LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

DATORTĪKLU MĀCĪBU PLATFORMAS PĀRVEIDE

BAKALAURA DARBS

Autors: **Ralfs Eduards Braunfelds**
Studenta apliecības Nr.: rb17055
Darba vadītājs: doc., Dr.sc.comp. Leo Trukšāns

RĪGA 2021

ANOTĀCIJA

Šis dokuments ir bakalaura darbs, kurš izstrādāts Latvijas Universitātes Datorikas fakultātes Datorzinātnes studiju programmas ietvaros. Dokumentā tiek veikti pētījumi par datortīklu un datorsistēmu mācību platformas *ReSeLa* pārveidi, kas iekļauj izpēti par populāru tiešsaistes mācību platformu iezīmēm un veicamo uzlabojumu tehnoloģiskajiem risinājumiem. Balstoties uz veiktajiem pētījumiem, tiek veikta sākotnējā jaunās platformas izstrāde, un dokumentā aprakstīta tās struktūra, *API*, izvēlētie risinājumi, trūkumi un nepieciešamie uzlabojumi, kā arī tiek aprakstīta un uzstādīta testa vide tās galvenajai ārējai saskarnei – *OpenStack*.

Atslēgas vārdi: Tiešsaistes mācību platforma, *ReSeLa* pārveide, uzlabojumu risinājumi, *OpenStack*

ABSTRACT

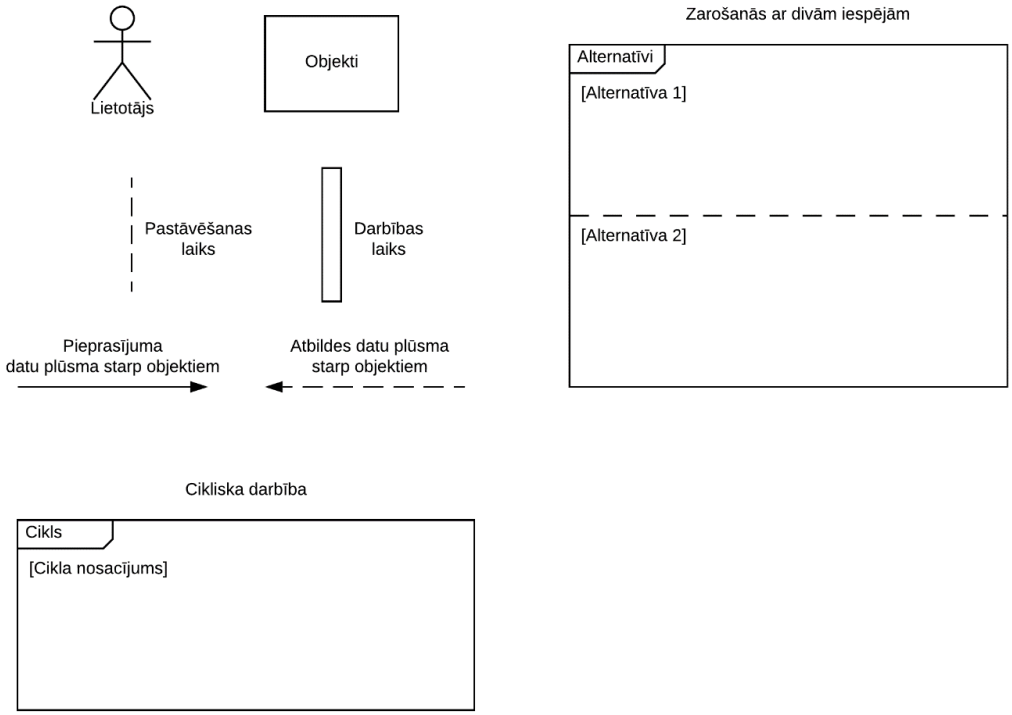
REMAKING OF A COMPUTER NETWORK LEARNING PLATFORM

This document is a Bachelor's Thesis, which has been produced in the scope of the University of Latvia Faculty of Computing Computer Science Bachelor's study programme. This document researches the remaking of the *ReSeLa* computer network learning platform, which includes researching the characteristics of other popular online learning platforms and technological solutions for the required improvements. Initial development of the new learning platform is then done based on research results and its main characteristics are documented, including the overall structure, *API*, used solutions and further development. Furthermore, a description and setup steps are given for the test environment of the platform's main dependency – *OpenStack*.

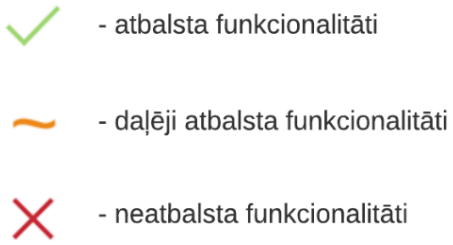
Keywords: online learning platform, *ReSeLa* platform remaking, improvement solutions, *OpenStack*

SATURA RĀDĪTĀJS

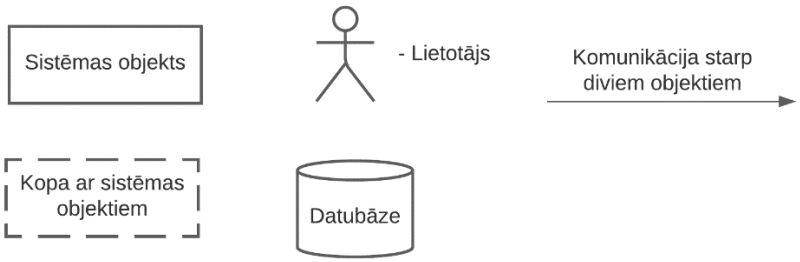
APZĪMĒJUMU SARAKSTS



1. attēls – **Secību diagrammas apzīmējumi**



2. attēls – **Ietvaru funkcionalitāšu atbalsta apzīmējumi**



3. attēls – **Sistēmas projektējumu apzīmējumi**

JĒDZIENI

Serveris – iekārta vai datorprogramma, kas piedāvā datus citām iekārtām caur tīklu.

Klients – iekārta vai datorprogramma, kas pieprasa datus no servera caur tīklu.

Pārlūks – programmatūra, kas spēj pieprasīt un attēlot mājaslapas.

***HTTP* –** protokols datu pārraidei caur tīklu.

***HTTP Header* dati –** *HTTP* savienojuma aprakstoša informācija.

Tīkla lietotne – lietotne, kurai piekļūst caur tīklu, izmantojot programmatūru, kā pārlūku.

Virtualizācija – process, kurā tiek darbināta virtuāla datorsistēma uz fiziskas datorsistēmas.

Virtuālā mašīna – virtualizēta datorsistēma.

Hipervizors – iekārta vai programmatūra, kura spēj izveidot un uzturēt virtuālās mašīnas.

Tīkla topoloģija – grafs, kura virsotnes atbilst tīkla iekārtām un šķautnes tīkla iekārtu savienojumiem.

Maršrutētājs – tīkla iekārta, kas nodrošina tīkla ziņojumu adresēšanu un maršrutēšanu.

Publiskais tīkls – tīkls, kuram var piekļūt jebkurš.

Privātais tīkls – tīkls, kuram var piekļūt tikai autorizēta kopa klientu.

***Proxy* –** tīkla serveris, kurš darbojās kā starpnieks tīkla komunikācijā.

Mākoņa pakalpojums – pakalpojums, kurš nodrošina datortīklu un datorsistēmu resursus pēc pieprasījuma.

***DHCP* –** tīkla administrēšanas protokols, parasti, lai automātiski piešķirtu adreses tīkla iekārtām.

***Load balancer* –** tīkla iekārta, kas veic godīgu tīkla pieprasījumu sadalīšanu starp vairākiem serveriem.

***ReSeLa* –** datortīklu un datorsistēmu tiešsaistes mācību platforma.

***OpenStack* –** mākoņa pakalpojums, kas nodrošina tīkla topoloģiju izveidošanu.

***IaaS* (*Infrastructure as a Service*) –** pakalpojumi, kuri nodrošina tīkla topoloģiju izveidošanu.

***API* (*Application Programming Interface*) –** jebkāda veida saskarne starp divām lietotnēm.

***JSON* –** datu apmaiņas protokols, kas nosaka datu formatēšanas sintaksi.

***JSON Object* –** *JSON* protokola datu tips, kas satur atslēgu un vērtību pārus.

***JSON Array* –** *JSON* protokola datu tips, kas satur sarakstu.

***String* –** datu tips, kas satur simbolu virknes.

***Integer* –** datu tips, kas satur veselus skaitļus.

Programmēšanas bibliotēka – programmēšanas valodas gatavo konstrukciju kopums, kuru var ieviest un izmantot citās programmās.

Programmēšanas ietvars – programmēšanas valodas gatavo konstrukciju un bibliotēku kopums, kuru var ieviest un izmantot citās programmās.

Skriptēšanas valoda – interpretējama programmēšanas valoda.

***PHP* –** skriptēšanas programmēšanas valoda paredzēta vispārīgai lietošanai un tīkla lietotņu izstrādei.

***Python* –** skriptēšanas programmēšanas valoda paredzēta vispārīgai lietošanai.

***JavaScript* –** skriptēšanas programmēšanas valoda, kuru spēj interpretēt tīkla pārlūki.

***React* –** tīkla lietotņu izstrādē izmantots *JavaScript* ietvars.

***HTML* –** valoda, kura tiek izmantota mājaslapu pamatā, lai noteiktu lapas struktūru.

***HTML div* komponente –** *HTML* valodas satura atdalīšanas komponente.

***HTML svg* komponente –** *HTML* valodas komponente, kura satur *SVG* grafiku.

***SVG* –** valoda, lai aprakstītu vektoru grafiku.

***HTML path* komponente –** *HTML* komponente, kura *svg* komponentes ietvaros definē līkni.

***CSS* –** valoda, kura tiek izmantota, lai mainītu *HTML* komponentēm dizainu.

***MikroTik* –** Latviešu uzņēmums, kurš specializējas tīkla iekārtu ražošanā.

***Linux* –** kopa operētājsistēmu, kuri ir bāzēti uz *Linux* operētājsistēmas kodolu.

***YAML* fails –** fails, kurš satur *YAML* formāta datus, kuru parasti izmanto konfigurācijas nolūkos.

***Stack Overflow* –** sadarbības un zināšanu apmaiņu platforma, populāra starp *IT* profesionāļiem.

***GitHub* –** pirmkoda uzturēšanas platforma.

***LDAP* –** protokols lietotāju direktoriju vaicājumu veikšanai. Tipiski izmantots autentifikācijas un autorizācijas nolūkos.

