LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

UZŅĒMUMU REĢISTRĀCIJAS SISTĒMAS SERVERA PUSES IZSTRĀDE MIKROSERISU ARHITEKTŪRĀ

KVALIFIKĀCIJAS DARBS

Autors: Alberts Bērziņš
Studenta apliecības Nr.: ab16115
Darba vadītājs: Valdis Prodnieks

RĪGA, 2018

ANOTĀCIJA

Kvalifikācijas darba ietvaros tika veikta uzņēmumu reģistrācijas sistēmas servera puses izstrāde mikroservisu arhitektūrā.

Sistēma sastāv no 2 daļām: lietotāju puse, ar kuru mijiedarbojas lietotājs, un servera puse, kura veic datu apstrādi. Šīs darbs apraksta servera pusi, tā darbības algoritmus, izmantotās tehnoloģijas, struktūru un sniedz kopējo informāciju par izstrādes procesu. Darba mērķi ir:

- Uzlabot strādājošu, bet morāli novecojošu uzņēmumu reģistrācijas procesu;
- Padarīt programmatūras testēšanas un automātiskās kvalitātes nodrošināšanas aspektu caurspīdīgāku un saprotamāku klienta pārstāvjiem;
- Nodrošināt sistēmas automātisko piegādi un izvietošanu.

Izstrādē izmantotie rīki un tehnoloģijas atvieglo sistēmas uzturēšanu un palīdz nodrošināt mērogojamību. Tā kā sistēma ir izstrādāta mikroservisu arhitektūrā, izmaiņu ieviešana ir paātrināta, salīdzinot ar monolītām sistēmām.

Atslēgvārdi: uzņēmumu reģistrācija, mikroservisi, Spring Boot, Docker

ABSTRACT

Within the scope of qualification work author performed enterprise registration system back-end development in microservice architecture.

System consists of 2 parts: the front-end that the user interacts with and the back-end the processes the data. This work describes back-end side, its algorithms, used technologies, structure and provides general information about the development process. The goals of this work are:

- To improve working but morally obsolete enterprise registration process;
- To make the software testing and automatic quality assurance aspect more transparent and understandable for the customer;
- To ensure automatic delivery and deployment of the system.

The tools and technologies used in the development facilitate maintenance of the system and help ensure scalability. Since system is designed in microservice architecture, the introduction of changes is accelerated compared to monolithic systems.

Enterprise registration system back-end development in microservice architecture

Keywords: enterprise registration, microservices, Spring Boot

SATURS

IEVADS

Liela projekta ietvaros var sastapties ar daudzām grūtībām, kuras var būtiski ietekmēt projekta turpmāko attīstību un uzturēšanu, kā arī negatīvi atspoguļoties uz klienta apmierinātību ar gala produktu. Viena no tādām problēmām ir sistēmas arhitektūra, kura var sarežģīt izstrādes procesu, ja tā bija izvēlēta nepareizi un savlaicīgi nebija pārdomāta.

Tipiska timeklī bāzēta sistēma sastāv no vairākām komponentēm, kuras izpilda kaut kādu savu darba daļu. Piemēram, tā var būt lietotāju apstrāde, uzdevumu apstrādājošie servisi, notikumu reģistrēšanas sistēma, dažādas datu bāzes utt. Tāpēc, atšķirībā no klasiskas pieejas, kad visas komponentes tiek atdalītas viena no otras loģiskajā līmenī, parādījās un guva popularitāti mikroservisu arhitektūra. Šajā pieejā sistēma tiek komponēta no vairākiem maziem servisiem. Servisi tiek neatkarīgi darbināti, patstāvīgi mērogoti un atjaunoti.

Šīs projekts ir idejas un koncepta pierādījums, kura tēma ir uzņēmumu reģistrācija. Sistēma ir sadalīta komponentēs: lietotāju saskarne, uzņēmumu, lietotāju, dalībnieku, amatpersonu un adrešu servisi un datu bāze. Šī darba ietvaros ir izstrādāta servera puses funkcionalitāte, kurā ietilpst četri servisi neieskaitot uzņēmumu servisu un datu bāze.

Projektā tiek pielietota DevOps pieeja. Ar to saprotu ciešu un aktīvu sadarbību starp programmētājiem, testētājiem un administratoriem kopīga produkta izstrādei un uzturēšanai visu tā dzīves laiku. DevOps ietver sevī automatizāciju, kur vien iespējams (kompilācija, visu veidu testi, uzstādīšana dažādās vidēs(CI , CD)).

Ir izveidots Kubernetes klasteris, kurā automātiski, izmantojot Docker virtualizācijas rīku, tiek uzstādīta jaunākā sistēmas versija. Tas pēc labās industrijas prakses nodrošina izolētu un nokonfigurētu izstrādes un testēšanas vidi.

VĀRDNĪCA

Adresācijas objekts – pilsēta, novads, pagasts, ciems, iela vai māja.

API - ir iepriekš definētu klašu, procedūru, funkciju, struktūru un konstanšu kopums, kas tiek pasniegts kā pielikums (bibliotēkas, servisi), kuru iespējams izmantot ārējiem programmatūras produktiem.

Bcrypt - kriptogrāfiska funkcija, kura tiek izmantota paroles šifrēšanai. Atšķiras ar savu lēnu izpildes laiku, kas iet tikai par labu, jo dod mazāk iespējas uzbrucējiem uzminēt paroli.

CI (Continuous Integration) - Programmatūras izstrādes pieeja, kurā tiek veikta bieža automātiska programmatūras uzstādīšana un testēšana jaunas sistēmas komponenšu integrācijai un citu problēmu agrākai atklāšanai un labošanai.

CD (Continuous Delivery) - programmatūras izstrādes pieeja, kurā komandas ražo jaunas programmatūras versijas īsos laika intervālos, nodrošinot programmatūras izvietošanu ražošanas vidē jebkurā laikā.

Docker - atvērtā pirmkoda rīks, kas automatizē lietotņu izvietošanu programmatūras konteineros.

Git - atvērtā pirmkoda versiju kontroles sistēma.

GitLab - mājaslapa un Git repozitoriju pārvaldības sistēma ar papildiespējām, piemēram, kļūdu izsekošanas sistēma, iebūvēta CI/CD iespēja, lietotnes pārraudzība utt.

HTTP - Lietotņu protokols, kas paredzēts datu apmaiņai starp tīmekļa serveriem un pārlūkprogrammām.

HttpStatus - servera atbildes daļa HTTP protokolā.

Java - objektorientēta programmēšanas valoda.

Kubernetes – Docker konteineru klasteru pārvaldības rīks.

Mikroservisi - mūsdienīga pieeja serviss-orientētai arhitektūrai. Mikroserviss ir sistēma daļa vērsta uz kaut kādas problēmas risināšanu, kura ir spējīga funkcionēt neatkarīgi no pārējiem servisiem.

Proxy - starpniekserveris, kas atrodas starp lietotāja lietojumprogrammu un īsto serveri.

Swagger - atvērtā pirmkoda rīks, kas palīdz izstrādātājiem izplānot, realizēt un dokumentēt tīmekļa servisus (API).

URL – standartizēta resursa adrese internetā.

1. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA

1.1. Ievads

1.1.1. Nolūks

Programmatūras prasību specifikācijas nolūks ir aprakstīt katru no būtiskām programmatūras prasībām tā, lai prasību apmierināšanu būtu iespējams verificēt un validēt

Atbilstoši programmatūras prasību specifikācijai tiks izstrādāta programmatūra. Šīs dokuments paredzēts programmatūras pasūtītājiem, lai varētu precīzi noformulēt nepieciešamās prasības pret programmu, un programmas izstrādātājiem, lai precīzi zinātu izstrādājamās sistēmas funkcijas.

1.1.2. Darbības sfēra

Sistēma paredzēta uzņēmumu reģistrācijas procesa atvieglošanai un paātrināšanai un reģistrēto uzņēmumu uzskaitei. Sistēmā tiek glabāta informācija par jau reģistrētiem uzņēmumiem un iesniegtajiem pieteikumiem jaunu uzņēmumu reģistrācijai.

1.1.3. Saistība ar citiem dokumentiem

Dokumentu noformēšanā ievērotas standarta LVS 68:1996 “Programmatūras prasību specifikācijas ceļvedis” prasības.

1.1.4. Pārskats

Dokuments sastāv no 5 daļām:

- Pirmajā daļā ir ievadinformācija, kas satur dokumenta nolūku, mērķi, definīciju skaidrojumu, citu dokumentu izmantošanu dokumenta tapšanas laikā;
- Otrajā daļā tiek dots ieskats, par vispārējo produkta struktūru, kā funkcijas, ierobežojumi, kā arī atkarības, no citiem faktoriem;
- Trešā nodaļā tiek norādītas funkcionālās prasības uz izstrādājamo programmu;
- Ceturtajā nodaļā aprakstīta ārējā saskarne;
- Piektajā nodaļā aprakstītas nefunkcionālās prasības uz izstrādājamo produktu.

1.2. Vispārējais apraksts

1.2.1. Esošā stāvokļa apraksts

Šobrīd uzņēmumu reģistrācija notiek pēc šāda scenārija:

- Manuāla maksājumu veikšana internetbankā;

- Formas (veidlapas) pildīšana ar roku;
- Pārējo dokumentu sagatavošana;
- Dokumentu elektroniskā parakstīšana;
- Dokumentu iesniegšana, izmantojot e-pakalpojumus.

Veidlapas joprojām ir jāaizpilda ar roku, tajās ir sarežģīti orientēties ne paša labākā projektējuma dēļ, kas noved pie lielas iespējas pieļaut kļūdu. Veidlapu ir daudz, kas vēl vairāk saasina šo problēmu. Elektroniskā parakstīšana nav tik sarežģīts process, toties arī var izraisīt problēmas, bet tas atkal ir saistīts ar iepriekšējo problēmu – veidlapas. Un galu galā tos dokumentus vajag iesniegt, kas laikam ir vienkāršākais no šī saraksta.

Var padomāt, ka pēc iesniegšanas grūtākais jau ir garām, bet ar ļoti lielu varbūtību iesniegtajos dokumentos bija pieļauta kļūda, bet uzzināt to var tikai pēc iesnieguma izskatīšanas, kā rezultātā visu šo procesu vajag atkārtot.

Visam šīm procesam ir liels optimizēšanas potenciāls, uz ko arī ir vērsta kvalifikācijas darba ietvaros izstrādātā sistēma. Reģistrācijas procesu ir iespējams realizēt pēc pašapkalpošanās principa, kurā dati tiks ievadīti tikai vienu reizi, un kurā nebūs lieku darbību.

1.2.2. Produkta perspektīva

Tā kā sistēmas realizācija notiek mikroservisu arhitektūrā, sistēmas pilnvērtīgam darbam ir nepieciešami visu mikroservisu vienlaicīgais darbs. Ārpus kvalifikācijas darba tika izstrādāti vēl 2 mikroservisi: lietotāju puse un pamatdatu serviss.

Lietotāju puse jeb lietotāju saskarne atbild par mijiedarbību starp lietotāju un visu pārējo sistēmu. Tā ir tīmekļa lietotne, kura saziņai ar pārējiem servisiem izmanto HTTP protokolu.

Pamatdatu serviss ir lietojumprogrammas saskarne (API), kuras uzdevums ir uzņēmumu pamatdatu apstrāde, galvenās funkcijas tajā: pamatdatu saglabāšana, dzēšana, izvadīšana pēc dažādiem parametriem (nosaukums, reģistrācijas numurs, iesniedzējs utt.)

Sistēmai nav nekādas atkarības no ārējām sistēmām.

1.2.3. Produkta funkcijas

Sistēmas galvenās funkcijas ir uzņēmumu reģistrācija un uzskaite, kurām ir sekojošas apakšfunkcijas un palīgfunkcijas:

- Lietotāju funkcijas - lietotāju reģistrācija, autentifikācija un autorizācija, profila dzēšana;
- Uzņēmumu reģistrācijas funkcijas:
  - Uzņēmumu pamatdatu apstrādes funkcijas - pamatdati ietver sevī informāciju par pašu uzņēmumu (nosaukums, reģistrācijas numurs utt.);
  - Uzņēmumu dalībnieku apstrādes funkcijas;

- Uzņēmumu amatpersonu apstrādes funkcijas;
- Adrešu apstrādes funkcijas, atbilstoši Latvijas adrešu klasifikatoru reģistram

1.2.4. Lietotāju raksturiezīmes

Par sistēmas lietotāju var būt jebkurš cilvēks, kam ir vēlme reģistrēt savu uzņēmumu. Lietotājiem nepieciešama elementārākā prasme darbam ar datoru un iepriekš uzstādīto pārlūkprogrammu, kā arī zināšanas par uzņēmuma reģistrācijas juridiskiem aspektiem.

1.2.5. Vispārējie ierobežojumi

Tiek pieņemts, ka lietojumprogrammas saskarnes būs pieejamas tikai sistēmas lietotāju saskarnes mikroservisam, bet pārējiem resursiem piekļuve būs ierobežota.

Personu identifikācijai tiek izmantots Latvijas Republikas personas kods. Gadījumā, ja lietotājam personas koda nav (piemēram, ārvalsts iedzīvotāji), identifikācija notiek pēc personas apliecinošā dokumenta. Rezultātā tiek pieņemts, ka mainoties personu apliecinošam dokumentam, cilvēka identitāti nav iespējams noteikt un šīs cilvēks tiek uzskatīts par jaunu, nepazīstamu sistēmai, personu. Kā arī tiek pieņemts, ka personas kodi netiek mainīti.

1.3. Funkcionālās prasības

Katrā funkcijā tiek pieņemts, ka apstrādes laikā vispirms tiek pārbaudīti visi nepieciešami validācijas kritēriji. Tas nozīme, ka vispirms, tiek pārbaudīts, vai visi obligātie lauki ir aizpildīti. Tad tiek pārbaudīts, vai ievadītie dati atbilst ievaddatu prasībām. Gadījumā, ja kaut kādas validācijas kritēriji nav izieti, piemēram, lietotājs neievadīja obligātu lauku, tad funkcijas darbība tiek pārtraukta un tiek izvadīts attiecīgais kļūdas paziņojums.

Tā kā katra funkcija ir lietojumprogrammas saskarnes funkcija, kuras izsaukšana notiek, izmantojot HTTP protokolu, funkciju izvade sastāv no funkcijas izpildes statusa (HttpStatus) un funkcijas datu apstrādes rezultāta (neobligāts). Šīs nodaļas ietvaros funkciju izvade tiek noformēta pēc šāda principa: *HttpStatus – paskaidrojums – datu apstrādes rezultāts.*

funkcijas izvadē var būt kā funkcijas izpildes status (HttpStatus), tā arī funkcijas datu apstrādes rezultāts.

1.3.1. Lietotāja funkcijas

1.3.1.1. Lietotāju reģistrācija

Ievads
Funkcija nepieciešama jaunu lietotāju reģistrācijai.
Ievade

Tā kā personu identifikācija notiek pēc personas koda vai personas apliecinošā dokumenta numura, dokumenta valsts, dokumenta institūcijas un dokumenta veida ir iespējami 2 dažādi ievaddatu veidi:

- Vārds
- Uzvārds
- Lietotājvārds
- Parole
- E-pasts
- Telefona numurs (neobligāts)
- Dzimšanas datums (neobligāts)

- Personas kods

vai

- Personas apliecinošā dokumenta numurs
- Personas apliecinošā dokumenta valsts
- Personas apliecinošā dokumenta datums
- Personas apliecinošā dokumenta veids
- Personas apliecinošā dokumenta institūcija

Tā kā dati tiek padoti vienā un tajā pašā funkcijā, viennozīmīgai datu formāta identificēšanai ir ieviests papildus lauks "irPK" ar loģisko Būla (boolean) tipu, kas norāda, kādu formātu sistēmai ir jāapstrādā.

Apstrāde

Tiek pārbaudīts vai lietotājvārds vai e-pasts nav aizņemts. Ja persona ar dotajiem datiem nav atrasta sistēmā – tā tiek izveidota. Pretējā gadījumā tiek pārbaudīts, vai padotie dati atbilst sistēmā glabātajiem un vai persona nav jau piereģistrēta. Tad notiek jauna lietotāja datu saglabāšana datu bāzē.
Parole pirms saglabāšanas tiek šifrēta, izmantojot "bcrypt" funkciju.

Izvade

Ir iespējamas šādas funkciju izvades:

- 200 – veiksmīga operācijas izpilde – lietotājvārds, vārds, uzvārds, e-pasts, telefons, irPk, personas kods, dzimšanas datums, dokumenta numurs, dokumenta valsts, dokumenta veids, dokumenta datums dokumenta institūcija
- 405 – lietotājvārds vai e-pasts ir aizņemts

- 406 – personas dati neatbilst sistēmā glabātajiem
- 407 – personai jau reģistrēts profils
- 408 – neparedzēta sistēmas kļūda (nepieciešams sazināties ar administratoru)

1.3.1.2. Lietotāja autentifikācija

Ievads
Funkcija nepieciešama lietotāju autentifikācijai
Ievade
• Lietotāja vārds • Parole
Apstrāde
Tiek pārbaudīts vai sistēmā ir reģistrēts lietotājs ar ievadīto lietotāja vārdu un vai šīm lietotājam parole sakrīt ar ievadīto. Ja šāds lietotājs ir atrasts, notiek lietotāja datu atlase no datu bāzes, lai atgrieztu informāciju par lietotāju funkcijas izvadē.
Izvade
• 200 – veiksmīga autentifikācija - lietotājvārds, vārds, uzvārds, e-pasts, telefons, irPk, personas kods, dzimšanas datums, dokumenta numurs, dokumenta valsts, dokumenta • 400 – lietotājs nav atrasts

1.3.1.3. Lietotāja datu atjaunošana

Ievads
Funkcija nepieciešama lietotāja datu atjaunošanai, t.sk. personas datu rediģēšanai. Ir iespējams atjaunot jebkuru lietotāja informāciju, izņemot personu identificēšanas parametrus.
Ievade
Ir nepieciešams ievadīt jaunu lietotāja informāciju, t.sk. tos datus, kuri paliek nemainīti. Tā kā lietotāja profils ir piesaistīts pie reālas personas, un personu identifikācija notiek pēc personas koda vai personas apliecinošā dokumenta numura un dokumenta valsts, ir iespējami 2 dažādi ievaddatu veidi: • Vārds • Uzvārds • Lietotājvārds • Parole • E-pasts • Telefona numurs (neobligāts) • Dzimšanas datums (neobligāts)

• Personas kods vai	• Personas apliecinošā dokumenta numurs • Personas apliecinošā dokumenta valsts • Personas apliecinošā dokumenta datums • Personas apliecinošā dokumenta veids • Personas apliecinošā dokumenta institūcija
Pēc “Lietotāja reģistrācijas” funkcijas principa ir ievests papildlauks “irPK”	
Apstrāde	
Sākumā notiek personas meklēšana pēc personas identifikācijas parametriem. Tad notiek pārbaude uz to, vai personai ir reģistrēts lietotāja profils. Tad tiek pārbaudīts, vai lietotāja vārds vai e-pasts nav jau aizņemts ar to nosacījumu, ka vecais lietotāja vārds vai e-pasts var palikt. Notiek jaunu lietotāja datu saglabāšana datu bāzē.	
Izvade	
• 200 – veiksmīga operācijas izpilde - lietotājvārds, vārds, uzvārds, e-pasts, telefons, irPk, personas kods, dzimšanas datums, dokumenta numurs, dokumenta valsts, dokumenta • 404 – persona nav atrasta • 405 – lietotāja profils nav atrasts • 406 – lietotājvārds ir aizņemts • 407 – e-pasts ir aizņemts • 408 – neparedzēta sistēmas kļūda (nepieciešams sazināties ar administratoru)	

1.3.1.4. E-pasts aizņemts

Ievads
Funkcija nepieciešama , lai pārbaudītu, vai e-pasts ir aizņemts (jau ir reģistrēts lietotājs ar ievadīto e-pastu). Funkcija tiek izmantota, lai lietotāju pusē reālā laikā paziņotu lietotājam par to, ka ievadītais e-pasts ir aizņemts.
Ievade
• E-pasts
Apstrāde
Notiek reģistrētā lietotāja meklēšana ar ievadīto e-pastu datu bāze.
Izvade
• 200 – veiksmīga operācijas izpilde – Būla tipa vērtība

1.3.1.5. Lietotājvārds aizņemts

Ievads
Funkcija nepieciešama , lai pārbaudītu, vai lietotājvārds ir aizņemts (jau ir reģistrēts lietotājs ar ievadīto lietotājvārdu). Funkcija tiek izmantota, lai lietotāju pusē reālā laikā paziņotu lietotājam par to, ka ievadītais lietotājvārds ir aizņemts.
Ievade
• Lietotājvārds
Apstrāde
Notiek reģistrētā lietotāja meklēšana ar ievadīto lietotājvārdu datu bāze.
Izvade
• 200 – veiksmīga operācijas izpilde – Būla tipa vērtība

1.3.1.6. Lietotāja identifikatora saņemšana

Ievads
Funkcija nepieciešama, lai saņemtu lietotāja identifikatoru, kurš tiek izmantots citu funkciju izpildes laikā.
Ievade
• Lietotājvārds
Apstrāde
Notiek reģistrētā lietotāja meklēšana ar ievadīto lietotājvārdu datu bāze. Ja lietotājs ir atrasts, tiek atlasīts lietotāja identifikators.
Izvade
• 200 – veiksmīga operācijas izpilde – lietotāja identifikators • 400 – lietotājs nav atrasts

1.3.1.7. Lietotāja profila dzēšana

Ievads
Funkcija nepieciešama lietotāju profila dzēšanai. Izpildes laikā netiek dzēsta informācija par personu.
Ievade
• Lietotājvārds
Apstrāde
Notiek pārbaude uz to, vai sistēmā ir lietotājs ar ievadīto lietotājvārdu. Ja lietotājs ir atrasts, viņa profils (lietotājvārds, parole, telefona numurs, e-pasts) tiek dzēst no datu bāzes.
Izvade
• 200 – veiksmīga operācijas izpilde • 404 – lietotājs nav atrasts

