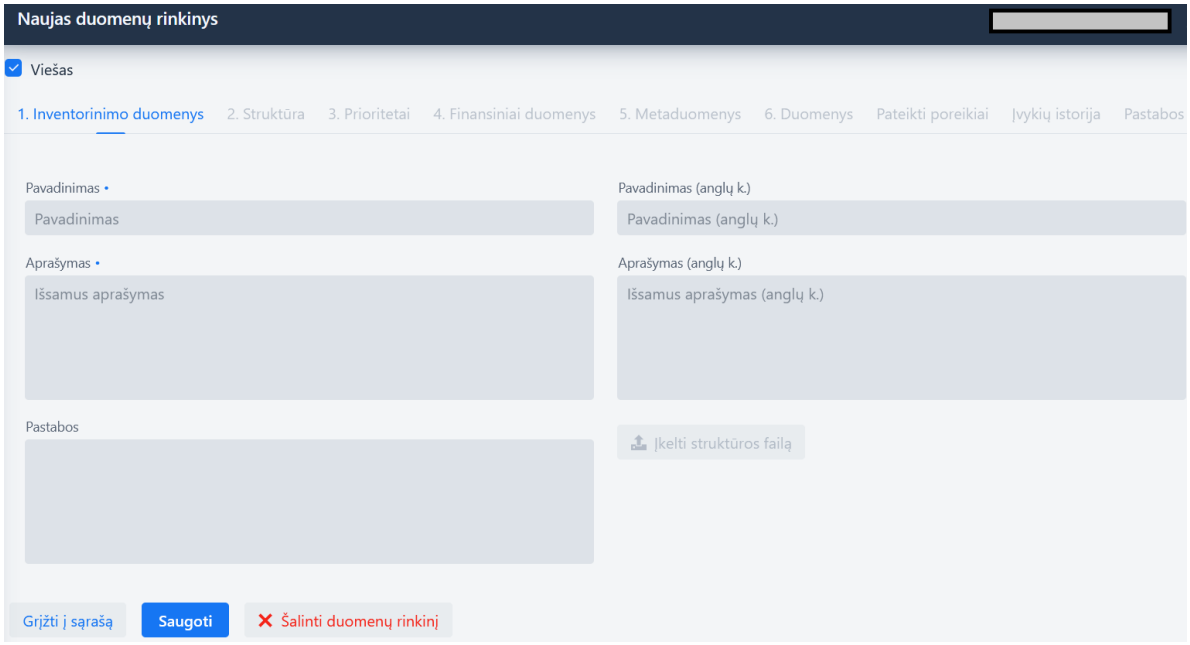
Sąrašas pildomas automatiškai, importuojant rinkinius iš atitinkamų portalų.

7.9 Duomenų rinkinio forma

7.9.1 Inventorizacija

1. Pagrindiniame meniu pasirinkite **[Duomenų rinkiniai]**.
2. Duomenų rinkinių sąrašo lange spauskite **[+ Naujas duomenų rinkinys]**.

   Pasirinkus duomenų rinkinį, atidaroma kortelė „1. Inventorinimo duomenys“.



33 pav. Duomenų rinkinio inventorinimo duomenų langas.

3. Užpildykite kortelę (*: privalomi laukai):
   - **Pavadinimas*:** pilnas duomenų rinkinio pavadinimas lietuvių kalba;

     Žymimas raudonai tol, kol užpildomas;
   - **Aprašymas*:** duomenų rinkinio aprašymas lietuvių kalba;

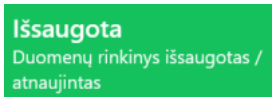
     Žymimas raudonai, kol neužpildomas;
   - **Pavadinimas (anglų k.):** pateikiamas pilnas duomenų rinkinio pavadinimas anglų kalba.;

     Jei pirma užpildysite lietuvišką lauką, sistema angliška pavadinimą paruoš automatiškai;
   - **Aprašymas (anglų k.):** pilnas duomenų rinkinio aprašymas anglų kalba;

     Jei pirma užpildysite lietuvišką lauką, sistema angliška pavadinimą paruoš automatiškai;

   SVARBU:
   - skirtas pastaboms apie atveriamų duomenų rinkinius, atsakingus asmenis, reikiamas sukurti paskyras rinkiniui tvarkyti. Vaizduojamas tik administracinėje aplinkoje;*
   - šiame lange įkelti struktūros ir spausti **[Įkelti struktūros failą]** neprivalote: struktūros failus siūlome pateikiami skiltyje **„2. Struktūra“**. Plačiau: *Duomenų rinkinio struktūros sukūrimas*.

4. Įsitikinkite, kad įvedėte teisingus duomenis ir spauskite **[Saugoti]**.



34 pav. Išsaugotos kortelės „1. Inventorinimo duomenys“ sistemos pranešimo fragmentas.

Išsaugojus užpildytą kortelę „1. Inventorinimo duomenys“, atveriamos sekančios kortelės.

5. Tęskite kitų kortelių suvedimą. Siūlome kuriant naują rinkinį informaciją įvesti nuosekliai.

7.9.2 Struktūra

Duomenų naudotojams bus aiškiau, kaip publikuojami duomenys, jei įkelsite ir struktūrą:

id	dataset	resource	base	model	property	type	ref	
						prefix	locn	
	datasets/gov/regitra/tp							
		csv				csv		https://www.regitra.lt/
				TransportoPriemone			marke, komercinis_pav	
					marke	string		MARKE
						lang	en	
					komercinis_pav	string		KOMERCINIS_PAV
						lang	en	
					gamintojo_pav	string		GAMINTOJO_PAV
						lang	en	
					gamintojo_pav_baz	string		GAMINTOJO_PAV_BAZ
						lang	en	
					tipas	string		TIPAS
						lang	en	
					variantas	string		VARIANTAS
						lang	en	
					versija	string		VERSIJA
						lang	en	
					es_tipo_patvirtinimo_nr	string		ES_TIPO_PATVIRTINIM
						lang	en	

35 pav. Struktūros failo Excel dokumente pavyzdys

SVARBU: Į Portalą vienam rinkiniui turi būti įkeliamas vienas duomenų struktūros aprašas (DSA).

Galima įkelti kelias vieno rinkinio versijas, tačiau vienam rinkiniui gali būti publikuojamas tik vienas DSA.

1. Duomenų rinkinio lange spauskite lango meniu punktą **„2. Struktūra“**.

36 pav. Duomenų rinkinio struktūros langas

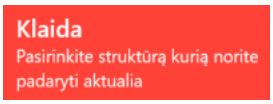
Duomenų rinkinio struktūros lange pateikiama pagrindinė informacija:

- **Pridėtas**: data ir laikas, kada struktūros failas buvo pridėtas;
- **Pavadinimas**: struktūros pavadinimas sistemoje, sukuriamas įkeliant struktūros failą;
- **Failo pavadinimas**: pilnas įkelto struktūros failo pavadinimas;
- **Standartinis:** požymis, kad duomenų rinkinys yra bendrinės struktūros;
- **Aktualus:** požymis, kad duomenų rinkinys yra naujausios versijos.

1. Pasirinkite reikiamą duomenų struktūros versiją ir spauskite [Padaryti aktualiu].

   Sėkmingai atlikus veiksmą, sistema pažymės struktūrą stulpelyje „Aktualus“ su verte „Taip“.

Jeigu nesate įkėlę duomenų struktūros, gausite klaidos pranešimą:



37 pav. „Daryti aktualiu“ klaidos langas, kai nepateikta duomenų struktūra

2. Pasirinkite norimą pašalinti duomenų struktūros versiją ir spauskite **[Šalinti failą]**.

   Sėkmingai atlikus veiksmą, sistema pašalins pasirinktą struktūrą iš sąrašo.

38 pav. „Šalinti failą“ patvirtinimo pranešimas

7.9.3 Prioritetai

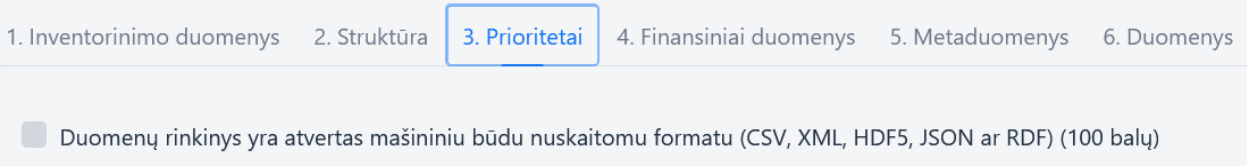
Pastaba: šiuo metu pildyti šios skiltіes laukų nėra privaloma.

Prioritetai naudojami sudarant organizacijos metų planus.

Atsižvelgus į suteikiamą finansavimą, apskaičiuojama, kuriems rinkiniams suteikiama pirmenybė.

- *prioritetus galima įvesti tik suinventorintiems rinkiniams.*
- *prioritetus galima redaguoti tol, kol rinkinys nėra įtrauktas į metinų rinkinių atvėrimo planą.*

1. Pasirinkto suinventorinto duomenų rinkinio lango viršuje meniu spauskite **[3. Prioritetai]**
2. Pažymėkite visus tinkamus laukus:



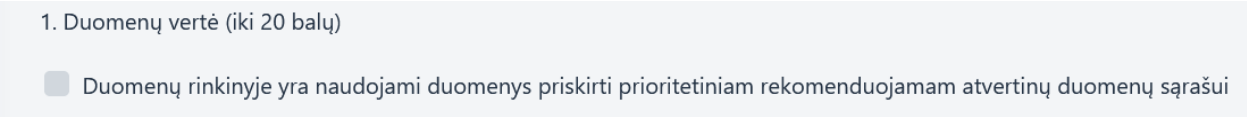
39 pav. Prioritetų kortelės pirmasis fragmentas

3. Jei galioja: pažymėkite lauką **„Duomenų rinkinys yra atvertas mašininiu būdu ...“**.

   Pasirinkimas suteikia maksimalų prioritetą, 100 balų.

   Pasirinkus daugiau laukų pasirinkti nebegalima.

4. Kitais atvejais, atskirai pažymėkite tinkamus laukus iš sąrašo.



40 pav. Prioritetų kortelės fragmentas „Duomenų vertė“

Laukas **1. Duomenų vertė (20 balų)**

Duomenų poreikis pagal duomenų tvarkytojo vertinimą. Pažymėkite, kokiems tikslams duomenys gali būti panaudojami.

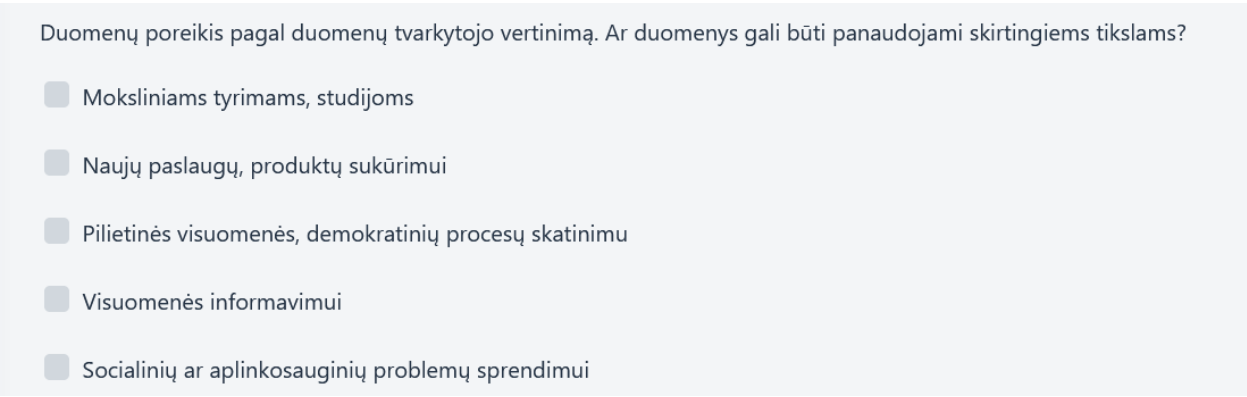
... rinkinyje yra naudojami duomenys priskirti prioritetiniam rekomenduojamam atvertinų duomenų sąrašui.

Poreikių pasirinkimai suskirstyti trimis laukais.

Kiekvienas iš jų turi po maksimalią prioritetų taškų sumą.

Laukas **Duomenų atvėrimo poreikis (50 balų)**

- Pažymima, kokiems poreikiams buvo pateiktos naudotojų užklausos: kokiems pasirinkimams iš sąrašo portalo naudotojai pateikė poreikį atverti duomenų rinkinį.



41 pav. Prioritetų kortelės fragmentas: „Duomenų poreikis“

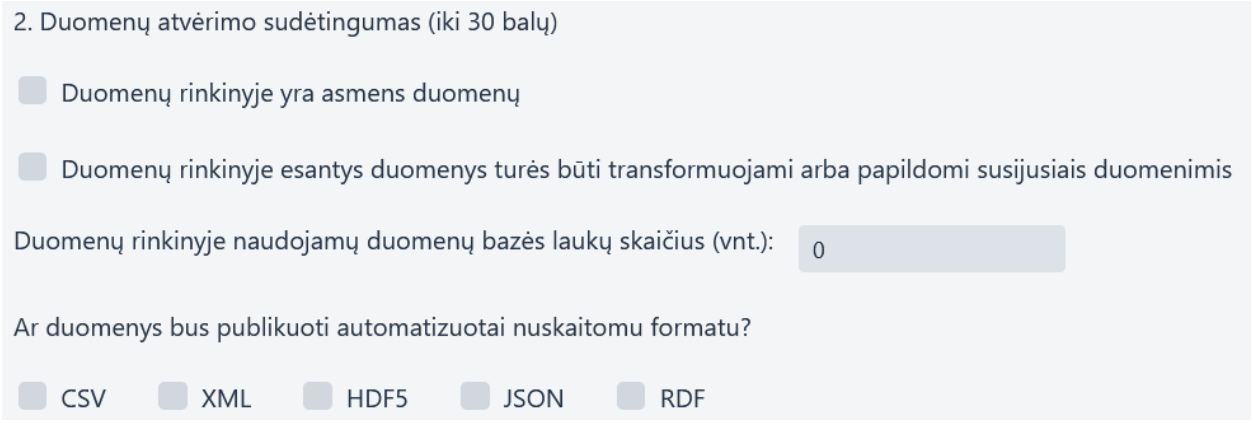
Sužymėkite atitinkamus pasirinkimus:

- Moksliniams tyrimams, studijoms
- Naujų paslaugų, produktų sukūrimui;
- Pilietinės visuomenės, demokratinių procesų skatinimu;
- Visuomenės informavimui;
- Socialinių ar aplinkosauginių problemų sprendimui;

Lauke **„Duomenų atvėrimo sudėtingumas“** taip pat galima pažymėti šiuos pasirinkimus:

Duomenų rinkinyje yra asmens duomenų

Duomenų rinkinyje esantys duomenys turės būti transformuojami arba papildomi susijusiais duomenimis.



42 pav. Prioritetų kortelės antras fragmentas

Į tekstinį lauką pagal poreikį įveskite duomenų rinkinyje naudojamų duomenų bazės laukų skaičių.

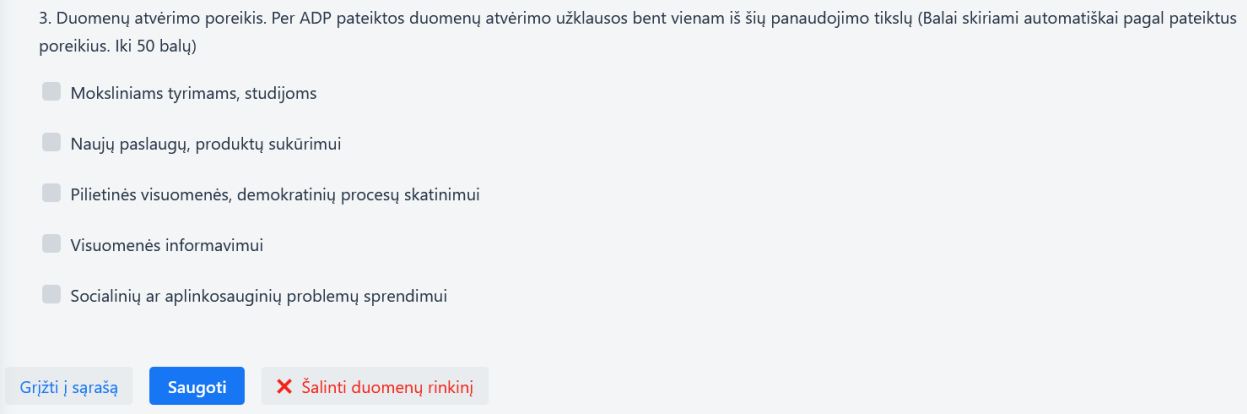
Lauko numatyta reikšmė „0“ (jei prioritetų pasirinkimai pildomi kiekviename lauke atskirai).

- pasirinkite, ar duomenys bus publikuoti automatizuotai nuskaitomu formatu, jei taip, kokiu formatu(-ais):

CSV XML HDF5 JSON RDF

Kiekvienas iš formatų taip pat suteikia reitingą duomenų rinkiniui, remiantis Tim Berners-Lee modeliu:

- 1 - priskiriama *PDF* ir analogiškiems formatams;
- 2 - priskiriamos *XLSX* ir analogiškiems formatams;
- 3 - priskiriamos *CSV* ir analogiškiems formatams;
- 4 - priskiriamos *JSON* ir analogiškiems formatams;
- 5 - priskiriamos *RDF* ir analogiškiems formatams, kai duomenys yra susieti su kitais duomenų rinkiniais.



43 pav. Prioritetų kortelės paskutinis fragmentas

5. Pasirinkite visus reikiamus laukus
6. Spauskite mygtuką **[Saugoti]** ir sistema išsaugos kortelės laukų informaciją.
7. Tęskite kitų kortelių suvedimą.

7.9.4 Finansai

Pastaba: šiuo metu pildyti šios skilties laukų nėra privaloma.

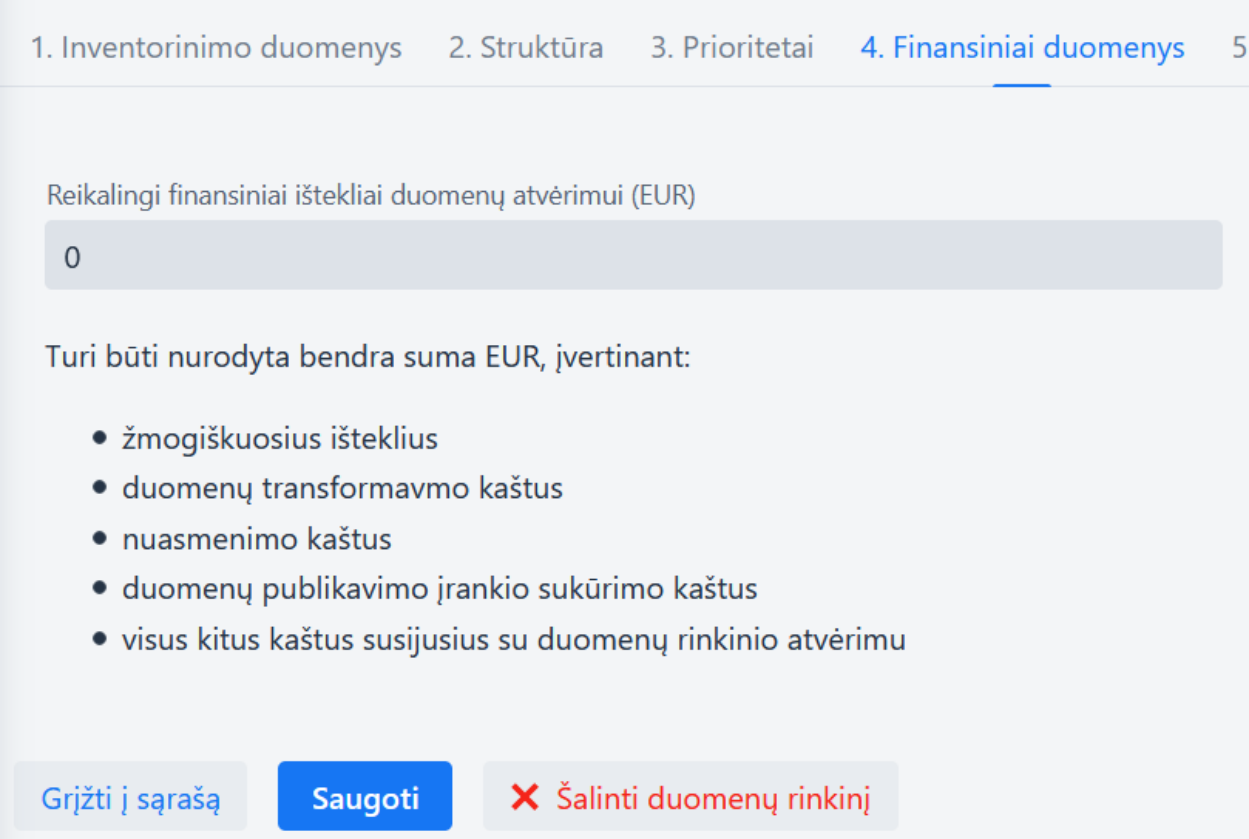
Duomenų rinkinių finansiniai duomenys įvedami siekiant apskaičiuoti reikalingą atverti sumą visiems metų plano rinkiniams. Daugiau: Darbas su metiniais planais.

Finansinius duomenis galima įvesti tik suinventorintiems rinkiniams su nustatytais prioritetais.

Duomenų rinkinio finansinius duomenis galima redaguoti tol, kol rinkinys nėra įtrauktas į metinį planą.

Įvertinus rinkinio finansavimą, jį galima įtraukti į metinį planą.

2. Pasirinkto duomenų rinkinio lango meniu spauskite **[4. Finansiniai duomenys]**.



44 pav. Finansinių duomenų kortelės langas

3. Lauke **„Reikalingi finansiniai ištekliai duomenų atvėrimui (EUR)“** įveskite sumą, reikalingą atverti duomenis.

   Rašykite natūraliaisiais skaičiais, be kablelių ir taškų.

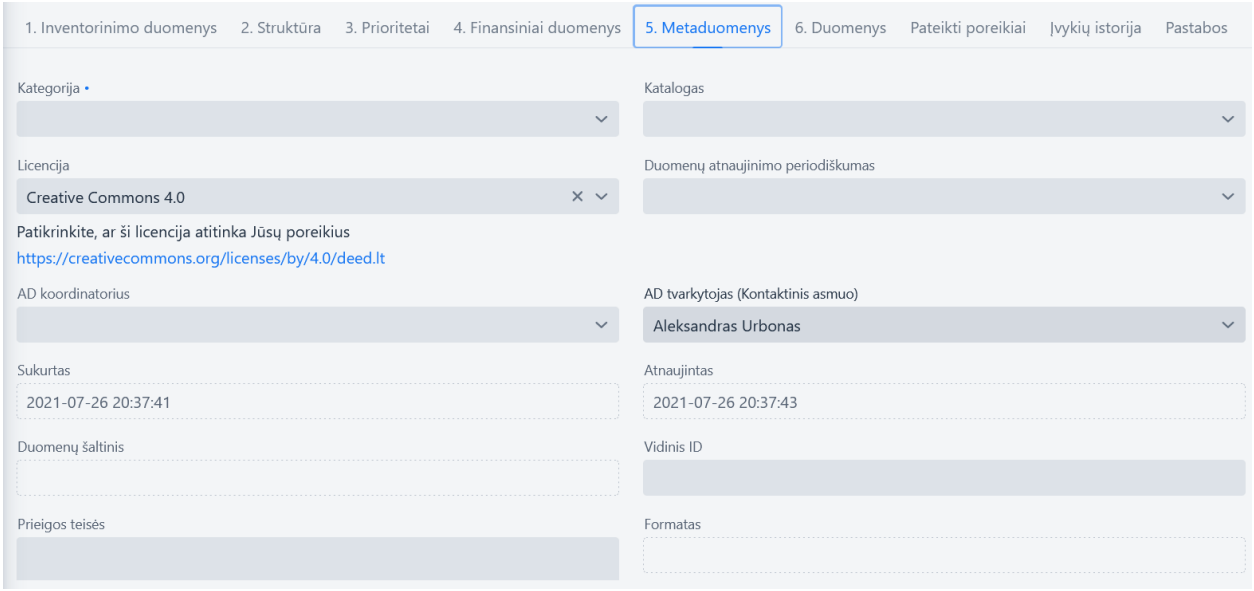
   Įvesta suma – viena iš sąlygų, pagal kurias vertinamas bendras rinkinio prioritetas metinių planų sąraše. Pirmenybė teikiama pigesniems rinkiniams.

4. Jei norite, išsaugokite pakeitimus ir tęskite kitų kortelių suvedimą.

7.9.5 Metaduomenys

Rinkinys jau turi būti suinventorintas, turėti įvestus prioritetus ir finansinius duomenis.

1. Spauskite **[5. Metaduomenys]**.



45 pav. Duomenų rinkinio metaduomenų įvedimo/redagavimo lango pirmas fragmentas

46 pav. Duomenų rinkinio metaduomenų įvedimo/redagavimo lango antras fragmentas

2. Užpildykite laukus pagal poreikį (* – privalomi; A – automatiniai):

- **Kategorija*:** iš sąrašo pasirenkama, kokiai kategorijai priklauso duomenų rinkinys;
- **Licencija:** iš sąrašo pasirenkama duomenų rinkiniui taikoma licencija.

  Pasirinkus kurią nors licenciją, pateikiama nuoroda į ją, kad galėtumėte patikrinti, ar pasirinkta licencija tinkama;

- **AD koordinatorius (A):** iš sąrašo pasirenkamas organizacijos koordinatorius;
- **Sukurtas (A):** data ir laikas, kada duomenų rinkinys buvo sukurtas;
- **Duomenų šaltinis (A):** duomenų rinkinio distribucijos šaltinio failo pavadinimas;
- **Prieigos teisės:** lauke įvardinama, kas turi prieigos teises prie duomenų rinkinio, jei prieiga yra ribota;
- **Periodo pradžia (A):** rinkinyje pateiktų duomenų periodo pradžia.

Laukas netaisomas: nustatomas automatiškai, nurodant anksčiausią visuose įkeltuose rinkinio failuose periodo pradžios datą (nurodoma keliant duomenų failą).

Neįkėlus nei vieno duomenų failo, laukas rodomas tuščias; nustatoma automatiškai.

Jei nežinoma, rašoma reikšmė „Nežinoma“;

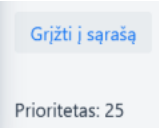
- **Raktiniai žodžiai*:** suvedami raktiniai žodžiai, tinkami duomenų rinkiniui, naudojami Portalo naudotojų viešoje aplinkoje atliekant išplėstinę paiešką. Visus raktinius žodžius rašykite mažosiomis raidėmis, tarpusavyje atskirkite kableliais. Rekomenduojama nurodyti iki 6 raktinių žodžių;
- **Įvertinimas:** viešoje aplinkoje registruotų naudotojų pateiktas bendras rinkinio įvertinimas 5 balų skalėje;
- **Kalba:** iš sąrašo pasirenkama kalba, kuria bus atveriamas duomenų rinkinys. *Kalbų skaičius neribojamas*;
- **Katalogas:** iš sąrašo pasirinkite „Lietuva“;
- **Duomenų atnaujinimo periodiškumas:** iš sąrašo pasirenkama, kokiu dažnumu atnaujinami rinkinio duomenys;
- **AD Tvarkytojas*:** iš sąrašo pasirenkamas organizacijos AD tvarkytojas.

  Laukas užpildomas automatiškai, tačiau gali būti redaguotas;
- **Atnaujintas (A):** paskutinio rinkinio atnaujinimo data ir laikas;
- **Vidinis ID:** rinkinio identifikatorius Portalo sistemoje;
- **Formatas (A):** formatas(-ai), kuriuo(-iais) atveriamas rinkinys;
- **Periodo pabaiga:** rinkinyje jau pateiktų duomenų periodo pabaiga.

  Nustatoma automatiškai, priklausomai nuo to, kokia vėliausia periodo pabaiga nurodyta įkeltuose failuose.

  Neįkėlus nei vieno duomenų failo, laukas rodomas tuščias;
- **Portalo duomenų rinkinio ID (A):** duomenų rinkinio unikalus identifikatorius sistemoje;
- **Atvirumas:** duomenų rinkinio atvirumo lygis skalėje nuo 1 iki 5.
- **Platinimo sąlygos:** įvedamos sąlygos platinti duomenų rinkinį.

  Lango apačioje pateikta prioriteto balų suma.



47 pav. Duomenų rinkinio metaduomenų įvedimo/redagavimo lango apatinis fragmentas

3. Užpildžius reikiamus laukus, galite spauskite mygtuką **[Saugoti]** arba pereikite prie kitų kortelių pildymo.

   Išsaugojus rinkinio būsena bus pakeista į **„Užpildyti metaduomenys“**.

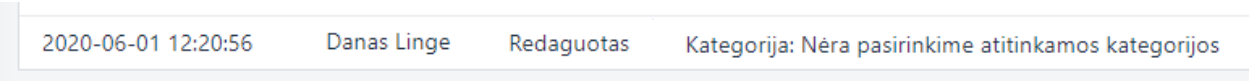
Jei spaudžiate **[Saugoti]** *neužpildę kai kurių privalomų laukų, net jei jie buvo neužpildyti ir išsaugoti prieš Jums atsidarant metaduomenų langą, sistema parodys papildomą langą, kuriame pateikiami privalomi neužpildyti DCAT laukai.*

Paaiškinimo lange galite tik įvesti lauko neužpildymo priežastį. Jei norite lauką užpildyti, paaiškinimo lange spauskite **[Atšaukti]** *ir grįšite prie pildymo.*

4. Tekstiniuose laukuose įveskite priežastį kiekvienam DCAT laukui, kurio neužpildėte.

48 pav. Duomenų rinkinio metaduomenų neužpildymo paaiškinimo lango pavyzdys

Nurodyta priežastis bus matoma poreikio peržiūros lango skiltyje **„Įvykių istorija“**, *komentaro lauke.*



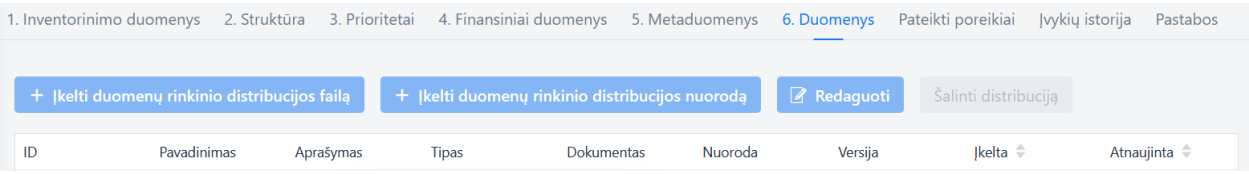
49 pav. DCAT lauko neužpildymo priežasties peržiūros pavyzdys

5. Įvedę priežastis neužpildyti privalomiems laukams, spauskite **[Saugoti]**.
6. Spauskite **[Saugoti]** metaduomenų įvedimo lange, kad išsaugotumėte įvestus duomenis arba pakeitimus.

7.9.6 Distribucijos

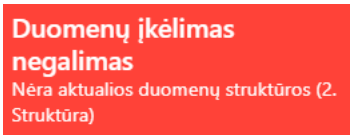
Rinkinio duomenys gali būti įkeliami, kai įkeltas duomenų struktūros aprašas **„2. Struktūra“**

1. Duomenų rinkinio lango meniu spauskite **[6. Duomenys]**.
2. Sistema parodys duomenų įkėlimui skirtą langą.



50 pav. Duomenų rinkinio duomenų distribucijos langas

SVARBU: jeigu nėra sukurta aktualios duomenų struktūros, vartotojui nebus leidžiama įkelti duomenis ir bus rodomas atitinkamas pranešimo langas:



51 pav. Duomenų rinkinio duomenų distribucijos lango klaidos pranešimas

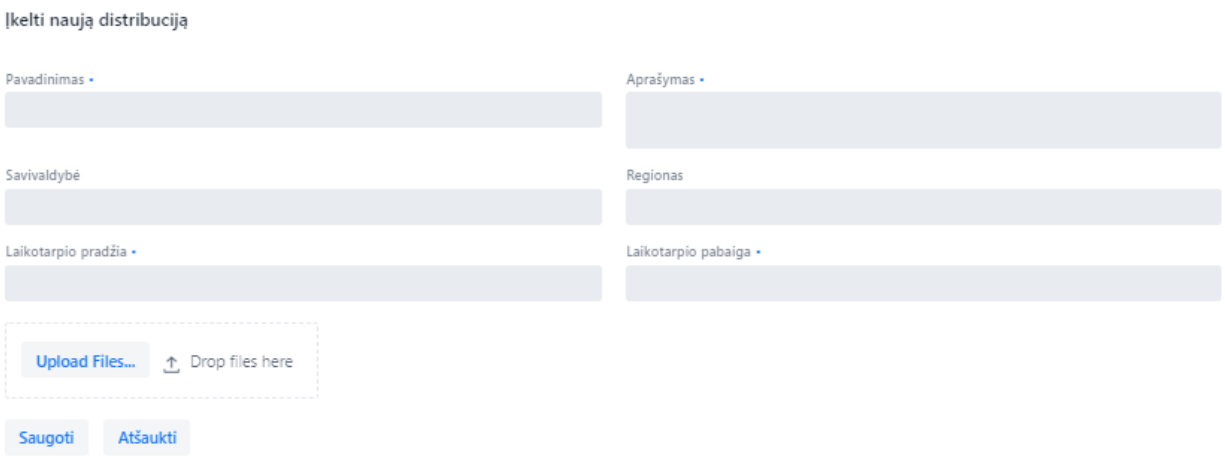
Lange pateiktas sąrašas įkeltų nuorodų ir failų, kuriuose pateikti duomenys, nurodant šią informaciją:

- **ID:** unikalus failo arba nuorodos ID sistemoje, sugeneruojamas automatiškai įkeliant;
- **Pavadinimas:** failo arba nuorodos pavadinimas sistemoje;
- **Aprašymas:** trumpas failo arba nuorodos aprašymas;
- **Tipas:** nurodoma „FILE“, jei tai yra failas, arba „URL“, jei tai nuoroda;
- **Dokumentas:** įkelto failo pilnas pavadinimas. *Jei tai nuoroda, laukas paliekamas tuščias*;
- **Nuoroda:** įkelta pilna nuoroda (su pradžia *http://* arba *https://*). *Jei tai failas, laukas paliekamas tuščias*;
- **Versija:** skaičiumi nurodoma, kelinta tai įkelta to failo arba nuorodos versija;
- **Įkelta:** data ir laikas, kada nuoroda arba failas buvo įkeltas;
- **Atnaujinta:** data ir laikas, kada nuoroda arba failas buvo paskutinį kartą atnaujintas.

3. Įkelkite naują distribuciją:

- kaip *failą*: spauskite **[+Įkelti duomenų rinkinio distribucijos failą]**.
- kaip *nuorodą*: spauskite **[+Įkelti duomenų distribucijos nuorodą]**.

Sistema parodys naujos distribucijos langą.



52 pav. Naujos distribucijos kaip failo įkėlimo langas

4. Užpildykite reikiamus laukus (* - privalomi):

- **Pavadinimas * :** Įkeliamo duomenų failo pavadinimas;
- **Savivaldybė:** Savivaldybė, kuriai priklauso įkeliamas duomenų rinkinio failas;
- **Laikotarpio pradžia * :** Įkeliamo failo laikotarpio pradžia;

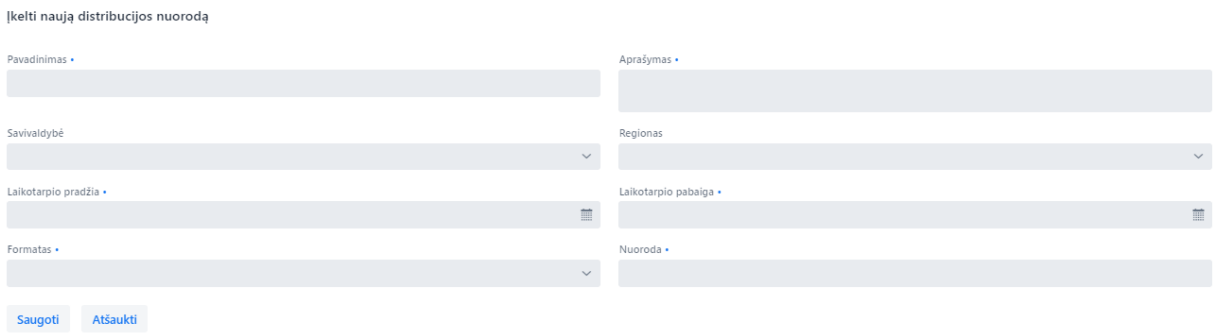
- **Aprašymas * :** Įkeliamo duomenų failo aprašymas;
- **Regionas:** Regionas, kuriam priklauso įkeliamas failas;
- **Laikotarpio pabaiga * :** Įkeliamo failo laikotarpio pabaiga.

  Failo įkėlimas privalomas pridedant distribuciją.

5. Spauskite mygtuką **[Upload files]** arba nutempkite failą iki lauko **{Drop files here}**.
6. Jei distribuciją keliate kaip nuorodą, nuorodos įkėlimo lange vietoje suveskite papildomus privalomus laukus:

- **Formatas*:*** iš sąrašo pasirinkite vieną duomenų distribucijos formatą;*
- **Nuoroda*:*** tiesioginė nuoroda į duomenų failą, kurį galima atsisiųsti.*

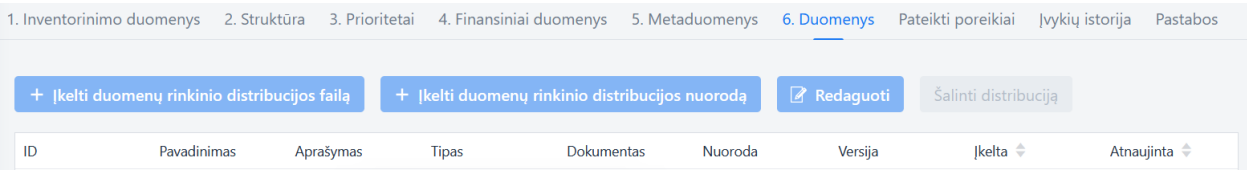
Pastaba: Jei pateikiate nuorodą, ne tiesiogiai į duomenų failą, o į svetainę, kurioje publikuojami duomenys, tuomet **Formatas:** laukelyje pasirinkite **URL**, kuris žymi, kad ši nuoroda yra ne į duomenis, o į svetainę iš kurios galima atsisiųsti duomenis.



53 pav. Distribucijos nuorodos įkėlimo langas

7. Užpildę laukus, spauskite **[Saugoti]**.

Naujai įkeltas failas ar nuoroda bus iškart matomi distribucijų lange esančiame sąraše:



54 pav. Duomenų distribucijų sąrašo pavyzdys

> **Redaguoti distribuciją:**

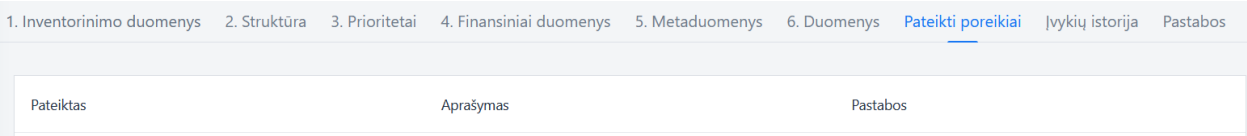
- Įkeltų distribuciją atnaujinimai atliekami per API ir UI.

> **Pašalinti distribuciją iš sąrašo:**

- pažymėkite ją sąraše ir spauskite **[Šalinti distribuciją]** ekrano viršuje.

7.9.7 Prašymai gauti duomenis

1. Rinkinio peržiūros lango meniu pasirinkite skiltį **„Pateikti poreikiai“**.



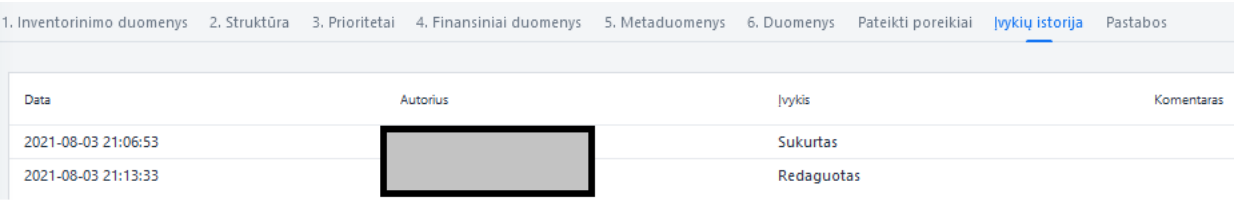
55 pav. Duomenų rinkiniui pateiktų poreikių peržiūros langas

Poreikių peržiūros lange galite tik peržiūrėti pateiktų poreikių informaciją:

- **Pateiktas:** data ir laikas, kada poreikis buvo pateiktas;
- **Aprašymas:** poreikio teikėjo pateiktas poreikio aprašymas;
- **Pastabos:** pastabos tekstas.

7.9.8 Istorija

1. **Duomenų rinkinio lango** viršuje esančiame meniu pasirinkite skiltį **„Įvykių istorija“**.



56 pav. Duomenų rinkinio istorijos langas

Lange pateikti rinkinio istorijos duomenys, automatiškai kaupiami nuo duomenų rinkinio sukūrimo:

- **Data:** data ir laikas, kai buvo atliktas veiksmas su duomenų rinkiniu (sukūrimas, redagavimas);
- **Autorius:** vardas ir pavardė paskyros, kuria atitinkamas veiksmas buvo atliktas, savininko;
- **Įvykis:** įvardinama, kas buvo atlikta: redaguotas rinkinys, pakeistas jo statusas (būsena) ir t.t.;
- **Komentaras:** koordinatoriaus paliktas komentaras redaguojant rinkinį (pvz., priežastys, kodėl tam tikri metaduomenų lango laukai palikti neužpildyti). Jei atliktas veiksmas nereikalavo paliekamo komentaro, laukas paliekamas tuščias.

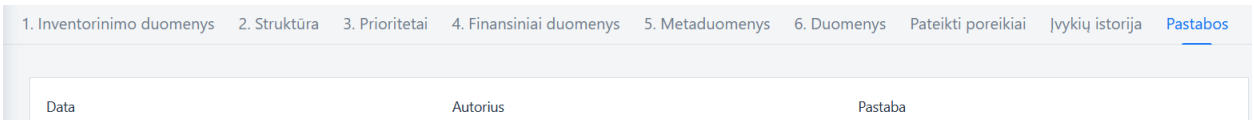
7.9.9 Pastabos

Tvarkant duomenų rinkinių atvėrimo planus, gali būti pateiktos pastabos.

Pastabos peržiūrimos per duomenų rinkinį arba metinį planą.

Peržiūrėti pastabas per metinį planą: sąraše pasirinkite metinio planą, stulpelyje **„Pastaba“**.

1. Pagrindiniame meniu pasirinkite **„Duomenų rinkiniai“**, rinkinių sąrašo lange pasirinkite reikiamą rinkinį;
2. Duomenų rinkinio lango viršuje esančiame meniu pasirinkite **„Pastabos“**.



57 pav. Duomenų rinkinio pastabų sąrašo lango fragmentas

Lange pateikiama pagrindinė pastabų informacija:

- **Data:** data ir laikas, kada pastaba buvo pateikta;
- **Autorius:** vardas ir pavardė pastabą parašiusios paskyros savininko;
- **Pastaba:** pateiktas pastabos tekstas.

Sąraše galite pasirinkti reikiamą pastabą, kad atvertumėte jos peržiūros langą ir galėtumėte perskaityti pilną pastabos tekstą.

58 pav. Duomenų rinkiniui pateiktos pastabos peržiūros langas

Pateiktų pastabų redaguoti negalima, tik peržiūrėti.

Naujos pastabos pateikiamos, kai vyr. koordinatorius atmeta iš naujo nacionaliniam planui pateiktą organizacijos duomenų rinkinį.

7.10 Atvėrimo planas

Metiniai planai sudaromi ruošiantis duomenų rinkinių atvėrimui.

Ruošiant metinius planus atsižvelgiama į duomenų prioritetą bei finansavimą.

Organizacijos koordinatorius atsako už organizacijos atveriamų duomenų rinkinių metinio plano:

- *Sudarymą;*
- *pateikimą vyr. koordinatoriui,*
- *plano redagavimą, jei vyr. koordinatorius planą atmeta,*
- *patvirtinimą organizacijos vardu, jei vyr. koordinatorius planą patvirtina.*

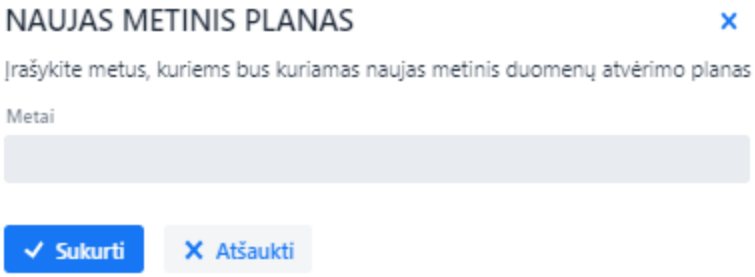
7.10.1 Metinio plano sudarymas

1. Pagrindiniame meniu pasirinkite **„Atvėrimo planai“**.
2. Atsivėrusiame metinių atvėrimų plane spauskite **[+ Naujas duomenų atvėrimo planas]**:

61 pav. Atvėrimo planų sąrašo langas

Lange pateikiama pagrindinė atvėrimo planų informacija:

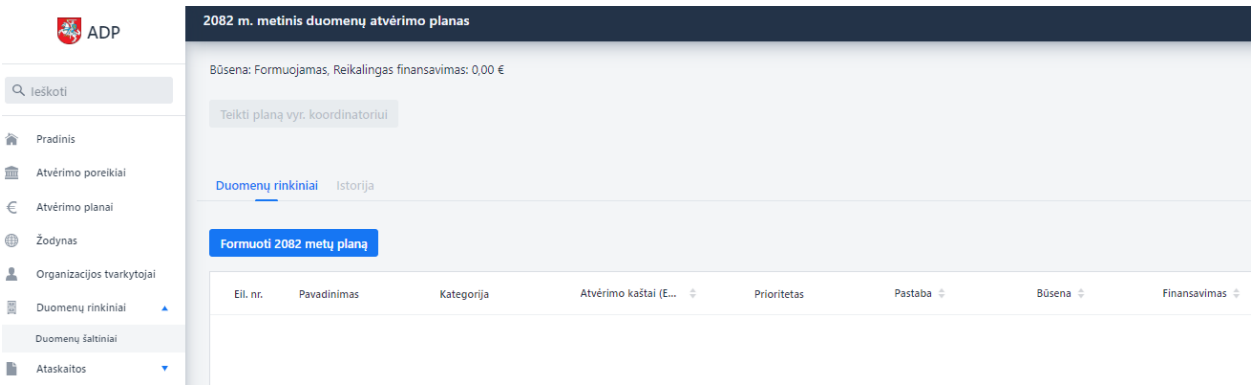
- **Eil. nr. –** atvėrimo plano eilės vieta sąraše;
- **Metai –** metai, kuriems skirtas atvėrimo planas;
- **Planuojami kaštai (EUR) –** suma eurais, reikalinga pilnai finansuoti duomenų rinkinį;
- **Planuojami duomenų rinkiniai –** skaičius duomenų rinkinių, kurie įtraukti į atvėrimo planą;
- **Numatytas finansavimas (EUR) –** suma eurais, skirta finansuoti metų planą;
- **Finansuoti duomenų rinkiniai –** duomenų rinkinių, kuriems buvo vyr. koordinatoriaus skirtas finansavimas, skaičius;
- **Būsena –** atvėrimo plano būsena. Galimos reikšmės:
  - *Formuojamas*: organizacija dar kuria/redaguoja planą;
  - *Pateiktas*: planas pateiktas vyr. koordinatoriui, bet į jį dar nebuvo atsakyta;
  - *Patvirtintas koordinatoriaus*: vyr. koordinatorius patvirtino planą;
  - *Patvirtintas organizacijos*: po vyr. koordinatoriaus patvirtinimo planas buvo vėl patvirtintas organizacijos.
3. Naujo metinio plano lange įveskite, kuriems metams norite sukurti atvėrimo planą ir spauskite **[Sukurti]**.