***Oauth2.0* –** autentifikācijas un autorizācijas ietvars, kas ļauj platformas lietotājiem pieslēgties ar citu platformu profiliem.

Datubāzes migrāciju skripts – skripts, kas nodrošina datubāzes struktūras vai datu izmaiņu.

Datubāzes *ER* modelis – datubāzes struktūras apraksts, kas norāda entītijas, to atribūtus un attiecības.

Komandrinda / vadības konsole – teksta bāzēta saskarne ar datoru.

Seriālais ports – ports uz iekārtām, kurš nodrošina asinhronu viena bitu informācijas apmaiņu.

Bezjē līkne – līkne, kurai ir viegli izmainīt formu un bieži tiek lietota datorgrafikā.

IEVADS

Tiešsaistes mācību platformas ir tīkla lietotnes, kurās tiek izmantotas vairākas metodes un tīkla tehnoloģijas, lai nodrošinātu tās lietotājiem kvalitatīvu vidi izglītības nolūkos. Šādās platformās lietotājiem parasti ir iespējas apgūt video vai teksta materiālus, veikt pārbaudes uzdevumus, piemēram, testa veidā, un bieži tiek piedāvāti arī tēmai pielāgoti interaktīvi risinājumi. Labs reprezentatīvs piemērs ir platforma *Chessable*, kur tiek piedāvāti tiešsaistes kursi šaha galda spēles apguvei. Minētajā platformā tiek piedāvāti populāru spēlētāju veidoti teksta un video materiāli, kā arī interaktīvi šaha galdi, kuri tiek izmantoti informācijas pasniegšanai un pārbaudījumu veikšanai, kas nodrošina lielu pievienoto vērtību mācību procesam.

Mūsdienās šādas platformas kļūst aktuālākas līdz ar plašāku Interneta pieejamību, jo mājaslapas var paļauties uz lielāku potenciālo lietotāju skaitu, un skolas var būt drošas, ka visiem studentiem mācību platforma būs pieejama. Papildus tam, tīkla tehnoloģijas ir pietiekami attīstījušās, lai šādas mācību platformas varētu piedāvāt vairākus interesantus interaktīvus risinājumus un sniegt lietotājiem pietiekamu pievienoto vērtību, lai tiešsaistes mācību platformas būtu vērts apsvērt izmantot klātienes apmācību vietā. Ir vērts arī pieminēt, ka mācību platformu nozīmība aug mūsdienu pandēmijas ārkārtas situācijas kontekstā, jo studenti ir spiesti kursus apgūt attālināti, un ir zaudēta iespēja klātienē veikt praktiskus darbus.

Darbā tiek apskatīta datortīklu un datorsistēmu tiešsaistes mācību platforma *ReSeLa* (*Remote Security Labs*), kura pašlaik vairs nav aktīva, bet ir tikusi izmantota Latvijas Universitātes Datorikas fakultātes datortīklu apguves kursos. Šī platforma gan vizuāli, gan funkcionāli ir novecojusi, un šī darba ietvaros tiek pētītas un īstenotas iespējas tā pārveidei.

Darba galvenais avots ir šī paša darba autora izstrādātais kursa darbs [1], kurā tika veikta analīze par trīs populārām mācību platformām – *Khan Academy*, *Chessable* un *Coursera* –, un doti ieteikumi funkcionalitāšu un dizaina iezīmēm, kuras vajadzētu pievienot atjaunotajā *ReSeLa* platformā, lai tā iekļautu modernākās mācību platformu prasības. Papildus, minētajā kursa darbā tika veikta sākotnēja analīze par vairākām izmantotajām tehnoloģijām platformas izstrādē, kuru šis bakalaura darbs paredz turpināt. Citi darbi par *ReSeLa* platformu iekļauj Ievas Lapiņas kvalifikācijas darbu [2] par virtuālu laboratoriju izveidošanu, izmantojot *OpenStack*, un bakalaura darbu [3], kurā tika veikta analīze par spēļu elementiem un to pievienošanu atjaunotajā *ReSeLa* platformā.

Darba nolūks:

- Sniegt pamatojumu pārveidotās *ReSeLa* platformas izmantotajiem risinājumiem;
- Dot lasītājiem informāciju par pieejamajām tehnoloģijām un rīkiem, kas varētu noderēt jebkuras mūsdienīgas tīkla lietotnes izstrādē.

Darba mērķi:

- Noskaidrot *ReSeLa* nepieciešamos uzlabojumus un apskatīt iespējas to ieviešanai;
- Iesākt jaunās *ReSeLa* platformas izstrādi, balstoties uz veiktajiem ieteikumiem šajā dokumentā.

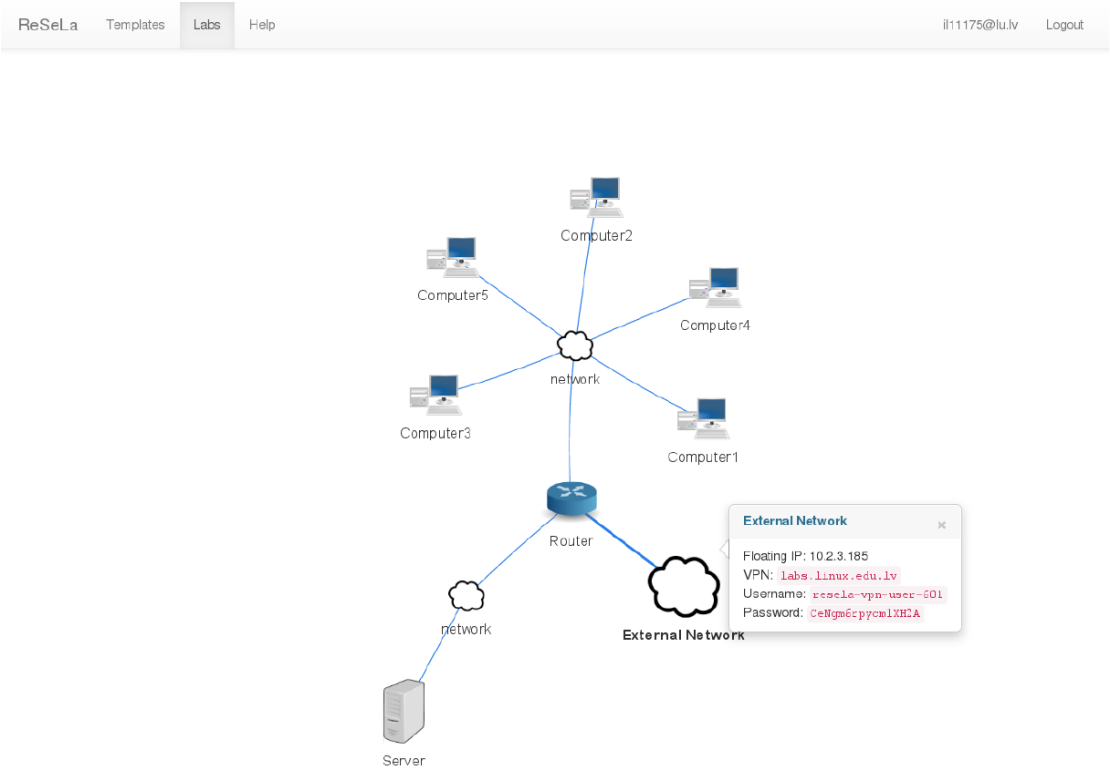
Darba uzdevumi, kuri tiks veikti turpmākajās nodaļās:

- Aprakstīt pašreizējās *ReSeLa* platformas iezīmes;
- Noteikt pašreizējās *ReSeLa* platformas nepieciešamos uzlabojumus;
- Veikt pētījumus par tehnoloģiskajiem risinājumiem uzlabojumu veikšanai;
- Aprakstīt testa vides uzstādīšanu *ReSeLa* platformas galvenajai ārējai saskarnei, *OpenStack*;
- Aprakstīt iesāktās jaunās platformas projektējumu, dizainu un izmantotos risinājumus;
- Aprakstīt iesāktās jaunās platformas *API*;
- Apkopot darbā iegūtos rezultātus;
- Veikt secinājumus par jaunās platformas iespējām un nākamajiem soļiem.

1. *ReSeLa* MĀCĪBU PLATFORMAS APRAKSTS

ReSeLa ir *PHP* skriptēšanas programmēšanas valodā izstrādāta tiešsaistes mācību platforma datorsistēmu un datortīklu apguvei, kura ir tikusi izmantota Latvijas Universitātes Datorikas fakultātes ietvaros, bet pašlaik vairs nav aktīva. Sīkākai informācijai par pašreizējās *ReSeLa* platformas struktūru un darbību skatīt Ievas Lapiņas kvalifikācijas darbu [2] par virtuālu laboratoriju izveidi *ReSeLa* vidē un bakalaura darba 4. nodaļu [3] par spēļu elementu pievienošanu mācību platformās.

Galvenās komponentes šajā platformā ir laboratorijas darbi, kuros studentiem ir dota tīkla topoloģija (skat. 1.1. attēlu), kura sastāv no tīkla savienojumiem starp vairākām tīkla iekārtām, kā datoriem, serveriem, maršrutētājiem un publiskajiem tīkliem.