1.3.2. Uzņēmumu dalībnieku funkcijas

1.3.2.1. Fiziskās personas pievienošana

Ievads
Uzņēmuma reģistrācijas laikā ir nepieciešams norādīt uzņēmuma dibināšanas dalībniekus. Šī funkcija nodrošina fiziskās personas pievienošanu.
Ievade
Tā kā personu identifikācija notiek pēc personas koda vai personas apliecinošā dokumenta numura, dokumenta valsts, dokumenta institūcijas un dokumenta veida ir iespējami 2 dažādi ievaddatu veidi: • Uzņēmuma identifikators • Vārds • Uzvārds • Dzimšanas datums (neobligāts) • Personas kods vai • Personas apliecinošā dokumenta numurs • Personas apliecinošā dokumenta valsts • Personas apliecinošā dokumenta datums • Personas apliecinošā dokumenta veids • Personas apliecinošā dokumenta institūcija Tā kā dati tiek padoti vienā un tajā pašā funkcijā, viennozīmīgai datu formāta identificēšanai ir ieviests papildus lauks “irPK” ar loģisko Būla (boolean) tipu, kas norāda, kādu formātu sistēmai ir jāapstrādā.
Apstrāde
Tiek pārbaudīts, vai eksistē uzņēmums ar ievadīto identifikatoru. Sistēmā tiek meklēta persona ar ievadītiem datiem. Ja persona bija atrasta, tiek pārbaudīts, vai visi ievadītie personas dati sakrīt ar sistēmā saglabātiem, un vai persona jau nav pievienota dotajam uzņēmumam kā dalībnieks. Pretējā gadījumā datu bāzē tiek saglabāta jauna persona. Datu bāzē tiek saglabāta jauna dalībnieka informācija.
Izvade
• 200 – veiksmīga operācijas izpilde • 406 – nav atrasts uzņēmums • 408 – dalībnieks jau ir pievienots • 409 – neparedzēta sistēmas kļūda (nepieciešams sazināties ar administratoru) • 410 – ievadītie personas dati nav korekti

1.3.2.2. Juridiskās personas pievienošana

Ievads
Uzņēmuma reģistrācijas laikā ir nepieciešams norādīt uzņēmuma dibināšanas dalībniekus. Šī funkcija nodrošina juridiskās personas pievienošanu.
Ievade
• Uzņēmuma identifikators • Reģistrācijas numurs • Nosaukums • Valsts
Apstrāde
Tiek pārbaudīts, vai eksistē uzņēmums ar ievadīto identifikatoru. Sistēmā tiek meklēta juridiska persona ar ievadītiem datiem. Ja persona nav atrasta un ievadītā valsts nav Latvija, tiek izveidota jauna persona (tiek pieņemts, ka sistēmā ir saglabāta informācija par visiem Latvijā reģistrētiem uzņēmumiem). Tiek pārbaudīts vai persona jau nav pievienota dotajam uzņēmumam kā dalībnieks. Datu bāzē tiek saglabāta jauna dalībnieka informācija.
Izvade
• 200 – veiksmīga operācijas izpilde • 406 – nav atrasts uzņēmums • 408 – dalībnieks jau pievienots • 409 – neparedzēta sistēmas kļūda (nepieciešams sazināties ar administratoru) • 410 – ievadītie personas dati nav korekti

1.3.2.3. Dalībnieku saraksta saņemšana (fiziskās personas)

Ievads
Funkcija nepieciešama uzņēmuma dalībnieku saraksta izvadīšanai (fiziskās personas)
Ievade
• Uzņēmuma identifikators
Apstrāde
Tiek meklēts uzņēmums ar ievadīto identifikatoru. Ja uzņēmums bija atrasts, tiek ģenerēts saraksts ar uzņēmuma dalībniekiem. Saraksts sastāv no ierakstiem ar sekojošo informāciju: dalībnieka identifikators, uzņēmuma identifikators, vārds, uzvārds, irPk, personas kods, dzimšanas datums, dokumenta numurs, dokumenta valsts, dokumenta veids, dokumenta datums dokumenta institūcija.
Izvade
• 200 – veiksmīga operācija izpilde – dalībnieku saraksts • 405 – uzņēmums nav atrasts

1.3.2.4. Dalībnieku saraksta saņemšana (juridiskās personas)

Ievads
Funkcija nepieciešama uzņēmuma dalībnieku saraksta izvadīšanai (juridiskās personas)
Ievade
• Uzņēmuma identifikators

Apstrāde
Tiek meklēts uzņēmums ar ievadīto identifikatoru. Ja uzņēmums bija atrasts, tiek ģenerēts saraksts ar uzņēmuma dalībniekiem. Saraksts sastāv no ierakstiem ar sekojošo informāciju: dalībnieka identifikators, uzņēmuma identifikators, juridiskās personas reģistrācijas numurs, juridiskās personas nosaukums, valsts
Izvade
• 200 – veiksmīga operācija izpilde – dalībnieku saraksts • 405 – uzņēmums nav atrasts

1.3.2.5. Dalībnieka dzēšana

Ievads
Funkcija nepieciešama viena dalībnieka dzēšanai
Ievade
• Dalībnieka identifikators
Apstrāde
Tiek pārbaudīts, vai eksistē dalībnieks ar ievadīto identifikatoru. Ja dalībnieks ir atrasts, informācija par to tiek dzēsta no datu bāzes.
Izvade
• 200 – veiksmīga operācijas izpilde • 405 – dalībnieks nav atrasts

1.3.2.6. Dalībnieku dzēšana

Ievads
Funkcija nepieciešama visu dalībnieku dzēšanai vienam uzņēmumam.
Ievade
• Uzņēmuma identifikators
Apstrāde
Tiek pārbaudīts, vai eksistē uzņēmums ar ievadīto identifikatoru. Ja uzņēmums ir atrasts, informācija par visiem tā dalībniekiem tiek dzēsta no datu bāzes.
Izvade
• 200 – veiksmīga operācijas izpilde • 405 – uzņēmums nav atrasts

1.3.2.7. Juridiskās personas datu pārbaude

Ievads
Funkcija nepieciešama juridiskās personas datu pārbaudei. Funkcija tiek izmantota ievaddatu validācijai lietotāju pusē. Par korektu juridisku personu tiek uzskatīta juridiska persona, kuras dati sakrīt ar sistēmā reģistrētiem, vai tā, kura vēl nav reģistrēta sistēmā. Izvade notiek Būla formātā, kur “true” ir korekts.
Ievade
• Valsts

• Nosaukums • Reģistrācijas numurs
Apstrāde
No datu bāzes tiek atlasīta juridiska persona, ja persona bija atrasta – dati ir korekti. Ja nosaukums vai reģistrācijas numurs ir aizņemts ievadītajā valstī – dati nav korekti, pretējā gadījumā – dati ir korekti.
Izvade
• 200 – veiksmīga operācijas izpilde – “true” vai “false”

1.3.2.8. Fiziskās personas ar personas kodu datu pārbaude

Ievads
Funkcija nepieciešama fiziskās personas ar personas kodu datu pārbaudei. Funkcija tiek izmantota ievaddatu validācijai lietotāju pusē. Par korektu fizisku personu tiek uzskatīta fiziska persona, kuras dati sakrīt ar sistēmā reģistrētiem, vai tā, kura vēl nav reģistrēta sistēmā. Izvade notiek Būla formātā, kur “true” ir korekts.
Ievade
• Vārds • Uzvārds • Personas kods
Apstrāde
No datu bāzes tiek atlasīta fiziska persona ar ievadīto personas kodu. Ja persona nebija atrasta – dati ir korekti. Ja atrastajai personai vārds vai uzvārds nesakrīt ar ievadīto – dati nav korekti, pretējā gadījumā dati ir korekti
Izvade
• 200 – veiksmīga operācijas izpilde – “true” vai “false”

1.3.2.9. Fiziskās personas bez personas koda datu pārbaude

Ievads
Funkcija nepieciešama fiziskās personas bez personas koda datu pārbaudei. Funkcija tiek izmantota ievaddatu validācijai lietotāju pusē. Par korektu fizisku personu tiek uzskatīta fiziska persona, kuras dati sakrīt ar sistēmā reģistrētiem, vai tā, kura vēl nav reģistrēta sistēmā. Izvade notiek Būla formātā, kur “true” apzīmē korektus datus.
Ievade
• Vārds • Uzvārds • Personas apliecinošā dokumenta numurs • Personas apliecinošā dokumenta valsts • Personas apliecinošā dokumenta institūcija • Personas apliecinošā dokumenta veids • Personas apliecinošā dokumenta datums • Dzimšanas datums • Dzīvesvietas valsts

Apstrāde
No datu bāzes tiek atlasīta fiziska persona ar ievadītiem personas apliecinošā dokumenta datiem (numurs, valsts, institūcija, veids). Ja persona nebija atrasta – dati ir korekti. Ja atrastajai personai pārējie dati nesakrīt ar sistēmā reģistrētiem – dati nav korekti, pretējā gadījumā dati ir korekti.
Izvade
• 200 – veiksmīga operācijas izpilde – “true” vai “false”

1.3.3. Uzņēmuma amatpersonu funkcijas

1.3.3.1. Amatpersonas pievienošana

Ievads
Uzņēmuma reģistrācijas laikā ir nepieciešams norādīt uzņēmuma amatpersonas, ko nodrošina šī funkcija.
Ievade
Tā kā personu identifikācija notiek pēc personas koda vai personas apliecinošā dokumenta numura, dokumenta valsts, dokumenta institūcijas un dokumenta veida ir iespējami 2 dažādi ievaddatu veidi: • Uzņēmuma identifikators • Vārds • Uzvārds • Dzimšanas datums (neobligāts) • Amats • Tiesības pārstāvēt uzņēmumu • Ar cik valdes locekļiem ir iespējams pārstāvēt uzņēmumu • Personas kods vai • Personas apliecinošā dokumenta numurs • Personas apliecinošā dokumenta valsts • Personas apliecinošā dokumenta datums • Personas apliecinošā dokumenta veids • Personas apliecinošā dokumenta institūcija

Tā kā dati tiek padoti vienā un tajā pašā funkcijā, viennozīmīgai datu formāta identificēšanai ir ieviests papildus lauks “irPK” ar loģisko Būla (boolean) tipu, kas norāda, kādu formātu sistēmai ir jāapstrādā.
Apstrāde
Tiek pārbaudīts, vai eksistē uzņēmums ar ievadīto identifikatoru. Sistēmā tiek meklēta persona ar ievadītiem datiem. Ja persona bija atrasta, tiek pārbaudīts, vai visi ievadītie personas dati sakrīt ar sistēmā saglabātiem, un vai persona jau nav pievienota dotajam uzņēmumam kā amatpersona. Pretējā gadījumā datu bāzē tiek saglabāta jauna persona. Datu bāzē tiek saglabāta jauna amatpersonas informācija.
Izvade
• 201 – veiksmīga operācijas izpilde • 405 – nekorektas tiesības • 406 – nav atrasts uzņēmums • 408 – amatpersona jau pievienota • 409 – neparedzēta sistēmas kļūda (nepieciešams sazināties ar administratoru) • 410 – ievadītie personas dati nav korekti

1.3.3.2. Amatpersonu saraksta saņemšana

Ievads
Funkcija nepieciešama uzņēmuma amatpersonu saraksta izvadīšanai.
Ievade
• Uzņēmuma identifikators
Apstrāde
Tiek meklēts uzņēmums ar ievadīto identifikatoru. Ja uzņēmums bija atrasts, tiek ģenerēts saraksts ar uzņēmuma amatpersonām. Saraksts sastāv no ierakstiem ar sekojošo informāciju: amatpersonas identifikators, uzņēmuma identifikators, tiesības pārstāvēt uzņēmumu, vārds, uzvārds, irPk, personas kods, dzimšanas datums, dokumenta numurs, dokumenta valsts, dokumenta veids, dokumenta datums dokumenta institūcija.
Izvade
• 200 – veiksmīga operācija izpilde – dalībnieku saraksts • 405 – uzņēmums nav atrasts

1.3.3.3. Amatpersonas dzēšana

Ievade
Funkcija nepieciešama vienas amatpersonas dzēšanai
Izvade
• Amatpersonas identifikators
Apstrāde
Tiek pārbaudīts, vai eksistē amatpersona ar ievadīto identifikatoru. Ja amatpersona ir atrasta, informācija par to tiek dzēsta no datu bāzes.
Izvade
• 200 – veiksmīga operācijas izpilde

• 405 – dalībnieks nav atrasts

1.3.3.4. Amatpersonu dzēšana

Ievads
Funkcija nepieciešama visu amatpersonu dzēšanai vienam uzņēmumam.
Ievade
• Uzņēmuma identifikators
Apstrāde
Tiek pārbaudīts, vai eksistē uzņēmums ar ievadīto identifikatoru. Ja uzņēmums ir atrasts, informācija par visām tā amatpersonām tiek dzēsta no datu bāzes.
Izvade
• 200 – veiksmīga operācijas izpilde • 405 – uzņēmums nav atrasts

1.3.4. Adrešu funkcijas

Adrešu funkcijas nepieciešamas, lai korekti ievadītu adreses lietotāju pusē. Adreses tiek apstrādātas atbilstoši Latvijas Republikas adrešu klasifikatoru reģistram, kur katrai ģeogrāfiskai vietai ir savs unikāls identifikators, ir viennozīmīgi noteicams kurā lielākajā ģeogrāfiskā vietā tā atrodas un kādās mazākās vietās tā ir sadalīta.

1.3.4.1. Pilsētu saraksta saņemšana

Ievads
Funkcija nepieciešama pilsētu izvadīšanai. Funkcijai nav ievaddatu.
Apstrāde
No datu bāzes tiek atlasītas visas pilsētas un tiek izveidots saraksts no ierakstiem ar sekojošu struktūru: identifikators, nosaukums
Izvade
• 200 – veiksmīga operācijas izpilde – pilsētu saraksts

1.3.4.2. Novadu saraksta saņemšana

Ievads
Funkcija nepieciešama novadu izvadīšanai. Funkcijai nav ievaddatu.
Apstrāde
No datu bāzes tiek atlasīti visi novadi un tiek izveidots saraksts no ierakstiem ar sekojošu struktūru: identifikators, nosaukums
Izvade
• 200 – veiksmīga operācijas izpilde – novadu saraksts

1.3.4.3. Pilsētu saraksta saņemšana ar vecāka identifikatoru

Ievads

Funkcija nepieciešama pilsētu izvadīšanai, kuras atrodas kādā lielākajā adresācijas objektā (novadā).
Ievads
• Vecāka identifikators
Apstrāde
No datu bāzes tiek atlasītas visas pilsētas, kuras atrodas iekš adresācijas objekta ar padoto identifikatoru, un tiek izveidots saraksts no ierakstiem ar sekojošu struktūru: identifikators, nosaukums
Izvade
• 200 – veiksmīga operācijas izpilde – pilsētu saraksts

1.3.4.4. Pagastu saraksta saņemšana ar vecāka identifikatoru

Ievads
Funkcija nepieciešama pagastu izvadīšanai, kuri atrodas kādā lielākajā adresācijas objektā (novadā).
Ievads
• Vecāka identifikators
Apstrāde
No datu bāzes tiek atlasīti visi pagasti, kuri atrodas iekš adresācijas objekta ar padoto identifikatoru, un tiek izveidots saraksts no ierakstiem ar sekojošu struktūru: identifikators, nosaukums
Izvade
• 200 – veiksmīga operācijas izpilde – pagastu saraksts

1.3.4.5. Ciemu saraksta saņemšana ar vecāka identifikatoru

Ievads
Funkcija nepieciešama ciemu izvadīšanai, kuri atrodas kādā lielākajā adresācijas objektā (novadā, pagastā).
Ievads
• Vecāka identifikators
Apstrāde
No datu bāzes tiek atlasīti visi ciemi, kuri atrodas iekš adresācijas objekta ar padoto identifikatoru, un tiek izveidots saraksts no ierakstiem ar sekojošu struktūru: identifikators, nosaukums
Izvade
• 200 – veiksmīga operācijas izpilde – ciemu saraksts

1.3.4.6. Ielu saraksta saņemšana ar vecāka identifikatoru

Ievads
Funkcija nepieciešama ielu izvadīšanai, kuras atrodas kādā lielākajā adresācijas objektā (pilsētā, pagastā, ciemā).

Ievads
• Vecāka identifikators
Apstrāde
No datu bāzes tiek atlasītas visas ielas, kuras atrodas iekš adresācijas objekta ar padoto identifikatoru, un tiek izveidots saraksts no ierakstiem ar sekojošu struktūru: identifikators, nosaukums
Izvade
• 200 – veiksmīga operācijas izpilde – ielu saraksts

1.3.4.7. Māju saraksta saņemšana ar vecāka identifikatoru

Ievads
Funkcija nepieciešama māju izvadīšanai, kuras atrodas kādā lielākajā adresācijas objektā (novadā, pilsētā, pagastā, ciemā, ielā).
Ievads
• Vecāka identifikators
Apstrāde
No datu bāzes tiek atlasītas visas mājas, kuras atrodas iekš adresācijas objekta ar padoto identifikatoru, un tiek izveidots saraksts no ierakstiem ar sekojošu struktūru: identifikators, nosaukums
Izvade
• 200 – veiksmīga operācijas izpilde – māju saraksts

1.3.4.8. Dzīvokļu saraksta saņemšana ar vecāka identifikatoru

Ievads
Funkcija nepieciešama dzīvokļu izvadīšanai, kuri atrodas kādā lielākajā adresācijas objektā (mājā).
Ievads
• Vecāka identifikators
Apstrāde
No datu bāzes tiek atlasīti visi dzīvokļi, kuri atrodas iekš adresācijas objekta ar padoto identifikatoru, un tiek izveidots saraksts no ierakstiem ar sekojošu struktūru: identifikators, nosaukums
Izvade
• 200 – veiksmīga operācijas izpilde – dzīvokļu saraksts

1.3.4.9. Vecāka adresācijas objekta saņemšana

Ievads
Funkcija nepieciešama informācijas par vecāku adresācijas objektu atrašanai, piemēram, uzzināt kurā ielā atrodas konkrētā māja.
Ievade
• Objekta identifikators

Apstrāde
Tiek pārbaudīts vai datu bāzē ir objekts ar ievadīto identifikatoru. Ja tāds objekts ir atrasts, tiek meklēts šī objekta vecākais.
Izvade
• 200 – veiksmīga operācijas izpilde – ievadītā adresācijas objekta identifikators, vecāka objekta identifikators, tips, nosaukums, saīsināts nosaukums, papildinformācija. • 400 – adresācijas objekts nav atrasts

1.3.4.10. Pilnas adreses ģenerēšana

Ievads
Funkcija nepieciešama pilnas adreses ģenerēšanai teksta formātā.
Ievade
• Adresācijas objekta identifikators
Apstrāde
Tiek pārbaudīts, vai datu bāzē ir saglabāts adresācijas objekts ar doto identifikatoru. Adreses ģenerēšana notiek tikai no dzīvokļa vai mājas, tāpēc sākumā tiek pārbaudīts vai padotais objekts ir dzīvoklis vai māja. Tad rekursīvi tiek atrasts objekta vecākais līdz būs sasniegta pilsēta vai novads, un pakāpeniski tiek ģenerēta teksta formāta adrese. Formāts: *māja-dzīvoklis iela, pilsēta* vai *māja-dzīvoklis iela, ciems, pagasts, novads.* Dzīvoklis, iela, ciems un pagasts var nebūt.
Izvade
• 200 – veiksmīga operācijas izpilde – adrese teksta formātā • 400 – adresācijas objekta nav atrasts

1.3.4.11. Valstu saraksta saņemšana

Ievads
Funkcija nepaciešama valstu un valstu kodu saraksta saņemšanai, atbilstoši ISO 3166 standartam.
Ievade
• Valoda, iespējamās vērtības: ”lv” vai “en”
Apstrāde
No datu bāzes tiek atlasītas visas valstis lietotāja pieprasītajā valodā. Tiek izveidots saraksts no ierakstiem ar sekojošu struktūru: valsts kods, valsts nosaukums
Izvade
• 200 – veiksmīga operācijas izpilde – valstu saraksts • 400 – nekorekta valoda

1.4. Ārējā saskarne

1.4.1. Lietotāja saskarne

Šajā sistēmā lietojumprogrammas saskarnēs (API) tiek pielietots Swagger rīks, kas ļauj grafiskā veidā apskatīt lietojumprogrammas saskarnes funkcijas, to izsaukšanas veidus, parametrus un izvades formātus, kā arī manuāli tās palaist.

Servera puses programmatūrai parasti nav tiešas saskarnes ar lietotāju, jo lietojumprogrammas saskarnes nav domāti publiskai lietošanai. Tomēr ir daži gadījumi, kad automātiski ģenerētā Swagger saskarne tiek publicēta:

- ja serviss ir publicējams lietotajam;
- ja gribam publicēt servisa dokumentāciju un testēšanas saskarni, piemēram, izstrādes un testa vidē. Tomēr produkcijas vidē piekļuve Swagger saskarnei tiek deaktivēta.

Šī sistēma ir tas gadījums, kad servisi nav publiski pieejami, bet izstrādes atvieglošanai Swagger tiek izmantots.

1.5. Nefunkcionālās prasības

1.5.1. Drošība

Lietotāja parolēm jāglabājas šifrēta veidā. Šifrēšanas algoritmam jābūt vienā virzienā, t.i. šifrēto paroli nav iespējams atšifrēt.

1.5.2. Uzturamība

Mikroservisiem jāsadala sistēmas iespējas dažādos domēnos pēc “Domain Driven Design” principa. Tas nozīmē, ka katram mikroservisam jāatbild par kaut kādas konkrētas sistēmas funkcionalitātes apgabalu.

Mikroservisu arhitektūrā pareizi izveidota sistēma ir labāk uzturama, nekā monolīta sistēma, jo izmaiņas sadalās pa atsevišķiem servisiem, kurus var ieviest produkcijā diezgan neatkarīgi vienu no otra. Līdz ar to izmaiņas var ieviest ātrākā laikā.

2. PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS

2.1. Ievads

2.1.1. Nolūks

Šīs dokuments ir programmatūras projektējuma apraksts projektam "Uzņēmumu reģistrācijas sistēmas servera puses izstrāde mikroservisu arhitektūrā". Tā nolūks ir aprakstīt sistēmas projektējumu (uzbūvi) atbilstoši izvirzītajām prasībām. Dokuments paredzēts sistēmas izstrādātājiem kā tehniskās specifikācijas un apraksta palīgmateriāls.

2.1.2. Darbības sfēra

Sistēma paredzēta uzņēmumu reģistrācijas procesa atvieglošanai un paātrināšanai un reģistrēto uzņēmumu uzskaitei.

Sistēma tiek veidota kā lielākas sistēmas daļa un satur sevī servera puses funkcionālo daļu. Tā tiek veidota uz HTTP/Java bāzes kā interneta lapas. Dati tiks glabāti PostgreSQL datu bāzē.

2.1.3. Saistība ar citiem dokumentiem

Dokumenta noformēšanā ievērotas standarta LVS 72:1996 "Ieteicamā prakse programmatūras projektējuma aprakstīšanai" prasības.