Metinio plano metų pasirinkimo langas

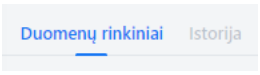
7.10.2 Metinio plano formavimas ir pateikimas

1. Pagrindiniame meniu spauskite pasirinkite **„Atvėrimo planai“**.
2. Atvėrimo planų lange pasirinkite planą, kurį norite formuoti:



62 pav. Metinio duomenų atvėrimo plano langas

Metinio duomenų atvėrimo plano lango viršuje – plano būsena ir reikalingas finansavimas, suma EUR.

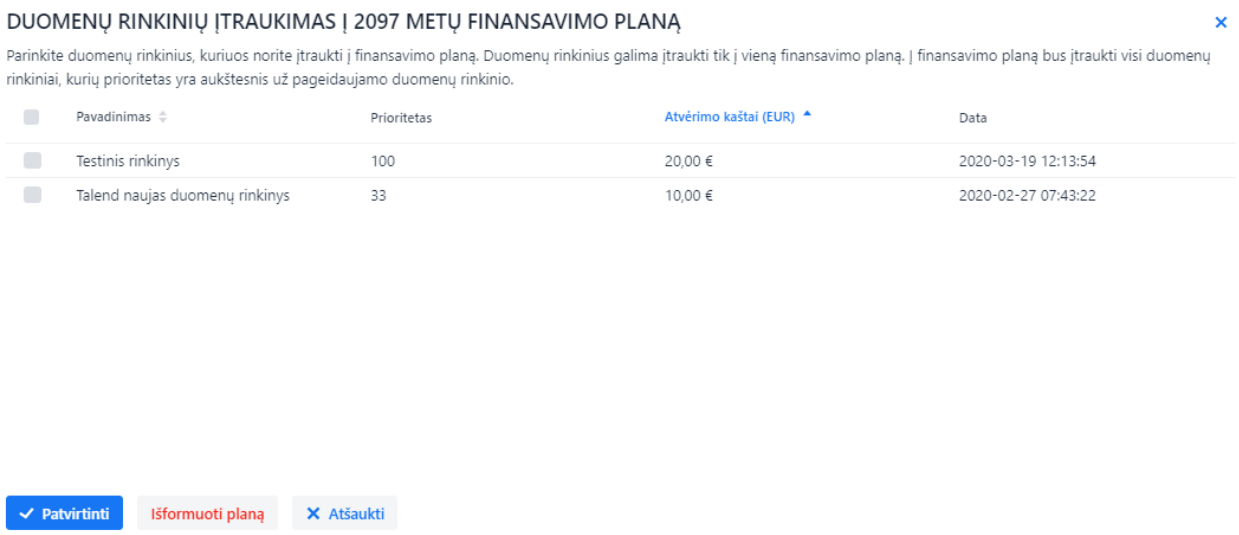


63 pav. Metinio duomenų atvėrimo plano lango poskyrių meniu

Numatyta lango įjungimo skiltis –**„Duomenų rinkiniai“**.

*Skiltis***„Istorija“***tampa prieinama (tapusi prieinama patamsėja) atlikus kokius nors veiksmus su planu.*

3. Pasirinkto metinio plano lange spauskite **[Formuoti <Jūsų pasirinkti metai> metų planą]**, esantį virš metinio duomenų atvėrimo plano lango poskyrių meniu.
4. Atvertame rinkinių įtraukimo lange, pasirinkite, kurie rinkinių poreikiai turi būti įtraukti į metinį planą, pažymėdami žymimąjį laukelį rinkinio kairėje pusėje.



64 pav. Pasirinkto metinio duomenų atvėrimo plano langas

Sąraše pateikti tik tie rinkiniai, kurie priskirti Jūsų organizacijai, yra nepriskirti kitam metiniam planui, neturi įkeltų duomenų ir kurių būsena yra **„Įvertintas finansavimas“** arba **„Užpildyti metaduomenys“**.

5. Įsitikinę, kad pasirinkote visus reikiamus rinkinius, spauskite **[Patvirtinti]**.

   Pastaba*: į metinį planą įtrauktų duomenų rinkinių prioritetų ir finansinių duomenų nebegalima redaguoti. Juos bus galima redaguoti tik pašalinus rinkinį iš plano. Daugiau:Metinio plano išformavimas*

6. Metinio plano lange spauskite **[Pateikti planą vyr. koordinatoriui]**.

Pateikto plano nebebus galima redaguoti, išformuoti ar pateikti iš naujo.

Tokie veiksmai atlikti su planu bus galimi tik vyr. koordinatoriui jį atmetus.

Vyr. koordinatorius, atmesdamas planą, pateikia pastabas atitinkamiems į planą įtrauktiems duomenų rinkiniams, nurodydamas ką konkrečiai reikia redaguoti.

7.10.3 Metinio plano išformavimas

Į metinį planą įtraukti duomenų rinkiniai negali būti įtraukti į kitus rinkinius bei negali būti redaguoti jų prioritetai ir finansiniai duomenys, net jei pateiktas planas buvo atmestas.

Dabartinį planą turite išformuoti, kad duomenų rinkinys nebebūtų jam priskirtas:

- *Kai rinkinys turi būti paliktas kitų metų planui;*
- *Kai reikia pakeisti prioritetus ir/arba finansinius duomenis.*

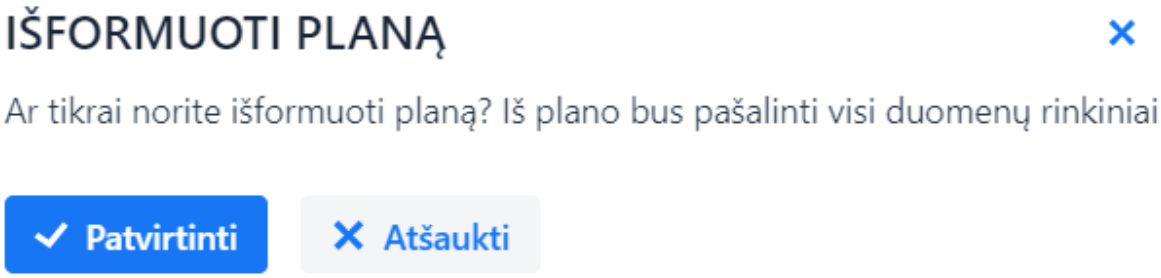
PASTABA: išformuoti negalite planų, kurie yra patvirtinti vyr. koordinatoriaus arba organizacijos, arba įtraukti į jau patvirtintą nacionalinį duomenų rinkinių atvėrimo planą.

> Išformuoti metinį planą:

1. Pasirinkite reikiamą planą sąraše ir plano lange spauskite **[Formuoti <metai> metų planą]**.
2. Rinkinių įtraukimo lango apačioje spauskite mygtuką **[Išformuoti planą]**.

65 pav. Metinio plano išformavimo mygtukas

3. Spauskite **[Patvirtinti]**, jei tikrai norite išformuoti planą.



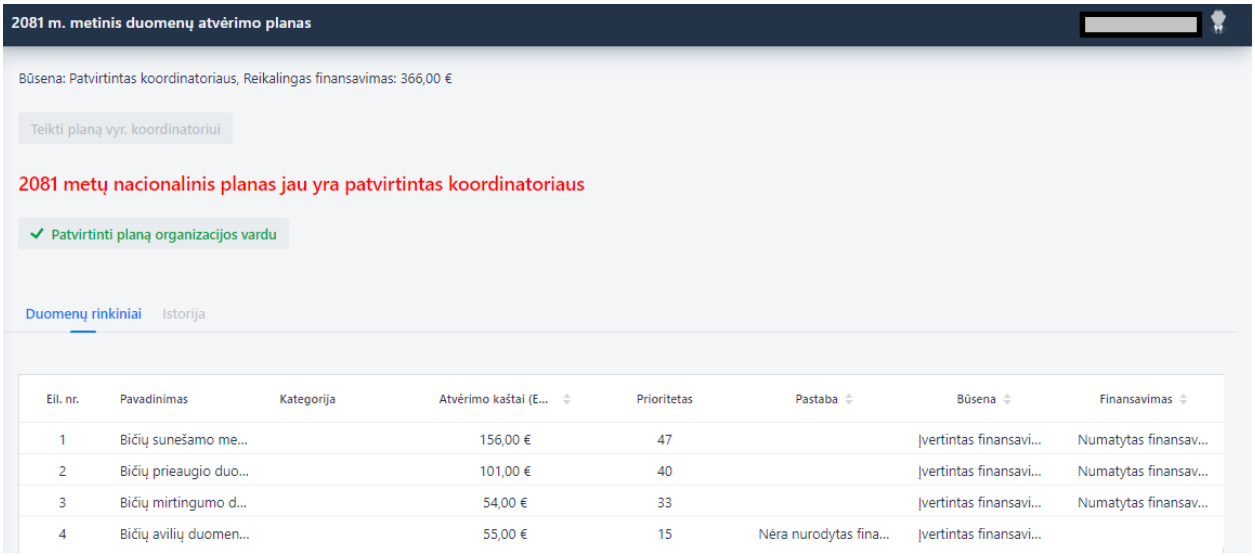
66 pav. Metinio plano išformavimo patvirtinimo langas

Planas tebeliks matomas planų sąraše ir jį bus galima formuoti iš naujo įtraukiant rinkinius, tačiau jis bus tuščias, o jam buvę priskirti rinkiniai nebebus jam priskirti ir juos bus galima priskirti kitiems metiniams planams.

7.10.4 Plano patvirtinimas organizacijos vardu

Kai metinis planas yra patvirtintas vyr. koordinatoriaus (plano būsena pasikeičia į **“Patvirtintas koordinatoriaus”***, organizacijos koordinatorius gali jį patvirtinti vyr. koordinatoriui).*

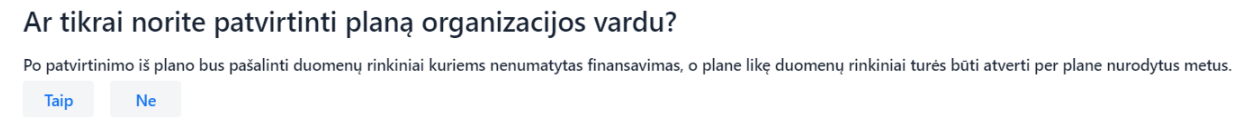
1. Pagrindiniame meniu spauskite pasirinkite **„Atvėrimo planai“**.
2. Atvėrimo planų lange pasirinkite planą, kurį norite patvirtinti (jo būsena turi būti **„Patvirtintas koordinatoriaus“**).
3. Plano lange spauskite mygtuką **[Patvirtinti planą organizacijos vardu]**:



67 pav. Metinio plano patvirtinimo organizacijos vardu pavyzdys

4. Pasirinkimo patvirtinimo lange spauskite **[Taip]**, kad patvirtintumėte planą organizacijos vardu.

   Patvirtinus, plano būsena pasikeis į „Patvirtintas organizacijos“ ir bus matomas vyr. koordinatoriui.



68 pav. Plano patvirtinimo organizacijos vardu langas

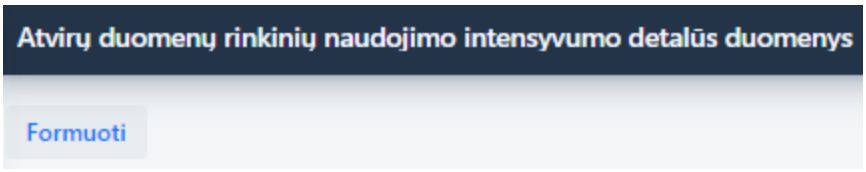
7.11 Ataskaitos

- Ataskaitoje **Atvirų duomenų rinkinių naudojimo intensyvumo detalūs duomenys** pateikiami duomenys apie *duomenų rinkinių* panaudojimą nurodytu laikotarpiu;
- Ataskaitoje **Atvirų duomenų rinkinių naudojimo intensyvumo detalūs duomenys (failams)** pateikiami duomenys apie *duomenų rinkinių rinkmenų* panaudojimą nurodytu laikotarpiu.

7.11.1 Ataskaitų kūrimas

> Suformuoti ataskaitą:

1. Pagrindiniame meniu išskleiskite punktą **„Ataskaitos“** ir paspauskite ant norimo ataskaitos tipo.
2. Pasirinktos ataskaitos lange spauskite **[Formuoti]**.



69 pav. Ataskaitos formavimo langas

3. Užpildykite reikiamus laukus
4. Formos apačioje spauskite **[Formuoti].**

Suformuota ataskaita bus pateikta ekrane.

> Atsisiųsti suformuotą rezultatą kaip Excel dokumentą:

1. Suformuotos ataskaitos lango viršuje spauskite **[Atsisiųsti XLSX]**.

> Performuoti tokio paties tipo ataskaitą:

1. Spauskite **[Formuoti]**.

   Formuojant ataskaitą nurodyti kriterijai yra pateikiami lango viršuje.

7.11.2 Paruoštų ataskaitų valdymas

> Rikiuoti stulpelį abėcėlės arba atvirkštine tvarka:

- Spauskite stulpelio pavadinimą.

> Filtruoti stulpelį:

- *Tekstiniams* laukams įveskite savo paiešką į tekstinį lauką po stulpelio pavadinimu.
- *Skaitiniams* laukams įveskite skaitines reikšmes į laukus „Nuo“ ir/arba „Iki“ po stulpelio pavadinimu.
- *Datoms* įveskite arba pasirinkite datas kalendoriuose arba įveskite jas laukuose „Nuo“ ir/arba „Iki“ po stulpelio pavadinimu.

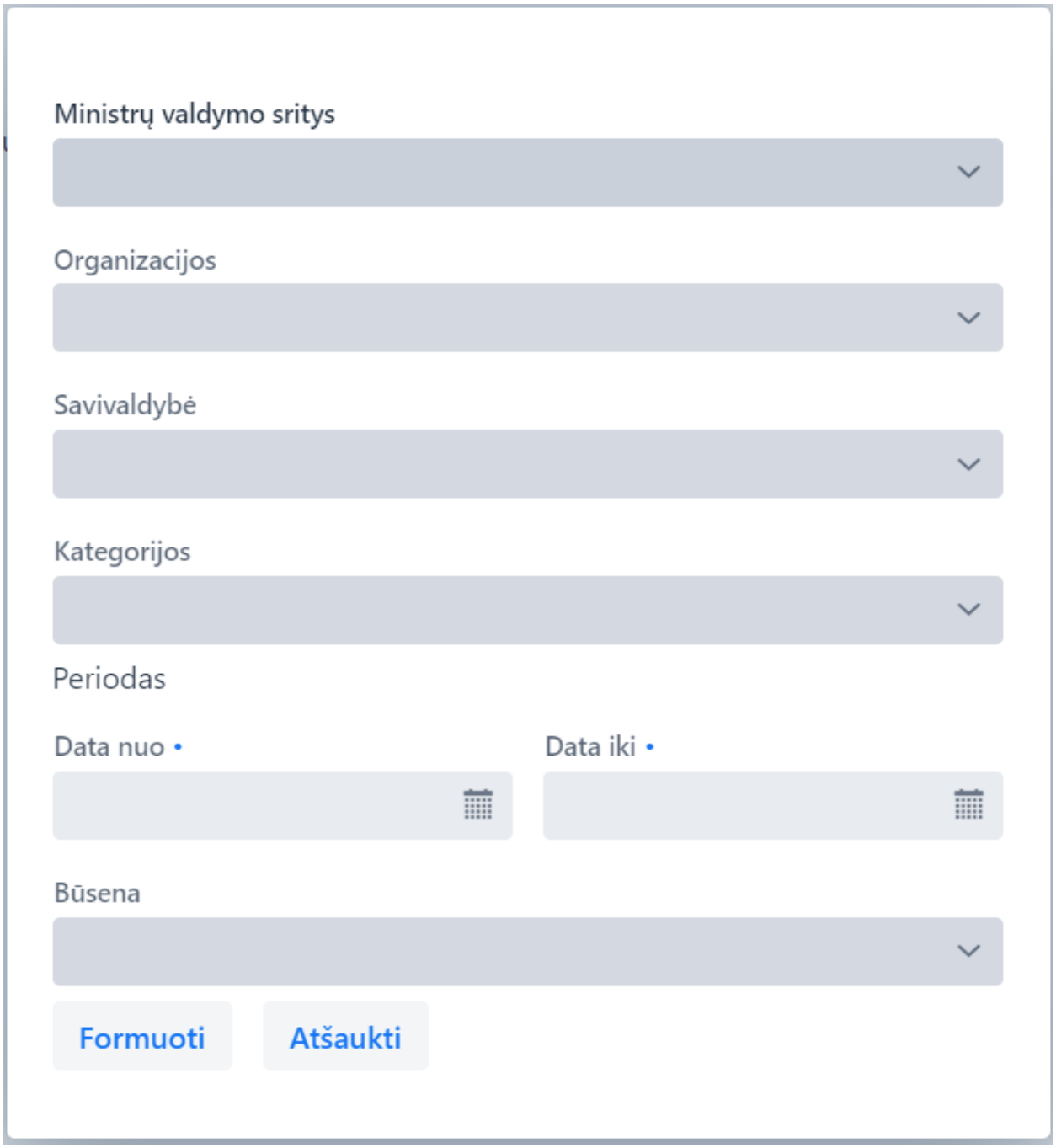
> Tvarkyti ataskaitos stulpelių eiliškumą:

- Nutempkite pasirinktą stulpelį iki reikiamos pozicijos.

> Keisti suformuotos ataskaitos stulpelius vietomis:

- Pasirinktą stulpelį nutempkite iki reikiamos pozicijos.

7.11.3 Ataskaita „Atvirų duomenų rinkinių naudojimo intensyvumo detalūs duomenys“



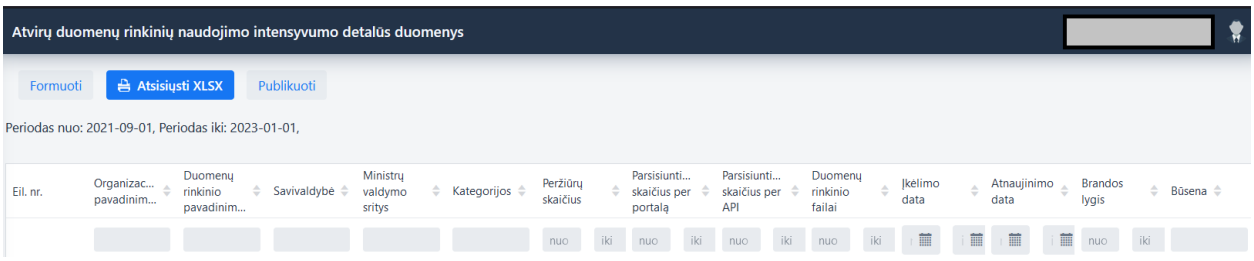
| *70 pav. Ataskaitos „Atvirų duomenų rinkinių naudojimo intensyvumo detalūs duomenys“ formavimo forma*

Ataskaitos laukai (* – privalomi):

- **Ministrų valdymo sritys**: iš sąrašo pasirinkite ministerijų pavadinimus, galite pasirinkti daugiau nei vieną.
- **Organizacijos:** pasirinkite iš sąrašo arba įveskite pavadinimus organizacijų, kurių duomenų rinkinius norima įtraukti į ataskaitą.
- **Savivaldybė:** pasirinkite iš sąrašo visas savivaldybes, su kuriomis susiję duomenų rinkiniai.

- **Kategorijos:** pasirinkite iš sąrašo arba įveskite (jei įvestis atitinka kurį nors sąrašo elementą) visas kategorijas sričių, su kuriomis susiję duomenų rinkiniai turi būti įtraukti į ataskaitą.
- **Data nuo:** pasirinkite iš kalendoriaus arba įveskite **laikotarpio, kuris įtraukiamas į ataskaitą,** pradžios datą.
- **Data iki * :** pasirinkite iš kalendoriaus arba įveskite **laikotarpio, kuris įtraukiamas į ataskaitą, pabaigos datą.**
- **Būsena:** pasirinkite iš sąrašo visas duomenų rinkinių, įtraukiamų į ataskaitą, būsenas.

Suformavus ataskaitą, duomenys pateikiami ataskaitos lange.



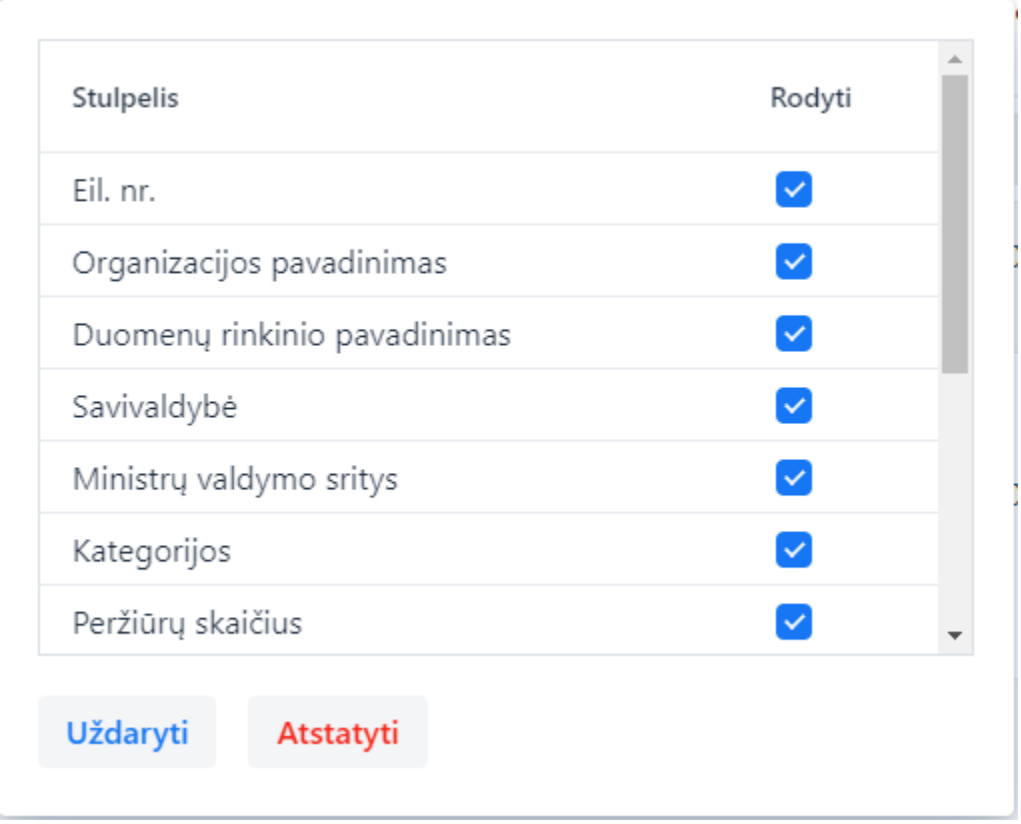
71 pav. Ataskaitos „Atvirų duomenų rinkinių naudojimo intensyvumo detalūs duomenys“ langas

Suformuotos ataskaitos lange pateikiami šie įtraukto duomenų rinkinio laukai:

- **Eil. nr.**: eilės numeris ataskaitoje.
- **Organizacijos pavadinimas:** savininkės organizacijos pavadinimas.
- **Duomenų rinkinio pavadinimas:** įtraukto duomenų rinkinio pavadinimas.
- **Savivaldybė:** priskirtos savivaldybės pavadinimas.
- **Ministrų valdymo sritys:** priskirtos ministerijos pavadinimas.
- **Kategorijos:** priskirtos kategorijos.
- **Peržiūrų skaičius:** duomenų rinkinio peržiūrų skaičius. (Skaitinis)
- **Parsisiuntimų skaičius per Portalą:** duomenų rinkinio parsisiuntimų per portalą skaičius. (Skaitinis)
- **Parsisiuntimų skaičius per API:** duomenų rinkinio parsisiuntimų per API skaičius. (Skaitinis)
- **Duomenų rinkinio failai:** duomenų rinkinio failų skaičius. (Skaitinis)
- **Įkėlimo data:** duomenų rinkinio įkėlimo į Portalą data. (Data)
- **Atnaujinimo data:** duomenų rinkinio atnaujinimo data. (Data)
- **Brandos lygis:** brandos lygio skaitmuo.
- **Būsena:** „Inventorintas“, „Suvesti duomenys“, „Užpildyti metaduomenys“, „Įvertintas finansavimas“, arba „Įvertinti prioritetai“.

> Atidaryti ataskaitos šablono keitimo langą:

1. Paspauskite ant bet kurio stulpelio pavadinimo dešiniuoju pelės klavišu.



72 pav. Ataskaitos „Atvirų duomenų rinkinių naudojimo intensyvumo detalūs duomenys“ šablono keitimo lango fragmentas

Lange pateikti stulpelių pavadinimai bei jų rodymo ataskaitoje būsena.

- Jei stulpelio būsenos reikšmė yra pažymėta varnele „☑“, stulpelis rodomas ataskaitoje.
- Jei stulpelio būsenos reikšmė yra tuščia („ “).

> Keisti stulpelio būseną:

- Paspauskite būsenos ikoną: varnelę „☑“ arba tuščią lauką „ “.

> Uždaryti šablono keitimo langą:

- Spauskite **[Uždaryti]**.

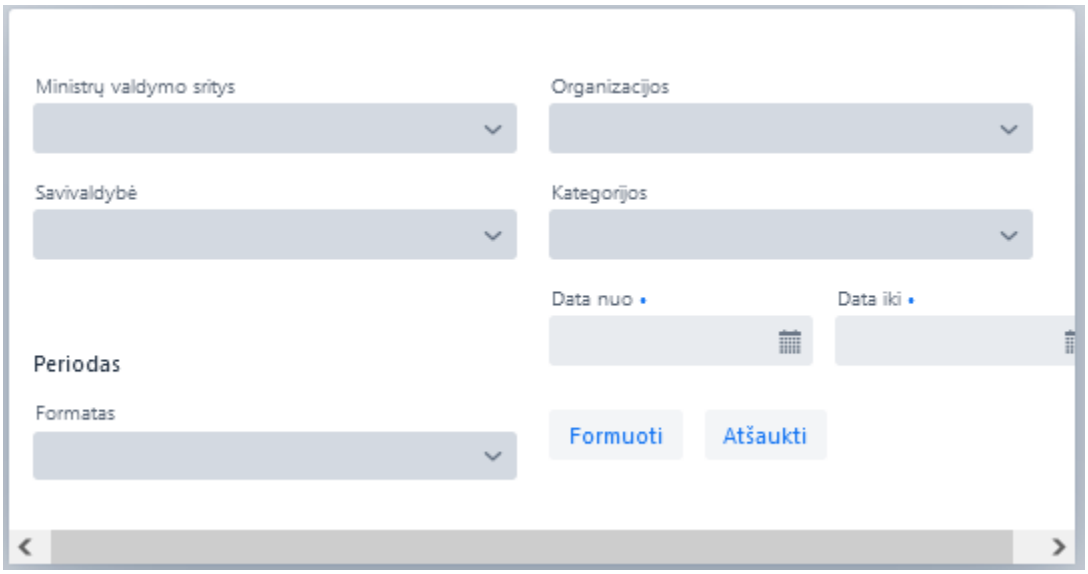
Uždarant langą bus išsaugotos naujausios reikšmės.

> Atstatyti pradines visų būsenų reikšmes:

- Spauskite **[Atstatyti]**.

Šablono keitimo langas bus uždarytas ir reikšmės bus atstatytos į pradines.

7.11.4 Ataskaita „Atvirų duomenų rinkinių naudojimo intensyvumo detalūs duomenys (failams)“

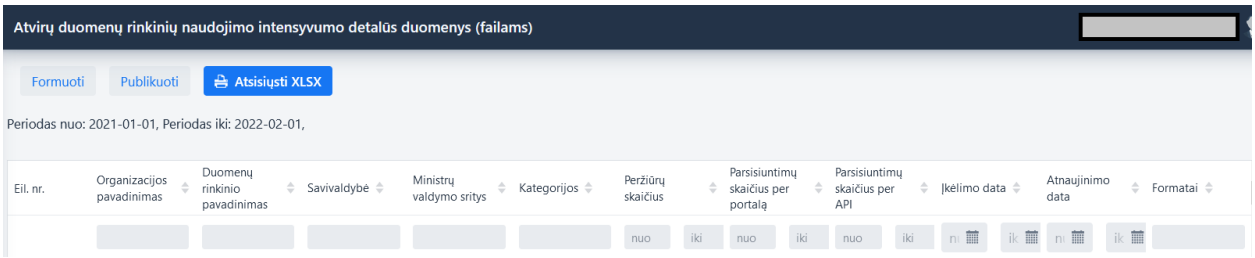


73 pav. Ataskaitos „Atvirų duomenų rinkinių naudojimo intensyvumo detalūs duomenys (failams)“ sudarymo forma

Ataskaitos laukai (* – privalomi):

- **Ministrų valdymo sritys:** iš sąrašo pasirinkite iki keleto ministerijų, su kuriomis susiję rinkiniai;
- **Organizacijos:** iš sąrašo pasirinkite arba įveskite organizacijas, kurių rinkinius norite įtraukti į ataskaitą.
- **Savivaldybė:** įveskite arba pasirinkti iš sąrašo savivaldybes, su kuriomis susiję duomenų rinkiniai. *Parenkamų savivaldybių skaičius nėra ribojamas.*
- **Kategorijos:** pasirinkti iš sąrašo, arba įvesti (jei įvestis atitinka kurį nors sąrašo elementą) kategorijas sričių, su kuriomis susiję duomenų rinkiniai turi būti įtraukti į ataskaitą. *Kategorijų skaičius neribojamas, laukas neprivalomas.*
- **Data nuo * :** pasirinkite iš kalendoriaus arba įveskite ataskaitos laikotarpio pradžios datą.
- **Data iki * :** pasirinkite iš kalendoriaus arba įveskite ataskaitos laikotarpio pabaigos datą.
- **Formatas** failų, įtraukiamų į ataskaitą, formatas, pasirenkamas iš sąrašo. *Pasirenkamų formatų skaičius neribojamas. Laukas neprivalomas.*

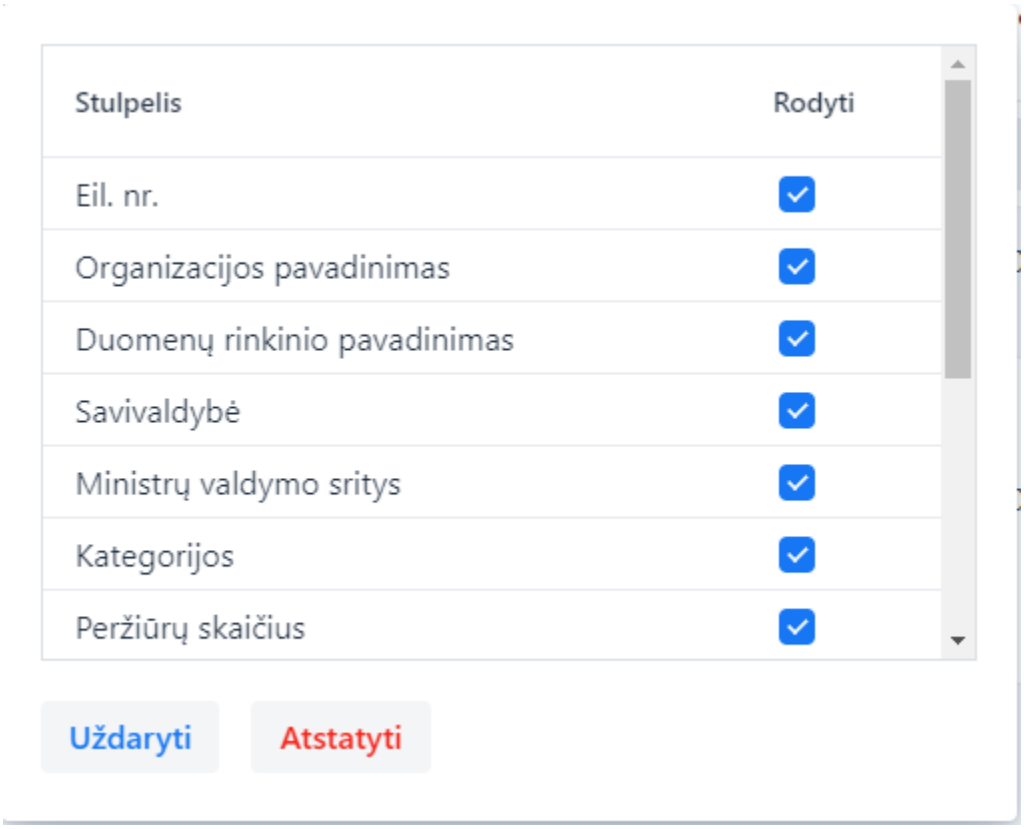
Suformavus ataskaitą, duomenys pateikiami ataskaitos lange:



74 pav. Ataskaitos „Atvirų duomenų rinkinių naudojimo intensyvumo detalūs duomenys (failams)“ langas

Žemiau pateikiami į ataskaitą įtraukto duomenų rinkinio laukai:

- **Eil. nr.:** duomenų rinkinio eilės numeris ataskaitoje.
- **Organizacijos pavadinimas:** savininkės organizacijos pavadinimas.
- **Duomenų rinkinio pavadinimas:** duomenų rinkinio pavadinimas.
- **Savivaldybė:** duomenų rinkinio pilnas savivaldybės pavadinimas. *Jei nėra priskirta, laukas tuščias.*
- **Ministrų valdymo sritys:** priskirtos ministerijos pavadinimas. *Jei nėra priskirta, laukas tuščias.*
- **Kategorijos:** Portale priskirtos kategorijos pavadinimas.
- **Peržiūrų skaičius:** duomenų rinkinio peržiūrų Portale skaičius.
- **Parsisiuntimų skaičius per Portalą:** įtraukto duomenų rinkinio parsisiuntimų per Portalą skaičius.
- **Parsisiuntimų skaičius per API:** įtraukto duomenų rinkinio parsisiuntimų per API skaičius.
- **Įkėlimo data:** įtraukto duomenų rinkinio įkėlimo į Portalą data.
- **Atnaujinimo data:** įtraukto duomenų rinkinio atnaujinimo data.
- **Formatai:** duomenų rinkinio formatas arba formatai. *Jei formatas nepriskirtas, laukas tuščias.*



75 pav. Ataskaitos „Atvirų duomenų rinkinių naudojimo intensyvumo detalūs duomenys“ šablono keitimo lango fragmentas

7.12 Partnerių API

Partnerių API operacijas galima vykdyti tik su organizacijai suteiktu API raktu.

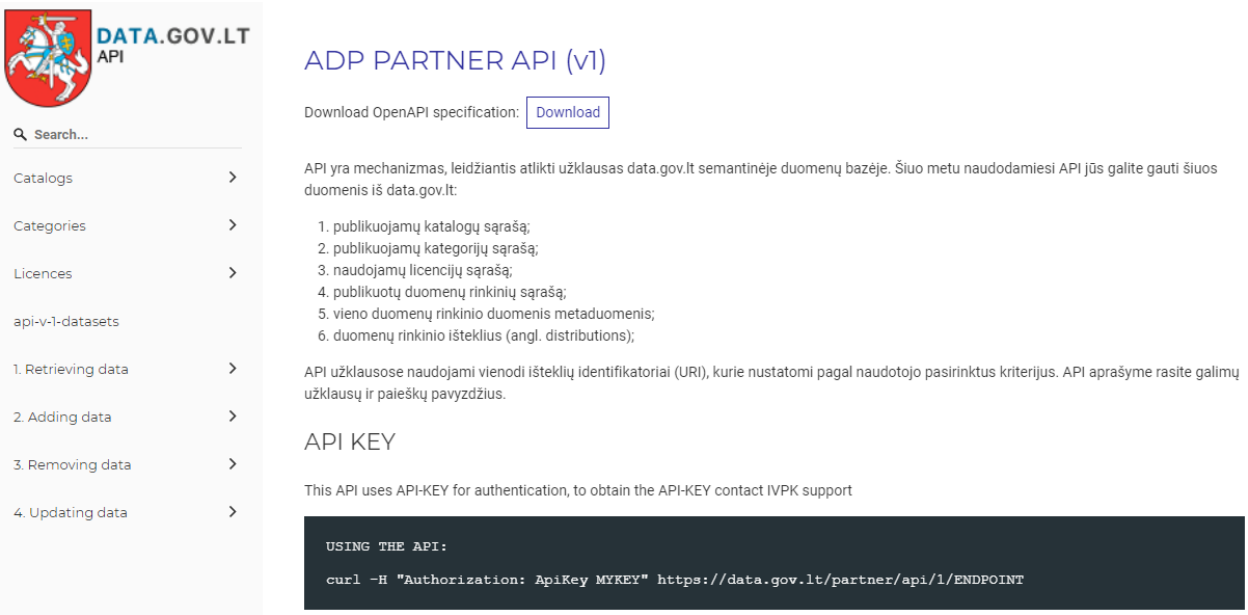
> Gauti organizacijai skirtą raktą:

1. Kreipkitės į Portalo vyr. koordinatorių.

> Pasiekti API:

1. Turėdami suteiktą raktą, naršyklėje įveskite nuorodą https://data.gov.lt/partner/api/1;

   Būsite nukreipti į API sąsajos puslapį su reikalinga informacija.



76 pav. API aplinkos fragmentas

7.12.1 Metaduomenų atnaujinimas

Jei Portale yra pateikta nuoroda į išorinį duomenų šaltinį, tada kiekvieną kartą kai atliekamas duomenų atnaujinimas, naudojantis Partnerių API reikia atnaujinti ir metaduomenis, kad būtų užfiksuotas duomenų atnaujinimo faktas.

Metaduomenis galite atnaujinti taip:

Python

```
import requests

# Configuration parameters (edited manually)
server = 'https://data.gov.lt/partner/api/1'
apikey = ''             # API Key (without semicolon).
dataset = 0             # Dataset id.
distribution = 0        # Distribution id.
data = {
    'url': '',          # URL to published data.
```

(continues on next page)

(tęsinys iš praeito puslapio)

```
    # For list of all parameters see:
    # https://data.gov.lt/partner/api/1#operation/
↪updateDatasetDistributionByIdUsingPATCH_1
}

# Metadata update
resp = requests.patch(
    f'{server}/datasets/{dataset}/distributions/{distribution}',
    headers={'Authorization': f'ApiKey {apikey}'},
    json=data,
)
resp.raise_for_status()
```

PHP

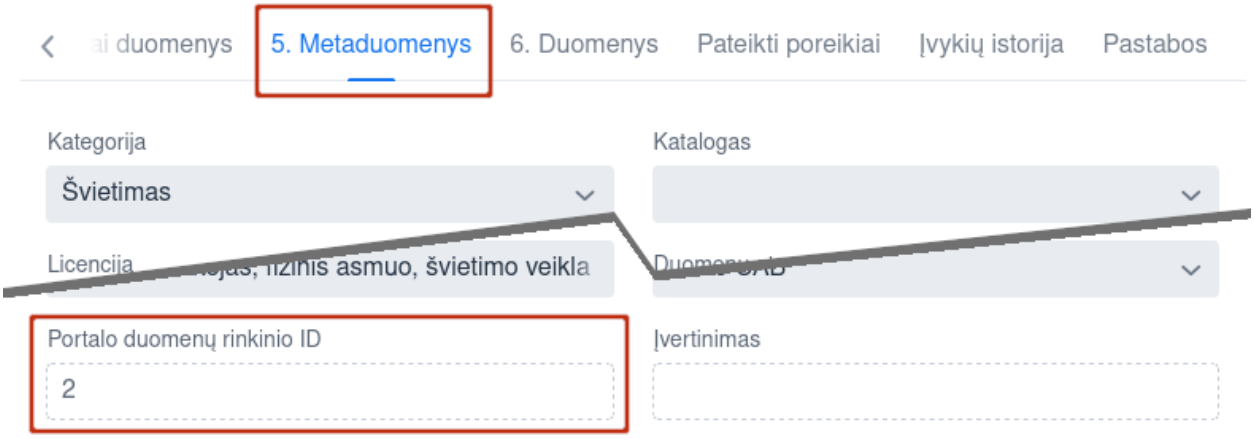
```
<?php

// Configuration parameters (edited manually)
$server = "https://data.gov.lt/partner/api/1";
$apikey = "";           // API Key (without semicolon).
$dataset = "";          // Dataset id.
$distribution = "";     // Distribution id.
$data = array(
    "url" => ""         // URL to published data.
    // For list of all parameters see:
    // https://data.gov.lt/partner/api/1#operation/
↪updateDatasetDistributionByIdUsingPATCH_1
);

# Metadata update
$curl = curl_init();
curl_setopt($curl, CURLOPT_CUSTOMREQUEST, "PATCH");
curl_setopt($curl, CURLOPT_POSTFIELDS, json_encode($data));
curl_setopt($curl, CURLOPT_URL,
            "$server/datasets/$dataset/distributions/$distribution");
curl_setopt($curl, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "Authorization: ApiKey $apikey"
));
curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);
$response = curl_exec($curl);
if (!$response) {
    die("Connection Failure");
}
$status = curl_getinfo($curl, CURLINFO_HTTP_CODE);
if ($status < 200 || $status >= 300) {
    die("Server responded with $status status code. " .
        "Full response from $server:\n\n$response");
}
curl_close($curl);

?>
```

Net jei distribucijos nuoroda nesikeičia, reikėtų įvykdyti šią API užklausą, kiekvieną kartą, kai buvo atnaujinti duomenys.

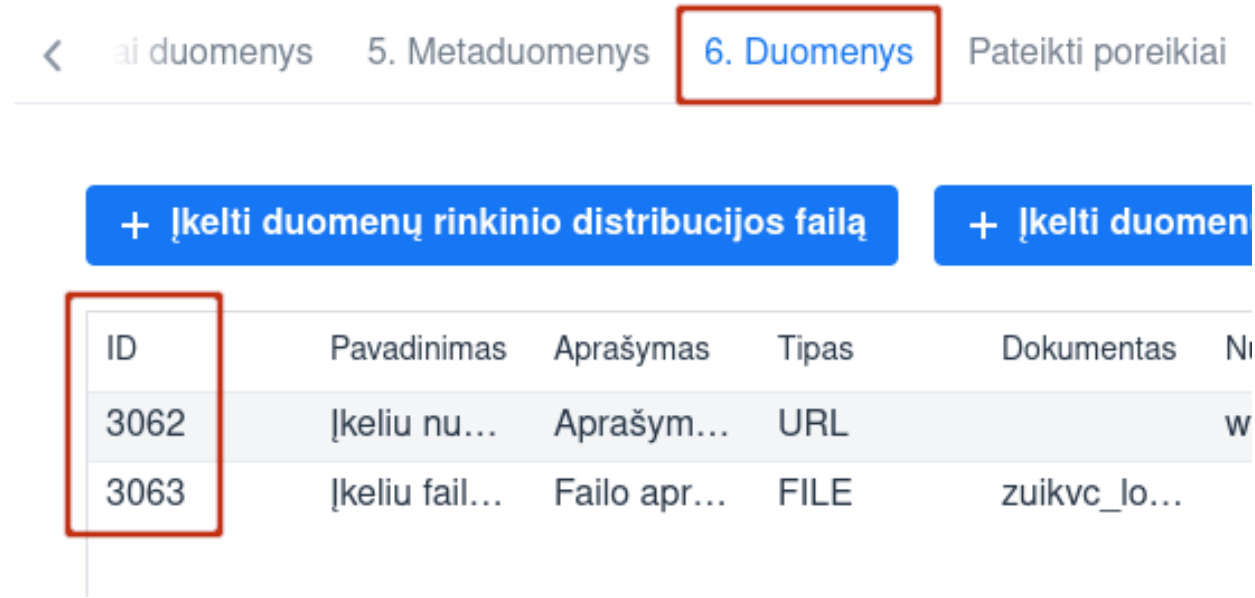
Toks iškvietimas reikalingas, kad būtų atnaujinta paskutinio duomenų atnaujinimo data. Kadangi patys duomenys nebūtinai keičiasi kiekvieno duomenų atnaujinimo metu, todėl būtina užfiksuoti patį atnaujinimo faktą.

Jei duomenys teikiami realiu laiku, tuomet šio API prieigos taško nebūtina kviesti.

dataset parametrą galite rasti čia:



distribution parametrą galite rasti čia:



Analogiškai *dataset* ir *distribution* parametrus galite gauti per API, pavyzdžiui naudojantis httpie, tai galite gauti taip:

```
$ SERVER=https://data.gov.lt/partner/api/1
$ AUTH="Authorization: ApiKey $API_KEY"
$ http GET $SERVER/datasets $AUTH
[
    {
        "id": 2,
        ...
```

(continues on next page)

(tęsinys iš praeito puslapio)

```
    },
    ...
]
$ http GET $SERVER/datasets/1857/distributions $AUTH
[
    {
        "id": 3062,
        ...
    },
    ...
]
```

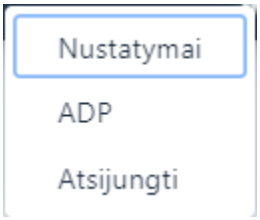
Šiame pavyzdyje, parametrai būtų tokie:

```
dataset = 2
distribution = 3062
```

Dėl išsamesnės informacijos apie Katalogo Partnerių API naudojimą, prašome žiūrėti Partnerių API dokumentacijoje.

7.13 Slaptažodžio keitimas

1. Prisijunkite prie sistemos.
2. Spustelėkite savo paskyros ikoną [] dešiniame ekrano kampe viršuje.
3. Po ikona išskleidžiamame kontekstiniame meniu spustelėkite **[Nustatymai]**.



77 pav. Paskyros kontekstinis meniu

4. Paskyros lange spustelėkite **[Keisti slaptažodį]**.
5. Slaptažodžio keitimo lange užpildykite visus privalomus laukus:

- dabartinį slaptažodį;
- naują slaptažodį;
- pakartokite naują slaptažodį.

Slaptažodis privalo būti saugus. (Daugiau: *Sąvokos*, „Saugus slaptažodis“)

78 pav. Slaptažodžio keitimo langas

6. Spauskite **[Keisti]**, kad išsaugotumėte naują slaptažodį.
7. Sekantį kartą jungiantis prie sistemos, prisijunkite naudodami naująjį slaptažodį.

