LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

RUBY PAPLAŠINĀJUMS ĒRTAI TULKOJUMU PĀRVALDEI

Ruby gem for easy translation management

KVALIFIKĀCIJAS DARBS

Programmēšana un datortīklu administrēšana -
pirmā līmeņa profesionālās augstākās izglītības studiju programma

Autors: **Roberts Groza**
Studenta apliecības Nr.: rg15012
Darba vadītāja: doc. Vineta Arnicāne

RĪGA 2017

ANOTĀCIJA

Kvalifikācijas darbā izstrādātais *Ruby* paplašinājums *Kakimasu* (japāņu valodā *rakstīt*) ir paredzēts ātrai un ērtai projekta tulkojumu pārvaldei. *Kakimasu* ļauj tulkot projektu no pārlūka, neaiztiekot YAML failus, kuros pēc noklusējuma glabājas tulkojumi, līdz ar to, projektu var iztulkot bez īpašām priekšzināšanām, turklāt ātri, kas ietaupa tik dārgo laiku un līdzekļus. Paplašinājumu var lietot kā izstrādātāji, tā paši klienti.

Paplašinājums ir izstrādāts *Ruby* programmēšanas valodā un tas ir saderīgs ar jebkuru *Ruby on Rails 5* projektu. Paplašinājuma izstrādē tika izmantotas arī HTML, CSS, Javascript tehnoloģijas.

RUBY ON RAILS, RUBY PAPLAŠINĀJUMS, TULKOŠANA, TULKOJUMU ATSLĒGA, I18N, GOOGLE TRANSLATE

ABSTRACT

RUBY GEM FOR EASY TRANSLATION MANAGEMENT

In this qualification work developed Ruby gem *Kakimasu* (in japanese *write*) is meant for fast and easy translation management. *Kakimasu* allows to translate project directly from browser without writing in YAML files, where translations are stored by default, thereby project can be translated without special knowledge, besides fast, it saves valuable time and money. This gem can be used by developers and even by customers themselves.

Kakimasu is developed with Ruby programming language and it is compatible with any Ruby on Rails 5 project. In development also were used HTML, CSS, Javascript technologies

RUBY ON RAILS, RUBY GEM, TRANSLATION, TRANSLATION KEY, I18N, GOOGLE TRANSLATE

SATURS

APZĪMĒJUMU SARAKSTS

Lietotājs – Jebkurš, kas izmanto izstrādāto paplašinājumu.

Izstrādātājs – Programmētājs, kas uztur vai izstrādā projektu, pie kura pievienots *Kakimasu*.

Ruby – atvērtā pirmkoda objektorientēta programmēšanas valoda, kurā uzsvars tiek likts uz vienkāršību un produktivitāti. Radusies 1995. gadā un autors ir Yukihiro Matsumoto.

Ruby on Rails – *Ruby* ietvars (framework).

Ruby paplašinājums – Paplašinājums (bibliotēka), kas ir pievienojams *Ruby* projektam.

I18n – *Ruby* paplašinājums, kas paredzēts tulkojumu nodrošināšanai projektā. Apzīmējums radies no angļu valodas vārda *internationalization.*

Tulkojuma atslēga – Identifikators (parasti simbolu virkne), kuram tiek piešķirts tulkojums katrā valodā.

Unikāla atslēga – Tulkojuma atslēga, kurai par spīti tam, ka sakrīt ar citu atslēgu citā failā, atšķiras tulkojums no otras atslēgas.

Goole translate – "Google" pakalpojums frāžu tulkošanai no vienas valodas uz citu.

YAML – (Yaml Ain't Markup Language). Valoda, kas paredzētu datu uzskaitei un glabāšanai failos cilvēkam lasāma teksta veidā.

PPS – Programmatūras prasību specifikācija.

PPA – Programmatūras projektējuma apraksts.

HTML – jeb Hiperteksta iezīmēšanas valoda ir izstrādāta pārlūkprogrammā attēlojamas insformācijas glabāšanai.

ERB (Iegultais Ruby) – Šablonu sistēma, kas palīdz iestrādāt Ruby kodu HTML dokumentā.

CSS – Īpaša stila lapas valoda, ko lieto, lai aprakstītu izskatu iezīmēšanas valodā veidotiem dokumentiem.

JQuery – Starpplatformu Javascript bibliotēka, kas ir veidota lai atvieglotu HTML klienta puses skriptu.

Hash – Iekšējās atmiņas objekts.

Redis serveris – Atvērta pirmkoda programma, kas nodrošina iekšējās atmiņas datu struktūru glabāšanu. Var tikt izmantots kā datu bāze vai kešatmiņa.

Popup – Uznirstošais logs, kas parādās blakus kādam objektam, bieži tiek izmantots, kā logs kāda objekta skaidrošanai.

Modal – Uznirstošais logs, kas pilda dialoga funkciju. Ļoti populārs risinājums, kad nepieciešama uznirstoša forma vai uznirstošs paziņojums.

IEVADS

Tēmas izvēle

Tēma tika izvēlēta pamatojoties uz klientu vajadzībām un izstrādātāju interesēm - pēc iespējas ērtāk un efektīvāk tulkot projektu. Katra klienta vēlmes mēdz būt dažādas un nepastāvīgas, piemēram, projekta sākumā tiek nolemts, ka projekts būs tikai angļu valodā un tas nebūs jātulko, taču pēkšņi, kad komanda gatavojas projekta nodošanai, klients paziņo, ka ir nepieciešama arī latviešu valoda, un pēc iespējas ātrāk. Tulkošanas process, īpaši, kad visa sistēma ir izstrādāta, mēdz būt ļoti dārgs un prasa daudz laika, taču tas nav pārāk sarežģīts. Tā kā tulkojumi glabājas projekta failos, tad šo uzdevumu nevar uzticēt cilvēkam bez pieredzes programmēšanā.

Nolūks

Kvalifikācijas darba mērķis ir izstrādāt funkcionējošu *Ruby* paplašinājumu, kas ļautu ātri un ērti tulkot un pārvaldīt tulkojumus *Ruby on Rails* projektos. Izstrādātais produkts atvieglos programmētāja darbu un dos iespēju klientam pašam izlabot kādu kļūdainu tulkojumu, bez programmētāju iesaistīšanas, līdz ar to ļaus ietaupīt dārgo laiku un līdzekļus.

Šī dokumenta nolūks ir norādīt visas produkta prasības, kas tiks realizētas kvalifikācijas darba ietvaros, aprakstīt programmas projektējumu un testēšanas kārtību.

Darbības sfēra

Izstrādātais *Ruby on Rails* paplašinājums *Kakimasu* ir paredzēts ātrai un ērtai projekta tulkošanai un tulkojumu pārvaldei. Galvenā paplašinājuma priekšorība ir ērtums un pieejamība, jo lietotājam nav jāraksta tulkojumi YAML failos (kā tas *Ruby on Rails* ir pēc noklusējuma). Līdz ar to, tulkošanu var veikt par lietotājs bez pieredzes darbā ar *Ruby on Rails*.

Pārskats

Šis dokuments sastāv no vairākām nodaļām:

1. Vispārējais apraksts – Aprakstītas produkta perspektīvas, apskatītas lietotāju raksturiezīmes, kā arī ietver sevī vispārējos ierobežojumus, pieņēmumus un atkarības saistībā ar izstrādāto paplašinājumu. Nodaļa paredzēta, lai sniegtu virspusēju ieskatu paplašinājuma darbībā.
2. Projekta pārvaldība – Nodaļā aprakstīts, kā tika organizēts projekts, kā tika pārvaldītas programmatūras versijas, nodrošināta programkoda kvalitāte, kā arī nodaļas beigās ir pieejama informācija par darbietilpības novērtējumu.
3. Programmatūras prasību specifikācija – Tiek izvirzītas paplašinājuma funkcionālās un nefunkcionālās prasības, kā arī tiek veikta moduļu dekompozīcija.
4. Programmatūras projektējuma apraksts – Nodaļā aprakstīts, kā *Ruby on Rails* vidē notiek tulkošana, kā tiek realizētas funkcijas un, kā arī lietotāja saskarnes projektējums.
5. Testēšanas dokumentācija – Nodaļā tiek aprakstīts, kā tiek veikta paplašinājuma testēšana un, testēšanas žurnāla veidā, tiek aprakstīti agrāko testēšanu iegūtie rezultāti.

Rezultāti – Tiek veikts apkopojums par paveikto, neliels ieskats tajā, kā veicās programkoda izstrādes laikā, kā arī izdarīti secinājumi.

Izmantotā literatūra – Nodaļā apkopota visa kvalifikācijas darba izstrādes laikā izmantotā literatūra.

Pielikums – Satur radītā programkoda fragmentus.

Saistība ar citiem dokumentiem

Prasību specifikācija ir izstrādāta saskāņā ar dokumentu LVS 68:1996 “Programmatūras prasību specifikācijas ceļvedis” [1]. Programmatūras projektējuma apraksts ir izstrādāts saskaņā ar document LVS 72:1996, “Ieteicamā programmatūras projektējuma apraksts” [2]. Kā arī testēšanas dokumentācija ir izstrādāta saskaņā ar LVS 70:1996, “Programmatūras testēšanas dokumentācija” [3].

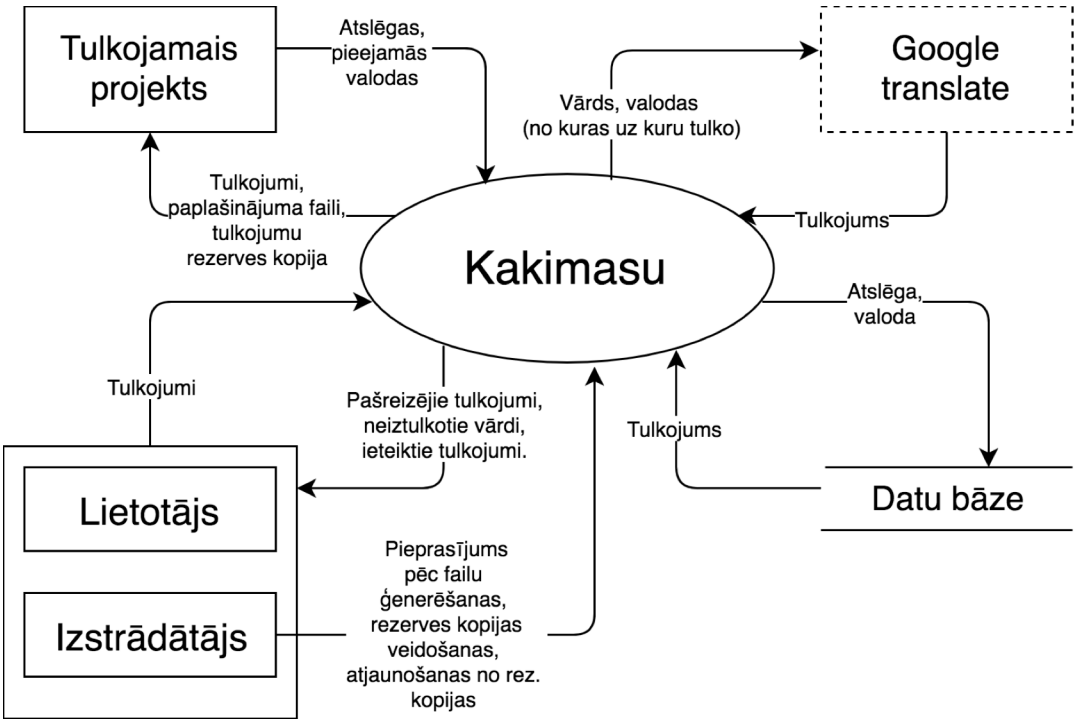
1. VISPĀRĒJAIS APRAKSTS

1.1. Produkta perspektīva

Kakimasu darbojas kā pilnvērtīgs Ruby paplašinājums. Tas ir pievienojams gatavam Ruby on Rails projektam, lai to iztulkotu.

Pašlaik priekš *WordPress* un *php* projektiem ir pieejamie vairāki tulkošanas risinājumi *Poedit2*, *Weglot u.c.,* taču tie neder *Ruby on Rails* projektiem. Ir pieejami daži paplašinājumi, kas nedaudz atvieglo tulkojumu pārvaldi, taču tie nav tik draudzīgi lietotājam, tie nav tik populāri un to repozitorijos jau vairākus gadus nav veiktas izmaiņa, piemēram, *express_translate* [4], *translate* [5]. Ir pieejami arī *Ruby* paplašinājumi ar *google translate* atbalstu, piemēram, *easy_translate* [6], taču viss, ko tas spēj ir atgriezt *google translate* tulkojumus. Pašlaik ir grūti atrast kādu universālu *Ruby* paplašinājumu, kas ļautu viegli un ātri iztulkot *Ruby on Rails* projektu un pārvaldīt tulkojumus.

1.2. Lietotāja raksturiezīmes



1.1. att. **0. līmeņa datu plūsmu diagramma**

Kā parādīts attēlā 1.1., tiek izdalītas 2 lietotāju grupas – lietotāji un izstrādātāji. Lietotāji ir visi, kas tulko projektu (administratori, moderatori, tulki, programmētāji), savukārt, lietotāju grupa "izstrādātājs" apraksta programmētājus, kas atbild par projekta izstrādi, uzturēšanu.

Lietotāji ar paplašinājuma palīdzību var veikt visas pamatfunkcijas (pievienot tulkojumus, izmantot *google translate* servisu, redzēt iztulkotās/neiztulkotās atslēgas). Izstrādātāji var veikt pieprasījumu pēc skatu vai piekļuves tiesību failu ģenerēšanas, lai tos varētu rediģēt, kā arī pēc tulkojumu rezerves kopijas izveidošanas un tulkojumu atjaunošanas no tās. Izstrādātājiem nav lietotāja saskarnes – izstrādātāju funkcijas tiek izsauktas no komandrindas ar attiecīgo pieprasījumu.

Kakimasu lietotājam ir nepieciešamas pamatprasmes darbā ar datoru un pārlūku. Nav nepieciešamas speciālas zināšanas darbam ar datu bāzēm vai YAML failiem. Izstrādātājam, kas pievieno paplašinājumu jābūt pamatprasmēm darbam ar *Ruby on Rails* ietvaru.

1.3. Vispārējie ierobežojumi

Projekts, kuram vēlās pievienot *Kakimasu* ir jābūt izstrādātam ar *Ruby on Rails* ietvaru, jo neskatoties uz to, ka projekts izstrādāts *Ruby* valodā, starp *Ruby* ietvariem var variēt tas kā tiek realizēta tulkošoana. Nepieciešama datu bāze, kurā uzglabāt tulkojumus. Lai strādātu google translate atbalsts, ir nepieciešams interneta pieslēgums.

1.4. Pieņēmumi un atkarības

Tiek pieņemts, ka, ja projektam tiek pievienots *Kakimasu*, tad ir uzstādīta datu bāze, kurā glabāt tulkojumus. Tiek pieņemts, ka programmētājs ir programmējis pēc labās prakses, un projektā viss teksts ir rakstīts kā tulkojumu atslēgas (nevis "statisks" teksts), lai uzreiz varētu pievienot tulkojumus, ja klients palūdz. Ja projekts nav paredzēts tulkošanai, un visur ir pierakstīts statisks teksts, tad programmētājiem ir jāpārraksta kods.

Tiek pieņemts, ka projektam ir uzinstalēti visi *Ruby* paplašinājumi, no kuriem ir atkarīgs *Kakimasu*. Ja nepieciešamie paplašinājumi nav uzinstalēti, tad tie tāpat tiks uzinstalēti kopā ar *Kakimasu*.

Kā *Kakimasu* pamatvaloda skatos tiek izmantota angļu valoda, ja lietotājs vēlas izmantot kādu citu valodu kā pamatvalodu, tad skatus nepieciešams tulkot gluži tāpat kā tiek tulkoti pārējie projekta skati. Tiek pieņemts, ka skatus, vismaz uz vienu no valodām, kuras tiek izmantotas projektā, tulko izstrādātājs.

1.5. Produkta funkcijas

Izstrādātajam *Ruby* paplašinājumam ir nepieciešams veikt šādas funkcijas:

1. Tulkojuma pievienošana vārdam.
2. Piešķirt dažādus tulkojumus vienai tulkojumu atslēgai, ja tās atrodas dažādos skatos.
3. Rādīt *google translate* piedāvāto tulkojumu.
4. Redzēt visus pievienotos tulkojumus visās valodās.
5. Redzēt visus projektā esošos tulkojumus.
6. Redzēt visus vārdus, kuriem kādā no valodām, trūkst tulkojuma.
7. Rādīt procentuāli, cik daudz vārdi ir pilnībā iztukoti.
8. Iespēja izstrādātājam pielāgot paplašinājuma skatus projekta vajadzībām.
9. Iespēja noteikt, kādām lietotāju grupām ir nodrošināta pieeja paplašinājuma funkcijām.
10. Izveidot rezerves kopiju visiem tulkojumiem.
11. Atjaunot tulkojumus no rezerves kopijas.