2.1.4. Pārskats

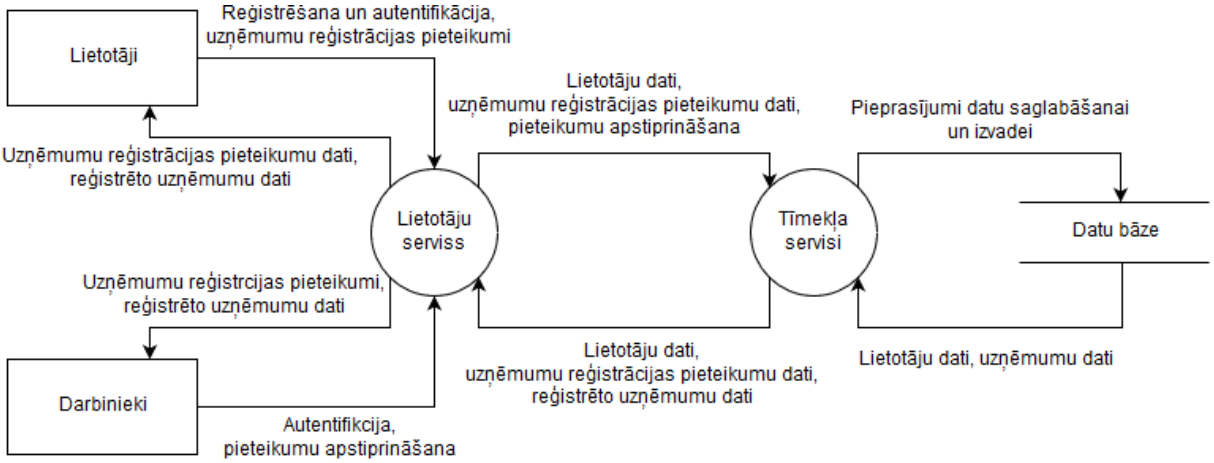
Programmatūras projektējuma apraksts sastāv no 4 daļām:

- Ievads – informācija, kas satur dokumenta nolūku, darbības sfēru, kā arī norāda uz saistību uz citiem dokumentiem;
- Diagrammas – sniedz grafisku reprezentāciju par to, kā sistēma darbojas;
- Datu bāzes projektējums – apraksta un grafiski reprezentē datu bāzes struktūru;
- Lietotāja ekrāna formu projektējums – pārskats par sistēmas grafisko saskarni.

2.2. Procesu projektējums

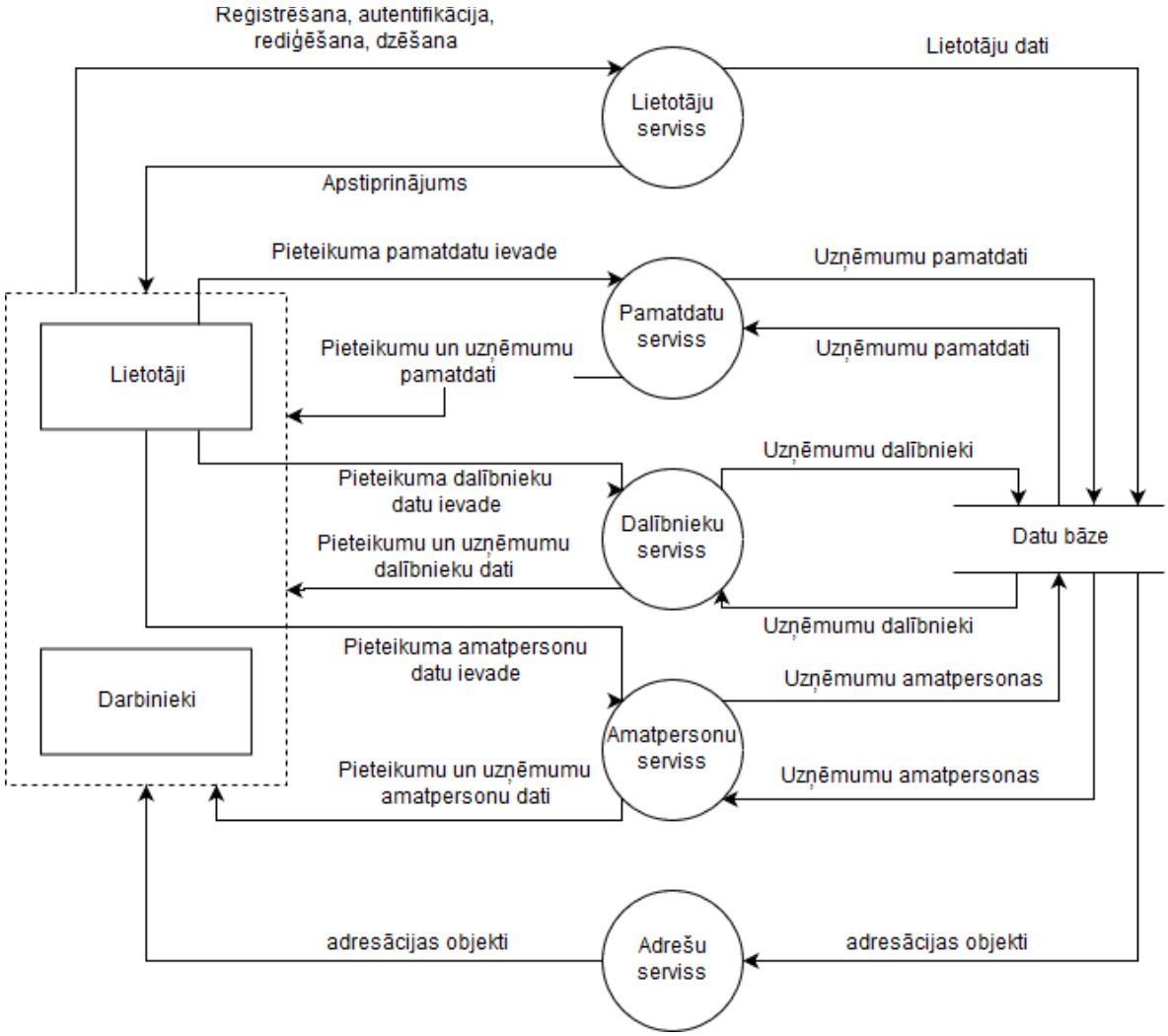
2.2.1. Datu plūsmu diagrammas

2.2.1.1. 0. līmenis



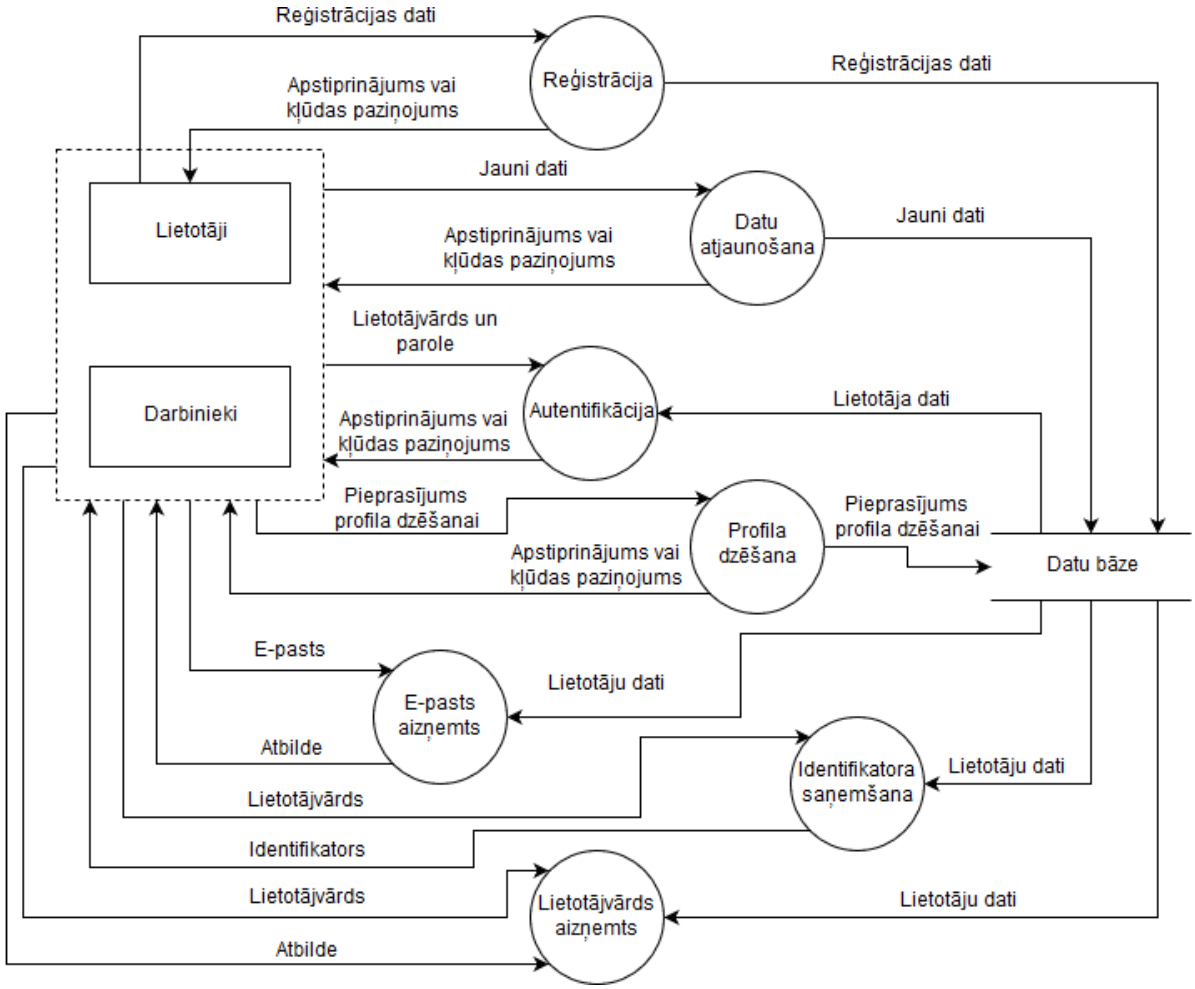
2.1. att. **Datu plūsmu diagramma 0. līmenis**

2.2.1.2. 1. līmenis



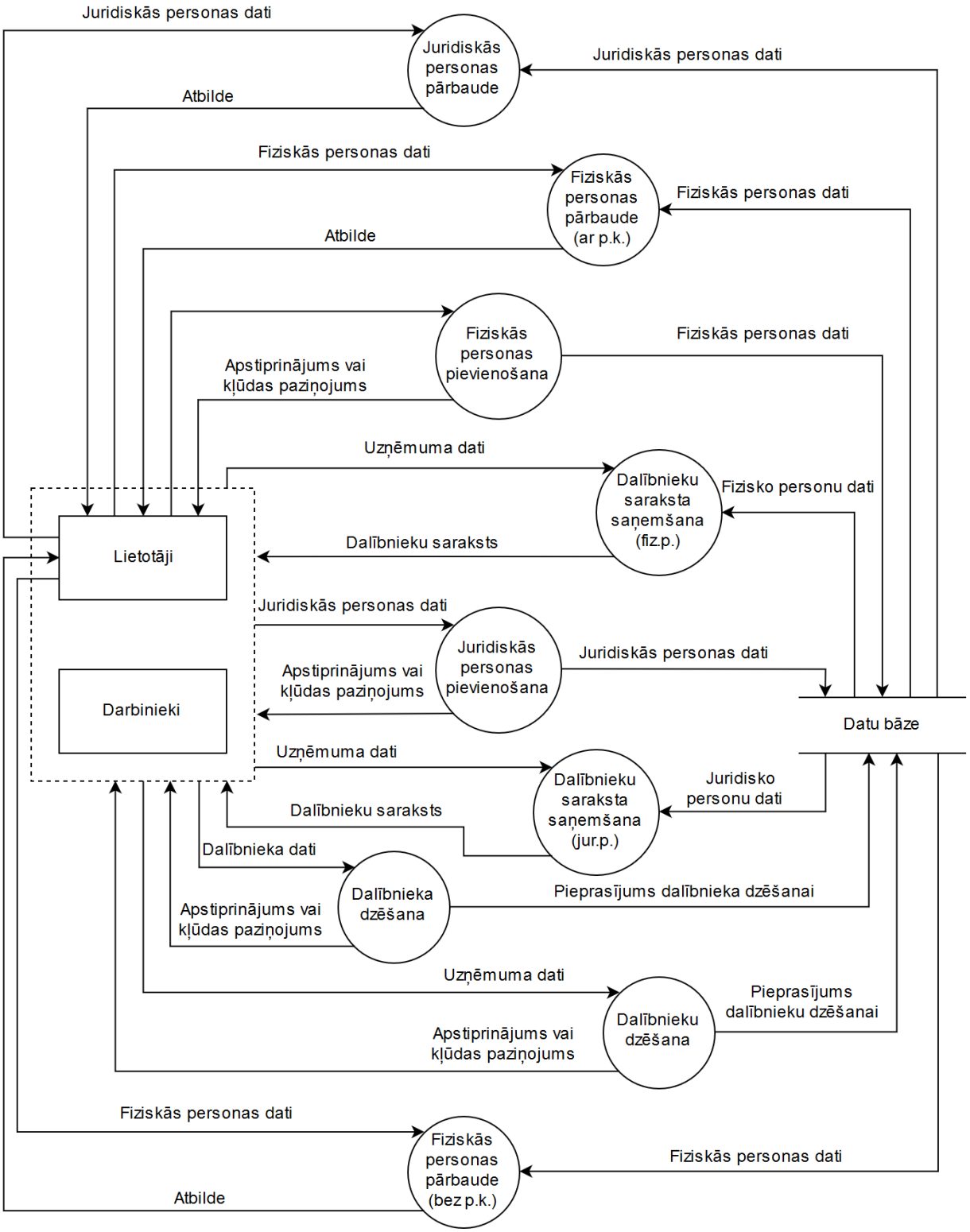
2.2. att. **Datu plūsmu diagramma 1. līmenis**

2.2.1.3. 2. līmenis. Lietotāju serviss



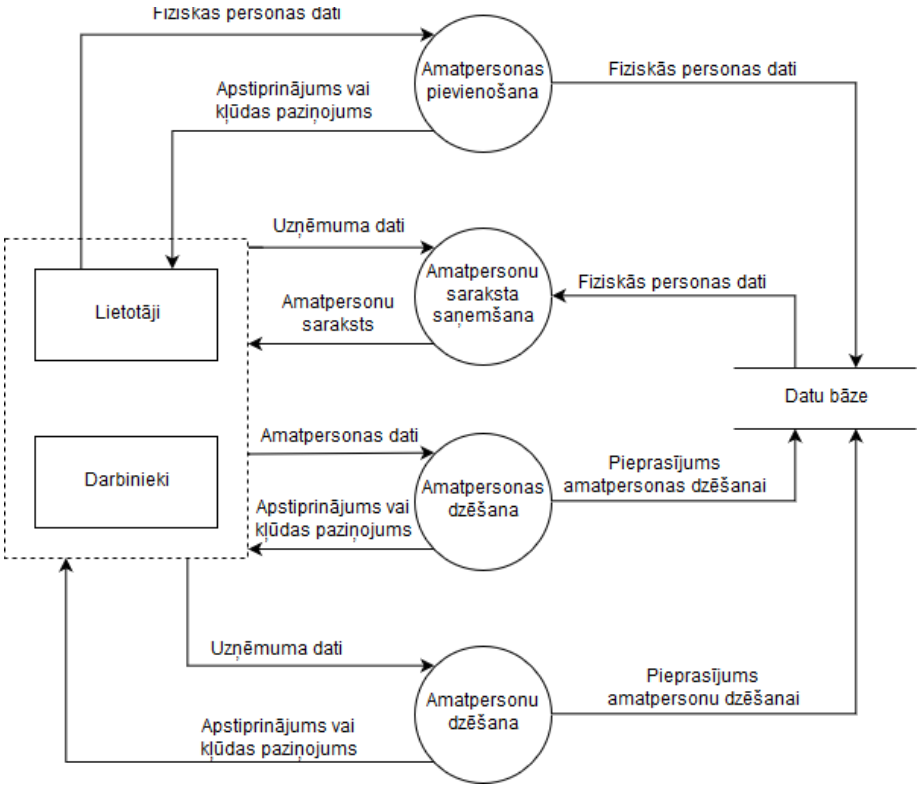
2.3. att. **Datu plūsmu diagramma 2. līmenis. Lietotāju modulis**

2.2.1.4. 2. līmenis. Dalībnieku serviss



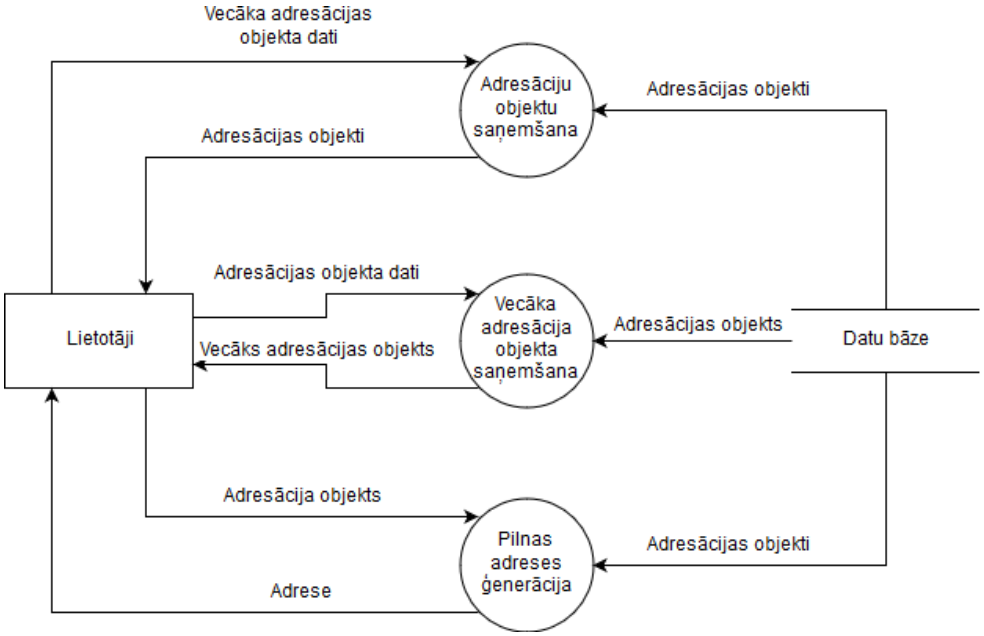
2.4. att. **Datu plūsmu diagramma 2. līmenis. Dalībnieku modulis**

2.2.1.5. 2. līmenis. Amatpersonu serviss



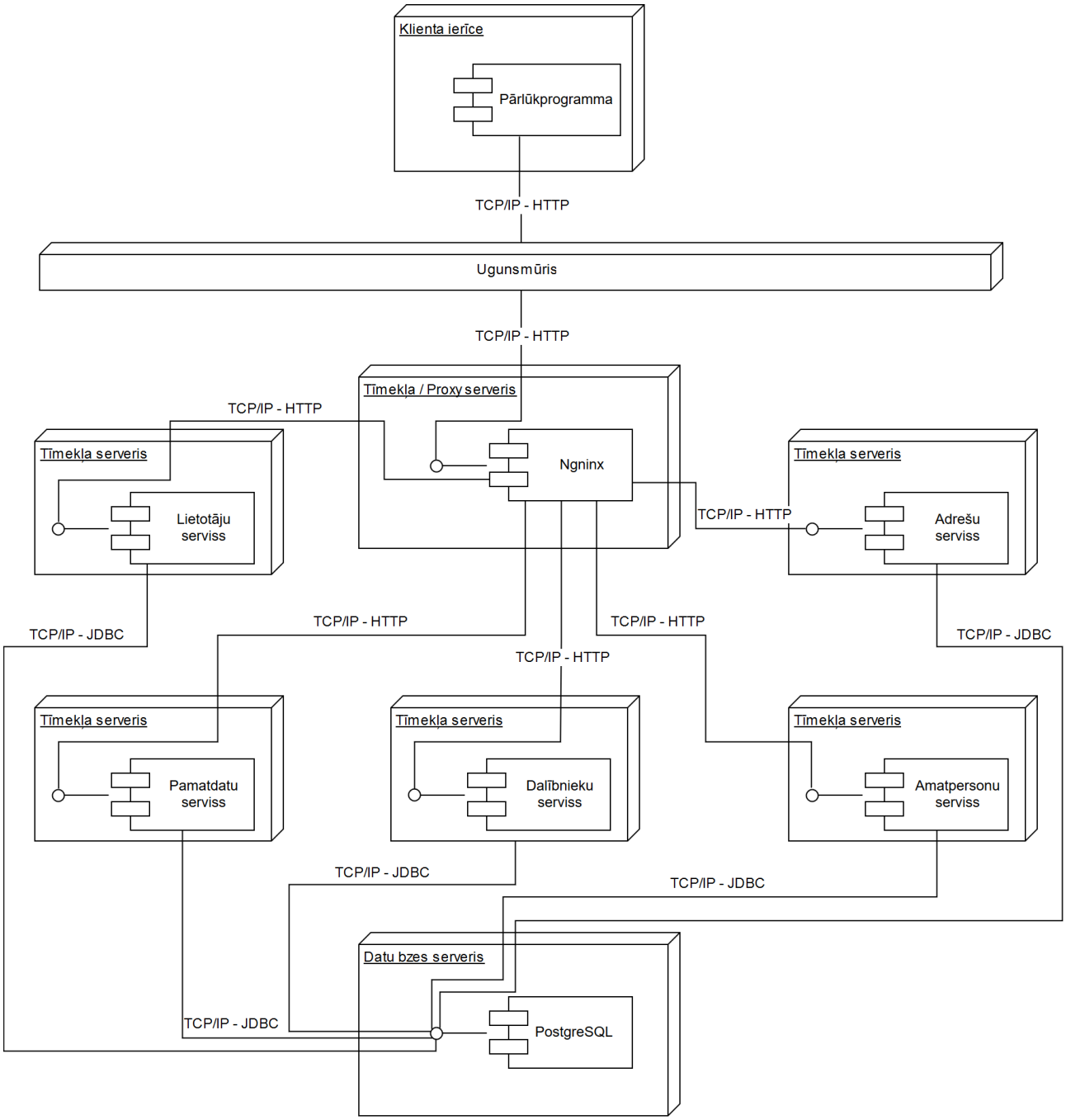
2.5. att. **Datu plūsmu diagramma 2. līmenis. Amatpersonu modulis**

2.2.1.6. 2. līmenis. Adrešu serviss



2.6. att. **Datu plūsmu diagramma 2. līmenis. Adrešu modulis**

2.2.2. Izvietošanas diagrammas

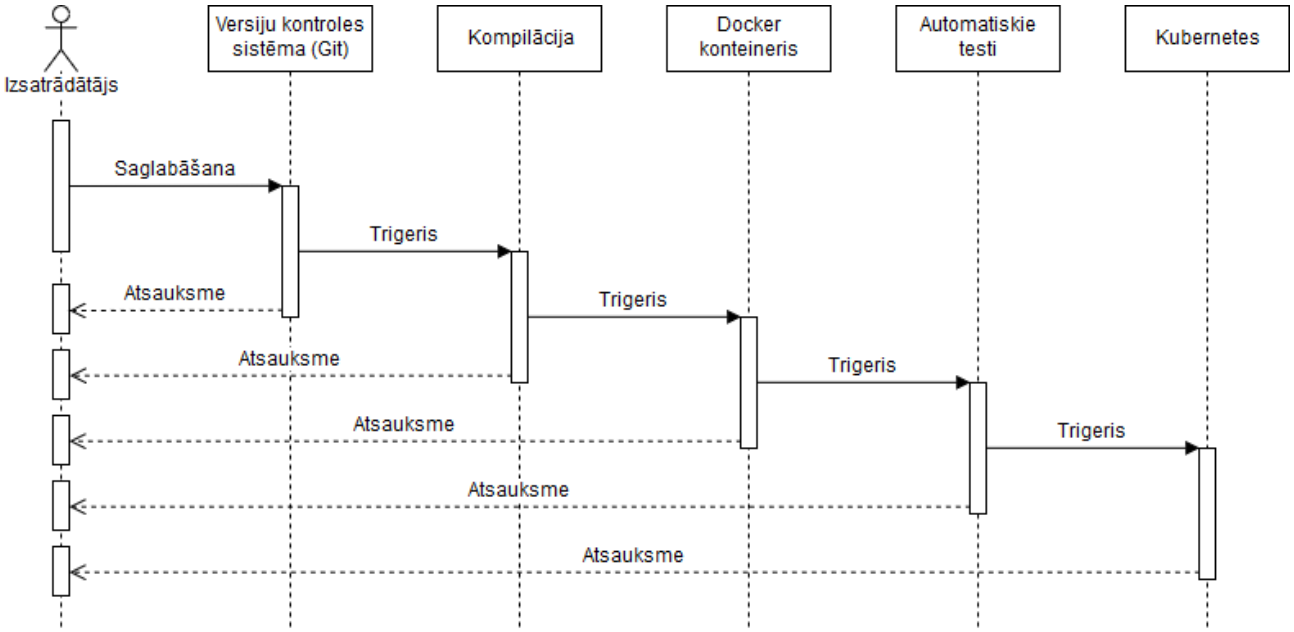


2.7. att. **Izvietošanas diagramma**

2.2.3. Secību diagrammas

2.2.3.1. Izvietošanas process

Pēc katras jaunākās versijas saglabāšanas *Git* versiju kontroles sistēmā notiek automātisko darbību virkne. Sākumā jaunākās projekta versijas pirmkods tiek kompilēts, kā rezultātā tiek izveidots projekta izpildāmais fails. Tālāk tiek izveidots *Docker* attēls, kurā ir šīs projekts. Šajā brīdī projekts ir gatavs uzstādīšanai, bet kvalitātes kontroles nepieciešamības dēļ, tiek palaisti automātiskie testi, kuri ietver kā vienībtestus, tā arī koda kvalitātes pārbaudes. Ja līdz šīm visi procesi notika bez kļūdām, projekta jaunākā versija tiek uzstādīta *Kubernetes* izstrādes vidē.