1.1. attēls – **Laboratorijas skats ar redzamu tīkla topoloģiju** [3]

Platforma laboratorijas darbu topoloģijas realizē ar virtuālu tīklu un virtuālām iekārtām un dod iespēju iekārtas attālināti konfigurēt, izmantojot platformā iekļautu komandrindas logu vai sniedzot lietotājiem informāciju pašrocīgai savienojumu veikšanai. Studentu mērķis ir veikt virtuālo iekārtu attālinātu konfigurēšanu atbilstoši laboratorijas darba uzdevumu prasībām un iegūt pēc iespējas lielāku punktu skaitu. Laboratorijas darbu virtuālās topoloģijas izveidošanai tiek

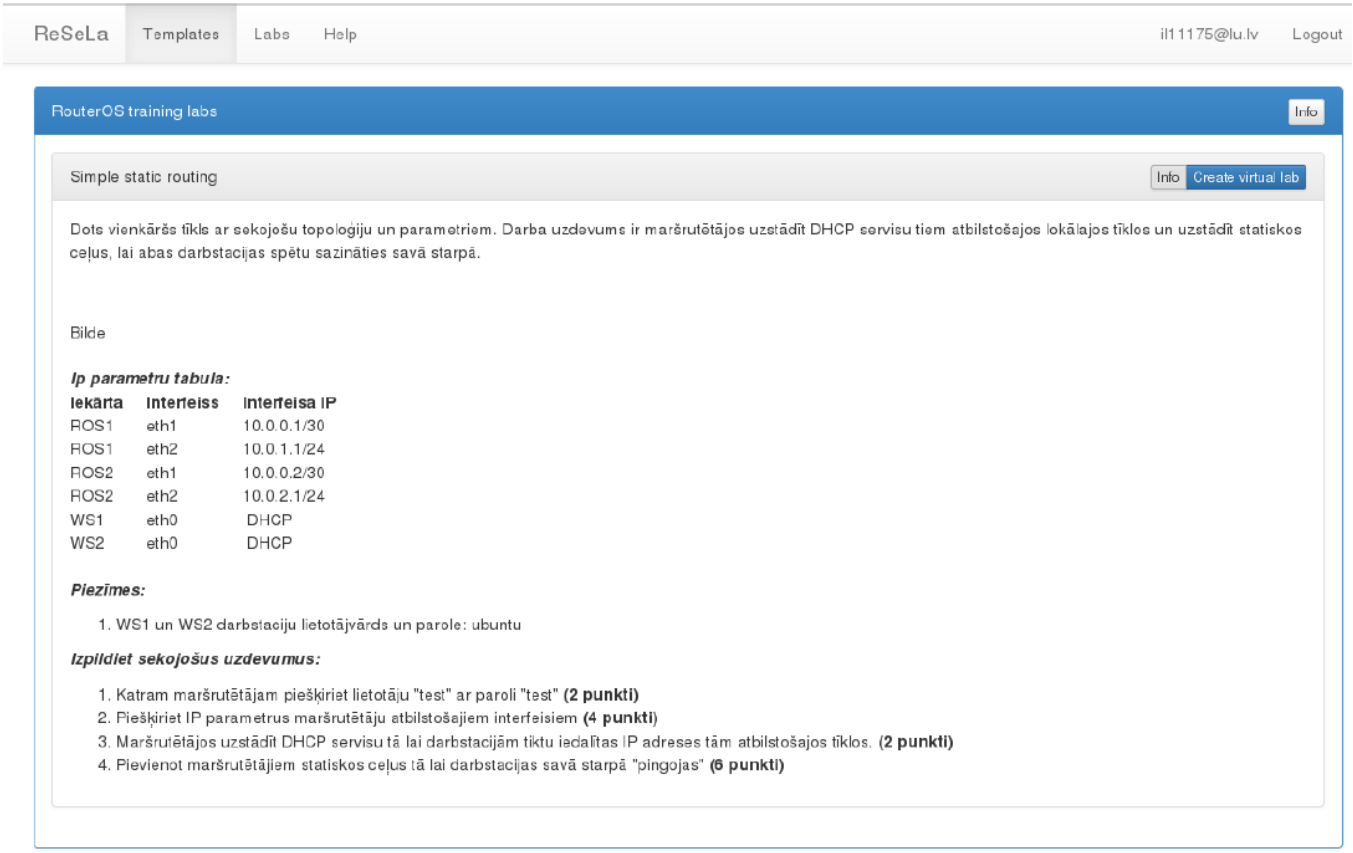
izmantots *OpenStack* mākoņa pakalpojums, kas piedāvā infrastruktūru kā pakalpojumu jeb *IaaS* (*Infrastructure as a Service*). Sīkāk par *OpenStack* skatīt 3. nodaļu, kur tiek veikta tās testa vides uzstādīšana.

Platformā ir pieejamas 3 lietotāju grupas – studenti, pasniedzēji un administratori –, kur administratoriem ir iespēja veidot jaunus kursus, kursu laboratorijas darbus (skat. 1.2. attēlu), lietotājus, piešķirt tiem pieeju laboratorijas darbiem un pievienot pasniedzējam uzraugāmus studentus.

1.2. attēls – **Laboratorijas izveidošana *ReSeLa* sistēmā** [2]

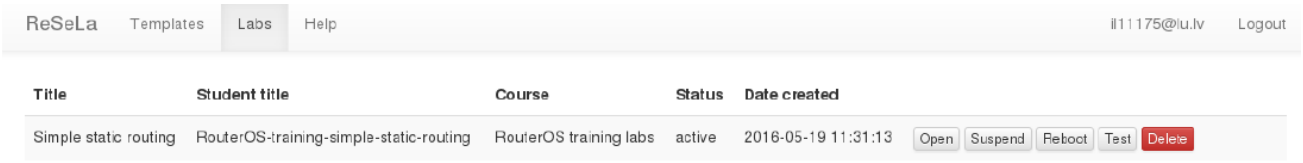
Pasniedzējiem ir iespējas apskatīt sev piesaistītos studentus, viņu laboratorijas darbu uzdevumu vērtējumus un piekļūt studentu virtuālajām iekārtām, lai apskatītu veiktās konfigurācijas.

Sadaļā "*Templates*" studentiem ir iespēja platformā aplūkot sev pieejamos kursus, to laboratorijas darbus un to aprakstus, kā arī pieprasīt laboratorijas darbu izveidi jeb topoloģiju realizēšanu *OpenStack* pakalpojumā. (skat. 1.3. attēlu).



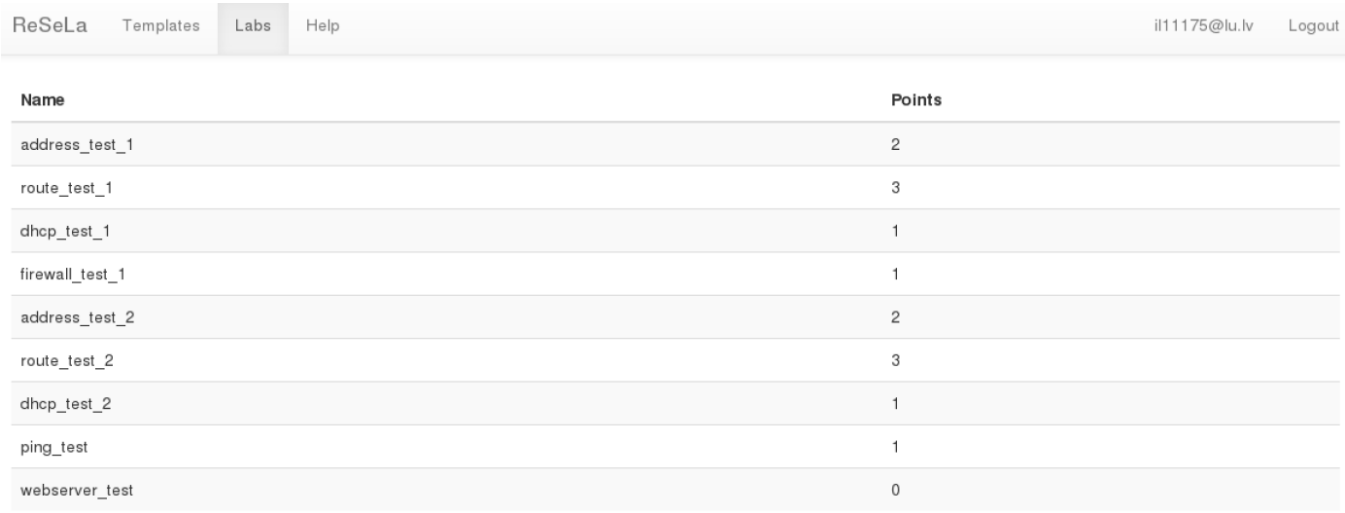
1.3. attēls – **Lietotāja skats ar laboratorijas aprakstu** [2]

Pēc laboratorijas darbu izveidošanas, students tos var aplūkot sadaļā "*Labs*", kur ir iespējams izdzēst, apturēt vai atsākt virtuālās topoloģijas, veikt virtuālo iekārtu konfigurēšanu un pieprasīt laboratorijas darbu uzdevumu automātisko testēšanu (skat. 1.4. attēlu). Katram studentam var tikt vienlaikus būt realizēta tikai viena laboratorijas darba topoloģija, lai ietaupītu platformas resursus.



1.4. attēls – **Lietotāja skats ar izveidotajām laboratorijām** [2]

Pēc automātiskās testēšanas pieprasīšanas, studentam ir iespēja apskatīt testa rezultātus (skat. 1.5. attēlu), kur tiek uzrādīti skaitliski vērtējumi katram no laboratorijas darba uzdevumiem.