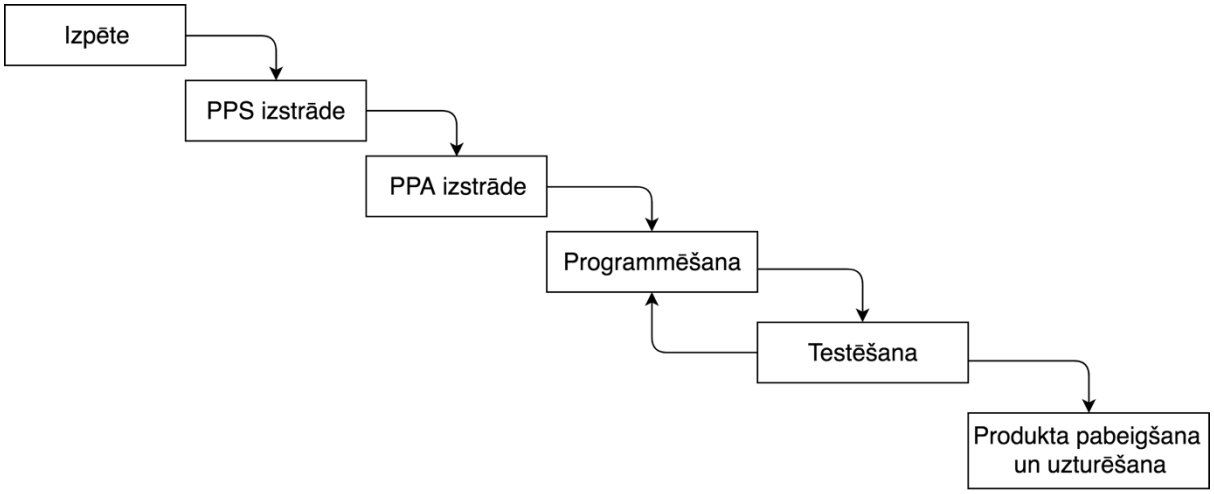
2. PROJEKTA PĀRVALDĪBA

2.1. Projekta organizācija

Projekts tiek organizēts pēc pielāgota iteratīvā izstrādes modeļa (skat. 2.1. att.). Projekta sākumā tiek veikta izpēte, kurā tiek noteiktas iespējas, tiek gūtas zināšanas par *Ruby* paplašinājumu izstrādi, kā arī analizēti citi *Ruby* paplašinājumi. Nākamajā posmā tiek izstrādāta programmatūras prasību specifikācija. Kad tiek apkopotas visas *Kakimasu* prasības, tad tiek izstrādāts programmatūras projektējuma apraksts. Posmā, kad ir zināmas programmatūras prasības un izstrādāts programmatūras projektējums, tiek sākta programkoda izstrāde. Programkoda izstrādes laikā tiek veiktas vairākas testēšanas, kurās tiek pārbaudītas pabeigtās funkcijas, un tās analizētas. Kad paplašinājums tiek pabeigts, tad tiek veikta pēdējā testēšana, lai pārliecinātos, ka viss tiešām strādā tā kā tas norādīts PPS.

Pirms projekta sākšanas tiek novērtēta darbietilpība un izstrādāts aptuvenais laika plāns, pie kura izstrādātājs cenšas pieturēties.

Projekts tiek izstrādāts *Ruby on Rails* vidē. Pirmkoda konfigurāciju pārvaldībai tiek izmantots GitHub repozitorijs. Kvalifikācijas darbs tiek noformēts izmantojot Microsoft Word 365.

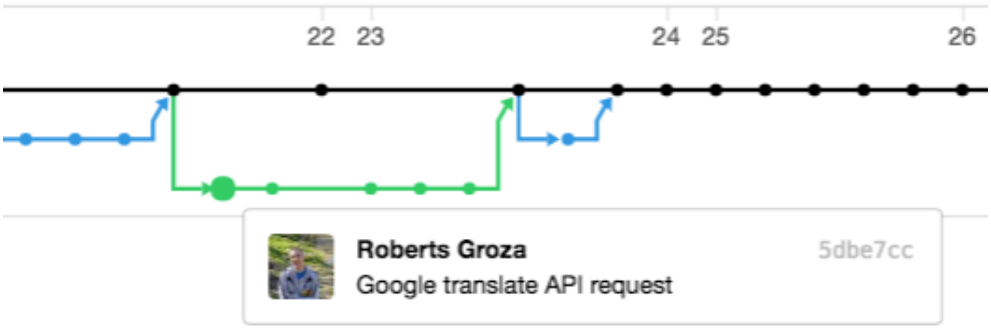


2.1. att. **Projekta organizācija**

2.2. Konfigurāciju pārvaldība

Programmatūras pirmkoda konfigurāciju pārvaldībai tiek izmantota GitHub versiju kontroles sistēma. Katras funkcijas izstrādei tiek veltīts savs zars, un tikai tad kad funkcija tiek izstrādāta un pārbaudīta, šis zars tiek sapludināta ar "master" zaru (skat. 2.2. att.), lai funkcijas izstrādes laikā nejauši izmainīta citu funkciju darbība.

Kakimasu repozitorijā tiek glabāta arī lietošanas pamācība, lai izstrādātājam būtu skaidrs, kā paplašinājums tiek instalēts, kā ir jāizmaina kods, un kā izmantot izstrādātājiem iebūtvētās funkcijas.



2.2. att. **Repozitorija 'zarošanās' jaunas funkcijas izstrādes laikā**

2.3. Kvalitātes nodrošināšana

Produkta kvalitātes nodrošināšanai tiek veiktas šādas darbības:

- Visur produkta pirmkoda izstrādē tiek izmantots vienots programmēšanas stils.
- Izstrādē tiek ievēroti Ruby koda noformēšanas standarti [7] (Tiek izmantots *Rubocop* [8], kas pārbauda programkoda stilu un sintaksi).
- Programmatūras kods satur komentārus.
- Īpaša uzmanība tiek pievērsta tam, lai izstrādāto kodu varētu viegli modificēt.
- Katra funkcija tiek izstrādāta savā zarā.

2.4. Darbietilpības novērtējums

Projekta sākumā tiek veikts darbietilpības novērtējums, balstoties uz personīgo pieredzi, kā arī konsultējoties ar pieredzējušu *Ruby* izstrādātāju (skat. 2.1. tabulu). Personīgais darbietilpības novērtējums tika noteikts pēc formulas $\frac{Optimistiskais + 4Reālistiskais + Pesimistiskais}{6}$.

Rēķinot darbietilpības novērtējumu, tiek pieņemts, ka vienā personmēnesī ir aptuveni 20 persondienas (160 personstundas). Pēc personīgā novērtējuma projekts tika novērtēts aptuveni uz 3,1 personmēnesi, taču pēc eksperta novērtējuma uz 3,2 personmēnešiem. Plānots, ka darbs tiks pabeigts laikā, jo darbs pie projekta tika sākts 20.02.2017

2.1. tabula

Projekta darbietilpības novērtējums

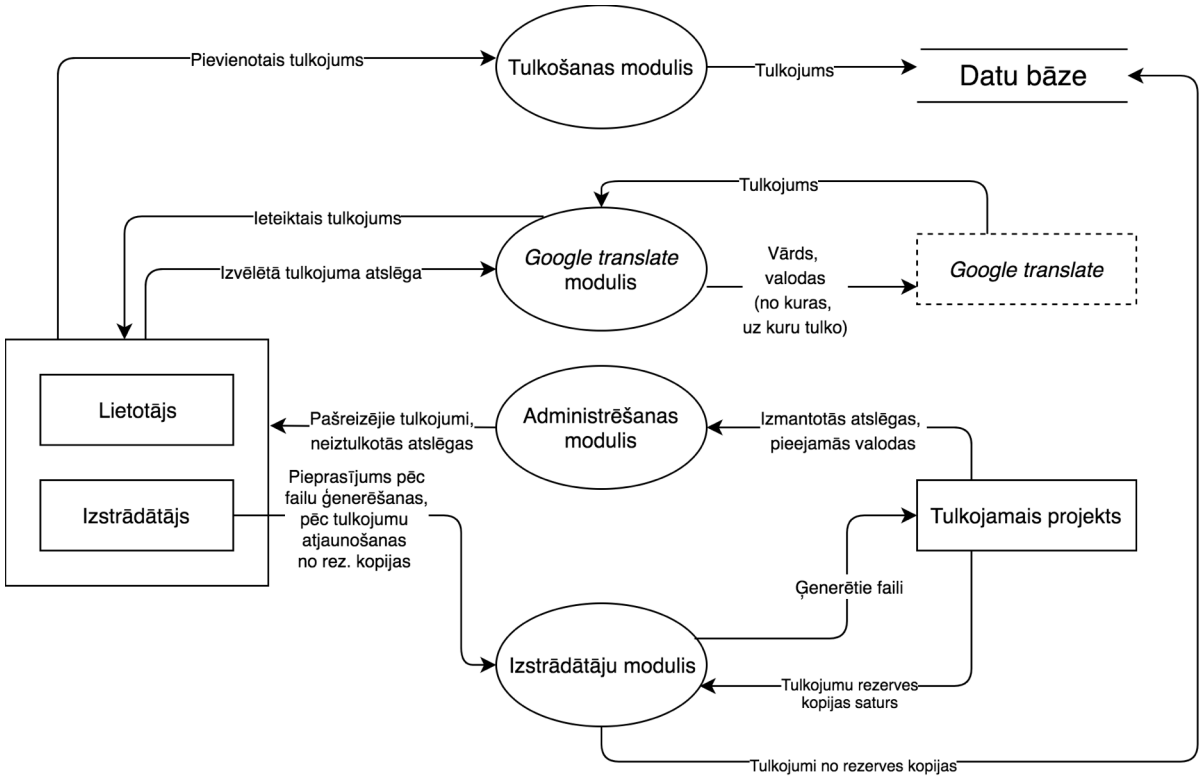
Projekta posms	Personīgais novērtējums			Eksperta novērtējums (h)
	Optimistiskais (h)	Reālistiskais (h)	Pesimistiskais (h)	
Izpēte	16	24	40	24
PPS izstrāde	40	56	72	56
PPA izstrāde	56	72	96	80
Vides uzstādīšana	8	16	20	16
Paplašinājuma pamatstruktūra	16	24	40	32
Tulkojumu pievienošana	64	80	96	80
Pievienoto tulkojumu atskaites	48	56	80	64
Google translate atbalsta pievienošana	24	32	48	32
Failu ģenerēšana rediģēšanai	8	16	24	8
Rezerves kopijas	32	48	72	64
Lietotāja saskarne	24	40	64	40
Testēšanas dokumentācija un testēšana	16	24	32	16
Kopā:	**352**	**488**	**684**	**512**
Novērtējums (personmēneši)	**3,1125**			**3,2**

3. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA

3.1. Funkcionālās prasības

3.1.1. Funkciju dalījums moduļos

Kā redzams 3.1. tabulā un 3.1. attēlā, izstrādātais paplašinājums ir iedalāms 4 funkciju moduļos – tulkošanas modulī, *Google translate* modulī, administrēšanas modulī, izstrādātāju modulī. Funkcijas tiek dalītas pēc tā, kam tās tiek paredzētas. Pirmie 3 moduļi ir paredzēti tulkojumu pārvaldei un tās paredzētas abām lietotāju grupām, izstrādātājiem un lietotājiem. Izstrādātāju moduļa funkcijas ir paredzētas tikai izstrādātajiem – programmētājiem, kas strādā pie projekta, jo tās tiek izsauktas no komandrindas, pie kuras pieeja ir tikai programmētājiem.



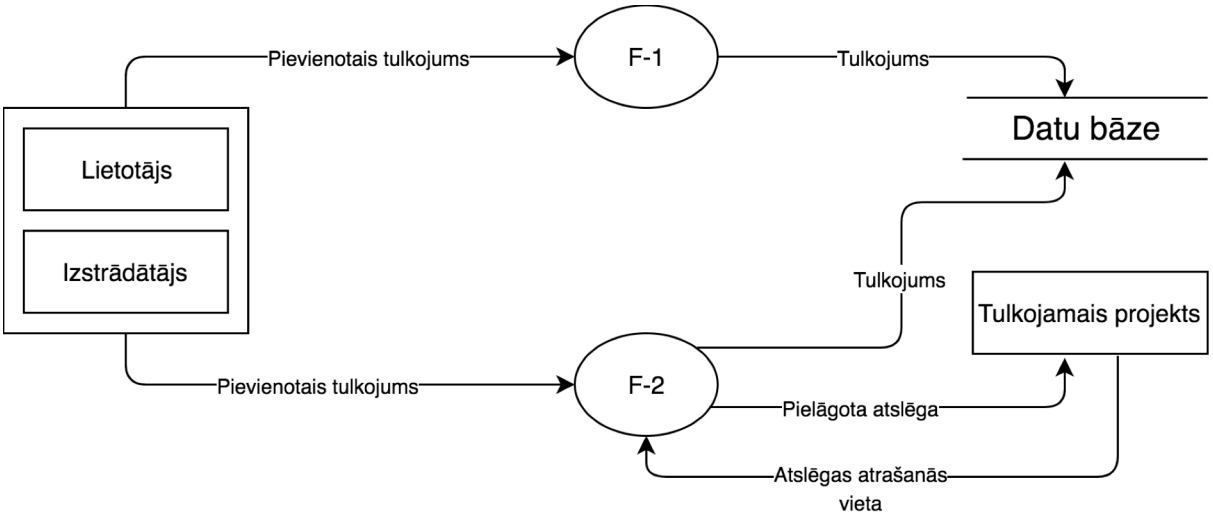
3.1. att. **1. līmeņa datu plūsmu diagramma**

3.1. tabula

Funkciju dalījums moduļos

Modulis	ID	Funkcija	Lietotāju grupas
Tulkošanas modulis	F-1	Tulkojuma pievienošana	Lietotājs, izstrādātājs
	F-2	Unikāla tulkojuma pievienošana vienādām atslēgām	Lietotājs, izstrādātājs
Google translate modulis	F-3	*Google translate* piedāvātie tulkojumi	Lietotājs, izstrādātājs
Administrēšanas modulis	F-4	Pievienoto tulkojumu atskaite	Lietotājs, izstrādātājs
	F-5	Visu projektā izmantoto atslēgu atrašana	Lietotājs, izstrādātājs
	F-6	Neiztulkoto atslēgu atskaite	Lietotājs, izstrādātājs
	F-7	Pilnībā iztulkoto atslēgu procents	Lietotājs, izstrādātājs
Izstrādātāju modulis	F-8	Skatu ģenerēšana	Izstrādātājs
	F-9	Piekļuves tiesību failu ģenerēšana	Izstrādātājs
	F-10	Tulkojumu rezerves kopijas izveide	Izstrādātājs
	F-11	Tulkojumu atjaunošana no rezerves kopijas	Izstrādātājs

3.1.2. Tulkošanas modulis



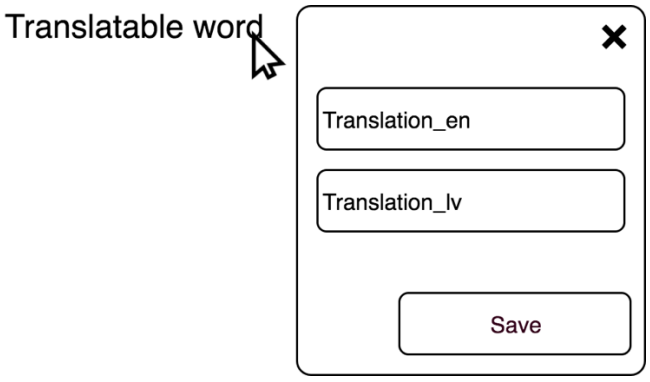
3.2. att. **2. līmeņa datu plūsmu diagramma "tulkošanas modulis"**

Tulkošanas modulī esošās funkcijas atbild par tulkojuma pievienošanu. Kā redzams attēlā 3.2., funkcijas saņem tulkojumus no lietotājiem un saglabā tos datu bāzē. Ar attiecīgā moduļa funkcijām (3.2. un 3.3. tabula) darbojas visi, kam ir piekļuves tiesības tulkojumu pievienošanai.

3.2. tabula

F-1 Tulkojuma pievienošana

Raksturojums
Funkcija paredzēta tam, lai atslēgai varētu pievienot tulkojumu.
Ievaddati
Lietotājs ievada tulkojumu (simbolu virkne, nav obligāta, var saturēt visus simbolus, ieskaitot pieturzīmes, pēdiņas un īpašos simbolus).
Apstrāde
1. Lietotājs novieto kursoru uz tulkojamā vārda (skat. 3.3. att.). 2. Lietotājs ievada tulkojumus un nospiež pogu “save”. 3. Tiek pārbaudīts, vai tulkojums ir ievadīts. • Ja formā nekas netiek ievadīts, tad nekas netiek izmainīts. • Ja ir ievadīta simbolu virkne, tad tulkojums tiek saglabāts un lapas uzreiz tiek izmainīts - parādās jaunais tulkojums.
Izvaddati
Nav
Paziņojumi
Nav

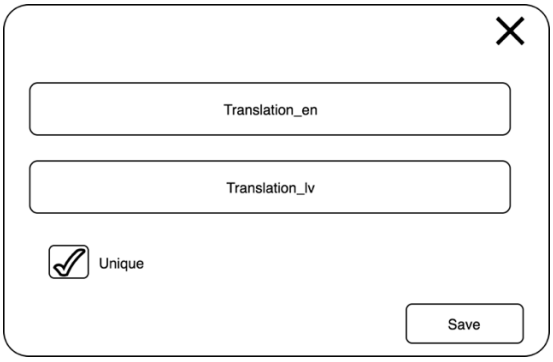


3.3. att. **Tulkojuma ievades formas uzmetums**

3.3. tabula

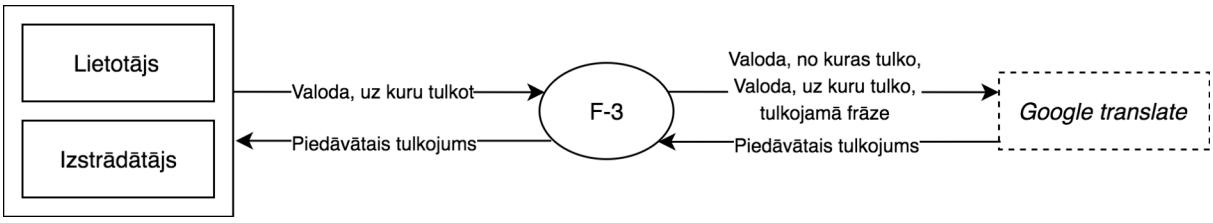
F-2 Unikāla tulkojuma pievienošana vienādām atslēgām

Raksturojums
Funkcija ļauj vienādām atslēgām dažādos failos piešķirt dažādus tulkojumus.
Ievaddati
Lietotājs atzīmē izvēles rūtiņu *unique*, un ievada tulkojumu (simbolu virkne, nav obligāta, var saturēt visus simbolus, ieskaitot pieturzīmes, pēdiņas un īpašos simbolus).
Apstrāde
1. Lietotājs ievada tulkojumu, atzīmē izvēles rūtiņu un nospiež pogu "save" (skat. 3.4. att.). 2. Tiek pārbaudīts, vai ir atzīmēts, ka atslēga ir *unique.* • Ja tiek atzīmēts, ka šī atslēga ir unikāla, tad atslēga tiek padarīta unikāla. 3. Funkcija pārbauda, vai lietotājs kaut ko ir ievadījis. • Ja lietotājs ievada simbolu virkni, tad tiek saglabāts jaunais tulkojums, un tiek rādīts paziņojums Nr. 1. • Ja lietotājs neko neizmaina, taču atzīmē, ka šī atslēga ir unikāla, tad tā tāpat tiek saglabāta kā unikāla, un tiek rādīts paziņojums Nr. 1. • Ja netiek atzīmēts, ka šī atslēga ir unikāla, un lietotājs neko neievada, tad funkcija neko neizmanina, un tiek rādīts paziņojums Nr. 2.
Izvaddati
Nav
Paziņojumi
1. "Translation successfully saved!" 2. "Nothing was changed."