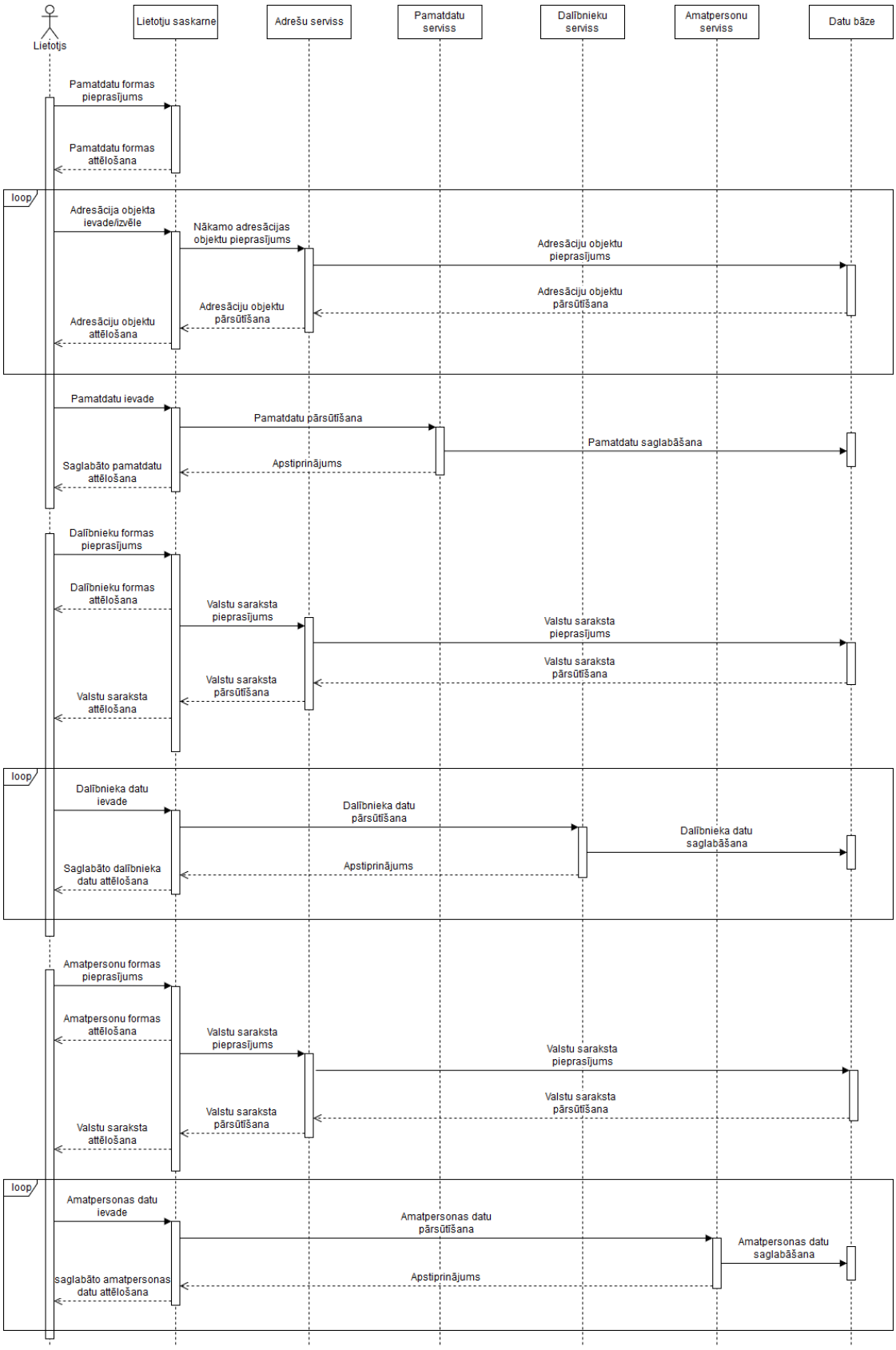


2.8. att. **Secību diagramma. Izvietošanas process**

2.2.3.2. Uzņēmuma reģistrācijas pieteikuma izveidošanas process

Šajā diagrammā tiek attēlots uzņēmuma reģistrācijas process, kurā ir arī neiekļautās šajā darbā daļas – lietotāju saskarne un pamatdatu serviss.

Process sākas ar uzņēmuma pamatdatu saglabāšanu, kurā ir iesaistīts adrešu serviss, kas nodrošina korektu uzņēmuma adrese ievadi. Tālāk notiek vairāku uzņēmumu dalībnieku saglabāšana, kurā arī ir iesaistīts adrešu serviss, bet šoreiz tas nodrošina adrešu klasifikatora darbību. Pēdējais solis ir uzņēmuma amatpersonu ievade, kas pēc darbības principa ir līdzīga dalībnieku ievadei.



2.9. att. **Secību diagramma. Uzņēmuma reģistrācijas pieteikuma izveidošanas process**

2.3. Datu bāzes projektējums

Sistēmā tiek izmantota datu bāze, kuras pamatā ir PostgreSQL objektu relāciju datu bāzes pārvaldības sistēma. Tā kā sistēmu nākotnē ir plānots attīstīt, datu bāze papildus nepieciešamiem laukiem ir lauki, kuri šajā sistēmas versijā netiek izmantoti, bet būs nepieciešami nākotnē. Tas neaizskar tekošo datu bāzes struktūru un kopējo sistēmas funkcionalitāti, bet dod iespēju izstrādāt sistēmu, ņemot vērā perspektīvā nepieciešamos laukus. Datu bāze sastāv no 10 tabulām, no kurām 8 ir sistēmas datu glabāšanai un 2 ir klasifikatoru tabulas.

2.3.1. Datu dekompozīcija

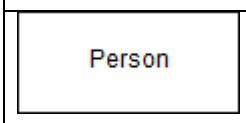
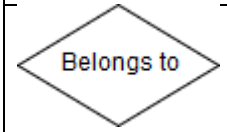
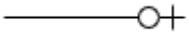
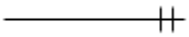
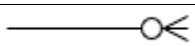
2.1. tabula
Datu dekompozīcija

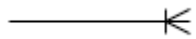
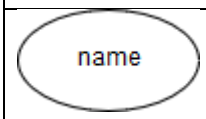
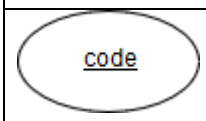
Tabulas nosaukums	Loģiskais nosaukums
Cla_address	Adrešu klasifikators
Cla_country	Valstu klasifikators
Foreign_person	Ārvalsts persona
Legal_entity	Juridiska persona
Lventity	Latvijā reģistrēta juridiska persona
Member	Uzņēmuma dalībnieki
Officer	Uzņēmuma amatpersonas
Person	Personas
Person_history	Personas datu vēsture
Registered_person	Reģistrētas personas

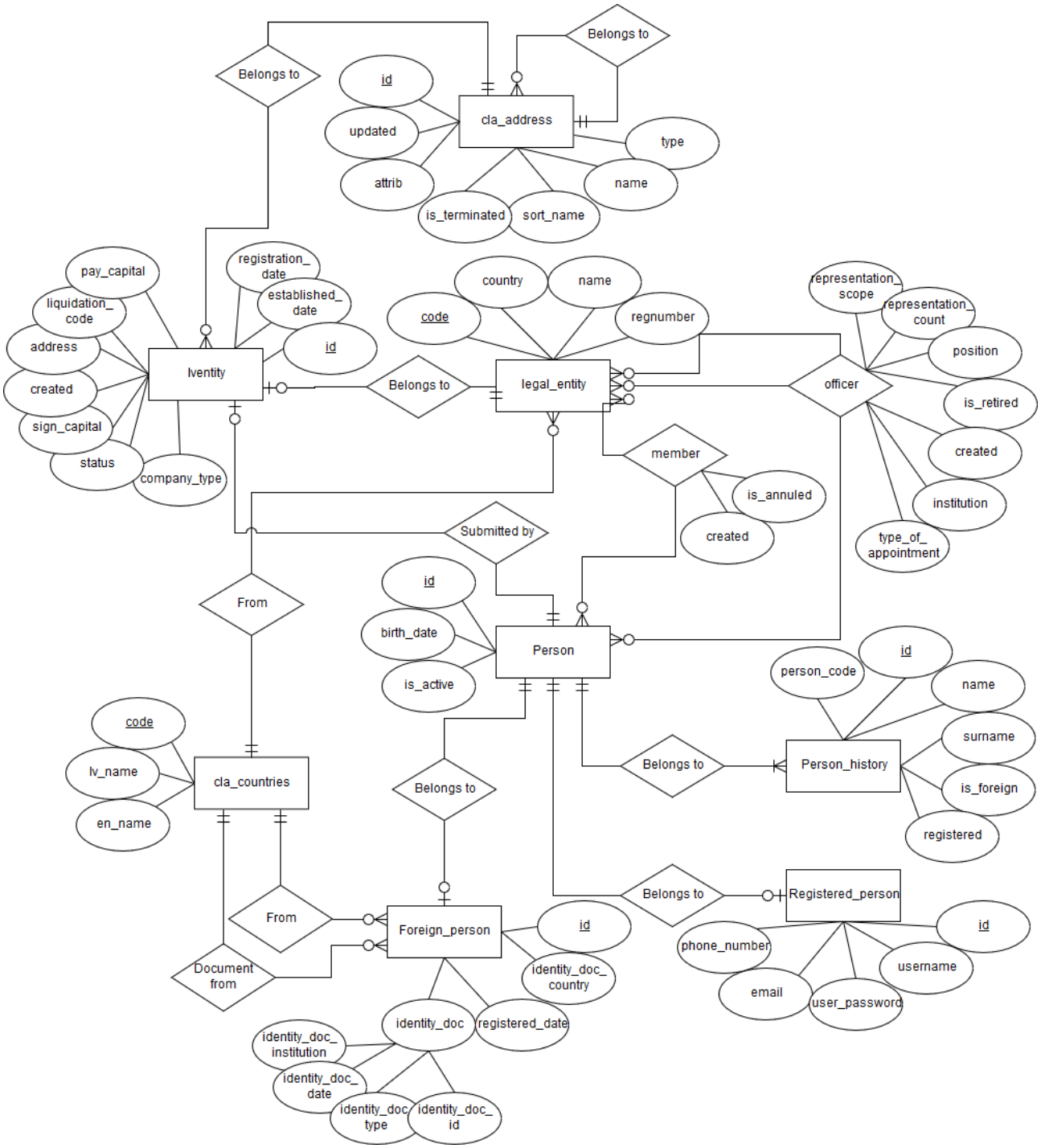
2.3.2. Konceptuālais ER modelis

Konceptuālā ER modeļa apzīmējumi:

2.2. tabula
Konceptuālā ER modeļa apzīmējumi

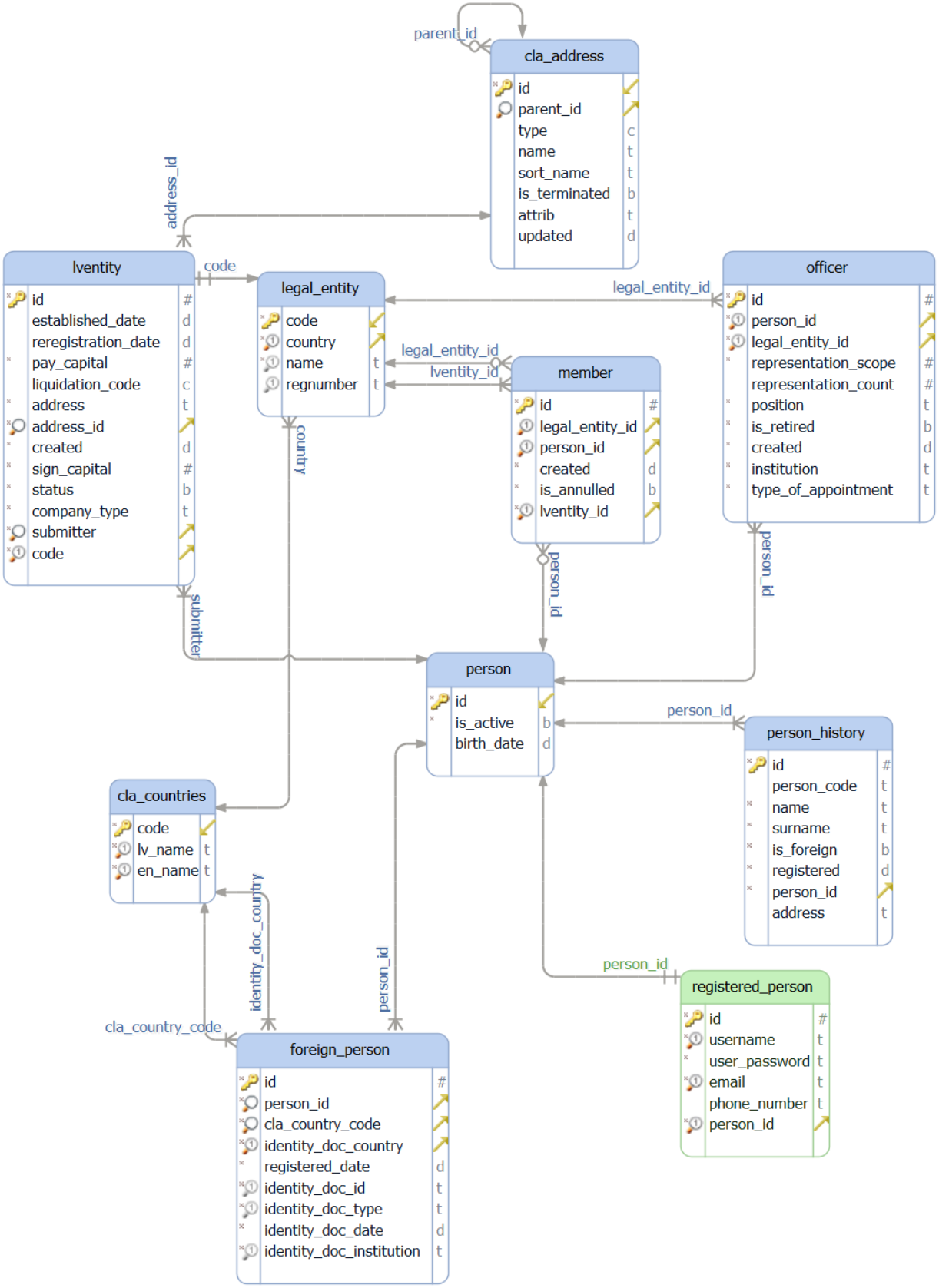
Apzīmējums	Paskaidrojums
	Datu bāzes entītija
	Attiecība starp datu bāzes entītijām
	Kardinalitāte nulle vai viens
	Kardinalitāte viens (un tikai viens)
	Kardinalitāte nulle vai daudz

	Kardinalitāte viens vai daudz
	Entītijas atribūts
	Entītijas atribūts (primārā atslēga)



2.10. att. **Konceptuālais ER modelis**

2.3.3. Fiziskais ER modelis



2.11. att. **Fiziskais ER modelis**

2.3.4. Datu bāzes tabulu projektējums

Datu bāzes tabulu aprakstā katra no tabulām ir aprakstīta, izmantojot šādu struktūru:

- Lauka nosaukums
- Datu tips
- NULL (Vai lauks ir neobligāts, '+' ja ir, '-' ja nav)
- Apraksts
- Piezīmes

Saīsinājumi:

- PK – Primary Key (primārā atslēga)
- FK – Foreign Key (ārējā atslēga)
- AI – Auto Increment (automātiski pieaugoša skaitliska vērtība)

Dažām tabulām, izmantojot PostgreSQL iebūvētas funkcijas, ir pielietoti ierobežojumi, kuri ir aprakstīti pēc tabulas apraksta.

2.3.4.1. Tabula "Person"

Tabulā tiek glabāta pamatinformācija par personām, kura nevar būt mainīta.

2.3. tabula
Datu bāzes tabulas "Person" projektējums

Lauka nosaukums	Datu tips	NULL	Apraksts	Piezīmes
id	integer	-	Personas identifikators	PK, AI
is_active	boolean	-	Lauks, kas identificē personas uzņēmējdarbības aktivitāti	Netiek izmantots tekošajā sistēmas versijā
birth_date	date	+	Personas dzimšanas datums	

2.3.4.2. Tabula "Person_history"

Tabula satur personas datus, kuri var mainīties, saglabājot iepriekšējos personas datus.

2.4. tabula
Datu bāzes tabulas "Person_history" projektējums

Lauka nosaukums	Datu tips	NULL	Apraksts	Piezīmes
id	Integer	-	Personas vēstures ieraksta identifikators	PK, AI
person_code	text	-	Personas kods	
name	char varying (80)	-	Vārds	
surname	char varying (80)	-	Uzvārds	

is_foreign	boolean	-	Vai cilvēks ir ārzemnieks	
registered	timestamp	-	Ieraksta izveidošanas datums un laiks	
person_id	integer	-	Personas identifikators	FK
address	char varying (100)	+	Adrese	Netiek izmantots tekošajā sistēmas versijā

Ierobežojumi:

- is_foreign = true OR person_code IS NOT NULL (vai nu persona ir ārzemnieks, vai nu personas kods ir obligāts)

2.3.4.3. Tabula “Foreign_person”

Tabula satur ārzemes personas datus.

2.5. tabula

Datu bāzes tabulas “Foreign_person” projektējums

Lauka nosaukums	Datu tips	NULL	Apraksts	Piezīmes
id	Integer	-	Ārzemes personas identifikators	PK, AI
person_id	integer	-	Personas identifikators	FK
Cla_country_code	char varying (2)	-	Dzīvesvietas valsts	FK
identity_doc_country	char varying (2)	-	Personas apliecinošā dokumenta valsts	FK
registered_date	timestamp	-	Ieraksta izveidošanas datums un laiks	
identity_doc_id	char varying (64)	-	Personas apliecinošā dokumenta numurs	
identity_doc_type	char varying (128)	-	Personas apliecinošā dokumenta tips	
identity_doc_date	date	-	Personas apliecinošā dokumenta izdošanas datums	
identity_doc_institution	char varying (128)	-	Personas apliecinošā dokumenta izdošanas institūcija	

Ierobežojumi:

- UNIQUE person_id (person_id ir unikāls)
- UNIQUE identity_doc_id, identity_doc_country, identity_doc_type, identity_doc_institution (Personas apliecinošā dokumenta dati ir unikāli)

2.3.4.4. Tabula “Registered_person”

Tabula satur datus par reģistrēto personu profiliem.

2.6. tabula

Datu bāzes tabulas “Registered_person” projektējums

Lauka nosaukums	Datu tips	NULL	Apraksts	Piezīmes
id	integer	-	Reģistrētās personas identifikators	PK, AI
username	char varying (16)	-	Lietotājvārds	
user_password	char varying (255)	-	Parole	Parole tiek glabāta šifrētā veidā
email	char varying (255)	-	E-pasts	
phone_number	char varying (32)	+	Telefona numurs	Netiek izmantots tekošajā sistēmas versijā
person_id	integer	-	Personas identifikators	FK

Ierobežojumi:

- UNIQUE email (email ir unikāls)
- UNIQUE person_id (person_id ir unikāls)
- UNIQUE username (username ir unikāls)
-

2.3.4.5. Tabula “Legal_entity”

Tabula satur pamatinformāciju par juridiskām personām. Reģistrācijas numura lauks ir neobligāts, jo uzņēmuma reģistrācijas pieteikumi tiek glabāti kā reāli uzņēmumi, bet tiem vēl nav piešķirts reģistrācijas numurs.

2.7. tabula

Datu bāzes tabulas “Legal_entity” projektējums

Lauka nosaukums	Datu tips	NULL	Apraksts	Piezīmes
code	integer	-	Juridiskās personas identifikators	PK, AI
country	char varying (2)	-	Valsts, kurā ir reģistrēta jur. persona	FK
name	char varying (100)	-	Jur. personas vārds (nosaukums)	

regnumber	char varying (32)	+	Reģistrācijas numurs	

Ierobežojumi:

- UNIQUE country, name (vienā valstī nevar būt vairāki uzņēmumi ar vienādu nosaukumu)
- UNIQUE country, regnumber (vienā valstī nevar būt vairāki uzņēmumi ar vienādu reģistrācijas numuru)

2.3.4.6. Tabula "Lventity"

Tabula satur papildinformāciju par Latvijā reģistrētām juridiskām personām.