SKYRIUS 8

Saugykla

Atvirų duomenų Saugykla yra sudedamoji atvirų duomenų Portalo dalis. Saugyklos paskirtis yra duomenų publikavimas *aukščiausiu brandos lygiu*, įvairiais formatais, per patogią mašininio duomenų skaitymo sąsają (*API*), laikantis aukščiausių duomenų publikavimo standartų.

Visi duomenų rinkiniai publikuojami Saugykloje yra apjungiami į vieną didelę duomenų bazę, kur duomenys gali būti jungiami tarpusavyje, pateikiami pilna apimtimi (angl. *in bulk*) arba pageidaujamais pjūviais. Suteikiamos priemonės duomenis gauti palaipsniniu būdu (angl. Incremental download).

Daugiau informacijos

8.1 Saugyklos diegimas

Jei dėl tam tikrų priežasčių negalite naudoti valstybinės duomenų publikavimo paslaugos pasiekiamos adresu get.data.gov.lt, tuomet galite Saugyklą pasileisti ir savo infrastruktūroje.

Čia rasite informaciją, kaip galite paleisti Saugyklą savo infrastruktūroje.

Saugykla veikia *Spinta* priemonės pagalba. *Spinta* gali veikit kaip komandinės eilutės įrankis, tačiau taip pat gali veikti ir duomenų publikavimo režimu.

Duomenų publikavimas gali veikti dviem režimais:

mode: internal Duomenys publikuojami iš Saugyklos vidinės duomenų bazės, kurioje laikoma šaltinio duomenų kopija. Vidinė duomenų bazė praturtina šaltinio duomenis metaduomenimis, kurie leidžia užtikrinti sklandesnį duomenų publikavimą.

mode: external Duomenys gali būti publikuojami tiesiai iš duomenų šaltinio (kol kas tai yra eksperimentinė funkcija, naudojama tik pasitikrinti, kaip atrodys duomenų publikavimas, prie atliekant realų duomenų perdavimą į Saugyklą).

Diegiant ir konfigūruojant duomenų publikavimo paslaugą rekomenduojame naudoti `model: internal` režimą, būtent toks režimas naudojamas ir get.data.gov.lt adresu.

8.1.1 Techniniai reikalavimai

Saugykla veikia ir yra testuota Linux operacinėse sistemose, konkrečiai naudojant Debian/Ubuntu distribucijas, todėl instrukcijos bus pateiktos būtent Debian/Ubuntu aplinkai. Diegimą galima atlikti ir kitose Linux distribucijose, tačiau tam tikros vietos nurodytos šioje dokumentacijoje turėtu būti priderintos taip, kad veiktų kitoje distribucijoje.

Saugykla yra sukurta naudojant Python programavimo kalbą, reikalinga Python 3.9 ar naujesnė versija. Naujose Saugyklos versija reikalavimas Python versijai gali keistis.

Dėl serverio resursų, tokių kaip CPU, RAM ir HDD, reikalingi resursai tiesiogiai priklauso nuo publikuojamų duomenų kiekio ir naudotojų srauto, kurie naudosis duomenų publikavimo paslauga.

Minimalūs reikalavimai Saugyklai, be duomenų ir su 5 naudotojais vienu metu besinaudojančiais publikavimo paslauga būtų 1 CPU, 512 Mb RAM ir 1G HDD laisvos vietos, kuri lieka pilnai įdiegus operacinę sistemą ir visas reikalingas priklausomybes.

Pati savaime Saugykla su visomis Python priklausomybes diske užima apie 500 Mb vietos, tačiau sunaudojamos vietos skaičius gali skirtis, skirtingose distribucijose.

Saugyklos veikimas turėtu būti nuolat stebimas ir reikiami resursai didinami, pagal poreikį.

8.1.2 Operacinės sistemos paruošimas

Saugykla turėtu būti diegiama ir leidžiama `spinta` naudotojo teisėmis, todėl reikia sukurti sisteminį naudotoją:

```
sudo useradd --system -g www-data --create-home --home-dir /opt/spinta spinta
```

Atkreipkite dėmesį, kad visose komandose, kurios prasideda `sudo`, komanda turi būti vykdoma administratoriaus teisėmis, tačiau visur kur nėra `sudo`, komanda turi būti vykdoma `spinta` naudotojo teisėmis. Tai yra svarbu, todėl nesupainiokite kokio naudotojo teisėmis vykdote komandas, priešingu atveju susidursite su sunkumais susijusiais su failų teisėmis.

8.1.3 PostgreSQL diegimas

Kai Saugykla veikia `mode: internal` režimu, jai reikalinga duomenų bazė, kurioje saugoma publikuojamų duomenų kopija. Rekomenduojame naudoti PostgreSQL:

```
sudo apt install postgresql
sudo -Hiu postgres createuser spinta
sudo -Hiu postgres createdb \
  --template=template0 \
  --owner=spinta \
  --encoding=UTF8 \
  --locale=C.UTF-8 \
  spinta
```

Žinomų apribojimų PostgreSQL versijai nėra. Rekomenduojame naudoti naujausią PostgreSQL versiją.

8.1.4 Python diegimas

Daugelis Linux distribucijų ateina su įdiegta Python versija, tačiau reikia įsitikinti, ar distribucijos Python versija yra pakankama:

```
python3 --version
```

Jei Python versija yra 3.9 ar naujesnė, tada galite pereiti prie sekančio žingsnio.

Jei versija yra žemesnė nei 3.9, tuomet reikės įsidiegti naujesnę Python versiją. Tai galite padaryti naudodami pyenv (dėl pačio pyenv diegimo skaitykite pyenv dokumentacijoje):

```
sudo apt update
sudo apt upgrade
sudo apt install -y \
  git make build-essential libssl-dev zlib1g-dev \
  libbz2-dev libreadline-dev libsqlite3-dev wget \
  curl llvm libncurses5-dev libncursesw5-dev \
  xz-utils tk-dev libffi-dev liblzma-dev \
  python-openssl
git clone https://github.com/pyenv/pyenv.git
export PYENV_ROOT=/opt/pyenv
/opt/pyenv/bin/pyenv install --list | grep -v - | tail
/opt/pyenv/bin/pyenv install 3.10.7  # Naudokite naujausią versiją
```

Naujausia Python versija bus įdiegta į `/opt/pyenv/versions/3.10.7/bin/python`.

Analogiškai, galite naudotis distribucijos teikiamais paketais, Ubuntu atveju galite daryti taip:

```
sudo apt update
sudo apt upgrade
sudo apt install software-properties-common
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt update
sudo apt install python3.10 python3.10-venv
```

Naujausia python versija bus pasiekiama `python3.10` komandos pagalba.

8.1.5 Spinta diegimas

Atkreipkite dėmesį, kad visos komandos diegiant Spinta turi būti vykdoma `spinta` naudotojo teisėmis ir iš `spinta` naudotojo namų katalogo `/opt/spinta`.

Aktyvų naudotoją ir katalogą galite pasikeisti taip:

```
sudo -Hsu spinta
cd
```

Saugykla veikia **Spinta_** priemonės pagalba, kuriai reikia Python. Rekomenduojama visus Python paketus diegti taip vadinamoje izoliuotoje Python aplinkoje, kurią galima susikurti taip (nepamirškite nurodyti jūsų naudojamos Python versijos numerį, kuris gali skirtis):

- Jei Python diegėte su **venv_**:

  ```
  /opt/pyenv/versions/3.10.7/bin/python -m venv env
  ```

- Jei Python diegėte distribucijos priemonėmis:

```
python3.10 -m venv env
```

Toliau spintą įdiegsite taip:

```
env/bin/pip install spinta
env/bin/spinta --version
```

8.1.6 Saugyklos konfigūravimas

Spinta yra konfigūruojama konfigūracijos failo pagalba, kurio, pagal nutylėjimą ieškoma aktyviame kataloge. Kur Spinta ieško konfigūracijos failo, galima patikrinti taip:

```
env/bin/spinta config config
```

Konfigūracijos failas vieta gali būti keičiama komandinės eilutės:

```
env/bin/spinta -o config=config.yml
```

Arba aplinkos kintamųjų pagalba:

```
SPINTA_CONFIG=/opt/spinta/config.yml
```

Kuriuos, taip pat galima pateikti ir `.env` faile.

Pats konfigūracijos failas `config.yml` turėtu atrodyti panašiai taip:

```yaml
config_path: /opt/spinta/config
default_auth_client: default
env: production
manifest: default

keymaps:
  default:
    type: sqlalchemy
    dsn: sqlite:////opt/spinta/var/keymap.db

backends:
  default:
    type: postgresql
    dsn: postgresql:///spinta

manifests:
  default:
    type: tabular
    path: /opt/spinta/manifest.csv
    backend: default
    mode: internal

accesslog:
  type: file
  file: /opt/spinta/logs/access.log
```

Prieš testuojant ar konfigūracija veikia, sukuriame reikalingus katalogus:

```
mkdir /opt/spinta/config
mkdir /opt/spinta/logs
mkdir /opt/spinta/var
```

Generuojame kriptografinius autorizacijos raktus:

```
env/bin/spinta genkeys
```

Sukuriame `default_auth_client`, kuriam suteiktos teisės bus naudojamos visiems neautorizuotiems klientams:

```
env/bin/spinta client add -n default --add-secret --scope - <<EOF
spinta_getone
spinta_getall
spinta_search
spinta_changes
EOF
```

Konkrečiai šiuo atveju suteikiamos visos skaitymo teisės.

Sukuriame klientą `myclient`, kuriam suteikiame rašymo teises (pasikeiskite kliento pavadinimą savo nuožiūra):

```
env/bin/spinta client add -n myclient --scope - <<EOF
spinta_set_meta_fields
spinta_getone
spinta_getall
spinta_search
spinta_changes
spinta_insert
spinta_upsert
spinta_update
spinta_patch
spinta_delete
EOF
```

Jei norite suteikti rašymo teistes tik į tam tikrą vardų erdvę, galite nurodyti tai `scope` pagalba, pavyzdžiui `spinta_datasets_gov_myorg_insert`, kas sureikia naujų objektų kūrimo teises į `datasets/gov/myorg` vardų erdvę.

Nepamirškite, kad `/opt/spinta/manifest.csv` faile, kaip nurodyta konfigūracijos faile, turite pateikti duomenų struktūros aprašą, kurio pagrindu veiks saugykla. Šioje vietoje reikia pateikti ne ŠDSA, o ADSA variantą, t.y. struktūros aprašą, kuriame pašalinta viskas, kas nereikalinga atvėrimui (žiūrėti *ŠDSA vertimas į ADSA*).

Galiausiai patikriname configūraciją:

```
env/bin/spinta config config backends manifests accesslog
```

Patikriname ar konfigūracijoje ir pateiktame struktūros apraše nėra klaidų.

```
env/bin/spinta check
```

Patikriname ar Spinta gali prisijungti prie duomenų bazės.

```
env/bin/spinta wait 1
```

Ir galiausiai, paleidžiame duomenų bazės migracijas, kurių metu pagal struktūros apraše pateiktus metaduomenis bus sukuriamos reikalingos lentelės PostgreSQL duomenų bazėje.

```
env/bin/spinta bootstrap
```

8.1.7 Web serverio diegimas ir konfigūravimas

Į Python virtualią aplinką įdiegiame Gunicorn:

```
env/bin/pip install gunicorn uvloop httptools
```

Sukuriame SystemD servisą (atkreipkite dėmesį, kad jūsų pasirinkta distribucija gali naudoti kitą servisų valdymo priemonę, tuomet šis pavyzdys netiks):

```
# /etc/systemd/system/gunicorn.service
[Unit]
Description=gunicorn daemon
After=network.target

[Service]
Type=notify
RuntimeDirectory=gunicorn
WorkingDirectory=/opt/spinta
EnvironmentFile=/opt/spinta/.env
ExecStart=/opt/spinta/env/bin/gunicorn -b 127.0.0.1:8000 -u spinta -g www-data -k␣
↪uvicorn.workers.UvicornWorker spinta.asgi:app
ExecReload=/bin/kill -s HUP $MAINPID
KillMode=mixed
TimeoutStopSec=5
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

Aktyvuokite servisą:

```
sudo systemctl enable gunicorn
sudo systemctl daemon-reload
sudo systemctl start gunicorn
```

Patikrinkite are servisas veikia:

```
sudo systemctl status gunicorn
```

Įdiegiame pasirinktą Web serverio paketą, šiuo atveju pavyzdys pateiktas Nginx:

```
sudo apt update
sudo apt install nginx
```

Sukuriame pasirinkto Web serverio, šiuo atveju Nginx, konfigūracijos failą (pakeiskite *example.org* į jūsų domeno pavadinimą):

```
# /etc/nginx/sites-available/example.org
server {
  listen 80;
  server_name example.org;

  location / {
    proxy_pass http://127.0.0.1:8000;
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
  }
}
```

Aktyvuojame konfigūracijos failą:

```
sudo ln -s /etc/nginx/sites-available/example.org /etc/nginx/sites-enabled/
```

Patikriname ar konfigūracija veikia:

```
sudo nginx -t
```

Perkraukite Nginx:

```
sudo systemctl restart nginx
```

Patikrinkite ar servisas veikia:

```
sudo systemctl restart nginx
```

Detalesnės instrukcijos apie tai, kaip konfigūruoti SSL sertifikatus ir kitus Gunicorn ar Nginx parametrus rasite minėtų projektų dokumentacijoje.

8.1.8 Spintos naujinimas

Norint atnaujinti Spinta versiją, jums reikia įvykdyti tokias komanas:

```
sudo -Hsu spinta
cd
env/bin/pip install --upgrade spinta
```

8.1.9 Struktūros aprašo naujinimas

Jei pasikeitė struktūros aprašas, jį galite atnaujinti taip:

```
scp manifest.csv example.org:/opt/spinta/manifest-new.csv
ssh example.org
sudo chown spinta:www-data /opt/spinta/manifest-new.csv
sudo -Hsu spinta
cd
env/bin/spinta check manifest-new.csv
cp manifest.csv manifest-old.csv
mv manifest-new.csv manifest.csv
```

(continues on next page)

(tęsinys iš praeito puslapio)

```
diff -y --suppress-common-lines manifest-old.csv manifest-new.csv
psql spinta
\q

env/bin/spinta bootstrap

exit  # spinta user
sudo systemctl restart gunicorn
sudo systemctl status gunicorn
exit  # server
```

Atkreipkite dėmesį, kad `spinta bootstrap` komanda gali sukurti tik naujas lenteles, kurios dar nebuvo sukurtos. Jei keitėsi lentelė, kuri jau buvo publikuojama anksčiau, tuomet, prieš `spinta bootstrap`, jums reikės pasižiūrėti, kas keitėsi (`diff` komanda) ir atitinkamai pakoreguoti duomenų bazės schema SQL užklausų pagalba (`psql` komanda).

Dažniausiai naudojamos SQL komandos schemos koregavimui pateiktos žemiau.

Lentelių sąrašo peržiūrą (PostgreSQL riboja lentelės pavadinimo ilgį, todėl ilgi pavadinimai gali būti trumpinami):

```
\dt "datasets/gov/myorg"*;
```

Lentelės trynimas:

```
drop table "datasets/gov/org/dataset/Model/:changelog";
drop table "datasets/gov/org/dataset/Model";
```

Lentelės pavadinimo keitimas:

```
alter table "datasets/gov/org/dataset/Model/:changelog"
    rename to "datasets/gov/org/dataset/Model2/:changelog";
alter table "datasets/gov/org/dataset/Model"
    rename to "datasets/gov/org/dataset/Model2";
```

Stulpelio pavadinimo keitimas:

```
alter table "datasets/gov/org/dataset/Model" rename column "abc" to "xyz";
```

Naujo stulpelio pridėjimas:

```
alter table "datasets/gov/org/dataset/Model" add "abc" integer;
```

Esamo stulpeio tipo keitimas:

```
alter table "datasets/gov/org/dataset/Model" alter "abc" type double precision;
```

8.1.10 Problemos ir sprendimai

Jei kažkas neveikia, pirmiausiai reikėtų žiūrėti servisų žurnalus, pavyzdžiui:

```
journalctl -u gunicorn -xe
journalctl -u nginx -xe
```

Žurnaluose dažniausiai būti pateikta visa informacija, kas suprasti kas ir kodėl neveikia.

8.2 Statusas ir planas

Priemonė „Spinta“ yra eksperimentinis projektas, šiuo metu aktyviai vystomas. Pagal programinės įrangos gyvavimo ciklo schema, „Spinta“ yra PRE-ALPHA etape.

Nors projektas yra aktyviai vystomas, tačiau jis jau yra naudojamas gamybinėje aplinkoje, kurią galite pasiekti šiais adresais:

https://get.data.gov.lt/ Saugyklos sritis skirta viešai atvirų duomenų prieigai. Ši sritis veikia tik skaitymo režimu ir skirta plačiajai visuomenės daliai.

https://put.data.gov.lt/ Saugyklos sritis skirta duomenų tiekėjams, kuriems suteikiama galimybė teikti duomenis į saugyklą. Ši sritis yra skirta tik duomenų tiekėjams.

Taip pat yra analogiškos aplinkos skirtos testavimui, prieš pereinant prie gamybinės aplinkos:

https://get-test.data.gov.lt/

https://put-test.data.gov.lt/

Kiekvienas pakeitimas projekto „Spinta“ kodo bazėje yra automatiškai testuojamas vykdant beveik 1000 testų, kurie dengia beveik 90% viso kodo. Todėl projektas yra gan stabilus.

8.2.1 Preliminarus projekto vystymo planas

Etapas	Pradžia	Pabaiga
PRE-ALPHA	2019 metų pradžia	2021 metų pabaiga
ALPHA	2021 metų pabaiga	2022 metų vidurys
BETA	2022 metų vidurys	2023 metų kovas
STABLE	2023 metų kovas	•

8.2.2 Ko tikėtis kiekvieno etapo metu?

PRE-ALPHA (iki 2021 metų pabaigos) Projektas jau bus naudojamas gamybinėje aplinkoje, tačiau reikėtu tikėtis, kad dalykai ne visada veiks, funkcijos bus ne iki galo išbaigtos ir esamas funkcionalumas gali keistis. Tačiau nepaisant minėtu trūkumų, esminės atvirų duomenų saugyklos funkcijos turėtu veikti gan stabiliai, todėl rekomenduojame aktyviai naudotis saugykla tiek teikiant, tiek gaunant duomenis, nes tik tokiu būdu geriau suprasime ko reikia galutiniam naudotojui.

Jei į saugyklą teikiate svarbius, didelę paklausą turinčius duomenis, rekomenduojame „Spinta“ projektą naudoti tik, kaip alternatyvią duomenų publikavimo priemonę, kartu publikuojant duomenis ir kitais kanalais, užtikrinančiais didesnį patikimumo lygį.

ALPHA (iki 2022 metų vidurio) Šio etapo metu didžiausias dėmesys bus skiriamas esamų funkcijų išbaigtumo didinimui, greitaveikos optimizavimui ir stabilumo didinimui. Šio etapo metu.

Taip pat šio etapo metu bus dirbama ir prie duomenų brandos kėlimo funkcijų įgyvendinimo, peržiūrint visus saugykloje publikuojamus duomenis ir siekiant juos transformuoti taip, kad jie būtų suderinami su Europos ir tarptautiniais standartais, kiek įmanoma atitiktų vieningą duomenų žodyną.

BETA (iki 2023 metų kovo mėnesio) Šio etapo metu, jokių naujų funkcijų kurti nebeplanuojama, bus didinamas esamų funkcijų stabilumas, greitaveika, taisomos pastebėtos klaidos.

Šio etapo metu rekomenduojame naudoti saugyklą, kaip pagrindinį atvirų duomenų publikavimo šaltinį.

STABLE (nuo 2023 metų kovo mėnesio) Šio etapo metu, bus vykdomas projekto palaikymas ir priežiūra, pastebėtų klaidų taisymas.

8.3 Prieš pradedant

8.3.1 Pavyzdžių skaitymas

Prieš pradedant skaityti verta susipažinti kokie įrankiai naudojami šios dokumentacijos pavyzdžiuose.

Pavyzdžiuose HTTP užklausos atliekamos naudojant patogią httpie programėlę.

Pavyzdžiui:

```
http /version
```

```
HTTP/1.1 200 OK
content-type: application/json

{
    "api": {
        "version": "0.0.1"
    },
    "build": null,
    "implementation": {
        "name": "Spinta",
        "version": "0.1.6"
    }
}
```

Šiame pavyzdyje `http` komanda buvo įvykdyta su `/version` argumentu.

Pavyzdžiuose nepateikiamas konkretus domenas, kadangi domenas gali skirtis priklausomai nuo to, kurią atvirų duomenų saugyklą naudojate, tačiau adreso dalis einanti po domeno visada vienoda.

Apart to, kad pavyzdžiuose nenurodomas domenas, visa kita veikia taip kaip parašyta httpie dokumentacijoje. Todėl rekomenduojame pirmiausia pasiskaityti su httpie dokumentacija, prieš pradedant gilintis į šią atvirų duomenų saugyklos dokumentaciją.

8.3.2 Duomenų modelis

Visuose pavyzdžiuose bus naudojamas vienas ir tas pats žemynų, šalių ir miestų duomenų modelis, kurios struktūra atrodo taip:

d	m	property	type	ref
datasets/gov/dc/geo				
	Continent			
		name	string	
	Country			
		code	string	
		name	string	
		continent	ref	continent
		capital	ref	city
	City			
		name	string	
		country	ref	country

Šalys priklauso žemynams, o miestai šalims. Kiekviena šalis turi vieną miestą sostinę.

8.4 API adreso struktūra

API yra generuojamas dinamiškai iš *DSA* `model` stulpelyje esančių modelio *kodinių pavadinimų*. Modelių pavadinimai gali turėti vardų erdves, vardų erdvės yra atskirtos `/` simboliu, pavyzdžiui:

```
/datasets/gov/dc/geo/Continent
```

Šis adresas sudarytas iš `datasets/gov/dc/geo` vardų erdvės ir `Continent` modelio pavadinimo.

`datasets` vardų erdvė rodo, kad duomenys yra žali, t.y. tokie kokie pateikti tam tikros įstaigos. Su laiku visų įstaigų duomenys bus transformuoti į vieningą šalies masto žodyną ir pavyzdžiui `datasets/gov/dc/geo/Continent` gali būti sulietas į vieną bendrą `Continent` modelį, šakninėje vardų erdvėje. Tačiau užtikrinant stabilų ir nekintantį API bus paliekami ir pradiniai žalių duomenų API taškai.

Konkrečiai visi `datasets` vardų erdvėje esantys modeliai turi aiškiai apibrėžtą struktūrą, pavyzdžiui nagrinėjant `datasets/gov/dc/geo/Continent` pavyzdį atskirų kelio komponentų prasmės būdų tokios:

- `datasets/` - vardų erdvė skirta žaliems pirminiams įstaigų duomenimis.
- `gov/` - vardų erdvė skirta valstybinių įstaigų duomenimis.
- `dc/` - konkrečios valstybinės įstaigos trumpinys.
- `geo/` - įstaigos atverto duomenų rinkinio trumpinys.
- `Continent` - duomenų modelis (arba lentelė).

8.4.1 Adreso parametrai

Adresas gali turėti parametrus, parametrai atrodo taip:

```
/:ns
/:changes
/:format/csv
```

Jei adreso kelio elementas prasideda `:` simboliu, tai po jo seka parametro pavadinimas ir jei konkretus parametras turi argument, gali sekti vienas ar daugiau argumentų.

8.4.2 Identifikatorius

Identifikatorius pat yra adreso parametras, tik jis neprasideda `:` simboliu, o eina iš karto po modelio pavadinimo ir tai turi būti UUID simbolių eilutė, pavyzdžiui:

```
/datasets/gov/dc/geo/Continent/77e0bb52-f8ae-448f-b4c2-7de6bb150ff0
```

Identifikatorius taip pat gali turėti argumentus, identifikatoriaus argumentai yra modelio savybė, pavyzdžiui:

```
/datasets/gov/dc/geo/City/7d473db2-d363-4318-9b84-138eb5d70f70/continent
```

Tokiu būdu yra galimybė gauti tik konkrečios modelio savybės duomenis.

8.4.3 Užklausa

URL Query dalyje, po `?` simbolio galima pateikti papildomus užklausos parametrus, pavyzdžiui:

```
/datasets/gov/dc/geo/Continent?select(name)&sort(name)
```

8.5 Rezervuoti pavadinimai

Vadovaujantis duomenų struktūros aprašo taisyklėmis, vardų erdvės, modeliai ir laukai turi būti pavadinti laikantis *kodinių pavadinimų* reikalavimų.

Visi pavadinimai, kuri prasideda _ simboliu, turi specialią prasmę ir praturtina duomenis, tam tikrais metaduomenimis.

Naudojami tokie specialieji pavadinimai:

_type Modelio ar vardų erdvės pavadinimas.

_id Unikalus objekto pavadinimas UUID formatu.

_revision Objekto revizijos numeris, šio numerio pagalba užtikrinamas duomenų vientisumas keičiant duomenis. Kiekvieną kartą keičiant duomenis būdina nurodyti ir revizijos numeris, keitimas leidžiamas tik tuo atveju, jei išsaugoto objekto ir keitimo revizijos numeriais sutampa. Kiekvieną kartą, kai objektas pasikeičia, keičiasi ir jo revizijos numeris.

_txn Transakcijos numeris. Vienos užklausos metu, gali būti keičiama daug objektų, visiems, vienu kartu keičiamiems objektas suteikiamas vienas transakcijos keitimo numeris.

_data Objektų sąrašas, kei keli objektai pateikiami kartu.

_cid Keitimo numeris naudojamas *changes* užklausos metu.

_created Data ir laikas, kada duomenys buvo keisti, naudojamas *changes* užklausos metu.

_op Užklausos veiksmo pavadinimas. Plačiau *Veiksmai*.

_where Objekto atranka naudojama `upsert` užklausose.

8.6 Vardų erdvės

Atvirų duomenų saugykla yra didelis katalogas, kuriame sudėta įvairių modelių duomenys. Katalogai vadinami vardų erdvėmis.

Aukščiausiame lygyje yra globali vardų erdvė:

```
http /
```

```
HTTP/1.1 200 OK
content-type: application/json

{
    "_data": [
        {
            "_id": "datasets/:ns",
            "_type": "ns",
            "title": "datasets"
        }
    ]
}
```

Globalioje vardų erdvėje yra kita vardų erdvė `datasets`. Žinome, kad `datasets` yra vardų erdvė, kadangi tai nurodyta `_type` savybėje, kurios reikšmė `ns`, kas reiškia *Name Space* arba *Vardų Erdvė* išvertus į Lietuvių kalbą.

Į vardu erdves reikia kreiptis nurodant `/:ns` parametrą:

```
http /datasets/gov/dc/geo/:ns
```

```
HTTP/1.1 200 OK
content-type: application/json

{
    "_data": [
        {
            "_id": "datasets/gov/dc/geo/Continent",
            "_type": "model",
            "title": "Continent"
        },
        {
            "_id": "datasets/gov/dc/geo/Country",
            "_type": "model",
            "title": "Country"
        },
        {
            "_id": "datasets/gov/dc/geo/City",
            "_type": "model",
```

(continues on next page)

(tęsinys iš praeito puslapio)

```
            "title": "City"
        }
    ]
}
```

Jei `/:ns` parametras nebūtų nurodytas, tada saugykla bandytų ieškoti modelio pavadinimu `datasets/gov/dc/geo` ir neradus tokio modelio būtų gražintas `404 Not Found` klaidos kodas.

8.7 Formatas

Saugykloje duomenys saugomi taip, kad juos būtų galima gauti įvairiais formatais.

Reikia atkreipti dėmesį, kad nors Saugykla gali duomenis grąžinti įvairiais formatais, tačiau nėra galimybės duomenis gauti konkretaus formato schemos pavidalu. Pavyzdžiui turint naujienų duomenis, nėra galimybės tokių duomenų gauti RSS formatu. Pats savaime RSS yra konkreti XML formato schema. Todėl Saugykla gali grąžinti duomenis XML formatu, specifine Saugyklose schema.

Saugykloje palaikomi tik bendrieji formatai, specializuoti, tam tikros srities formatai nepalaikomi. Norint gauti duomenis tam tikru specializuotu formatu, Saugykloje teikiamus duomenis reikia konvertuoti į pageidaujamą specializuotą formatą.

Siekiant užtikrinti duomenų perdavimo ir skirtingų formatų palaikymą, visiems duomenų laukams, modeliams ir vardų erdvėms taikomi *kodinių pavadinimų* reikalavimai, *specialiosios paskirties duomenys* pateikiami su _ prefiksu, tokiu būdu atskiriant duomenis, nuo metaduomenų.

Grąžinant duomenis tam tikru formatu, gražinamų duomenų schema gali būti skirtinga *skirtingo pobūdžio užklausoms*. Pavyzdžiui, jei duomenų prašoma `getone` būdu, JSON formatu, tuomet rezultatas bus:

```
{
    "_type": "...",
    "_id": "...",
    "data": "...",
}
```

Jei duomenų prašoma `getall` būdu, tada rezultatas bus toks:

```
{
    "_type": "...",
    "_data": [
        {
            "_id": "...",
            "data": "...",
        }
    ]
}
```

Duomenų kitu formatu galima paprašyti adreso gale nurodžius `/:format/csv`, kur `csv` yra pageidaujamas formatas. Šiuo atveju, bus grąžintas toks rezultatas:

```
_type,_id,data
...,...,...
```

Rezervuotų laukų sąrašas pateiktas skyriuje *Rezervuoti pavadinimai*.

8.8 Ryšiai tarp objektų

Plačiau apie ryšius tarp objektų struktūros apraše skaitykite *Ryšiai tarp modelių*.

Duomenų laukai, kurių `property.type` yra `ref` įgauna objekto, į kurį rodoma formą. Tarkime, jei turime tokį struktūros aprašą:

d	r	b	m	property	type	ref	level
datasets/gov/example/countries							
			Country				4
				name	string		4
			City				4
				name	string		4
				country	ref	Country	4

Tada `City.country` įgaus `Country` objekto pavidalą:

```
{
    "_type": "datasets/gov/example/countries/City",
    "_id": "78035ad0-8f59-4d59-8867-60ea856ba26f",
    "name": "Vilnius",
    "country": {
        "_id": "3c65deaa-8ef8-46d9-8b00-38b22bb91f95"
    }
}
```

Tokia forma naudojama todėl, kad Saugykla leidžia jungti skirtingų modelių duomenis tarpusavyje. Todėl, tam tikrais atvejais, gali būti pateikiamas ne tik siejamo objekto identifikatorius, bet ir kiti to objekto laukai, pavyzdžiui, jei duomenų atrankai naudotume tokią užklausą:

```
/City?select(_id, name, country._id, country.name)
```

Tuomet atsakymas būtų toks:

```
{
    "_id": "78035ad0-8f59-4d59-8867-60ea856ba26f",
    "name": "Vilnius",
    "country": {
        "_id": "3c65deaa-8ef8-46d9-8b00-38b22bb91f95",
        "name": "Lietuva"
    }
}
```

Tais atvejais, kai prašomas tik objektas, kuris yra ryšys su kitu objektų, pavyzdžiui:

```
/City?select(name, country)
```

Gausime tokį atsakymą:

```
{
    "name": "Vilnius",
    "country": {
        "_id": "3c65deaa-8ef8-46d9-8b00-38b22bb91f95",
    }
}
```

Tie patys duomenys CSV formatu būtų pateikti taip:

```
name,country._id
Vilnius,3c65deaa-8ef8-46d9-8b00-38b22bb91f95
```

8.8.1 3 ir žemesnis brandos lygis

Atkreipkite dėmesį, kad ryšiai tarp objektų, taip, kaip aprašyta aukščiau, veikia tik tuo atveju, jei *ref* duomenų lauko brandos lygis (*level*) yra didesnis nei 3.

Jei *ref* duomenų lauko brandos lygis yra 3 ar žemesnis, tada jungimui naudojamas jungiamo modelio pirminis raktas (*model.ref*) arba nepirminis raktas nurodytas prie pačio *ref* lauko (žiūrėti *Jungimas per nepirminį raktą*). Jei nenurodytas nei *model.ref*, nei *kitas laukas*, tada jungimas daromas per `_id` lauką, tačiau netikrinama ar toks `_id` egzistuoja jungiamame modelyje.

Pavyzdžiui, jei turime tokį struktūros aprašą:

d	r	b	m	property	type	ref	level
datasets/gov/example/countries							
			Country			code	4
				code	string		4
				name	string		4
			City				4
				name	string		4
				country	ref	Country	3

Tada, `City.country` jungimas su `Country` galimas tik per `code`, o ne per `_id`, kadangi `City.country` yra 3 brandos lygio, kas nurodo, kad jingimas daromas ne per pirminį raktą.

Šiuo atveju, `City` objektas atrodys taip:

```
{
    "_type": "datasets/gov/example/countries/City",
    "_id": "78035ad0-8f59-4d59-8867-60ea856ba26f",
    "name": "Vilnius",
    "country": {
        "code": "lt"
    }
}
```

Analogiškai, jei `Country` modelis neturi :data;`model.ref`, tada jungimas daromas, per `Country` lauką, kuris nurodytas `City.country` *`property.ref`* stulpeyje, pavyzdžiui:

d	r	b	m	property	type	ref	level
datasets/gov/example/countries							
			Country				4
				code	string		4
				name	string		4
			City				4
				name	string		4
				country	ref	Country[code]	3

Šiuo atveju, `City` objektas atrodys lygiai taip pat, kaip anksčiau:

```
{
    "_type": "datasets/gov/example/countries/City",
    "_id": "78035ad0-8f59-4d59-8867-60ea856ba26f",
    "name": "Vilnius",
    "country": {
        "code": "lt"
    }
}
```

Kadangi prie `City.country` nurodyta, kad jungimas daromas per `code` (`Country[code]`).

Jei nenurodytas ne modelio su kuriuo daromas jungimas priminis raktas, nei prie lauko nurodyta, per kurį modelio lauką daromas jungimas, tada jungimas daromas per `_id`, netikrinant ar toks `_id` egzistuoja ar ne. Pavyzdžiui:

d	r	b	m	property	type	ref	level
datasets/gov/example/countries							
			Country				4
				code	string		4
				name	string		4
			City				4
				name	string		4
				country	ref	Country	3

Šiuo atveju, `City` objektas atrodys taip:

```
{
    "_type": "datasets/gov/example/countries/City",
```

(continues on next page)

(tęsinys iš praeito puslapio)

```
    "_id": "78035ad0-8f59-4d59-8867-60ea856ba26f",
    "name": "Vilnius",
    "country": {
        "_id": "f23bb52b-bb55-4072-92b8-a8d9d77c9849"
    }
}
```

Tačiau `Country` objektas su `_id f23bb52b-bb55-4072-92b8-a8d9d77c9849` gali ir neegzistuoti, joks tikrinimas nedaromas.

8.9 Autorizacija

Norint gauti atvirus duomenis autorizacija nereikalinga, tačiau norint keisti saugykloje esančius duomenis are įkelti naujus, būtina autorizacija.

Autorizacija atliekama OAuth standarto pagalba. Kol kas yra palaikoma tik client credentials autorizavimo būdas.

Norint atlikti rašymo operacijas, pirmiausiai reikia, kad saugykloje jums būtų sukurta paskyra. Tada naudodamiesi paskyros prisijungimo duomenimis galite gauti autorizacijos raktą, kuris leis atlikti rašymo operacijas.

Autorizacijos raktas gaunamas taip:

```
SERVER=https://put.data.gov.lt
CLIENT=myclient
SECRET=mysecret
SCOPES="
spinta_set_meta_fields
spinta_getone
spinta_getall
spinta_search
spinta_changes
spinta_datasets_gov_myorg_insert
spinta_datasets_gov_myorg_upsert
spinta_datasets_gov_myorg_update
spinta_datasets_gov_myorg_patch
spinta_datasets_gov_myorg_delete
spinta_datasets_gov_myorg_wipe
"
TOKEN=$(
    http \
        -a $CLIENT:$SECRET \
        -f $SERVER/auth/token \
        grant_type=client_credentials \
        scope="$SCOPES" \
    | jq -r .access_token
)
AUTH="Authorization: Bearer $TOKEN"
echo $AUTH

http "$SERVER/" "$AUTH"
```

Pavyzdyje `$SCOPES` kintamasis yra tarpo simboliais atskirtų leidimu sąrašas. Leidimų pavadinimai formuojami taip:

```
spinta_$ns_$action
spinta_$model_$action
spinta_$model_$property_$action
```

`$action` reikšmės aprašytos skyriuje *Veiksmai*.

Kiekvienas klientas gali gauti skirtingą `$SCEOPES` leidimų sąrašą ir jei prašysite daugiau leidimų, nei jūsų klientui yra suteikta, gausite klaidą.

8.10 Veiksmai

Užklausos skirstomos į šiuos veiksmus:

getone Vieno objekto duomenys, pagal pateiktą `_id`.

getall Visų objektų duomenys.

search Objektų duomenys taikant duomenų atrankos užklausą.

changes Keitimų žurnalas.

insert Naujo objekto kūrimas.

upsert Pilnas objekto perrašymas arba naujo objekto kūrimas. Jei pagal pateiktą užklausą objektas egzistuoja, tuomet jis perrašomas, jei objektas neegzistuoja, tuometi sukuriamas naujas.

update Pilnas objekto perrašymas, net jei pateikiami ne visi laukai, objektas perrašomas pilnai, laukams, kurie nebuvo pateikti suteikiant pirmines reikšmes.

patch Dalinis objekto keitimas, kai keičiamas ne visas, o tik dalis objekto, tam tikri objekto laukai.

delete Duomenų šalinimas. Tokiu būdu pašalinti duomenys išsaugomi keitimų žurnale.

wipe Pilnas ir neatstatomas duomenų pašalinimas, naudojama tik testavimo tikslams.

8.11 Skaitymo veiksmai

8.11.1 getall

Šios užklausos pagalba galima gauti visus konkretaus modelio ar visos vardų erdvės duomenis.

```
http GET /datasets/gov/dc/geo/Continent
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "_type": "datasets/gov/dc/geo/Continent",
    "_data": [
        {
            "_id": "abdd1245-bbf9-4085-9366-f11c0f737c1d",
            "_revision": "16dabe62-61e9-4549-a6bd-07cecfbc3508",
            "_txn": "792a5029-63c9-4c07-995c-cbc063aaac2c",
            "continent": "Europe"
```

(continues on next page)

(tęsinys iš praeito puslapio)

```
        }
    ]
}
```

Taip pat žiūrėkite:

- *Rūšiavimas*
- *Objektų skaičius*
- *Duomenų skaitymas paketais*

8.11.2 getone

Šios užklausos pagalba galima gauti vieną konkretų objektą pagal unikalų objekto identifikatorių. Pavyzdžiui

```
http GET /datasets/gov/dc/geo/Continent/abdd1245-bbf9-4085-9366-f11c0f737c1d
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "_type": "datasets/gov/dc/geo/Continent",
    "_id": "abdd1245-bbf9-4085-9366-f11c0f737c1d",
    "_revision": "16dabe62-61e9-4549-a6bd-07cecfbc3508",
    "_txn": "792a5029-63c9-4c07-995c-cbc063aaac2c",
    "continent": "Europe"
}
```

Taip pat žiūrėkite:

- *getall*

8.11.3 changes

Šios užklausos pagalba galima gauti visų duomenų keitimų sąrašą. Ši užklausa yra skirta tęstiniam duomenų atnaujinimui. Tam, kad nereikėtų kiekvieną kartą iš naujo siųsti visų duomenų. Šios funkcijos pagalba, galite gauti tik tai, kas pasikeitė, nuo jūsų paskutinio siuntimo.

Atkreipkite dėmesį, kad pasikeitimų žurnalas yra skirtas tik duomenų sinchronizavimui ir keitimai yra periodiškai išvalomi. Jei norite gauti istorinius duomenis, tuomet tai turėtu atsispindėti pačiuose duomenyse, o ne keitimų žurnale. Keitimų žurnalas nėra skirtas istorinių duomenų saugojimui.

Norint sinchronizuoti duomenis, prieš pradedant sinchronizavimą, pirmiausiai rekomenduojama atsisiųsti visus duomenis naudojant *getall* užklausą. Siųsti viso keitimo žurnalo nėra prasmės, kadangi žurnalas yra nuolat valomas ir jame paliekama tik kelių savaičių ar kelių mėnesių keitimai, priklausomai nuo duomenų rinkinio.

Tačiau prieš siunčiant visus duomenis, reikia išsisaugoti paskutinio keitimo ID, tai galite padaryti šios užklausos pagalba:

```
http GET /datasets/gov/dc/geo/Continent/:changes/-1
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "_type": "datasets/gov/dc/geo/Continent",
    "_data": [
        {
            "_cid": "11",
            "_id": "abdd1245-bbf9-4085-9366-f11c0f737c1d",
            "_revision": "16dabe62-61e9-4549-a6bd-07cecfbc3508",
            "_txn": "792a5029-63c9-4c07-995c-cbc063aaac2c",
            "_created": "2021-07-30T14:03:14.645198",
            "_op": "insert",
            "continent": "Europe"
        }
    ]
}
```

Čia argumentas `-1` nurodo, kad norite gauti paskutinį įrašą keitimų žurnale. Šiame pavyzdyje, paskutinio keitimo ID yra 11, keitimo ID yra nurodomas `_cid` duomenų lauke.

Išsisaugojus paskutinio keitimo ID, galite atsisiųsti visus duomenis naudodamiesi *getall* funkcija.

Atsisiuntus duomenis, galite pradėti sinchronizavimą, tokios užklausos pagalba:

```
http GET /datasets/gov/dc/geo/Continent/:changes/11?limit(100)
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "_type": "datasets/gov/dc/geo/Continent",
    "_data": [
        {
            "_cid": "11",
            "_id": "abdd1245-bbf9-4085-9366-f11c0f737c1d",
            "_revision": "16dabe62-61e9-4549-a6bd-07cecfbc3508",
            "_txn": "792a5029-63c9-4c07-995c-cbc063aaac2c",
            "_created": "2021-07-30T14:03:14.645198",
            "_op": "insert",
            "continent": "Europe"
        }
    ]
}
```

Šiuo atveju, užklausoje nurodomas išsaugotas keitimo id 11.

Atsakyme visada pateikiamas keitimas, kurio ID nurodėte. Rekomenduojama visada patikrinti ar pirmas grąžintas keitimas yra būtent tas, kurį nurodėte, papildomai rekomenduojama patikrinti ar sutampa `_id` ir `_revision` su tuo, ką jūs gavote paskutinį kartą.

Toks patikrinimas reikalingas tam, kad įsitikinti, ar keitimų žurnalas nebuvo išvalytas arba duomenys nebuvo perkrauti iš naujo. Jei patikrinimo metu, kažkas nesutampa, toliau sinchronizavimą reikėtų nutraukti ir pakartoti viską iš naujo, gaunat paskutinio keitimo ID, atsisiunčiant duomenis ir tik tada sinchronizuojant.

Rekomenduojame sinchronizavimą daryti nedideliais puslapiais, pavyzdžiui po 100 įrašų, puslapio dydis nurodomas

`limit(100)` parametro pagalba. Nuskaičius vieną, puslapį, jei gavote daugiau nei vieną keitimą, tada galite kartoti užklausą dar kartą, su paskutiniu gautu keitimo ID, tol, kol gausite vieną keitimą, tą kurį nurodėte kaip argumentą.

Jei daugiau keitimų nėra, sekančią užklausą rekomenduojame daryti tada, kai praeina prie duomenų rinkinio nurodytas duomenų atnaujinimo periodiškumas. Pavyzdžiui, jei duomenys atnaujinami kasdien, tada sinchronizavimą taip pat reikėtų daryti kasdien.

Dažnesnis sinchronizavimas nei nurodytas duomenų atnaujinimo periodiškumas yra beprasmis, kadangi visais atvejais gausite tuščią keitimų sąrašą.

Keitimų įrašai turi tokius metaduomenis apie kiekvieną keitimą:

_cid Monotoniškai didėjantis keitimo numeris (keitimo ID).

_id Pakeisto objekto unikalus numeris (UUID formatu).

_revision Objekto revizijos numeris, naudojamas norint atlikti keitimo operaciją. Kiekvieną kartą, kai duomenys keičiami išduodamas naujas revizijos numeris.

_txn Tranzakcijos numeris, nurodo kokie keitimai buvo atlikti konkrečios keitimo tranzakcijos metu.

_created Data ir laikas, kada buvo atliktas keitimas.

_op Keitimo operacijos pavadinimas, gali būti tokie variantai:

insert Sukurtas naujas objektas.

patch Objekto keitimas, pateikiami tik toks reikšmės, kurios buvo pakeistos. Atkreipkite dėmesio, kad jei tam tikras laukas nebuvo pakeistas, jo keitimo įraše nebus. Jei laukas anksčiau buvo ir dabar yra ištrintas, tada keitimo įraše ištrintasis laukas bus pateiktas su `null` reikšme.

delete Objektas buvo ištrintas.

Atkreipkite dėmesį, kad vienam objektui, gali būti viena `insert` operacija, daug `patch` operacijų ir vienas `delete`.

Įgyvendinant duomenų atnaujinimą `:changes` protokolu, reikia interpretuoti `_op` reikšmę ir atlikti atitinkamą operaciją savo pusėje.

Pavyzdžiui, jei gavote `_op=insert`, tuomet, savo pusėje, galite susikurti naują įrašą. Jei gavote `_op=patch`, tada, savo pusėje turėtumėte atnaujinti įrašą, pagal nurodyti `_id`. Ir atitinkamai, jei gavote `_op=delete`, tai reiškia, kad objektas buvo ištrintas, turėtumėte jį ištrinti ir savo pusėje, pagal `_id` reikšmę.

Keitimu žurnalas yra naudingas, tais atvejais, kai duomenys dažnai keičiasi ir norite greitai gauti informaciją, kas ir kada pasikeitė. Taip pat, jei duomenų yra labai daug ir jų siuntimas užtrunka daug laiko, tada galite atsisiųsti duomenis vieną kartą ir toliau sekti keitimus.

8.12 Duomenų užklausos

Visos duomenų užklausos yra pateikiamos URL query dalyje, po `?` žymės. Tai kas yra rašoma po `?` yra *Formulės*, tokios pačios, kurios yra naudojamos duomenų struktūros aprašo *`prepare`* stulpelyje.

Atkreipkite dėmesį, kad dalis einanti po `?` turi būti tinkamai koduota naudojanti RFC 3986 specifikaciją. Kai kurie klientai tai padaro automatiškai, kai kurie ne. Visuose pavyzdžiuose pateikiamas nekoduotas variantas dėl geresnio skaitomumo.

Viso užklausoje naudojamos funkcijos yra atskiriamos `&` simboliu. Pavyzdžiui:

```
/example/Model?(col1="xyz"|col2<42)&select(col1,col2)&sort(col3)&limit(5)
```

8.12.1 Jungimas

Duomenis galima sujungti, jei duomenų struktūroje yra nurodyti ryšiai tarp modelių.

Jungimas atliekamas `.` (taško) pagalba, pavyzdžiui, norime prie miestų duomenų prijungti šalis, kurios pateiktos atskirame modelyje tuomet jungimą galime atlikti taip:

```
/example/City?select(country.name)
```

Čia, `country` laukas priklauso miesto modeliui, o `name` laukas priklauso šalied modeliui.

Taip apjungtus duomenų laukus galima naudoti atranko, filtravime ir rūšiavime.

8.12.2 Atranka

Yra galimėbė gauti ne visus duomenų laukus, o tik tim tikrus, kurie nurodyti užklausoje. Tokią atranką galima atlikti naudojant `select()` funkciją. Pavyzdžiui:

    /example/Model?select(prop1, prop2, prop3)

8.12.3 Filtravimas

Galima atlikti duomenų atranką, pagal pateiktą filtrą. Palaikomi tokie filtravimo operatoriai:

- `prop = value` - atranka pagal tikslią reikšmę,
- `prop != value` - atranka objektų, kurių reikšmė neatitinka nurodytai,
- `prop` [>, >=, <, <=] `value` - atranka palyginant ar reikšmų didesnė ar mažesnė.
- `prop.contains(value)` - atranka objektus, jei `value` yra `prop` reikšmės dalis.
- `prop.startswith(value)` - atrenka objektus, kurie prasideda `value` reikšme.

Tais atvejais jei `value` yra simbolių eilutė, reikia naudoti kabutes.

Filtruojanti taip pat galima naudoti AND ir OR operatorius, pavyzdžiui:

```
/example/Model?prop=value1&prop=value2

/example/Model?prop=value1|prop=value2

/example/Model?prop=value1&(prop=value2|prop=value3)
```

8.12.4 Rūšiavimas

Duomenis rūšiuoti galima pasitelkus `sort()` funkciją:

- `sort(prop)` - rūšiuoja didėjančia tvarka.
- `sort(-prop)` - rūšiuoja mažėjančia tvarka.

Kalima rūšiuoti pagal kelis stulpelius, pavyzdžiui:

```
/example/Model?sort(prop1,-prop2,prop3)
```

8.12.5 Objektų skaičiaus ribojimas

Norint apriboti grąžinamų objektų skaičių galima naudoti `limit()` funkciją, pavyzdžiui:

/example/Model?limit(10)

Tokia užklausa grąžins ne daugiau kaip 10 objektų.

8.12.6 Objektų skaičius

Norinti gauti vieno modelio objektų skaičių galima panaudoti `count()` funkciją:

```
http GET /datasets/gov/dc/geo/Continent?count()
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "_type": "datasets/gov/dc/geo/Continent",
    "_data": [
        {
            "count()": 3856
        }
    ]
}
```

8.12.7 Duomenų skaitymas paketais

Norint gauti visus tam tikro modelio duomenis, ne visus iš karto, o tam tikro dydžio paketais, galima duomenų skaitymą atlikti taip:

```
http GET /datasets/gov/dc/geo/Continent?sort(_id)&limit(1)&_id>"36cec98e-7237-43a5-ad2a-
↪58cf29d65e96"
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "_type": "datasets/gov/dc/geo/Continent",
    "_data": [
        {
            "_id": "abdd1245-bbf9-4085-9366-f11c0f737c1d",
            "_revision": "16dabe62-61e9-4549-a6bd-07cecfbc3508",
            "_txn": "792a5029-63c9-4c07-995c-cbc063aaac2c",
            "continent": "Europe"
        }
    ]
}
```

Šiame pavyzdyje užklausoje naudojami tokie parametrai:

`sort(_id)` Rūšiuojame duomenis pagal `_id` lauko reikšmes.

`limit(1)` Ribojame grąžinamų objektų skaičių iki 10, tai reiškia, kad mūsų paketo dydis bus 10 objektų.

> Gražinamų duomenų kiekį galima riboti ne tik įrašų skaičiumi, tačiau ir duomenų kiekiu, nurodant `limit("1M")`, kur `1M` reiškia gražinamų duomenų kiekį megabaitais.

`_id>"36cec98e-7237-43a5-ad2a-58cf29d65e96"` Atrenkame tik tuos duomenis, kurie yra didesni už nurodytą reikšmę, šiuo atveju reikšmė yra `36cec98e-7237-43a5-ad2a-58cf29d65e96`.

Kadangi `_id` yra unikalus, todėl šis laukas, gali būti naudojamas, kaip kursorius, skaitant duomenis paketais. Nuskaičius kiekvieną paketą, norint gauti sekantį, reikia pakeisti `_id>"?"` klaustuku pažymėtą vietą, paskutinio paketo objekto `_id` reikšme`.

Norint iš karto žinoti, kiek viso bus paketų, galim pirmiausiai *užklausti kiek viso yra objektų* ir gautą skaičių padalinti iš paketo dydžio.

8.13 Rašymo veiksmai

Atliekant rašymo veiksmus, Saugykla priima viską, ką bandote į ją rašyti, su sąlyga, kad rašomos reikšmės atitinka nurodytus tipus.

Todėl perduodant duomenis į Saugyklą reikia išsisaugoti būseną, ką jau esate atidavę, o kas dar nebuvo perduota. Kuriant naują objektą `insert` užklausos pagalba, Saugykla išduoda išorinį objekto raktą, kuris neturėtu niekada keistis. Norint atnaujinti tą patį objektą, reikėtų daryti ne `insert`, o `patch` veiksmą, siunčiant tik tas reikšmes, kurios pasikeitė.

Kartais nutinka taip, kad išsiuntus užklausą, negaunate atsakymo iš Saugyklos, dėl kokios nors klaidos, tokiais atvejais nėra galimybės sužinoti ar duomenys buvo įrašyti ar ne. Tokiu atveju turėtumėte daryti `search` arba `getone` užklausą, nurodant vidinį arba išorinį pirminį raktą, kad įsitikintumėte ar duomenys buvo įrašyti ar ne. Jei duomenys nebuvo įrašyti, tuomet rašymo užklausą turėtumėte pakartoti.

Iliustruojant visą tai psiaudokodu, rašymas į Saugyklą turėtu vykti taip:

```
state = State()
store = Store('put.data.gov.lt')

# Sinchronizuojam, ką reikia naujinti, ką trinti, o ką publikuoti pirmą
# kartą.
state.update()

# Kol ne visi objektas publikuoti, kartojam...
while not state.empty():

    # Skaitom objektus, kuriuos reikia publikuoti.
    for obj in state.objects:
        try:

            # Objektas buvo ištrintas šaltinyje, triname Saugykloje.
            if obj.deleted:
                store.delete(obj)

            # Objektas nebuvo publikuotas, publikuojam pirmą kartą.
            if obj.created:
                store.insert(obj)

            # Objektas jau buvo publikuotas ir pasikeitė nuo paskutinio
            # publikavimo. Atnaujinam.
```

(continues on next page)

(tęsinys iš praeito puslapio)

```
            if obj.changed:
                store.patch(obj)

        # Klaidos atveju:
        except HTTPError, IOError:
            # Jei objektas turėjo ir buvo ištrintas, triname iš eilės.
            # šaliname iš eilės.
            if obj.deleted and not store.exists(obj):
                state.remove(obj)

            # Jei objektas turėjo ir buvo publikuotas, triname iš eilės.
            if obj.created and store.exists(obj):
                state.remove(obj)

            # Visais kitais atvejais, paliekame objektą eilėje ir kartojame
            # publikavimą, sekančios iteracijos metu.

        # Jei klaidų nebuvo, šaliname objektą iš eilės.
        else:
            state.remove(obj)
```

Šiame pavyzdyje į `state` objektą įtraukiami visi šaltinio objektai, kuriuos reikia publikuoti, t.y. tuos, kurie dar nebuvo publikuoti, kurie pasikeitė nuo paskutinio publikavimo, arba buvo ištrinti šaltinyje.

Jei publikavimo metu įvyko klaida, kurios metu nebuvo gautas atsakymas iš Saugyklos, tikriname `store.exists(obj)` pagalba, ar realiai publikavimas įvyko ar ne. Jei ne, paliekame objektą eilėje ir sekančios iteracijos metu, bandome dar kartą.

Pats `store.exists(obj)` kreipiasi į Saugyklą ir bando gauti objektą, pagal jo vidinį arba išorinį raktą. Kas yra išoriniai ir vidiniai raktai, paaiškinta žemiau.

Vidinis pirminis raktas

Vidinis pirminis raktas yra vienas ar daugiau *property*, kurie yra įrašomi į struktūros aprašo *model.ref*, taip nurodant lentelės pirminį raktą.

Jei visi *property* nurodyti *model.ref* yra atviri, t.y. `access=open`, tuomet, galite užtikrintai žinoti ar konkretus duomenų objektas buvo įrašytas ar ne.

Išorinis pirminis raktas

Išorinis pirminis raktas `_id` yra generuojamas pačios Saugyklos, tačiau turint `spinta_set_meta_fields` teises, galite išorinį pirminį raktą, UUIDv4 formatu generuoti patys. Tokiu atveju, nebūtina publikuoti vidinio pirminio rakto ir nebūtina laukti atsakymo iš Saugyklos, nes išorinis raktas generuojamas jūsų pusėje ir klaidos atveju galite pasitikrinti ar objektas buvo įrašytas ar ne.

8.13.1 insert

```
http POST /datasets/gov/dc/geo/Continent $auth <<EOF
{
    "continent": "Europe"
}
EOF
```

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: /datasets/gov/dc/geo/Continent/abdd1245-bbf9-4085-9366-f11c0f737c1d

{
    "_type": "datasets/gov/dc/geo/Continent",
    "_id": "abdd1245-bbf9-4085-9366-f11c0f737c1d",
    "_revision": "16dabe62-61e9-4549-a6bd-07cecfbc3508",
    "_txn": "792a5029-63c9-4c07-995c-cbc063aaac2c",
    "continent": "Europe"
}
```

8.13.2 upsert

`upsert` veiksmas pirmiausiai patikrina ar jau yra sukurtas objektas atitinkantis `_where` sąlygą, jei yra, tada vykdo `patch` veiksmą, jei nėra, tada vykdo `update` veiksmą.

```
http POST /datasets/gov/dc/geo/Continent $auth <<EOF
{
    "_op": "upsert",
    "_where": "name='Africa'",
    "continent": "Africa"
}
EOF
```

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: /datasets/gov/dc/geo/Continent/b8f1edaa-220d-4e0b-b59b-dc27555a0fb5

{
    "_type": "datasets/gov/dc/geo/Continent",
    "_id": "b8f1edaa-220d-4e0b-b59b-dc27555a0fb5",
    "_revision": "988969c3-663b-4edf-bd64-861a3f1b1d1c",
    "_txn": "2c5feac6-1d72-48f6-ae63-8f2304693b21",
    "continent": "Africa"
}
```

8.13.3 update

`update` veiksmas pilnai perrašo objektą. Jei tam tikros objekto savybės nenurodomos, data tū savybių reikšmės pakeičiamos pagal nutylėjimą naudojamomis reikšmėmis.

Vykdant `update` taip pat būtina perduoti `_revision` revizijos numeri. Jei revizijos numeris nesutaps, su tuo, kas jau yra duomenų bazėje, tada duomenys nebus keičiami ir bus gražinta klaida. Tai reikalinga tam, kad būtų užtikrintas duomenų vientisumas.

```
http PUT /datasets/gov/dc/geo/Continent/b8f1edaa-220d-4e0b-b59b-dc27555a0fb5 $auth <<EOF
{
    "_revision": "988969c3-663b-4edf-bd64-861a3f1b1d1c",
    "continent": "Africa"
}
EOF
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Location: /datasets/gov/dc/geo/Continent/b8f1edaa-220d-4e0b-b59b-dc27555a0fb5

{
    "_type": "datasets/gov/dc/geo/Continent",
    "_id": "b8f1edaa-220d-4e0b-b59b-dc27555a0fb5",
    "_revision": "988969c3-663b-4edf-bd64-861a3f1b1d1c",
    "_txn": "2c5feac6-1d72-48f6-ae63-8f2304693b21",
}
```

Jei duomenys duomenų bazėje ir duomenys perduoti `update` užklausos metu yra identiški, tada duomenų bazėje niekas nekeičiama. Tačiau, jei duomenys skiriasi, tada į keitimų žurnalą išsaugoma tai, kas buvo pakeista ir sukuriam nauja revizija. Taip pat fiksuojama nauja transakcija.

`update` atsakyme grąžinamos tiks tos savybės, kurių reikšmės buvo pakeistos, Jei reikšmės nepasikeitė, tada jos pateikiamos atsakyme.

8.13.4 patch

`patch` veikia panašiai, kaip ir `update`, tačiau objekto pilnai neperrašo, keičia tik tas savybes, kurios nurodytos.

```
http PATCH /datasets/gov/dc/geo/Continent/b8f1edaa-220d-4e0b-b59b-dc27555a0fb5 $auth <
↪<EOF
{
    "_revision": "988969c3-663b-4edf-bd64-861a3f1b1d1c",
    "continent": "Africa"
}
EOF
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Location: /datasets/gov/dc/geo/Continent/b8f1edaa-220d-4e0b-b59b-dc27555a0fb5

{
    "_type": "datasets/gov/dc/geo/Continent",
    "_id": "b8f1edaa-220d-4e0b-b59b-dc27555a0fb5",
```

(continues on next page)

(tęsinys iš praeito puslapio)

```
    "_revision": "988969c3-663b-4edf-bd64-861a3f1b1d1c",
    "_txn": "2c5feac6-1d72-48f6-ae63-8f2304693b21",
}
```

8.13.5 delete

Trina objektą. Objektas pilnai nėra ištrinamas, jis vis dar lieka keitimų žurnale ir gali būti atstatytas.

```
http DELETE /datasets/gov/dc/geo/Continent/b8f1edaa-220d-4e0b-b59b-dc27555a0fb5 $auth
```

```
HTTP/1.1 204 No Content
```

8.13.6 wipe

Pilnai ištrina objektą, įskaitant ir objekto pėdsakus keitimo žurnale. Tokiu būdu ištrinto objekto atstatyti neįmanoma.

Pastaba: Naudoti tik išimtiniais atvejais.

`wipe` operacija naudojama tik išimtiniais atvejais, jei pastebėta esminių klaidų duomenų įkėlimo skriptuose, duomenys dėl įvairių klaidų buvo sugadinti ir pan. Duomenų įkėlimo procesą geriausia išsitestuoti *testinėje aplinkoje*.

Duomenų įkėlimo praktika, kai visi publikuoti duomenys ištrinami ir įkeliami iš naujo nerekomenduotina, kadangi tokiu atveju dingsta duomenų keitimo istorija, gali pasikeisti objektų identifikatoriai. Duomenys turi būti pirmą kartą įkeliame, o tada atnaujinama tik tai, kas pasikeitė.

```
http DELETE /datasets/gov/dc/geo/Continent/b8f1edaa-220d-4e0b-b59b-dc27555a0fb5/:wipe
↪$auth
```

```
HTTP/1.1 200 OK

{
    "wiped": true
}
```

8.14 Grupiniai rašymo veiksmai

Vienos HTTP užklausos metu galima vykdyti rašymo veiksmus grupei objektų. Tokiu būdu, veiksmai vykdomi vienoje duomenų bazės transakcijoje užtikrinant duomenų vientisumą.

Grupiniai veiksmai gali būti vykdomi dviem būdais, paprastuoju ir srautiniu.

8.14.1 Paprastasis grupinis rašymas

Paprastasis grupinis rašymas vykdomas per POST užklausą, kurioje yra pateiktas `_data` masyvas veiksmų.

Vykdant grupinius veiksmus būdina nurodyti `_op` veiksmą ir `_type` modelio pavadinimą, kuriam taikomas veiksmas.

Grupinis rašymas gali būti vykdomas konkrečiam vienam modeliui, arba keliems skirtingiems modeliams, vykdant užklausą vardų erdvės kontekste.

```
http POST /datasets/gov/dc/geo/Continent $auth <<EOF
{
    "_data": [
        {
            "_op": "insert",
            "_type": "datasets/gov/dc/geo/Continent",
            "continent": "Africa"
        },
        {
            "_op": "insert",
            "_type": "datasets/gov/dc/geo/Continent",
            "continent": "Europe"
        }
    ],
}
EOF
```

```
HTTP/1.1 207 Multi-Status
Content-Type: application/json

{
    "_data": [
        {
            "_type": "datasets/gov/dc/geo/Continent",
            "_id": "b8f1edaa-220d-4e0b-b59b-dc27555a0fb5",
            "_revision": "988969c3-663b-4edf-bd64-861a3f1b1d1c",
            "_txn": "2c5feac6-1d72-48f6-ae63-8f2304693b21",
            "continent": "Africa"
        },
        {
            "_type": "datasets/gov/dc/geo/Continent",
            "_id": "abdd1245-bbf9-4085-9366-f11c0f737c1d",
            "_revision": "16dabe62-61e9-4549-a6bd-07cecfbc3508",
            "_txn": "2c5feac6-1d72-48f6-ae63-8f2304693b21",
            "continent": "Europe"
        }
    ]
}
```

8.14.2 Srautinis grupinis rašymas

Paprastasis grupinis rašymas skirtas situacijoms, kai reikia atlikti veiksmus su nedidele grupe objektų. Tačiau jei objektų labai daug, galima vykdyti srautinį rašymą.

Srautinis rašymas priima duomenis JSONL formatu ir įeinančiame JSONL sraute skaito po vieną eilutę ir toje eilutėje pateiktą objektą iš karto vykdo. Tuo tarpu paprasto grupinio rašymo metu, visi objektai užkraunami į atmintį ir tik data įrašomi į duomenų bazę.

Kadangi srautinio grupinio rašymo metu objektai skaitomi ir rašomi vienas po kito, tai leidžia perduoti neribotą kiekį objektų rašymui.

Srautinio grupinio rašymo užklausa atrodo taip:

```
http POST /datasets/gov/dc/geo/Continent $auth Content-Type:application/x-ndjson <<EOF
{"_op": "insert", "_type": "datasets/gov/dc/geo/Continent", "continent": "Africa"}
{"_op": "insert", "_type": "datasets/gov/dc/geo/Continent", "continent": "Europe"}
EOF
```

```
HTTP/1.1 207 Multi-Status
Content-Type: application/json

{
    "_txn": "2c5feac6-1d72-48f6-ae63-8f2304693b21",
    "_status": {
        "insert": 2
    }
}
```

Srautinio rašymo užklausai būtina perduoti `Content-Type` antrašte su viena iš šių reikšmių:

```
application/x-ndjson
application/x-jsonlines
```

Tada bus vykdomas srautinis veiksmų vykdymas.

Srautinės užklausos atsakymas yra santrauka api tai, kiek kokių veiksmų buvo įvykdyta ir transakcijos numeris. Naudojant transakcijos numerį, atskiros užklausos metu, galima gauti visų pakeistų objektų identifikatorius `_id` ir revizijos numerius `_revision` ir informaciją apie tai, kas tiksliai buvo pakeista.

SKYRIUS 9

Spinta

Kad būtų paprasčiau atverti duomenis, rekomenduojame naudoti įrankį pavadinimu Spinta, kuris sukurtas duomenų atvėrimo automatizavimui. Spinta leidžia automatizuotai generuoti duomenų struktūros aprašus, juos patikrinti ar nėra klaidų, perduoti duomenis į *saugyklą* ir *publikuoti* atvertus duomenis aukščiausiu *brandos lygiu* ir laikantis geriausių atvirų duomenų publikavimo praktikų.

Jei naudodamiesi Spinta radote kokių nors klaidų ar turite kitų pastabų, galite pranešti apie klaidą, kad galėtume ją pataisyti.

9.1 Diegimas

9.1.1 Techniniai reikalavimai

Techniniai reikalavimai gali skirtis, priklausomai nuo to, kokiu tikslu naudojama Spinta priemonė. Jei naudojama tik duomenų atvėrimui, o ne publikavimui tuomet užtenka:

CPU 1 CPU, šiuo metu perduodant duomenis nėra naudojamas lygiagretinimas, todėl bus naudojamas tik vienas CPU, ateityje tai gali keistis.

RAM 512Mb, duomenys skaitomi srautiniu būdu, todėl nepriklausomai nuo šaltinio dydžio, naudojamas fiksuotas RAM kiekis.

Vienas Spinta procesas naudoja apie 100Mb RAM.

HDD Priklauso nuo duomenų kiekio.

Duomenų perdavimui iš šaltinio į saugyklą, saugomi papildomi duomenys:

1. Vidinių ir publikuojamų pirminių raktų sąsaja, saugoma `~/.local/share/spinta/keymap.db` Sqlite duomenų bazėje. Duomenys atrodo taip:

```
bb969358-ce9e-4255-b596-
↪c748f6885332|bf8b4530d8d246dd74ac53a13471bba17941dff7|BINDATA...
522a3615-8527-4eb7-8327-
↪977fe4383dcd|c4ea21bb365bbeeaf5f2c654883e56d11e43c44e|BINDATA...
```

(continues on next page)

(tęsinys iš praeito puslapio)

```
9be3e60b-d557-4596-a370-
↪660f3c337772|9842926af7ca0a8cca12604f945414f07b01e13d|BINDATA...
60c2f4da-c32a-4fba-a39a-
↪8e85252a77ad|a42c6cf1de3abfdea9b95f34687cbbe92b9a7383|BINDATA...
ab2baaa6-508d-4069-9a45-
↪53bce46676ca|8dc00598417d4eb788a77ac6ccef3cb484905d8b|BINDATA...
```

Saugomas išorinis raktas, vidinio rakto sha1 ir vidinio rakto reikšmė MsgPack formatu.

Šios lentelės dydis tiesiogiai proporcingas šaltinio įrašų skaičiui ir šaltinio lentelių pirminių raktų dydžiui.

Vidutiniškai, 10^6 įrašų telpa į 200Mb.

2. Perduodamų duomenų būsenos lentelė, kuri saugoma `~/.local/share/spinta/push` kataloge. Kiekvienai saugyklai į kurią perduodami duomenys sukuriama atskira būsenos lentelė, Sqlite formatu. Būsenos duomenys atrodo taip:

```
bb969358-ce9e-4255-b596-
↪c748f6885332|5d28bd0ccad38701a5fcc775259b91e53bf3b1b3|2022-02-10 13:26:39.
↪255110
522a3615-8527-4eb7-8327-
↪977fe4383dcd|0711b352478dda05732c4448e689aa9246986911|2022-02-10 13:26:39.
↪255602
9be3e60b-d557-4596-a370-
↪660f3c337772|ea85925a127b84a30d7be40daa150236954a39f6|2022-02-10 13:26:39.
↪255976
60c2f4da-c32a-4fba-a39a-
↪8e85252a77ad|7d5d3486ff456a2419476c4d7e6b2a73ae7bca22|2022-02-10 13:26:39.
↪256167
ab2baaa6-508d-4069-9a45-
↪53bce46676ca|fe5e2696e1768bc40ecce6b7418723d06e62a53a|2022-02-10 13:26:39.
↪256342
```

Saugomas išorinis pirminis raktas, visų perduodamų duomenų sha1 ir data, kad buvo perduoti duomenys paskutinį kartą.

Vidutiniškai, 10^6 įrašų talpa į 200Mb.

Apytiksliai, vienam milijonui įrašų reikėtų turėti bend 0.5G laisvos disko vietos.

Duomenų atvėrimo priemonė Spinta yra sukurta naudojant Python technologiją. Todėl prieš diegiant, jūsų naudojamoje aplinkoje turi būti įdiegta Python 3.9 arba naujesnė versija.

9.1.2 Debian/Ubuntu

Pirmiausia, prieš atliekant diegimą, reikėtų pasirinkti kokiame failų sistemos kataloge diegsite priemones. Rekomenduojame diegti į `/opt/spinta` katalogą.

Jei diegimą atliekate serveryje, tuomet verta sukurti atskirą naudotoją, kurio teisėmis bus leidžiamos priemonės, tai padaryti galite taip:

```
$ sudo useradd --system --create-home --home-dir /opt/spinta spinta
$ sudo -u spinta -s --set-home
$ cd
```

Toliau visus veiksmus atliksime `/opt/spinta` kataloge.

Pirmiausia reikėtų įsitikinti ar jūsų naudojama distribucijos versija turi reikalinga Python versiją, tai galite pažiūrėti taip:

```
$ python3 --version
```

Jei turite 3.9 ar naujesnę versiją, tuomet galite pereiti prie *Python paketų diegimas* žingsnio.

Naujesnę Python versiją galite įsidiegti pasirinkdami vieną iš dviejų galimų variantų:

- *Variantas 1*: naudojant pyenv, kuris atsisiūs Python išeities kodą ir sukompiliuos reikiamą Python versiją. Šis variantas yra universalesnis, tačiau sudėtingesnis ir reikalaujantis daugiau laiko.
- *Variantas 2* Naudojant PPA repozitoriumus, kurie veiks tik Ubuntu aplinkoje, tačiau reikiamą Python versiją gausite žymiai paprasčiau.

Naujesnės Python versijos diegimas naudojant pyenv

Pirmiausia mums reikia įdiegti pyenv ir visus šiai priemones ir Python kompiliavimui reikalingus paketus:

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install -y \
    git make build-essential libssl-dev zlib1g-dev \
    libbz2-dev libreadline-dev libsqlite3-dev wget \
    curl llvm libncurses5-dev libncursesw5-dev \
    xz-utils tk-dev libffi-dev liblzma-dev \
    python-openssl
$ curl https://pyenv.run | bash
```

Naujausios Python versijos diegimas naudojant pyenv daromas taip:

```
$ .pyenv/bin/pyenv install 3.10.7
```

Jei diegiate Spintą kitoje Linux distribucijoje, reikalingų paketų sąrašą galite rasti pyenv dokumentacijoje.

Atlikus naujos Python versijos diegimo veiksmus susikuriame izoliuotą aplinką, kurioje diegsime reikalingus Python paketus:

```
$ .pyenv/versions/3.10.7/bin/python -m venv venv
```

Naujesnės Python versijos diegimas naudojant PPA

Naujausios Python versijos diegimas naudojant PPA daromas taip:

Pirmiausiai mums reikia įdiegti PPA repozitoriumų valdymo priemones:

```
$ sudo apt update
$ sudo apt install software-properties-common
$ sudo add-apt-repository ppa:deadsnakes/ppa
```

Ir galiausiai įdiegiame pageidaujamą Python versiją:

```
$ sudo apt update
$ sudo apt install python3.10 python3.10-venv
```

Atlikus naujos Python versijos diegimo veiksmus susikuriame izoliuotą aplinką, kurioje diegsime reikalingus Python paketus:

```
$ python3.10 -m venv venv
```

Python paketų diegimas

Kai jau turite tinkamą Python versiją ir esate susikūrė izoliuotą Python aplinką, Spinta galima įdiegti taip:

```
$ venv/bin/pip install spinta
```

Galiausiai, įdiegus Spinta paketą, reikia aktyvuoti izoliuotą aplinką, kad galėtumėte toliau dirbti su Spinta paketo teikiama komanda `spinta`:

```
$ source venv/bin/activate
```

Tai padarius, galite patikrinti ar komanda `spinta` veikia:

```
$ spinta --version
0.1.9
```

Ši komanda turi išvesti, Spinta priemonės versijos numerį.

Pastaba: Atkreipkite dėmesį, kad `spinta` komanda yra pasiekiama tik tada, kai yra aktyvuota Python virtuali aplinka:

```
$ source venv/bin/activate
```

Python virtualios aplinkos aktyvavimas galioja tol, kol yra aktyvi terminalo sesija.

9.1.3 Windows

Tiesioginio Windows palaikymo nėra, tačiau Spinta galima įdiegti ir naudoti per Windows Subsystem for Linux (WSL). Informaciją apie tai, kaip įsidiegti WSL galite rasti Microsoft Windows dokumentacijoje.

Renkantis Linux distribuciją iš Microsoft Store rekomenduojame rinktis Ubuntu.

Įsidiegus ir pasileidus Ubuntu per WSL, toliau sekite *Debian/Ubuntu* instrukcijas.

9.1.4 Galimos problemos ir jų sprendimai

Jei įvykdžius sekančią komandą:

```
$ curl https://pyenv.run | bash
```

Gaunate tokią klaidą:

```
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 285 100 285 0 0 396 0 --:--:-- --:--:-- --:--:-- 395
curl: (60) SSL certificate problem: self signed certificate in certificate chain
More details here: https://curl.haxx.se/docs/sslcerts.html
```

(continues on next page)

(tęsinys iš praeito puslapio)

```
curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the web page mentioned above.
```

Tuomet įsitikinkite, kad jūsų ugniasienė neblokuoja prieigos prie išorinių resursų. Taip pat galite laikinai sustabdyti antivirusinė, kuri taip pat gali blokuoti tokio pobūdžio komandų vykdymą.

Kitas variantas, `curl` komandą galite vykdyti su `-k` argumentu.

Panaši situacija gali pasitaikyti ir vykdant:

```
.pyenv/bin/pyenv install $PYVER
```

Šios komandos vykdymo metu galite gauti tokią klaidą:

```
Downloading Python-3.9.5.tar.xz...
-> https://www.python.org/ftp/python/3.9.5/Python-3.9.5.tar.xz
error: failed to download Python-3.9.5.tar.xz

BUILD FAILED (Ubuntu 20.04 using python-build 2.0.0)
```

Tokių atveju įsitikinkite ar ugniasienė leidžia kreiptis į išore ir pabandykite laikinai sustabdyti antivirusinę programą.

9.2 Atnaujinimas

Norint atnaujinti Spinta paketą, reikia atlikti tokius žingsnius:

1. Komandų eilutėje pakeisti aktyvų kelią, kur yra įdiegta Spinta:

   ```
   cd /kelias/iki/spinta
   ```

2. Atnaujinti Spinta paketą:

   ```
   venv/bin/pip install --upgrade spinta
   ```

9.3 Konfigūravimas

Įdiegus Spinta jokia papildoma konfigūracija nereikalinga, kadangi visus reikalingus parametrus galima perduoti per komandinės eilutės argumentus, pavyzdžiui:

```
$ spinta inspect -r sql sqlite:///sqlite.db -o sdsa.xlsx
```

9.3.1 config.yml

Norint išvengti jautrių duomenų perdavimo per komandinę eilutę ir pageidaujant, kad duomenų bazės prisijungimo parametrai nebūtų įrašomi į struktūros aprašą, dalį parametrų galima iškelti į konfigūracijos failą:

Listing 1: config.yml

```
backends:
  mydb:
    type: sql
    dsn: sqlite://sqlite.db
```

Iškėlus duomenų bazės konfigūracijos parametrus į konfigūracinį failą, komandų eilutėje `-o config=config.yml` nurodoma configūracinio failo vieta, `-r sql mydb` nurodo pavadinimą iš `backends` sąrašo:

```
$ spinta -o config=config.yml inspect -r sql mydb -o sdsa.xlsx
```

9.4 ŠDSA generavimas

Spinta leidžia automatiškai generuoti *DSA* lentelę iš duomenų šaltinio.

Tarkime, jei turime SQLite duomenų bazę su viena lentele:

```
$ sqlite3 sqlite.db <<EOF
CREATE TABLE COUNTRY (
    NAME TEXT
);
EOF
```

Tada iš tokio duomenų šaltinio, *DSA* lentelę galima sugeneruoti taip:

```
$ spinta inspect -r sql sqlite:///sqlite.db
d | r | b | m | property | type   | ref | source
dataset                  |        |     |
  | sql                  | sql    |     | sqlite:///sqlite.db
                         |        |     |
  |   |   | Country      |        |     | COUNTRY
  |   |   |   | name     | string |     | NAME
```

Šiuo atveju, kadangi nenurodėme kur saugoti sugeneruotą *DSA* lentelę, ji buvo tiesiog išvesta į ekraną.

`-r` argumentui perduoti du argumentai `sql` ir `sqlite:///sqlite.db`, kurie atitinka *`resource.type`* ir *`resource.source`*.

Jei norima *DSA* lentelę išsaugoti į Excel lentelę, tada argumento `-o` pagalba galima nurodyti kelią iki failo, kuriame reikia išsaugoti *DSA* lentelę XLSX formatu:

```
$ spinta inspect -r sql sqlite:///sqlite.db -o sdsa.xlsx
```

DSA lentelę, išsaugotą XLSX formatu galima atsidaryti ir redaguoti naudojant LibreOffice Calc, Excel ar kitomis skaičiuoklės programomis. Tačiau taip pat lentelės turinį galima peržiūrėti ir Spintos pagalba:

```
$ spinta show manifest.csv
d | r | b | m | property | type   | ref | source
```

(continues on next page)

(tęsinys iš praeito puslapio)

```
dataset                  |        |     |
  | sql                  | sql    |     | sqlite:///sqlite.db
                         |        |     |
  |   |   | Country      |        |     | COUNTRY
  |   |   |   | name     | string |     | NAME
```

Jei turite daug duomenų šaltinių, galima juos visus surašyti į *DSA* lentelę, ir tada paleisti `inspect` komandą, kuri nuskaitys visus lentelėje esančius duomenų šaltinius ir kiekvienam iš jų sugeneruos duomenų struktūros aprašus.

Naują *DSA* lentelę galite pradėti kurti taip:

```
$ spinta init sdsa.xlsx
```

Ši komanda sugeneruos tuščią *DSA* lentelę:

```
$ spinta show sdsa.xlsx
d | r | b | m | property | type   | ref | source
```

Tada, šią lentelę galite atsidaryti su jūsų mėgiama skaičiuoklės programa ir užpildyti turimus duomenų šaltinius, pavyzdžiui, tokia užpildyta lentelė galėtų atrodyti taip:

```
$ spinta show resources.xlsx

d | r | b | m | property | type   | ref | source
dataset                  |        |     |
  | sql                  | sql    |     | sqlite:///sqlite.db
```

Struktūros generavimas daromas panašiai, kaip ir nurodant resursus `-r` argumentų pagalba, tik šį karta reikia nurodyti kelia iki *DSA* lentelės:

```
$ spinta inspect resources.xlsx -o sdsa.xlsx
$ spinta show sdsa.xlsx
d | r | b | m | property | type   | ref | source
dataset                  |        |     |
  | sql                  | sql    |     | sqlite:///sqlite.db
                         |        |     |
  |   |   | Country      |        |     | COUNTRY
  |   |   |   | name     | string |     | NAME
```

Analogiškai *DSA* lentelės generuojamos ir visiems kitiems *`resource.type`* formatams.

9.4.1 CSV

Pastaba: Kol kas Spinta neturi įmontuoto CSV formato palaikymo, todėl ši rekomendacija yra laikinas trūkstamo CSV palaikymo apėjimas. Ateityje planuojama integruoti Dask karkasą, kurio dėka atsiras CSV ir daugelio kitų formatų palaikymas.

Norint gauti pradinė ŠDSA variantą iš CSV failų, pirmiausiai CSV failus reikėtų importuoti į SQLite duomenų bazę:

```
$ sqlite3 data.db -csv ".import table1.csv table1"
$ sqlite3 data.db -csv ".import table2.csv table2"
$ sqlite3 data.db -csv ".import table3.csv table3"
```

Tokiu būdu importavus duomenis į SQLite, duomenų struktūros aprašas generuojamas taip:

```
$ spinta inspect -r sql sqlite:///data.db -o sdsa.xlsx
```

Jei pageidaujate, trūkstamus metaduomenis, tokius kaip duomenų laukus, pirminius raktus ar ryšius galite pateikti naudodami DB Browser for SQLite programą. Tačiau tą patį galite padaryti ir skaičiuoklės pagalba, redaguodami ŠDSA lentelę.

9.4.2 SQL

Jei norite struktūros aprašą nuskaityti iš vienos konkrečios duomenų bazės schemos, tada galite naudoti `-f` parametrą, kuris leidžia nurodyti papildomus parametrus:

```
$ spinta inspect -r sql sqlite:///sqlite.db -f "connect(schema: 'MYSCHEMA')" -o sdsa.xlsx
```

SQLite

Generuojant *DSA* iš SQLite duomenų bazės, jokių papildomų paketų diegti nereikia. `inspect` komanda atrodys taip:

```
$ spinta inspect -r sql sqlite:///data.db -o sdsa.xlsx
```

Atkreipkite dėmesį, kad absoliutus kelias atrodo taip:

```
sqlite:////data.db
```

O reliatyvus atrodo taip:

```
sqlite:///data.db
```

PostgreSQL

Generuojant *DSA* iš PostgreSQL duomenų bazės, jums papildomai reikia įdiegti tokį Python paketą:

```
$ pip install psycopg2-binary
```

O `inspect` komanda atrodys taip:

```
$ spinta inspect -r sql postgresql+psycopg2://user:pass@host:port/db -o sdsa.xlsx
```

MySQL

Generuojant *DSA* iš MySQL duomenų bazės, jums papildomai reikia įdiegti tokį Python paketą:

```
$ pip install pymysql
```

O `inspect` komanda atrodys taip:

```
$ spinta inspect -r sql mysql+pymysql://user:pass@host:port/db -o sdsa.xlsx
```

MySQL (<5.6)

`pymysql` biblioteka palaiko MySQL >= 5.6 ir MariaDB >= 10 versijas. Jei naudojate labai seną MySQL versiją, tuomet, vietoj `pymysql` reikėtų naudoti senesnę mysqlclient biblioteką, kuri palaiko MySQL >= 3.23.32. `mysqlclient` diegimui pirmiausia reikės įsidiegti tokius sisteminius paketus:

```
$ sudo apt install build-essential python3-dev default-libmysqlclient-dev
```

O data ir pačią `mysqlclient` biblioteką:

```
pip install mysqlclient
```

`inspect` komanda atrodys taip:

```
spinta inspect -r sql mysql+mysqldb://user:pass@host:port/db -o sdsa.xlsx
```

p.s. jei vis dar naudojate tokią seną MySQL versiją, laikas atsinaujinti!

Microsoft SQL Server

Generuojant *DSA* iš Microsoft SQL Server duomenų bazės, jums papildomai reikia įdiegti FreeTDS paketą:

```
$ sudo apt install freetds-bin
```

Ir pymssql Python paketą:

```
$ pip install pymssql
```

Toliau reikia sukonfigūruoti FreeTDS, rekomenduojame naudoti tokį konfigūracijos failą:

```
[global]
tds version = 7.4
port = 1433
client charset = utf-8
```

`inspect` komanda atrodys taip:

```
$ spinta inspect -r sql mssql+pymssql://user:pass@host:port/db -o sdsa.xlsx
```

Oracle

Generuojant *DSA* iš Oracle duomenų bazės, jums papildomai reikia įdiegti cx_Oracle paketą:

```
$ pip install cx_Oracle
```

Dėl papildomos informacijos apie Oracle jungtį, skaitykite cx_Oracle dokumentacijoje.

`inspect` komanda atrodys taip:

```
$ spinta inspect -r sql oracle+cx_oracle://user:pass@host:port/db -o sdsa.xlsx
```

9.5 ŠDSA atnaujinimas

Po to, kai yra sugeneruojamas pradinis ŠDSA ir papildomas trūkstamais duomenimis, dažniausiai po tam tikro laiko, šaltinio duomenų struktūra keičiasi ir karts nuo karto reikia atnaujinti esamą ŠDSA ir šaltinio, įtraukiant naujausius pakeitimus šaltinyje.

Tai galima padaryti tokiu būdu:

```
$ spinta inspect sdsa.xlsx -o sdsa_updated.xlsx
```

`sdsa_updated.xlsx` faile išliks visi metaduomenys, kuris buvo pradiniame `sdsa .xlsx`, papildant juos naujais metaduomenimis iš šaltinio.

9.6 ŠDSA testavimas prieš publikuojant

Kai jau yra parengtas ŠDSA variantas iš kurio galima atverti duomenis pirmiausia reikia patikrinti are ŠDSA yra be klaidų. Tai galima padaryti šios komandos pagalba:

```
$ spinta check sdsa.xlsx
```

Jei klaidų nėra, tuomet papildomai, galima paleisti duomenų publikavimo priemonę, kuri duomenis publikuos tiesiai iš pirminio duomenų šaltinio. Duomenų publikavimas iš pirminio šaltinio turi tam tikrų apribojimų, tačiau ši galimybė leidžia peržiūrėti, kaip atrodys publikuojami duomenys, prie juos publikuojan viešai.

Kaip atrodys publikuojami duomenys, galite peržiūrėti taip:

```
$ spinta run --mode external sdsa.xlsx
```

Ši komanda paleis HTTP serverį 127.0.0.1:8000 adresu, atsidarę šį adresą naršyklėje galėsite peržiūrėti kaip atrodo duomenys.

9.7 ŠDSA vertimas į ADSA

ŠDSA yra toks duomenų struktūros aprašas, kuris yra susietas su duomenų šaltiniu, yra užpildytas *source* stulpelis.

Verčiant ŠDSA į ADSA, iš esmės pašalinami *source* ir *prepare* stulpelių duomenys, o taip pat pašalinamos visos eilutės, kurių *access* yra mažesnis, nei open.

ŠDSA vertimą į ADSA galima daryti automatiškai taip:

```
$ spinta copy sdsa.xlsx --no-source --access open -o adsa.csv
```

9.8 Duomenų publikavimas į Saugyklą

Prieš publikuojant duomenis į *Saugyklą*, Saugykloje turi būti įkeltas *duomenų struktūros aprašas*. Saugykla gali priimti tik duomenis, turinčius *DSA*.

Taip pat, prieš publikuojant duomenis, Saugykloje turi būti užregistruotas klientas, kuriam suteikiamos rašymo į saugyklą teisės. Klientui suteikiamos rašymo teisės į tam tikrą vardų erdvę, todėl skirtingi klientai, gali rašyti duomenis tik į tam tikrą, jiems skirtą vardų erdvę.

Kliento autorizacijos duomenys turėtu būti pateikiami `credentials.cfg` faile. `credentials.cfg` failo ieškoma $XDG_CONFIG_HOME/spinta kataloge (pavyzdžiui `~/.config/spinta/credentials.cfg`). Šio failo formatas atrodo taip:

```
[myclient@put.data.gov.lt]
client = myclient
secret = verysecret
scopes =
  spinta_getall
  spinta_getone
  spinta_search
  spinta_changes
  spinta_datasets_gov_myorg_insert
  spinta_datasets_gov_myorg_upsert
  spinta_datasets_gov_myorg_update
  spinta_datasets_gov_myorg_patch
  spinta_datasets_gov_myorg_delete
```

Čia nurodomas kliento pavadinimas, slaptažodis ir leidimai (`scopes`). Suteiktas leidimas skaityti visus duomenis ir rašyti tik į `datasets/gov/myorg` vardų erdvę.

Kol kas kliento kūrimas Saugykloje yra daromas rankiniu būdu, atskiru paklausimu, tačiau planuojama tai automatizuoti.

Galiausiai, įkėlus duomenų struktūros aprašą į Katalogą, iš Katalogo įkėlus aprašą į saugyklą ir turinti klientą Saugykloje, galima publikuoti duomenis į saugyklą tokiu būdu:

```
$ spinta push sdsa.csv -o myclient@put.data.gov.lt
```

Vietoj `sdsa.csv` galima naudoti ir `sdsa.xlsx`, abu formatai, tiek CSV, tiek XLSX yra palaikomi.

Dar vienas dalykas, į kurį reikėtu atkreipti dėmesį yra būsenos ir objektų identifikatorių failai. Kadangi `spinta push` komanda į Saugyklą siunčia tik tuos duomenis kurie dar nebuvo siųsti arba kurie pasikeitė, kad tai veiktų saugoma duomenų perdavimo į Saugyklą būsena ir identifikatoriai. Būsena saugoma SQLite duomenų bazėje, $XDG_DATA_HOME/spinta/push/{remote}.db faile (pavyzdžiui `~/.local/share/spinta/push/get_data_gov_lt.db`). Identifikatoriai saugomi `$XDG_DATA_HOME/spinta/keymap.db` SQLite faile (pavyzdžiui `~/.local/share/spinta/keymap.db`. Priklausomai nuo duomenų kiekio šie failai gali užimti gan daug vietos. Būsenos ir identifikatorių failuose saugomi Saugykloje suteikti objektų identifikatoriai, vietiniai identifikatoriai ir duomenų kontrolinės sumos.

Kadangi `spinta push` komanda saugo būseną, šią komandą galima leisti daug kartų ir ji tęs duomenų perdavimą nuo tos vietose kur buvo baigta paskutinį kartą.

Rekomenduojama šią duomenų publikavimo komanda įtraukti į automatiškai vykdomų užduočių sąrašą, kad duomenys būtų publikuojamai automatiškai, pavyzdžiui kas naktį arba kas valandą.

Reikėtu atkreipti dėmesį į tai, kad vienu metu reikėtu leisti tik vieną `spinta push` komandos procesą.

`spinta push` komanda, prieš siunčiant duomenis, pirmiausiai suskaičiuoja kiek yra objektų, kurie bus siunčiami, kad galėtų atvaizduoti progreso juostą. Jei jūsų šaltinis yra lėtas galite naudoti `--no-progress-bar`, kad neskaičiuotų objektų, pavyzdžiui:

```
$ spinta push sdsa.csv -o myclient@put.data.gov.lt --no-progress-bar
```

Kitus galimus komandinės eilutės argumentus galite sužinoti taip:

```
$ spinta push --help
```

SKYRIUS 10

Duomenų struktūros aprašas

Čia rasite pilna duomenų struktūros aprašo (*DSA*) lentelės specifikaciją.

Duomenų struktūros aprašas yra lentelė, kurią galima redaguoti skaičiuoklės pagalba. Lentelėje pateikiama informacija apie vieno ar kelių duomenų šaltinių struktūrą.

Duomenų struktūros aprašas skirtas tam, kad būtų galima automatizuoti duomenų atvėrimo veiklas. Įstaigoms, atveriančioms duomenis, daugeliu atveju turėtu užtekti paruošti tik *DSA* lentelę. Turint parengtą *DSA* lentelę, visos kitos veiklos yra pilnai automatizuojamos.

Vieninga aprašų struktūros parengimo metodika sudaro galimybę automatizuoti duomenų atvėrimo ir publikavimo procesus bei užtikrinti atveriamų duomenų kokybę, vientisumą ir viso proceso stebėseną.

Duomenų struktūros aprašas gali būti rengiamas jau atvertiems duomenims (*ADSA*) ir dar neatvertiems duomenims (*ŠDSA*).