1.5. attēls – **Lietotāja skats ar testa rezultātiem** [2]

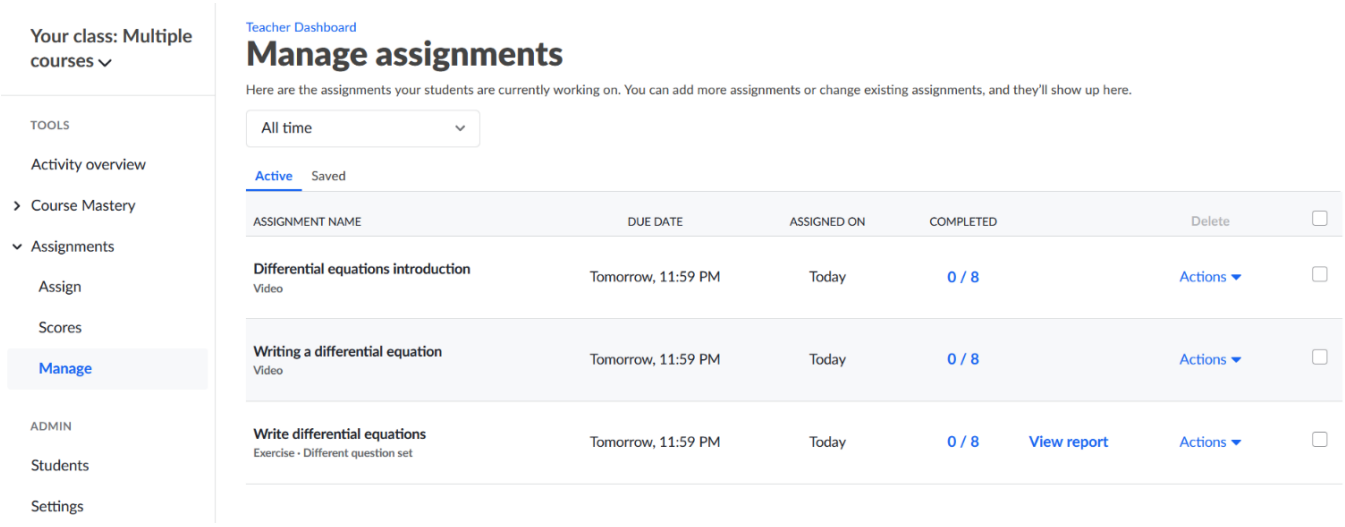
2. *ReSeLa* MĀCĪBU PLATFORMAS VEICAMIE UZLABOJUMI

Šajā nodaļā tiek aprakstīti visi veicamie uzlabojumi pašreizējai *ReSeLa* platformai, balstoties uz pašreizējās platformas iezīmēm un citu populāru mācību platformu analīzi, kā arī tiek veikti pētījumi uzlabojumu iespējamajiem risinājumiem.

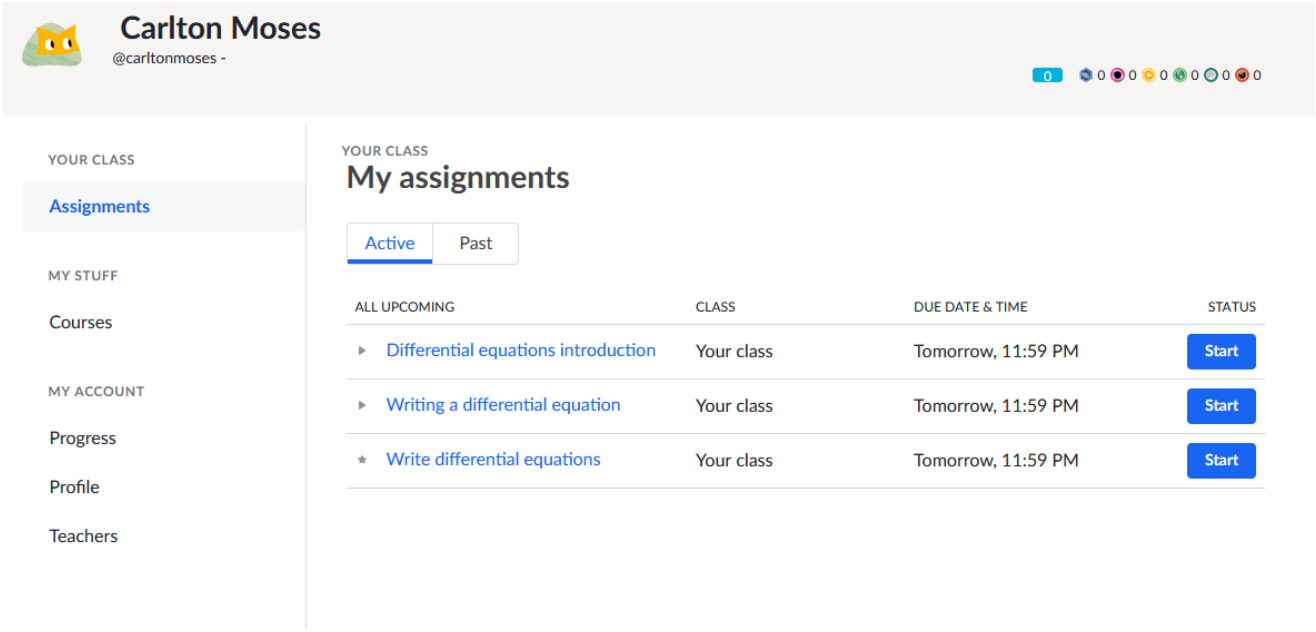
2.1. Dizains un funkcionalitātes

Šajā nodaļā tiek aprakstīti galvenie rezultāti un secinājumi no kursa darbā [1] veiktās analīzes par populāru mācību platformu – *Khan Academy*, *Chessable*, *Coursera* – iezīmēm. Secinājumi rāda, ka pašreizējās *ReSeLa* platformas saturs ir izkārtots līdzīgi un satur visas analīzes laikā sastaptās lietotāju grupas – students, pasniedzējs un administrators. Neskatoties uz to, ir vairākas citas dizaina iezīmes, kuras *ReSeLa* platforma nesatur, bet ir pieejamas analizētajās mācību platformās.

Attēlos 2.1 un 2.2 ir redzams, ka citās mācību platformās, kā *Khan Academy*, ir plašākas iespējas pasniedzēju un studentu sadarbībai, kur pasniedzējiem ir iespējas veidot savas virtuālās klases, kurās var pievienot citus lietotājus kā studentus, uzdot kursu materiālus ar noteiktiem izpildes termiņiem un apskatīt savu studentu rezultātus uzdevumos. Virtuālās klases ir pieejamas kā atsevišķs skats un pastāv neatkarīgi no pamata platformas funkcionalitātes, kur lietotāji var veikt kursu un uzdevumu izpildi ārpus virtuālo klašu ietvara.

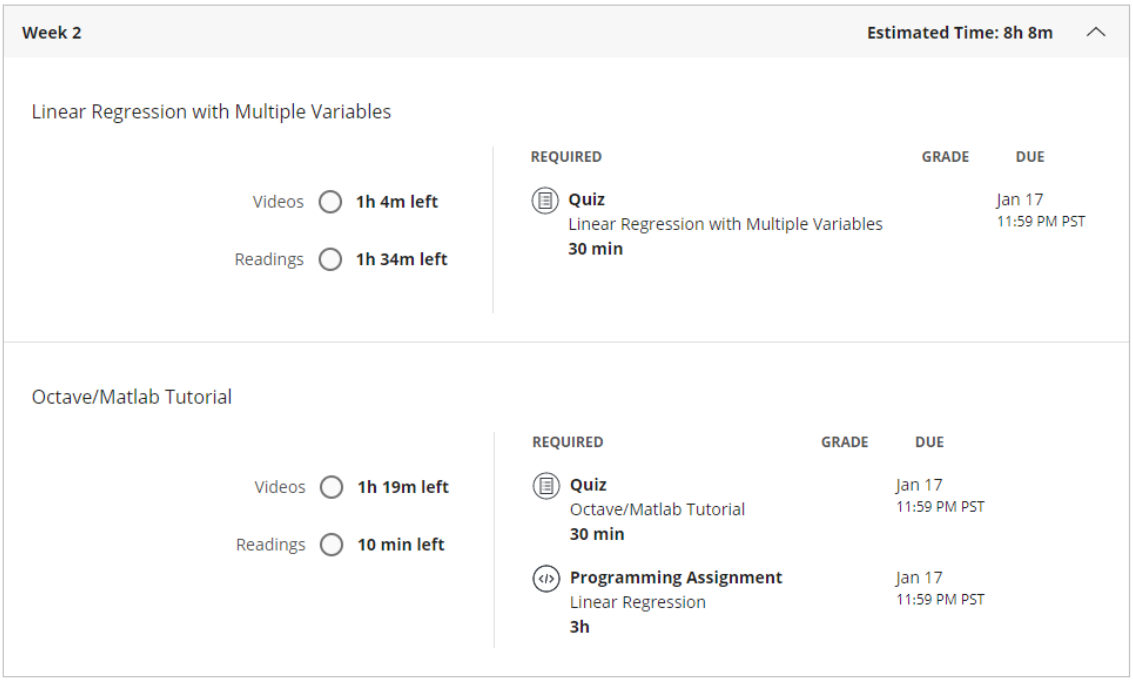


2.1. attēls – ***Khan Academy* virtuālās klases pasniedzēja skats**



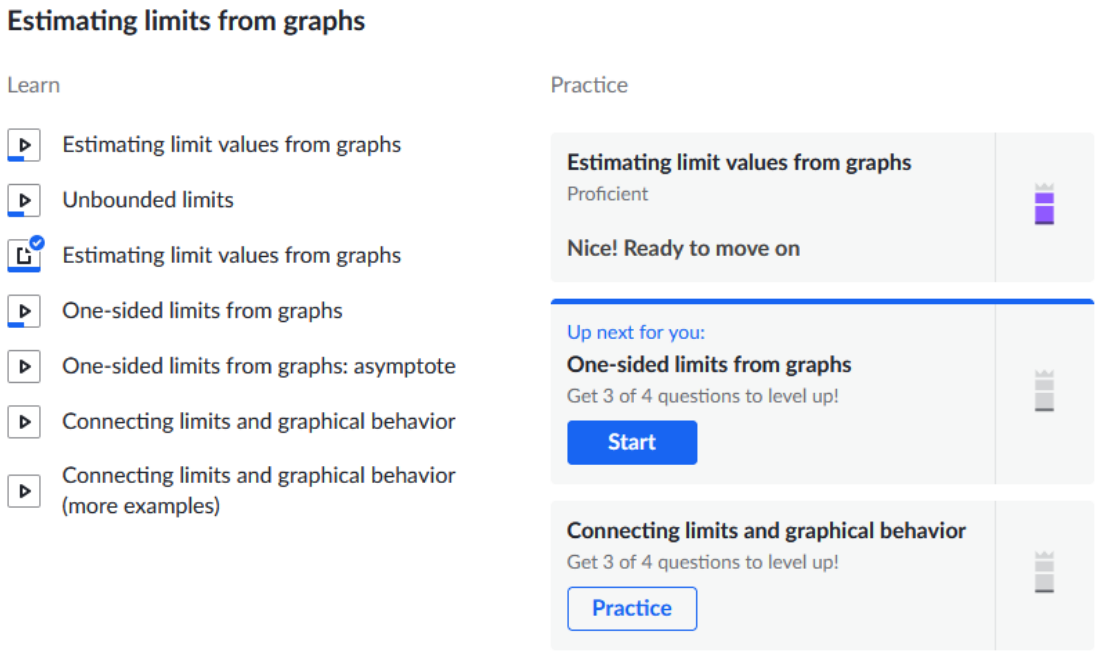
2.2. attēls – ***Khan Academy*** **virtuālās klases studenta skats**

Kursu materiālu daudzveidība citās mācību platformās ir daudz plašāka un iekļauj lasāmvielu, video materiālus, testus un cita veidu uzdevumus, kā programmēšanas uzdevumus (skat. 2.3. attēlu). Pašreizējā *ReSeLa* platforma satur tikai laboratorijas darbus, un ir redzama iespēja pievienot pārējos materiālu veidus.