2.8. tabula

Datu bāzes tabulas "Lventity" projektējums

Lauka nosaukums	Datu tips	NULL	Apraksts	Piezīmes
id	integer	-	Latvijā reģistrētās jur. personas identifikators	PK, AI
established_date	date	+	Uzņēmuma reģistrācijas pieteikuma datums	
registration_date	date	+	Uzņēmuma reģistrācijas datums	
pay_capital	numeric (15, 2)	-	Apmaksātais kapitāls	
liquidation_code	character (1)	+	Uzņēmuma likvidācijas kods	Netiek izmantots tekošajā sistēmas versijā
address	char varying (255)	-	Adrese teksta formātā	
address_id	integer	-	Adreses identifikators	FK
created	date	-	Ieraksta izveidošanas datums	
sign_capital	numeric (15, 2)	-	Parakstītais kapitāls	
status	boolean	-	Uzņēmuma statuss	Reģistrēts vai uzņēmuma reģistrācijas pieteikums
company_type	char varying (3)	-	Uzņēmuma tips	"SIA", "AS" utt.

submitter	integer	-	Pieteikuma iesniedzēja identifikators	FK
code	integer	-	Juridiskās personas identifikators	FK

Ierobežojumi

- UNIQUE code (code ir unikāls)

2.3.4.7. Tabula “Member”

Tabula satur informāciju par uzņēmuma dalībniekiem. Par dalībnieku var būt gan fiziskā persona, gan juridiskā.

2.9. tabula

Datu bāzes tabulas “Member” projektējums

Lauka nosaukums	Datu tips	NULL	Apraksts	Piezīmes
id	integer	-	Dalībnieka identifikators	PK, AI
legal_entity_id	integer	+	Juridiskās personas identifikators, kura ir dalībnieks	FK
person_id	integer	+	Fiziskās personas identifikators, kura ir dalībnieks	FK
created	date	-	Ieraksta izveidošanas datums	
is_annuled	boolean	-	Vai dalībnieks ir anulēts	Netiek izmantots tekošajā sistēmas versijā
lventity_id	integer	-	Juridiskās personas identifikators, kurai ir pievienots dalībnieks	FK

Ierobežojumi:

- legal_entity_id IS NOT NULL AND person_id IS NULL OR person_id IS NOT NULL AND legal_entity_id IS NULL (dalībnieks var būt vai nu fiziska persona, vai juridiska persona – tikai viena no tām)
- UNIQUE person_id, lventity_id (viena un tā pati fiziska persona nevar būt reģistrēta kā dalībnieks vienam uzņēmumam vairākas reizes)
- UNIQUE legal_enitty_id, lventity_id (viena un tā pati juridiska persona nevar būt reģistrēta kā dalībnieks vienam uzņēmumam vairākas reizes)

2.3.4.8. Tabula “Officer”

Tabula satur informāciju par uzņēmuma amatpersonām. Par amatpersonu var būt tikai fiziska persona.

2.10. tabula

Datu bāzes tabulas “Officer” projektējums

Lauka nosaukums	Datu tips	NULL	Apraksts	Piezīmes
id	integer	-	Amatpersonas identifikators	PK, AI
person_id	integer	-	Fiziskās personas identifikators, kura ir amatpersona	FK
legal_entity_id	integer	-	Juridiskās personas identifikators, kurai ir pievienota amatpersona	FK
representation_scope	integer	-	Pārstāvēšanas tiesības	1 – tikai atsevišķi; 2- kopīgi ar visiem; 3 – kopīgi ar priekšsēdētāju; 4 – kopīgi ar vismaz __ valdes locekļiem
representation_count	integer	+	Ar cik valdes locekļiem ir iespējams pārstāvēt sabiedrību	
position	char varying (50)	-	Amats	
is_retired	boolean	-	Vai amatpersona ir atvaļināta	
created	date	-	Ieraksta izveidošanas datums	
institution	char varying (64)	-	Institūcija	
type_of_appointment	char varying (64)	-	Iecelšanas veids	

Ierobežojumi:

- UNIQUE person_id, legal_entity_id (viena un tā pati fiziska persona nevar būt reģistrēta kā amatpersona vienam uzņēmumam vairākas reizes)

- representation_scope <> 4 AND representation_count IS NULL OR representation_scope = 4 AND representation_count IS NOT NULL (representation_count jābūt aizpildītam tikai, ja representation_scope = 4)

2.3.4.9. Tabula “Cla_countries”

Tabula satur valstu klasifikatora datus.

2.11. tabula

Datu bāzes tabulas “Cla_countries” projektējums

Lauka nosaukums	**Datu tips**	**NULL**	**Apraksts**	**Piezīmes**
code	char varying (2)	-	Valsts kods pēc “ISO 3166-1 alfa-2” standarta	PK
lv_name	char varying (64)	-	Valsts nosaukums latviešu valodā	
en_name	char varying (64)	-	Valsts nosaukums angļu valodā	

Ierobežojumi:

- UNIQUE lv_name (lv_name ir unikāls)
- UNIQUE en_name (en_name ir unikāls)
-

2.3.4.10. Tabula “Cla_address”

Tabula satur adrešu klasifikatora datus. Tabulā ir rekursīva ārējā atslēga, kas nodrošina visu adresācija objektu glabāšanu vienā tabulā.

2.12. tabula

Datu bāzes tabulas “Cla_address” projektējums

Lauka nosaukums	**Datu tips**	**NULL**	**Apraksts**	**Piezīmes**
id	integer	-	Adresācijas objekta identifikators	PK
parent_id	integer	-	Vecāka adresācijas objekta identifikators	FK
type	char varying (3)	-	tips	Māja, iela, pilsēta utt., izmantojot tipa identifikatoru
name	char varying (200)	-	Nosaukums	
sort_name	char varying (200)	-	Pilns nosaukums	

is_terminated	boolean	-	Vai objekts ir novecojis	
attrib	char varying (32)	+	Papildus informācija	
updated	timestamp	+	Pēdējais objekta atjaunošanas datums un laiks	

2.4. Saskarnes apraksts

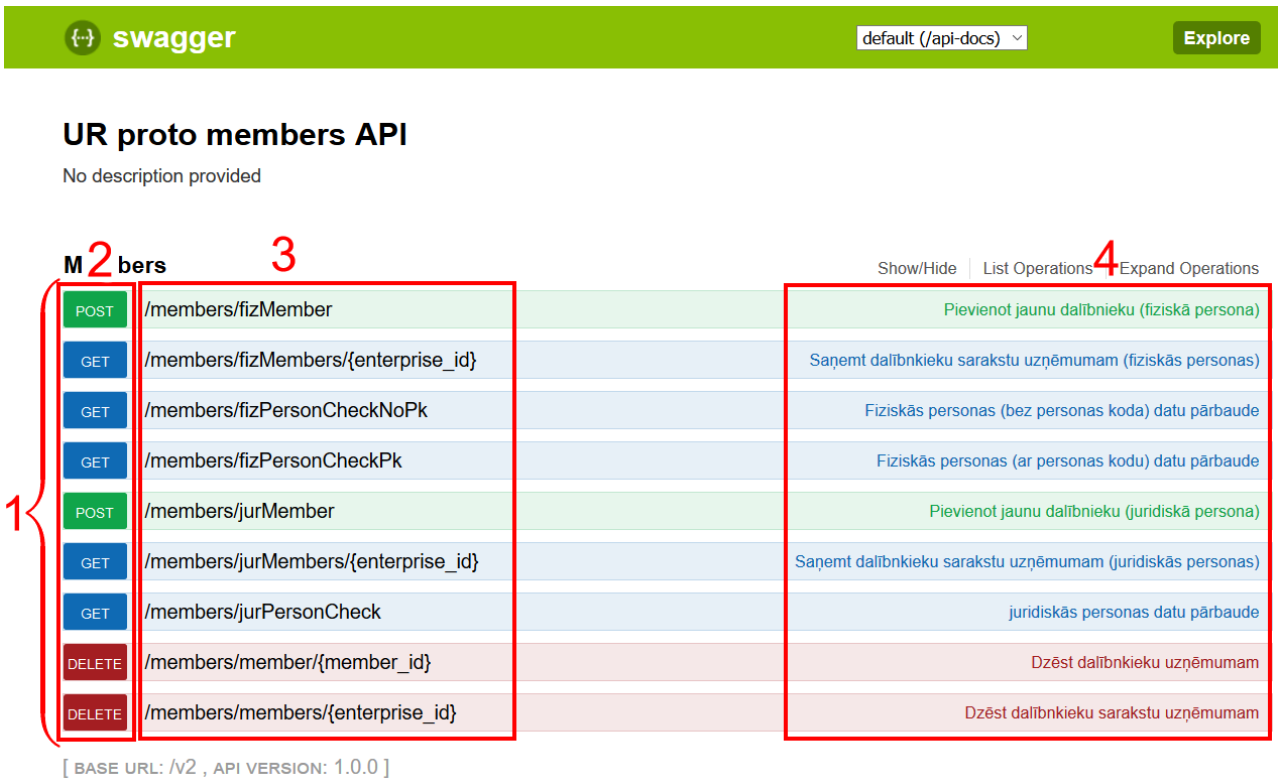
Gala lietotāja darbam ar sistēmu ir paredzēta tīmekļa lietotne, kas realizēta kā atsevišķs mikroserviss. Tajā notiek sistēmas un lietotāju mijiedarbība, bet tas attiecas uz klienta puses izstrādi, kas nav šī darba daļa. Šajā nodaļā ir aprakstīta servera puses servisu automātiski ģenerētā lietotāja saskarne. Tā ir realizēta, izmantojot Swagger rīku, kas nodrošina grafisko saskarni servisiem. Šajā nodaļā tiks aprakstīta Swagger saskarne, parādīti un paskaidroti rīka elementi un iespējas. Tā kā šīs rīks ir izmantots visos tīmekļa servisos un atšķirības tajos ir tikai saturiskas, aprakstīts būs tikai viens no tiem, bet pārējie ir veidoti pēc tāda paša principa.

Swagger ir domāts izstrādātājiem, lai vieglāk orientēties tīmekļa servisu funkcijās, ātri saņemt informāciju par tām un nepieciešamības gadījumā manuāli palaist servisa funkcijas. Produkcijā piekļuve Swagger saskarnei tiek deaktivēts, izņemot publiski pieejamos servisus.

2.4.1. Galvenais skats

Galvenajā skatā ir iespējams redzēt:

1. sarakstu ar visām tīmekļa servisa funkcijām;
2. funkcijas izsaukuma metodi (POST, GET utt.);
3. izsaukuma adresi (URL);
4. funkcijas aprakstu.

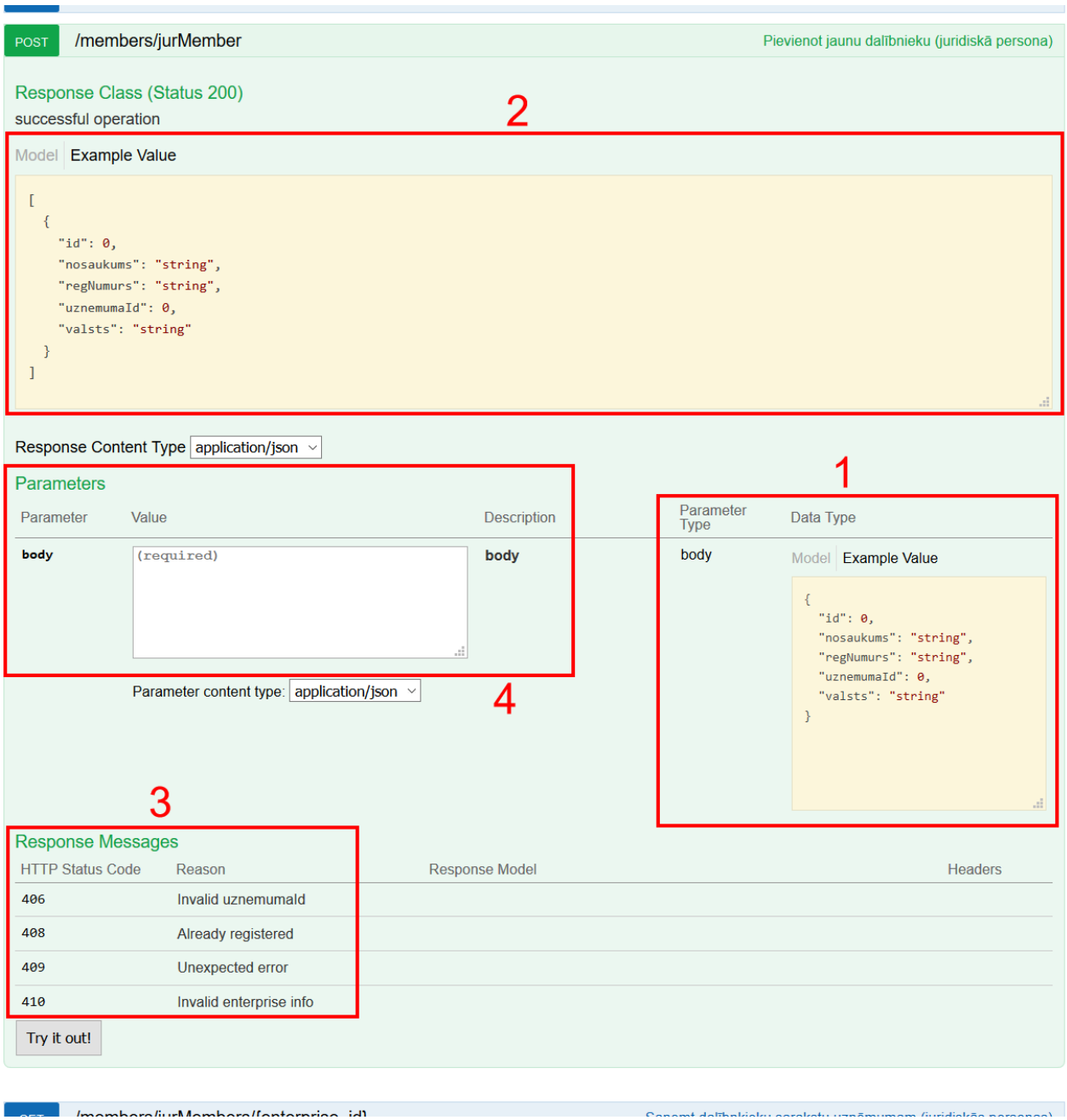


2.12. att. **Saskarne. Galvenais skats.**

2.4.2. Paplašinātās funkcijas skats

Nospiežot vienu no funkcijām, ir iespējams apskatīt papildinformāciju par šo funkciju:

1. Izvaddatu formātu;
2. Ievaddatu formātu;
3. Iespējamie atbildes HTTP statusa kodi un to paskaidrojumi;
4. Lauks, kurā funkcijas manuālas palaišanas gadījumā tiek ierakstīti ievaddati

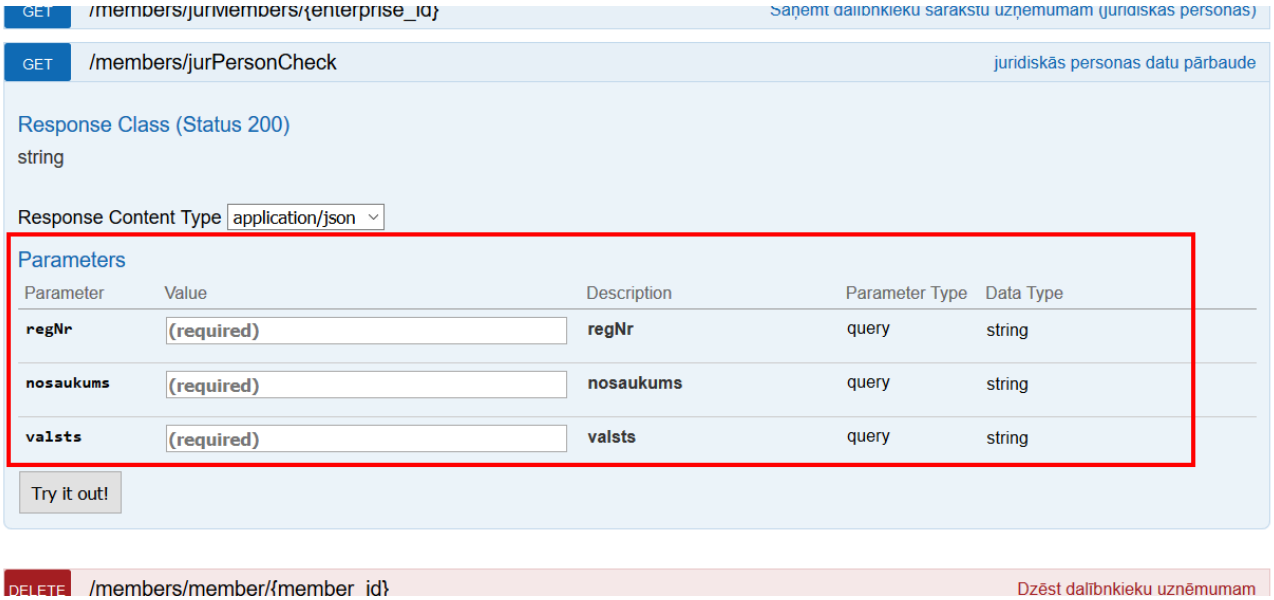


2.13. att. **Saskarne. Funkcijas papildinformācija.**

Tajā pašā skatā ir iespējams apskatīt ievaddatu tipus un kuri no mainīgajiem ir obligāti. Piemērā zemāk ir redzams, kā var izskatīties funkcija ar aizpildītu ievadlauku, kurā dati tiek nosūtīti HTTP pieprasījumā ķermenī (*body*)

2.14.att. **Saskarne. Datu nosūtīšana pieprasījuma ķermenī**

Piemērā zemāk ir redzams, kā izskatās funkcija, kurā dati tiek nosūtīti URL vaicājumā (*query*). Atšķirība ir tāda, ka katrai vērtībai ir savs ievadlauks.



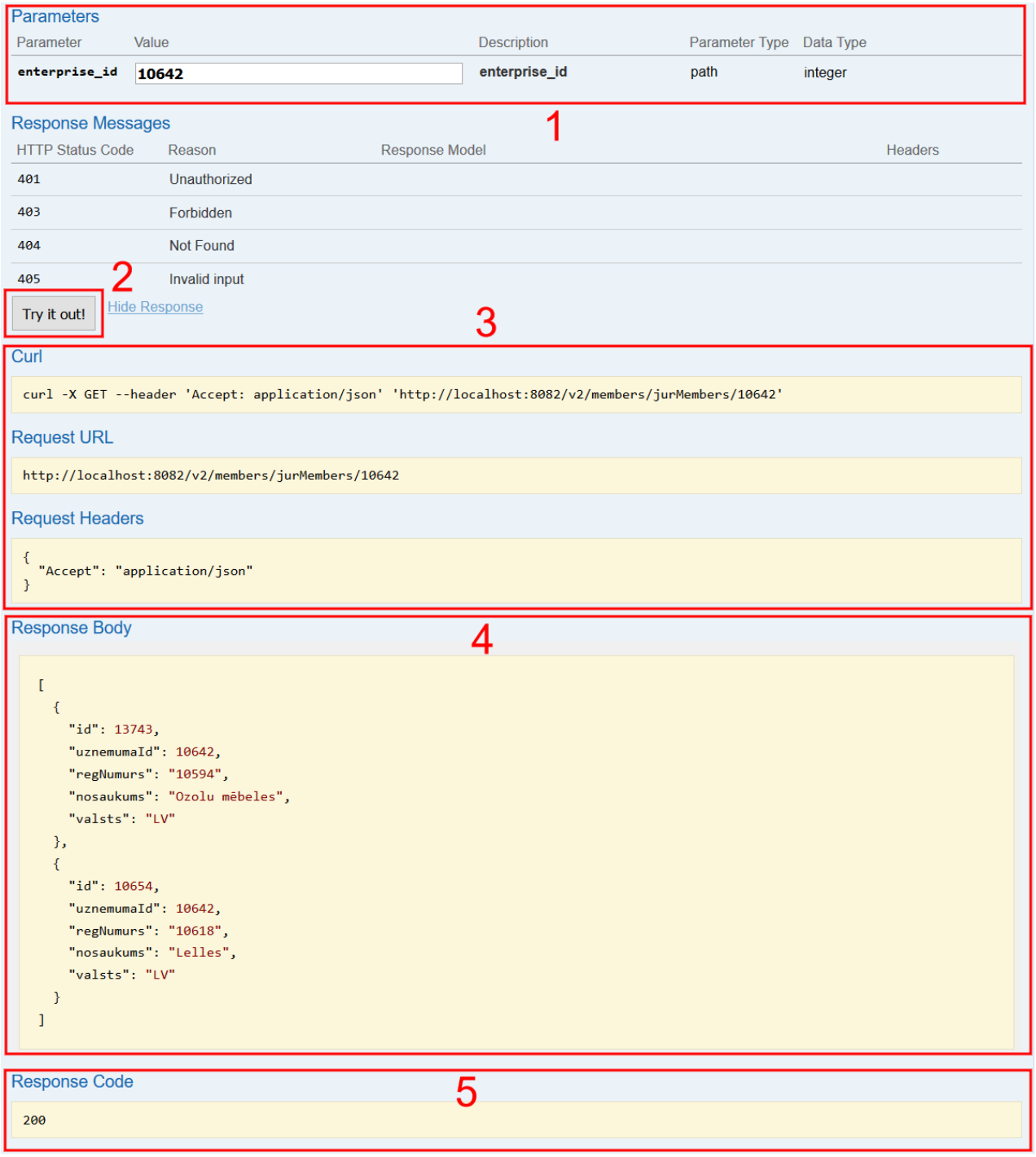
2.15. att. **Saskarne. Datu nosūtīšana pieprasījuma vaicājumā**

2.4.3. Skats funkcijas izsaukumam

Funkcijas izsaukums notiek tajā pašā skatā, kur var redzēt funkcijas paplašinātās iespējas, ievadot nepieciešamos datus un nospiežot attiecīgo pogu.

Skatā atrodas:

1. Ievaddati
2. Funkcijas izsaukuma poga
3. Funkcijas izsaukuma papildinformācija
4. Funkcijas izvaddati
5. HTTP atbildes kods



2.16. att. **Saskarne. Skats funkcijas izsaukumam**

3. TESTĒŠANAS DOKUMENTĀCIJA

3.1. Ievads

Šajā nodaļā tiek aprakstīti izstrādāto tīmekļa servisu testēšanas rezultāti. Testēšanā tika izmantota gan "baltās kastes" princips, kurā sistēma tiek testēta ņemot vērā programmatūras kodu, izmantojot vienībtestēšanu ar izstrādes vidē iebūvētām tehnoloģijām, gan "melnās kastes" princips, kurā tiek testēta tikai sistēmas funkcionalitāte, izmantojot papildus testēšanas rīku SoapUI.