10.1 Lentelės formatas

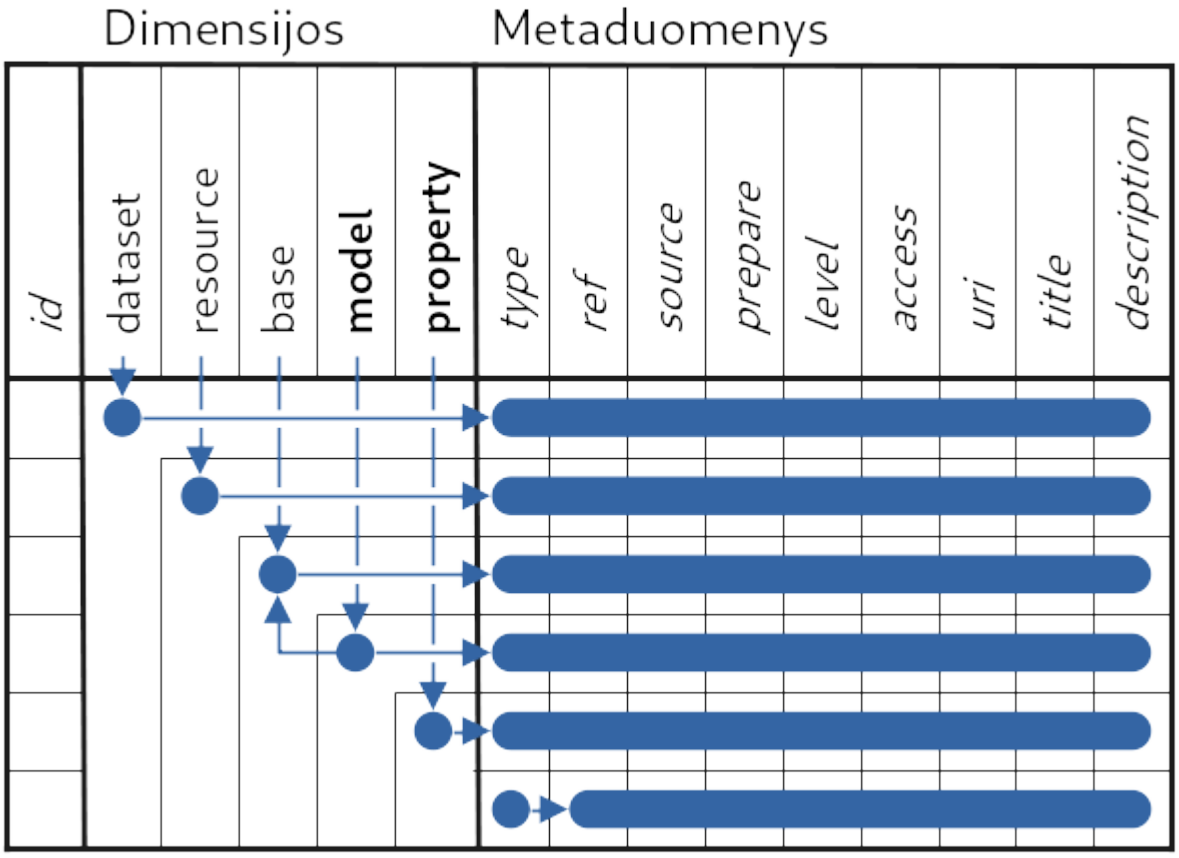
DSA yra sudarytas taip, kad būtų patogu dirbti tiek žmonėms, tiek programoms. Žmonės su *DSA* lentele gali dirbti naudojantis, bet kuria skaičiuoklės programa ar kitas pasirinktas priemones. Kadangi *DSA* turi aiškią ir griežtą struktūrą, lentelėje pateiktus duomenis taip pat gali lengvai nuskaityti ir interpretuoti kompiuterinės programos.

Tais atvejais, kai su *DSA* lentele dirba žmonės, lentelė gali būti saugoma įstaigos pasirinktos skaičiuoklės programos ar kitų priemonių formatu.

Automatizuotoms priemonėms *DSA* turi būti teikiamas CSV formatu laikantis **RFC 4180** taisyklių, failo koduotė turi būti UTF-8.

10.2 Lentelės struktūra

Rengiant duomenų struktūros aprašus darbas vyksta su viena lentele. Lentelė sudaryta iš 15 stulpelių. Iš 15 stulpelių, pirmasis stulpelis `id` yra eilutės identifikatorius, kuris pildomas automatiškai, toliau seka 5 *dimensijų* stulpeliai ir likę 9 stulpeliai yra metaduomenys apie duomenis.



Lentelė sudaryta hierarchiniu principu. Kiekvienas metaduomenų stulpelis gali turėti skirtingą prasmę, priklausomai nuo dimensijos. Todėl toliau dokumentacijoje konkrečios dimensijos stulpelis yra žymimas nurodant tiek dimensijos, tiek metaduomenis pavadinimus, pavyzdžiui `model.ref`, kuris nurodo `ref` stulpelį, esantį `model` dimensijoje.

Ką reiškia kiekvienas stulpelis paaiškinta žemiau.

`id`

Eilutės identifikatorius

Unikalus elemento numeris, gali būti sveikas, monotoniškai didėjantis skaičius arba UUID. Svarbu užtikrinti, kad visi elementai turėtu unikalų id.

Šis stulpelis pildomas automatinėmis priemonėmis, siekiant identifikuoti konkrečias metaduomenų eilutes, kad būtų galima atpažinti metaduomenis, kurie jau buvo pateikti ir po to atnaujinti.

Šio stulpelio pildyti nereikia.

10.2.1 Dimensijos

Duomenų struktūros aprašo lentelė sudaryta hierarchiniu principu. Kiekvienos lentelės eilutės prasmę apibrėžia viena iš penkių *dimensijų*. Kiekvienoje eilutėje gali būti užpildytas tik vienas dimensijos stulpelis.

Be šių penkių dimensijų, yra kelios *papildomos dimensijos*, jos nurodomos `type` stulpelyje, neužpildžius nei vieno dimensijos stulpelio.

`dataset`
: **Duomenų rinkinys**

  Kodinis duomenų rinkinio pavadinimas. Atitinka dcat:Dataset prasmę. Žiūrėti *Duomenų rinkinys*.

`resource`
: **Duomenų šaltinis**

  Kodinis duomenų šaltinio pavadinimas. Atitinka dcat:Distribution prasmę. Žiūrėti *Duomenų šaltinis*.

`base`
: **Modelio bazė**

  Kodinis bazinio modelio pavadinimas. Atitinka rdfs:subClassOf prasmę (`model` `rdfs:subClassOf` `base`). Žiūrėti *Modelio bazė*.

`model`
: **Modelis (lentelė)**

  Kodinis modelio pavadinimas. Atitinka rdfs:Class prasmę. Žiūrėti *Duomenų modelis*.

`property`
: **Savybė (stulpelis)**

  Kodinis savybės pavadinimas. Atitinka rdfs:Property prasmę. Žiūrėti *Savybė*.

10.2.2 Metaduomenys

Kaip ir minėta aukščiau, kiekvienos metaduomenų eilutės prasmė priklauso nuo *Dimensijos*. Todėl, toliau dokumentacijoje, kalbant apie tam tikros dimensijos stulpelį, stulpelis bus įvardinamas pridedant dimensijos pavadinimą, pavyzdžiui `model.ref`, kas reikštų, kad kalbama apie `ref` stulpelį, `model` dimensijoje.

`type`
: **Tipas**

  Prasmė priklauso nuo dimensijos. Žiūrėti *Duomenų tipai*.

  Jei nenurodytas nei vienas *dimensijos stulpelis*, tuomet šiame stulpelyje nurodoma *papildoma dimensija*.

`ref`
: **Ryšys**

  Prasmė priklauso nuo dimensijos. Žiūrėti *Ryšiai tarp modelių*, *Matavimo vienetai* ir *Klasifikatoriai*.

`source`
: **Šaltinis**

  Duomenų šaltinio struktūros elementai. Žiūrėti *Duomenų šaltiniai*.

`prepare`
: **Formulė**

  Formulė skirta duomenų atrankai, nuasmeninimui, transformavimui, tikrinimui ir pan. Žiūrėti *Formulės*.

`level`
: **Brandos lygis**

  Duomenų brandos lygis, atitinka 5 Star Data. Žiūrėti *Brandos lygiai*.

`access`
: **Prieiga**

  Duomenų prieigos lygis. Žiūrėti *Prieigos lygiai*.

`uri`
: **Žodyno atitikmuo**

  Sąsaja su išoriniu žodynu. Žiūrėti *Išoriniai žodynai*.

`title`
: **Pavadinimas**

  Elemento pavadinimas.

`description`
: **Aprašymas**

  Elemento aprašymas. Galima naudoti Markdown sintaksę.

Visi stulpeliai lentelėje yra neprivalomi. Stulpelių tvarka taip pat nėra svarbi. Pavyzdžiui jei reikia apsirašyti tik globalių modelių struktūrą, nebūtina įtraukti `dataset`, `resource` ir `base` stulpelių. Jei norima apsirašyti tik prefiksus naudojamus `uri` lauke, užtenka turėti tik prefiksų aprašymui reikalingus stulpelius.

Įrankiai skaitantys *DSA*, stulpelius, kurių nėra lentelėje turi interpretuoti kaip tuščius. Taip pat įrankiai neturėtų tikėtis, kad stulpeliai bus išdėstyti būtent tokia tvarka. Nors įrankių atžvilgiu stulpelių tvarka nėra svarbi, tačiau rekomenduotina išlaikyti vienodą stulpelių tvarką, tam kad lenteles būtų lengviau skaityti.

10.3 Vardų erdvės

`dataset` ir `model` esantys pavadinimai turi būti globaliai (Lietuvos mastu) unikalūs. Kad užtikrinti pavadinimų unikalumą `dataset` ir `model` pavadinimai formuojami pasitelkiant vardų erdves.

`/<standard>/`
: **Standartų vardų erdvė**

  Standartų vardų erdvė formuojama egzistuojančių standartų ir išorinių žodynų pagrindu suteikiant vardų erdvei <standard> standarto sutrumpintą pavadinimą. Pavyzdžiui atvirų duomenų katalogo metaduomenys turėtų keliauti į `/dcat/` vardų erdvę. Standartų sutrumpintus pavadinimus rekomenduojame imti iš Linked Open Vocabularies katalogo.

`/datasets/<type>/<org>/`
: **Įstaigų vardų erdvė**

  Konkrečios organizacijos vietinė rinkinio vardų erdvė. Rekomenduojama <org> reikšmei naudoti organizacijos trumpinį, kad bendras modelio pavadinimas nebūtų per daug ilgas.

  Galimos <type> reikšmės:

  `gov`
  : Valstybinės įstaigos.

  `com`
  : Verslo įmonės.

`/datasets/<type>/<org>/<dataset>/`
Įstaigų duomenų rinkinių vardų erdvė

Įstaigos duomenų rinkinio vardų erdvė į kurią patenka visi įstaigos duomenų modeliai.

Viskas, kas eina po `/datasets/<type>/<org>/` yra įstaigos vardu erdvė, kurioje įstaiga, gali įsitraukti papildomas vardų erdves, pavyzdžiui `/datasets/<type>/<org>/<isr>/<dataset>`, kuri `<isr>` yra Informacinės sistemos ar Registro trumpinys.

`/provisional/`
Duomenų rinkiniai turintys negalutinę struktūrą

Šioje vardų erdvėje talpinamos visos kitos vardų erdvės, kurių duomenų struktūra nėra galutinė ir gali keistis, be atskiro įspėjimo.

Visos duomenų rinkinius rekomenduojame pirmiausiai kelti į šią duomenų erdvė ir įsitikimus, kad duomenų struktūra yra stabili, perkelti į kitą atitinkamą vardų erdvė.

Naujai atveriami *duomenų struktūros aprašai* sudaromi *ŠDSA* pagrindu. Įprastai duomenų bazių struktūra nėra kuriama vadovaujantis standartais. Vidinės struktūros dažniausiai kuriamos vadovaujantis sistemai keliamais reikalavimais. Todėl naujai atveriamų duomenų rinkiniai turi keliauti į duomenų rinkinio vardų erdvę `/datasets/<type>/<org>/<dataset>/`, išlaikant pirminę duomenų struktūrą ir neprarandant duomenų.

Tačiau su laiku, dalis įstaigos duomenų iš duomenų rinkinio vardų erdvės turėtu būti perkeliami į globalią duomenų erdvę. Į globalią duomenų erdvę pirmiausiai turėtų būti perkeliami tie duomenys, kurie yra plačiai naudojami. Perkėlimas į globalią duomenų erdvę nepanaikina duomenų rinkinio iš ankstesnės vardų erdvės, tiesiog duomenų rinkinio duomenų pagrindu kuriama kopija į globalią duomenų erdvę.

10.4 Reliatyvūs pavadinimai

Modelio pavadinimas gali būti absoliutus arba reliatyvus. Absoliutūs pavadinimai prasideda / simboliu, reliatyvus pavadinimai prasideda be / simbolio ir yra jungiami su vardų erdvės pavadinimu, kurios kontekste yra apibrėžtas modelis.

Pavyzdžiui, turinti tokį duomenų struktūros aprašą:

id	d	r	b	m	property	type
1	**dcat**					ns
2				**dataset**		
3					title	
4	**datasets/gov/ivpk/adk**					
5		adk				
6			**/dcat/dataset**			alias
7				**dataset**		
8					title	

Matome, kad yra apibrėžti du modeliai:

- `dcat/dataset`
- `datasets/gov/ivpk/adk/dataset`

Vienas `dataset` modelis yra apibrėžtas `dcat` vardų erdvės kontekste, kitas `datasets/gov/ivpk/adk` vardų erdvės kontekste.

Kai modelio pavadinimas yra naudojamas vardų erdvės kontekste ir pavadinimas neprasideda / simboliu, tada tai yra reliatyvus modelio pavadinimas. Reliatyvus modelio pavadinimas yra jungiamas su vardų erdvės pavadinimu, kurios kontekste yra apibrėžtas modelis.

Jei tam tikros vardų erdvės kontekste norime įvardinti modelį, kuris yra už tos vardų erdvės konteksto ribų, būtina naudoti absoliutų modelio pavadinimą, kuris prasideda / simboliu. Taip yra padaryta 6-oje eilutėje, kur nurodyta, kad `datasets/gov/ivpk/adk/dataset` bazė yra `dcat/dataset` modelis iš kitos vardų erdvės.

Visais atvejais, kai modelio pavadinimas naudojamas nenurodant jokio vardų erdvės konteksto, / simbolio pavadinimo pradžioje naudoti nereikia. Pavyzdžiui šiame tekste įvardinti `dcat/dataset` ir `datasets/gov/ivpk/adk/dataset` modelių pavadinimai neprasideda / simboliu.

10.5 Dimensijos

Dimensijos apibrėžia duomenų metaduomenų detalumo lygį. Stulpeliai *dataset*, *resource*, *base*, *model* ir *property* yra naudojami kaip *DSA* dimensijos. *dataset* yra aukščiausia dimensija, *property* žemiausia. *dataset* ir *resource* dimensijos atitinka DCAT žodyną ir užtikrina trečia duomenų brandos lygį, o žemiau esantys *base*, *model* ir *property* atitinka RDFS žodyną ir užtikrina penktą duomenų brandos lygį. Vienoje lentelės eilutėje gali būti užpildytas ne daugiau kaip vienas dimensijos stulpelis. Užpildytasis dimensijos stulpelis nustato visų kitų stulpelių prasmę.

id	d	r	b	m	property	title
1	datasets/gov/ivpk/adk					Atvirų duomenų katalogas
2		adk				Atvirų duomenų katalogo duomenų bazė
3			/dcat/dataset			Duomenų rinkinys
4				dataset		Duomenų rinkinys
5					title	Duomenų rinkinio pavadinimas

Pavyzdyje aukščiau, taupant vietą, dalies dimensijų pavadinimai sutrumpinti iki vienos raidės ir įtraukti ne visi stulpeliai, o tik *id* ir *title* metaduomenų stulpeliai. Pavyzdyje matome, kad vienoje eilutėje užpildytas tik vienas dimensijos stulpelis, o *title* stulpelio prasmė keičiasi priklausomai nuo dimensijos reikšmės. Toliau specifikacijoje konkrečios dimensijos stulpeliai įvardijami pateikiant tiek dimensijos, tiek metaduomens stulpelio pavadinimus, kad būtų aiškiau apie kurios dimensijos metaduomenį kalbama, pavyzdžiui *model.title* leidžia suprasti kad kalbama apie „Duomenų rinkinys“ reikšmę 4-oje eilutėje.

Be minėtų dimensijų stulpelių *DSA* lentelėje gali būti naudojami papildomos metaduomenų dimensijos, kai nurodoma *type* reikšmė ir nepateikiama nei viena dimensijos stulpelio reikšmė. Pavyzdžiui:

id	d	r	b	m	property	type	ref	uri
1						prefix	dcat	http://www.w3.org/ns/dcat#

Šiuo atveju *prefix* tampa dar viena dimensija, leidžianti pateikti metaduomenis apie naudojamų URI prefiksus. Analogiškai, kaip ir su kitomis dimensijomis, dimensijos ir metaduomens pavadinimus galima apjungti, pavyzdžiui *prefix.ref* apibūdina tik *prefix* dimensijai priklausančius *ref* stulpelius.

Dimensijos leidžia suskirstyti metaduomenis į hierarchinę struktūrą. Todėl *DSA* lentelės eilučių eiliškumas yra svarbus, kadangi žemiau esančios eilutės priklauso aukščiau esančiai dimensijai. Tas pats galioja ir pagalbinėms *dimensijoms*.

Nors lentelėje sudaro tik 15 stulpelių, tačiau pasitelkiant 5 pagrindinius dimensijas ir keletą papildomų dimensijų, atsiranda galimybė išsamiai aprašyti visą duomenų šaltinio struktūrą.

10.5.1 Duomenų rinkinys

DSA lentelėje *duomenų rinkinys* nurodomas tam, kad būtų išlaikomas ryšys tarp *DSA* ir *ADK*. Atliekant duomenų inventorizaciją, automatiškai generuota *DSA* lentelė turi būti suskirstoma į *duomenų rinkinius*. Tada priemonių pagalba automatiškai sukuriami pirminiai *ADK* metaduomenys apie *duomenų rinkinius*, kuriuos vėliau reikia papildyti rankiniu būdu prisijungus prie ADK. Automatizuota priemonė sukūrus duomenų rinkinių įrašus *ADK*, papildys *DSA* lentelę, į `dataset.ref` įrašant *ADK* sukurto duomenų rinkinio identifikatorių. Tokiu būdu, sekantį kartą vykdant sinchronizaciją, jei `dataset.ref` yra užpildytas, bus atnaujinami jau sukurti *ADK duomenų rinkinių* įrašai.

Į *ADK* turi būti publikuojami tik tie duomenų rinkiniai iš DSA, kurių `dataset.access` reikšmė yra `public` arba `open`.

`dataset.`**`source`**
: Jei nenurodyta, naudoti https://data.gov.lt/ adresą.

  Nenaudojama, jei `dataset.type` yra `ns`.

`dataset.`**`prepare`**
: Nenaudojama.

`dataset.`**`type`**
: Jei nenurodyta, naudoti `ivpk` reikšmę. type nurodo *API* formatą, kuriuo automatiškai pildomi duomenų rinkinių metaduomenys atvirų duomenų portale.

  Galimos reikšmės:

  `ns`
  : Atitinka vardų erdvę, kurioje pateikiami duomenų rinkiniai. Naudojamas tais atvejais, kai norima pateiki papildomus metaduomenis vardų erdvei, pavydžui pavadinimą ar parašymą.

  `ckan`
  : Atitinka duomenų rinkinį iš CKAN duomenų katalogo.

  `ivpk`
  : Atitinka duomenų rinkinį iš data.gov.lt duomenų katalogo.

`dataset.`**`ref`**
: *Duomenų rinkinio* duomenų kataloge identifikatorius.

  Nenaudojamas jei `dataset.type` yra `ns`.

`dataset.`**`level`**
: Nenaudojamas.

`dataset.`**`access`**
: Viso duomenų rinkinio ar vardų erdvės *Prieigos lygiai*. Paveldimas.

`dataset.`**`title`**
: Duomenų rinkinio ar vardų erdvės pavadinimas.

`dataset.`**`description`**
: Duomenų rinkinio ar vardų erdvės aprašymas.

Skaidymas į *duomenų rinkinius* turi būti atliekamas tokiu principu, kad visi tarpusavyje susiję *modeliai* patektų į vieną *duomenų rinkinį*. Teoriškai, absoliučiai visi *modeliai* gali būti susiję tarpusavyje, skaidymą reikėtų daryti pagal tematinį *modelių* tarpusavio ryšį, o ne pagal reliacinius ryšius.

Jei duomenys yra išskaidyti pagal laiką, vietove ar kitus kriterijus į skirtingus duomenų šaltinius, tokie duomenys turėtų būti apjungti į vieną modelį *Modelio bazė* pagalba ir turėtų priklausyti vienam *duomenų rinkiniui*. Tą pačią semantinę prasmę turintys duomenys neturėtų būti išskaidyti keliuose *duomenų rinkiniuose*.

Pavyzdys:

d	r	b	m	property	type	
datasets/gov/ivpk					ns	Informacinės visuomenės plėtros komitetas
datasets/gov/ivpk/adp					ns	Lietuvos atvirų duomenų portalas
datasets/gov/ivpk/adp/catalog						Lietuvos atvirų duomenų katalogas
datasets/gov/ivpk/adp/store						Lietuvos atvirų duomenų saugykla

Šiame pavyzdyje apibrėžtos dvi vardų erdvės ir du rinkiniai.

Kiekviena organizacija turėtu deklaruoti tik savo vardų erdvės metaduomenis. Globalios vardų erdvės, tokios kaip `datasets` ir `datasets/gov` yra administruojamos vyriausiojo duomenų atvėrimo koordinatoriaus.

10.5.2 Duomenų šaltinis

ŠDSA atveju *duomenų šaltinis* bus vidinis duomenų bazių serveris, kažkoks vidinis katalogas kuriame yra lentelių failai ar koks nors vidinis API.

ADSA atveju, *duomenų šaltinis* gali būti nenurodytas, tai reiškia, kad duomenų rinkinio duomenys dar nėra publikuoti. Jei duomenys jau yra publikuoti, tada *ADSA duomenų šaltinis* turi rodyti į publikuotus atvertus duomenis, tai gali būti nuorodos į CSV failus, į viešą JSON API ir pan.

Duomenų šaltinio įrašas taip pat naudojamas tam, kad automatiškai atnaujinti *ADK* esančius *duomenų rinkinius*, patelkiant konkrečias nuorodas į konkrečius duomenų failus. Analogiškai kaip ir *Duomenų rinkinys* atveju, *`resource.ref`* stulpelyje nurodomas duomenų šaltinio identifikatorius iš *ADK*.

`resource.`**`type`**
: Duomenų šaltinio tipas. Galimos reikšmės:

  `sql`
  : Reliacinės duomenų bazės

  `csv`
  : CSV lentelės

  `tsv`
  : TSV lentelės

  `json`
  : JSON resursai

  `jsonl`
  : JSON lines resursai

  `geojson`
  : GeoJSON failai

  `xml`
  : XML resursai

  `html`
  : HTML puslapiai

  `xlsx`
  : Excel lentelės (naujasis OOXML formatas)

  `xls`
  : Excel lentelės (senasis formatas)

  `ods`
  : ODT skaičiuoklės formatas

zip
ZIP failų archyvas.

`resource.`**source**
Priklauso nuo *resource.source*. Žiūrėti *Duomenų šaltiniai*.

`resource.`**ref**
Resurso kodinis pavadinimas, kuris yra apibrėžtas atskirai duomenų struktūros apraše, ar konfigūracijos failuose.

Kai vienas resursas nurodo kitą, tuomet aprašomas resursas išplečia kitą resursą į kurį rodo arba naudoja kitą resursą, kaip konteinerį, kuriame saugomi duomenys.

Išplėtimo atveju, galima pateikti tam tikro resurso konfidencialius duomenis, tokius, kaip slaptažodis, atskirai nuo duomenų struktūros aprašo, konfigūracijos failuose, o duomenų struktūros apraše išplėsti resursą, pateikiant ne konfidencialius duomenis.

Tokiais atvejais, kaip duomenų failai saugomi ZIP archyvuose arba FTP serveryje, galima atskirai aprašyti ZIP ar FTP resursą, o po to, aprašant konkretų duomenų failo resursą, pateikti nuorodą, kuriame kitame resurse aprašomas failas yra.

`resource.`**level**
Duomenų šaltinio brandos lygis, vertinant tik pagal formatą, nežiūrint į šaltinyje esančių duomenų turinį.

`resource.`**access**
Viso duomenų šaltinio prieigos lygis. Paveldimas.

`resource.`**title**
Duomenų šaltinio pavadinimas.

`resource.`**description**
Duomenų šaltinio aprašymas.

Duomenų šaltinio *resource.type* reikšmė apibrėžia kokią *ETL* priemonę naudoti skaitant duomenis iš duomenų šaltinio. Automatizuota duomenų priemonė skirta įstaigos duomenų atvėrimui turėtų palaikyti tik tokius duomenų šaltinius, kurie naudojami įstaigos vidinėje infrastruktūroje.

Esant poreikiui gali būti įgyvendintas palaikymas naujiems duomenų šaltiniams.

10.5.3 Modelio bazė

Pastaba: Kol kas modelių apjungimas naudojant vieną bazę nėra įgyvendintas.

Modelio bazė naudojama kelių modelių (lentelių) susiejimui arba apjungimui. Kadangi įvairiuose duomenų šaltiniuose dažnai pasitaiko duomenų, kuriuose saugomos tą pačią semantinę prasmę turinčios lentelės, *base* stulpelyje galima nurodyti kaip skirtingos lentelės siejasi tarpusavyje.

base.type stulpelyje nurodoma kokiu būdu lentelės yra susiję. *ETL* priemonė vadovaujantis *base* informacija duomenis automatiškai transformuoja ir sujungia kelias lenteles į vieną.

Modeliai ne tik susiejami semantiškai tarpusavyje, bet taip pat suliejami ir dviejų modelių duomenys naudojant laukų sąrašą nurodytą *base.ref* stulpelyje. *base.ref* stulpelyje nurodyti laukai naudojami norint unikaliai identifikuoti *model* lentelėje esančią eilutę, kuri atitinka *base* lentelėje esančią eilutę. Tai reiškia, kad modelis ir jo bazė turi vienodus identifikatorius.

Siejant *model* ir *base* duomenis tarpusavyje, *model* lentelė įgauna lygiai tokius pačius unikalius identifikatorius, kurie yra base lentelėje. Tai reiškia, kad *model* lentelėje negali būti duomenų, kurių nėra *base* lentelėje.

model.property laukams, kurie sutampa su *base* modelio laukais, nenurodomas *property.type*, tokiu būdu nurodoma, kad *model.property* turi tą pačią semantinę prasmę, kaip ir *base*, tačiau *model* gali turėti ir papildomų laukų, kurių nėra *base* modelyje, tokiu atveju *property.type* turi būti nurodomas.

Visi *base.ref* laukai turi būti aprašyti tiek *base*, tiek *model* modeliuose, tai reiškia, kad *base.ref* gali būti naudojami tik tiek laukai, kurie neturi tipo.

Jei *base* stulpelyje nurodoma / reikšmė, tai reiškia, kad *model* neturi bazės, arba modelio bazė yra panaikinama. / naudojamas tais atvejais, kai norima vieną ar kelis modelius prijungti prie vienos bazės, tačiau sekantys modeliai nebeturi priklausyti jokiai bazei.

Sinonimai

Tais atvejais, kai visi *model* laukai neturi *property.type*, tada toks modelis laikomas *base* sinonimu ir iš esmės saugomas tik identifikatorius.

Tačiau, jei bent vienas *property.type* yra nurodytas, tada modelis įgyja fizinę reprezentaciją ir turi vieną ar kelis savo laukus, kurių nėra *base* modelyje.

Kiekvieną kartą saugant duomenis per kitą modelį į bazę, bazės modelio istorijoje išsaugoma informacija iš kokio modelio atėjo duomenys.

Paveldimumas

model paveldi visus laukus, įskaitant ir tuos, kurie nėra nurodyti prie *model* laukų sąrašo. Tai reiškia, kad galima skaityti ir rašyti duomenis į *base*, per *model*. Jei skaitomas ar rašomas laukas, kurio nėra *model* laukų sąraše, tada to lauko duomenys sakomi iš arba rašomi į *base* modelį.

Visi modelio laukai, kurie neturi tipo, fiziškai yra priskiriami *base* modeliui.

Dubliavimas

Laukai pavadinimai modelyje, kurie turi tą pačią semantinę prasmę, kaip ir bazėje turi sutapti su pavadinimais nurodytais bazėje. Tačiau, jei yra nurodomas jų tipas, tada tie duomenys dubliuojami, laikant, kad duomenys skiriasi nuo bazės, nepaisant to, kad semantiškai jie yra vienodi.

Turint tokius dubliuojamus laukus su nurodytais tipais, jei norima pasiekti bazės lauką, galima naudoti `_base.prop` išraišką, kuri nurodo, kad norima pasiekti bazėje esančius duomenis, laukui tuo pačiu pavadinimu.

`base.`**`source`**
: Nenaudojamas.

`base.`**`prepare`**
: Išimtiniais atvejais, kai nėra galimybės lentelių susieti ar apjungti įprastiniais metodais, galima pasitelkti formules, kurių pagalba galima įgyvendinti nestandartinius lentelių apjungimo atvejus.

`base.`**`type`**
: Nenaudojamas.

`base.`**`ref`**
: *model.property* reikšmė, kurios pagalba *model* objektai siejami su *base* objektais. Jei susiejimas pagal vieną model property yra neįmanomas, galima nurodyti kelis *model.property* pavadinimus atskirtus kableliu.

  Galima naudoti tik tuos *model.property*, kurie neturi nurodyto *property.type*, kas reiškia, kad toks pat laukas turi būti tiek *base*, tiek *model* laukų sąraše.

  Tais atvejais, kai *base.ref* rodo į modelio lauką, kuris turi tipą, tada *base.level* negali būti didesnis nei 3, kadangi jei modelio laukas turi tipą, tai reiškia, kad jo duomenys nesutampa su bazės duomenimis ir todėl jungimas negali būti daromas.

`base.`**`level`**
: *Brandos lygis*, nurodantis modelio susiejamumą su nurodytu baziniu modeliu. Plačiau žiūrėti *Ryšiai tarp modelių | Brandos lygis*.

Jei brandos lygis yra žemesnis nei 3, tada identifikatorių siejimas nėra atliekamas, tokiu būdu tiesiog nurodomas semantinis susiejimas metaduomenų, o ne duomenų lygmenyje.

`base.access`
Nenaudojamas.

Paaiškinimas, ką reiškia kiekviena savybė.

Pavyzdys be išorinio duomenų šaltinio:

d	r	b	m	property	type	ref
example						
			Location			
				name@lt	text	
				population	integer	
		Location				
			City			
				name@lt		
				population		
			Village			
				name@lt		
				population		
				region	ref	Location
		/				
			Country			
				name@lt		
				population		

Šiame pavyzdyje:

- `City` ir `Village` priklauso vienai bazei `Location`.
- Kadangi `Location` turi savybes `name@lt` ir `population`, tai `City` ir `Village` modeliuose tą pačią semantinę prasmę turinčios savybės turi turėti lygiai tokius pačius pavadinimus, o *`property.type`* turi būti tuščias. Kai *`property.type`* yra tuščias, tai reiškia, kad savybė ateina iš bazinio modelio.
- Kadangi `Village` turi papildomą *`property`* su nurodytu *`property.type`*, tai reiškia, kad `name` ir population` priklauso bazei, tačiau `region` priklauso `Village` modeliui ir jo nėra bazėje.
- Kadangi `Country` semantiškai nėra tas pats, kas `Gyvenviete`, nors ir turi tokias pačias savybes, atskiriame ją nuo `Location` bazės, priskirdami `/` bazei, kas reiškia, kas bazės nėra.

Pavyzdys su išoriniu duomenų šaltiniu:

d	r	b	m	property	type	ref	source
example							
			Location			id	
				id	integer		
				name@lt	text		
				population	integer		
		Location				name@lt	
			City			name@lt	CITY
				name@lt			NAME
				population			POPULATION
			Village			name@lt	VILLAGE
				name@lt			VILLAGE
				population			POPULATION
				region	ref	Location	REGION

Šiame pavyzdyje esminis skirtumas yra tas, kad nurodyta kaip daromas jungimas. `City` ir `Village` su `Location` jungiame per `name@lt` lauką.

10.5.4 Duomenų modelis

Duomenų modelis apibrėžia duomenų grupę turinčią tas pačias savybes. Skirtinguose duomenų šaltiniuose ir formatuose, duomenų modelis gali būti išreikštas skirtingomis formomis, pavyzdžiui `sql` duomenų šaltinio atveju, modelis aprašo vieną duomenų bazės lentelę.

Kiekvienas modelis turi turėti pirminį raktą, unikalų modelio duomenų identifikatorių. Pirminis raktas aprašomas pateikiant vieną ar kelias *`model.property`* reikšmes *`model.ref`* stulpelyje, kurios kartu unikaliai identifikuoja kiekvieną duomenų eilutę.

Išimtiniais atvejais, kai modelio duomenų laukų reikšmės turi būti generuojamos dinamiškai ar kitais nestandartiniais atvejais yra galimybė nurodyti model.type reikšmę. Jei *`model.type`* yra pateiktas, tada už modelio duomenų generavimą, įeinančių duomenų tikrinimą ir visos kitos su modeliu susijusios dalys gali būti pritaikytos konkretaus modelio atvejui. Tačiau, jei reikia keisti tik duomenų pateikimą, užtenka naudoti *`model.prepare`* formules.

`model.`**`source`**
: Modelio pavadinimas šaltinyje. Prasmė priklauso nuo *`resource.type`*.

`model.`**`prepare`**
: Formulė skirta duomenų filtravimui ir paruošimui, iš dalies priklauso nuo *`resource.type`*.

  Taip pat skaitykite: *Duomenų atranka*.

`model.type`
: Jei nurodytą, naudoti išplėstą modelio variantą, jei nenurodyta palikti tuščią. Jei tuščia, naudoti standartinį modelio variantą.

  Gali būti įrašoma reikšmė `absent`, kuri nurodo, kad modelis buvo ištrintas.

`model.ref`
: Kableliu atskirtas sąrašas *`model.property`* reikšmių, kurios kartu unikaliai identifikuoja vieną duomenų eilutę (pirminis lentelės raktas).

`model.level`
: Modelio *brandos lygis*, nusakantis pačio modelio brandos lygį, pavyzdžiui ar nurodytas pirminis raktas, ar modelio pavadinimas atitinka kodiniams pavadinimams kelimus reikalavimus.

`model.access`
: Modeliui priklausančių laukų *prieigos lygis*. Paveldimas.

`model.uri`
: Sąsaja su *išoriniu žodynu*.

`model.title`
: Modelio pavadinimas.

`model.description`
: Modelio aprašymas.

`model.property`
: Modeliui priklausantis duomenų laukas.

10.5.5 Savybė

Duomenų laukas atspindi tam tikrą modelio savybę arba tai gali būti lentelės stulpelis, jei duomenų šaltinis yra lentelė.

`property.source`
: Duomenų lauko pavadinimas šaltinyje. Prasmė priklauso nuo *`resource.type`*.

`property.prepare`
: Formulė skirta duomenų tikrinimui ir transformavimui arba statinės reikšmės pateikimui.

`property.type`
: Nurodomas loginis duomenų tipas. Dėl galimų tipų sąrašo žiūrėti *Duomenų tipai*.

  Loginis duomenų tipas yra toks tipas, kurį tikitės gauti publikuojant duomenis per API. Loginis tipas gali skirtis nuo duomenų šaltinio tipo.

  Visi duomenų tipai gali turėti tokius parametrus:

  - `required` - nurodo, kad šis duomenų laukas yra privalomas, tai reiškia, kad šio duomenų lauko reikšmė visada turi būti pateikta. Pagal nutylėjimą visi modelio duomenų laukai yra neprivalomi.

  Kai kurie duomenų tipai, gali turėti konkrečiam duomenų tipui pateikiamus papildomus parametrus, tokie parametrai nurodomi skliausteliuose.

  Dupmenų tipų pavyzdžiai:

  - `integer`
  - `integer required`
  - `geometry`
  - `geometry(linestringm, 3345) required`

`property.`**`ref`**
: Priklauso nuo `property.type`, nurodo matavimo vienetus, laiko ar vietos tikslumą, *klasifikatorių* arba *ryšį su kitais modeliais*. Ką tiksliai reiškia šis laukas, patikslinta skyrelyje *Duomenų tipai*.

`property.`**`level`**
: Nurodo duomenų lauko brandos lygį. Žiūrėti *Brandos lygiai*.

`property.`**`access`**
: Nurodo prieigos prie duomenų lygį. Žiūrėti skyrių *Prieigos lygiai*.

`property.`**`uri`**
: Sąsaja su išoriniu žodynu. Žiūrėti *Išoriniai žodynai*.

`property.`**`title`**
: Duomenų lauko pavadinimas. Šis pavadinimas yra skirtas skaityti žmonėms ir bus rodomas duomenų laukų sąrašuose ir antraštėse. Jei nenurodyta, bus naudojamas *`property`* kodinis pavadinimas.

`property.`**`description`**
: Duomenų lauko aprašymas.

`property.`**`enum`**
: Žiūrėti *Klasifikatoriai*.

10.6 Papildomos dimensijos

10.6.1 Išorinių žodynų prefiksai

Sąsają su išoriniais žodynais galima pateikti *`model.uri`* ir *`property.uri`* stulpeliuose. Tačiau prieš naudojant žodynus, pirmiausia reikia apsirašyti žodynų prefiksus. Žodynų prefiksai aprašomi taip:

`prefix`
: `prefix.`**`ref`**
  : Prefikso pavadinimas.

  `prefix.`**`uri`**
  : Išorinio žodyno URI.

  `prefix.`**`title`**
  : Prefikso antraštė.

  `prefix.`**`description`**
  : Prefikso aprašymas.

Rekomenduojama naudoti LOV prefiksus.

Aprašyti prefiksai gali būti naudojami *`model.uri`* ir *`property.uri`* stulpeliuose tokiu būdu: `prefix:name`.

Pavyzdys:

d	r	b	m	property	type	ref	uri
dataset1							
					prefix	spinta	https://github.com/atviriduomenys/manifest/issues/
						manifest	https://github.com/atviriduomenys/spinta/issues/
						vadovas	https://atviriduomenys.readthedocs.io/
						dct	http://purl.org/dc/dcmitype/
dataset2							
					prefix	dcat	http://www.w3.org/ns/dcat#
						dct	http://purl.org/dc/terms/
						dctype	http://purl.org/dc/dcmitype/
						foaf	http://xmlns.com/foaf/0.1/
						owl	http://www.w3.org/2002/07/owl#
						prov	http://www.w3.org/ns/prov#
						rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
						rdfs	http://www.w3.org/2000/01/rdf-schema#
						sdo	http://schema.org/
						skos	http://www.w3.org/2004/02/skos/core#
						vcard	http://www.w3.org/2006/vcard/ns#
						xsd	http://www.w3.org/2001/XMLSchema#

Prefiksai turi būti apibrėžti duomenų rinkinio kontekste, kadangi skirtingi duomenų rinkiniai gali naudoti skirtingus prefiksus, tiems pateims URI. Pavyzdžiui abiejusoe rinkinyje pavyzdyje aukščiau, `dct` ir `dctype` rodo į tą patį URI.

10.6.2 Klasifikatoriai

Tam tikri duomenų laukai turi fiksuotą reikšmių variantų aibę. Dažnai duomenų bazėse fiksuotos reikšmės saugomos skaitine forma ar kitais kodiniais pavadinimais. Tokias fiksuotas reikšmes duomenų struktūros apraše galima pateikti neužpildant hierarchinių stulpelių ir nurodant `type` reikšmę `enum`, pavyzdžiui:

id	d	r	b	m	property	type	ref	source	prepare	level	access	uri	title	description
1	datasets/example/places													
2		places				sql		sqlite://						
3				Place			id	PLACES						
4					id	integer		ID		3	open			
5					type	string		CODE		3	open			
6						enum		1	"city"				City	
7								2	"town"				Town	
8								3	"village"				Village	
9					name	string		NAME		3	open			

Šiame pavyzdyje `Place.type` laukas yra klasifikatorius, kurio reikšmės yra kodai 1, 2 ir 3, kurios duomenų struktūros apraše keičiamos į `city`, `town` ir `village`, papildomai `title` stulpelyje nurodant reikšmės pavadinimą.

Jei tas pats klasifikatorius gali būti naudojamas kelios skirtingose vietos, tada galima iškelti klasifikatorių ir suteikti jam pavadinimą, pavyzdžiui:

id	d	r	b	m	property	type	ref	source	prepare	level	access	uri	title	description
1	datasets/example/places													
6						enum	place	1	"city"				City	
7								2	"town"				Town	
8								3	"village"				Village	
2		places				sql		sqlite://						
3				Place			id	PLACES						
4					id	integer		ID		3	open			
5					type	string	place	CODE		3	open			
9					name	string		NAME		3	open			

Šiuo atveju, klasifikatoriui buvo suteiktas pavadinimas `place` įrašytas `enum.ref` stulpelyje, 6 eilutėje. O `Place.type` laukui, `prepare` stulpelyje nurodyta, kad šis laukas naudoja vardinį `place` klasifikatorių.

`enum`

`enum.ref`
: Pasirinkimų sąrašo pavadinimas.

`enum.source`
: Pateikiama originali reikšmė, taip kaip ji saugoma duomenų šaltinyje. Pateiktos reikšmės turi būti unikalios ir negali kartotis.

  Jei pageidaujama aprašyti tuščią šaltinio reikšmę, tada *property.prepare* celėje reikia nurodyti formulę, kuri tuščią reikšmę pakeičia, į kokią nors kitą. Formulės pavyzdys:

```
swap('', '-')
```

`enum.prepare`
: Pateikiama reikšmė, tokia kuri bus naudojama atveriant duomenis. *model.prepare* filtruose taip pat bus naudojama būtent ši reikšmė.

  `enum.prepare` reikšmės gali kartotis, tokiu būdu, kelios skirtingos `enum.source` reikšmės bus susietos su viena `enum.prepare` reikšme.

`enum.access`
: Klasifikatoriams galima nurodyti skirtingas prieigos teises, tokiu atveju, naudotojas turintis `open` prieigą matys tik tuos duomenis, kurių klasifikatorių reikšmės turi `open` prieigos teises, visi kiti bus išfiltruoti.

`enum.title`
: Fiksuotos reikšmės pavadinimas.

`enum.description`
: Fiksuotos reikšmės aprašymas.

Pagal nutylėjimą, jei *property.prepare* yra tuščias ir *property* turi *Klasifikatoriai* sąrašą, tada jei šaltinis turi neaprašytą reikšmę, turėtų būti fiksuojama klaida.

Jei yra poreikis fiksuoti tik tam tikras reikšmes, o visas kitas palikti tokias, kokios yra šaltinyje, tada *property.prepare* stulpelyje reikia įrašyti `self.choose(self)`.

10.6.3 Parametrai

Parametrai leidžia iškelti tam tikras duomenų paruošimo operacijas į parametrus kurie gali būti naudojami *Dimensijos*, kurioje apibrėžtas parametras kontekste. Parametrai gali gražinti *iteratorius*, kurių pagalba galima dinamiškai kartoti *resource* duomenų skaitymą, panaudojant aprašytus parametrus. Taip pat parametrų pagalba galima sudaryti reikšmių sąrašus, kurių pagalba galima kartoti *resource* su kiekviena reikšme.

Parametrai dažniausiai naudojami žemesnio brandos lygio duomenų šaltiniams aprašyti, o taip pat API atvejais, kai duomenys atiduodame dinamiškai.

Parametrai aprašomi pasitelkiant papildomą *Parametrai* dimensiją.

id	d	r	b	m	property	type	ref	source	prepare
1	datasets/example/cities								
2		places				csv		https://example.com/{}.csv	
3				Country			id	countries	
4					code	string		CODE	
5					title	string		TITLE	
6				City			country, title	cities/{country.code}	
7						param	country	Country	select(code)
8					country	ref	Country		param("country").code
9					title	string		TITLE	

param

`param.`**ref**
: Parametro *kodinis pavadinimas*.

`param.`**prepare**
: Formulė, kuri grąžina sąrašą reikšmių aprašomam parametrui.

`param.`**source**
: Jei reikšmė pateikta, tada ši reikšmė perduodama formulei kaip `self`. Pavyzdžiui, jei *`param.prepare`* pateikta formulė `select(code)`, o *`param.source`* nurodyta `Country`, tai formulė bus iškviesta taip `select("Country", code)`.

Jei parametro reikšmė yra *iteratorius*, tada *dimensija*, kurios kontekste yra aprašytas *parametras* yra kartojama tiek kartų, kiek reikšmių grąžina *iteratorius*.

Jei yra keli *Parametrai* grąžinantys *iteratorius*, tada iš visų *iteratorių* sudaroma Dekarto sandauga ir *`resource`* dimensija vykdoma su kiekviena sandaugos rezultato reikšme.

Jei sekančioje *DSA* eilutėje, einančioje po eilutės, kurioje aprašytas *Parametrai*, nenurodytas *`type`* ir neužpildytas joks kitas *dimensijos* stulpelis, tada parametras tampa *iteratoriumi*, kurio reikšmių sąrašą sudaro sekančiose eilutėse patektos *`source`* ir *`prepare`* reikšmės. Pavyzdžiui anksčiau pateiktą pavyzdį galima būtų perdaryti taip:

id	d	r	b	m	property	type	ref	source	prepare
1	datasets/example/cities								
2		places				csv		https://example.com/{}.csv	
3				Country			id	countries	
4					code	string		CODE	
5					title	string		TITLE	
6				City			country, title	cities/{country}	
7						param	country		"lt"
7									"lv"
7									"ee"
8					country	ref	Country		param("country")
9					title	string		TITLE	

Šiame pavyzdyje, parametras `country` grąžins tris šalies kodus: lt, lv ir ee, kurie bus panaudojami `cities/{country}` pavadinime, pakeičiant `{country}` dalį.

Parametrai reikšmės pasiekiamos naudojanti pavadinimą įrašytą *param.ref* stulpelyje. Pavyzdžiui, jei *param.ref* stulpelyje įrašyta `x`, tada `x` parametro reikšmę galima gauti taip:

source
: `{x}`.

prepare
: `x` arba `param(x)`.

Parametrų generavimui galima naudoti tokias formules:

param.prepare
: **range**(*stop*)
  Sveikų skaičių generavimas nuo 0 iki `stop`, `stop` neįeina.

  range(*start*, *stop*)

  Sveikų skaičių generavimas nuo `start` iki `stop`, `stop` neįeina.

  scalar(*name*)
  Jei nurodytas *param.source*, tada imama tik `name` lauko reikšmė, o ne visi modelio laukai.

Jei užpildytas *param.source* stulpelis, tada *param.prepare* stulpelyje galima naudoti filtrą nurodyto *param.source* modelio duomenims filtruoti, o naudojant parametrus galima nurodyti ir modelio laukų pavadinimus, pavyzdžiui:

source
: `{x.field}`.

prepare
: `x.field` arba `param(x).field`.

10.6.4 Reikšmių sukeitimas

Tam tikrais atvejais duomenis tenka normalizuoti parenkant tam tikrą reikšmę jei tenkinama nurodyta sąlyga. Tokias situacijas galima aprašyti pasitelkiant *switch* dimensiją.

switch
: `switch.`**source**
  Reikšmė, kuri bus atveriama.

  `switch.`**prepare**
  Sąlyga, naudojant einamojo modelio laukus. Jei sąlyga tenkinama, tada laukui priskiriama *switch.source* reikšmė. Jei sąlyga netenkinama, tada bandoma tikrinti sekančią sąlygą. Parenkama ta reikšmė, kurios pirmoji sąlyga tenkinama.

  Jei *switch.prepare* yra tuščias, tada sąlyga visada teigiama ir visada grąžinama *switch.source* reikšmė.