3.4. att. **Tulkojuma ievades formas ar atribūtu "unique" uzmetums**

3.1.3. *Google translate* modulis



3.5. attēls: **2. līmeņa datu plūsmu diagramma "Google translate modulis"**

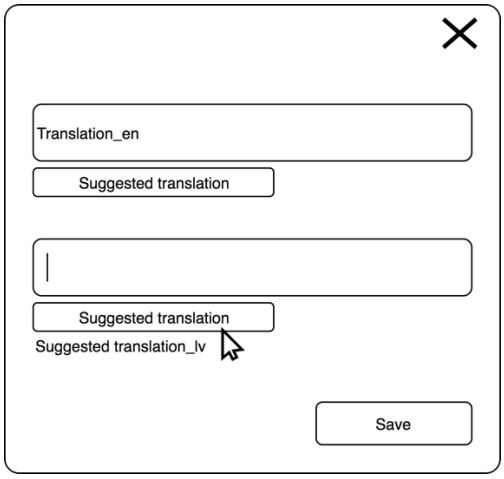
Modulis nodrošina *Google tranlsate* servisa atbalstu lietotājam. Kā redzams attēlā 3.5., modulī esošā funkcija sadarbojas ar *Google translate* servisu, un lietotājam parāda izvēlētās frāzes piedāvāto tulkojumu. Modulī ir tikai viena funkcija - F-3 (skat. 3.4. tabulu)

3.4. tabula

F-3 "Google translate" piedāvātie tulkojumi

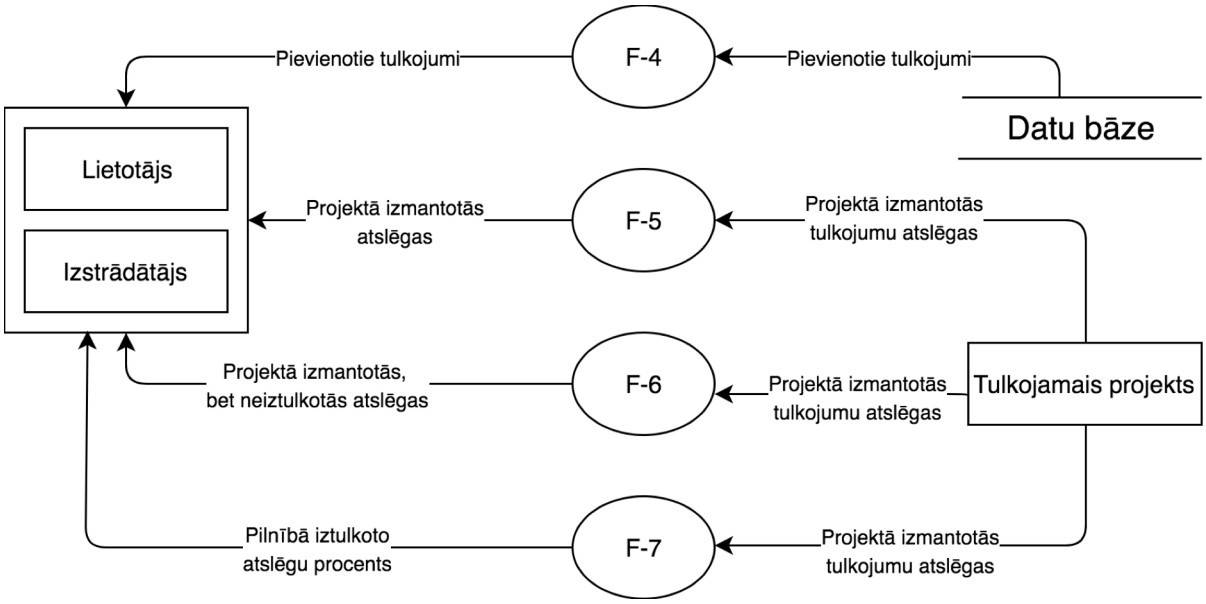
Raksturojums
Funkcija rāda lietotājam *google translate* piedāvāto tulkojumu.
Ievaddati
Frāze vai vārds, kuru tulkot un valoda, uz kuru tulkot.
Apstrāde
1. Lietotājs tulkošanas formā nospiež uz pogas "suggested translations" (skat. 3.6. att.). 2. Tiek pārbaudīts, vai tulkojuma atslēgai kādā no valodām ir tulkojums. • Ja atslēgai nav norādīts neviens tulkojums, tad rāda paziņojumu Nr. 2. 3. Tiek pārbaudīts, vai ir savienojums ar internetu. • Ja nevar nodibināt savienojumu ar servisu, tad rāda paziņojumu Nr. 1. 4. Lietotājam tiek parādīts *google translate* atgrieztais tulkojums.
Izvaddati
Google translate atgrieztais tulkojums.

Paziņojumi
1. “Unable to connect google translate service. Please make sure that you are connected to the internet!” 2. “There is no word to translate. Please give at least one translation to the key.”



3.6. att. **Tulkojuma formas ar “Suggested translation” pogu skice**

3.1.4. Administrēšanas modulis



3.7. att. **2. līmeņa datu plūsmu diagramma “Administrēšanas modulis”**

Modulī esošās funkcijas pārsvarā iegūst datus no tulkojamā projekta (kā redzams attēlā 3.7.). Modulis paredzēts tam, lai lietotājs varētu ērti pārvaldīt projektā esošos tulkojumus – ir pieejama pievienoto tulkojumu atskaite (3.5. tabula), visu projektā izmantoto tulkojumu atslēgu atrašana (3.6. tabula), pilnībā neiztulkoto atslēgu atskaite (3.7. tabula) un pilnībā iztulkoto atslēgu procents (3.8. tabula).

3.5. tabula

F-4 Pievienoto tulkojumu atskaite

Raksturojums
Funkcija parāda visus lietotāja pievienotos tulkojumus, lai varētu atrast pretrunas (ja tādas ir).
Ievaddati
Nav
Apstrāde
1. Tiek meklēti visi ar *Kakimasu* pievienotie tulkojumi. • Ja tulkojumi tika atrasti, tad tie tiek parādīti tabulas veidā. • Ja neviens tulkojums netika atrasts, tad tiek rādīts ziņojums Nr. 1.
Izvaddati
Visi lietotāja ievadītie tulkojumi.
Paziņojumi
1. “No translations were found.”

3.6. tabula

F-5 Visu projektā izmantoto atslēgu atrašana

Raksturojums
Funkcija lietotājam parāda visas projektā izmantotās atslēgas, un to atrašanās vietu.
Ievaddati
Nav
Apstrāde

3.6. tabulas turpinājums

1. Tiek meklētas visas projektā izmantotās tulkojumu atslēgas. • Ja tiek atrastas tulkojumu atslēgas, tad, tabulas veidā, tiek parādīti visi to tulkojumi, atrašanās vieta un "edit" poga tulkojumu mainīšanai. • Ja vispār netiek atrasta neviena atslēga, tad tiek rādīts paziņojums Nr. 1.
Izvaddati
Tabula ar visām projektā izmantotajām atslēgām.
Paziņojumi
1. "There was no translation keys found".

3.7. tabula

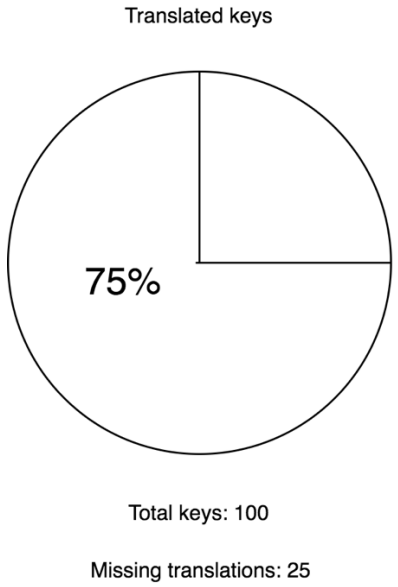
F-6 Neiztulkoto atslēgu atskaite

Raksturojums
Funkcija lietotājam parāda visas tulkojumu atslēgas, kurām vismaz vienā valodā nav tulkojuma.
Ievaddati
Nav
Apstrāde
1. Tiek meklētas visas tās atslēgas, kurām kādā no valodām trūkst tulkojums. • Ja tiek atrastas tādas atslēgas, tad, tabulas veidā, tiek parādīti visi to tulkojumi, trūkstošie tulkojumi tiek iekrāsoti sarkanā krāsā, tiek parādīta atslēgas atrašanās vieta, kā arī "edit" poga tulkojumu pievienošanai. • Ja nav nevienas tādas atslēgas, tad tiek parādīts paziņojums Nr. 1.
Izvaddati
Tabula ar neiztulkotajām atslēgām.
Paziņojumi
1. "There is no untranslated key!"

3.8. tabula

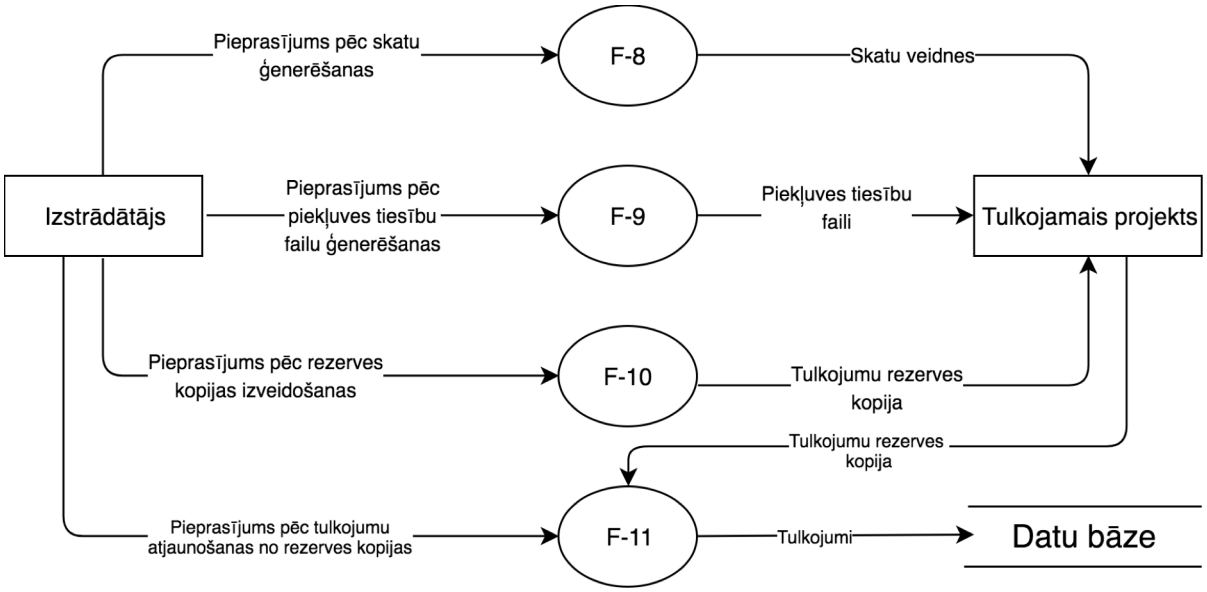
F-7 Pilnībā iztulkoto atslēgu procents

Raksturojums
Funkcija rāda, cik procenti no visām atslēgām ir iztulkotas visās valodās, lai radītu lietotājam priekšatu par to, cik ļoti projekts ir iztulkots.
Ievaddati
Nav
Apstrāde
1. Tiek saskaitīts, cik kopā projektā ir izmantotas atslēgas. 2. Tiek saskaitītas tās atslēgas, kurām kādā no valodām trūkst tulkojums. 3. Aprēķins, cik procentuāli no visām atslēgām, ir pilnībā iztulkots. 4. Tiek izveidota apļveida diagramma ar pilnībā iztulkoto atslēgu procentu (skat. 3.8 att.)
Izvaddati
Iztulkoto atslēgu īpatsvars apļveida diagrammas veidā, kopējais projektā izmantoto atslēgu skaits un pilnībā neiztulkoto atslēgu skaits.
Paziņojumi
Nav



3.8. Att. **F-7 atgrieztās diagrammas uzmetums**

3.1.5. Izstrādātāju modulis



3.9. att. **2. līmeņa datu plūsmu diagramma "Izstrādātāju modulis"**

Modulī esošās funkcijas nodrošina to, ka izstrādātājs var rediģēt *Kakimasu* iestatījumus (Skatu rediģēšana (3.9. tabula), dizaina pielāgošana, piekļuves tiesību definēšana (3.10. tabula)). Modulī ietilpst arī funkcija rezerves kopiju veidošanai (3.11. tabula) un funkcija tulkojumu atjaunošanai no rezerves kopijas (3.12. tabula). Kā redzams 3.9. attēlā modulis ir paredzēts tikai izstrādātāju grupai.

3.9. tabula

F-8 Skatu ģenerēšana

Raksturojums
Funkcija nepieciešama tam, lai izstrādātājs varētu rediģēt paplašinājuma skatus, un pielāgot tos savam projekta
Ievaddati
Izstrādātājs komandrindā ievada pieprasījumu pēc skatu ģenerēšanas (*rails generate kakimasu:views*).
Apstrāde
1. Funkcija projektā izveido paplašinājuma skatus, un rāda ziņojumu Nr. 1.

3.9. tabulas turpinājums

Izvaddati
Skatu veidnes rediģēšanai.
Paziņojumi
1. “Views was successfully generated!”

3.10. tabula

F-9 Piekļuves tiesību failu ģenerēšana

Raksturojums
Funkcija nepieciešama tam, lai izstrādātājs varētu pielāgot lietotāju piekļuves tiesības savam projektam.
Ievaddati
Izstrādātājs komandrindā ievada pieprasījumu pēc tiesību failu ģenerēšanas (*rails generate kakimasu:policy*).
Apstrāde
1. Funkcija projektā izveido failus piekļuves tiesību konfigurēšanai, un izdrukā paziņojumu nr. 1.
Izvaddati
Faili piekļuves tiesību rediģēšanai.
Paziņojumi
1. “Policies was successfully generated!”

3.11. tabula

F-10 Tulkojumu rezerves kopijas izveide

Raksturojums
Funkcija nepieciešama tam, lai varētu izveidot rezerves kopiju visiem ar *Kakimasu* pievienotajiem tulkojumiem

3.11. tabulas turpinājums

Ievaddati
Izstrādātājs komandrindā ievada pieprasījumu veidot rezerves kopiju (*rails generate kakimasu:backup*).
Apstrāde
1. Funkcija atrod visu ar *Kakimasu* pievienotos tulkojumus un saglabā tos YAML failā, un rāda paziņojumu Nr. 1. • Ja netiek atrasta neviens pievienotais tulkojums, tad tiek rādīts paziņojums nr. 2.
Izvaddati
YAML fails ar visiem saglabātajiem *Kakimasu* tulkojumiem.
Paziņojumi
1. “Translations successfully stored!” 2. “Backup can not be done. There are no translations!”

3.12. tabula

F-11 Tulkojumu atjaunošana no rezerves kopijas

Raksturojums
Funkcija nepieciešama tam, lai varētu atjaunot visus ar *Kakimasu* pievienotos tulkojumus no rezerves kopijas.
Ievaddati
Izstrādātājs komandrindā ievada pieprasījumu pēc tulkojumu atjaunošanas no rezerves kopijas (*rails generate kakimasu:restore_backup*).
Apstrāde
1. Funkcija saņem tulkojumu rezerves kopiju. • Ja rezerves kopija netiek atrasta, tiek rādīts paziņojums nr. 1. 2. Visi tulkojumi tiek saglabāti datu bāzē un tiek rādīts paziņojums nr. 2.
Izvaddati
Nav

3.12. tabulas turpinājums

Paziņojumi
1. “Backup file can’t be found!” 2. “Translations successfully restored!”

3.2. Lietotāju saskarnes prasības

Paplašinājuma ekrānformām jābūt viegli saprotamām un ātri apgūstamām. Jebkurā *Kakimasu* skatā ir jābūt hipersaitei uz citu skatu. Lietotāju saskarnē ir paredzēti šādi skati:

1. *Kakimasu* sākumlapa – Atrodas poga, kas iespējo vārdu tulkošanu ar “popover” formu, kas tiek parādīta lietotājam, kad uz tulkojamā vārda tiek novietots kursors. Tiek rādīts, cik daudz tulkojumu atslēgām ir pievienoti tulkojumi visās valodās (F-7 funkcija).
2. “Key management” skats – Pieejama tabula ar tulkojumu atslēgām, kurām kādā no valodām nav tulkojuma. Pieejama tabula ar visām projektā izmantotajām tulkojumu atslēgām.
3. “Translation panel” skats – Pieejama tabula, kurā attēloti visi ar *Kakimasu* pievienotie tulkojumi. Pieejama arī “manuālo” tulkojumu pievienošanas forma, ar kuru var pievienot tulkojumu, zinot tulkojuma atslēgu (paredzēta to tulkojumu atslēgu tulkošanai, kuras neparādās skatos, piemēram, kļūdu paziņojumi).