2.3. attēls – ***Coursera*** **materiālu daudzveidība**

Visās analizētajās platformās tika saskatīta tendence ar skaidriem vizuāliem elementiem lietotāju informēt par pašreizējo progresu kursos un laboratorijas darbos (skat. 2.4. attēlu), bet *ReSeLa* pašlaik ir pieejami tikai skaitliski testa rezultāti kā progresa indikatori.

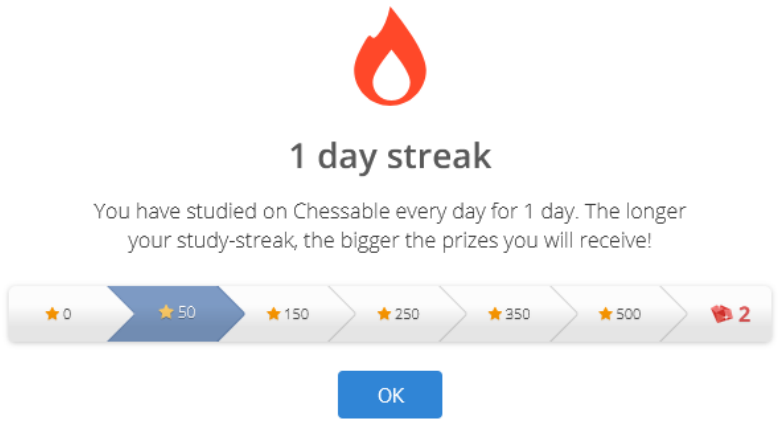


2.4. attēls – ***Khan Academy*** **kursa tēmu progresa indikatori**

2.1.1. Spēļu elementi

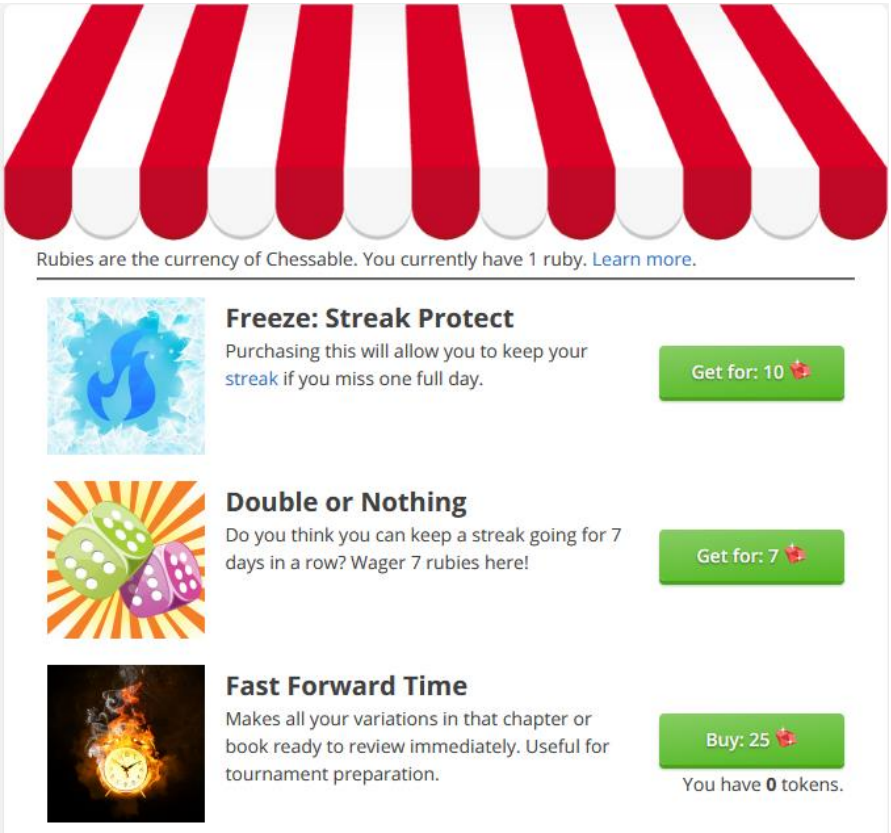
Viena no galvenajām dizaina iezīmēm, kas tika saskatīta visās analizētajās mācību platformās, kuras saturēja mazāk formālus kursus, ir *gamification* jeb spēļu elementu pievienošana mācību procesam, lai palielinātu lietotāju motivāciju kursu apguvei [1, 3]. *ReSeLa* pašlaik nekādu spēļu vai citu motivējošu elementu iekļaušana nav sastopama.

Spēļu elementi parasti izpaužas kā lietotāja uzslavēšana un punktu piešķiršana, kurus lietotājs var nopelnīt, platformu apmeklējot katru dienu (skat. 2.5. attēlu) vai par mācību procesa aktivitāšu paveikšanu, kā video noskatīšanos un pārbaudes uzdevumu veiksmīgu izpildīšanu.



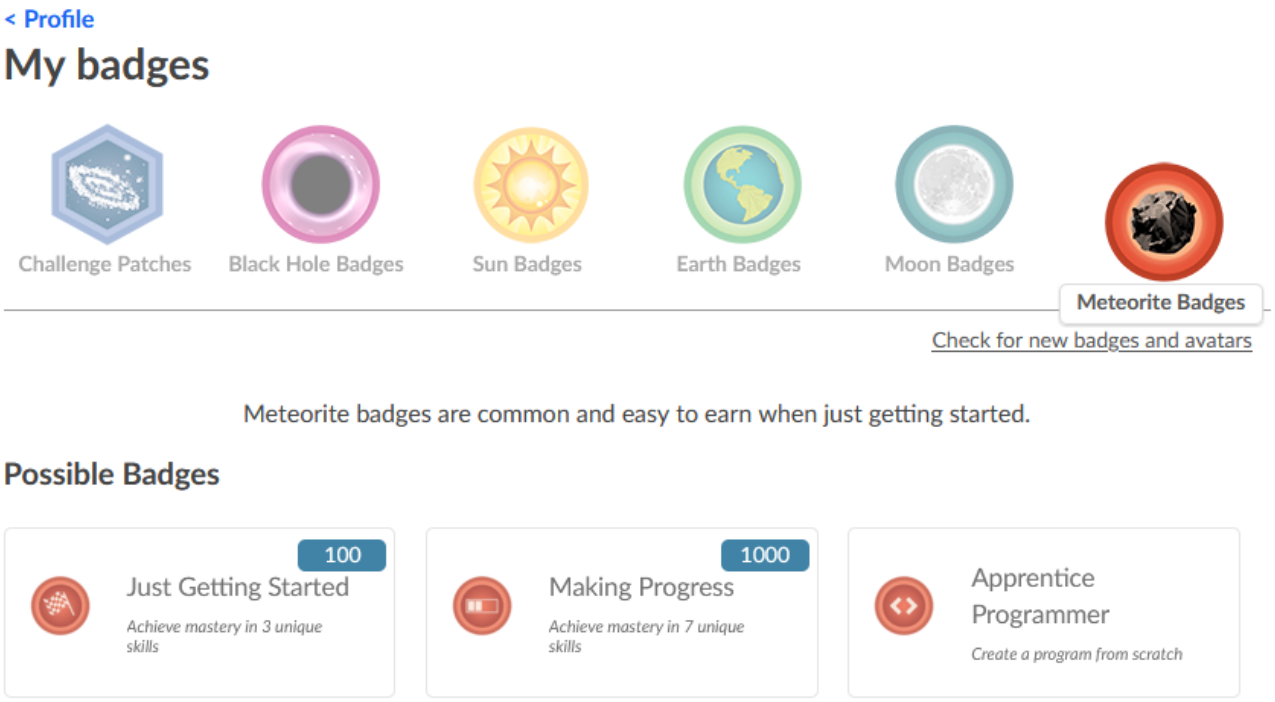
2.5. attēls – ***Chessable*** **punkti par platformas ikdienas apmeklējumu** [1]

Dažās platformās, kā *Khan Academy*, šie punkti ir tikai kolekcionēšanas nolūkos, bet citās platformās, kā *Chessable*, punktu skaits nosaka lietotāja līmeni, un ar tiem ir iespēja iegādāties uzlabojumus profilam, kā pieeju pašlaik lietotājam slēgtiem materiāliem (skat. 2.6. attēlu).



2.6. attēls – ***Chessable*** **punktu tērēšanas iespējas** [1]

Vēl viens populārs spēļu elements ir medaļas (skat. 2.7. attēlu), kuras parasti ir sadalītas grūtības pakāpēs un, kuras lietotājs var nopelnīt par noteiktu aktivitāšu paveikšanu, kā, piemēram, noteiktu kopu kursu pabeigšanu vai 100 kopēju stundu pavadīšanu, skatoties video materiālus. Attēlā 2.7 ir redzams, ka, lai vairāk motivētu lietotājus nopelnīt medaļas, tiek lietoti vairāki interesanti nosaukumi, kā *Black Hole* medaļu grūtības pakāpe, kā arī dažu medaļu nosaukumi attēlo statusu, kā *Apprentice Programmer* jeb programmēšanas māceklis, kuru lietotājs ir sasniedzis līdz ar medaļas nopelnīšanu.