10.6.5 Komentavimas

Dirbant su *DSA* yra galimybė komentuoti eilutes, naudojant papildomą *comment* dimensiją, kurią galima naudoti bet kurios kitos dimensijos kontekste.

comment

comment.id
Komentaro numeris.

comment.ref
Komentuojamo vieno ar kelių kableliu atskirtų *property* pavadinimai. Galima nurodyti ne tik stulpelio pavadinimą, bet ir dimensiją.

comment.source
Komentaro autorius.

comment.prepare
Keitimo pasiūlymas, naudojant `create()`, `update` ir `delete()` funkcijas. Pavyzdžiui:

```
update(property: "pavadinimas@lt", type: "text")
```

Šiuo atveju nurodoma, kad siūloma keisti `property` pavadnimą į `pavadinimas@lt`, o `type` į `text`.

comment.level
Nurodoma, kad patenkinus keitimo sliūlymą, kuris nurodytas *comment.prepare* stulplyje, komentuojamai eilutei gali būti suteiktas nurodytas brandos lygis.

comment.access
Nurodoma, ar komentaras gali būti publikuojamas viešai.

private Komentaras negali būti publikuojamas viešai. Šis prieigos lygis naudojamas pagal nutylėjimą.

open Komentaras gali būti publikuojamas viešai.

comment.uri
Viena ar kelios kableliu atskirtos šaltinio nuorodos, kuri pateikta daugiau informacijos apie tai, kas komentuojama. Taip pat gali būti nurodytas kito komentaro *comment.id*, nurodant, kad tai yra atsakymas į ankstesnį komentarą.

URI pateikiami sutrumpinta forma, naudojant prefikstus. Žiūrėti skrių *Išoriniai žodynai*.

comment.title
Komentaro data, ISO 8601 formatu.

comment.description
Komentaro tekstas.

Pavyzdys

d	r	b	m	property	type	ref	prepare	level	access	uri
example										
					prefix	spinta				https://github.com/atviriduomenys/manifest/is
						manifest				https://github.com/atviriduomenys/spinta/issu
						vadovas				https://atviriduomenys.readthedocs.io/
			Imone					2		
					comment	base	update(base: "/jar/JuridinisAsmuo", ref: "id")	4	open	spinta:205, manifest:1290
					comment	ref	update(ref: "id")	4	open	vadovas:dsa/dimensijos.html#model.ref
				id	integer			4	open	
				pavadinimas	string			2	open	
					comment	ref	update(property: "pavadinimas@lt", type: "text")	4	open	spinta:204

10.6.6 Daugiakalbiškumas

`title` ir `description` stulpeliuose tekstas rašomas lietuvių kalba, tačiau galima pateikti tekstą ir kita kalba, panaudojus papildomą `lang` dimensiją, kurią reikia naudoti prieš eilutę, kuriai pateikiamas tekstas kita kalba.

lang

lang.**ref**
: ISO 639-1 dviejų simbolių kalbos kodas.

lang.**title**
: Pavadinimas `lang.ref` stulpelyje nurodyta kalba.

lang.**description**
: Aprašymas `lang.ref` stulpelyje nurodyta kalba.

10.6.7 Struktūros keitimas

Laikui einant, pirminių duomenų šaltinių arba jau atvertų duomenų struktūra keičiasi, papildoma naujais *modeliais* ar *savybėmis*, keliant duomenų brandos lygį seni duomenys keičiami naujais, aukštesnio brandos lygio duomenimis.

Visi šie struktūros ar pačių duomenų pasikeitimai fiksuojami papildomos `migrate` dimensijos pagalba, kuri gali būti naudojama, bet kurios kitos dimensijos kontekste.

Pastaba: Migracijos naudojamos tik tuo atveju, kai keičiasi duomenų struktūra arba patys duomenys. Jei keičiasi tik

metaduomenys, tai migracijų sąraše neatsispindi.

id	d	r	b	m	property	type	ref	prepare	level	title	description
1						migrate				2021-12-21 16:29	Pirmoji migracija.
2						migrate	1			2021-12-21 16:33	Antroji migracija.
3						migrate	2			2022-06-21 16:41	Trečioji migracija.
	datasets/example/migrate										
				Country			id				
					id	integer			4		
					code	string			3		
						migrate	1	create(level: 2)			
						migrate	3	update(level: 3)			
					name	string					
						migrate	2	create()			

Pavyzdyje aukščiau matome, kad šis duomenų struktūros aprašas turi tris migracijas:

1. Pirmosios migracijos metu sukuriamas pradinis duomenų struktūros variantas. Pirmoji migracija nežymima prie modelių ir duomenų laukų, nebent daromas keitimas, tuomet įtraukiam ir pirmoji migracija, kad būtų matoma, kas keitėsi. Būtent toks atvejis parodytas prie `Country.code` lauko, kuri trečiojo migracijoje keičiamas brandos lygis.
2. Antrosios migracijos metu buvo įtrauktas naujas duomenų laukas `Country.name`.
3. Trečiosios migracijos metu, buvo keičiami `Country.code` lauko duomenys, pakeitimo metu brandos lygis buvo pakeltas iki trečio. Atkreipkite dėmesį, kad metaduomenų pasikeitimas, kaip šiuo atveju, žymimas migracijose tik tuo atveju, jei tai yra susiję su pačių duomenų pasikeitimu.

   Jei brandos lygis būtų pakeistas, nekeičiant pačių duomenų, tuomet tokio pakeitimo nereikėtų įtraukti į migracijų sąrašą.

   Kadangi trečiojoje migracijoje buvo atliktas su ankstesne versija nesuderinamas pakeitimas, tai šios migracijos data yra 6 mėnesiai ateityje, kadangi nesuderinamos migracijos pirmiausia paskelbiamos, o įgyvendinamos tik praėjus 6 mėnesiams nuo paskelbimo.

migrate

migrate.id

Migracijos numeris (UUID). Kiekvienos migracijos metu gali būti atliekama eilė operacijų, visos operacijos fiksuojamos naudojant migracijos numerį.

Visų migracijų sąrašas pateikiamas, kai *migrate* nepriklauso jokiam dimensijos kontekstui.

migrate.ref

Ankstesnės migracijos numeris, pateiktas *migrate.id* stulpelyje, arba tuščia, jei prieš tai jokių kitų

migracijų nebuvo.

Naudojamas jei *migrate* nepatenka į jokios dimensijos kontekstą.

Jei *migrate* aprašomas dimensijos kontekste, tada šis stulpelis nenaudojamas.

migrate.**prepare**
Migracijos operacija. Galimos tokios operacijos:

create()
Priklausomai nuo dimensijos konteksto, prideda naują modelį, arba savybę.

Funkcijai galima perduoti `ref` ir kitus vardinius argumentus, kurie atitinka *DSA* lentelės metaduomenų stulpelių pavadinimus.

update()
Taikomas tik duomenų laukams ir nurodo, kad buvo pakeistos esamų duomenų reikšmės, keičiant reikšmių dimensiją, matavimo vienetus, formatą ir kita.

Funkcijai galima perduoti `ref` ir kitus vardinius argumentus, kurie atitinka *DSA* lentelės metaduomenų stulpelių pavadinimus.

Perduodami tik tie vardiniai argumentai, kuriuos atitinkantys metaduomenys keičiasi.

delete()
Priklausomai nuo dimensijos konteksto, šalina modelį ar savybę.

Pašalinto modelio ar savybės *type* keičiamas į `absent` reikšmę.

filter(*where*)
Naudojamas *property* kontekste, kai vykdoma duomenų migracija. Nurodo, kad migracija taikoma tik `where` sąlygą tenkinantiems duomenims.

Be šių pagrindinių migracijos operacijų, galima naudoti kitas duomenų transformavimo operacijas, kurios vykdomos su kiekviena duomenų eilute ir atlikus pateiktas transformacijos funkcijas, pakeista reikšmė išsaugoma.

migrate.**title**
Migracijos įvykdymo data ir laikas. Migracijos laikas ir data gali būti ir ateityje, tuo atveju, jei daromas nesuderinamas keitimas.

Naudojamas tik tada, kai *migrate* nepatenka į jokios dimensijos kontekstą.

migrate.**description**
Migracijos atliekamo pakeitimo trumpas aprašymas.

10.7 Duomenų tipai

absent

Žymi *savybę*, kuri buvo ištrinta ir nebenaudojama. Žiūrėti *Struktūros keitimas*.

boolean

Loginė reikšmė.

Brandos lygis

1

- Duomenyse nėra vientisumo, kartais `true` pateikta kaip `1`, kartais akip `taip`, arba `yes`.

2

- Visi duomenys pateikti vienoda, tačiau nestandartine forma.

3

- Duomenys pateikti standartine forma. Standartinė forma priklauso nuo pasirinkto duomenų saugojimo formato. Pavyzdžiui JSON formatu `boolean` tipas išreiškiamas kaip `true` ir `false`.

`integer`

Sveikas skaičius.

Mažiausia galima reikšmė: `-2147483648`.

Didžiausia galima reikšmė: `2147483647`.

`property.ref` stulpelyje, nurodomi *Matavimo vienetai*.

Brandos lygis

1

- Duomenys pateikti skirtingais vienetais, pavyzdžui dalis duomenų pateikti metrais, dalis kilometrais ir dalis milimetrais.
- Amžius pateiktas atskirai, nurodant metus, menesius ir dienas. Šiuo atveju brandos lygis yra 1, kadangi neaišku, kiek yra dienų metuose ir mėnesiuose, kadangi skirtingi metai ir skirtingi mėnesiai turi skirtingą dienų skaičių. Tokiais atvejais duomenis reikia pateikti dienomis.

2

- Duomenys pateikti išskaidant vieną reikšmę į kelias reikšmes, skirtingais vienetais. Duomenys turėtu būti pateikti mažiausiu deltaumu, viename duomenų lauke. Pavyzdžiui atstumas pateiktas atskirais duomenų laukais, kur viename nurodomas atstumas kilometrais, kitame metrais, trečiame milimetrais. Šiuo atveju, duomenys turėtu būti pateikiami milimetrais.

3

- Duomenys yra kiekybiniai, tačiau *`ref`* stulpelyje nenurodyti vienetai.

`number`

Realusis skaičius, apvalinamas naudojant slankiojo kablelio aritmetiką, kur sveikoji skaičiaus dalis gali būti šešių skaitmenų dydžio.

`property.ref` stulpelyje, nurodomi *Matavimo vienetai*.

Sveikoji dalis atskiriama `.` simbolių.

`binary`

Dvejetainiai duomenys. Bendras baitų skaičius turi būti ne didesnis nei 1G.

10.7.1 Tekstiniai duomenys

Tekstiniai duomenys skirstomi į du skirtingus tipus `string` ir `text`.

`string`

Simbolių eilutė. Neriboto dydžio, tačiau fiziškai simbolių eilutė turėtu būti ne didesnė, nei 1G.

Simboliu eilutė turėtu būti pateikta UTF-8 koduote.

Šiuo tipu žymimi duomenų laukai, kuriuose tekstas pateiktas ne žmonių kalba. Tai gali būti įvairūs kategoriniai duomenys, identifikatoriai ar kito pobūdžio simbolių eilutės, kurios nėra užrašytos natūraliąja žmonių kalba.

`text`

Natūraliaja žmonių kalba užrašytas tekstas.

Galima nurodyti kokia kalba užrašytas tekstas naudojant ISO 639-1 kodus. Kalbos kodas nurodomas *`property`* stulpelyje, prie pavadinimo įrašant `@<kodas>`, kur `<kodas>` yra pakeičiamas į dviejų raidžių kalbos kodą. Pavyzdžiui `pavadinimas@lt`. Plačiau apie tai RDF Turtle specifikacijoje iš kur ir buvo pasiskolintas toks kalbų žymėjimas.

Tekstas turėtu būti pateikta UTF-8 koduote. Jei šaltinyje tekstas nėra UTF-8 koduotės, tuomet galima *`prepare`* stulepyje įrašoų formulių pagalba galima nurodyti transformavimo taisykles iš šatinio naudojamos į UTF-8 koduotę.

`property.ref` galima pateikti teksto formatą, nadojant vieną iš šių formatų:

- `html` - tekstas pateiktas HTML formatu.
- `md` - tekstas pateiktas Markdown formatu.
- `rst` - tekstas pateitkas reStructuredText formatu.
- `tei` - tekstas pateiktas TEI formatu.

Pavyzdys:

d	r	b	m	property	type	ref
example						
			Country			
				name@lt	text	
				description@lt	text	html

Šiame pavyzdyje `@lt` nurodo, kad šalies pavadinimai ir aprašymai pateikti Lietuvių kalba. Papildomai, šalies aprašymo teksto formatas yra HTML tipo.

10.7.2 Data ir laikas

`datetime`

Data ir laikas atitinkantis ISO 8601.

Mažiausia galima reikšmė: `0001-01-01T00:00:00`.

Didžiausia galima reikšmė: `9999-12-31T23:59:59.999999`.

Pagal ISO 8601 standartą, data gali būti pateikta tokia forma:

```
YYYY-MM-DD[*HH[:MM[:SS[.fff[fff]]]][+HH:MM[:SS[.ffffff]]]]
```

Simbolis * reiškia, kad galima pateikti bet kokį vieną simbolį, dažniausiai naudojamas tarpo simbolis, arba raidė `T`.

`property.ref` stulpelyje, nurodomas datos ir laiko tikslumas sekundėmis. Tikslumą galima nurodyti laiko vienetais, pavyzdžiui `Y`, `D`, `S`, arba `5Y`, `10D`, `30S`. Visi duomenys turi atitikti vienodą tikslumą, tikslumas negali varijuoti. Galimi vienetų variantai:

Reikšmė	Prasmė
Y	Metai
M	Mėnesiai
Q	Metų ketvirčiai
W	Savaitės
D	Dienos
H	Valandos
T	Minutės
S	Sekundės
L	Milisekundės
U	Mikrosekundės
N	Nanosekundžės

Brandos lygis

1

- Data ir laikas pateikti naudojant skirtingus formatus, pavyzdžiui `2020-01-31`, `01/31/2020`, `31.1.20`.
- Data ir laikas pateikti laisvu tekstu, pavyzdžiui `2020 paskutinę pirmo mėnesio dieną`.

2

- Duomenys pateikti nesatandartiniu formatu, tačiau visi duomenys pateikti vienodu formatu. Pavyzdžiui visi duomenys pateikti `01/31/2020` formatu.
- Duomenys pateikti atskiruose laukuose, pavyzdžiui metai pateikti viename `integer` tipo lauke, o ketvirtis, kitame `integer` tipo lauke. Norint didesnio brandos lygio, duomenys turi būti viename `date` tipo lauke su `property.ref = Q`.

3

- Duomenys pateikti standartiniu ISO 8601 formatu.
- Nenurodytas *`property.ref`*, kuriame turėtu būti pateiktas duomenų tikslumas.

`date`

Tas pats kas `datetime` tik dienos tikslumu. Šio tipo reikšmės taip pat turi atitikti ISO 8601:

```
YYYY-MM-DD
```

Jei norima nurodyti datą žemesnio nei dienos tikslumo, tada vietoj mėnesio ir dienos galima naudoti `01` ir *`property.ref`* stulpelyje nurodyti tikslumą:

Reikšmė	Prasmė
Y	Metai
M	Mėnesiai
Q	Metų ketvirčiai
W	Savaitės
D	Dienos

`time`

Dienos laikas, be konkrečios datos. Šio tipo reikšmės, kaip ir kiti su laiku sisję tipai turi atitikti ISO 8601:

```
HH[:MM[:SS[.fff[fff]]]][+HH:MM[:SS[.ffffff]]]
```

Jei norima nurodyti žemesnio nei sekundžių tikslumo laiką, tada vietoj minučių ir/ar sekundžių galima naudoti `00` ir *`property.ref`* stulpelyje nurodyti tikslumą:

Reikšmė	Prasmė
H	Valandos
T	Minutės
S	Sekundės
L	Milisekundės
U	Mikrosekundės
N	Nanosekundžės

`temporal`

Apibrėžtis laike.

Šis tipas atitinka `datetime`, tačiau nurodo, kad visas model yra apibrėžtas laike, būtent pagal šią savybę. Tik viena model savybė gali turėti `temporal` tipą. Pagal šios savybės reikšmes apskaičiuojamas ir įvertinamas dct:temporal.

10.7.3 Erdviniai duomenys

`geometry`

Erdviniai duomenys. Duomenys pateikiami WKT formatu, naudojant EPSG duomenų bazės parametrus, skirtingoms projekcijoms išreikšti.

`property.ref` stulpelyje nurodomas tikslumas metrais. Tikslumą galima pateikti naudojanti SI vienetus, pavyzdžiui `m`, `km` arba `10m`, `100km`.

`geometry` tipas gali turėti du argumentus `geometry(form, crs)`:

- `form` - geometrijos forma

- `crs` - koordinačių sistema

Pats tipas gali būti pateiktas vienu iš šių variantų:

- `geometry(form, crs)` - nurodant formą ir koordinačių sistemą
- `geometry(crs)` - nurodant tik koordinačių sistemą
- `geometry(form)` - nurodant tik formą
- `geometry` - be argumentų.

Geometrijos forma (`form`)

Galimi tokie geometrijos tipai:

- `point` - taškas.
- `linestring` - linija.
- `polygon` - daugiakampis (pradžios ir pabaigos taškai **turi** sutapti).
- `multipoint` - keli taškai.
- `multilinestring` - kelios linijos.
- `multipolygon` - keli daugiakampiai (kiekvieno daugiakampio pradžios ir pabaigos taškai **turi** sutapti).

Kiekviena iš formų gali turėti tokias galūnes nurodančias papildomą dimensiją:

- `z` - aukštis.
- `m` - pasirinktas matmuo (pavyzdžiui laikas, atstumas, storis ir pan.)
- `zm` - aukštis ir pasirinktas matmuo.

Jei geometrijos forma nenurodyta, tada duomenys gali būti bet kokios geometrinės formos. Jei forma nurodyta, tada visi duomenys turi būti tik tokios formos, kokia nurodyta.

Koordinačių sistema (`crs`)

Antrasis `geometry` argumentas nurodomas pateikiant SRID numerį, kuris yra konkrečios koordinačių sistemos identifikacinis numeris EPSG duomenų bazėje. Jei koordinačių sistemos numeris nenurodytas, tuomet daroma prielaida, kad erdviniai duomenys atitinka `4326` (WGS84) koordinačių sistemą.

Svarbu, kad pateikiant duomenis, koordinačių ašių eiliškumas atitiktų tokį eiliškumą, kuris nurodytas EPSG parametrų duomenų bazėje, konkrečiai koordinačių sistemai, kuria pateikiami duomenys.

Pilną SRID kodų sąrašą galite rasti epsg.io svetainėje. Keletas dažniau naudojamų SRID kodų:

		ašis #1		ašis #2		
SRID	CRS	kryptis	žymėjimas	kryptis	žymėjimas	vienetai
4326	WGS84	šiaurė	latitude (platuma)	rytai	longitude (ilguma)	laipsniai
3346	LKS94	šiaurė	x (abscisė)	rytai	y (ordinatė)	metrai
3857	WGS84 / Pseudo-Mercator	rytai	x (abscisė)	šiaurė	y (ordinatė)	metrai
4258	ETRS89	šiaurė	latitude (platuma)	rytai	longitude (ilguma)	laipsniai

Atkreipkite dėmesį, kad LKS94 koordinačių sistemoje geometrinės ašys neatitinka matematinių ašių ir yra sukeistos vietomis. Įprastai šiaurė ir y ašis yra viršuje, tačiau LKS94 atveju šiaurėje yra x ašis.

> Ašinio meridiano projekcija yra abscisių (x) ašis. Šios ašies teigiamoji kryptis nukreipta į šiaurę. Ordinačių (y) ašies teigiamoji kryptis nukreipta į rytus.
>
> —https://www.e-tar.lt/portal/lt/legalAct/TAR.6D575923F94A

Prieš publikuojant duomenis, galite pasitikrinti, ar koordinačių ašys pateikiamos teisinga tvarka, naudotami taško atvaizdavimo įrankį.

Pavyzdžiui, norint patikrinti Vilniaus Katedros varpinės bokšto taško koordinates, LKS94 (EPSG:3346) sistemoje, galite naršyklės adreso juostoje pateikti šį adresą:

https://get.data.gov.lt/_srid/3346/6061789/582964

Jei ašių eiliškumas teisingas, gausite tašką ten kur tikėjotės, jei ašys sukeistos vietomis, tada taškas žemėlapyje gali būti visai kitoje vietoje, nei tikėjotės.

Adreso formatas:

```
/_srid/{srid}/{ašis1}/{ašis2}
```

- `{srid}` - EPSG duomenų bazėje esančios koordinačių sistemos SRID kodas
- `{ašis1}` - pirmosios ašies reikšmė (kryptis priklauso nuo `{srid}`)
- `{ašis2}` - antrosios ašies reikšmė (kryptis priklauso nuo `{srid}`)

Pavyzdžiai (strukūros aprašas)

- `geometry` - WGS84 projekcijos, bet kokio tipo geometriniai objektai.
- `geometry(3346)` - LKS94 projekcijos, bet kokio tipo geometriniai objektai.
- `geometry(point)` - GWS84 projekcijos, bet `point` tipo geometriniai objektai.
- `geometry(linestringm, 3345)` - LKS94 projekcijos, `linestringm` tipo geometriniai objektai su pasirinktu matmeniu, kaip trečia dimensija.

Pavyzdžiai (duomenys)

Vilniaus Katedros varpinės bokšto taškas, LKS94 (EPSG:3346) koordinačių sistemoje:

```
POINT (6061789 582964)
```

Brandos lygis

1

- Nenurodytas koordinačių sistema ir duomenys pateikti skirtingomis koordinatėmis.
- Sumaišytos ašys, pavyzdžiui vieni duomenys pateikiami x, y, kiti y, x.
- Sumaišyti vienetai, pavyzdžiui vieni duomenys pateikti metrais, kiti laipsniais.
- Pateiktas adresas, nenurodant adreso koordinačių.

2

- Nenurodyta koordinačių sistema, tačiau visi duomenys pateikti naudojant vienodą koordinačių sistemą.

3

- Nenurodytas *property.ref*, kuriame turėtu būti pateiktas duomenų tikslumas metrais.

spatial

Apibrėžtis erdvėje.

Šis tipas atitinka `geometry`, tačiau nurodo, kad visas model yra apibrėžtas erdvėje, būtent pagal šią savybę. Tik viena model savybė gali turėti `spatial` tipą. Pagal šios savybės reikšmes apskaičiuojamas ir įvertinamas dct:spatial.

10.7.4 Valiuta

money

Valiuta. Saugomas valiutos kiekis, nurodant tiek sumą, tiek valiutos kodą naudojant ISO 4217 kodus.

Valiutos kodas nurodomas *property.ref* stulpelyje.

Pavyzdys:

d	r	b	m	property	type	ref	source
example							
			Product				PRODUCT
				price	money	EUR	PRICE

Jei valiutos suma ir pavadinimas saugomi atskirai, tuomet valiutą galima aprašyti taip:

d	r	b	m	property	type	ref	source	prepare
example								
			Product				PRODUCT	
				amount			PRICE	
				currency			CURRENCY_CODE	
				price	money			money(amount, currency)

Šio tipo duomenys pateikiami viena iš šių formų:

```
123
123.45
123 EUR
123.45 EUR
```

10.7.5 Failai

`file`

Šis duomenų tipas yra sudėtinis, susidedantis iš tokių duomenų:

id Laukas, kuris unikaliai identifikuoja failą, šis laukas duomenų saugojimo metu pavirs failo identifikatoriumi, jam suteikiant unikalų UUID.

name Failo pavadinimas.

type Failo media tipas.

size Failo turinio dydis baitais.

content Failo turinys.

Šiuos metaduomenis galima perduoti `file()` funkcijai, kai vardinius argumentus. Pavyzdžiui:

d	r	b	m	property	type	source	prepare	access
datasets/example								
			Country					
				name	string	NAME		open
				flag_file_name	string	FLAG_FILE_NAME		private
				flag_file_data	binary	FLAG_FILE_DATA		private
				flag	file		file(name: flag_file_name, content: flag_file_data)	open

Šiame pavyzdyje, iš `flag_file_name` ir `flag_file_data` laukų padaromas vienas `flag` laukas, kuriame panaudojami duomenys iš dviejų laukų. Šiuo atveju, `flag_file_name` ir `flag_file_data` laukai tampa pertekliniais, todėl `access` stulpelyje jie pažymėti `private`.

Analogiškai, tokius pačius duomenis galima aprašyti ir nenaudojant formulių:

d	r	b	m	property	type	source	prepare	access
datasets/example								
			Country					
				name	string	NAME		open
				flag	file			open
				flag._name		FLAG_FILE_NAME		open
				flag._content		FLAG_FILE_DATA		open

`image`

Paveiksliukas. `image` tipas turi tokias pačias savybes kaip `file` tipas.

10.7.6 Išoriniai raktai

Taip pat žiūrėkite: *Ryšiai tarp modelių*.

Išoriniai raktai iš dalies yra panašūs į sudėtinius tipus, kadangi laukas, kuris rodo į kitą objektą, yra traktuojamas, kaip kitas objektas.

ref

Ryšys su modeliu. Šis tipas naudojamas norint pažymėti, kad lauko reikšmė yra *property.ref* stulpelyje nurodyto modelio objektas.

Pagal nutylėjimą, jungimas su kito modelio objektais daromas per siejamo pirminį raktą (*model.ref*), tačiau yra galimybė nurodyti ir kitą, nebūtinai pirminį raktą.

Jei jungimas daromas, ne per pirminį raktą, tuomet, laukai per kuriuos daromas jungimas nurodomi *property.ref* stulpelyje laužtiniuose sklaustuose, pavyzdžiui:

```
Country[code]
```

Čia jungiama su `Country` modeliu, per `Country` modelio `code` duomenų lauką.

Jei laukas, per kurį daromas jungimas nenurodytas, pavyzdžiui:

```
Country
```

Tada, jungimas daromas per `Country` modelio pirminį raktą, kuris nurodytas *model.ref* stulpelyje.

Šio objekto reikšmės yra pateikiamos, kaip dalis objekto į kurį rodoma. Jei `ref` tipo lauko brandos lygis (*property.level*) yra 4 ar didesnis, tuomet šio duomenų tipo reikšmės atrodo taip:

```
{"_id": "69c98b0f-9e4e-424b-9575-9f601d79b68e"}
```

Jei brandos lygis (*property.level*) yra žemesnis nei 4, tada reikšmė atrodo taip:

```
{"id": "69c98b0f-9e4e-424b-9575-9f601d79b68e"}
```

Čia `id` yra *model.ref* arba *kitas laukas*, per kurį daromas jungimas. Jei nenurodytas nei *model.ref*, nei *kitas laukas*, tada jungimas daromas per `_id`, tačiau netikrinama ar toks `_id` egzistuoja jungiamame modelyje.

backref

Atgalinis ryšys su modeliu.

Šis tipas naudojamas norint pažymėti, kad tam tikras kitas modelis turi `ref` tipo lauką, kuris rodo į šį modelį. Šis laukas pats duomenų neturi, tai tik papildomas metaduomuo, padedantis geriau suprasti ryšius tarp modelių.

Taip pat žiūrėkite *Jungimas atgaliniu ryšiu*.

generic

Dinaminis ryšys su modeliu.

Šis tipas naudojamas tada, kai yra poreikis perteikti dinaminį ryšį, t. y. duomenys siejami ne tik pagal id, bet ir pagal modelio pavadinimą. Tokiu būdu, vieno modelio laukas gali būti siejamas su keliais modeliais.

Taip pat žiūrėkite *Polimorfinis jungimas*.

Šis duomenų tipas yra sudėtinis, susidedantis iš tokių duomenų:

object_model Pilnas modelio pavadinimas, su kuriuo yra siejamas objektas.

object_id `object_model` modelio objekto id.

10.7.7 Sudėtiniai tipai

`object`

Kompozicinis tipas.

Šis tipas naudojamas apibrėžti sudėtiniams duomenims, kurie aprašyti naudojant kelis skirtingus tipas. Kompozicinio tipo atveju property stulpelyje komponuojami pavadinimai atskiriami taško simboliu.

Sudarant duomenų modelį, rekomenduojama laikytis plokščios struktūros ir komponavimą įgyvendinti siejant modelius per `ref` ar `generic` tipus.

`array`

Masyvas.

Šis tipas naudojamas apibrėžti duomenų masyvams. Jei masyvo elementai turi vienodus tipus, tada elemento tipas pateikiamas property pavadinimo gale prirašant [] sufiksą, kuris nurodo, kad aprašomas ne pats masyvas, o masyvo elementas.

Rekomenduojama vengti naudoti šį tipą, siekiant išlaikyti plokščią duomenų modelį. Vietoje `array` tipo rekomenduojama naudoti `backref`.

10.7.8 Kiti tipai

`url`

Unikali resurso vieta (URL) (angl. *Uniform Resource Locator*).

Šis tipas naudojamas pateikiant nuorodas į išorinius šaltinius.

https://en.wikipedia.org/wiki/Uniform_Resource_Locator

`uri`

Unikalus resurso identifikatorius (URI) (angl. *Uniform Resource Identifier*).

Šis tipas naudojamas tais atvejais, kai pateikiamas išorinio resurso identifikatorius, RDF duomenų modelyje tai yra subjeto identifikatorius.

https://en.wikipedia.org/wiki/Uniform_Resource_Identifier

10.8 Kodiniai pavadinimai

Kadangi *DSA* lentelė skirta naudoti tiek žmonėms tiek automatizuotoms priemonėms, tam tikros lentelės dalys privalo naudoti sutartinius kodinius pavadinimus. Kodiniams pavadinimams keliami griežtesni reikalavimai, kadangi šiuos pavadinimus interpretuos automatizuotos priemonės.

Visi *DSA* lentelės stulpelių pavadinimai turi būti užrašyti tiksliai taip, kaip nurodyta, kad kompiuterio programos galėtų juos atpažinti.

Kodiniai pavadinimai rašomi naudojant tik lotyniškas raidas. Lietuviškų raidžių naudoti negalima, todėl geriausia pavadinimus užrašyti anglų kalba, arba pakeičiant lietuviškas raides į lotyniškos raidės analogą.

Deja, vis dar pasitaiko vietų, kuriose palaikoma tik lotyniška abėcėlė, todėl ir keliamas toks reikalavimas, siekiant užtikrinti maksimalų suderinamumą tarp skirtingų sistemų.

Pavadinimai turėtu būti rašomi laikantis tokio stiliaus:

10.8.1 Vardų erdvių

Pavyzdys: `datasets/gov/abbr/short/word`

Visos mažosios raidės, stengiantis naudoti vieno žodžio trumpus pavadinimus arba žodžio trumpinius. Kadangi vardų erdvė rašoma prie kiekvieno modelio pavadinimo, todėl reikia stengtis vardų erdvių ir duomenų rinkinių pavadinimus išlaikyti kiek įmanoma trumpesnius.

Vardų erdvės pavadinimai užrašomi daugiskaita ir turi prasidėti mažąja raide.

10.8.2 Modeliai

Pavyzdys: `UpperCamelCase`

Kiekvieno modelio pavadinimo pirma raidė didžioji, kitos mažosios. Pavadinimo žodžiai atskiriami juos užrašant iš didžiosios raidės. Tarp žodžių neturi būti nei tarpų, nei kitų skyrybos ženklų.

Modelio pavadinimai užrašomi vienaskaita išskyrus atvejus, kai subjekto pavadinimas neturi vienaskaitos žodžio formos, pavyzdžiui rašom `Pajamos`, nes tokio žodžio kaip `Pajama` nėra.

Modelio pavadinimas turi prasidėti didžiąja raide.

Modelio pavadinimas turi atspindėti duomenų subjekto tipą. Duomnų subjektas yra dalykas turintis pavadinimą ar unikalų identifikatorių. Duomenų subjekto tipas yra subjektų grupė priklausančių tai pačiai kategorijai ar klasei.

Nekartojame vardų erdvės

Modelio pavadinime nekartojamas vardų erdvės, kurioje yra modelis.

Pavyzdys, kaip nereikėtų daryti: `example/planets/EarthPlanet`. Šioje vietoje nereikia kartoti `Planet`, kadangi tai atsispindi vardų erdvės pavadinime `planets`.

10.8.3 Duomenų laukai

Pavyzdys: `snake_case`

Visi duomenų lauko žodžiai rašomi mažosiomis raidėmis, atskiriami pabraukimo ženklu `_`.

Duomenų lauko pavadinimas turi prasidėti mažąja raide.

Ryšiai tarp modelių

`ref` tipo laukai rašomi be `id` ar `_id` galūnės, kadangi jis yra perteklinis.

`ref` tipo laukai atspindi ne konkretų identifikatorių, o visą objektą. Konkretus identifikatorius yra rezervuotas pavadinimas ir duomenų struktūros apraše nenurodomas.

Pavyzdžiui vietoje `country_id`, kurio tipas yra `ref`, reikėų rašyti `country`.

m	property	type	ref
Country			
	name@lt	text	
City			
	name@lt	text	
	country	ref	Country

Tais atvejais, kai duomenys yra denormalizuoti, duomenų lauko pavadinimas užrašomas su tašku, nurodant duomenų lauką iš siejamo modelio. Plačiau apie tai *Denormalizuoti duomenys*.

Nekartojame modelio pavadinimo

Visi modelio duomenų laukai yra konkretaus modelio laukai, todėl nereikia kartoti duomenų laukuose modelio pavadinimo, pavyzdžiui vietoje tokių pavadinimų:

m	property
City	
	city_id
	city_name

Reikėtų rašyti taip:

m	property
City	
	id
	name

Jei kiti modeliai siejami su `City`, tada nurodant tarkim `city_name` iš kito modelio, reikėtų rašyti `city.city_name`. Todėl `city.name` yra aiškesnis pavadinimas, kuriame nesikartoja modelio pavadinimas.

Nekartojame duomenų tipo pavadinimo

Duomenų lauko pavadinime nereikia kartoti duomenų tipo pavadinimo.

Pavyzdžiui taip nereikėtų daryti:

m	property	type
City		
	founded_date	date

Reikėtų rašyti taip:

m	property	type
City		
	founded	date

Nėra prasmės kartoti duomenų tipo, lauko pavadinime.

10.9 Matavimo vienetai

10.9.1 Apibrėžtis laike

`date` ir `datetime` tipo duomenų laukams gali būti žymimas ir *laiko* duomenų tikslumas, pavyzdžiui:

d	r	b	m	property	type	ref	level
datasets/example							
			Matavimas			id	
				id	integer		4
				laikas	datetime	1S	4
				vieta	geometry(point, 4326)	1m	4

Šiuo atveju, nurodyta, kad laukas `laikas` yra 1 sekundės tikslumu, o `vieta` 1 metro tikslumu.

Žymint laiko tikslumą, galite naudoti tokius sutartinius simbolius (atkreipkite dėmesį, kad šie vienetai veikia tik su `date` ir `datetime` tipais):

Simbolis	Prasmė
Y	Metai
Q	Metų ketvirčiai
M	Mėnesiai
W	Savaitės
D	Dienos
H	Valandos
T	Minutės
S	Sekundės
L	Milisekundės
U	Mikrosekundės
N	Nanosekundžės

10.9.2 Apibrėžtis erdvėje

`geometry` tipo duomenų laukams galibūti žymimas *erdvinių* duomenų tikslumas, pavyzdžiui:

Simbolis	Prasmė
nm	Nanometrai (10^9 m)
mm	Milimetrai (10^3 m)
cm	Centimetrai (10^2 m)
m	Metrai
km	Kilometrai (10^3 m)

10.9.3 Kokybiniai duomenys

Kokybiniai duomenys skirstomi į dvi kategorijas:

- pavadinimai ir identifikatoriai
- kategoriniai duomenys

Pavadinimai ir identifikatoriai *`property.ref`* stulpelyje neturi jokio žymėjimo ir jiems suteikiamas `4` brandos lygis.

Kategoriniai duomenys žymini naudojant papildomą `enum` dimensiję, kurioje išvardinamos visos galimos kategorinių duomenų reikšmės.

Jei kategoriniai duomenys yra paliginami, pavyzdžiui:

- puikiai
- gerai
- vidutiniškai
- blogai

Tada, tokiems duomenims, turi būti naudojamas `integer` tipas, kad nebūtų prarastos palyginamosios savybės.

Jei duomenys yra nepalyginami, pavyzdžiui:

- raudona
- geltona
- žalia
- mėlyna

tada nebūtina naudoti `integer` tipą.

10.9.4 Kiekybiniai duomenys

Matavimo vienetai naudojant SI simbolius, išvestinius SI simbolius ir simbolius patvirtintus naudojimui su SI, pateikiami *`property.ref`* stulpelyje.

Pateikus vienetus, laukui gali būti suteikiamas 4-as brandos lygis.

Pavyzdys:

d	r	b	m	property	type	ref	level
datasets/example							
			Matavimas			id	
				id	integer		4
				temperatura	number	°C	4
				svorlis	number	kg	4
				plotas	number	m^2	4
				turis	number	m^3	4
				greitis	number	km/h	4

Vienetai užrašomi naudojant matematinę notaciją, kurioje galima naudoti skaičius, daugybos ir dalybos simbolius, kėlimą laipsniu ir atskirų vienetų sudėtį:

Žymėjimas	Reikšmė
. *	Daugyba
/	Dalyba
(tarpas)	Sudėtis
^(+-)(skaičius) arba $^{0\,1\,2\,3\,4\,5\,6\,7\,8\,9}$	Kėlimas laipsniu

Pavyzdžiai:

m
1m
10m
m^2
m^2
km^{10}
$kgm^2s^3A^1$
kg*m^2*s^-3A^-1
$8kgm^2s^3A^1$
mg/l
g/m^2
mg/m^3

mm
U/m^2
U/m^3
‰
ha
min
h
bar
U
10^6s
10^6s
/m^3
yr
3mo
yr 2mo 4wk
°C
°

Prefiksai

Kiekybiniai matavimo vienetai gali turėti tokius prefiksus:

Žymėjimas	10^n	Priešdėlis
Y	10^{24}	yotta
Z	10^{21}	zetta
E	10^{18}	exa
P	10^{15}	peta
T	10^{12}	tera
G	10^{9}	giga
M	10^{6}	mega
k	10^{3}	kilo
h	10^{2}	hecto
da	10^{1}	deca
d	10^{-1}	deci
c	10^{-2}	centi
m	10^{-3}	milli
µ	10^{-6}	micro
n	10^{-9}	nano
p	10^{-12}	pico
f	10^{-15}	femto
a	10^{-18}	atto
z	10^{-21}	zepto
y	10^{-24}	yocto

Vienetai

Specialiejie vienetai

Žymėjimas	Pavadinimas
U	vienetai (keikis vienetais)
%	procentai

Laiko vienetai

Naudojami tik tais atvejais, kai matuojamas laiko kiekis, o ne data ir laikas. Datos ir laiko (`date` ir `datetime` tipai) tikslumui žymėti, naudojamos kitos žymės.

Žymėjimas	Pavadinimas
s	sekundė
min	minutė
h	valanda
d	diena (24 valandos)
wk	savaitė (7 dienos)
mo	mėnuo (28-31 diena arba 4 savaitės)
yr	metai (354.37 dienos arba 12 mėnesių)

SI Baziniai vienetai

Žymėjimas	Pavadinimas
m	metre
g	gram
s	second
A	ampere
K	kelvin
mol	mole
cd	candela

SI Išvestiniai vienetai

Žymėjimas	Pavadinimas
Hz	hertz
rad	radian
sr	steradian
N	newton
Pa	pascal
J	joule
W	watt
C	coulomb
V	volt
F	farad
	ohm
S	siemens
Wb	weber
T	tesla
H	henry
°C	degree Celsius
lm	lumen
lx	lux
Bq	becquerel
Gy	gray
Sv	sievert
kat	katal

Kiti vienetai

Žymėjimas	Pavadinimas
au	astronomical unit
°	degree
	arcminute
	arcsecond
ha	hectare
l	litre
L	litre
t	tonne
Da	dalton
eV	electronvolt
Np	neper
B	bel
dB	decibel
Gal	gal (acceleration)
u	unified atomic mass unit
var	volt-ampere reactive
pc	parsec
c_0 arba c_0	natural unit of speed
	natural unit of action

continues on next page

Table 1 – tęsinys iš praeito puslapio

Žymėjimas	Pavadinimas
m arba m_e	natural unit of mass
e	atomic unit of charge
a_0 arba a_0	atomic unit of length
E_h	atomic unit of energy
M	nautical mile
kn	knot
Å	ångström
a	are
b	barn
bar	bar
atm	standard atmosphere
Ci	curie
R	roentgen
rem	rem
erg	erg
dyn	dyne
P	poise
st	stokes
Mx	maxwell
G	gauss
Oe	ørsted
sb	stilb
ph	phot
Torr	torr
kgf	kilogram-force
cal	calorie
	micron
xu	x-unit
	gamma (mass, magnetic flux density)
	lambda
Jy	jansky
mmHg	millimetre of mercury

10.10 Ryšiai tarp modelių

Pateikiant metaduomenis apie ryšius tarp modelių, duomenų *brandos lygis* pakeliamas iki ketvirto lygio.

Ryšiai tarp modelių aprašomi tais atvejais, kai vienoje duomenų lentelėje naudojami identifikatoriai iš kitos lentelės.

10.10.1 Jungimas per pirminį raktą

Pavyzdžiui, jei turime tokias dvi duomenų lenteles:

Country		
id	name	code
1	Lietuva	lt
2	Latvija	lv

City		
id	name	country
1	Vilnius	lt
2	Kaunas	lt
3	Ryga	lv

Šiuo atveju, jei norime parengti aukščiau pateiktų duomenų struktūros aprašą, jis atrodytų taip:

id	d	r	b	m	property	type	ref	level
1	datasets/gov/example/countries							
2				Country			code	4
3					id	integer		4
4					name	string		4
5					code	string		4
6				City			id	4
7					id	integer		4
8					name	string		4
9					country	ref	Country	4

Šiame duomenų struktūros apraše, 9-oje eilutėje `country` stulpelio tipas yra `ref`, tai reiškia, kad šis stulpelis yra kito modelio išorinis raktas. `property.ref` stulpelyje nurodyta kurio modelio išorinis raktas šis stulpelis yra. Šiuo atveju, tai yra `Country` modelis, kuris apibrėžtas 2-oje eilutėje.

Pagal nutylėjimą, ryšys su kitu modeliu nustatomas naudojant kitos lentelės pirminį raktą nurodytą *`model.ref`* stulpelyje. Šiuo atveju, `City .country` yra jungiamas per `Country.code`. Tai reiškia, kad `City.country` duomenų tipas turi sutapti su `Country.code` duomenų tipu, kuris yra `string`.

`property.ref` reikšmė gali būti pateikiama vienu iš šių variantų:

`property.ref`

`model`
`model` nurodo kito *`model`* pavadinimą kurio *`model.ref`* siejamas su *`property`*.

Jei *`model.ref`* pirminiam raktui naudoja daugiau nei vieną lauką, tada *`property.source`* laukas turi būti tuščias, o *`property.prepare`* turi būti pateikiamos kableliu atskirtos property reikšmės, kurios bus naudojamos susiejimui.

`model[property]`
Tais atvejais, kai *`property`* duomenys nesutampa su siejamo *`model.ref`*, galima nurodyti *`property`* iš *`model`*.

`model[*property]`
Jei susiejimui reikia daugiau nei vieno duomenų lauko ir jie nesutampa su model.ref, tada galima nurodyti kelias property reikšmes atskirtas kableliu. Tačiau šiuo atveju taip pat būtina nurodyti ir *`property.prepare`* kelias reikšmes atskirtas kableliu, o *`property.source`* reikšmė turi būti tuščia. *`property.`*

prepare stulpelyje nurodomi kiti modelio *property* pavadinimai iš kurių duomenų reikšmių turi būti formuojamas sudėtinis raktas.

10.10.2 Jungimas per nepirminį raktą

Jei modelius reikia jungti ne per pirminį raktą, o per kitus laukus, tada naudojama `model[property]` forma.

Pavyzdžiui, jei turime tokius duomenis:

Country		
id	name	code
1	Lietuva	lt
2	Latvija	lv

City		
id	name	country
1	Vilnius	lt
2	Kaunas	lt
3	Ryga	lv

Kur `Country` pirminis raktas yra `id` ir norime jungti `City.country` per `Country.code`, tuomet duomenų struktūros aprašas atrodys taip:

d	d	r	b	m	property	type	ref	level
1	datasets/gov/example/countries							
2				Country			id	4
3					id	integer		4
4					name	string		4
5					code	string		4
6				City			id	4
7					id	integer		4
8					name	string		4
9					country	ref	Country[code]	4

9-oje eilutėje `property.ref` stulpelyje pateikta `Country[code]` reikšmė, kuri `Country` nurodo su kokiu modeliu jungiame, o `code` nurodo su kokiu `Country` stulpeliu jungiame. Jei pateiktas tik modelis, tada jungiama per to modelio pirminį raktą, jei pateiktas stulpelis laužtiniuose skliausteliuose, tada jungiama per nurodytą stulpelį.

10.10.3 Jungimas per kompozicinį raktą

Jei modelius reikia jungti per kelis laukus, tada naudojama `model[*property]` forma, kur laužtiniuose skliaustuose pateikiami keli stulpeliai atskirti kableliais.

Pavyzdžiui, jei turime tokius duomenis:

Country		
id	name	code
1	Lietuva	lt
2	Latvija	lv

City			
id	name	country	country_id
1	Vilnius	lt	1
2	Kaunas	lt	1
3	Ryga	lv	2

Kur `City` su `Country` yra jungiamas per du `country` ir `country_id` stulpelius, tuomet reikia įtraukti išvestinį duomenų lauką, kuriame formulės įrašomos į *`property.prepare`* pagalba apjungiami keli laukai į vieną kompozicinį raktą. Šiuo atveju duomenų struktūros aprašas atrodys taip:

d	d	r	b	m	property	type	ref	prepare	level
1	datasets/gov/example/countries								
2				Country			id		4
3					id	integer			4
4					name	string			4
5					code	string			4
6				City			id		4
7					id	integer			4
8					name	string			4
9					country_code	string			4
10					country_id	integer			4
11					country	ref	Country[id,code]	country_id, country_code	4

Čia matome, kad 11-oje eilutėje buvo įtrauktas išvestinis laukas `country`, kuris išskaičiuojamas apjungiant `country_id` ir `country_code`. O ryšiui su `Country`, laužtiniuose skliausteliuose nurodyti du laukai iš jungiamo `Country` modelio. Abiejų jungiamų pusių pateiktas laukų sąrašas turi būti vienodo eiliškumo, o jungiami laukai turi turėti vienodus tipus.

Jei `Country` pirminis raktas būtų kompozicinis, pavyzdžiui `id, code`, tuomet, 11-oje eilutėje `property.ref` užtektu nurodyti tik `Country`.

10.10.4 Jungimas atgaliniu ryšiu

Pastaba: Tokio tipo jungimas kol kas dar nėra įgyvendintas.

Jungiant modelius atgaliniu ryšiu kuriamas išvestinis arba virtualus laukas, kuriame analogiškai kaip ir paprasto ryšio atveju, apjungiami du modeliai, tik šiuo atveju kuriamas daug su vienas tipo ryšys.