3.2. Testējamās raksturiezīmes

Katrs tīmekļa serviss ir saskarne starp lietotāja pusi un datu bāzi, kura apstrādā un pārsūta datus no viena uz otru.

Katram servisam ir 3 testa veidi:

- vienībtesti, kuru rezultāta tiek pārbaudītas funkcijas, kuras tiek izsauktas no lietotāju puses;
- vienībtesti, kuru rezultātā tiek pārbaudītas sistēmas iekšējās funkcijas. Šīs funkcijas nodrošina darbu ar datu bāzi;
- SoapUI vienībtesti, kuri tiek izpildīti jau strādājošā servisā. Šo testu izpildes laikā notiek reālu pieprasījumu nosūtīšana un atbilžu saņemšana. Testi tiek apvienoti testpiemēros (*TestCase*), kuru rezultātā notiek kādas konkrētas situācijas simulācija. Testpiemēra testi ir saistīti viens ar otru un veido vienu testu virkni.

Visi testi ir automatizēti, bet dažiem testiem ir nepieciešami specifiski ievaddati, kuri var mainīties atkarībā no sistēmā tekošiem glabātajiem datiem (pārsvarā attiecas uz SoapUI testiem) tāpēc testu ievaddati ir parametrizēti.

Dažos gadījumos vairāki testi tiek apvienoti vienā, ja šajos testos tek pārbaudīti līdzīgi gadījumi.

3.3. Testpiemēru specifikācija

Veiktie testi tiek aprakstīti tabulas formā. Tabulas sastāv no testa nosaukuma, testa apraksta, sagaidāmā rezultāta, funkcionālās prasības numura (ja tiek testēta funkcionālā

prasība) un testpiemēra numura (SoapUI testu gadījumā). Visiem testiem tiek sagaidīts, ka tie izpildās korekti, atbilstoši sagaidāmām rezultātam.

3.3.1. Lietotāju serviss

3.1. tabula
Lietotāju servisa funkciju vienībtesti

Nosaukums	Apraksts	Sagaidāmais rezultāts	Funkcionālās prasības Nr.
E-pasts aizņemts	Notiek divi funkcijas izsaukumi. Pirmajā gadījumā tiek pārbaudīts aizņemts e-pasts, otrajā neaizņemts.	Abos gadījumos HTTP atbildes kods ir 200. Pirmajā gadījumā Būla vērtība ir patiesa, otrajā aplama.	1.3.1.4
Lietotāja autentifikācija	Notiek divi funkcija izsaukumi. Pirmajā gadījumā lietotājvārds un parole nav korekti, otrajā ir.	Pirmajā gadījumā HTTP atbildes kods ir 400, otrajā ir 200 un saņemtais lietotāja personas vārds un uzvārds sakrīt ar sistēmā glabāto.	1.3.1.2
Lietotāja reģistrācija 1	Notiek divi funkcijas izsaukumi. Pirmajā gadījumā tiek reģistrēts lietotājs ar aizņemtiem lietotājvārdu un e-pastu, otrajā ar brīviem. Lietotāja personas dati abos gadījumos ir korekti.	Pirmajā gadījumā HTTP atbildes kods ir 405, otrajā ir 201.	1.3.1.1
Lietotāja reģistrācija 2	Funkcija tiek izsaukta ar kļūdainiem ievaddatiem. Tiek reģistrēta persona, kura jau ir saglabāta datu bāzē, bet vārds un uzvārds nesakrīt ar glabāto.	HTTP atbildes kods 406.	1.3.1.1
Lietotāja reģistrācija 3	Funkcija tiek izsaukta ar korektiem datiem, bet personai jau ir piereģistrēts cits profils.	HTTP atbildes kods 407.	1.3.1.1
Lietotāja datu atjaunošana 1	Funkcija tiek izsaukta ar korektiem datiem, bet personai nav piereģistrēts profils.	HTTP atbildes kods 405.	1.3.1.3

Lietotāja datu atjaunošana 2	Notiek divi funkcijas izsaukumi. Pirmajā gadījumā jauns lietotājvārds jau ir aizņemts, otrajā e-pasts ir aizņemts.	Pirmajā gadījumā HTTP atbildes kods ir 406, otrajā ir 407.	1.3.1.3
Lietotāja datu atjaunošana 3	Funkcija tiek izsaukta ar korektiem ievaddatiem.	HTTP atbildes kods ir 201, saņemtais e-pasts un lietotājvārds sakrīt ar nosūtīto.	1.3.1.3
Lietotājvārds aizņemts	Notiek divi funkcijas izsaukumi. Pirmajā gadījumā tiek pārbaudīts aizņemts lietotājvārds, otrajā neaizņemts.	Abos gadījumos HTTP atbildes kods ir 200. Pirmajā gadījumā Būla vērtība ir patiesa, otrajā aplama.	1.3.1.5

3.2. tabula

Lietotāju servisa iekšējo funkciju vienībtesti

Nosaukums	Apraksts	Sagaidāmais rezultāts
Personas saglabāšana	Tiek izveidotas 3 personas. Pirmā satur vairākus personas datu ierakstus (*person history*) un ir personas kods, otrajai personas identifikācijai tiek izmantoti personas apliecinošā dokumenta dati, trešajai ir piereģistrēts profils un ir personas kods. (Šo personu dati tiek izmantoti nākamajos testos).	Saglabātie ieraksti sakrīt ar pieprasījumā aizsūtītiem.
Personas meklēšana pēc personas koda	Notiek personas meklēšana pēc personas koda. Tiek izmantots trešās personas personas kods.	Tiek atrasta persona. Personai ir profils un lietotājvārds sakrīt ar iepriekš saglabāto.
Personas meklēšana pēc personas apliecinoša dokumenta datiem	Notiek personas meklēšana pēc otrās personas dokumenta datiem.	Tiek atrasta persona. Personas dzīvesvietas valsts sakrīt ar iepriekš saglabāto.
Personas profila meklēšana pēc lietotājvārda	Notiek personas profila meklēšana pēc trešās personas lietotājvārda.	Tiek atrasts lietotāja profils. Profilam ir piesaistīta persona un profila e-pasts sakrīt ar iepriekš saglabāto.
Personas meklēšana pēc lietotājvārda	Notiek personas meklēšana pēc trešās personas lietotājvārda.	Tiek atrasta persona, kurai ir reģistrēts profils un profila e-pasts sakrīt ar iepriekš saglabāto.

Personas meklēšana pēc e-pasta	Notiek personas meklēšana pēc trešās personas e-pasta.	Tiek atrasta persona, kurai ir reģistrēts profils un profila lietotājvārds sakrīt ar iepriekš saglabāto.
Jaunākā personas datu ieraksta meklēšana pēc lietotāja identifikatora	Notiek jaunākā personas datu ieraksta meklēšana, izmantojot pirmās personas identifikatoru (tas tiek atrasts ar citas funkcijas palīdzību).	Tiek atrasts personas datu ieraksts, kura vārds un uzvārds sakrīt ar pēdējo saglabāto.
Valsts meklēšana pēc valsts koda	Notiek valsts meklēšana pēc "LV" koda.	Tiek atrasta valsts. Valsts nosaukums latviešu valodā ir "Latvija", bet angļu valodā ir "Latvia"
Personas profila dzēšana	Notiek personas meklēšana pēc lietotājvārda. Tad profils tiek dzēsta. Notiek šīs pašas personas meklēšana pēc lietotājvārds.	Pirmajā meklēšanas izsaukumā tiek atrasta persona. Otrajā meklēšanas izsaukumā persona vairs atrasta netiek.

3.3. tabula

Lietotāju servisa SoapUI vienībtesti

Nosaukums	Apraksts	Sagaidāmais rezultāts	Funkcionālās prasības Nr.	Testpiemēra numurs
Lietotājvārds aizņemts	Tiek pārbaudīts vai vēlamais lietotājvārds ir brīvs.	Lietotājvārds ir brīvs.	1.3.1.5	1
E-pasts aizņemts	Tiek pārbaudīts vai vēlamais e-pasts ir brīvs.	E-pasts ir brīvs.	1.3.1.4	1
Lietotāja reģistrācija	Tiek reģistrēts lietotājs ar korektiem datiem un brīvu lietotājvārdu un paroli.	HTTP atbildes kods ir 201.	1.3.1.1	1
Lietotāja identifikatora saņemšana	Tiek izsaukta funkcija lietotāja identifikatora saņemšanai ar tikko piereģistrētā lietotāja lietotājvārdu.	Funkcija atgrieza identifikatoru.	1.3.1.6	1
Lietotājvārds aizņemts	Tiek pārbaudīts vai tikko piereģistrētā lietotāja lietotājvārds ir aizņemts	Lietotājvārds ir aizņemts.	1.3.1.5	1
E-pasts aizņemts	Tiek pārbaudīts vai tikko piereģistrētā	E-pasts ir aizņemts.	1.3.1.4	1

	lietotāja lietotājvārds ir aizņemts.			
Lietotājvārds aizņemts	Tiek pārbaudīts vai cits lietotājvārds ir brīvs.	Lietotājvārds ir brīvs.	1.3.1.5	1
Lietotāja datu atjaunošana	Lietotāja profila dati tiek atjaunoti. Lietotājam tiek piešķirts jauns lietotājvārds, kas bija pārbaudīts iepriekšējā testā.	HTTP atbildes kods ir 201.	1.3.1.3	1
Lietotāja autentifikācija	Notiek lietotāja autentifikācija ar piereģistrētā lietotāja atjaunoto lietotājvārdu un paroli.	HTTP atbildes kods ir 200. Funkcijas atgrieztie personas dati atbilst reģistrētā lietotāja datiem.	1.3.1.2	1
Lietotāja profila dzēšana	Notiek reģistrētā lietotāja profila dzēšana.	HTTP atbildes kods ir 200.	1.3.1.7	1
Lietotāja autentifikācija	Notiek lietotāja autentifikācija ar dzēstā lietotāja lietotājvārdu un paroli.	HTTP atbildes kods ir 400.	1.3.1.2	1

3.3.2. Uzņēmumu dalībnieku serviss

3.4. tabula

Uzņēmumu dalībnieku servisa funkciju vienībtesti

Nosaukums	Apraksts	Sagaidāmais rezultāts	Funkcionālās prasības Nr.
Fiziskās personas pievienošana 1	Notiek divi funkcijas izsaukumi. Pirmajā gadījumā tiek pievienota persona, kura jau ir reģistrēta šīm uzņēmumam kā dalībnieks, otrajā personas dati nesakrīt ar sistēmā glabātajiem.	Pirmajā gadījumā HTTP atbildes kods ir 408, otrajā ir 410.	1.3.2.1

Fiziskās personas pievienošana 2	Notiek personas ar personas kodu pievienošana, kuras dati jau ir glabāti sistēmā.	HTTP atbildes kods ir 200.	1.3.2.1
Fiziskās personas pievienošana 3	Notiek personas ar personas kodu pievienošana, kuras dati vēl nav glabāti sistēmā.	HTTP atbildes kods ir 200.	1.3.2.1
Fiziskās personas pievienošana 4	Notiek personas ar personas identificējošo dokumentu pievienošana, kuras dati vēl nav glabāti sistēmā.	HTTP atbildes kods ir 200.	1.3.2.1
Juridiskās personas pievienošana 1	Notiek divi funkcijas izsaukumi. Pirmajā gadījumā tiek pievienota jur. persona, kura jau ir reģistrēta šīm uzņēmumam kā dalībnieks, otrajā jur. personas dati nesakrīt ar sistēmā glabātajiem.	Pirmajā gadījumā HTTP atbildes kods ir 408, otrajā ir 410.	1.3.2.2
Juridiskās personas pievienošana 2	Notiek jur. personas pievienošana, kuras dati jau ir glabāti sistēmā.	HTTP atbildes kods ir 200.	1.3.2.2
Juridiskās personas pievienošana 3	Notiek jur. personas pievienošana, kuras dati vēl nav glabāti sistēmā.	HTTP atbildes kods ir 200.	1.3.2.2
Dalībnieku saraksta saņemšana (fiziskās personas)	Notiek funkcijas izsaukums uzņēmumam, kuram ir zināms, ka ir 1 dalībnieks (fiziskā persona).	HTTP atbildes kods ir 200 un atgriežamo ierakstu skaits ir 1.	1.3.2.3
Dalībnieku saraksta saņemšana (juridiskā personas)	Notiek funkcijas izsaukums uzņēmumam, kuram ir zināms, ka ir 1 dalībnieks (juridiskā persona).	HTTP atbildes kods ir 200 un atgriežamo ierakstu skaits ir 1.	1.3.2.4
Fiziskās personas ar personas kodu datu pārbaude	Notiek divi funkcijas izsaukumi. Pirmajā gadījuma funkcijai tiek padoti nekorekti personas dati, otrajā korekti.	Abos gadījumos HTTP atbildes kods ir 200. Pirmajā gadījumā Būla vērtība ir aplama, otrajā patiesa.	1.3.2.8
Fiziskās personas bez personas koda datu pārbaude	Notiek divi funkcijas izsaukumi. Pirmajā gadījuma funkcijai tiek padoti nekorekti personas dati, otrajā korekti.	Abos gadījumos HTTP atbildes kods ir 200. Pirmajā gadījumā Būla vērtība ir aplama, otrajā patiesa.	1.3.2.9

Juridiskās personas datu pārbaude	Notiek divi funkcijas izsaukumi. Pirmajā gadījuma funkcijai tiek padoti nekorekti jur. personas dati, otrajā korekti.	Abos gadījumos HTTP atbildes kods ir 200. Pirmajā gadījumā Būla vērtība ir aplama, otrajā patiesa.	1.3.2.7

3.5. tabula

Uzņēmumu dalībnieku servisa iekšējo funkciju vienībtesti

Nosaukums	Apraksts	Sagaidāmais rezultāts
Fiziskās personas saglabāšana	Tiek izveidotas divas personas. Pirmā satur vairākus personas datu ierakstus (*person history*) un ir personas kods, otrajai personas identifikācijai tiek izmantoti personas apliecinošā dokumenta dati.	Saglabātie ieraksti sakrīt ar pieprasījumā aizsūtītiem.
Juridiskās personas saglabāšana	Tiek izveidotas divas jur. personas. Pirmajai nav reģistrācijas numura (reģistrācija nav līdz galam pabeigta), otrā ir reģistrēta ārzemēs.	Saglabātie ieraksti sakrīt ar pieprasījumā aizsūtītiem.
Dalībnieka saglabāšana	Tiek izveidoti divi dalībnieki pirmajam uzņēmumam. Pirmais ir fiziska persona, otrais ir juridiska persona.	Saglabātie ieraksti sakrīt ar pieprasījumā aizsūtītiem.
Uzņēmuma meklēšana pēc valsts un nosaukuma	Notiek divas uzņēmuma meklēšanas. Pirmajā gadījumā tiek meklēts neeksistējošs uzņēmums, otrajā meklēšana notiek pēc otrā uzņēmuma valsts un nosaukuma.	Pirmajā gadījumā nekas netiek atrasts, otrajā tiek atrasts uzņēmums, kuram reģistrācija numurs sakrīt ar otrā uzņēmuma reģistrācijas numuru.
Uzņēmuma meklēšana pēc valsts un reģistrācijas numura	Notiek divas uzņēmuma meklēšanas. Pirmajā gadījumā tiek meklēts neeksistējošs uzņēmums, otrajā meklēšana notiek pēc otrā uzņēmuma valsts un reģistrācijas numura.	Pirmajā gadījumā nekas netiek atrasts, otrajā tiek atrasts uzņēmums, kuram nosaukums sakrīt ar otrā uzņēmuma nosaukumu.

Uzņēmuma meklēšana pēc identifikatora	Notiek uzņēmuma meklēšana pēc otrā uzņēmuma identifikatora.	Tiek atrasts uzņēmums, kura dati sakrīt ar otrā uzņēmuma datiem.
Personas meklēšana pēc personas koda	Notiek personas meklēšana pēc pirmās personas personas koda	Tiek atrasta persona.
Jaunākā personas datu ieraksta meklēšana pēc lietotāja identifikatora	Notiek jaunākā personas datu ieraksta meklēšana, izmantojot pirmās personas identifikatoru.	Tiek atrasts personas datu ieraksts, kura vārds un uzvārds sakrīt ar pēdējo saglabāto.
Dalībnieka (fiziskā persona) meklēšana pēc personas identifikatora un uzņēmuma identifikatora	Notiek divas meklēšanas. Pirmajā gadījumā tiek meklēts neeksistējošs dalībnieks, otrajā notiek pirmā dalībnieka meklēšana.	Pirmajā gadījumā nekas netiek atrasts, otrajā tiek atrasts dalībnieks.
Dalībnieka (juridiskā persona) meklēšana pēc	Notiek divas meklēšanas. Pirmajā gadījumā tiek meklēts neeksistējoša dalībnieks, otrajā notiek otrā dalībnieka meklēšana.	Pirmajā gadījumā nekas netiek atrasts, otrajā tiek atrasts dalībnieks.
Dalībnieku meklēšana pēc uzņēmuma identifikatora	Notiek dalībnieku meklēšana pēc pirmā uzņēmuma identifikatora.	Tiek atrasti divi dalībnieki.
Dalībnieku (fizisko personu) meklēšana pēc uzņēmuma identifikatora	Notiek dalībnieku meklēšana pēc pirmā uzņēmuma identifikatora.	Tiek atrasts viens dalībnieks.
Dalībnieku (juridisko personu) meklēšana pēc uzņēmuma identifikatora	Notiek dalībnieku meklēšana pēc pirmā uzņēmuma identifikatora.	Tiek atrasts viens dalībnieks.
Dalībnieka meklēšana pēc identifikatora	Tiek meklēts dalībnieks pēc pirmā dalībnieka identifikators.	Tiek atrasts dalībnieks, kura dati sakrīt ar pirmā dalībnieka datiem
Valsts meklēšana pēc valsts koda	Notiek valsts meklēšana pēc “LV” koda.	Tiek atrasta valsts. Valsts nosaukums latviešu valodā ir “Latvija”, bet angļu valodā ir “Latvia”.

3.6. tabula

Uzņēmumu dalībnieku servisa “SoapUI” vienībtesti

Nosaukums	Apraksts	Sagaidāmais rezultāts	Funkcionālās prasības Nr.	Testpiemēra numurs
Fiziskās personas pievienošana	Uzņēmumam kā dalībnieks tiek pievienota fiziska	HTTP atbildes kods ir 200.	1.3.2.1	1

	persona ar korektiem datiem.			
Dalībnieku saraksta saņemšana (fiziskās personas)	Tiek izsaukta funkcija dalībnieku saraksta saņemšanai uzņēmumam, kuram tikko bija pievienots dalībnieks.	HTTP atbildes kods ir 200.	1.3.2.3	1
Dalībnieka dzēšana	Uzņēmumam tiek dzēsts pievienots dalībnieks.	HTTP atbildes kods ir 200.	1.3.2.5	1
Dalībnieka dzēšana	Notiek dalībnieka dzēšana, kurš bija izdzēsts iepriekšējā testā.	HTTP atbildes kods ir 405.	1.3.2.5	1
Juridiskās personas pievienošana	Uzņēmumam kā dalībnieks tiek pievienota juridiska persona ar korektiem datiem.	HTTP atbildes kods ir 200.	1.3.2.2	2
Dalībnieku saraksta saņemšana (juridiskās personas)	Tiek izsaukta funkcija dalībnieku saraksta saņemšanai uzņēmumam, kuram tikko bija pievienots dalībnieks.	HTTP atbildes kods ir 200.	1.3.2.4	2
Dalībnieka dzēšana	Uzņēmumam tiek dzēsts pievienots dalībnieks.	HTTP atbildes kods ir 200.	1.3.2.5	2
Dalībnieka dzēšana	Notiek dalībnieka dzēšana, kurš bija izdzēsts iepriekšējā testā.	HTTP atbildes kods ir 405.	1.3.2.5	2

3.3.3. Uzņēmumu amatpersonu serviss

3.7. tabula
Uzņēmumu amatpersonu servisa funkciju vienībtesti

Nosaukums	Apraksts	Sagaidāmais rezultāts	Funkcionālās prasības Nr.
Amatpersonas pievienošana 1	Notiek divi funkcijas izsaukumi. Pirmajā gadījumā tiek pievienota persona, kura jau ir reģistrēta šīm uzņēmumam kā amatpersona, otrajā personas dati nesakrīt ar sistēmā glabātajiem.	Pirmajā gadījumā HTTP atbildes kods ir 408, otrajā ir 410.	1.3.3.1
Amatpersonas pievienošana 2	Notiek personas ar personas kodu pievienošana, kuras dati jau ir glabāti sistēmā.	HTTP atbildes kods ir 200.	1.3.3.1
Amatpersonas pievienošana 3	Notiek personas ar personas kodu pievienošana, kuras dati vēl nav glabāti sistēmā.	HTTP atbildes kods ir 200.	1.3.3.1
Amatpersonu saraksta saņemšana	Notiek funkcijas izsaukums uzņēmumam, kuram ir zināms, ka ir 1 amatpersona.	HTTP atbildes kods ir 200 un atgriežamo ierakstu skaits ir 1.	1.3.3.2

3.8. tabula
Uzņēmumu dalībnieku servisa iekšējo funkciju vienībtesti

Nosaukums	Apraksts	Sagaidāmais rezultāts
Fiziskās personas saglabāšana	Tiek izveidotas divas personas. Pirmā satur vairākus personas datu ierakstus (*person history*) un ir personas kods, otrajai personas identifikācijai tiek izmantoti personas apliecinošā dokumenta dati.	Saglabātie ieraksti sakrīt ar pieprasījumā aizsūtītiem.
Juridiskās personas saglabāšana	Tiek izveidotas divas jur. personas. Pirmajai nav reģistrācijas numura	Saglabātie ieraksti sakrīt ar pieprasījumā aizsūtītiem.