3.3 Nefunkcionālās prasības

3.3.1. Funkcionālā piemērotība

- Paplašinājumam jānodrošina tulkojumu saglabāšana arī pēc servera restartēšanas.
- Paplašinājumam jānodrošina visas funkcijas, izņemot F-3, arī tad, ja nav pieejas interneta savienojumam.

3.3.2. Veiktspēja

- Paplašinājuma darbība nedrīkst ietekmēt projekta veiktspēju, kad ar to nedarbojas.
- Ar *Kakimasu* pievienotajiem tulkojumiem jātiek ielādētiem tikpat ātri, kā tradicionālajā veidā pievienotajiem tulkojumiem.

3.3.3. Saderība

- Paplašinājumam jābūt saderīgam ar *Ruby 2.3.0* un jaunākām versijām.
- Jāatbalsta Google translate serviss.

3.3.5. Drošība

- Izmantot šo paplašinājumu, kā arī paplašinājuma skatiem var piekļūt tikai lietotāji ar atbilstošajām tiesībām.

3.3.6. Uzturamība

- Programkodam jābūt komentētam, lai izstrādātājs to varētu viegli uzlabot.

3.3.7. Uzticamība

- Paplašinājumam jānodrošina iespēja veidot rezerves kopijas un atjaunot no tām tulkojumus.

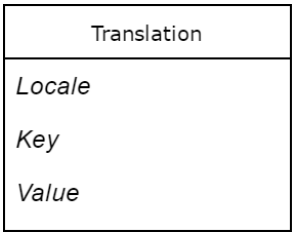
4. PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS

Kā programmēšanas valoda tiek izvēlēta *Ruby* 2.3.2, jo tā ir jaunākā *Ruby* versija, kas ir stabila. Datu bāzes funkcijas nodrošināšanai tiek izvēlēts Redis serveris [9], jo tas atvērtā pirmkoda risinājums, kas ir populārs izstrādātāju vidū iekšējās atmiņas *hash* glabāšanai (iemesls tuvāk aprakstīts 4.1. nodaļā). Lietotāju saskarnes izveidē izmantotas Bootstrap 3 un jQuery bibliotēkas.

Piekļuves tiesību nodrošināšanai tiek izmantots *Ruby* paplašinājums *Pundit* [10]. Tiek izmantots *Pundit* paplašinājums tāpēc, ka tas ir ērts, objektorientēts risinājums, kas strauji gūst popularitāti *Ruby* izstrādātāju vidū.

4.1. Tulkojumu glabāšana

Ruby paplašinājumā *I18n* [11], kas pēc noklusējuma tiek izmantots tulkojumu nodrošonāšanai *Ruby on Rails* projektā, tulkojums ir objekts, kas sastāv no valodas koda, atslēgas un vērtības (skat. 4.1. att.).



4.1. att. **Tulkojuma objekta struktūra**

Pēc noklusējumi *I18n* paplašinājums tulkojumus glabā YAML failos, kas nodrošina "cieti" ieprogrammētus tulkojumus. Lai pievienotu tulkojumus dinamiski, būtu nepieciešamas funkcijas, kas ieraksta tulkojumus YAML failos, taču tām būtu liels risks sabojāt faila struktūru (YAML failos ļoti liela nozīme ir ierobījumam (no angļu val. *indentation*), ja tas tiek sajaukts, tad *I18n* vairs nav spējīgs korekti darboties). 4.2. attēlā parādīts, kā pēc noklusējuma tiek glabāti tulkojumi YAML failā.

4.2. att. **Tulkojumu pieraksts YAML failā**

I18n jau piedāvā gatavus risinājumus – *I18n_backend*, ja gadījumā izstrādātājs vēlas tulkojumus sistēmā organizēt savādākā veidā. Tulkojumu pievienošanu dinamiski spēj nodrošināt *key_value backend*, kas dod iespēju glabāt tulkojumus datu bāzē vai kešatmiņā. Pēc noklusējuma *I18n* izmanto *simple_backend*, kas no YAML failiem ielādē tulkojumus kešatmiņā, kā *hash* objktu. Pluss tādai pieejai ir ātrdarbība, jo tulkojumi visu *Rails servera* darbības laiku ir jau ielādēti atmiņā, un tie pēc katra pieprasījuma nav no jauna jāielādē.

Ja tulkojumus glabātu datu bāzē, tad tie pēc katra pieprasījuma tiktu ielādēti no jauna, tāpēc katrs pieprasījums būs par dažām milisekundēm ilgāks. Tā kā ir svarīgi, lai *Kakimasu* neietekmētu projekta ātrdarbību, veicot pieprasījumus, kas nav saistīti ar paplašinājumu, tad tulkojumi tiks glabāti kešatmiņā kā *hash* objekts. Problēma, kas rodas saistībā ar šo pieeju, ir tā, ka pēc servera restartēšanas, kešatmiņa tiek notīrīta, un visi tulkojumi pazustu, tāpēc ir nepieciešams risinājums izveidotā *hash* objekta glabāšanai. Šajai problēmai pastāv risinājums – *Redis* serveris. *Redis* serveris tiek izmantots *hash* objektu glabāšanai un, līdz ar to, nebūtu vairs jāuztraucas par kešatmiņas notīrīšanos pēc servera restartēšanas. Pluss *Redis* serverim ir arī tas, ka ir *Ruby* paplašinājums *Redis* atbalstam.

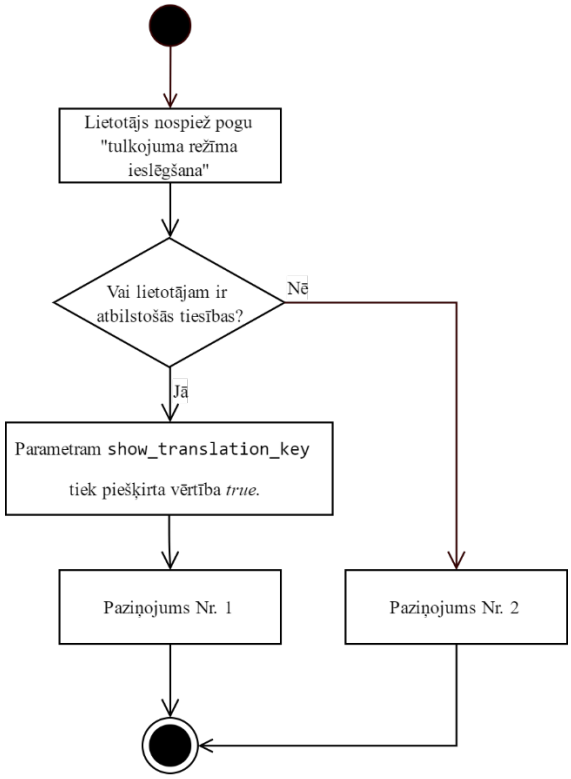
Ja pilnībā pārietu uz *I18n key_value_backend* projektā, kurā jau ir ievadīti tulkojumi YAML failos, tad tie vairs netiktu ņemti vērā, un, būtībā, visi sākotnējie tulkojumi būtu jāpārraksta par jaunu, tāpēc tas radītu nevajadzīgas izmaksas. Divu dažādu *I18n* aizmuguru (*backend*) darbību nodrošina *chain_backend*, tāpēc tiek izlemts, ka tiks izmantota *key_value_backend* un *simple_backend* ķēde, kas nodrošinās gan iespēju ērti dinamiskā veidā pievienot tulkojumu, gan "cieti" ierakstīt tulkojumus YAML failos.

Saistībā ar šādu pieeju rodas ierobežojumi – nepieciešams *Redis* serveris, lai *Kakimasu* varētu veiksmīgi strādāt. Lai gan *Redis* ir atvērtā pirmkoda programmatūra, tomēr ir neliela iespēja, ka programmētājs nevēlās to uzstādīt, tādēļ tuvākā nākotnē ir paredzēts izstrādāt *Kakimasu_active_record*, kas glabātu tulkojumus SQL tipa datu bāzē.

4.2. Procesu projektējums

Šajā nodaļā ir aprakstīts, kā tiek realizētas izstrādātā *Ruby* paplašinājuma *Kakimasu* netriviālākās funkcijas un mehānismi.

4.2.1. "Tulkošanas režīma" ieslēgšana

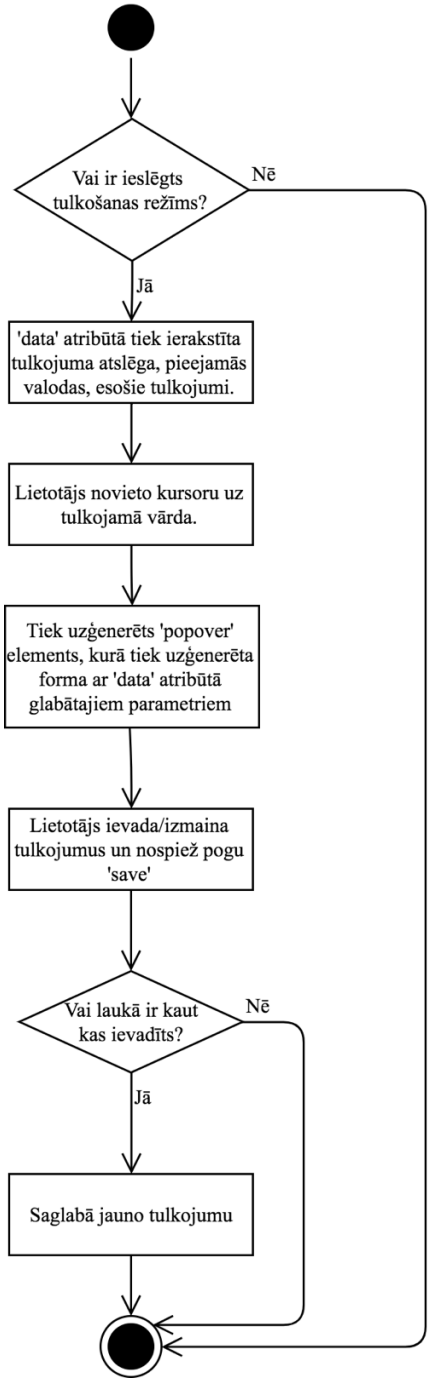


4.3. att. **Tulkošanas režīma ieslēgšana**

"Tulkošanas režīms" tiek ieslēgts, kad lietotājs nospiež pogu "turn on translation mode" (skat. 4.3. att.). Vispirms paplašinājums pārbauda, vai lietotājam, kas izsauca šo funkciju, ir atbilstošās piekļuves tiesības. Ja lietotāja piekļuves tiesības atbilst noteiktajām, tad parametram *show_translation_key*, tiek piešķirta vērtība *true*. Sekmīgas ieslēgšanās gadījumā, lietotājam tiek parādīts paziņojums par veiksmīgu tulkošanas režīma ieslēgšanu, taču neveiksmes gadījumā, tiek parādīts paziņojums par to, ka lietotājam nav atbilstošo piekļuves tiesību. Tulkojuma režīma izslēgšana notiek attiecīgi otrreiz nospiežot "tulkošanas režīma ieslēgšanas" pogu.

Šāda tulkošanas režīma ieslēgšana ir paredzēta, lai lietotājam būtu iespēja izmantot funkciju F-1, kā arī, lai varētu nodrošināt to, ka lietotāji bez nepieciešamajām piekļuves tiesībām nevar pievienot tulkojumus.

4.2.2. F-1 Tulkojuma pievienošana

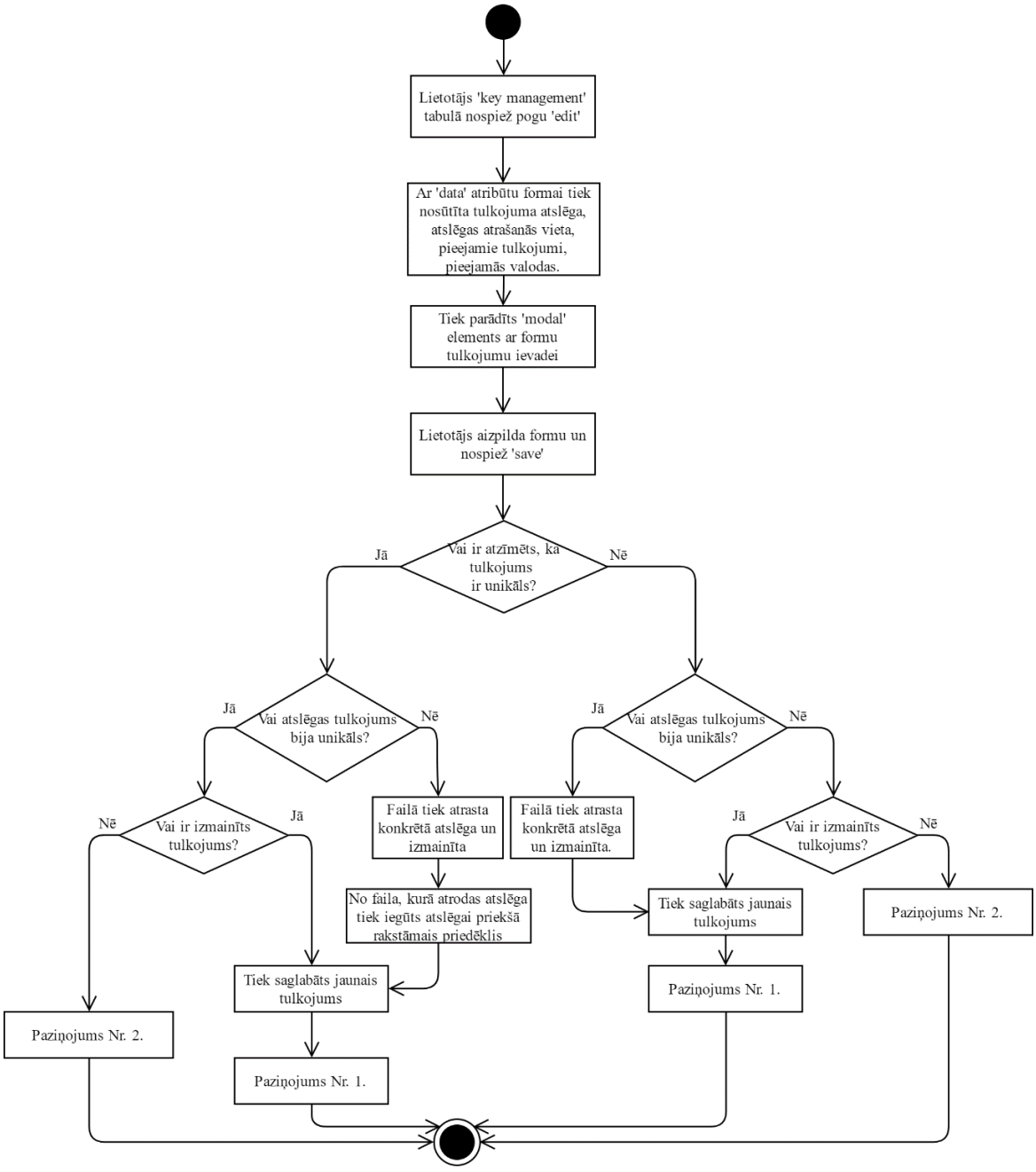


4.4. att. **Tulkojuma pievienošana**

Funkcijas darbība tiek aizsākta tikai tad, ja ir aktivizēts tulkošanas režīms (skat. 4.4. att.). Veiksmīgai funkcijas darbībai, ievades formai ir nepieciešams padot tulkojuma atslēgu, esošos tulkojumus un pieejamās valodas, taču tiešā ceļā tas nav izdarāms. Nepieciešamos parametrus Javascript funkcijai ir iespējams padot ar "data" atribūta palīzību, tas tiek realizēts pārrakstot *I18n* tulkošanas funkciju, kas tulkojuma vietā atgriež "" elementu ar nepieciešamajiem parametriem.

Kad lietotājs novieto kursoru uz tulkojamā vārda, tiek izsaukta Javascript funkcija, kas ģenerē "popover" elementu, kas parādās pie izvēlētā vārda. "Popover" elementā tiek uzģenerēta forma tulkojuma saglabāšanai. Kad lietotājs nospiež saglabāšanas pogu, tad katrā valodā tiek pārbaudīts, vai laukā ir kaut kas ievadīts. Ja lauks ir atstāts tukšs, tad nekas netiek izmainīts, taču, ja laukā ir ievadīta simbolu virkne, tad tiek saglabāts jaunais tulkojums.