2.7. attēls – ***Khan Academy* nopelnāmas medaļas** [1]

2.1.2. Secinājumi par dizainu un funkcionalitātēm

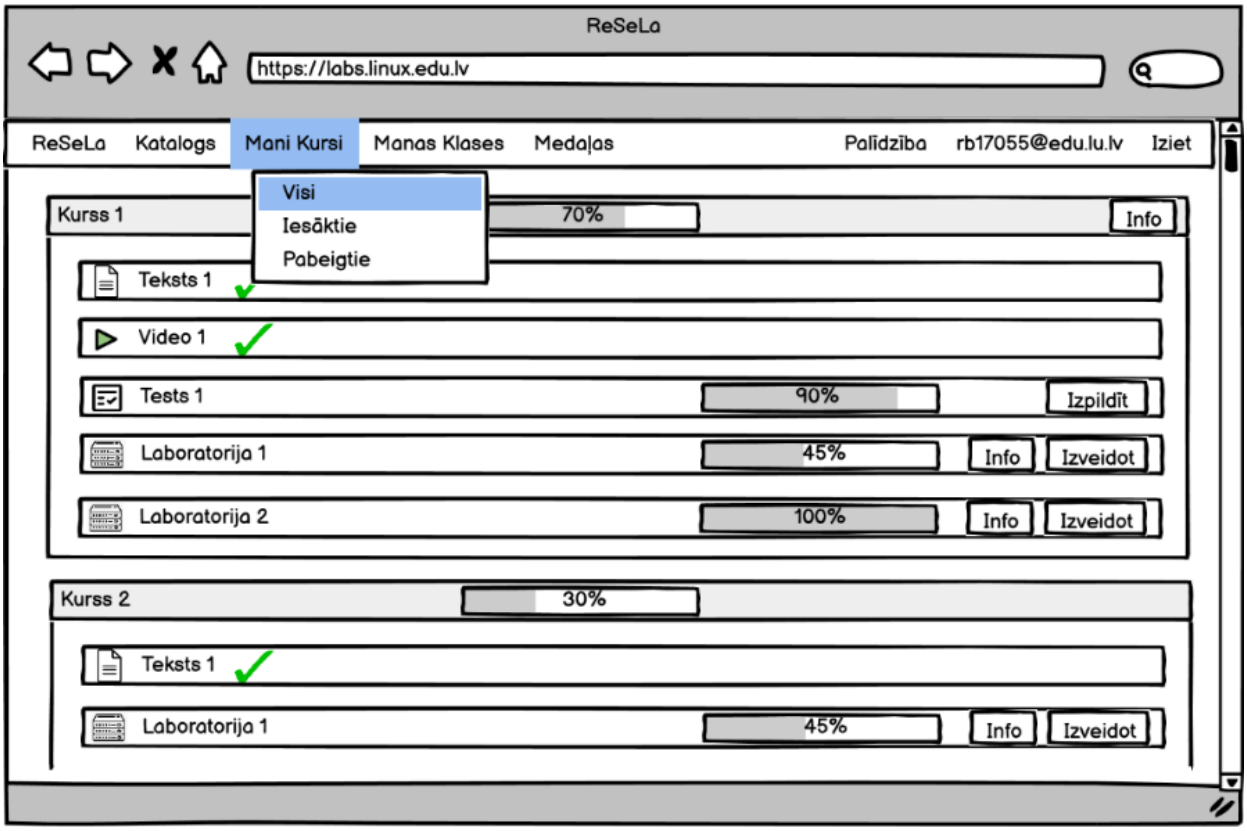
Balstoties uz augstāk minētajiem pašreizējās *ReSeLa* platformas dizaina un funkcionalitātes trūkumiem, kursa darbā [1] tika veikts atjaunotās platformas projektējums (attēli 2.8 līdz 2.11), lai virzītu pārveidotās platformas izstrādi. Lai saskaņotos ar iepriekšējiem pētījumiem, tika turpināti Ievas Lapiņas bakalaura darbā [3] iekļautie dizaina un funkcionalitāšu projektējumi (attēli 2.8 un 2.11), lai pievienotu spēļu elementus *ReSeLa* platformai. Visi zemāk redzamie attēli ir tikai skices un neattēlo gala produkta dizainu un visas iekļautās funkcionalitātes.

Ievas Lapiņas bakalaura darbā [3] tika izstrādāts datortīklu un datorsistēmu kursu mācību plāns, kur koka veidā tiek attēlotas visas Latvijas Universitātes datortīklu administrēšanas kursos

apskatītās tēmas, kā, piemēram, *MikroTik* maršrutētāju konfigurēšana un *IPv6* tehnoloģijas. Attēlā 2.8 ir redzams projektētais kataloga jeb visu pieejamo kursu un laboratorijas darbu skats, kur izveidotais kokveida mācību plāns var kalpot kā spēļu elements, kur lietotājam visi kursi ir redzami, bet, lai tos varētu iesākt, ir nepieciešams izpildīt visus iepriekšējos kursus attiecīgajā koka zarā. Šāds spēļu elements nodrošinātu skaidru progresa indikatoru, un iecere ir motivēt lietotājus sasniegt kursus, kuri atrodas zemāk kokā.

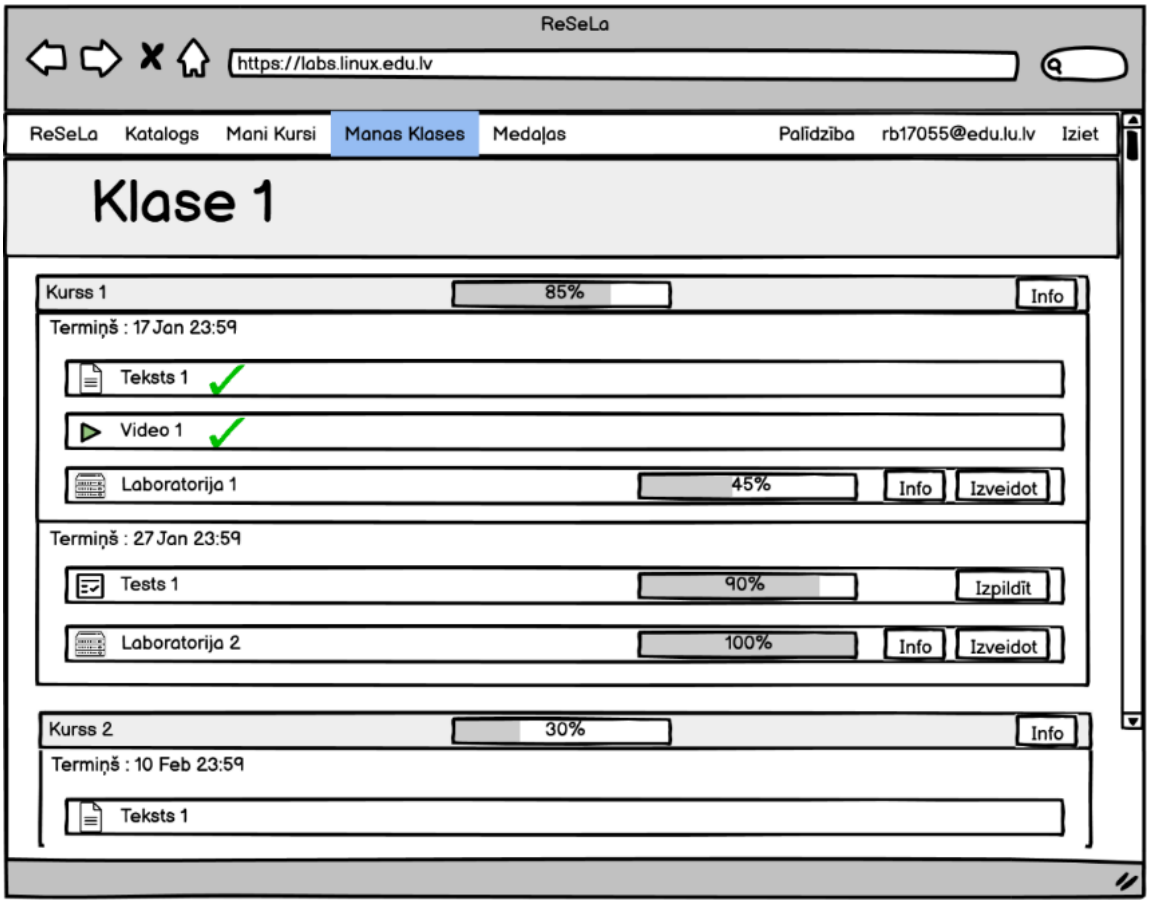
2.8. attēls – **Plānotais pieejamo kursu skats** [3]

Attēlā 2.9 ir redzams projektētais lietotāja iesākto laboratorijas darbu skats, kurš satur informāciju par lietotāja iesāktajiem un pabeigtajiem kursu materiāliem. Katram kursam un tā materiālam ir pievienota vizuāla komponente, lai skaidri attēlotu tā progresu ar zaļu ķeksi vai progresa indikatoru. Kursiem ir pievienoti vairāku veidu materiāli, kā lasāmviela, video, testi un laboratorijas darbi.