	(reģistrācija nav līdz galam pabeigta), otrā ir reģistrēta ārzemēs.	
Amatpersonas saglabāšana	Pirmā persona tiek saglabāta ka amatpersona pirmajam uzņēmumam.	Saglabātie ieraksti sakrīt ar pieprasījumā aizsūtītiem.
Uzņēmuma meklēšana pēc valsts un nosaukuma	Notiek divas uzņēmuma meklēšanas. Pirmajā gadījumā tiek meklēts neeksistējošs uzņēmums, otrajā meklēšana notiek pēc otrā uzņēmuma valsts un nosaukuma.	Pirmajā gadījumā nekas netiek atrasts, otrajā tiek atrasts uzņēmums, kuram reģistrācija numurs sakrīt ar otrā uzņēmuma reģistrācijas numuru.
Uzņēmuma meklēšana pēc valsts un reģistrācijas numura	Notiek divas uzņēmuma meklēšanas. Pirmajā gadījumā tiek meklēts neeksistējošs uzņēmums, otrajā meklēšana notiek pēc otrā uzņēmuma valsts un reģistrācijas numura.	Pirmajā gadījumā nekas netiek atrasts, otrajā tiek atrasts uzņēmums, kuram nosaukums sakrīt ar otrā uzņēmuma nosaukumu.
Uzņēmuma meklēšana pēc identifikatora	Notiek uzņēmuma meklēšana pēc otrā uzņēmuma identifikatora.	Tiek atrasts uzņēmums, kura dati sakrīt ar otrā uzņēmuma datiem.
Personas meklēšana pēc personas koda	Notiek personas meklēšana pēc pirmās personas personas koda.	Tiek atrasta persona.
Personas meklēšana pēc personas apliecinoša dokumenta datiem	Notiek personas meklēšana pēc otrās personas dokumenta datiem.	Tiek atrasta persona. Personas dzīvesvietas valsts sakrīt ar iepriekš saglabāto.
Amatpersonas meklēšana pēc personas identifikatora un uzņēmuma identifikatora	Notiek divas meklēšanas. Pirmajā gadījumā tiek meklēta neeksistējoša amatpersona, otrajā notiek pirmās amatpersonas meklēšana.	Pirmajā gadījumā nekas netiek atrasts, otrajā tiek atrasta amatpersona.
Amatpersonu meklēšana pēc uzņēmuma identifikatora	Notiek amatpersonu meklēšana pēc pirmā uzņēmuma identifikatora.	Tiek atrasts viena amatpersona.
Amatpersonas meklēšana pēc identifikatora	Tiek meklēta amatpersona pēc pirmās amatpersonas identifikatora.	Tiek atrasta amatpersona, kuras dati sakrīt ar pirmā dalībnieka datiem.
Valsts meklēšana pēc valsts koda	Notiek valsts meklēšana pēc “LV” koda.	Tiek atrasta valsts. Valsts nosaukums latviešu valodā ir “Latvija”, bet angļu valodā ir “Latvia”

3.9. tabula

Uzņēmumu dalībnieku servisa "SoapUI" vienībtesti

Nosaukums	Apraksts	Sagaidāmais rezultāts	Funkcionālās prasības Nr.	Testpiemēra numurs
Fiziskās personas pievienošana	Uzņēmumam kā amatpersona tiek pievienota fiziska persona ar korektiem datiem.	HTTP atbildes kods ir 200.	1.3.3.1	1
Amatpersonu saraksta saņemšana	Tiek izsaukta funkcija amatpersonu saraksta saņemšanai uzņēmumam, kuram tikko bija pievienota amatpersona.	HTTP atbildes kods ir 200.	1.3.3.2	1
Amatpersonas dzēšana	Uzņēmumam tiek dzēsts pievienota amatpersona.	HTTP atbildes kods ir 200.	1.3.3.3	1
Amatpersonas dzēšana	Notiek amatpersonas dzēšana, kura bija izdzēsta iepriekšējā testā.	HTTP atbildes kods ir 405.	1.3.3.3	1

3.3.4. Adrešu serviss

3.10. tabula

Adrešu servisa funkciju vienībtesti

Nosaukums	Apraksts	Sagaidāmais rezultāts	Funkcionālās prasības Nr.
Ciemu saraksta saņemšana ar vecāka identifikatoru	Notiek funkcijas izsaukums ar vecāka adresācijas objekta identifikatoru, kurā ir zināms, ka ir vismaz viens ciems.	HTTP atbildes kods ir 200 un funkcijas izvade nav tukša.	1.3.4.5
Dzīvokļu saraksta saņemšana ar vecāka identifikatoru	Notiek funkcijas izsaukums ar vecāka adresācijas objekta identifikatoru, kurā ir zināms, ka ir vismaz viens dzīvoklis.	HTTP atbildes kods ir 200 un funkcijas izvade nav tukša.	1.3.4.8

Ielu saraksta saņemšana ar vecāka identifikatoru	Notiek funkcijas izsaukums ar vecāka adresācijas objekta identifikatoru, kurā ir zināms, ka ir vismaz viena iela.	HTTP atbildes kods ir 200 un funkcijas izvade nav tukša.	1.3.4.6
Māju saraksta saņemšana ar vecāka identifikatoru	Notiek funkcijas izsaukums ar vecāka adresācijas objekta identifikatoru, kurā ir zināms, ka ir vismaz viena māja.	HTTP atbildes kods ir 200 un funkcijas izvade nav tukša.	1.3.4.7
Novadu saraksta saņemšana	Notiek funkcijas izsaukums.	HTTP atbildes kods ir 200 un funkcijas izvade nav tukša.	1.3.4.2
Pagastu saraksta saņemšana ar vecāka identifikatoru	Notiek funkcijas izsaukums ar vecāka adresācijas objekta identifikatoru, kurā ir zināms, ka ir vismaz viens pagasts.	HTTP atbildes kods ir 200 un funkcijas izvade nav tukša.	1.3.4.4
Vecāka adresācijas objekta saņemšana	Notiek funkcijas izsaukums ar adresācijas objekta identifikatoru, kuram ir zināms kāds ir vecākais adresācijas objekts.	HTTP atbildes kods ir 200. Funkcijas izvade nav tukša un atgrieztā adresācijas objekta dati sakrīt ar sagaidāmiem.	1.3.4.9
Pilsētu saraksta saņemšana	Notiek funkcijas izsaukums.	HTTP atbildes kods ir 200 un funkcijas izvade nav tukša.	1.3.4.3
Pilnas adreses ģenerēšana 1	Notiek funkcijas izsaukums ar konkrētā dzīvokļa identifikatoru, kuram ir zināma pilna adrese.	HTTP atbildes kods ir 200. Funkcijas izvade sakrīt ar sagaidāmo.	1.3.4.10
Pilnas adreses ģenerēšana 2	Notiek funkcijas izsaukums ar konkrētās mājas identifikatoru, kurai ir zināma pilna adrese.	HTTP atbildes kods ir 200. Funkcijas izvade sakrīt ar sagaidāmo.	1.3.4.10

Pilsētu saraksta saņemšana ar vecāka identifikatoru	Notiek funkcijas izsaukums ar vecāka adresācijas objekta identifikatoru, kurā ir zināms, ka ir vismaz viena pilsēta.	HTTP atbildes kods ir 200 un funkcijas izvade nav tukšs.	1.3.4.3
Valstu saraksta saņemšana 1	Notiek funkcijas izsaukums ar valstu nosaukumiem latviešu valodā.	HTTP atbildes kods ir 200 un funkcijas izvade nav tukšs.	1.3.4.11
Valstu saraksta saņemšana 2	Notiek funkcijas izsaukums ar valstu nosaukumiem angļu valodā.	HTTP atbildes kods ir 200 un funkcijas izvade nav tukšs.	1.3.4.11

3.10. tabula

Adrešu servisa iekšējo funkciju vienībtesti

Nosaukums	Apraksts	Sagaidāmais rezultāts
Adresācijas objekta meklēšana pēc identifikatora.	Notiek adresācijas objekta meklēšana, zinot tā identifikatoru.	Tiek atrasts adresācijas objekts, kura identifikators sakrīt ar padoto.
Vecāka adresācijas objekta meklēšana.	Notiek vecāka adresācijas objekta meklēšana, zinot tā iekšējā adresācijas objekta identifikatoru.	Tiek atrasts adresācijas objekts, kas sakrīt ar sagaidāmo.
Adresācijas objektu meklēšana 1.	Notiek adresācijas objektu meklēšana pēc to veida. Ir iepriekš zināms, ka rezultātā jābūt viens ieraksts.	Tiek atrasts saraksts ar adresācijas objektiem, kurā ir viens objekts.
Adresācijas objektu meklēšana 1.	Notiek adresācijas objektu meklēšana pēc to veida un vecāka objekta identifikatoru. Ir iepriekš zināms, cik objektu jābūt sarakstā.	Tiek atrasts saraksts ar adresācijas objektiem. Objektu skaits sakrīt ar sagaidāmo.

4. PROJEKTA ORGANIZĀCIJA

Projekta izstrādes sākumā bija zināms kopējais darba plāns, bet tas nebija pietiekami detalizēts. Projekts bija sadalīts sīkākās daļās, kur katra daļa atbilst savam mikroservisam, un pirms jauna mikroservisa izstrādes prasības bija precizētas, apkopotas, saskaņotas un tika izveidota prasību specifikācija šīm servisam. Tad vadoties pēc izstrādātās specifikācijas, kas kalpoja par uzdevumu sarakstu, tika veidots programmatūras projektējuma apraksts un veikti programmēšanas darbi. Programmēšanas gaitā notika regulāras, parasti reizi divās dienās, tikšanās ar pārējiem izstrādes komandas dalībniekiem, kur bija apspriesti padarītie darbi un sastaptās grūtības, saskaņoti tuvākā laika plāni. Izstrādes laikā tika veiktas arī izmaiņas sākotnējā plānā, jo tajā tika atklātas neskaidrības un nepilnības. Citiem vārdiem, izstrāde notika pēc SCRUM metodoloģijas.

Sistēma bija izstrādāta komandā, bet šī darba ietvaros ir aprakstīta tikai patstāvīgi izstrādātā sistēmas daļa, kā arī pēc nepieciešamības tika pieminēta pārējā sistēmas daļa labākā priekšstata izveidošanai par kopējo sistēmas struktūru.

Pēc visu mikroservisu izstrādes sekoja testēšanas stadija. Tajā tika izveidoti vienībtesti gandrīz katrai sistēmas funkcijai. Testu rakstīšanas laikā tika atklātas vairākas iepriekš nesastaptas sistēmas kļūdas un nepilnības, kuras bija novērstas. Rezultātā visi testi tiek veiksmīgi izieti, kas norāda uz to, ka iejaukšanās jau strādājošā sistēmā neizraisīja jaunas kļūdas.

Visa projekta izstrādes laika paralēli notika arī sistēmas piegādes automatizācijas pasākumi, kas būtiski ietekmēja izstrādes procesu, bet rezultātā tika panākts automātiska sistēmas attēla izveidošana un tā izvietošana izstrādes vidē.

5. KVALITĀTES NODROŠINĀŠANA

Darba izstrādes laikā tika veidota programmatūras dokumentācija , kurā tika aprakstītas sistēmas nepieciešamās prasības, veikts programmatūras projektējums, kā arī vēlāk tika aprakstīta testēšanas dokumentācija. Programmējot sistēmu tika stingri pārbaudīt, vai sistēmas uzvedība un reakcija atbilst prasību specifikācijā prasībās minētajam.

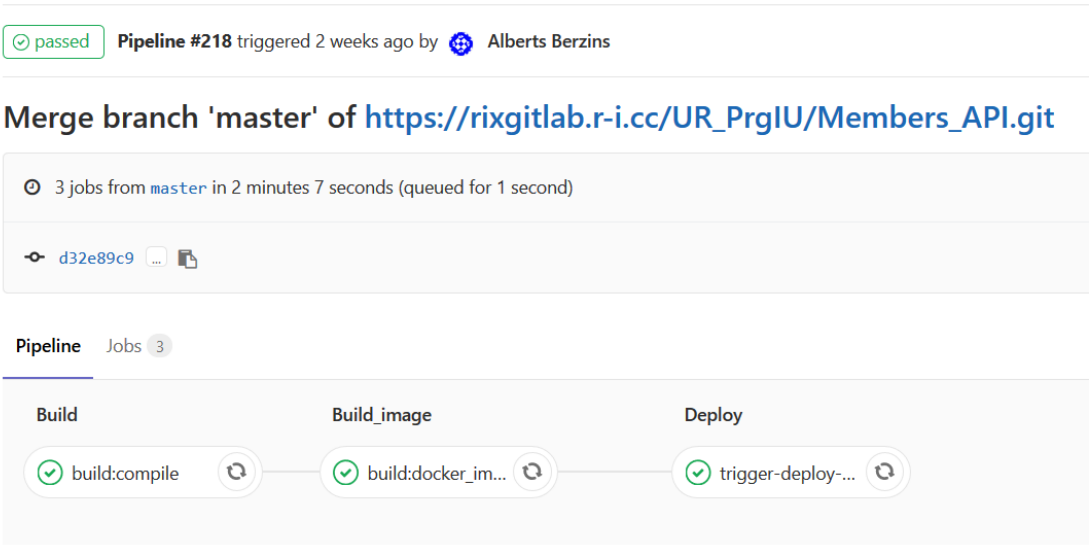
Programmatūras kods tika komentēts un pēc iespējas labāk un pareizāk strukturēts, lai to spētu viegli lasīt un pārvaldīt ikkatrs izstrādātājs. Tika ņemti vērā arī "Spring Boot" ietvara un "Swagger" rīka dokumentācijā aprakstītie ieteikumi.

Koda kvalitātes nodrošināšanai tika ievērota izmantotās programmēšanas valodas labā prakse, kā arī "SonarLint" un "SonarQube" rīki, kuri nodrošina nepārtrauktu koda kvalitātes automātisko analīzi. Izmaiņas regulāri tika saglabātas versiju kontroles sistēmā. Lai nodrošinātu programmatūras korektu darbību, pēc izstrādes pabeigšanas tika veikti vienībtesti, kas nodrošina sistēmas pareizu uzvedību, atbilstoši sākumā uzstādītiem mērķiem.

6. KONFIGURĀCIJU PĀRVALDĪBA

Programmatūras pirmkoda glabāšanai tika izmantota Git sistēma, ar "GitLab" repozitoriju pamatā. Tas nodrošina koda un versiju izmaiņu kontroli, kā arī koordinē programmatūras izstrādi starp vairākiem izstrādātājiem vienlaicīgi. Pēc katras izmaiņu saglabāšanas repozitorija notiek automatizēto darbību virkne, kura noved pie jaunākās versijas uzstādīšanas izstrādes vidē. Katrs sistēmas mikroserviss ir realizēts kā atsevišķs projekts, kā rezultātā sistēma ir viegli mērogojama un katra sistēmas daļa ir salīdzinoši viegli aizvietojama.

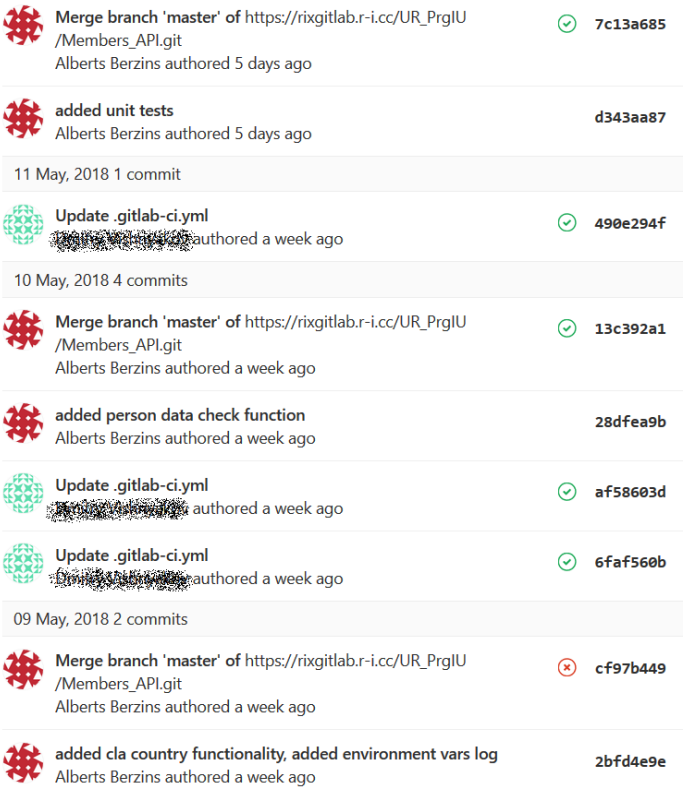
Zemāk ir redzams (*6.1. att.*), kā izskatās projekta automātiskā izvietošana. Attēlā ir redzama saglabātās versijas pamatinformācija un automātiski izpildīto darbību saraksts. Vizualizācijai tiek izmantotas GitLab iebūvētās iespējas:



6.1. att. **Git commit skats GitLab lapā**

Attēlā (*6.1. att.*) var redzēt, ka notiek 3 darbības: projekta kompilācija, Docker attēla izveidošana un izvietošana. Katras darbības kreisajā pusē tiek atzīmēts tās statuss, kurš šajā gadījumā ir visur veiksmīgs.

Attēlā zemāk (*6.2. att.*) ir attēlots saglabāto versiju vēsture, kur arī var redzēt kopējo automātiskās izvietošanas procesa statusu katrai versijai.



6.2. att. **Git commit history skats GitLab lapā**

Augstāk aprakstītas sistēmas īpatnības noved pie tā, ka sistēmai jābūt viegli uzstādāmai jebkurā tam piemērotā vidē, kā arī visām sistēmas parametriem jābūt viegli konfigurējamiem. Runājot par servera puses daļu, kas ir šī darba tēma, viena no lielākajām un svarīgākajām problēmām šajā virzienā ir servisu un datu bāzes savstarpējā konfigurācija. Datu bāzei nekas specifisks netika darīts, bet servisu konfigurācija notiek, izmantojot vides mainīgos (*Environment variables*). Rezultātā, ir iespējams uzstādīt savienojumu, neaiztiekot sistēmas pirmkodu, kas noved pie iespējas uzstādīt vienu un to pašu sistēmas versiju vairākās vidēs.

Datu bāzes satura rezerves kopiju veidošanai nebija nepieciešamības, toties datu bāzes uzstādīšanas izstrādes vides dēļ, datu bāzes jaunākās versijas tika regulāri saglabātas atsevišķajā *Docker* attēlā, kas savā ziņā ir rezerves kopēšana.

Var secināt, ka konfigurācijas pārvaldei šajā projektā ir vērsta lielāka uzmanība, nekā tas notiek citos projektos, kas ir saistīts ar sistēmas prasībām un specifiku.

7. DARBIETILPĪBAS NOVĒRTĒJUMS

7.1. COCOMO II

Darbietilpības novērtēšu, par pamatu ņemot sarakstīto pirmkoda rindiņu skaitu. Izmantojot izstrādes vidē iebūvētos rīkus tika konstatēts, ka manis veidotā sistēmas daļa satur aptuveni 2800 Java koda rindiņas, no kurām aptuveni 900 ir atkalizmantotas. Sadalījums starp servisiem ir aptuveni šāds:

- Lietotāju serviss – 730 rindiņas
- Uzņēmumu dalībnieku serviss – 950 rindiņas
- Uzņēmumu amatpersonu serviss – 540 rindiņas
- Adrešu serviss – 580 rindiņas

Novērtēšanai tika izmantota "COCOMO II" programmatūras darbietilpības prognozēšanas un plānošanas kalkulators. Kalkulators ņem vērā arī projekta mērogošanas vērtības:

7.1. tabula

"COCOMO II" darbietilpības aprēķina parametri

Iespaidošais faktors	Vērtība
Produkta	
Nepieciešamā programmatūras uzticamība	Zems
Datu bāzes izmērs	Zems
Produkta sarežģītība	Zems
Produkta atkalizmantojamība	Augsts
Dzīves cikla modeļa dokumentācijas vajadzība	Zems
Platformas	
Laika ierobežojumi	Vidējs
Datu krātuves ierobežojumi	Vidējs
Vides mainīgums	Augsts
Personāla	
Analītiskas spējas	Vidējs
Programmētāju spējas	Zems
Personāla nemainība	Ļoti augsts
Programmatūras pieredze	Zems
Programmēšanas valodas pieredze	Vidējs

Projekta	
Programmatūras izstrādes rīku izmantošana	Vidējs
Modernā programmēšanas prakse	Vidējs
Nepieciešamais izstrādes laiks	Vidējs

Ņemot vērā šos faktorus un koda rindiņas skaitu, kalkulators aprēķināja darbietilpību kā 4.6 personmēnešus.

Reālas izstrādes laiks bija aptuveni 3 personmēneši. Šādu atšķirību starp prognozēto un faktisko darbietilpību varētu izskaidrot ar to, ka "COCOMO II" kalkulators ir vairāk piemērots lieliem projektiem, kuriem ir vairāk izstrādātāji, kā arī ietekmējošie faktori bija izvēlēti subjektīvi.

7.2. PERT

Tā kā "COCOMO II" darbietilpības aprēķina metodes rezultāts ir tālu no realitātes, darbietilpība tika aprēķināta arī pēc alternatīvās metodes "PERT".

"PERT" metodē izstrādātājs pats novērtē aptuvenu darba apjomu un patērēto tam laiku, un izveido prognozi. Darbietilpības aprēķinam ir nepieciešamas 3 vērtības:

- t_a – optimistiskā prognoze
- t_m – reālā prognoze
- t_b – pesimistiskā prognoze

Sagaidāmā darbietilpība tiek aprēķināta pēc formulas $t_e = \frac{t_a+4t_m+t_b}{6}$

Zemāk ir autora novērtētā darbietilpība katram mikroservisam cilvēkdienās.

7.2. tabula

Prognozētā darbietilpība mikroservisiem

Mikroservisa nosaukums	**Optimistiskā prognoze**	**Reālā prognoze**	**Pesimistiskā prognoze**
Lietotāji	8	13	18
Dalībnieki	12	15	22
Amatpersonas	6	10	13
Adreses	6	11	15

$$t_e = \frac{32 + 4 * 49 + 68}{6} = 49.33 \sim 50$$

"PERT" metode parādīja, ka aptuvenā darbietilpība mikroservisiem ir 50 dienas, kas ir tuvu reālam patērētam laikam, bet aprēķinos netika ņemts vērā patērētais laiks Docker attēlu

izveidošanai un saistīto skriptu rakstīšanai, datu bāzes izveidošanas laiks, kā arī tika patērētais laiks servisu integrācijai ar lietotāju pusi. Rezultātā kopējais patērētais laiks ir aptuveni 60 dienas, kas ir 3 personmēneši.

8. SECINĀJUMI

Kvalifikācijas darba ietvaros tika izstrādāta uzņēmumu reģistrācijas sistēmas servera puse, kurā ietilpst kā funkcionējošs programmatūras kods, tā arī pavaddokuments tam, kas savukārt sastāv no prasību specifikācijas, programmatūras projektējuma apraksta un testēšanas dokumentācijas.

Projekta izstrādes sākums aizkavējās, tā kā liela laika daļa bija veltīta mācībām. Var atzīmēt to, ka autoram neviena no izmantotām projektā tehnoloģijām nebija iepriekš pazīstama, tāpēc jau izstrādes laikā bija iegūtas jaunas zināšanas un pieredze.

Šī darba izstrādes laikā autors ieguva svarīgas zināšanas un prasmes darbā ar Java programmēšanas valodu un Spring Boot ietvaru. Tika iegūtas zināšanas par programmatūras testēšanu un piegādes automatizāciju, strādājot ar Docker un Kubernetes. Darbs deva iespēju iepazīties un detalizētāk izpētīt mūsdienu pieeju lielu projektu izstrādei.