4.2.3. F-2 Unikāla tulkojuma pievienošana vienādām atslēgām



4.5. att. **Unikāla tulkojuma pievienošana vienādām atslēgām**

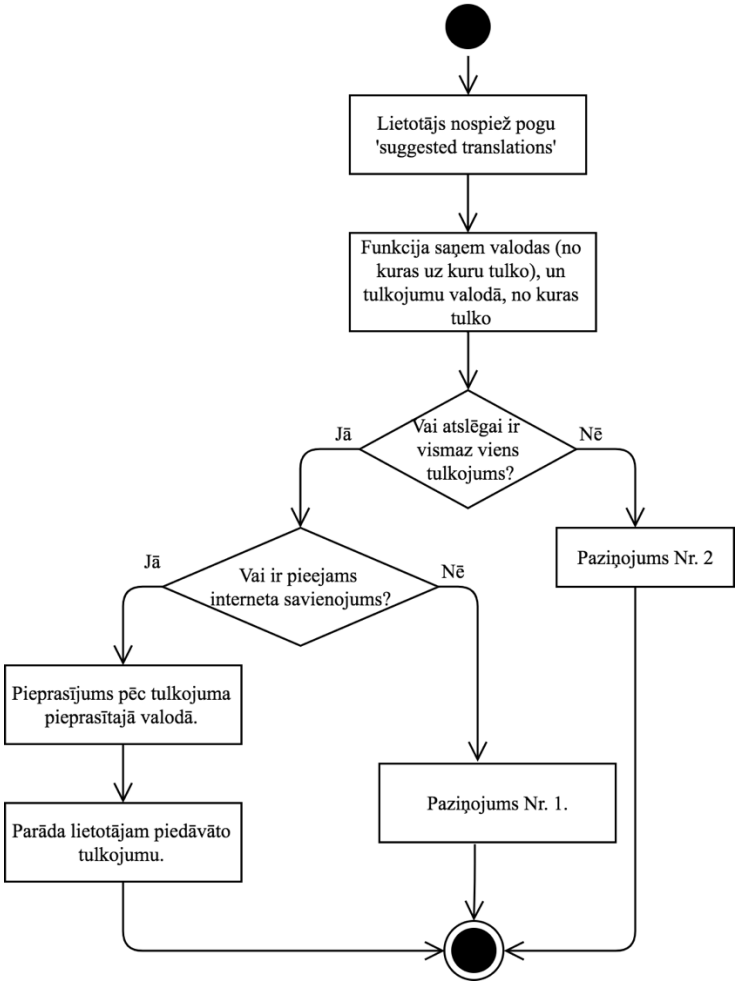
F-2 funkcijas darbības nodrošināšanai tiek izmantots I18n “lazy lookup” mehānisms, kad tulkojuma atslēgai priekšā tiek pielikta faila atrašanās vieta. Piemēram, ja failā ar atrašanās vietu *app/views/books/index.html.erb* ir atslēga *key*, tad šīs atslēgas “lazy lookup” versija ir *books.index.key*. Šis mehānisms ir paredzēts novērst tādus gadījumus, ja divos dažādos skatos ir vienādas tulkojuma atslēgas, bet tām ir dažādi tulkojumi, jo pēc noklusējuma abām atslēgām tiktu piešķirts vienāds tulkojums. Lai *I18n* varētu noteikt, ka šīs atslēgas tulkojums ir jāmeklē pēc “lazy lookup” principa, vietā, kur tā tiek izsaukta, priekšā pieraksta punktu (piemēram, ja normāli tulkojuma atslēgu pieraksta šādi - *t(‘key’)*, tad pēc “lazy lookup” principa to pieraksta šādi - *t(‘.key’)*).

Pievienot unikālu tulkojumu atslēgai var no “key management” tabulas (skat. 4.5. att.), jo, lai izveidotu “lazy lookup” tulkojumu atslēgu ir nepieciešams zināt faila, kurā atrodas atslēga, atrašanās vietu. Šīs tabulas tulkojumu ievades formā atrodas izvēles rūtiņa “unique”. Šī izvēles rūtiņa norāda, vai šī atslēga ir unikāla (pierakstīta pēc “lazy lookup” principa) vai arī visos failos tās tulkojums sakrīt.

Ja lietotājs redz, ka šajai atslēgai visās vietās ir viens tulkojums, taču vēlas, lai konkrētajā failā būtu cits tulkojums, tad pie tulkojuma ievades viņš atzīmē “unique”. Brīdī, kad lietotājs nospiež saglabāšanas pogu, funkcija pārbauda, vai šī atslēga jau iepriekš bija unikāla. Ja iepriekš nebija unikāla, tad funkcija atslēgu tādu padara. Vispirms failā, kurā izmantota konkrētā atslēga, tiek atrasta atslēga, un tai priekšā tiek pierakstīts punkts, lai *I18n* zinātu, ka būs jāizmanto “lazy lookup” princips. Tad no attiecīgā faila atrašanās vietas tiek izveidots priedēklis, kurš tiks rakstīts priekšā atslēgai.

Ja lietotājs vēlās atgriezt to, ka atslēgai visur ir viens tulkojums, tad viņam ir jāpārliecinās, ka izvēles rūtiņa “unique” nav ieķeksēta. Ja funkcija redz, ka tagad nav atzīmēts “unique”, bet iepriekš bija, tad failā tiek atrasta izmantotā atslēga, un tiek noņemts punkts, kas atrodas pirms atslēgas (ja atslēga bija *t(‘.key’)*, tad tā tiek pārveidota atpakaļ par *t(‘key’)*).

4.2.4. F-3 “Google translate” piedāvātie tulkojumi



4.6 att. **“Google translate” piedāvātie tulkojumi**

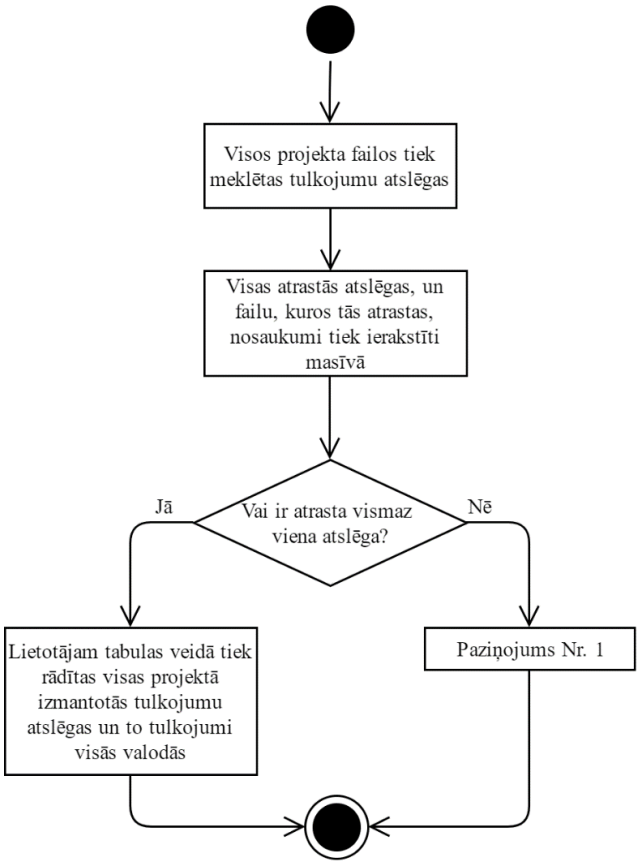
Funkcija tiek izsaukta, kad tulkošanas “popover” logā lietotājs nospiež pogu “suggested translations” (skat. 4.6. att.). Funkcijas darbības pamatā ir *google translate API*. Piemēram, ja mēs vēlētos iztulkot vārdu – “vārds”, no latviešu valodas uz angļu valodu, tad tiktu sūtīts pieprasījums uz adresi *https://translate.googleapis.com/translate_a/single?client=gtx&sl=lv&tl=en&dt=t&q=vārds*. Kā atbilde tiek saņemts masīvs, kura pirmais elements ir populārākais tulkojums (kas mūs visvairāk interesē).

Brīdī, kad lietotājs nospiež šo pogu, funkcija no formas iegūst valodu valodu, uz kuru lietotājs vēlās iztulkot vārdu. Lai funkcija veiksmīgi darbotos, ir nepieciešams, ka atslēgai ir vismaz viens tulkojums, valodā, kas nav tā valoda, uz kuru vēlās tulkot. Ja atslēgai nav neviena tulkojuma, tad lietotājam tiek rādīts paziņojums nr. 2, tāpēc, ka funkcija pēc atslēgas nevar paredzēt, kāds būs tulkojums, jo atslēgas teksts ne vienmēr sakrīt ar tulkojuma tekstu. Ja atslēgai ir tulkojums kādā no valodām, tad tiek noskaidrots, kas tā par valodu, kurā ir tulkojums,

un tad no tās valodas tiek tulkots uz jauno valodu. Ja vairākās valodās ir tulkojumi, tad tiks tulkots no pirmās valodas no augšas, kurai ir tulkojums un kura nav valoda no kuras tulko. Ja lietotājs pieprasa "ieteikto tulkojumu" valodai, kurā vienīgajā atslēgai ir piešķirts tulkojums, tad funkcija vienkārši atgriež to pašu vārdu.

Pirms tiek sūtīts pieprasījums uz *google translate* API, tiek pārbaudīts, vai ir interneta savienojums. Ja interneta savienojuma nav, tad tiek rādīts paziņojums nr. 1. Ja internets ir, tad tiek sūtīts pieprasījums uz *google translate* API un iegūtais rezultāts tiek parādīts lietotājam.

4.2.5. F-5 Visu projektā izmantoto atslēgu atrašana

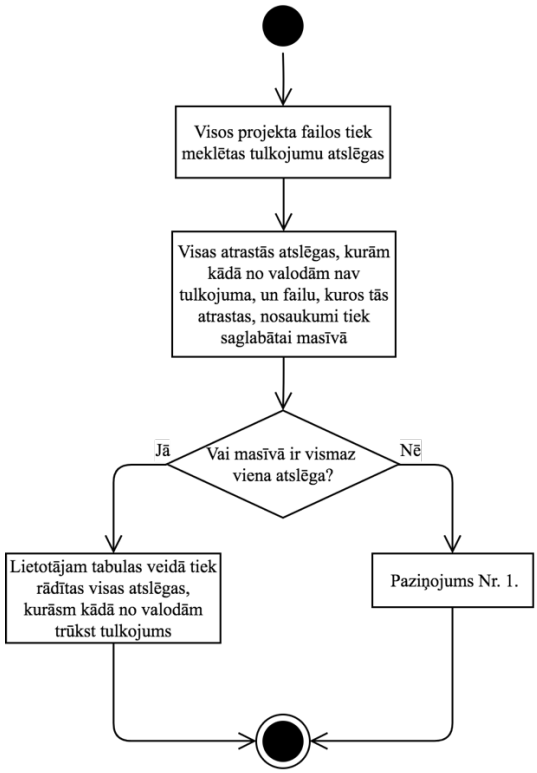


4.8. att. **Visu projektā izmantoto tulkojumu atslēgu atrašana**

Projektā izmantotās atslēgas tiek meklētas pēc regulārās izteiksmes, jo veidu skaits, kā pierakstīt atslēgas tulkojumā ir ierobežots (tātad ir iespējams uzrakstīt visaptverošu regulāro izteiksmi, kas aptver visus gadījumus). Atslēgu meklēšana notiek pa failiem (skat. 4.8. att.). Visas failā esošās atslēgas tiek saglabātas masīvā, un masīva beigās tiek saglabāta izskatītā faila atrašanās vieta. Meklēšanas beigās tiek iegūts masīvs, kas sastāv no vairākiem masīviem – katram failam savs masīvs.

Ja nav atrasta neviena atslēga, tad lietotājam, protams, par to tiek paziņots (tā var notikt, ja projekts, pie kura ir pievienots *Kakimasu* nav ticis tulkots). Savukārt, ja masīvā ir vismaz viena atslēga, tad visas atslēgas lietotājam tiek parādītas tabulas veidā.

4.2.6. F-6 Neiztulkoto atslēgu atskaite

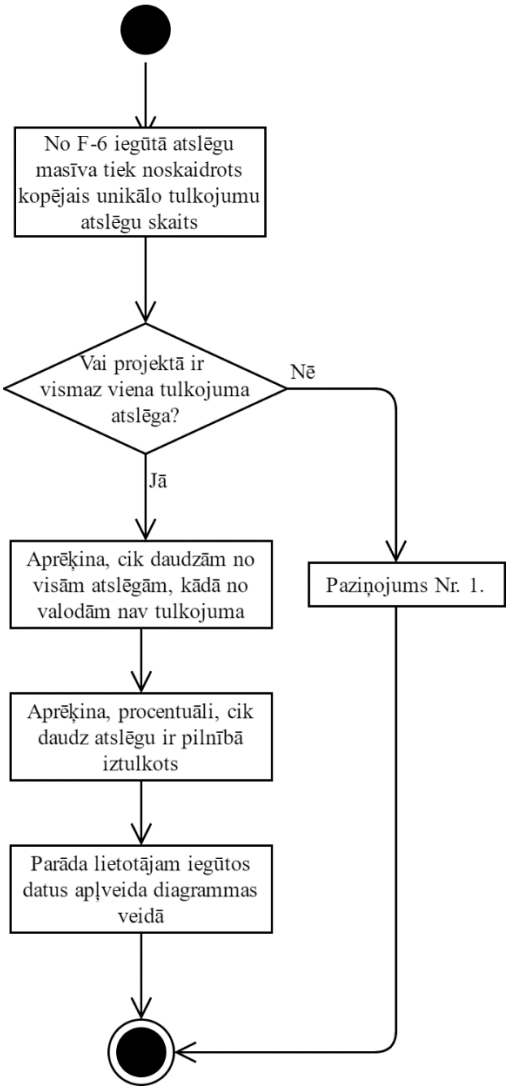


4.9. att. **Neiztulkoto atslēgu atrašana**

Funkcija F-6 darbojas pēc līdzīga principa, kā darbojas funkcija F-5, taču ir viena būtiska atšķirība – atrastās atslēgas tiek filtrētas (skat. 4.9. att.). Pirms atslēga tiek saglabāta masīvā, tiek pārbaudīts, vai atslēgai ir tulkojumi visās valodās. Ja kādā no valodām atslēgai nav tulkojuma, tad atslēga tiek saglabāta masīvā. Masīva beigās, tāpat kā funkcijā F-5, tiek saglabāta faila atrašanās vieta. Beigās, tāpat kā iepriekšējajā funkcijā tiek iegūts masīvs, kas sastāv no vairākiem masīviem.

Ja masīvā nav nevienas atslēgas, tad lietotājam tiek rādīts paziņojums nr. 1. Savukārt, ja masīvā ir vismaz viena atslēga, tad tās lietotājam tiek parādītas tabulas veidā, kur iztrūkstošā tulkojuma šūna tiek iekrāsota sarkanā krāsā.

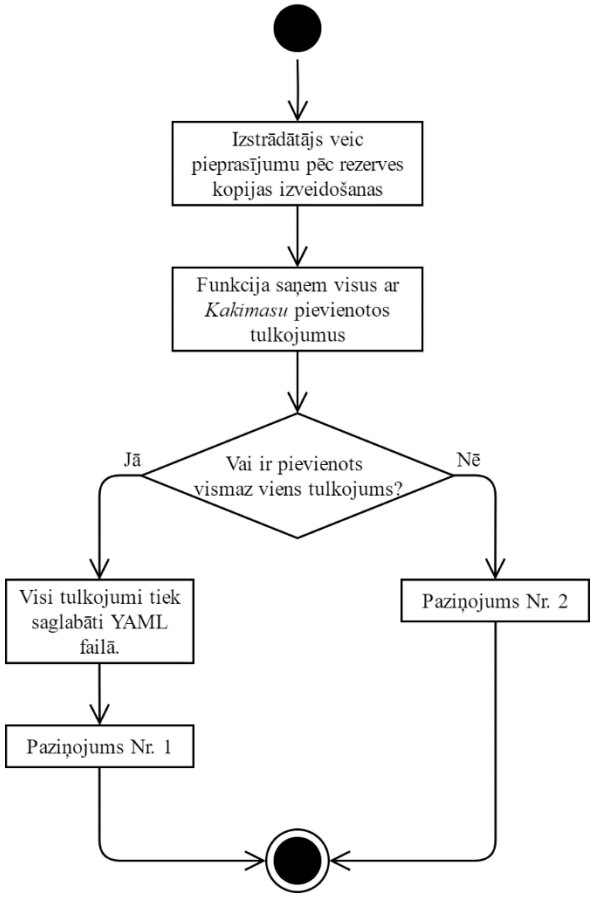
4.2.7. F-7 Pilnībā iztulkoto atslēgu procents



4.10. att. **Pilnībā iztulkoto atslēgu procenta aprēķināšana**

Funkcija F-7 sadarbojas ar funkciju F-5, t.i. no funkcijas F-5 iegūtā masīva tiek iegūts projektā kopējais izmaintoto atslēgu skaits (skat. 4.10. att.). Ja projektā nav nevienas atslēgas, tad lietotājam vietā, kur tiktu rādīts pilnībā iztulkoto atslēgu procents, tiek rādīts paziņojums nr. 1. Ja masīvā ir atslēgas, tad tiek uzzināts, cik daudzām no visām atslēgām, kādā no valodām nav tulkojuma. Tad attiecīgi procentuāli tiek izteikts, cik daudz atslēgu no visām ir pilnībā iztulkotas. Iegūtie rezultāti tiek rādīti lietotājam apļveida diagrammas veidā, un tiek rādīts pilnībā neiztulkoto atslēgu skaits.

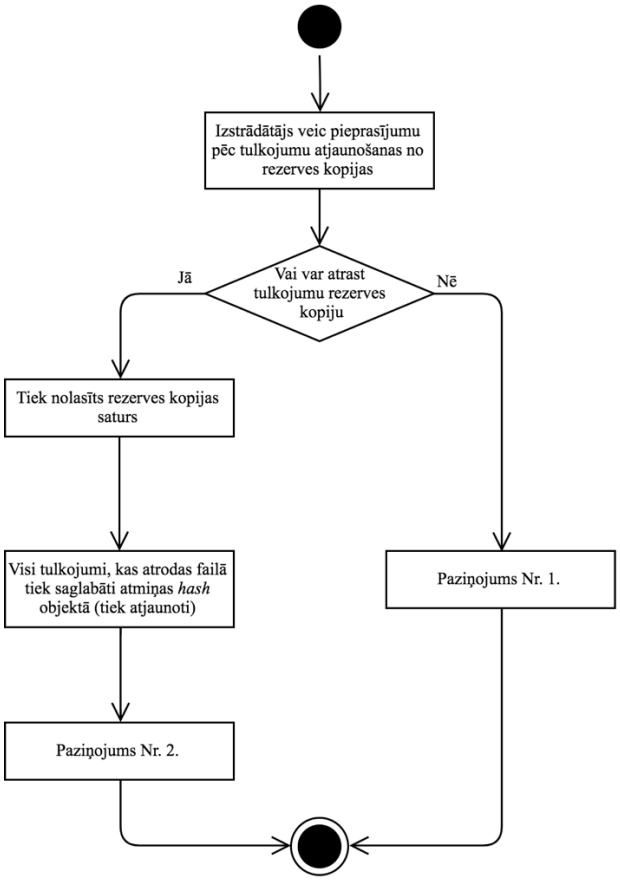
4.2.8. F-10 Tulkojumu rezerves kopijas izveide



4.11. att. **Tulkojumu rezerves kopijas izveide**

Funkcija paredzēta tam, lai varētu izveidot rezerves kopiju visiem tulkojumiem, kas pievienoti ar *Kakimasu*. Rezerves kopiju būs iespējams izveidot veicot pieprasījumu no komandrindas (skat. 4.11. att.). Vispirms, protams, tiek pārbaudīts, vai vispār ir tulkojumi, kas pievienoti ar *Kakimasu*. Ja nav neviens tulkojums, ko saglabāt, tad tiek rādīts paziņojums nr. 2, un nekas netiek darīts. Ja ir vismaz viens tulkojums, tad visi tulkojumi tiek saglabāti YAML failā pēc tāda paša principa, kādā *I18n* glabā tulkojumus (skat. nodaļu 4.1. Tulkojumu glabāšana), un tiek rādīts paziņojums nr. 1.