Pavyzdžiui, jei turime tokius duomenis:

Country	
id	name
1	Lietuva
2	Latvija

City		
id	name	country
1	Vilnius	1
2	Kaunas	1
3	Ryga	2

Tai norint sukurti atgalinį ryšį iš `City` modelio į `Country` modelį, duomenų struktūros aprašas atrodys taip:

d	d	r	b	m	property	type	ref	level
1	datasets/gov/example/countries							
2				Country			id	4
3					id	integer		4
4					name	string		4
5					cities	backref	City	4
6				City			id	4
7					id	integer		4
8					name	string		4
9					country	ref	Country	4

Čia atgalinis ryšys nurodytas 5-oje eilutėje, pateikiant virtualų `Country.cities` lauką, kuris jungiamas per `City.country` lauką, kadangi `City.country` turi ryšį su `Country`.

Jei `City` modelyje būtų pateikti keli stulpeliai susieti su `Country`, tada 5-oje eilutėje `property.ref` reikšmė turėtų nurodyti konkretų lauką, per kurį jungiama, pavyzdžiui `City[country]`.

10.10.5 Polimorfinis jungimas

Pastaba: Tokio tipo jungimas kol kas dar nėra įgyvendintas.

Polimorfinis jungimas yra toks ryšys tarp modelių, kai vieno modelio laukas yra siejamas su daugiau nei vienu kitu modeliu. Tokiam ryšiui nurodyti polimorfinis laukas turi dvi reikšmes, išorinio modelio pavadinimą ir to modelio stulpelio per kurį jungiama reikšmę.

Country	
id	name
1	Lietuva
2	Latvija

City		
id	name	country
1	Vilnius	1
2	Ryga	2

Event			
id	name	object_id	object_model
1	Įkūrimas	1	datasets/gov/example/countries/Country
2	Įkūrimas	2	datasets/gov/example/countries/Country
3	Įkūrimas	1	datasets/gov/example/countries/City
4	Įkūrimas	2	datasets/gov/example/countries/City

Pavyzdyje aukščiau matome, kad yra du modeliai `Country` ir `City`, kuriuos jungia `Event` modelis per `object_id` ir `object_model` laukus. Pavyzdžiui `Event` kurio `id` yra 1, siejamas su `Country` modeliu, kurio `id` yra 1.

Tokių duomenų struktūros aprašas atrodys taip:

d	d	r	b	m	property	type	ref	prepare	level
1	datasets/gov/example/countries								
2				Country			id		4
3					id	integer			4
4					name	string			4
5					cities	backref	City		4
6				City			id		4
7					id	integer			4
8					name	string			4
9					country	ref	Country		4
10				Event			id		4
11					id	integer			4
12					name	string			4
13					object_id	integer			4
14					object_model	string			4
15					object	generic	Country	object_model, object_id	4
16							City		

15-oje eilutėje įtrauktas virtualus `Event.object` laukas, kuris 15-oje ir 16-oje eilutėse, *`property.ref`* stulpelyje išvardina du modelius `Country` ir City`, su kuriais jungiamas šis laukas, per `object_model` ir `object_id` laukus, kurie aprašyti atskirai.

`object_id` ir `object_model` aprašomi atskirai tik todėl, kad duomenys ateina iš išorinio šaltinio. Jie duomenys rašomi tiesiogiai į *Saugyklą*, tada atskirai `generic` laukų apsirašyti nereikia.

10.10.6 Denormalizuoti duomenys

Denormalizuoti duomenų laukai yra tokie laukai, kurie pateikti viename modelyje, tačiau pagal semantinę prasmę priklauso skirtingiems modeliams.

Dažniausiai duomenų normalizavimas atveriant duomenis yra nepageidaujamas ir duomenų struktūra turėtu būti transformuojama į skirtingus modelius, pagal semantinę prasmę. Plačiau apie duomenų normalizavimą galite skaityti skyriuje *Normalizavimas*.

Tačiau tais atvejais, kai vis dėlto norima pateikti duomenis denormalizuotoje formoje, duomenų struktūros apraše galima nurodyti, kurie duomenų laukai yra denormalizuoti.

Pavyzdys, kaip atrodo denormalizuotų duomenų laukų žymėjimas:

d	r	b	m	property	type	ref	level
example							
			Country			code	4
				code	string		4
				name@en	text		4
			City				3
				name@en	text		4
				country	ref	Country	4
				country.code			2
				country.name@en			2
				country.name@lt	text		2

Šiame pavyzdyje turime tokius laukus:

`country` Šis laukas yra `ref` tipo, tai reiškia, kad šiame lauke saugomas `Country` modelio identifikatorius, kurio pagalba `City` galima susieti su `Country`.

`ref` tipo duomenys yra sudėtiniai, tai reiškia, kad per `ref` tipo lauką galima pasiekti siejamo modelio laukus, nurodant kito modelio laukus po taško.

Todėl pagal nutylėjimą `country ref Country` yra tas pats, kas `country._id ref Country`, tik `._id` dalis nenurodoma.

`country.code` ir `country.name@en` Šie laukai yra denormalizuoti, tai reiškia, kad jie priklauso `Country` modeliui, tačiau duomenys yra dubliuojami ir pateikiami dviejose vietose, prie `Country` ir prie `City.country`.

Kadangi `City.country` yra `ref` tipo, tai po taško, galima nurodyti kitus šiam siejamam modeliui priklausančius laukus iš kito modelio.

Atkreipkite dėmesį, kad denormalizuotiems laukams nepildomas `type` stulpelis, kadangi šių laukų tipas turi sutapti su siejamo modelio laukų tipais, taip pat turi sutapti ir laukų pavadinimai.

`country.name@lt` Tais atvejais, kai siejamame modelyje (šiuo atveju `Country` modelyje) nėra tam tikrų laukų, tuomet galima juose pateikti ir prie `City.country`, tačiau tokiu atveju, būtina nurodyti `type`.

10.10.7 Brandos lygis

Apibrėžiant ryšius tarp modelių, brandos lygis įrašomas *`level`* stulpelyje atlieka svarbų vaidmenį. Nuo brandos lygio, priklauso, kaip turi būti interpretuojamas išorinis raktas, siejamas su kitu modeliu.

1 brandos lygis: Susiejimas neįmanomas Duomenys pateikti tokia forma, kurios pagalba dviejų modelių jungimas nėra įmanomas.

Pavyzdžiui, pateikta adreso tekstinė forma, kuri nesutampa su tekstine forma pateikiama oficialiame adresų registre arba naudojamas toks tam tikras identifikatorius, kuris nėra suderintas su siejamo modelio pirminiu raktu.

2 brandos lygis: Susiejimas nepatikimas Duomenys pateikiami tam tikra forma, kuri neužtikrina patikimo duomenų susiejimo, tačiau siejimui atliekamas pagal siejamo modelio atributą, kuris negarantuoja unikalaus objekto identifikavimo.

Pavyzdžiui siejimas daromas pagal pavadinimą, kuris gali keistis arba ne visais atvejais sutampa.

3 brandos lygis: Susiejimas ne per pirminį raktą Duomenims susieti naudojamas patikimas identifikatorius, kuris yra surištas siejamo modelio pirminiu raktu, tačiau naudojamas ne pirminis raktas, o kitas identifikatorius.

4 brandos lygis: Susiejimas per pirminį raktą Susiejimas daromas per pirminį raktą.

Susiejimas neįmanomas

Jei `ref` tipui nurodytas 1 arba žemesnis brandos lygis, tai reiškia, duomenų jungimas nėra įmanomas. Tokiu atveju, atveriant duomenis, `property` įgaus tokį tipą, koks yra lauko su kuriuo siejamas ryšys tipas.

Pavyzdžiui:

d	r	b	m	property	type	ref	level
example							
			Country			name@lt	4
				name@lt	text		4
			City			name	4
				name@lt	text		4
				country	ref	Country	1

Šiuo atveju, `City.country` yra siejamas su `Country.name`. Kadangi `City.country` brandos lygis yra 2, tai reiškia, kad `City.country` ir `Country.name` pavadinimai nesutampa ir jungimo atlikti neįmanoma. Tokiu atveju, `City.country` tipas bus ne `ref`, o toks pat, kaip `Country.name`, t.y. `text`.

Tačiau, metaduomenyse išliks informacija, apie tai, kad šios lentelės yra susijusios, tačiau dėl prasto duomenų brandos lygios, realus susiejimas nėra įmanomas.

Jei modeliai yra susiję, tačiau, tokio duomenų lauko, per kurį galima būtų atlikti susiejimą iš vis nėra, tuomet, tokį lauką galima sukurti, nurodant brandos lygį 0. Pavyzdžiui:

d	r	b	m	property	type	ref	level
example							
			Country			name@lt	4
				name@lt	text		4
				name@en	text		0
			City			name	4
				name@en	text		4
				country	ref	Country[name@en]	1

Šioje vietoje `City.country` tampa `country@en`, kurio tipas yra `text`. O į `Country` yra įtrauktas papildomas laukas `name@en`, per kurį ir atliekamas susiejimas, t.y. per kurį galėtu būti atliktas susiejimas, jei toks laukas egzistuotų ne tik `City.country`, bet ir `Country.name@en`.

Susiejimas nepatikimas

Jei `ref` tipui suteiktas 2 brandos lygis, tai reiškia, kad susiejimas yra įmanomas, tačiau nėra garantijos, kad jis veiks visais atvejais.

Susiejimas laikomas nepatikimu, tada, kai siejimas atliekamas ne patikimo unikalaus identifikatoriaus pagalba, o per pavadinimą ar panašiais būdais.

Pavadinimai gali keistis, gali dubliuotis, gal skirtis jų užrašymo forma, todėl toks jungimas laikomas nepatikimu.

Toks jungimas ir 2 brandos lygio žymėjimas taikomas tik tais atvejais, kai jungimas daromas, per jungiamo modelio atributą. Pavyzdžiui:

d	r	b	m	property	type	ref	level
example							
			Country			name@lt	4
				name@lt	text		4
			City			name	4
				name@lt	text		4
				country	ref	Country	2

Šiuo atveju, kadangi `City.country` brandos lygis yra `2`, tai reiškia, kad `City.country` duomenys yra realiai paimti iš `Country.name@lt`. Jei `City.country` būtų paimti ne iš `Country.name@lt`, o iš kokio nors kito šaltinio ir gali nesutapti, tada brandos lygis turėtu būti `1`.

Tai reiškia, kad `2` brandos lygis žymimas tik tais atvejais, kai išorinis raktas yra paimtas iš siejamo modelio atributo.

Susiejimas ne per pirminį raktą

Jei `ref` tipui suteiktas 3 ar didesnis brandos lygis, vadinasi susiejimas yra patikimas. Duomenys siejami naudojant patikimus unikalius identifikatorius, kurie nesidubliuoja, nesikeičia ir užrašomi visada vienodai.

Dažniausiai patikimais identifikatoriais laikomi sveiki skaičiai, tam tikri sutartiniai kodai ir kiti specializuoti identifikatoriai, tokie kaip UUID.

Tačiau, naudojamas ne pirminis raktas, o kitas duomenų laukas. Pavyzdžiui:

d	r	b	m	property	type	ref	level
example							
			Country			id	4
				id	integer		4
				code	string		4
				name@lt	text		4
			City			name	4
				name@lt	text		4
				country	ref	Country[code]	3

Skirtumas tarp 3 ir 4 brandos lygio iš esmės susijęs su duomenų saugojimu Saugykloje ar kitoje vietoje, kur pirminiai raktai yra generuojami ir jų negalima keisti. Jei naudojamas 3 brandos lygis, tuomet saugykloje saugomas, ne išorinis saugyklos identifikatorius UUID, o vidinis duomenų rinkinio identifikatorius.

Publikuojant duomenis iš tam tikro šaltinio, išoriniai raktas visada turėtu būti konvertuojami į išorinį pirminį raktą, tačiau tais atvejais, jei dėl tam tikrų priežasčių tas nėra daroma, tuomet žymimas 3 brandos lygis ir publikuojami ne išoriniai pirminiai raktai, o šaltinio vidiniai.

Pavyzdžiui, jei turime tokius duomenis:

example/Country			
_id	id	code	name@lt
4dbb1b77-a930-4f2a-8ef4-f05b89f0fcfe	1	lt	Lietuva

Ir jei `City.country` turi brandos lygį 3, tada `City` duomenys atrodys taip:

example/City		
_id	name@lt	country._id
096e054e-7a4c-44cc-8f27-98af815080d5	Vilnius	lt

Susiejimas per pirminį raktą

Šiuo atveju, brandos lygis žymimas `4` ir skirtumas nuo `3` brandos lygio yra toks, kad duomenyse naudojamas išorinis pirminis raktas. Pavyzdžiui:

d	r	b	m	property	type	ref	level
example							
			Country			id	4
				id	integer		4
				code	string		4
				name@lt	text		4
			City			name	4
				name@lt	text		4
				country	ref	Country	4

Turint tokį struktūros aprašą, kur `City.country` brandos lygis yra `4`, duomenys atrodys taip:

example/Country			
_id	id	code	name@lt
4dbb1b77-a930-4f2a-8ef4-f05b89f0fcfe	1	lt	Lietuva

example/City		
_id	name@lt	country._id
096e054e-7a4c-44cc-8f27-98af815080d5	Vilnius	4dbb1b77-a930-4f2a-8ef4-f05b89f0fcfe

Matome, kad `City.country._id` yra `Country` pirminis raktas. Tai reiškia, kad vidiniai duomenų rinkinio raktai konvertuojami į išorinius.

10.11 Brandos lygiai

Duomenų brandos lygis nurodomas *`level`* stulpelyje.

Duomenų brandos lygis atitinka 5 Open Data skalę, tačiau adaptuota duomenų struktūros aprašo kontekstui. Papildomai įtrauktas nulinis lygis, kai duomenys nekaupiami, tačiau yra reikalingi ir yra parengtas jų *duomenų struktūros aprašas*.

Tim Berners Lee brandos lygius aprašo, kaip pavyzdį pasitelkiant duomenų distribucijų formatus. Tačiau duomenų distribucijų formatai yra labai netikslus pavyzdys. Rengiant duomenų struktūros aprašą, brandos lygis vertinamas kiekvienam duomenų laukui atskirai, todėl tarkime CSV failas, gali būti didesnio arba mažesnio nei trečias brandos lygis, priklausomai nuo CSV faile esančių duomenų turinio. Tačiau grubiai vertinant, vidutiniškai CSV failai turi daugiau ar mažiau 3 brandos lygį, koks ir yra nurodytas Tim Berners Lee pavyzdžiuose.

Taip pat reikėtų atkreipti dėmesį, kad duomenų brandos lygis ar formatas nėra susijęs su duomenų atvirumu. Duomenys gali būti pateikti aukščiausiu 5 brandos lygiu, tačiau pats duomenų prieinamumas gali būti visiškai uždaras, siauram naudotojų ratui, kurie turi ribotą ir saugų prieigos kanalą prie duomenų.

Rengiant duomenų struktūros aprašą, reikėtų nurodyti ne šaltinio duomenų brandos lygį, o galutinį brandos lygį, kuris yra gaunamas atlikus visas duomenų struktūros apraše nurodytas transformacijas.

level

0

Nekaupiama

Duomenys nekaupiami. Duomenų rinkinys užregistruotas atvirų duomenų portale. Užpildyta `dataset` eilutė.

Plačiau apie brandos lygio kėlimą skaitykite skyriuje *Nekaupiami duomenys*.

Pavyzdžiai

Imone		
imones_id	imones_pavadinimas	rusis
42	UAB "Įmonė"	1

Filialas				
ikurimo_data	atstumas	imones_id	imones_pavadinimas._id	tel_nr

Struktūros aprašas							
d	r	b	m	property	type	ref	level
example							
			Imone			imones_id	4
				imones_id	integer		4
				imones_pavadinimas	string		2
				rusis	integer		2
			Filialas				0
				ikurimo_data	string		0
				atstumas	string		0
				imone	ref	Imone	0
				tel_nr	string		0

- **Duomenų nėra** - nuliniu brandos lygiu žymimi duomenys, kurių nėra, pavyzdžiui jei įmonės filialų duomenų niekas nefiksuoja.
- **Duomenys nepublikuojami** - nuliniu brandos lygiu žymimi duomenys, kurie fiziškai egzistuoja, tačiau nėra publikuojami jokia forma.
- **Ribojamas duomenų naudojimas** - nuliniu brandos lygiu žymimi duomenys, kuri yra prieinami, tačiau pagal tokias naudojimo sąlygas, kurios nėra suderinamos su atvirų duomenų licencijomis.

1

Publikuojama

Duomenys publikuojami bet kokia forma. Užpildyta `resource` eilutė.

Plačiau apie brandos lygio kėlimą skaitykite skyriuje *Duomenys be aiškios struktūros*.

Pirmu brandos lygiu žymimi duomenų laukai, kurių reikšmės neturi vientisumo, tarkime ta pati reikšmė gali būti pateikta keliais skirtingais variantais.

Pavyzdžiai

Imone		
imones_id	imones_pavadinimas	rusis
42	UAB "Įmonė"	1

Filialas				
ikurimo_data	atstumas	imones_id._id	imo-nes_pavadinimas	tel_nr
vakar	1 m.	1	Įmonė 1	+370 345 36522
2021 rugpjūčio 1 d.	1 m	1	UAB Įmonė 1	8 345 36 522
1/9/21	1 metras	1	Įmonė 1, UAB	(83) 45 34522
21/9/1	0.001 km	1	„ĮMONĖ 1“, UAB	037034536522

Struktūros aprašas							
d	r	b	m	property	type	ref	level
example							
			Imone			id	4
				imones_id	integer		2
				imones_pavadinimas	string		2
				rusis	integer		2
			Filialas				3
				ikurimo_data	string		1
				atstumas	string		1
				imones_id	ref	Imone	1
				imones_pavadinimas	string		1
				tel_nr	string		1

- **Neaiški struktūra** - pirmu brandos lygiu žymimi duomenys, kuriuose nėra aiškios struktūros, pavyzdžiui `ikurta` datos formatas nėra vienodas, kiekviena data užrašyta vis kitokiu formatu.
- **Nėra vientisumo** - pirmu brandos lygiu žymimi duomenuys, kuruose nėra vientisumo, pavyzdžiui `atstumas` užrašytas laikantis tam tikros struktūros, tačiau skirtingais vienetais.
- **Neįmanomas jungimas** - pirmu brandos lygiu žymimi duomenys, kurių neįmanoma arba sudėtinga sujungti. Pavyzdžiui `Filialas` duomnų laukas `imone` naudoja tam tikrą identifikatorių, kuris nesutampa nei su vienu iš `Imone` atributų, pagal kuriuose būtų galima identifikuoti filialo įmonę.

2

Struktūruota

Duomenys kaupiami struktūruota, mašininiu būdu nuskaitoma forma, bet kokiu formatu. Užpildytas *`property.source`* stulpelis.

Plačiau apie brandos lygio kėlimą skaitykite skyriuje *Nestandartinio formato duomenys*.

Antru brandos lygiu žymimi duomenų laukai, kurie pateikti vieninga forma arba pagal aiškų ir vienodą šabloną. Tačiau pateikimo būdas nėra standartinis. Nestandartinis duomenų formatas yra toks, kuris neturi viešai skelbiamos ir atviros formato specifikacijos arba kuris nėra priimtas kaip standartas, kurį prižiūri tam tikra standartizacijos organizacija.

Pavyzdžiai

Imone		
imones_id	imones_pavadinimas	rusis
42	UAB "Įmonė"	1

Filialas				
ikurimo_data	atstumas	imones_id	imones_pavadinimas._id	tel_nr
1/9/21	1 m.	1	UAB "Įmonė"	(83) 111 11111
2/9/21	2 m.	1	UAB "Įmonė"	(83) 222 22222
3/9/21	3 m.	1	UAB "Įmonė"	(83) 333 33333
4/9/21	4 m.	1	UAB "Įmonė"	(83) 444 44444

Struktūros aprašas							
d	r	b	m	property	type	ref	level
example							
			JuridinisAsmuo			kodas	4
				kodas	integer		4
				pavadinimas@lt	text		4
			Imone			imones_id	2
				imones_id	integer		2
				imones_pavadinimas	string		2
				rusis	integer		2
			Filialas				3
				ikurimo_data	string		2
				atstumas	string		2
				imones_id	integer		2
				imones_pavadinimas	string		2
				tel_nr	string		2

- **Nestandartiniai duomenų tipai** - antru brandos lygiu žymimi duomenys, kurių nurodytas tipas neatitinka realaus duomenų tipo. Pavyzdžiui:
  - `ikurimo_data` - nurodytas `string`, turėtu būti `date`.
  - `imones_pavadinimas` - nurodytas `string`, turėtu būti `text`.
  - `atstumas` - nurodytas `string`, turėtu būti `integer`.
- **Nestandartinis formatas** - antru brandos lygiu žymimi duomenys, kurie pateikti nestandartiniu formatu. Standartinis duomenų pateikimas nurodytas prie kiekvieno duomenų tipo skyriuje *Duomenų tipai*. Payvzdžiui:
  - `ikurimo_data` - nurodytas `DD/MM/YY`, turėtu būti `YYYY-MM-DD`.
  - `atstumas` - nurodyta `X m.`, turėtu būti `X`.
  - `tel_nr` - nurodytas `(XX) XXX XXXXX`, turėtu būti `+XXX-XXX-XXXXX`.
- **Nestandartiniai kodiniai pavadinimai** - antru brandos lygiu žymimi duomenys, kurių kodiniai pavadinimai, neatitinka *standartinių reikalavimų keliamų kodiniams pavadinimams*. Pavyzdžiui:
  - `imones_id` - dubliuojamas modelio pavadinimas, turėtu būti `id`.
  - `imones_pavadinimas` - dubliuojamas modelio pavadinimas, turėtu būti `pavadinimas`.
  - `ikurimo_data` - dubliuojamas tipo pavadinimas, turėtu būti `ikurta`.

- **Nepatikimi identifikatoriai** - antru brandos lygiu žymimi duomenys, kurių `ref` tipui naudojami nepatikimi identifikatoriai, pavyzdžiui tokie, kaip pavadinimai, kurie gali keistis arba kartotis. Pavyzdžiui:
  - `imones_pavadinimas` - jungimas daromas per įmonės pavadinimą, tačiau šiuo atveju kito varianto nėra, nes `Filialas.imones_id` nesutampa su `Imone.imones_id`.
- **Denormalizuoti duomenys** - antru brandos lygiu žymimi duomenys, kurie dubliuoja kito modelio duomenis ir yra užrašyti nenurodant, kad tai yra duomenys dubliuojantys kito modelio duomenis. Pavyzdžiui:
  - `Filialas.imones_id` turėtu būti `Filialas.imone.imones_id`.
  - `Filialas.imones_pavadinimas` turėtu būti `Filialas.imone.imones_pavadinimas`.

  Plačiau apie denormalizuotus duomenis skaitykite skyriuje *Denormalizuoti duomenys*.
- **Nenurodytas susiejimas** - antru brandos lygiu žymimi duomenys, kurie siejasi su kitu modeliu, tačiau tokia informacija nėra pateikta metaduomenyse. Pavyzdžiui:
  - `Filialas.imone` - `Filialas` siejasi su `Imone`, per `Filialas.imones_pavadiniams`, todėl turėtu būti nurodytas `imone ref Imone` ryšys su `Imone`.
- **Neatitinka modelio bazės** - antru brandos lygiu žymimi duomenys, kurie priklauso vienai semantinei klasei, tačiau duomenų schema nesutampa su bazinio modelio schema. Pavyzdžiui:
  - `Imone` - priklauso semantinei klasei `JuridinisAsmuo`, tačiau tai nėra pažymėta metaduomenyse.
  - `Imone.imones_id` turėtu būti `Imone.kodas`, kad sutaptu su baze (`JuridinisAsmuo.kodas`).
  - `Imone.imones_pavadinimas` turėtu būti `Imone.pavadinimas@lt`, kad sutaptu su baze (`JuridinisAsmuo.pavadinimas@lt`).
- **Nenurodytas enum kodinėms reikšmėms** - antru brandos lygiu žymimi kategoriniai duomenys, kurių reikšmės pateiktos sutartiniais kodinais, kurių prasmė nėra aiški. Pavyzdžiui:
  - `Imone.rusis` - įmonės rūšis žymima skaičiais, tačiau nėra aišku, kokks skaičius, ką rei6kia, todėl reikia pateitki `enum` sąrašą, kuriame būtų nurodyta, ką koks skaičius reiškia. Plačiau skaityti *Klasifikatoriai*.

3

Standartizuota

Duomenys saugomi atviru, standartiniu formatu. Užpildytas `property.type` stulpelis ir duomenys atitinka nurodytą tipą.

Plačiau apie brandos lygio kėlimą skaitykite skyriuje *Duomenys be metaduomenų*.

Trečias brandos lygis suteikiamas tada, kai duomenys pateikti vieninga forma, vieningu masteliu, naudojamas formatas yra standartinis, tai reiškia, kad yra viešai skelbiama ir atvira formato specifikacija arba pats formatas yra patvirtintas ir prižiūrimas kokios nors standartizacijos organizacijos.

Pavyzdžiai

Imone		
id	pavadinimas@lt	rusis
42	UAB "Įmonė"	juridinis

Filialas				
ikurta	atstumas	imone._id	imone.pavadinimas@lt	tel_nr
2021-09-01	1	42	UAB "Įmonė"	+37011111111
2021-09-02	2	42	UAB "Įmonė"	+37022222222
2021-09-03	3	42	UAB "Įmonė"	+37033333333
2021-09-04	4	42	UAB "Įmonė"	+37044444444

Struktūros aprašas									
d	r	b	m	property	type	ref	level	prepare	title
example									
			JuridinisAsmuo			kodas	4		
				kodas	integer		4		
				pavadinimas@lt	text		4		
		JuridinisAsmuo					4		
			Imone			kodas	4		
				kodas			4		
				pavadinimas@lt			4		
				rusis	string		3		
		/							
			Filialas				3		
				ikurta	date		3		
				atstumas	integer		3		
				imone	ref	Imone	3		
				imone.kodas			4		
				imone.pavadinimas@lt			4		
				tel_nr	string		4		

- **Nenurodytas pirminis raktas** - trečiu brandos lygiu žymimi duomenys, kurie neturi nurodyto pirminio rakto *model.ref* stulpelyje. Pavyzdžiui:
  - `Filialas` - nenurodytas pirminis raktas *model.ref* stulpelyje.
- **Nenurodyt vienetai** - trečiu brandos lygiu žymimi kiekybiniai duomenys, kuriems nėra nurodyti matavimo vienetai *property.ref* stulpelyje. Pavyzdžiui:
  - `atstumas` - nenurodyta, kokiais vienetais matuojamas atstumas.

- **Nenurodyti tikslumas** - trečiu brandos lygiu žymimi laiko ir erdviniai duomenys, kuriems nėra nurodytas matavimo tikslumas. Matavimo tikslumas nurodomas `property.ref` stulpelyje. Pavyzdžiui:
  - `ikurta` - nenurodytas datos tikslumas, turėtu būti `D` - vienos dienos tiksumas.
- **Siejimas ne per priminį raktą** - trečiu brandos lygiu žymimi `ref` tipo duomenų laukai, kurie siejami ne per perminį raktą `_id`, o per kitą identifikatorių. Pavyzdžiui:
  - `Filialas.imone` - siejimas atliekamas per `Imone.kodas`, o ne per `Imone._id`.
- **Neaprašyti kategoriniai duomenys** - trečiu brandos lygiu žymimi kategoriniai duomenys, kurių reikšmės pačios savaime yra aiškios, tačiau neišvardintos struktūros apraše. Pavyzdžiui:
  - `Imone.rusis` - įmonės rūšies kategorijos duomenys yra pateikta tekstine forma, tačiau, struktūros apraše nėra išvardintos visos galimos kategorijos ir pats duomenų laukas nėra pažymėtas, kaip kategorinis.

4

Identifikuojama

Duomenų objektai turi aiškius, unikalius identifikatorius. Užpildyti *`model.ref`* ir *`property.ref`* stulpeliai.

Pastaba: *`property.ref`* stulpelis pildomas šiais atvejais:

- Jei duomenų laukas yra išorinis raktas (žiūrėti *Išoriniai raktai*).
- Jei duomenų laukas yra kiekybinis ir turi matavimo vienetus (žiūrėti *Matavimo vienetai*).
- Jei duomenų laukas žymi laiką ar vietą (žiūrėti *Data ir laikas* ir *Erdviniai duomenys*).

Plačiau apie brandos lygio kėlimą skaitykite skyriuje *Duomenys be identifikatorių*.

Ketvirtas duomenų brandos lygis labiau susijęs ne su pačių duomenų formatu, bet su metaduomenimis, kurie lydi duomenis.

Duomenų struktūros apraše *`model.ref`* stulpelyje, pateikiamas objektą unikaliai identifikuojančių laukų sąrašas, o *`property.type`* stulpelyje įrašomas `ref` tipas, kuris nurodo ryšį tarp dviejų objektų.

Pavyzdžiai

Imone			
_id	id	pavadinimas@lt	rusis
26510da5-f6a6-45b0-a9b9-27b3d0090a58	42	UAB "Įmonė"	1

Filialas							
_id	id	ikurta	atstumas	imone._id	imone.id	imone.pavadinimas	tel_nr@lt
63161bd2-158f-4d62-9804-636573abb9c7	1	2021-09-01	1	26510da5-f6a6-45b0-a9b9-27b3d0090a58	42	UAB "Įmonė"	+37011111111
65ec7208-fb97-41a8-9cfc-dfedd197ced6	2	2021-09-02	2	26510da5-f6a6-45b0-a9b9-27b3d0090a58	42	UAB "Įmonė"	+37022222222
2b8cdfa6-1396-431a-851c-c7c6eb7aa433	3	2021-09-03	3	26510da5-f6a6-45b0-a9b9-27b3d0090a58	42	UAB "Įmonė"	+37033333333
1882bb9e-73ee-4057-b04d-d4af47f0aae8	4	2021-09-04	4	26510da5-f6a6-45b0-a9b9-27b3d0090a58	42	UAB "Įmonė"	+37044444444

Struktūros aprašas									
d	r	b	m	property	type	ref	level	prepare	title
example									
			JuridinisAsmuo			kodas	4		
				kodas	integer		4		
				pavadinimas@lt	text		4		
		JuridinisAsmuo					4		
			Imone			kodas	4		
				id	integer		4		
				pavadinimas@lt	text		4		
				rusis	integer		4		
					enum			1	Juridinis
								2	Fizinis
		/							
			Filialas			id	4		
				id	integer		4		
				ikurta	date	D	4		
				atstumas	integer	km	4		
				imone	ref	Imone	4		
				imone.id			4		
				imone.pavadinimas@lt			4		
				tel_nr	string		4		

- **Nesusieta su standartiniu žodynu** - ketvirtu brandos lygiu žimimi duomenys, kurie nėra susieti su standartiniais žodynais ar ontologijomis. Siejimas su žodynais atliekamas `model.uri` ir `property.uri` stulpeluose.

5

Susieta su išoriniu žodynu

Modeliai iš įstaigų duomenų rinkinių vardų erdvės susieti su modeliais iš standartų vardų erdvės, užpildytas `base` eilutė. Standartų vardų erdvėje esantiems *modeliams* ir jų *savybėms* užpildytas `uri` stulpelis.

Daugiau apie vardų erdves skaitykite skyrelyje: *Vardų erdvės*.

Plačiau apie brandos lygio kėlimą skaitykite skyriuje *Duomenys apibrėžti nestandartiniu žodynu*.

Penkto brandos lygio duomenys yra lygiai tokie patys, kaip ir ketvirto brandos lygio, tačiau penktame brandos lygyje, duomenys yra praturtinami metaduomenimis, pateikiant nuorodas į išorinius žodynus arba bend jau pateikiant aiškius pavadinimus ir aprašymus, užpildant `title` ir `description` stulpelius.

Penktame brandos lygyje visas dėmesys yra sutelkiamas yra semantinę duomenų prasmę.

Pavyzdžiai

Imone			
_id	id	pavadinimas@lt	rusis
26510da5-f6a6-45b0-a9b9-27b3d0090a58	42	UAB "Įmonė"	1

Filialas							
_id	id	ikurta	atstumas	imone._id	imone.id	imone.pavadinimas@lt	tel_nr
63161bd2-158f-4d62-9804-636573abb9c7	1	2021-09-01	1	26510da5-f6a6-45b0-a9b9-27b3d0090a58	42	UAB "Įmonė"	tel:+37011111111
65ec7208-fb97-41a8-9cfc-dfedd197ced6	2	2021-09-02	2	26510da5-f6a6-45b0-a9b9-27b3d0090a58	42	UAB "Įmonė"	tel:+37022222222
2b8cdfa6-1396-431a-851c-c7c6eb7aa433	3	2021-09-03	3	26510da5-f6a6-45b0-a9b9-27b3d0090a58	42	UAB "Įmonė"	tel:+37033333333
1882bb9e-73ee-4057-b04d-d4af47f0aae8	4	2021-09-04	4	26510da5-f6a6-45b0-a9b9-27b3d0090a58	42	UAB "Įmonė"	tel:+37044444444

Struktūros aprašas										
d	r	b	m	property	type	ref	level	uri	prepare	title
example										
					prefix	foaf		http://xmlns.com/foaf/0.1/		
						dct		http://purl.org/dc/terms/		
						schema		http://schema.org/		
			JuridinisAsmuo			kodas	4			
				kodas	integer		4			
				pavadinimas@lt	text		4			
		JuridinisAsmuo					4			
			Imone			id	5	foaf:Organization		
				id			5	dct:identifier		
				pavadinimas@lt			5	dct:title		
				rusis	integer		4			
					enum				1	Juridinis
									2	Fizinis
		/								
			Filialas			id	5	schema:LocalBusiness		
				id	date	1D	5	dct:identifier		
				ikurta	date	1D	5	dct:created		
				atstumas	integer	km	5	schema:distance		
				imone	ref	Imone	5	foaf:Organization		
				imone.id	integer		5	dct:identifier		
				imone.pavadinimas@lt	text		5	dct:title		
				tel_nr	string		5	foaf:phone		

10.12 Prieigos lygiai

Duomenų prieigos lygis nurodomas *access* stulpelyje.

access

private
Vidiniam naudojimui

Duomenys skirti tik vidiniam konkrečios sistemos naudojimui.

protected
Pakartotiniam naudojimui

Duomenys gali būti naudojami integracijai su išorinėmis sistemomis.

public
Viešam naudojimui

Duomenys skirti viešam naudojimui, tačiau duomenų panaudojimo tikslai ribojami.

open
Atviram naudojimui

Duomenys skirti viešam naudojimui, neribojant panaudojimo tikslo.

Viešam pakartotiniam naudojimui gali būti teikiami tik `public` ir `open` prieigos lygio duomenys.

`public` duomenys gali būti teikiami tik autorizuotiems duomenų valdytojams, kurie yra susipažinę ir sutinka su duomenų naudojimo taisyklėmis ir naudoja duomenis tik nurodytu tikslu (*purpose limitation*), laikosi BDAR reikalavimų. Asmens duomenys gali būti viešinami tik public ar žemesniu prieigos lygiu.

`open` duomenys turėtu būti teikiami atvirai be jokios autorizacijos ir neribojant duomenų naudojimo tikslo. Asmens duomenys negali būti teikiami `open` prieigos lygiu.

Prieigos lygiai gali būti paveldimi iš aukštesneės dimensijos. Tačiau žemesnė dimensija apsprendžia realų prieigos lygį. Pavyzdžiui jei *dataset.access* yra `private`, o toje *dataset* dimensijoje esantis *property* yra `open`, tada visos to *property* aukštesnės dimensijos taip pat tampa `open`, nors visos kitos dimensijos yra `private`, nes paveldi *dataset.access* reikšmę.

10.13 Duomenų šaltiniai

10.13.1 SQL

resource.source
Duomenų bazės URI. Duomenų bazės URI formuojamas naudojant tokį ABNF šabloną:

```
uri = type ["+" driver] "://"
      [user [":" password] "@"]
      host [":" port]
      "/" database ["?" params]
```

Šablone naudojamų kintamųjų aprašymas:

type
Duomenų bazių serverio pavadinimas:

sqlite

`postgresql`

`mysql`

`oracle`

`mssql`

`driver`
Konkretaus duomenų bazių serverio tvarkyklė naudojama komunikacijai su duomenų baze.

`user`
Naudotojo vardas jungimuisi prie duomenų bazės.

`password`
Duomenų bazės naudotojo slaptažodis.

`host`
Duomenų bazių serverio adresas.

`port`
Duomenų bazių serverio prievadas.

`database`
Konkrečios duomenų bazės pavadinimas.

`params`
Papildomi parametrai Query string formatu.

`resource.prepare`
Formulė skirta papildomiems veiksmams reikalingiems ryšiui su duomenų baze užmezgimui ir duomenų bazės paruošimui, kad būtų galima skaityt duomenis.

`resource.type`
Galimos reikšmės: *sql*.

`resource.prepare`
`connect`(*dsn*, *schema: str = None*, *encoding: str = 'utf-8'*)

Parametrai

- `dsn` -- Duomenų bazės URI, kaip nurodyta *resource.source*.
- `schema` -- Duomenų bazės schema.
- `encoding` -- Duomenų bazės koduotė.

Naudojama tais atvejais, kai jungiantis prie duomenų bazės reikia perduoti papildomus parametrus.

`model.source`
Duomenų bazėje esančios lentelės pavadinimas.

`property.source`
Lentelės stulpelio pavadinimas.

10.13.2 CSV

resource.type
Galimos reikšmės: *csv*, *tsv*.

resource.source
Žiūrėti *Failai*.

resource.prepare
tabular(*sep: ','*)
Nurodoma kaip CSV faile atskirti stulpeliai. Pagal nutylėjimą *separator* reikšmė yra ,.

model.source
Nenaudojama, kadangi CSV resursas gali turėti tik vieną lentelę.

model.prepare
Žiūrėti *Stulpeliai lentelėje*.

property.source
Žiūrėti *Stulpeliai lentelėje*.

10.13.3 JSON

resource.type
Galimos reikšmės: *json*, *jsonl*.

resource.source
Žiūrėti *Failai*.

model.source
JSON objekto savybės pavadinimas, kuri rodo į masyvą reikšmių, kurios bus naudojamos kaip modelio duomenų eilutės. Kiekvienas masyvo elementas atskirai aprašomas `property` dimensijoje. Jei JSON objektas yra kompleksinis žiūrėti *Kompleksinės struktūros*.

property.source
JSON objekto savybė, kurioje pateikiami aprašomo stulpelio duomenys.

property.prepare
Žiūrėti *Kompleksinės struktūros*.

10.13.4 XML

resource.type
Galimos reikšmės: *xml*, *html*.

resource.source
Žiūrėti *Failai*.

model.source
XPath iki elementų sąrašo kuriame yra modelio duomenys.

model.prepare
Jei neužpildyta, vykdoma `xpath(self)` funkcija.

xpath(*expr*)
Vykdo nurodyta *expr*, viso XML dokumento kontekste.

property.source
: XPath iki elemento kuriame yra duomenys.

model.prepare
: Jei neužpildyta, vykdoma `xpath(self)` funkcija, iš *model* gauto elemento kontekste.

10.13.5 XLSX

resource.type
: Galimos reikšmės: *xlsx*, *xls* arba *odt*.

resource.source
: Žiūrėti *Failai*.

model.source
: Skaičiuoklės faile esančio lapo pavadinimas.

model.prepare
: Žiūrėti *Stulpeliai lentelėje*.

property.source
: Žiūrėti *Stulpeliai lentelėje*.

10.14 Išoriniai žodynai

Išorinių žodynų pagalba, galima susieti aprašomus duomenis su išoriniais žodynais. Susiejimas atliekamas *model.uri* ir *property.uri*, naudojant *išorinių žodynų URI prefiksus*.

Pavyzdžiui turint tokį duomenų šaltinį:

country	
id	name
1	Lietuva

city		
id	name	country
1	Vilnius	1

Ir šį šaltinį atitinkančią *DSA* lentelę:

id	d	r	b	m	property	type	ref	uri
						prefix	locn	http://www.w3.org/ns/locn#
							dbpedia-owl	http://dbpedia.org/ontology/
	datasets/example/geo							
		salys				sql		
				Country			id	locn:Location
					id	integer		dct:identifier
					name@lt	text		locn:geographicName
					capital	ref	City	dbpedia-owl:capital
				City			id	locn:Location
					id	integer		dct:identifier
					name@lt	text		locn:geographicName
					country	ref	Country	dbpedia-owl:country

Jei *property* pavadinimai turi @ žymes, tada generuojant RDF, prie reikšmės pridedama atitinkama kalbos žymė.

Galima duomenis eksportuoti RDF Turtle formatu, kas atrodytų taip:

```
@base <https://get.data.gov.lt/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix locn: <http://www.w3.org/ns/locn#> .
@prefix dbpedia-owl: <http://dbpedia.org/ontology/> .

<datasets/example/geo/Country/eb09946c-26e1-4698-a298-7bb1e468b165>
    a locn:Location ;
    dct:identifier 1 ;
    locn:geographicName "Lietuva"@lt ;
    dbpedia-owl:capital <datasets/example/geo/City/b54c21f6-08b8-4bdd-b785-be1cb2e93a98>␣
↪.

<datasets/example/geo/City/b54c21f6-08b8-4bdd-b785-be1cb2e93a98>
    a locn:Location ;
    dct:identifier 1 ;
    locn:geographicName "Vilnius"@lt ;
    dbpedia-owl:country <datasets/example/geo/Country/eb09946c-26e1-4698-a298-
↪7bb1e468b165> .
```

Analogiškai, tie patys duomenys gali būti eksportuojami RDF/XML formatu:

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
    xml:base="https://get.data.gov.lt/"
    xmlns:dct="http://purl.org/dc/terms/"
    xmlns:locn="http://www.w3.org/ns/locn#"
    xmlns:dbpedia-owl="http://dbpedia.org/ontology/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

    <locn:Location
        rdf:about="datasets/example/geo/Country/eb09946c-26e1-4698-a298-7bb1e468b165"
        dct:identifier="1">
        <locn:geographicName xml:lang="lt">Lietuva</locn:geographicName>
        <dbpedia-owl:capital
            rdf:resource="datasets/example/geo/City/b54c21f6-08b8-4bdd-b785-be1cb2e93a98
↪" />
```

(continues on next page)

(tęsinys iš praeito puslapio)

```
    </locn:Location>

    <locn:Location
        rdf:about="datasets/example/geo/City/b54c21f6-08b8-4bdd-b785-be1cb2e93a98"
        dct:identifier="1">
        <locn:geographicName xml:lang="lt">Vilnius</locn:geographicName>
        <dbpedia-owl:country
            rdf:resource="datasets/example/geo/Country/eb09946c-26e1-4698-a298-
↪7bb1e468b165" />
    </locn:Location>
</rdf:RDF>
```

Išoriniai žodynai suteikia galimybę eksportuoti duomenis *RDF* formatu.

Jei struktūros apraše nėra užpildytas *uri* stulpelis, data, turėtu būti generuojamas tokie RDF duomenys:

```
@base <https://get.data.gov.lt/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix locn: <http://www.w3.org/ns/locn#> .
@prefix dbpedia-owl: <http://dbpedia.org/ontology/> .

<datasets/example/geo/Country/eb09946c-26e1-4698-a298-7bb1e468b165>
    a <datasets/example/geo/Country> ;
    <datasets/example/geo/Country/id> 1 ;
    <datasets/example/geo/Country/name> "Lietuva"@lt ;
    <datasets/example/geo/Country/capital> <datasets/example/geo/City/b54c21f6-08b8-4bdd-
↪b785-be1cb2e93a98> .

<datasets/example/geo/City/b54c21f6-08b8-4bdd-b785-be1cb2e93a98>
    a <datasets/example/geo/City> ;
    <datasets/example/geo/City/id> 1 ;
    <datasets/example/geo/City/name> "Vilnius"@lt ;
    <datasets/example/geo/City/country> <datasets/example/geo/Country/eb09946c-26e1-4698-
↪a298-7bb1e468b165> .
```

10.14.1 Subjekto URI

Pagal nutylėjimą *subjekto* URI yra automatiškai generuojamas ir atrodo taip:

```
https://get.data.gov.lt/datasets/example/geo/Country/eb09946c-26e1-4698-a298-7bb1e468b165
```

Tačiau naudojant kontroliuojamus žodynus, galima nurodyti kitą identifikatorių tokiu būdu:

m	property	type	ref	uri
Country			id	locn:Location
	id	integer		
	uri	uri		locn:Location
	name@lt	text		

Jei *property.uri* sutampa su *model.uri* ir *property.type* yra *uri*, tada formuojant duomenis RDF formatu naudojame ne generuotą subjekto URI, o naudojame lauko reikšmę, kurio *property.uri* sutampa su *model.uri*.

Gali būti ne daugiau kaip vienas *property.uri* su *property.type uri*, kuris sutampa su *model.uri*.

Jei yra keli *uri* tipo laukai, kurie identifikuoja tą patį subjektą, tada kitiems atvejams reikia naudoti ne *model.uri*, o owl:sameAs.

Jei `uri` reikšmė bus `https://sws.geonames.org/597427/`, tada gautume tokius RDF duomenis:

```
@base <https://get.data.gov.lt/example/> .
@prefix locn: <http://www.w3.org/ns/locn#> .

<https://sws.geonames.org/597427/>
    a locn:Location ;
    <Country/id> 1 ;
    <Country/name> "Lietuva"@lt .
```

Atkreipkite dėmesį, kad pats `uri` laukas nėra įtrautkas į RDF duomenis.

Analogiškai, jei *ref* tipo laukas rodo į modelį, kurio *model.uri* sutampa su *property.uri* kuris yra *ref* tipo, tada *ref* lauko reikšmė taip pat įgyja ne gnenruotą URI, o URI iš duomenų.

Pratęsiant tą patį pavyzdį:

m	property	type	ref	uri
Country			id	locn:Location
	id	integer		
	uri	uri		locn:Location
	name@lt	text		
City			id	locn:Location
	id	integer		
	name@lt	text		
	country	ref	Country	locn:Location

Gautumo tokius duomenis:

```
@base <https://get.data.gov.lt/example/> .
@prefix locn: <http://www.w3.org/ns/locn#> .

<https://sws.geonames.org/597427/>
    a locn:Location ;
    <Country/id> 1 ;
    <Country/name> "Lietuva"@lt .