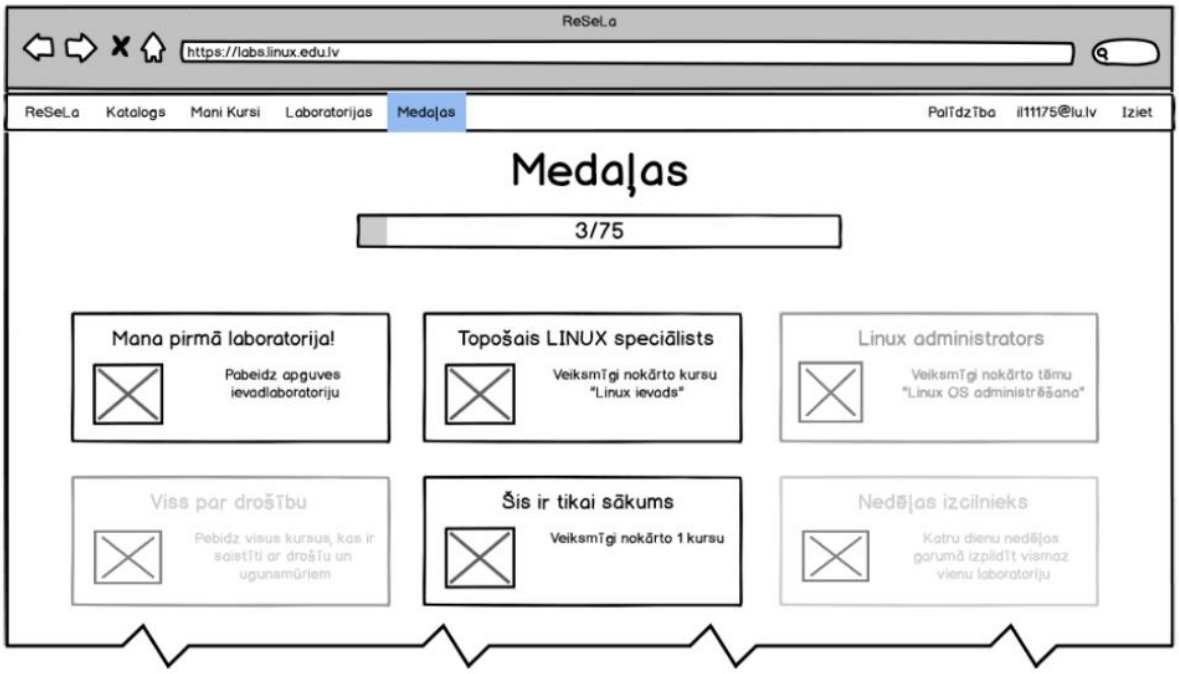
2.9. attēls – **Plānotais lietotāja iesākto laboratorijas darbu skats** [1]

Attēlā 2.10 ir redzams projektētais virtuālo klašu skats, kur pasniedzējiem ir iespēja veidot savas klases, pievienot tiem studentus, kursus un materiālus no visiem platformā pieejamajiem, noteikt to izpildes termiņus, apskatīt pievienoto studentu vērtējumus un piekļūt laboratorijas darbos izveidotajām virtuālajām iekārtām, lai apskatītu studentu veiktās konfigurācijas.



2.10. attēls – **Plānotais studenta skats klasē** [1]

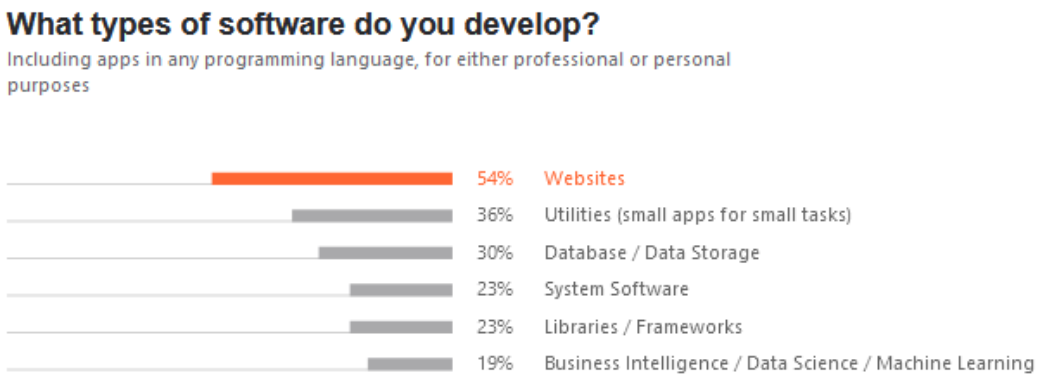
Kā papildus spēļu elementu, *ReSeLa* platformai ir ieteikts pievienot nopelnāmo medaļu sadaļu (skat. 2.11. attēlu), kuras lietotājs var nopelnīt, sasniedzot noteiktus priekšnosacījumus, kā veiksmīgu visu laboratorijas darbu izpildīšanu viena kursa ietvaros vai vienas nedēļas ietvaros katru dienu izpildīts vismaz viens kursa materiāls. Lai vairāk motivētu lietotājus medaļas nopelnīt, to nosaukumi ir veidoti ar spēles un izaicinājuma noskaņojumu, kur dažas medaļas attēlo statusu, kā "Topošais *Linux* speciālists", kuru lietotājs ir sasniedzis līdz ar attiecīgās medaļas nopelnīšanu.



2.11. attēls – **Plānotais nopelnāmo medaļu skats** [3]

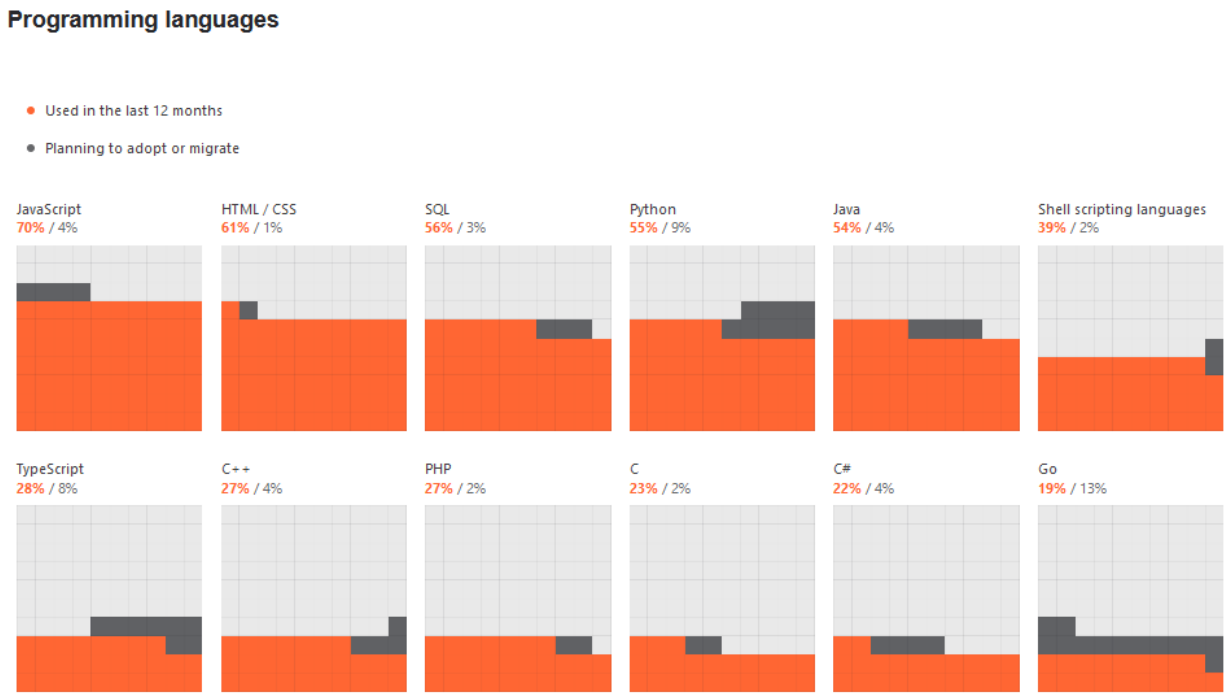
2.2. Platformas izstrādes programmēšanas valoda un ietvari

Pašreizējā *ReSeLa* platforma ir izstrādāta, izmantojot *PHP* skriptēšanas programmēšanas valodu, bet, līdz ar šīs platformas pārveidi, ir iespēja veikt pētījumus un secinājumus par mūsdienās aktuālāku programmēšanas valodu izvēli. Kā redzams 2.12. un 2.13. attēlos, pēc *Jetbrains The State of Developer Ecosystem 2020* aptaujas [4], mājaslapu izstrādātāji aptaujā ir vispopulārākie jeb 54% no visiem respondentiem, bet tikai 27% no visiem respondentiem ir izmantojuši *PHP* programmēšanas valodu pēdējos 12 mēnešos, kas jau ir indikators, ka mūsdienās tīkla lietotnēm tiek izmantotas citas tehnoloģijas.



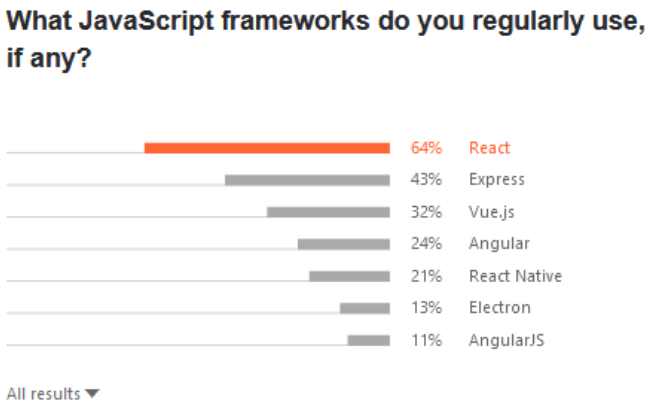
2.12. attēls – ***JetBrains* aptaujas rezultāti par programmatūras tipu** [4]

Attēlā 2.13 ir arī redzams, ka populārākā programmēšanas valoda virs *PHP*, kura tiek izmantota tīkla lietotņu izstrādē, ir *JavaScript*, tāpēc ir vērts apskatīt sīkāk *JavaScript* pielietotos ietvarus



2.13. attēls – ***JetBrains* aptaujas rezultāti par programmēšanas valodām** [4]

Attēlā 2.14. [5] ir redzams, ka populārākie *JavaScript* ietvari ir *React* (64%), *Express* (43%) un *Vue.js* (32%), kur abi *React* un *Vue.js* pielieto *single-page app* jeb vienu-lapu lietotnes pieeju [6].