4.2.9. F-11 Tulkojumu atjaunošana no rezerves kopijas



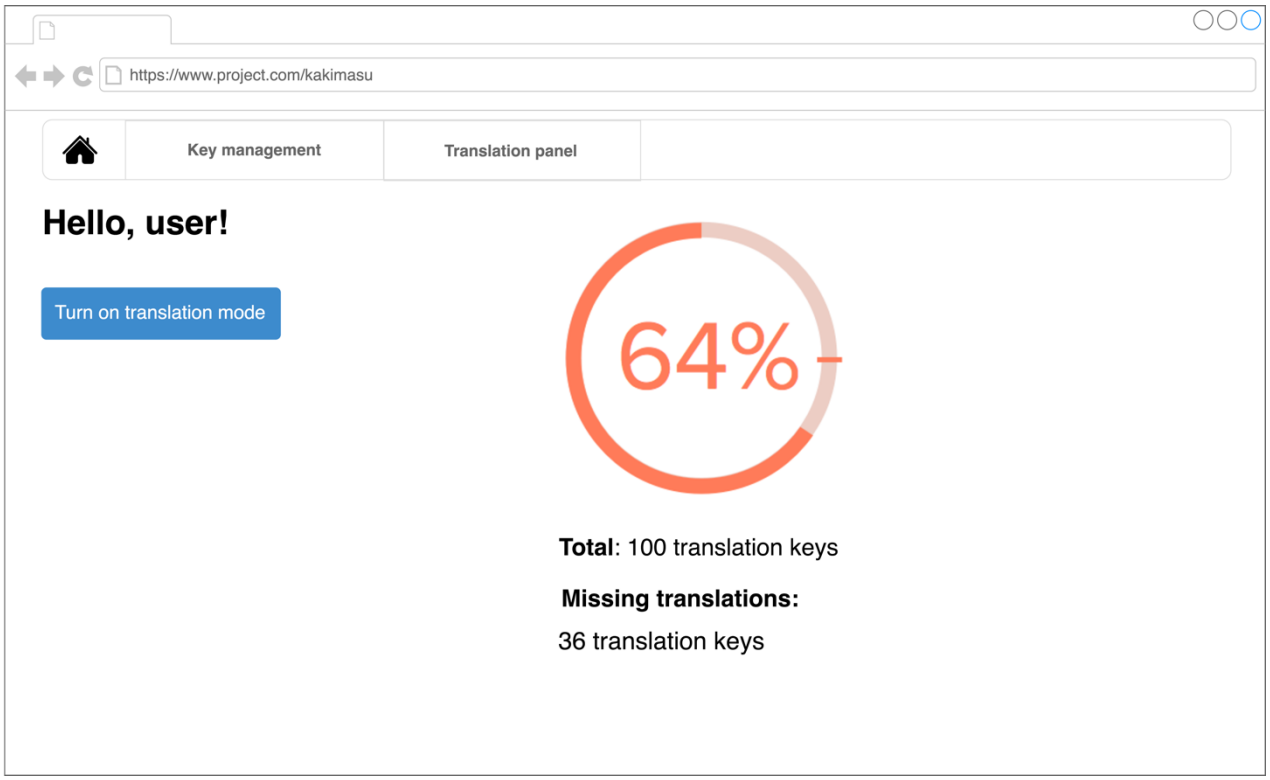
4.12. att. **Tulkojumu atjaunošana no rezerves kopijas**

Funkcija F-11 darbojas pretēji funkcijai F-10. Funkcija paredzēta tam, lai varētu atjaunot tulkojumus, kas ir rezerves kopijā. Funkciju, tāpat kā pārējās izstrādātāju moduļa funkcijas, iespējams izsaukt ar attiecīgo pieprasījumu no komandrindas (skat. 4.12. att.). Vispirms tiek meklēta rezerves kopija. Ja tā netiek atrasta, tad komandrindā tiek parādīts paziņojums Nr. 1. Ja rezerves kopija tiek atrasta, tad funkcija nolasa visu tās saturu.

Funkcija atrod visus rezerves kopijā esošos tulkojumus un saglabā tos *hash* objektā, gluži tāpat kā jaunus tulkojumus. Kad visi tulkojumi ir atjaunoti, tad komandrindā tiek parādīts paziņojums nr. 2.

4.3. Lietotāja saskarnes projektējums

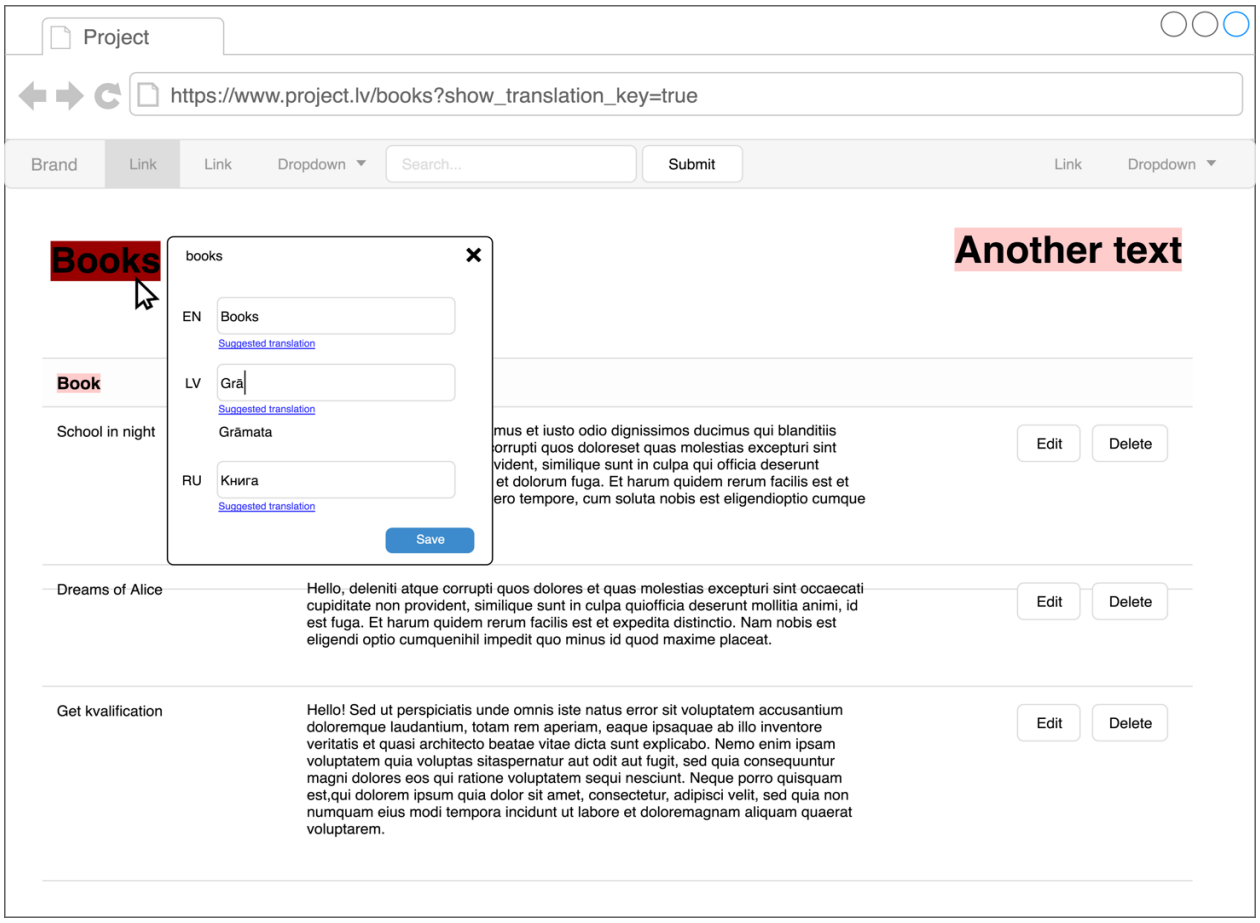
4.3.1 *Kakimasu* sākumlapa



4.13. att. ***Kakimasu*** **sākumlapa**

Kad lietotājs pievieno adresei "*/kakimasu*", viņš tiek nogādāts uz *Kakimasu* sākumlapu. Šinī skatā, tāpat kā citos *Kakimasu* skatos, ir navigācijas josla, no kuras var nonākt citos skatos. Zem sveiciena lietotājam atrodas "turn on translation mode" poga (skat. 4.13. att.), kas nodrošina paplašinājuma "ieslēgšanos" (skat. 4.3. att.). Kad tiek ieslēgts "tulkošanas režīms", poga tiek nomainīta uz pogu "turn off translation mode". Pa labi no sveiciena un pogas atrodas diagramma, kas parāda, cik liels procents no visām atslēgām ir pilnībā iztulkots (F-7 funkcija). Zem diagrammas ir redzams kopējais projektā izmantoto atslēgu skaits (vienādas atslēgas divos dažādos failos tiek uzskatītas kā divas dažādas atslēgas), un zem kopējā skaita ir atrodams to atslēgu skaits, kurām vismaz vienā valodā nav piešķirts tulkojums.

4.3.2. Tulkojuma pievienošanas “popup” forma



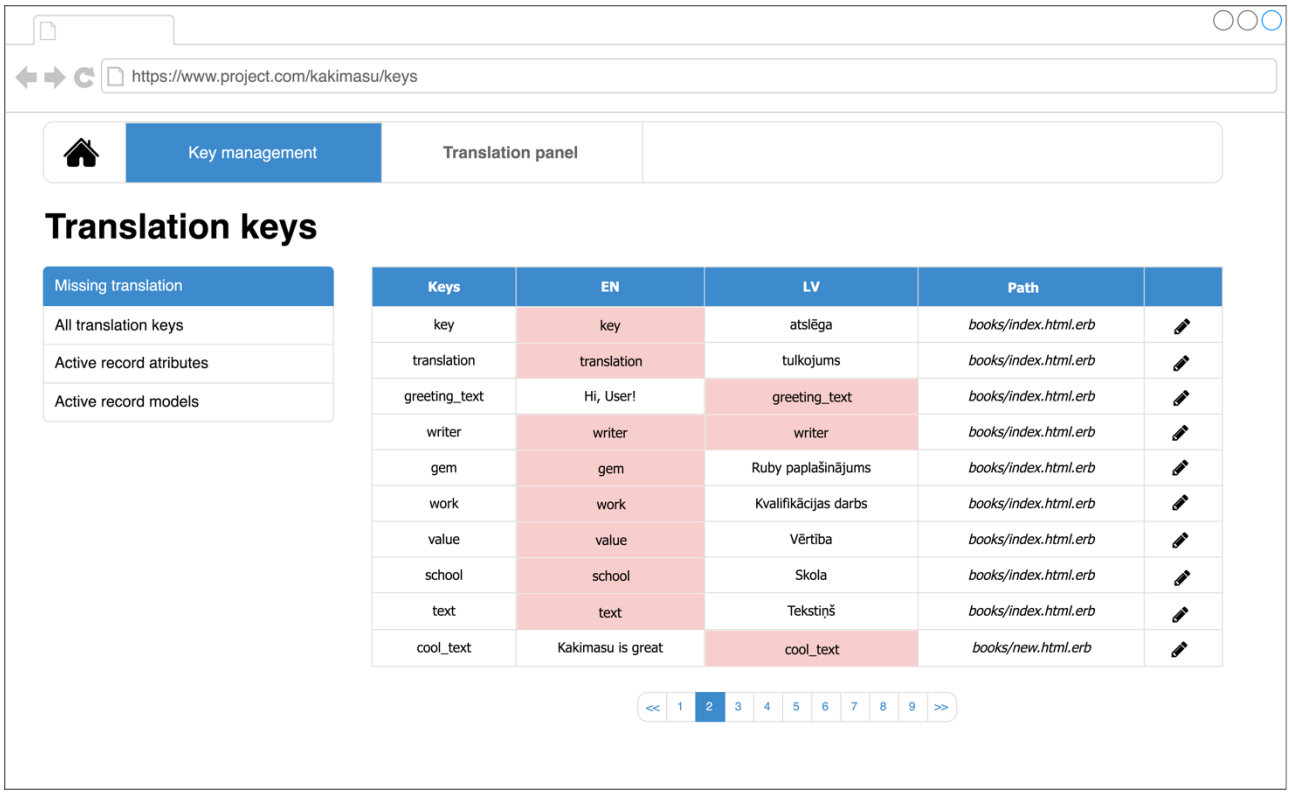
4.14. att. **Tulkojuma pievienošana**

Kad ir ieslēgts “tulkošanas režīms”, tad visiem tulkojamajiem vārdiem projektā, pie kura pievienots *Kakimasu*, fons iekrāsojas gaiši sarkans (skat. 4.14. att.). Gaiši sarkanais fons ir paredzēts tam, lai lietotājam būtu skaidrs, kurus vārdus viņš var iztulkot. Kad lietotājs novieto peles kursoru uz tulkojamā vārda, tā fons tiek iekrāsots tumši sarkans, un parādās tulkošanas “popover” elements (skat. 4.4. att.). Tumši sarkanais fons domāts tam, lai lietotājam būtu skaidrs, kuru vārdu viņš pašlaik tulko.

Tulkošanas formas augšpusē ir redzama tulkojuma atslēga, kurai viņš piešķirs tulkojumu. Formā ir tik daudz lauku, cik projektam norādīto valodu (ja projektā ir norādītas 3 valodas: latviešu, krievu, angļu, tad formā ir 3 lauki). Katrā laukā tiek ievadīts tulkojums attiecīgajā valodā. Zem tulkojuma lauka atrodas poga “suggested translation”, kuru nospiežot tiek izpildīta funkcija F-3. Kad lietotājs nospiež pogu “save”, tad tiek pabeigta funkcijas F-1 darbība, un tiek saglabāti jaunie tulkojumi.

Pēc jaunā tulkojuma pievienošanas “popover” elements pazūd un lapas saturs tiek pārlādēts.

4.3.3. “Key management” skats



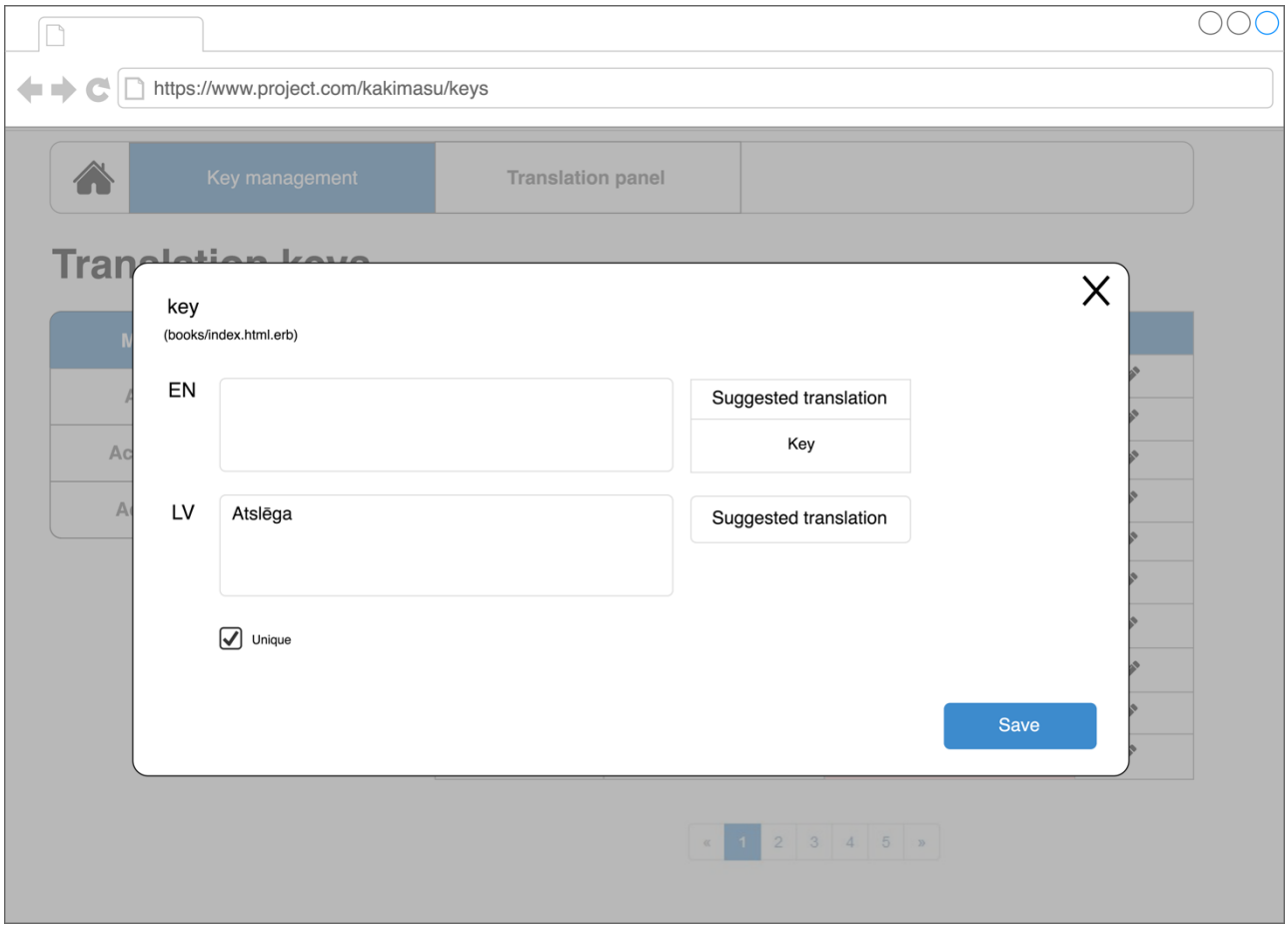
4.15. att. **“Key management” skats**

“Key management” skats ir paredzēts F-5 un F-6 funkciju iegūto rezultātu attēlošanai (skat. 4.15. att.). Virsraksts “translation keys” norāda lietotājam, ka redzamajās tabulās ir attēlota informācija saistībā ar tulkojumu atslēgām.