Kopumā varu secināt, ka darbs ir veiksmīgi paveikts, uzstādītie mērķi ir sasniegti un ir iegūtas vērtīgas zināšanas. Skatoties uz paveikto darbu ir pārliecība, ka tam ir liels attīstīšanas potenciāls un perspektīva.

ATSAUCES

1. Valsts adrešu reģistrs [tiešsaite] – [atsauce 30.04.2018.]. Pieejams: http://www.vzd.gov.lv/lv/par-mums/darbibas-jomas/adresu-registrs/
2. Swagger rīka oficiālā dokumentācija [tiešsaite] – [atsauce 01.05.2018]. Pieejams: https://swagger.io/specification/
3. Lietojumprogrammas saskarne [tiešsaite] – [atsauce 21.05.2018]. Pieejams: https://lv.wikipedia.org/wiki/Lietojumprogrammas_saskarne
4. "What is DevOps" [tiešsaite] – [atsauce 22.05.2018]. Pieejams: https://theagileadmin.com/what-is-devops/
5. Docker oficiālā dokumentācija [tiešsaite] – [atsauce 22.05.2018]. Pieejams: https://docs.docker.com/
6. Spring Boot oficiālā dokumentācijas [tiešsaite] – [atsauce 22.05.2018]. Pieejams: https://docs.spring.io/spring-boot/docs/current/reference/html/
7. "Piedzīvojumi ar Uzņēmumu reģistra e-pakalpojumiem. Sabojāti nervi vai pastaiga parkā?" [tiešsaite] – [atsauce 23.05.2018]. Pieejams: http://blumentals.mozello.lv/blogs/params/post/1441656/piedzivojumi-ar-uznemumu-registra-e-pakalpojumiem
8. "Microservices Architecture and Design Principles" [tiešsaite] – atsauce [23.05.2018]. Pieejams: https://www.ness.com/microservices-architecture-and-design-principles-2/
9. "PERT" darbietiplības metode [tiešsaite] – atsauce [24.05.2018]. Pieejams: https://www.slideshare.net/tomeh/program-evaluation-review-technique-pert

PIELIKUMI

Koda fragmenti

Swagger redaktora kods

Zemāk ir Swagger redaktora kods YAML formātā, no kura tiek ģenerēts projekta sākumpunkts.

```yaml
swagger: '2.0'
info:
  version: 1.0.0
  title: UR proto officers API
host: localhost
basePath: /v2
schemes:
  - http
paths:
  /officers:
    post:
      tags:
        - Officers
      summary: Pievienot jaunu amatpersonu
      description: ''
      operationId: addOfficer
      consumes:
        - application/json
      produces:
        - application/json
      parameters:
        - in: body
          name: body
          description: ""
          required: true
          schema:
            $ref: '#/definitions/Officer'
      responses:
        '200':
          description: successful operation
          schema:
            items:
              $ref: '#/definitions/Officer'
        '405':
          description: Invalid input
        '409':
          description: already registered
    delete:
      tags:
        - Officers
      summary: Dzēst amatpersonu uzņēmumam
      description: ''
      operationId: deleteOfficer
      produces:
        - application/json
      parameters:
        - in: query
          name: officer_id
          description: ""
          type:
            integer
```

```
          required: true
      responses:
        '200':
          description: successful operation
        '405':
          description: Invalid input
  /officers/{enterprise_id}:
    get:
      tags:
        - Officers
      summary: Saņemt amatpersonu sarakstu uzņēmumam
      description: ''
      operationId: getOfficers
      produces:
        - application/json
      parameters:
        - in: path
          name: enterprise_id
          description: ""
          type:
            integer
          required: true
      responses:
        '200':
          description: successful operation
          schema:
            type: "array"
            items:
              $ref: '#/definitions/Officer'
        '405':
          description: Invalid input
    delete:
      tags:
        - Officers
      summary: Dzēst amatpersonu sarakstu uzņēmumam
      description: ''
      operationId: deleteOfficers
      produces:
        - application/json
      parameters:
        - in: path
          name: enterprise_id
          description: ""
          type:
            integer
          required: true
      responses:
        '200':
          description: successful operation
        '405':
          description: Invalid input
definitions:
  Officer:
    type: object
    properties:
      id:
        type: integer
      lietotajaId:
        type: integer
      uznemumaId:
        type: integer
      vards:
        type: string
```

```
      uzvards:
        type: string
      irPK:
        type: boolean
      PK:
        type: string
      dzimsanasDatums:
        type: string
        format: date
      dokNr:
        type: string
      dokValsts:
        type: string
      dzivesvietasValsts:
        type: string
      amats:
        type: string
      tiesibas:
        type: number
      tiesibasSkaits:
        type: number
    required:
      - uznemumaId
      - lietotajaId
      - vards
      - uzvards
      - irPK
      - dzimsanasDatums
      - tiesibas
```

Java API funkciju deklarācijas.

Zemāk ir Amatpersonu lietojumprogrammas saskarnes (API) funkciju saraksts bez realizācijas, šo no šī saraksta tiek ģenerēta Swagger saskarne. Kods zemāk var nesakrist ar augstāk minēto Swagger redaktora kodu, tā kā tas tika izmantots tikai ka servisa pamats un bija vairākreiz rediģēts.

```
@Api(value = "officers", description = "the officers API")
public interface OfficersApi {

    @CrossOrigin(origins = "*", allowedHeaders = "*")
    @ApiOperation(value = "Pievienot jaunu amatpersonu", nickname =
"addOfficer", notes = "", tags={ "Officers", })
    @ApiResponses(value = {
            @ApiResponse(code = 200, message = "successful operation"),
            @ApiResponse(code = 405, message = "Invalid tiesibas"),
            @ApiResponse(code = 406, message = "Invalid uznemumaId"),
            @ApiResponse(code = 410, message = "Invalid person info"),
            @ApiResponse(code = 408, message = "Already registered"),
            @ApiResponse(code = 409, message = "Unexpected error")})
    @RequestMapping(value = "/officers",
        produces = { "application/json" },
        consumes = { "application/json" },
        method = RequestMethod.POST)
    ResponseEntity<Void> addOfficer(@ApiParam(value = "" ,required=true )
@Valid @RequestBody OfficerApi body);

    @CrossOrigin(origins = "*", allowedHeaders = "*")
    @ApiOperation(value = "Dzēst amatpersonu uzņēmumam", nickname =
"deleteOfficer", notes = "", tags={ "Officers", })
```

```
    @ApiResponses(value = {
        @ApiResponse(code = 200, message = "successful operation"),
        @ApiResponse(code = 405, message = "Invalid input") })
    @RequestMapping(value = "/officers",
        produces = { "application/json" },
        method = RequestMethod.DELETE)
    ResponseEntity<Void> deleteOfficer(@NotNull @ApiParam(value = "",
required = true) @Valid @RequestParam(value = "officer_id", required =
true) Integer officerId);

    @CrossOrigin(origins = "*", allowedHeaders = "*")
    @ApiOperation(value = "Dzēst amatpersonu sarakstu uzņēmumam", nickname
= "deleteOfficers", notes = "", tags={ "Officers", })
    @ApiResponses(value = {
        @ApiResponse(code = 200, message = "successful operation"),
        @ApiResponse(code = 405, message = "Invalid input") })
    @RequestMapping(value = "/officers/{enterprise_id}",
        produces = { "application/json" },
        method = RequestMethod.DELETE)
    ResponseEntity<Void> deleteOfficers(@ApiParam(value = "",required=true)
@PathVariable("enterprise_id") Integer enterpriseId);

    @CrossOrigin(origins = "*", allowedHeaders = "*")
    @ApiOperation(value = "Saņemt amatpersonu sarakstu uzņēmumam", nickname
= "getOfficers", notes = "", response = OfficerApi.class, responseContainer
= "List", tags={ "Officers", })
    @ApiResponses(value = {
        @ApiResponse(code = 200, message = "successful operation", response
= OfficerApi.class, responseContainer = "List"),
        @ApiResponse(code = 405, message = "Invalid input") })
    @RequestMapping(value = "/officers/{enterprise_id}",
        produces = { "application/json" },
        method = RequestMethod.GET)
    ResponseEntity<List<OfficerApi>> getOfficers(@ApiParam(value =
"",required=true) @PathVariable("enterprise_id") Integer enterpriseId);

}
```

Amatpersonas pievienošana

Apjomīgākā funkcija no Amatpersonu servisa.

```
public ResponseEntity<Void> addOfficer(@ApiParam(value = "" ,required=true
)  @Valid @RequestBody OfficerApi body) {

    HttpHeaders headers = new HttpHeaders();
    headers.add("Content-Type", "application/json; charset=UTF-8");


    if (body.getTiesibas()<1 || body.getTiesibas()>4 ||
            body.getTiesibas()==4 && ( body.getTiesibasSkaits()==null ||
body.getTiesibasSkaits()<1 ))
        return new ResponseEntity<Void>(headers,
HttpStatus.METHOD_NOT_ALLOWED); //wrong tiesības

    LegalEntity legalEntity =
this.service.findLegalEntityById(body.getUznemumaId()); //find legal entity
    if (legalEntity == null ||
            legalEntity != null && !legalEntity.getCountry().equals(lv))
//if there is no such entity
        return new ResponseEntity<Void>(headers,
HttpStatus.NOT_ACCEPTABLE);
```

```
    Person person;
    if (body.isIrPK()) //find person
        person = service.findPersonByPersonCode(body.getPersonasKods());
    else
        person = service.findForeignPersonByIdentity(body.getDokNr(),
body.getDokValsts(), body.getDokInstit(), body.getDokVeids());

    if (person != null){ //if there is this person in DB
        PersonHistory personHistory =
service.findNewestPersonHistoryByPersonId(person.getId()); //get this
person
        if (!body.getVards().equals(personHistory.getName()) || //check if
stored and given data matches
                !body.getUzvards().equals(personHistory.getSurname()) ||
                person.getBirthDate() != null && body.getDzimsanasDatums()
!= null &&
!body.getDzimsanasDatums().equals(converter.convertToEntityAttribute(person
History.getPerson().getBirthDate())) ||
                !body.isIrPK() &&
!body.getDzivesvietasValsts().equals(personHistory.getPerson().getForeignPe
rson().getClaCountryCode().getCode())) {//in case its foreign person
            return new ResponseEntity<Void>(headers, HttpStatus.GONE);
//wrong data about person passed
        }
        if (this.service.findOfficerByUserIdLegalEntityId(person.getId(),
legalEntity.getCode())!=null) //if person is already officer of this entity
            return new ResponseEntity<Void>(headers,
HttpStatus.REQUEST_TIMEOUT); //already registered
    }
    else { //create new person
        person = new Person();
        PersonHistory personHistory = new PersonHistory();
        List<PersonHistory> personHistories = new
ArrayList<PersonHistory>();
        if (body.isIrPK())
            personHistory.setIsForeign(false);
        else {
            personHistory.setIsForeign(true);
            ForeignPerson foreignPerson = new ForeignPerson();

foreignPerson.setIdentityDocCountry(service.findCountryByCode(body.getDokVa
lsts()));

foreignPerson.setClaCountryCode(service.findCountryByCode(body.getDzivesvie
tasValsts()));
            foreignPerson.setIdentityDocId(body.getDokNr());

foreignPerson.setIdentityDocDate(converter.convertToDatabaseColumn(body.get
DokDatums()));
            foreignPerson.setIdentityDocInstitution(body.getDokInstit());
            foreignPerson.setIdentityDocType(body.getDokVeids());
            foreignPerson.setRegisteredDate(new
Timestamp(System.currentTimeMillis()));
            foreignPerson.setPerson(person);
            person.setForeignPerson(foreignPerson);
        }
        personHistory.setName(body.getVards());
        personHistory.setSurname(body.getUzvards());
        personHistory.setPersonCode(body.getPersonasKods());
        personHistory.setRegistered(new
Timestamp(System.currentTimeMillis())); //current
        personHistory.setPerson(person);
```

```
            personHistories.add(personHistory);

            person.setIsActive(true);

person.setBirthDate(converter.convertToDatabaseColumn(body.getDzimsanasDatu
ms()));
            person.setPersonHistories(personHistories);

            try { //save person
                this.service.savePerson(person);
            } catch (Exception e) {
                e.printStackTrace();
                return new ResponseEntity<Void>(headers, HttpStatus.CONFLICT);
            }

        }

        Officer officer = new Officer(); //create links
        officer.setLegalEntity(legalEntity);
        legalEntity.getOfficers().add(officer);
        officer.setPerson(person);
        if (person.getOfficers() == null) //if person have no list of officers
(if we just created person)
            person.setOfficers(new ArrayList<Officer>());
        person.getOfficers().add(officer);

        officer.setCreated(new Timestamp(System.currentTimeMillis()));
        officer.setIsRetired(false);
        officer.setPosition(body.getAmats());
        officer.setRepresentationScope(body.getTiesibas());
        if (body.getTiesibas()==4)
            officer.setRepresentationCount(body.getTiesibasSkaits());
        officer.setInstitution(body.getInstitucija());
        officer.setTypeOfAppointment(body.getIecelsanasVeids());

        try { //save officer
            this.service.saveOfficer(officer);
        } catch (Exception e) {
            e.printStackTrace();
            return new ResponseEntity<Void>(headers, HttpStatus.CONFLICT);
        }
        return new ResponseEntity<Void>(headers, HttpStatus.OK);
}
```

Pilsētu saraksta atgriešana

Viena no adresācijas objektu atgriešanas funkcijām, kura izveido pilsētu sarakstu. Zem tās ir arī palīgfunkcija.

```
public ResponseEntity<AddressList> getPilsetas() {

    HttpHeaders headers = new HttpHeaders();
    headers.add("Content-Type", "application/json; charset=UTF-8");
    headers.add("Access-Control-Allow-Origin", "*");

    String accept = request.getHeader("Accept");
    if (accept != null && accept.contains("application/json")) {
        return
ResponseEntity.status(HttpStatus.OK).headers(headers).body(getAddressObject
sById("104", null));
    }
```

```
    return new ResponseEntity<AddressList>(headers,
HttpStatus.NOT_IMPLEMENTED);
}


public AddressList getAddressObjectsById (String type, Integer parentId){

    List<ClaAddress> addresses = new ArrayList<ClaAddress>();
    if (parentId == null)
        addresses = this.service.getAddressesByType(type); //run query to
DB and save result in addresses list
    else
        addresses = this.service.getAddressesByType(type, parentId);

    AddressList addressList = new AddressList(); //create list that we will
return later
    for (ClaAddress temp: addresses){ //transform list of adresses to
another list of adresses (remove unnecessary fields)
        AddressItem obj = new AddressItem();
        obj.setId(temp.getId());
        obj.setName(temp.getName());
        addressList.addAddressesItem(obj);
    }
    return addressList;
}
```

GitLab CI

Automātiskā servisa izvietošana notiek ar zemāk redzamā skripta palīdzību, kurā serviss tiek kompilēts, no tā tiek izveidots Docker attēls un vadība tiek nodota nākamajai funkcijai.

Izveidošana:

```
image: maven:latest
services:
  - docker:dind

variables:
  DOCKER_HOST: 'tcp://localhost:2375'
  DOCKER_DRIVER: overlay
  KUBE_LATEST_VERSION: 'v1.10.0'
  REGISTRY: https://registry-ur.exigenservices.com/
  REGISTRY_IMAGE: registry-ur.exigenservices.com/swagger-api-address
  IMAGE_NAME: 'address'
  IMAGE_PORT: '8080'
  IMAGE_SERV: '12340'
  ARTIFACT: 'address-api-1.0.0.jar'
  K8S_EXT: 'YES'

stages:
  - build
  - build_image

build:compile:
  stage: build
  script:
    - env
    - mvn clean package sonar:sonar -
Dsonar.host.url=http://sonarqube.dev:9000
    - cp -f ./target/$ARTIFACT /var/tmp
  only:
```

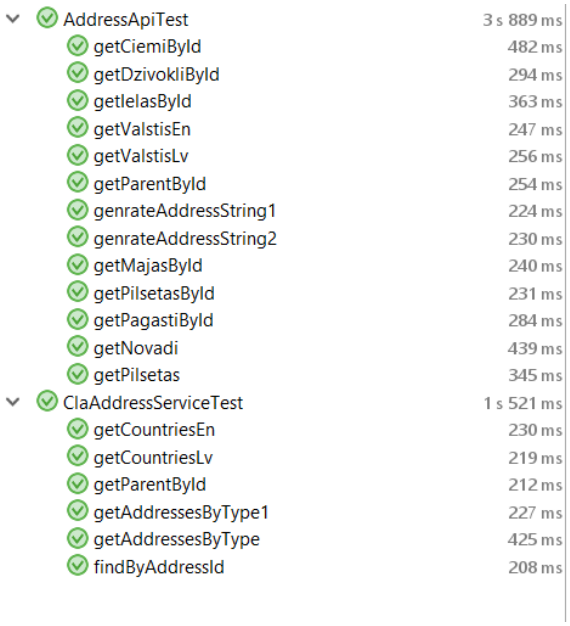
```
    - /^v(?:0|[1-9]\d*)\.(?:0|[1-9]\d*)\.(?:0|[1-9]\d*)\.?(?:0|[1-9]\d*)?$/
    - master

.build_template: &build_image
  stage: build_image
  image: docker:latest
  services:
    - docker:dind
  script:
    - echo $REGISTRY_IMAGE
    - echo $IMAGE_TAG
    - mv -f /var/tmp/$ARTIFACT ./Docker/$ARTIFACT
    - docker login -u rixnexustech -p $REGYSTRY_PASSWORD $REGISTRY
    - cd ./Docker
    - docker build --pull -t $REGISTRY_IMAGE:$IMAGE_TAG .
    - docker push $REGISTRY_IMAGE:$IMAGE_TAG
    - apk add --update ca-certificates
    - apk add --update -t deps curl
    - >
      curl -X POST
      -F token=e0b2f58c6768175d281a32ed2c819c
      -F ref=master
      -F "variables[IMAGE_NAME]=$IMAGE_NAME"
      -F "variables[IMAGE_NAMESPACE]=$IMAGE_NAMESPACE"
      -F "variables[IMAGE_REGISTRY]=$REGISTRY_IMAGE"
      -F "variables[IMAGE_TAG]=$IMAGE_TAG"
      -F "variables[IMAGE_PORT]=$IMAGE_PORT"
      -F "variables[IMAGE_SERV]=$IMAGE_SERV"
      -F "variables[K8S_EXT]=$K8S_EXT"
      https://rixgitlab.r-i.cc/api/v4/projects/54/trigger/pipeline

build-image-dev:
  <<: *build_image
  variables:
    IMAGE_NAMESPACE: 'dev'
    IMAGE_TAG: 'latest'
  only:
    - master

build-image-stage:
  <<: *build_image
  variables:
    IMAGE_NAMESPACE: 'stg'
    IMAGE_TAG: $CI_COMMIT_TAG
  only:
    - /^v(?:0|[1-9]\d*)\.(?:0|[1-9]\d*)\.(?:0|[1-9]\d*)\.?(?:0|[1-9]\d*)?$/
```

Dockerfile:

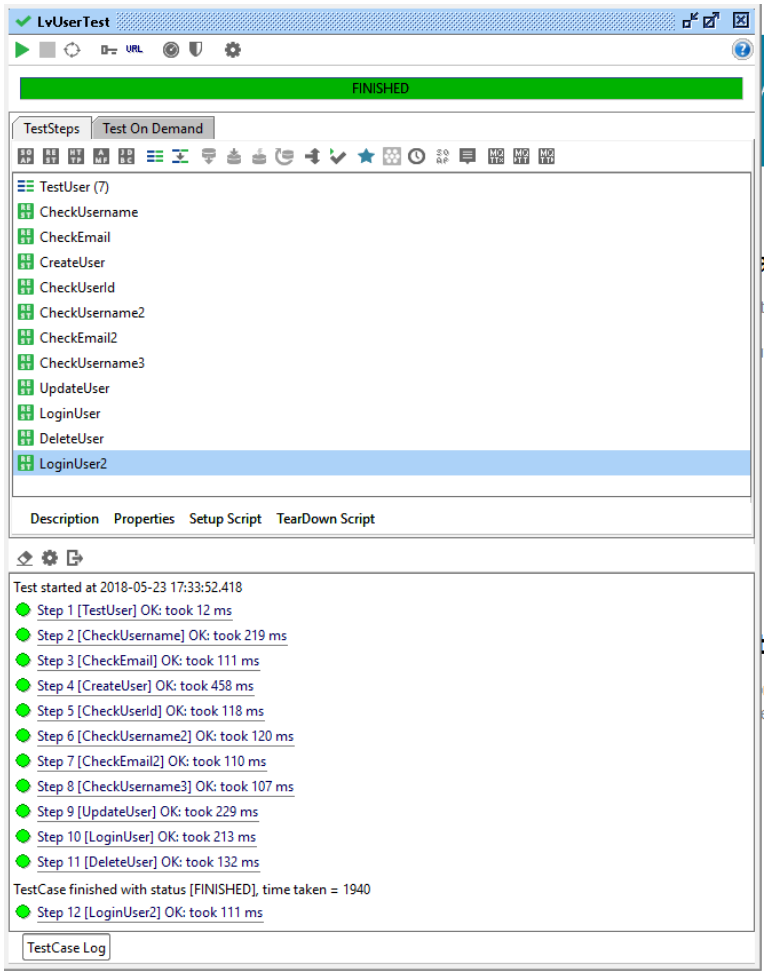
```
FROM openjdk:8-jre-alpine
COPY ./address-api-1.0.0.jar /usr/src/swagger-api/
COPY ./application.properties /usr/src/swagger-api/
COPY ./wait-for.sh /usr/src/swagger-api/
RUN chmod 777 /usr/src/swagger-api/*

WORKDIR /usr/src/swagger-api
EXPOSE 8080
CMD ["java", "-jar", "address-api-1.0.0.jar", "--
spring.config.location=./application.properties"]
```

Testēšanas rezultātu piemēri

Zemāk ir redzami adrešu servisa vienībtestu izpildes rezultāti, izmantojot IDE (integrētā izstrādes vide) iebūvētos rīkus.



Zemāk ir redzami lietotāju servisa testi, izmantojot SoapUI rīku.



Kvalifikācijas darbs „***Uzņēmumu reģistrācijas sistēmas servera puses izstrāde mikroservisu arhitektūrā***” 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: ***Alberts Bērziņš*** **_____ ___.05.2018**.

Rekomendēju darbu aizstāvēšanai
Darba vadītājs: ***M. mat, Valdis Prodnieks*** **_____ ___.05.2018.**

Recenzents: ***Dr.dat. Juris Rāts***

Darbs iesniegts 28.05.2018.
Kvalifikācijas darbu pārbaudījumu komisijas sekretāre: *Darja Solodovņikova*_____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē
___.06.2018. prot. Nr. _____

Komisijas sekretārs(-e): _____