<City/b54c21f6-08b8-4bdd-b785-be1cb2e93a98>
    a locn:Location ;
    <City/id> 1 ;
    <City/name> "Vilnius"@lt ;
    <City/country> <https://sws.geonames.org/597427/> .
```

10.15 Formulės

Formulės rašomos *prepare* stulpelyje. Formulių pagalba galima atlikti įvairius duomenų transformavimo, nuasmeninimo, filtravimo ir kokybės tikrinimo veiksmus.

Kadangi yra labai didelė įvairovė duomenų formatų ir duomenų valdymo mechanizmų, siekiant suvaldyti visą šią įvairovę *DSA* formulės leidžia vieningai aprašyti veiksmus su duomenimis. Vėliau formulės verčiamos į vieningą AST, kurį gali interpretuoti automatizuotos priemonės, priklausomai nuo duomenų šaltinio ir konteksto ir DSA sluoksnio.

10.15.1 Gramatika

Formulių sintaksė atitinką šią ABNF gramatiką:

```
formula     = testlist
testlist    = test *("," test) *1","
test        = or
or          = and *("|" and)
and         = not *("&" not)
not         = "!" not / comp
comp        = expr *(compop expr)
expr        = term *(termop term)
term        = factor *(factorop factor)
factor      = sign factor / composition
composition = atom *trailer
atom        = "(" *1group ")"
            / "[" *1list "]"
            / func / value / name
group       = test *("," test) *1","
list        = test *("," test) *1","
trailer     = "[" *1filter "]" / method / attr
func        = name call
method      = "." name call
call        = "(" *1arglist ")"
arglist     = argument *("," argument) *1","
argument    = test / kwarg
kwarg       = name ":" test
filter      = test *("," test) *1","
attr        = "." name
value       = null / bool / integer / number / string / star
compop      = ">=" / "<=" / "!=" / "=" / "<" / ">"
termpop     = "+" / "-"
factorop    = "*" / "/" / "%"
sign        = "+" / "-"
star        = "*"
name        = ~/[a-z_][a-z0-9_]*/i
number      = ~/\d+(\.\d+)?/
integer     = ~/0|[1-9]\d*/
bool        = "false" / "true"
null        = "null"
string      = ~/(?!"").*?(?<!\\)(\\\\)*?"|'(?!'').*?
                (?<!\\)(\\\\)*?'/i
```

10.15.2 Sintaksės medis

Formulės verčiamos į vieningą abstraktų sintaksės medį. Vieningas abstraktus sintaksės medis leidžia atskirti formulės skaitymo ir interpretavimo veiklas.

Abstraktus sintaksės medis sudarytas iš vienodų elementų turinčių tokias savybes:

name
: Funkcijos pavadinimas.

args
: Funkcijos argumentų sąrašas, kurį gali sudaryti konkrečios reikšmės ar kiti medžio elementai, veiksmai.

Visos formulėje naudojamos išraiškos sintaksės medyje verčiamos į funkcijų ir argumentų medį. Pavyzdžiui `test("a", "b")` bus verčiamas į:

```
{
    "name": "test",
    "args": ["a", "b"],
}
```

10.15.3 Funkcijų iškvietimas

Formulės susideda iš vykdomų funkcijų sekos. Pavyzdžiui funkcijos pavadinimu `test` vykdymas formulėje atrodys taip:

```
test()
```

Aukščiau pavyzdyje pateikta formulė vykdo funkciją `test`, be argumentų. Tačiau funkcijos gali turėti pozicinius ir vardinius argumentus.

10.15.4 Poziciniai argumentai

Poziciniai argumentai perduodami taip:

```
test(a, b, c)
```

Pavyzdyje, funkcijai `test` perduodami trys argumentai `a`, `b` ir `c`. Šioje dokumentacijoje, tais atvejais, kai funkcijos pozicinių argumentų skaičius nėra fiksuotas, naudojama `*args` išraiška, kur * nurodo, kad pozicinių argumentų gali būti 0 ar daugiau.

10.15.5 Vardiniai argumentai

Vardiniai argumentai funkcijai perduodami taip:

```
test(a: 1, b: 2, c: 3)
```

Pozicinius argumentus būtina perduoti tiksliai tokia tvarka, kokios tikisi funkcija. Tačiau vardinius argumentus, galima perduoti, bet kuria tvarka.

Jei vardinių argumentų sąrašas nėra fiksuotas, dokumentacijoje toks argumentų sąrašas užrašomas `**kwargs` forma, kur ** nurodo, kad vardinių argumentų gali būti 0 ar daugiau.

10.15.6 Alternatyvus funkcijos iškvietimas

Funkcijų iškvietimas gali būti užrašomas įprastiniu būdu, pavyzdžiui:

```
test(test(test(a), b), c)
```

Arba funkcijų grandinės (angl. Method chain) būdu:

```
a.test().test(b).test(c)
```

Kadangi formulės dažnai naudojamos tam tikros reikšmės transformavimui, todėl dažnai formulė yra lengviau skaitoma, naudojant funkcijų grandinę.

`test(a)` yra `a.test()` arba `test(a, b)` ir `a.test(b)` yra ekvivalentūs (UFCS).

10.15.7 Standartinės funkcijos

Priklausomai nuo duomenų šaltinio ar konteksto gali būti naudojami skirtingi veiksmai, tačiau žemiau yra pateikti bendrosios paskirties veiksmai:

`func.`**`bind`**`(`*name*`)`
: Rodo į reikšmę pavadinimu `name` iš konteksto. Reikšmės ieškoma tokia tvarka:

  - *`var()`*
  - *`param()`*
  - *`item()`*
  - *`prop()`*

`func.`**`prop`**`(`*name*`)`
: Modelio savybė pavadinimu `name` iš *`property`* stulpelio.

`func.`**`item`**`(`*name*`)`
: Sąrašo elemento savybė pavadinimu `name`.

`func.`**`param`**`(`*name*`)`
: Parametras pavadinimu `name`. Žiūrėti *Parametrai*.

`func.`**`var`**`(`*name*`)`
: Kintamasis apibrėžtas *`set()`* funkcijos pagalba.

`func.`**`self`**`()`
: Rodo į aktyvią reikšmę, naudojamas *`property.prepare`* formulėse.

`func.`**`or`**`(`**args*`)`
: Taip pat galima naudoti tokia išraiška:

  ```
  a | b | c
  ```

  Grąžiną pirmą netuščią reikšmę. Pirmoji netuščia reikšmė nutraukia sekančių `args` argumentų interpretavimą.

`func.`**`and`**`(`**args*`)`
: Taip pat galima naudoti tokia išraiška:

  ```
  a & b & c
  ```

  Grąžina pirmą tuščią reikšmę arba paskutinę reikšmę, jei prieš tai esančios reikšmės netuščios.

func.**not**(*arg*)
Taip pat galima naudoti tokia išraiška:

```
!arg
```

Jei arg tuščia grąžina true, priešingu atveju false.

func.**eq**(*a*, *b*)
Taip pat galima naudoti tokia išraiška:

```
a = b
```

a lygus b.

func.**ne**(*a*, *b*)
Taip pat galima naudoti tokia išraiška:

```
a != b
```

a nelygus b.

func.**lt**(*a*, *b*)
Taip pat galima naudoti tokia išraiška:

```
a < b
```

a mažiau už b.

func.**le**(*a*, *b*)
Taip pat galima naudoti tokia išraiška:

```
a <= b
```

a mažiau arba lygu už b.

func.**gt**(*a*, *b*)
Taip pat galima naudoti tokia išraiška:

```
a > b
```

a daugiau už b.

func.**ge**(*a*, *b*)
Taip pat galima naudoti tokia išraiška:

```
a >= b
```

a daugiau arba lygu už b.

func.**add**(*a*, *b*)
Taip pat galima naudoti tokia išraiška:

```
a + b
```

a ir b suma.

func.**sub**(*a*, *b*)
Taip pat galima naudoti tokia išraiška:

```
a - b
```

a ir b skirtumas.

func.**mul**(*a*, *b*)
Taip pat galima naudoti tokia išraiška:

```
a * b
```

a ir b sandauga.

func.**div**(*a*, *b*)
Taip pat galima naudoti tokia išraiška:

```
a / b
```

a ir b dalyba.

func.**mod**(*a*, *b*)
Taip pat galima naudoti tokia išraiška:

```
a % b
```

a ir b modulis.

func.**positive**(*a*)
Taip pat galima naudoti tokia išraiška:

```
+a
```

Gali būti interpretuojamas skirtingai, priklausomai nuo konteksto. Įprastiniu atveju keičia skaičiaus ženklą.

func.**negative**(*a*)
Taip pat galima naudoti tokia išraiška:

```
-a
```

Gali būti interpretuojamas skirtingai, priklausomai nuo konteksto. Įprastiniu atveju keičia skaičiaus ženklą.

func.**tuple**(**args*)
Taip pat galima naudoti tokia išraiška:

```
(*args)
```

Grupė argumentų.

() Tuščia grupė.

a, b Tas pats, kas `tuple(a, b)`.

func.**list**(**args*)
Taip pat galima naudoti tokia išraiška:

```
[*args]
```

Sąrašas reikšmių.

func.**getattr**(*object*, *attr*)
Taip pat galima naudoti tokia išraiška:

```
object.attr
```

Gaunamos reikšmės pagal atributą arba raktą.

func.**getitem**(*object*, *item*)
Taip pat galima naudoti tokia išraiška:

```
a[item]
```

Gaunamos reikšmės pagal atributą arba raktą.

getitem() gali būti interpretuojamas kaip sąrašo reikšmių filtras:

```
a[b > c]
```

func.**dict**(***kwargs*)
Taip pat galima naudoti tokia išraiška:

```
{a: b}
```

Sudėtinė duomenų struktūra.

func.**set**(***kwargs*)
Taip pat galima naudoti tokia išraiška:

```
{a, b}
```

Reikšmių aibė.

func.**op**(*operator*)
Taip pat galima naudoti tokia išraiška:

```
a(*)
```

Operatoriai gali būti naudojami kaip argumentai.

func.**stack**(*columns*, *values*, *exclude*)
Visus stulpelius išskyrus `exclude` verčia į vieną stulpelių eilutei suteikiant `columns` pavadinimą, o reikšmių stulpeliui `values` pavadinimą. Pavyzdžiui:

vertinimas	2015P2	2016P2
Neigiamai	0	1
Teigiamai	39	28

Tokiai lentelei pritaikius `stack("data", "rodiklis", ["vertinimas"])` transformaciją, gausime tokį rezultatą:

vertinimas	data	rodiklis
Neigiamai	2015P2	0
Neigiamai	2016P2	1
Teigiamai	2015P2	39
Teigiamai	2016P2	28

func.**datetime**(*str*, *format*)
Parse datetime from str, using strftime format.

func.**date**(*str*, *format*)
Parse date from str, using strftime format.

func.**date**(*datetime*)
Return date from datetime.

10.15.8 Failai

Dažnai duomenys teikiami failų pavidalu, kurie gali būti saugomi tiek lokaliai failų sistemoje, tiek nuotoliniame serveryje. Failai gali būti suspausti ir patalpinti į archyvo konteinerius. *DSA* leidžia aprašyti įvairius prieigos prie duomenų, saugomų failuose, atvejus.

resource.source
Nutolusiame serveryje saugomo failo *URI* arba kelias iki lokalaus katalogo. Lokalaus katalogo kelias gali būti pateikiamas tiek *POSIX*, tiek *DOS* formatais, priklausomai nuo to, kokioje operacinėje sistemoje failai saugomi.

resource.prepare
func.**file**(*resource*, *encoding: 'utf-8'*)

Parametrai

- **resource** -- Kelias arba URI iki failo.
- **encoding** -- Failo koduotę.

Ši funkcija leidžia nurodyti failo koduotę, jei failas yra užkoduotas kita, nei UTF-8 koduote. Pilną palaikomų koduočių sąrašą galite rasti šiame sąraše.

func.**extract**(*resource*, *type*)

Parametrai

- **resource** -- Kelias arba URI iki archyvo failo arba failo objektas.
- **type** -- Archyvo tipas.

Išpakuoja archyvą, kuriame saugomi failai. Galimos `type` reikšmės:

zip

tar

rar

Funkcijos rezultatas yra archyvo objektas, kuris leidžia pasiekti esančius archyvo failus *getitem()* funkcijos pagalba.

func.**decompress**(*resource*, *type*)

Parametrai

- **resource** -- Kelias arba URI iki archyvo failo arba failo objektas.
- **type** -- Archyvo tipas.

Taikomas srautinis failo glaudinimo filtras. Galimos `type` reikšmės:

gz

bz2

xz

10.15.9 Stulpeliai lentelėje

CSV ar skaičiuoklių lentelėse stulpelių pavadinimai pateikiami pačioje lentelėje. Eilutė, kurioje surašyti pavadinimai nebūtinai gali būti pirma. Stulpelių pavadinimai gali būti pateikti keliose eilutėse iš kurių formuojamos kompleksinės struktūros (žiūrėti *Kompleksinės struktūros*). Įvairias situacijas galima aprašyti formulių pagalba.

model.prepare
func.**header**(**line*)

null
Lentelėje eilučių pavadinimų nėra. Tokiu atveju, *property.source* stulpelyje reikia pateikti stulpelio numerį, pradedant skaičiuoti nuo 0.

line
Nurodomas eilutės numeris, pradedant eilutes skaičiuoti nuo 0, kur yra pateikti lentelės stulpelių pavadinimai. Pagal nutylėjimą stulpelių pavadinimų ieškoma pirmoje eilutėje.

***line**
Jei lentelė turi kompleksinę stulpelių struktūrą, tada galima pateikti daugiau nei vieną eilutės numerį iš kurių bus nustatomi stulpelių pavadinimai.

func.**head**(*n*)
Praleisti n einančių po stulpelių pavadinimų eilutės.

func.**tail**(*n*)
Ignoruoti n eilučių failo pabaigoje.

property.source
Jei naudojamas *header(null)*, tada nurodomas stulpelio numeris, pradedant nuo 0.

Jei naudojamas *header(line)*, tada nurodomas stulpelio pavadinimas, toks koks įrašytas lentelės line eilutėje.

Jei naudojamas *header(*line)*, tada nurodomas stulpelio pavadinimas, toks koks įrašymas lentelės pirmajame line argumente.

property.prepare
Jei naudojamas header(*line), žiūrėti *Kompleksinės struktūros*.

10.15.10 Duomenų atranka

Duomenų filtravimui naudojamas *model.prepare* stulpelis, kuriame galima apriboti iš šaltinio skaitomų duomenų imtį.

Tarkime, jei turime tokias dvi duomenų lenteles:

COUNTRIES	
COUNTRY	CODE
Lietuva	lt
Latvija	lv

CITIES		
ID	CITY	COUNTRY
1	Vilnius	lt
2	Kaunas	lt
3	Ryga	lv

Jei norėtume atveri ne visų šalių duomenis, o tik Lietuvos, tada duomenų struktūros aprašas turėtu atrodyti taip:

d	r	b	m	property	type	ref	source	prepare
datasets/example/countries								
	salys				sql		sqlite://	
			Country			code	COUNTRIES	**code = "lt"**
				name	string		COUNTRY	
				code	string		CODE	
			City			id	CITIES	
				id	integer		ID	
				name	string		CITY	
				country	ref	Country	COUNTRY	

Kaip ir visur, formulės reikia naudoti pavadinimus ne iš *source* stulpelio, o iš *property*, *model* arba *dataset*.

Jei lentelės yra susijusios ryšiais tarpusavyje, užtenka filtrą nurodyti tik vienoje lentelėje, visose kitose susijusios lentelėse filtrai bus taikomi automatiškai, kad užtikrinti duomenų vientisumą.

Nurodant filtrus yra galimybė naudoti ne tik vienos lentelės laukus, bet ir susijusių lentelių laukus, pavyzdžiui yra galimybė nurodyti tokį filtrą:

d	r	b	m	property	type	ref	source	prepare
			City			id	CITIES	**country.code = "lt"**

Tačiau šiuo atveju, toks filtras būtų perteklinis, nes toks filtras generuojamas automatiškai ir susijusio `Country` modelio, kadangi negalime publikuoti Latvijos miestų, jei publikuojama tik Lietuvos šalis.

Pilnas galimų filtrų sąrašas:

`model.prepare`

`a = b`
: a ir b reikšmės yra lygios.

`a != b`
: a nelygu b.

`a > b`
: a daugiau už b.

`a < b`
: a mažiau už b.

`a >= b`
: a daugiau arba lygu b.

`a <= b`
: a mažiau arba lygu b.

`a.in(b)`
: a lygi bent vienai iš b sekos reikšmių.

`a.notin(b)`
: a nelygi nei vienai iš b sekos reikšmių.

`a.contains(b)`
: a seka savyje turi b seką.

`a.startswith(b)`
: a seka prasideda b seka.

a.endswith(b)
a seka baigiasi b seka.

a & b
a ir b.

a | b
a arba b.

sort(+a, -b)
Rūšiuoti didėjimo tvarka pagal a ir mažėjimo tvarka pagal b.

10.15.11 Periodiškumas

Periodiškumui nurodyti naudojamas model.prepare stulpelis, kuriame galima naudoti tokias formules:

model.prepare
func.**cron**(*line*)
Duomenų atnaujinimo laikas, analogiškas cron formatui.

`line` argumentas aprašomas taip:

nm n-toji minutė, `n` 0-59.

nh n-toji valanda, `n` 0-23.

nd n-toji mėnesio diena, `n` 1-31.

$d Paskutinė mėnesio diena.

nM n-tasis mėnuo, `n` 1-12.

nw n-toji savaitės diena, `n` 0-6 (sekmadienis-šeštadienis).

n#iw n-toji savaitės diena, `i`-toji mėnesio savaitė, `i` 1-6.

n$iw n-toji savaitės diena, `i`-toji savaitė nuo mėnesio galo, `i` 1-6.

, Kableliu galim atskirt kelias laiko vertes.

- Brūkšneliu galima atskirti laiko verčių intervalą.

/ Pasvyruoju brūkšniu galima atskirti laiko verčių kartojimo žingsnį.

Laiko vertės atskiriamos tarpo simbolių. Jei laiko vertė nenurodyta, reiškia įeina visos įmanomos laiko vertės reikšmės.

func.**hourly**()
cron('0m')

func.**daily**()
cron('0m 0d')

func.**weekly**()
cron('0m 0h 0w')

func.**monthly**()
cron('0m 0h 1d')

func.**yearly**()
cron('0m 0h 1d 1M')

10.15.12 Statinės reikšmės

Statinės reikšmės arba konstantos duomenų laukams gali būti nurodomos *property.prepare* stulpelyje naudojant formulės sintaksę. Plačiau apie formules žiūrėti *Formulės* skyrelyje.

10.15.13 Transformavimas

property.prepare stulpelyje gauta šaltinio reikšmė gali būti pasiekiama per self kintamąjį.

property.prepare formulėje gali būti aprašomos kelios reikšmės atskirtos kableliu, tai naudojama ryšio laukams, kai ryšiui aprašyti reikia daugiau nei vieno duomenų lauko.

Formulėje galima naudoti kitus to pačio modelio property pavadinimus, kai aprašomo *property* reikšmės formuojamos dinamiškai naudojant viena ar kelis jau aprašytus laukus.

property.prepare stulpelyje galima naudoti tokias formules:

property.prepare

func.**null**()
: Grąžina `null` reikšmę, jei toliau einančios transformacijos grąžina `null`.

func.**replace**(*old*, *new*)
: Pakeičia visus `old` į `new` simbolių eilutėje.

func.**re**(*pattern*)
: Grąžina atitinkančią reguliariosios išraiškos `pattern` reikšmę arba pirmos grupės reikšmę jei naudojama tik viena grupė arba reikšmių grupę jei `pattern` yra daugiau nei viena grupė.

func.**cast**(*type*)
: Konvertuoja šaltinio tipą į nurodytą `type` tipą. Tipų konvertavimas yra įmanomas tik tam tikrais atvejais. Jei tipų konvertuoti neįmanoma, tada metodas turėtų grąžinti klaidą.

func.**split**()
: Dalina simbolių eilutę naudojant `s+` *reguliariąją išraišką*. Grąžina masyvą.

func.**strip**()
: Pašalina tarpo simbolius iš pradžios ir pabaigos.

func.**lower**()
: Verčia visas raides mažosiomis.

func.**upper**()
: Verčia visas raides didžiosiomis.

func.**len**()
: Grąžina elementų skaičių sekoje.

func.**choose**(*default*)
: Jei šaltinio reikšmė nėra viena iš *Klasifikatoriai*, tada grąžinama default reikšmė.

  Jei `default` nepateiktas, grąžina vieną iš *property.enum* reikšmių, jei duomenų šaltinio reikšmė nėra viena iš *property.enum*, tada grąžinama klaida.

func.**switch**(**cases*)

func.**case**(*cond*, *value*)

func.**case**(*default*)
: Grąžina `value`, jei tenkina `cond` arba `default`. Jei `case(default)` nepateiktas, tada grąžina pradinę reikšmę.

  Jei, `cases` nepateikti, grąžina vieną iš *switch.source* reikšmių, tenkinančių switch prepare sąlygą.

func.**swap**(*old*, *new*)
Swaps an `old` value with `new`, if `self` is equal to `old`.

func.**return**()
Nutraukia transformacijų grandinę ir grąžina reikšmę.

func.**set**(*name*)
Išsaugo reikšmę į kintamąjį `name`.

func.**url**()
Skaido URI į objektas turintį tokias savybes:

scheme URI schema.

netloc Visada URI dalis tarp scheme ir path.

username Naudotojo vardas.

password Slaptažodis.

host Domeno vardas arba IP adresas.

port Prievado numeris.

path Kelias.

query URL dalis einanti tarp ? ir #.

fragment URL dalis einanti po #.

func.**query**()
Skaido URI query dalį į parametrus.

func.**path**()
Skaido failų sistemos arba URI kelią į tokias savybes:

parts Skaido kelią į dalis (plačiau).

drive Diskas (plačiau).

root Šaknis (plačiau).

Kompleksinės struktūros

Daugelis duomenų šaltiniu turi galimybę saugoti kompleksines struktūros. Jei duomenys yra kompleksiniai, tada *property.source* stulpelyje galima nurodyti tik duomens pavadinimą iš pirmojo lygmens, gilesniuose lygmenyse esančius duomenis galima aprašyti naudojant formules *property.prepare* stulpelyje.

Analogiškai duomenų atranką galima daryti ir model eilutėse, jei tai leidžia duomenų šaltinis.

Kaip pavyzdį naudosime tokią *JSON* duomenų struktūrą:

```
{
    "result": {
        "count": 1,
        "results": [
            {
                "type": "dataset",
                "tags": ["CSV"]
            }
        ]
```

(continues on next page)

(tęsinys iš praeito puslapio)

```
    }
}
```

property.prepare

func.**getattr**(*object*, *name*)

Grąžina `object` savybe `name`.

```
>>> self.result.count
1
```

func.**getitem**(*object*, *item*)

Grąžina `object` objekto `item` savybę arba `object` masyvo `item` elementą.

```
>>> self["result"]["count"]
1
```

getitem() ir *getattr()* gali būti naudojami kartu.

```
>>> self.result.results[0].type
"dataset"
```

getitem() gali būti naudojamas, kaip masyvo elementų filtras pateikiant filtro sąlygą.

```
>>> self.result.results[tags = "CSV"].type
["dataset"]

>>> self.result.results[item(tags) = "CSV"].type
["dataset"]
```

Norint gauti visus masyvo elementus, galima naudoti tokią išraišką:

```
>>> self.result.results[].tags[]
["CSV"]
```

Analogiška struktūra gali būti gaunama ir lentelėse, kai stulpelių pavadinimai nurodyti keliose eilutėse, pavyzdyje pateiktą struktūrą atitiktų tokia lentelė:

result		
count	results	
	type	tags
1	dataset	CSV

Šioje lentelėje stulpelių pavadinimai pateikti trijose eilutėse, todėl model.prepare reikėtų naudoti *header(0, 1, 2)*.

SKYRIUS 11

Sąvokos

TGIĮ Teisės gauti informaciją ir duomenų pakartotinio naudojimo įstatymas.

Šis įstatymas įpareigoja valstybės ir savivaldybių institucijas ir joms pavaldžius subjektus atverti duomenis.

Kelios citatos iš įstatymo:

4 straipsnis

1. Institucijos ir valstybės valdomi subjektai privalo teikti pareiškėjams ar jų atstovams duomenis, įskaitant pakartotiniam naudojimui skirtus duomenis, išskyrus šio įstatymo ir kitų įstatymų nustatytus atvejus.

15 straipsnis

1. Visi institucijos ar valstybės valdomo subjekto duomenys turi būti inventorizuoti laikantis principo, kad duomenys gali būti skelbiami pakartotinai naudoti, jeigu tai neprieštarauja šiam ir kitiems įstatymams. Inventorizuotų duomenų sąrašas turi būti skelbiamas Lietuvos atvirų duomenų portale.

2. Institucijos ir valstybės valdomi subjektai turi sudaryti duomenų, dėl kurių yra pateiktos užklausos Lietuvos atvirų duomenų portale arba kurių pakartotinis naudojimas, institucijos ir valstybės valdomo subjekto vertinimu, gali kurti pridėtinę vertę, rinkinius ir juos skelbti šiame portale, jeigu tai neprieštarauja šiam ir kitiems įstatymams.

17 straipsnis

1. Lietuvos atvirų duomenų portalas yra valstybės informacinė sistema, skirta duomenų rinkiniams ir jų metaduomenims sisteminti ir skelbti naudojant vienodą metaduomenų aprašymo formatą, taip pat vieno langelio principu institucijų ir valstybės valdomų subjektų sudarytiems duomenų rinkiniams ir jų metaduomenims ieškoti, peržiūrėti, parsisiųsti, pareiškėjų užklausoms registruoti ir kitoms paslaugoms, susijusioms su šios informacinės sistemos paskirtimi, teikti.

5. Institucijos ir valstybės valdomi subjektai privalo užtikrinti, kad inventorizuotų duomenų sąrašai ir sudaryti duomenų rinkiniai Lietuvos atvirų duomenų portale bus surasti ir pasiekiami šio portalo tvarkytojo nustatyta tvarka ir priemonėmis.

18 straipsnis.

Pareiškėjo teisės gali būti ginamos šiais būdais:

1) pareiškėjas turi teisę apskųsti institucijos veiksmus, neveikimą ar administracinį sprendimą, taip pat institucijos vilkinimą atlikti jos kompetencijai šiuo įstatymu priskirtus veiksmus Viešojo administravimo įstatymo nustatyta tvarka;

2) pareiškėjas turi teisę apskųsti valstybės valdomo subjekto veiksmus ar neveikimą, taip pat valstybės valdomo subjekto vilkinimą atlikti jo kompetencijai šiuo įstatymu priskirtus veiksmus tam pačiam valstybės valdomam subjektui arba bendrosios kompetencijos teismui.

Europos sveikumo karkasas Rekomendacijų rinkinys apie tai, kaip užtikrinti didesnį skaitmeninį sveikumą tarp Europos šalių.

Rekomendacijų sąrašas:

2. Publish the data you own as open data unless certain restrictions apply.

3. Ensure a level playing field for open source software and demonstrate active and fair consideration of using open source software, taking into account the total cost of ownership of the solution.

41. Establish procedures and processes to integrate the opening of data in your common business processes, working routines, and in the development of new information systems.

42. Publish open data in machine-readable, non-proprietary formats. Ensure that open data is accompanied by high quality, machine-readable metadata in non-proprietary formats, including a description of their content, the way data is collected and its level of quality and the licence terms under which it is made available. The use of common vocabularies for expressing metadata is recommended.

43. Communicate clearly the right to access and reuse open data. The legal regimes for facilitating access and reuse, such as licences, should be standardised as much as possible.

44. Put in place catalogues of public services, public data, and interoperability solutions and use common models for describing them.

45. Where useful and feasible to do so, use external information sources and services while developing European public services.

atvirų duomenų direktyva 2019 m. birželio 20 d. Europos Parlamento ir Tarybos direktyva (ES) 2019/1024 dėl atvirųjų duomenų ir viešojo sektoriaus informacijos pakartotinio naudojimo.

duomenų valdymo aktas 2020 m. lapkričio 25 d. Europos Parlamento ir Tarybos reglamento (ES) pasiūlymas 2020/0340 dėl Europos duomenų valdymo (Duomenų valdymo aktas).

aplinkos kintamasis Angliškai tai vadinama *environment variables*, tai yra operacinės sistemos aplinkos kintamieji.

Plačiau apie tai skaitykite Vikipedijoje.

ADP Atvirų duomenų portalas, sudarytas iš *atvirų duomenų katalogo* ir *duomenų saugyklos*.

ADK Lietuvos atvirų duomenų katalogas, prieinamas adresu data.gov.lt.

ADS *Atvirų duomenų saugykla*, skirta pakartotinio panaudojimo duomenų publikavimui, valstybinė atvirų duomenų saugykla pasiekiama get.data.gov.lt adresu.

DSA *Duomenų struktūros aprašas* yra lentelė, kurioje išsamiai aprašyta tam tikro duomenų šaltinio duomenų struktūra. DSA lentelę sudaro penkios dimensijos (duomenų rinkinys, resursas, bazė, modelis, savybė) ir dešimt metaduomenų stulpelių.

ADSA *DSA* lentelė, kurioje aprašomi jau atverti ir viešai prieinami duomenys.

ŠDSA *DSA* lentelė, kurioje aprašoma neatvertų, *pirminio duomenų šaltinio* duomenų struktūra.

didelės vertės duomenys

aukštos vertės duomenys Duomenys apibrėžti *atvirų duomenų direktyvos* 5 skyriuje.

Aukštos vertės duomenų sritys yra šios:

- Geoerdviniai duomenys
- Aplinka ir žemės stebėjimai
- Meteorologiniai duomenys
- Statistika (demografiniai ir ekonominiai rodikliai)
- Įmonės ir įmonių savininkai
- Judumas

BDAR 2016 m. balandžio 27 d. Europos Parlamento ir Tarybos reglamentas (ES) 2016/679 dėl fizinių asmenų apsaugos tvarkant asmens duomenis ir dėl laisvo tokių duomenų judėjimo ir kuriuo panaikinama Direktyva 95/46/EB (Bendrasis duomenų apsaugos reglamentas).

duomenų serializavimo formatas Duomenys gali būti serializuojami įvairiais formatais, pavyzdžiui YAML formatu:

```
type: project
title: Manifestas
```

JSON formatu:

```
{"type": "project", "title": "Manifestas"}
```

Turtle formatu:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
<http://atviriduomenys.lt> a foaf:Project;
    rdfs:label "Manifestas" .
```

MessagePack dvejetainiu formatu, kurio turinys pateiktas naudojant BASE64 koduotę:

```
gqR0eXBlp3Byb2plY3SkbmFtZapNYW5pZmVzdGFz
```

Visuose šiuose pavyzdžiuose yra pateikti tie patys duomenys, tačiau naudojami skirtingi duomenų serializavimo formatai, koduotės ir skirtingi žodynai.

brandos lygis Duomenų brandos lygiai yra apibrėžti 5 Open Data svetainėje. Viso yra penki brandos lygiai, tačiau papildomai verta įtraukti ir nulinį brandos lygį, kai duomenų poreikis yra, tačiau duomenys nekaupiami arba negali būti publikuojami dėl teisinių ar kitų apribojimų.

5 Open Data svetainėje brandos lygia apibrėžti, kaip pavyzdį nurodant formatus. Nors formatus galima naudoti kaip pavyzdį labai abstrakčiai apibūdinant ką reiškia brandos lygiai, tačiau tikslus brandos lygis gali būti suteiktas tik atskiriems duomenų laukams, o ne formatui.

Duomenų brandos lygiai yra tokie:

0 Duomenys nekaupiami, tačiau poreikis tokiems duomenims yra. Gali būti ir tokių atvejų, kai duomenys yra kaupiami, tačiau dėl teisinių ar kitų priežasčių negali būti publikuojami.

1 Duomenys kaupiami ir publikuojami viešai, bet kokia forma ir bet kokiu formatu. Pavyzdžiui datos tipo laukas gali būti pateikiamas įvairiais formatais „Pirmadienis“, „2021 gegužės 10 d.“, „5/10/21“ ir pan. Kadangi šiuo atveju data gali būti užrašyta bet kokia forma ir bet kokiu tikslumu, nėra galimybės automatinėmis priemonėmis patikimai nuskaityti tokių duomenų.

2 Publikuojami duomenys turi aiškią, mašininiu būdu nuskaitomą struktūrą, tačiau pateikiami nestandartinių arba nuosavybiniu formatu. Pavyzdžiui datos tipo lauko duomenys pateikiami nestandartiniu formatu, tačiau visos reikšmės pateiktos naudojant tą patį formatą, „5/10/21“, „6/10/21“ ir pan. Šiuo atveju, automatiškai

nuskaityti tokius duomenis įmanoma tik papildomai įgyvendinant duomenų nuskaitymo priemones, kuriose yra įgyvendintas būtent tokio nestandartinio formato duomenų skaitymas.

3 Duomenys pateikiami naudojant standartinį formatą. Lietuvos atvirų duomenų kontekste, *standartiniai formatai yra apibrėžti duomenų struktūros aprašo specifikacijoje*. Pavyzdžiui datos tipo lauko duomenys pateikiami standartiniu ISO 8601 formatu. Kadangi duomenys yra pateikti standartiniu formatu, pačio formato specifikacija yra atvira ir viešai publikuojama, o duomenų nuskaitymo priemonės tokį atvirą formatą palaiko, todėl tokių duomenų nuskaitymui nereikia įdėjo jokio papildomo darbo.

4 Kiekvienas publikuojamų duomenų *objektas* turi unikalų identifikatorių ir naudojant tokius unikalius objektų identifikatorius, skirtingų tipų objektai siejami tarpusavyje. Kartu su duomenimis pateikiami ir metaduomenys apie tai, kaip skirtingų tipų objektai siejasi tarpusavyje.

Pavyzdžiui miesto tipo objektui „Vilnius“ yra suteiktas unikalus identifikatorius *6868eca7-0ae1-4390-83d0-7af642a62863*, o šalies tipo objekto „Lietuva“ duomenų lauko „sostinė“ reikšmė yra objekto „Vilnius“ unikalus identifikatorius *6868eca7-0ae1-4390-83d0-7af642a62863*.

Turint tokį brandos lygį, duomenis galima ne tik nuskaityti, bet ir jungti tarpusavyje, o jungiant skirtingus duomenis tarpusavyje atsiveria daugiau galimybių juos naudoti įvairiuose taikymuose.

5 Kartu su publikuojamais duomenimis, pateikiami ir metaduomenys apie tai, kaip publikuojami duomenys siejasi su kitais viešaisiais duomenų žodynais (ontologijomis). Pavyzdžiui datos duomenų laukas yra susiejamas su „Dublin Core Metadata Initiative“ publikuojama ontologija, nurodant, kad datos lauko semantinė prasmė yra tokia pati, kaip apibrėžta dcterms:created ontologijoje. Šiuo atveju, nurodoma, kad datos laukas būtent yra tam tikro resurso sukūrimo data.

Kai duomenys yra susieti su išoriniais žodynais, atsiranda galimybė įgyvendinti tokias priemones, kurios veiktų universaliai, nepriklausomai nuo duomenų šaltinio ar duomenų kilmės.

kanoniniai duomenys Kanoniniai duomenys yra tarsi duomenų etalonas, kuris nusako kokios duomenų reikšmės yra teisingos. Pavyzdžiui įmonės pavadinimas gali būti užrašomas įvairiausiomis formomis, pavyzdžiui:

Įmonės kodas	Įmonės pavadinimas
-	UAB "Duomesta"
-	UAB „Duomesta“
-	Duomesta
-	DUOMESTA
-	Uždaroji akcinė bendrovė Duomesta
-	Duomesta, UAB
-	DSTA UAB

Jei duomenų rinkinyje nėra pateiktas įmonės registracijos kodas, tada unikaliai identifikuoti įmonę yra gan sudėtinga.

Tačiau turint autoritetingus kanoninius duomenis:

Įmonės kodas	Įmones pavadinimas
111111111	UAB "Duomesta"

Užduotis unikaliai identifikuoti įmonę pasidaro paprastesnė. Todėl kanoniniai duomenys yra labai svarbūs.

kodinis pavadinimas Pavadinimas, kuriam keliami tam tikri apribojimai.

manifestas Atvirų duomenų manifestas yra *DSA* lentelių rinkinys, kuriuose aprašyti duomenų šaltiniai ir juose esančių duomenų struktūra.

Žodis manifestas yra kilęs iš programavimo srityje naudojamo termino Manifesto failas, kuriame pateikiami metaduomenys apie programinio paketo sandarą.

Duomenų kontekste, žodis manifestas turėtu būti suprantamas, kaip metaduomenų lentelė apie įvairiuose duomenų šaltiniuose publikuojamus duomenis.

metaduomenys Duomenys apie duomenis yra vadinami metaduomenimis. Pavyzdžiui duomenų struktūros aprašas konkrečiam CSV duomenų failui gali būti vadinamas CSV failo metaduomenimis.

normalizavimas Duomenų normalizavimas yra duomenų struktūros transformavimo procesas taikant taip vadinamas normalines formas, tam kad sumažinti duomenų pasikartojimą.

Plačiau apie tai skaitykite Vikipedijoje.

prieigos taškas Prieigos taškas yra *REST API* terminas, nurodantis URL kelio dalį iki tam tikro resurso.

Plačiau skaitykite Vikipedijoje.

REST API Representational State Transfer (REST) yra taisyklių ir rekomendacijų rinkinys sirtas *web servisams* kurti.

Plačiau skaitykite Vikipedijoje.

web servisas Web servisas yra interneto paslauga skirta automatizuotiems robotams. Interneto svetainės dažniausiai yra skirtos žmonėms, tačiau web servisai yra skirti mašioms, kurios gali komunikuoti viena su kita.

Plačiau skaitykite Vikipedijoje.

YAML YAML yra *duomenų serializavimo formatas*, kuris skirtas ne tik mašininiam skaitymui, bet su šio formato turiniu tiesiogiai gali dirbti ir žmogus. YAML formato pavyzdys:

```
container:
  name: value
```

YAML yra sukurtas JSON formatu pagrindu, siekant palengvinti darbą su JSON serializuotais duomenimis žmonėms. Analogiškas pavyzdys JSON formatu atrodo taip:

```
{"container": {"name": "value"}}
```

viešasis žodynas Viešieji žodynai, dar vadinami ontologijomis, šie žodynai dažnai yra gerai dokumentuoti ir skelbiami viešai, jie yra skirti globaliam susietųjų duomenų tinkui kurti (angl. *linked data*).

sisteminis pavadinimas Sisteminis pavadinimas yra naudojamas objektų identifikavimui ir yra naudojamas URL nuorodose ir visur kitur, kure reikia nurodyti ryšį su objektų, naudojamas to objekto sisteminis pavadinimas.

Sisteminis pavadinimas sudaromas tik iš lotyniškų raidžių ir -_/ simbolių.

pirminis duomenų šaltinis Įstaigos ar kitos organizacijos pagrindinis duomenų šaltinis.

DCAT Duomenų katalogo žodynas (angl. Data Catalog Vocabulary) yra standartas skirtas duomenų rinkiniams aprašyti. Aprašant duomenis DCAT standartu reikėų vadovautis *DCAT-AP* specifikacijomis.

DCAT-AP DCAT-AP (DCAT Application Profile) yra specifikacija, detalizuojanti DCAT naudojima, nurodant kurios DCAT klasės ir savybės yra privalomos, kurios rekomenduojamos ir kaip jas naudoti.

dimensija Dimensija yra metaduomenų, aprašomų DSA lentelėje, grupė. DSA lentelėje metaduomenys skirstomi į tokias dimensijas:

- duomenų rinkinys
- resursas
- bazė
- modelis
- savybė

Kiekviena dimensija turi skirtingą metaduomenų detalumo lygį.

Plačiau apie dimensijas: *Dimensijos*.

duomenų rinkinys Duomenų rinkinys apibrėžia turimus arba pageidaujamus duomenis, reikalingus konkrečios organizacijos, konkrečiai veiklai vykdyti.

Duomenų rinkinys gali būti registras, informacinės sistemos duomenų bazė, interneto svetainės duomenų bazė, skaičiuoklės lentelė, dokumentų katalogas arba duomenys, kurie dar nėra kaupiami, tačiau yra reikalingi tam tikrai veiklai vykdyti.

Duomenų rinkinio fizinė reprezentacija, tai yra patys duomenys yra vadinami *distribucija*. Duomenų rinkinyje gali būti daugiau nei viena distribucija, jei fiziškai duomenys yra suskaidyti pagal vietos, laiko, detalumo, struktūros elementus, natūralios kalbos ar kitus kriterijus.

Dažnai duomenų rinkinys painiojamas su distribucija. Duomenų rinkinys apibrėžia tam tikrą grupę duomenų, kurie nebūtinai fiziškai egzistuoja, tuo tarpu distribucija yra fiziniai duomenys įeinantys į duomenų rinkinio sudėtį.

Duomenų rinkiniai neskaidomi pagal vietos, laiko, detalumo, struktūros ar kitus kriterijus.

Plačiau apie tai, kaip duomenų rinkiniai aprašomi duomenų struktūros apraše skaitykite skyriuje *Duomenų rinkinys*.

Duomenų rinkinys atitinka dcat:Dataset apibrėžimą.

distribucija Distribucija yra duomenų rinkinio fizinė reprezentacija. Vienas duomenų rinkinys gali būti sudarytas iš kelių distribucijų, tuos pačius duomenis pateikiant skirtingais formatais, suskaidant duomenis pagal laiko, vietos ar kitus kriterijus, tuos pačius duomenis pateikiant skirtingu detalumu arba pateikiant agreguotus duomenis įvairiais pjūviais.

Duomenų struktūros aprašo kontekste, distribucija yra tas pats, kas *resource*.

Distribucija atitinka dcat:Distribution apibrėžimą:

bazė Bazė arba loginė klasė yra modelių grupė turinčių bendras savybes ir vienodą semantinę prasmę.

Dažnai skirtingų organizacijų veikloje naudojami duomenų rinkiniai turi vienodą semantinę prasmę. Pavyzdžiui, daugelis organizacijų turi naujienų duomenis. Norint visų organizacijų naujienų duomenis aprašyti vieningai, galima pasitelkti vieną bazę, arba vieną duomenų rinkinį, kurio struktūrą naudoja visi kiti rinkiniai. Tai bazė būtent ir būtų struktūros šablonas pagal kurį būtų sudaromos visi kitų analogiškų rinkinių struktūra.

Bazė yra tas pats, kas *modelis* arba tiksliau modelio šablonas.

Duomenų struktūros aprašo kontekste api bazę plačiau skaitykite skyriuje *Modelio bazė*.

modelis Modelis yra gan plati sąvoka turinti daug prasmių, priklausomai nuo konteksto. Šioje dokumentacijoje, modelis yra duomenų struktūros aprašo dalis leidžianti aprašyti duomenis pateiktus įvairiais formatais.

Tiksli modelio prasmė priklauso nuo duomenų šaltinio, kurio duomenys yra aprašomi:

- CSV failo atveju, modelis yra CSV faile esanti lentelė,
- Excel failo atveju, modelis yra kiekviena lentelė (arba lapas) esanti Excel faile,
- SQL duomenų bazių atveju, modelis yra viena duomenų bazės lentelė,
- JSON dokumento atveju, modelis yra kiekvienas masyvas esantis JSON dokumente,
- XML atveju, modelis yra kiekvienas elementų masyvas esantis XML faile.

Duomenų rinkiniai aprašo konkretaus autoriaus duomenis, skirtingi autoriai gali naudoti tuos pačius duomenis, todėl duomenys skirtinguose rinkiniuose gali dubliuotis. Tuo tarpu modeliai aprašo duomenis pagal jų semantinę prasmę, nepriklausomai nuo autoriaus, tai leidžia apjungti skirtingų autorių naudojamus duomenis, pagal jų semantinę prasmę, modelių pagalba.

DSA lentelėje atitinka `model`. Duomenų modelį atitinkanti fizinė reprezentacija nurodoma `source` stulpelyje. `source` gali būti duomenų bazės lentelė, CSV failas ar kita, priklauso nuo duomenų šaltinio tipo. Sąsaja su išoriniais žodynais pateikiama `uri` stulpelyje. Siejant su išoriniais žodynais, pateikiama nuoroda į rdfs:Class.

savybė *Duomenų modeliui* priklausančių informacinių *objektų* savybė, pavyzdžiui miesto pavadinimas, šalis kuriai priklauso miestas. *DSA* lentelėje atitinka `property`. Atitinka rdfs:Property arba lentelės stulpelį.

subjektas Subjektas lietuvių kalboje vadinamas veiksniu, duomenų kontekste įvardija objektą apie kurį eina kalba.

Tarkime saknyje „Namas turi stogą“ subjektas yra Namas, todėl, kad kalba eina apie namą.

objektas Vienas duomenų įrašas sudarytas iš savybių ir savybėms priskirtų reikšmių. Informacinis objektas turi turėti unikalų identifikatorių. Atitinka rdfs:Resource arba lentelės vieną eilutę.

žodynas Duomenų kontekste, žodynas yra susitarimas, kokiais pavadinimais vadinami objektai ir jų savybės. Dažniausiai kiekvienas duomenų rinkinys turi savo vidinį naudojamą žodyną, visas Lietuvos atvirų duomenų modelis turi savo vidinį žodyną, kuris suvienodina skirtingus duomenų rinkinių naudojamus žodynus. Yra *viešieji žodynai*, dar vadinami ontologijomis, kurie yra skelbiami viešai ir skirti globaliam susietųjų duomenų tinklui kurti.

Duomenų kontekste, žodynas yra tiesiog *modelių* ir *savybių* pavadinimų rinkinys. Skirtingi duomenų šaltiniai dažniausiai naudoja skirtingus žodynus, t.y. naudoja skirtingus *modelių* ir *savybių* pavadinimus.

Duomenų struktūros aprašas leidžia skirtinguose duomenų šaltiniuose naudojamus pavadinimus suvienodinti, taip, kad visi šaltiniai naudotų vieningą žodyną.

Vieningo žodyno sudarymas yra gan sudėtinga užduotis, todėl, *DSA* leidžia prie vieningo žodyno pereiti palaipsniui:

- pirmiausia sudaromas vieno duomenų rinkinio žodynas,
- kuris palaipsniui transformuojamas į Lietuvos vieningą žodyną,
- o Lietuvos vieningas žodynas palaipsniui transformuojamas į globalų žodyną, nurodant sąsajas su išoriniais žodynais ir standartais.

Žodynai sudaromi pasitelkiant *vardų erdves*.

API Programavimo sąsaja (*angl. Application Programming Interface*).

duomenų šaltinis Resursas, kuriame saugomi duomenys. Toks resursas tampa duomenų šaltiniu, kai tokius duomenis norima pakartotinai panaudoti, tokiu atveju, iš pakartotinio panaudojimo perspektyvos toks resursas tampa duomenų šaltiniu.

ETL Duomenų ištraukimas, transformavimas ir užkrovimas (*angl. Extract Transform Load*).

iteratorius Tam tikra funkcija, kuri grąžina keletą lementų, tačiau ne visus iš karto, o po vieną.

URI Vieningas resurso identifikatorius (*angl. Uniform Resource Identifier*).

POSIX Universali operacinių sistemų sąsaja (*angl. Portable Operating System Interface*) - standartas apibrėžiantis operacinių sistemų sąsają, kad skirtingos operacinės sistemos būtų suderinamos tarpusavyje.

https://en.wikipedia.org/wiki/POSIX

DOS MS-DOS.

reguliarioji išraiška Simbolių seka apibrėžianti tam tikrą šabloną tekste (angl. Regular Expression).

JSON Atviras duomenų formatas (angl. JavaScript Object Notation).

RDF Duomenų modelis sudarytas iš subjekto, predikato ir objekto tripletų (angl. Resource Description Framework).

IVPK Informacinės visuomenės plėtros komitetas.

- Rodyklė

Python Module Index

f

Indeksas

Symbols

A

B

C

D

N

O

P

Q

R

S

T

U

V

W

X

Y