Pa kreisi no tabulas atrodas attēlojamās atslēgu kategorijas izvēlne. Tajā var izvēlēties tādas kategorijas kā “missing translations”, kurā tiek attēlotas visas tās atslēgas, kurām kādā no valodām nav tulkojuma (F-6). Nākamā kategorija ir “all translation keys”, kas rāda lietotājam visas projektā pieejamās atslēgas (F-5). Tālākās kategorijas ir domātas datu bāzes lauku tulkošanai (atribūtu tulkošanai un modeļu nosaukumu tulkošanai). Iespējams, nākotnē parādīsies vajadzība pēc jaunām kategorijām, un tad tās tiks pievienotas šajai izvēlnei.

Tabulā tiek rādītas visas atslēgas no izvēlētās kategorijas. Pirmā kategorija, kas tiek rādīta lietotājam ieejot šinī skatā, ir “missing translations”, jo tajā atrodas lietotājam visaktuālākā informācija – kas vēl projektā nav līdz galam iztulkots. Tabulā šūnas, kurās trūkst tulkojuma, tiek iekrāsotas gaiši sarkanā krāsā, lai pievērstu lietotāja uzmanību. Blakus tulkojumiem tiek rādīta arī atslēgas atrašanās vieta. Nospiežot uz zīmulīša ikonas (“edit” poga), lietotājam tiek atvērta forma tukojuma ievadei (skat. 4.16. att.).

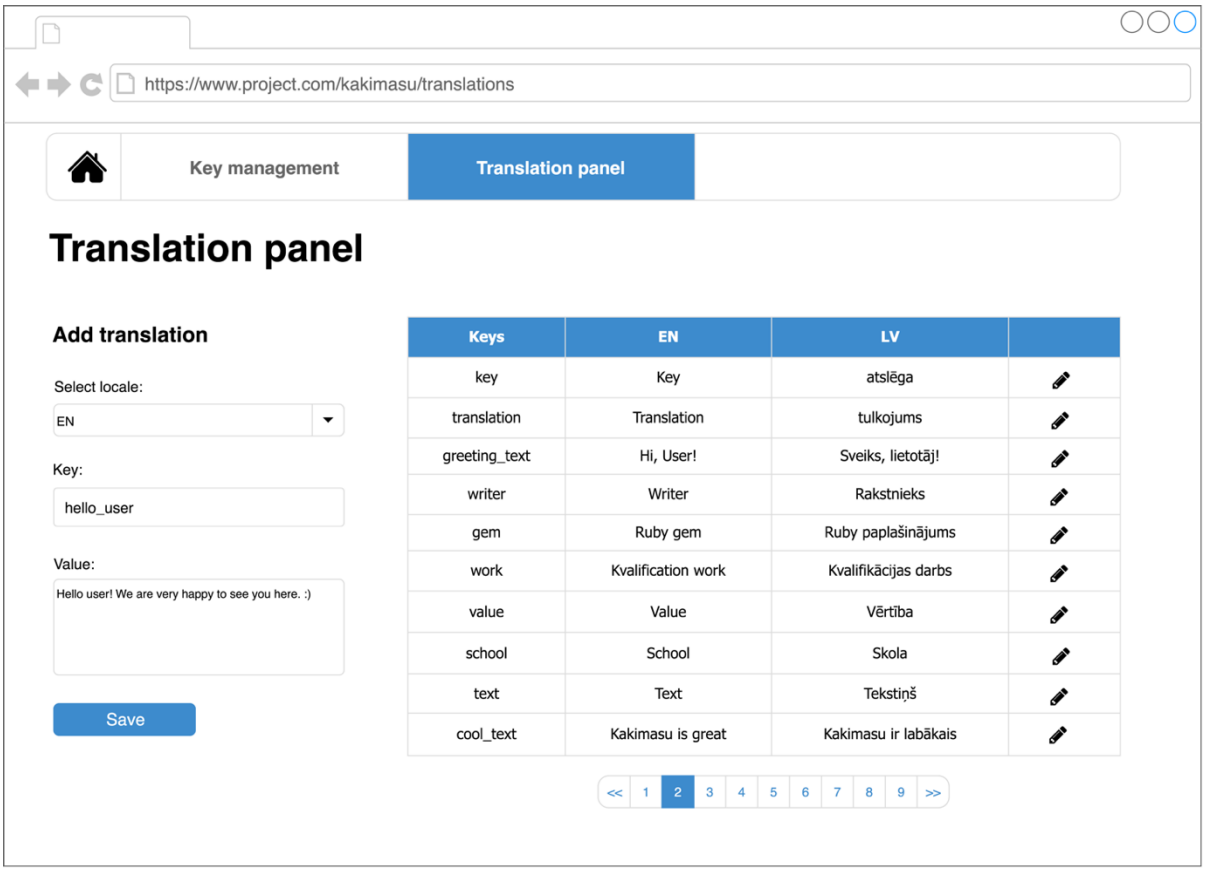
4.3.4. Tulkošanas “modal” forma



4.16. att. **Tulkošanas forma *Kakimasu* tabulās.**

Formas augšā tiek rādīta atslēga, kurai lietotājs pašlaik pievieno tulkojumu (skat. 4.16. att.). Zem atslēgas tiek rādīta arī atslēgas atrašanās vieta, lai lietotājam būtu mazāk iespēju nokļūdīties. Formā katrai valodai ir lauks, kurā tiek ievadīts vai labots tulkojums. Pa labi no formas atrodas poga “suggested translation”, kuras nospiešana izpilda funkciju F-3. Pēc “suggested translation” pogas nospiešanas, zem tās tiek parādīts *google translate* atgrieztais tulkojums. Zem pēdējā ievades lauka atrodas izvēles rūtiņa “unique”, kas paredzēta funkcijas F-2 nodrošināšanai (skat. 4.5. att.). Ieķeksēta rūtiņa nozīmē, ka atslēga ir unikāla (tiek izmantots *I18n “lazy lookup”* mehānisms), savukārt, neatķeksēta nozīmē pretējo. Formas apakšā atrodas poga tulkojuma saglabāšanai. Kad nospiesta poga “save”, tad forma pazūd, un lapas saturs tiek pārlādēts.

4.3.5. “Translation panel” skats



4.17. att. **“Translation panel” skats**

“Translation panel” skats sastāv no divām daļām (skat. 4.17. att.). Labajā pusē atrodas tabula ar visiem ar *Kakimasu* pievienotajiem tulkojumiem (F-4). Tabulā tiek parādīta tulkojuma atslēga un visi tai piešķirtie tulkojumi visās valodās. Līdzīgi kā “key management” skatā, ja kādā no valodām atslēgai nav tulkojuma, tad tā šūna tiek iekrāsota sarkana, lai pievērstu lietotāja uzmanību. Galvenā atšķirība starp abu skatu tabulām ir tā, ka šajā tabulā netiek rādītas visas projektā pievienotās atslēgas, līdz ar to ir vieglāk pārvaldīt tieši ar *Kakimasu* pievienotos tulkojumus. Tāpat kā “key management” tabulās, arī šinī tabulā nospiežot uz zīmulīša ikonas var izlabot tulkojumus (tiesa formā nav “unique” izvēles rūtiņas, jo katrai atslēgai nav zināma faila, kurā tā tiek izmantota, atrašanās vieta).

Galvenā atšķirība starp abiem skatiem ir tā, ka “translation panel” skatā ir iespēja pievienot tulkojumu “ar roku” – formā var izvēlēties valodu, ievadīt atslēgu un tulkojumu, un tiek izveidots tulkojums. Šī forma īpaši noder izstrādātājiem, piemēram, ja izstrādātājs vēlās iztulkot kļūdu paziņojumus, jo kļūdu paziņojumu atslēgas reti tiek izmantotas skatos, tāpēc lietotājs tos nevarēs iztulkot tradicionālajā *Kakimasu* veidā – ar “popup” formu, bet tāpat ievadīt tulkojumus caur šo formu ir daudz ērtāk nekā tos ievadīt YAML failos.

5. TESTĒŠANAS DOKUMENTĀCIJA

5.1. Testēšanas organizācija

Projekta izstrādes gaitā paplašinājums tiek vairākas reizes testēts, lai pārliecinātos, ka visas funkcijas darbojas korekti. Katras paplašinājuma testēšanas laikā, tiek testēta katra funkcija, pat ja tā iepriekš ir tikusi testēta un nav uzrādījusi kļūdu, lai pārliecinātos, ka pēc veiktajām izmaiņām nav izmainījusies agrāk uzprogrammētas funkcijas darbība.

5.2. Testpiemēri

Testpiemēru sastādīšanā liela uzmanība tiek pievērsta dažādiem robežgadījumiem, kas varētu likt funkcijām "kļūdīties". Ar testpiemēriem tiek pārbaudītas visi PPS apskatītie varianti (funkcijas tiek pārbaudītas tā, lai vismaz vienu reizi tiktu parādīts katrs kļūdu paziņojums). Testpiemēri tiek grupēti pēc moduļiem.

5.2.1. Tulkošanas modulis

Testpiemēri, kas attiecas uz tulkošanas moduli, tiek apskatīti 5.1. tabulā.

5.1. tabula

Tulkošanas moduļa funkciju testpiemēri

TP nr.	**Apraksts**			**Funkcija**
-	**Ievaddati**	**Soļi**	**Sagaidāmais rezultāts**	**-**
1.	Tulkojums.	1.Kursors tiek novietots uz kāda no vārdiem. 2. Tulkošanas formā tiek ievadīts tulkojums. 3. Tiek nospiesta poga "save".	Tiek saglabāts tulkojums, un pēc lapas pārlādēšanas tiek parādīts jaunais tulkojums.	F-1

5.1. tabulas turpinājums

2.	Tukšs tulkojums.	1. Kursors tiek novietots uz kāda no vārdiem, kuram ir tulkojums. 2. No tulkošanas tiek izdzēsts esošais tulkojums, un tā vietā nekas netiek ievadīts. 3. Tiek nospiesta poga "save".	Nekas netiek izmainīts.	F-1
3.	Tulkojums un ieķeksēta izvēles rūtiņa "unique".	1. "Key management" tabulā tiek nospiesta "edit" poga tulkojumam, kura atslēga nav unikāla. 2. Tiek ievadīts tulkojums un ieķeksēta izvēles rūtiņa "unique". 3. Tiek nospiesta poga "save".	Konkrētajai atslēga failā priekšā tiek pievienots punkts (atslēga tiek padarīta unikāla). Paziņojums nr. 1.	F-2
4.	-	1. "Key management" tabulā tiek nospiesta "edit" poga tulkojumam, kura atslēga ir unikāla. 2. Tiek nospiesta poga "save".	Paziņojums nr. 2.	F-2

5.1. tabulas turpinājums

5.	Tulkojums un atķeksēta izvēles rūtiņa "unique".	1. "Key management" tabulā tiek nospiesta "edit" poga, tulkojumam, kura atslēga ir unikāla. 2. Tiek noņemts ķeksis no izvēles rūtiņas "unique". 3. Tiek nospiesta poga "save".	No konkrētās atslēgas failā tiek noņemts punkts (atslēga tiek padarīta neunikāla). Paziņojums nr. 1.	F-2

5.2.2. *Google translate* modulis

Testpiemēri *google translate* moduļa funkcijas testēšanai tiek apskatīti 5.2. tabulā

5.2. tabula

***Google translate* moduļa testpiemēri**

TP nr.	**Apraksts**			**Funkcija**
	Ievaddati	**Soļi**	**Sagaidāmais rezultāts**	**-**
6.	-	1. Tulkošanas formā, kurā vārdam vismaz vienā valodā ir tulkojums, valodai, kurā nav tulkojuma tiek nospiesta poga "suggested translation".	*Google translate* ieteiktais tulkojums.	F-3

5.2. tabulas turpinājums

7.	-	1. Tulkošanas formā, kurā atslēgai nevienā valodā nav norādīts tulkojums, tiek nospiesta poga "suggested translation".	Paziņojums nr. 2.	F-3
8.	-	1. Datoram, pie kura notiek testēšana, uz brīdi tiek atslēgts interneta savienojums. 2. Tulkošanas formā tiek nospiesta poga "suggested translation".	Paziņojums nr. 1.	F-3

5.2.3. Administrēšanas modulis

Administrēšanas moduļa funkcijas tiek testētas ar 5.3. tabulā esošajiem testpiemēriem.

5.3. tabula

Administrēšanas moduļa funkciju testpiemēri

TP nr.	**Apraksts**			**Funkcija**
-	**Ievaddati**	**Soļi**	**Sagaidāmais rezultāts**	**-**
9.	-	1. Tiek ieslēgts "translation panel" skats.	Tabulā tiek parādītas visi ar *Kakimasu* pievienotie tulkojumi. Par katru tulkojumu tiek parādīts – atslēga, tulkojumi visās valodās, "edit" poga.	F-4

5.3. tabulas turpinājums

10.	-	1. Tiek ieslēgts “key management” skats. 2. tiek izvēlēta kategorija “all translations”, tad tiek izvēlēta kategorija “active record attributes”, un tad tiek izvēlēta kategorija “active record models”.	Tabulā tiek parādītas visas atslēgas, kas pieder izvēlētajai kategorijai. Par katru atslēgu tiek parādīts – atslēga, tulkojumi visās valodās, atslēgas atrašanās vieta un “edit” poga.	F-5
11.	-	1. Tiek ieslēgts “key management” skats. 2. Pārliecinās, vai aktīvā kategorija ir “missing translations”.	Tabulā tiek parādītas visas atslēgas, kurām trūkst tulkojuma. Valodas šūna, kurā trūkst tulkojuma tiek iekrāsota sarkana. Par katru atslēgu tiek parādīts – atslēga, tulkojumi visās valodās (ja tādi ir), atslēgas atrašanās vieta, “edit” poga.	F-6
12.	Tulkojums.	1. Tiek ieslēgts “key management” skats. 2. Pārliecinās, vai aktīvā kategorija ir “missing translations”. 3. Kādam no tabulā esošajiem vārdiem tiek pievienots tulkojums.	Šūnā, kurā atslēgai nebija tulkojuma, parādās jaunais tulkojums, un tā vairs nav sarkana. Ja pēc tulkojuma pievienošanas atslēgai visās valodās ir tulkojumi, tad atslēga pazūd no tabulas.	F-6

5.3. tabulas turpinājums

13.	-	1. Tiek atvērta *Kakimasu* sākumlapa.	Apļveida diagramma ar pilnībā iztulkoto atslēgu procentu, kopējo projektā izmantoto atslēgu skaitu un pilnībā neiztulkoto atslēgu skaitu.	F-7

5.2.4. Izstrādātāju modulis

5.4. tabulā tiek aprakstīti testpiemēri, kas tiek izmantoti izstrādātāju moduļa funkciju testēšanā.

5.4. tabula

Izstrādātāju moduļa funkciju testpiemēri

TP nr.	**Apraksts**			**Funkcija**
-	**Ievaddati**	**Soļi**	**Sagaidāmais rezultāts**	**-**
14.	Pieprasījums pēc skatu ģenerēšanas.	1. Komandrindā tiek ierakstīts pieprasījums pēc skatu ģenerēšanas – *rails generate kakimasu:views*.	Tiek rādīts paziņojums nr. 1. un projektā tiek uzģenerēti *Kakimasu* skati.	F-8
15.	Pieprasījums pēc piekļuves tiesību failu ģenerēšanas.	1. Komandrindā tiek ierakstīts pieprasījums pēc piekļuves tiesību failu ģenerēšanas – *rails generate kakimasu:policy*.	Tiek rādīts paziņojums nr. 1. un projektā tiek uzģenerēti faili, kuros definētas *Kakimasu* piekļuves tiesības.	F-9

16.	Pieprasījums pēc tulkojumu rezerves kopijas izveidošanas.	1. Komandrindā tiek ierakstīts pieprasījums pēc tulkojumu rezerves kopijas izveidošanas – *rails generate kakimasu:backup*. 2. Tiek nodzēsts iekšējās atmiņas *hash* objekts, kurā glabājas visi tulkojumi. 3. Komandrindā tiek ierakstīts pieprasījums pēc vēl vienas tulkojumu rezerves kopijas izveidošanu.	Pēc pirmā pieprasījuma visi tulkojumi tiek saglabāti tulkojumu YAML failā un tiek parādīts paziņojums nr. 1. Kad tiek nodzēsti visi tulkojumi no *hash* objekta, tad pārliecinās, vai projektā tulkojumi nav zuduši (tie veiksmīgi tiek ielasīti no YAML tulkojumu faila). Pēc atkārtotā pieprasījuma izveidot rezerves kopiju tiek atgriezts paziņojums nr. 2, jo *hash* objektā nav neviena tulkojuma.	F-10
17.	Pieprasījums pēc tulkojumu atjaunošanas no rezerves kopijas.	1. Tiek nodzēsts iekšējais atmiņas *hash* objekts, kurā glabājas visi tulkojumi. 2. Komandrindā tiek ierakstīts pieprasījums pēc tulkojumu atjaunošanas no rezerves kopijas – *rails generate kakimasu:restore_backup.* 3. Tiek izdzēsta tulkojumu rezerves kopija. 4. Vēlreiz tiek veikts pieprasījums pēc F-11.	Pēc pirmā pieprasījuma, tiek atjaunoti visi rezerves kopijā esošie tulkojumi un parādīts paziņojums nr. 2. Pēc otrā pieprasījuma, kad ir izdzēsta rezerves kopija, tiek parādīts paziņojums nr. 1.	F-11

5.2.5 Nefunkcionālo prasību testpiemēri

Testpiemēri nefunkcionālo prasību testēšanai ir aprakstīti 5.5. tabulā. *Google translate* servisa atbalsts tiek pārbaudīts 6.-8. testpiemēros, kā arī projekta uzticamības prasības tiek pārbaudītas 16. un 17. testpiemērā.

5.5. tabula

Nefunkcionālo prasību testpiemēri

TP nr.	**Apraksts**			**Nefunkcionālā prasība**
-	**Ievaddati**	**Soļi**	**Sagaidāmais rezultāts**	**-**
18.	-	1. Tiek restartēts *Rails* serveris.	Visi pievienotie tulkojumi ir saglabājušies.	Funkcionālā piemērotība
19.	-	1. Tiek izslēgts interneta pieslēgums. 2. Tiek veikti visi augstāk minētie testpiemēri.	Visas funkcijas izņemot F-3 spēj darboties arī bez interneta savienojuma.	Funkcionālā piemērotība
20.	-	1. Tiek mērīta veiktspēja diviem identiskiem projektiem (vienam ir pievienots *Kakimasu*, otram nav).	Pieprasījumi abos projektos izpildās līdzīgā laikā.	Veiktspēja
21.	-	1. Neautorizējies lietotājs pievieno adresei beigās "../translations".	Neautorizējies lietotājs nevar piekļūt *Kakimasu* skatiem.	Drošība

5.5. tabulas turpinājums

22.	-	1. Autorizējies lietotājs pievieno adresei beigās "/translations".	Autorizējies lietotājs var piekļūt paplašinājuma skatiem.	Drošība
23.	-	1. Neautorizējies lietotājs pievieno adresei beigās "*?show_translation_key = true"*.	Neautorizēts lietotājs nevar ieslēgt "tulkošanas režīmu", līdz ar to nevar pievienot tulkojumus.	Drošība

5.3. Testēšanas žurnāls

Paplašinājuma testēšana projekta sākumā tiek veikta reizi mēnesī (martā un aprīlī). Kad projekts tuvojas termiņam un izstrādāto funkciju skaits kļūst lielāks, testēšanas biežums tiek palielināts. 5.6. tabulā ir redzams testēšanas žurnāls. Ar '+' tiek apzīmēti testi, kas noritējuši veiksmīgi, ar '-' tiek apzīmēti tie testi, kuros ir radusies kļūda, un blakus ir kļūdas apraksts. Tā kā, kad tiek veikta testēšana, dažas funkcijas var nebūt gatavas, tad ar 'N' tiek apzīmēti tie testi, kuri netiek veikti, tāpēc, ka funkcija nav gatava.

5.6. tabula

Testēšanas žurnāls

TP nr.	17.03.2017	21.04.2017	1.05.2017	15.05.2017	28.05.2017
1.	+	+	+	+	+
2.	- (Tukša simbolu virkne tiek saglabāta kā tulkojums)	+	+	+	+
3.	N	+	+	+	+
4.	N	+	+	+	+

5.	N	- (ja failā ir vairākas vienādas atslēgas, tad tās tiek sabojātas, un skats vairs nestrādā).	+	+	+
6.	N	N	N	- (Ja tulkojamā simbolu virkne ir vairāki teikumi, tad tiek iztulkots tikai pirmais teikums).	+
7.	N	N	N	- (Netiek parādīts brīdinājums, kāpēc vārds netiek tulkots).	+
8.	N	N	N	+	+
9.	+	+	+	+	+
10.	N	+	+	+	+
11.	N	+	+	+	+
12.	N	+	+	+	+
13.	N	N	N	+	+
14.	+	+	+	+	+
15.	N	N	+	+	+
16.	N	N	N	N	+
17.	N	N	N	N	+
18.	+	+	+	+	+
19.	+	+	+	+	+
20.	+	+	+	+	+
21.	N	N	+	+	+
22.	N	N	+	+	+
23.	N	N	+	+	+

REZULTĀTI

Kvalifikācijas darba ietvaros patstāvīgi tika izstrādāts programmatūras prasību specifikācijai atbilstošs *Ruby* paplašinājums *Kakimasu*. Kvalifikācijas darbs tika izstrādāts laika posmā: 2017. gada 12. februāris – 2017. gada 28. maijs.

Paplašinājuma izstrādes laikā tika nostiprinātas zināšanas *Ruby* programmēšanas valodā, kā arī apgūts, kā tiek veidoti *Ruby* paplašinājumi ("Ruby gems"). Tika apgūtas jQuery (Javascript) un CSS tehnoloģijas. Tuvāk tika iepazīts *Ruby on Rails* noklusētais tulkošanas mehānisms un tas, kā var pārrakstīt citu *Ruby* paplašinājumu metodes, lai tās pielāgotu savām vajadzībām. Gūtās zināšanas ļoti noderēs nākamo *Ruby* projektu izstrādē.

Kvalifikācijas darba laikā nācās risināt tādas problēmas kā, jQuery versiju nesaskaņas, javascript funkciju izpilde, ja ir ieslēgta/izslēgta *turbolinks* iezīme, tabulas dalīšana lapās (*pagination*), ja tiek apstrādāts nestandarta masīvs (masīvs, kas sastāv no dažādu garumu masīviem).

Kakimasu izstrādes laikā radās idejas par papildus funkcijām, ko varētu pievienot nākamajās versijās – tulkojumu atslēgu tabulas filtri pēc skatiem, tulkojumu importēšana no dažādiem failiem, lai nodrošinātu ieteikto tulkojumu rādīšanu lietotājam arī tad, kad nav pieejams interneta savienojums. Diemžēl šīs ieceres netika realizētas kvalifikācijas darba laikā ierobežotā laika dēļ, taču tās plānots izstrādāt tuvākajā laikā.

Pielikumā ir apskatāmi, pēc autora domām, nozīmīgākie programkoda fragmenti.

IZMANTOTĀ LITERATŪRA

1. LVS 68:1996 "Programmatūras prasību specifikācijas ceļvedis"
2. LVS 72:1996, "Ieteicamā programmatūras projektējuma apraksts"
3. LVS 70:1996, "Programmatūras testēšanas dokumentācija"
4. Karl. Express_translate. https://rubygems.org/gems/express_translate (skatīts 13.03.2017.)
5. Bcotton. Translate. https://rubygems.org/gems/translate (satīts 13.03.2017.)
6. Crepezzi, J. Easy_translate. https://rubygems.org/gems/easy_translate (skatīts 13.03.2017.)
7. Ruby documentation https://www.ruby-lang.org/en/documentation/ (skatīts 20.02.2017.)
8. Batsov B., Arvidsson J., Nakayama Y. Rubocop. https://rubygems.org/gems/rubocop (skatīts 22.02.2017.)
9. Redis https://redis.io/ (skatīts 20.02.2017.)
10. Niclas J., Elabs AB. Pundit https://rubygems.org/gems/pundit (skatīts 10.04.2017)
11. Fuchs S., Harvey J., Aimonetti M., Soller S., Moore S. I18n. https://rubygems.org/gems/i18n (skatīts 20.02.2017.)

PIELIKUMI

1. Pielikums. Key_controller.rb

```
module Kakimasu
  class KeysController < ::ApplicationController
    # Include neccessary helpers
    include SearchTranslationsHelper
    include GroupingHelper
    include SearchKeyHelper
    include PaginationHelper

    before_action :categories
    before_action :set_keys_category

    def index
      # Check if user has permissions
      authorize :translation, :index?
      policy_scope(Translation)

      set_page_size(5)                              # set number of keys in one page
      params[:page] = 1 unless params[:page]        # if page number is not stated, then by default it is 1.
page
      @keys = search_for_translations(@category)    # search for used keys
      @active_tab = 'keys'                          # sets active tab in navigation
    end

    def create
      # Authorize user
      authorize :translation, :create?
      # Check if path to translation key is given
      enable_lazy_lookup unless params[:path].blank?
      # Save entered translations
      I18n.available_locales.each do |l|
        params[l].gsub!("'", "&#39;")
        I18n.backend.store_translations(l, { params[:key] => params[l] }, escape: false) unless
params[l].blank?
      end
```

```
    I18n.backend.store_translations(params[:language], { params[:key] => params[:value] }, escape: false) if params[:language]
    # Process ajax process diferently
    if request.xhr?
      respond_to do |format|
        format.js {}
      end
    else
      redirect_to kakimasu_keys_path(keys_category: @category)
    end
  end

  private
  # All available categories in "key management" view
  def categories
    @categories = ['missing_translations', 'all_translation_keys', 'active_record_attribute_translations', 'active_record_model_translations']
  end

  # Set current category
  def set_keys_category
    @category = params[:keys_category] || 'missing_translations'
  end

  # Enable/disable 'lazy lookup' on key based on checkbox 'unique' value
  def enable_lazy_lookup
    if params[:lazy_lookup]
      lazy_lookup_key(params[:key], params[:path])
      params[:key] = prefix(params[:path]) + '.' + params[:key] unless params[:key].start_with?(prefix(params[:path]))
    else
      remove_lazy_lookup(params[:key], params[:path])
      params[:key] = params[:key].split('.').last
    end
  end
 end
end
```

```
# Sets translation key for lazy lookup or unsets it.
Module SearchKeyHelper
  def lazy_lookup_key(key, path)
    path = 'app/views/' + path
    file = File.read(path)

    matches = Array.new           #All places where this key is used in given file
    locations = Array.new         #Indexes which determines where to make changes
    file_length_differential = 0  #File length difference because substitutions will change file size.

    # Finds all places where given translation key is used
    file.scan(/[=|\s][t][\s]*[(]?[\s]*['|:]#{key}/) do |c|
      matches.push [c, $~.offset(0)[0]]
    end

    matches.each do |array|
      if array.first.match /[=|\s][t][\s]*[(]?[\s]*[:]\w*/  # if translation written with : not '' then its special
case
        # Key with ':key' is 61ubstituted with "'.key"
        index = array.first.index(/:#{key}/)
        file.slice!(array.last + index + file_length_differential, (":#{key}").length)
        file.insert(array.last + index + file_length_differential, "'.#{key}")
        file_length_differential += ("'.#{key}").length – (":#{key}").length
      elsif key[0] != '.'
        index = array.first.index(key)
        locations.push array.last + index + file_length_differential
      end
    end

    # Insert '.' just before key for lazy lookup
    locations.each_with_index do |location, index|
      file.insert(location + index, '.')
    end
```

```
    File.write(path, file)
  end

  def remove_lazy_lookup(key, path)
    path = 'app/views/' + path
    file = File.read(path)

    # Get rid of prefix of has lazy lookup prefix
    if key.include? '.'
      key = key.split('.').last
      key.prepend '.'
    end

    matches = Array.new    # All places where given key is used in file
    locations = Array.new  #Indexes which determintes where to make changes

    key.slice! 0 if key[0] == '.'
    file.scan(/[=|\s][t][\s]*[(]?[\s]*['][.]#{key}/) do |c|
      matches.push [c, $~.offset(0)[0]]
    end

    matches.each do |array|
      index = array.first.index('.' + key)
      locations.push array.last + index
    end

    locations.each_with_index do |location, index|
      file.slice!(location – index)    # Removes '.' from match
    end

    File.write(path, file)
  end
end
```

3. Pielikums. Search_translations_helper.rb

```
Module SearchTranslationsHelper
 # Method to get prefix for lazy lookup
 def prefix(path)
  prefix = path.split('/')
  prefix[-1] = prefix.last.split('.').first
  prefix = prefix.join('.')
 end

 # Method returns if key has missing translation in one of locales
 def missing_translations?(key)
  missing_translation = false
  I18n.available_locales.each do |locale|
   if !I18n.exists?(key, locale)
    missing_translation = true  # If in one of locales key has no translation, method returns true
   end
  end
  missing_translation
 end

 # Method searches used translations keys in a file
 def find_keys(file, path)
  array = file.scan(/[=|\s][t][\s]*[(]?[\s]*['|:]([.]*\w*)/).uniq
  keys = Array.new

  # If key has '.' in front of it, it should be completed as lazy lookup key.
  Array.each_with_index do |key, index|
   if key.first[0] == '.'
    array[index] = key.first.insert(0, prefix(path))
   end
  end
  array
 end

 # Method returns array of keys with missing translations from given array
```

```
def find_missing_translations_keys(array)
  missing_translations_keys = []

  array.each do |key|
    missing_translations_keys.push key if missing_translations?(key)
  end

  missing_translations_keys
end

# Method searches used active record human attribute name keys in given file
def find_attribute_keys(file)
  array = file.scan(/(\w*).human_attribute_name[(][:](\w*)/).uniq
  keys = []
  # Keys are transformed into correct active record attribute keys form
  array.each do |arr|
    arr = 'activerecord.' + 'attributes.' + arr.first.downcase + '.' + arr.last
    keys.push arr
  end
  keys
end

# Method searches used Model name keys in given file
def find_model_name_keys(file)
  array = file.scan(/(\w*).model_name.human/).uniq
  keys = []
  # Keys are transformed into correct active record model name keys form
  array.each do |arr|
    arr = 'activerecord.models.' + arr.first.downcase
    keys.push arr
  end
  keys
end

# Method returns array with requested translation keys
def search_for_translations(category)
  keys = []
```

```
    Dir.glob("app/views/**/*").each do |path|
      if File.file?(path)
        file = File.read(path)
        case category
        when 'active_record_attribute_translations'
          keys.push find_attribute_keys(file)
        when 'active_record_model_translations'
          keys.push find_model_name_keys(file)
        when 'all_translation_keys'
          path.slice! "app/views/"
          keys.push find_keys(file, path).push path
        when 'missing_translations'
          path.slice! "app/views/"
          # push in array only those keys which have missing translations
          keys.push find_missing_translations_keys(find_keys(file, path)).push path
        end
      end
    end

    case category
    when 'active_record_attribute_translations'
      keys.flatten.uniq
    when 'active_record_model_translations'
      keys.flatten.uniq
    else
      keys
    end
  end
end
```

```
module Kakimasu
  module Generators
    class BackupGenerator < Rails::Generators::Base
      # Groups all added keys from TRANSLATION_STORE
      grouped_keys_by_locale = TRANSLATION_STORE.keys.group_by { |el| el.partition('.').first }

      # Checks if there is at least one translation in TRANSLATION_STORE
      if TRANSLATION_STORE.keys.count == 0
        puts '************************************************'
        puts 'Backup can not be done. There are no translations!'
        puts '************************************************'
      else

        backup_file = '.kakimasu_backup.yml'

        # Check if backup file already exists
        unless File.file?(backup_file)
          puts 'Creating backup_file...'

          # If there is no backup file then it is created here
          File.write(backup_file, "")
        end

        # Read Kakimasu backup file
        file_content = File.read(backup_file)

        # Modify file content - write all translations in there
        I18n.available_locales.each do |locale|
          position = 0      # Position in string where to start write
          exists = false     # Tells if this locale is saved in backup (if not then after all translations will be
writed 'new line character' to seperate languages)

          # Search where to start insert new translations in string (Searching goes by lines)
          file_content.each_line do |line|
            position += line.length
```

```
        # If locale is found then break the loop
        if line.start_with?("#{locale}:")
          exists = true
          break
        end
      end

      # If locale doesn't exist in the file already then start writing from file start and insert new line just to seperate languages
      unless exists
        position = 0
        file_content.insert(0, "\n")
      end

      # If there is at least one translation in current language just then will insert new translations
      unless grouped_keys_by_locale[locale.to_s].nil?
        grouped_keys_by_locale[locale.to_s].each do |key|
          splitted_key = key.split('.')

          # Case if translation is not unique (it can be normally saved without special indentation requirements)
          if splitted_key.count == 2
            translation_key = key.partition('.').last
            translation = TRANSLATION_STORE[key]

            file_content.insert(position, "  #{translation_key}: #{translation}\n")

          # Case if translation is unique (Special indentation requirements)
          elsif splitted_key.count > 2
            unique_translation = "     # Variable for unique translation with indentation

            splitted_key.each_with_index do |subkey, index|
              unless index == 0
                indentation = '  ' * index
                translation = TRANSLATION_STORE[key]
```

```
            # Last 'subkey' after colon has translation
            if index == splitted_key.count - 1
              unique_translation += "#{indentation}#{subkey}: #{translation}\n"
            else
              unique_translation += "#{indentation}#{subkey}:\n"
            end
          end
        end

        file_content.insert(position, unique_translation)
      end
    end
  end

  # Insert locale in front of all its translations if it wasnt already there
  file_content.insert(0, "#{locale}:\n") unless exists
end

# Open Kakimasu backup file and write new content
File.open(backup_file, 'w+') do |f|
  f.write(file_content)
end

# Success message
puts "
puts "*******************************************************"
puts "Translations successfully stored in #{backup_file}!"
puts "*******************************************************"
  end
 end
end
```

Kvalifikācijas darbs „***Ruby paplašinājums ērtai tulkojumu pārvaldei***” izstrādāts Latvijas Universitātes Datorikas fakultātē.

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

Autors: ***Roberts Groza*** _____ **___.05.2017**.

Rekomendēju darbu aizstāvēšanai
Darba vadītāja: ***Doc. Vineta Arnicāne*** _____ **___.05.2017.**

Recenzents: ***Dr. dat. Zane Bičevska***

Darbs iesniegts Datorikas fakultātē 29.05.2017.
Dekāna pilnvarotā persona:
docente, Dr.dat. Darja Solodovņikova_____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē
___.06.2017. prot. Nr. _____

Komisijas sekretārs(-e): _____