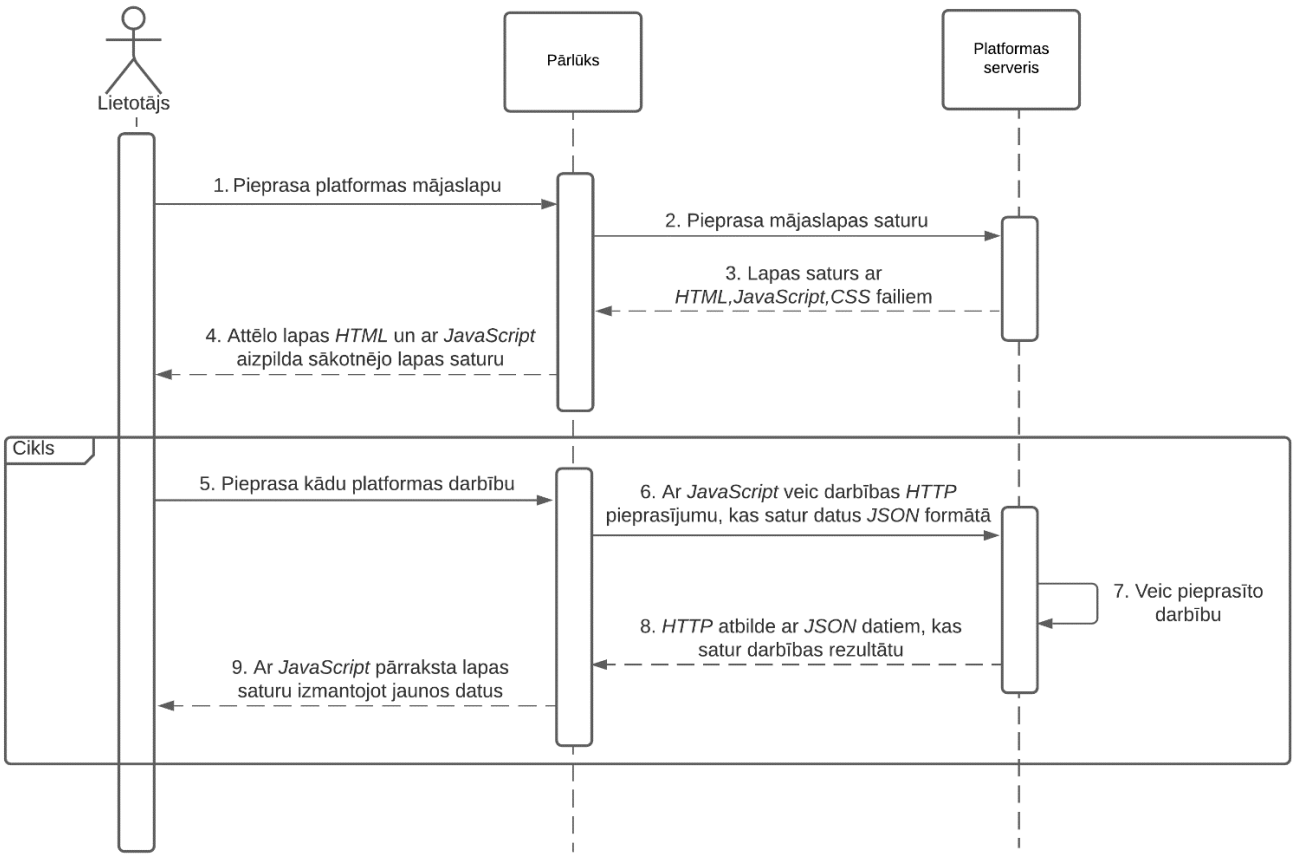
2.14. attēls – ***JetBrains* aptaujas rezultāti par *JavaScript* izstrādes ietvariem** [5]

Papildus, pēc līdzīgas aptaujas *React* bija populārākais *JavaScript* ietvars arī pagājušajā gadā, un šajā gadā tā popularitāte ir pieaugusi par 10% [5] jeb ir pamatojums, ka *React* kļūst par tīkla lietotņu izstrādātāju iecienītāko ietvaru. Arī kursa darbā [1] veiktie pētījumi liecināja, ka populāras mācību platformas, kā *Khan Academy* un *Chessable*, darbu sludinājumos platformas attīstībai tiek meklēti izstrādātāji, kuri pārzin tieši *React* ietvaru. Šo iemeslu dēļ, ir vērts apskatīt vienu-lapu pieejas piedāvātos uzlabojumus un izvērtēt to devumu jaunās platformas izstrādē.

Aptauja arī uzrādīja, ka *Express* ir ļoti populārs *JavaScript* ietvars, bet tas ir paredzēts serveru izstrādei, balstoties uz *Node.js* tehnoloģiju, un 2.2.2. nodaļā ir pamatota citas programmēšanas valodas izmantošana servera izstrādē.

2.2.1. Vienu-lapu lietotne

Vienu-lapu lietotņu pieejas galvenā īpašība [6, 7] ir, ka serveru un klientu puses lietotnes ir skaidri atdalītas (skat. 2.15. attēla secības diagrammu) jeb lietotāja pārlūks paredz no servera saņemt nepieciešamos *HTML, JavaScript* un *CSS* failus tikai vienu reizi lapas apmeklēšanas brīdī.



2.15. attēls – **Vienu lapu lietotņu darbības secību diagramma**

Saņemtais *HTML* dokuments parasti ir minimāls un satur tikai vienu tukšu *div* elementu [8], bet tā saturs visa lietotāja sesijas laikā, atbilstoši lietotāja darbībām, tiek pārrakstīts, izmantojot *JavaScript*, kas visu nepieciešamo informāciju satura izveidošanai iegūst no servera caur *HTTP API* izsaukumiem, kura pieprasījumu un atbildes dati parasti ir *JSON* strukturēti dati. Ir iespējams arī veidot jauktu variantu, kur dažas no lietotāja darbībām pieprasīs jaunu lapu, bet dažas tikai datus, lai atjaunotu pašreizējo skatu ar *JavaScript*.

Šādai pieejai ir vairāki ieguvumi [6, 7], kur galvenais ir lapas satura izmainīšana bez lapas atkārtotas ielādes, jo *JavaScript* ir spējīgs reālā laikā lapas saturu pārrakstīt ar jauniem datiem no servera. Papildus, mājaslapās ir daudzas komponentes, kā navigācijas komponente, kuras mainās reti, un ar *JavaScript* ir iespējams reālā laikā pārrakstīt tikai tās komponentes, kuras tajā brīdī ir nepieciešams. Šādi mājaslapas uzlabo lietotāja pieredzi, jo tās izskatās ātrākas un vairāk kā atsevišķas datorprogrammas, jo kāda veida saturs vienmēr tiek lietotājam rādīts, kaut arī fonā tiek gaidīta atbilde no servera par jauniem datiem.

Liela daļa no skaitļošanas, piemēram, lietotāja skata izveidošana, tiek pārcelta no centralizēta servera uz katra klienta pārlūku, un serverim nav visiem klientu pieprasījumiem jāatbild ar apjomīgiem *HTML, JavaScript* un *CSS* failiem, bet tikai ar *JSON* strukturētu informāciju. Abi šie faktori nozīmē, ka serveris spēs ātrāk apstrādāt klientu pieprasījumus un ķerties klāt nākamajiem, ietaupot servera resursus.

Pastāv arī vairāki trūkumi [7] vienu-lapu pieejai, kur galvenie ir sākotnējās lapas ielādes laiks, jo sākumā tiek saņemts *JavaScript* pirmkods, lai spētu apstrādāt visas lietotnes darbības, un izstrādes sarežģītība, jo ir nepieciešams apgūt ietvara izmantošanu, kur ietvari, kā *React*, bieži paļaujas uz daudzām trešo pušu atbalsta bibliotēkām un katrai ir nepieciešams apgūt tās darbību. Viens piemērs ir tīkla lapu adreses mainīšana un atgriešanās uz iepriekšējo lapu, kur, bez papildus bibliotēku, kā *React Router,* izmantošanas, šīs darbības nav iespējamas. Papildus, veicot lietotnes izstrādi pēc šīs pieejas, ir jāpievērš lielāka uzmanība drošībai, jo daļa no skaitļošanas ir pārnesta no droša un uzticama centrāla servera uz neuzticama klienta pārlūku, un ir nepieciešams aizsargāt servera *API*, nevis katru lapu atsevišķi, kā tas ir klasiskajā izstrādes pieejā.

Pārveidotajai *ReSeLa* platformai būtu daudzi ieguvumi no vienu-lapu pieejas, jo, balstoties uz 2.1.2. nodaļā veiktajiem projektējumiem, ir redzams, ka vairāki dati atkārtojas, piemēram, kursu un laboratorijas darbu nosaukumi, ko ar vienu-lapu pieeju ir viegli optimizēt – pieprasīt no servera attiecīgos datus lietotāja sesijas sākumā un katru reizi tos pašus datus atkārtoti izmantot, lieki

neveicot datu pieprasījumus un ietaupot serverim nepieciešamo veicamo datubāzes vaicājumu skaitu. Pastāv arī iespēja atjaunināt tikai nepieciešamās komponentes, piemēram, atzīmēt tikai vienu koka virsotni jeb laboratorijas darbu kā iesāktu vai lietotāja iesākto laboratorijas darbu skatā nomainīt tikai vienam laboratorijas darbam topoloģijas izveides statusu. Spēles parasti ir atsevišķas datorprogrammas, un jebkādu lieku ielādes laiku tās parasti cenšas minimizēt. Tā kā vienu-lapu lietotnes pieeja nodrošina atsevišķas datorprogrammas izskatu, jo atkārtotas lapu ielādes netiek veiktas, tad tā ir ļoti piemērota jaunajai platformai, kura paredz nodrošināt spēles sajūtu.

Visu šo iemeslu un vienu-lapu pieejas popularitātes dēļ, jauno platformu ir ieteicams izstrādāt ar *JavaScript* programmēšanas valodas *React* ietvaru, kurš ir populārākais no visiem vienu-lapu pieejas ietvariem (skat. 2.2. nodaļas sākumu).

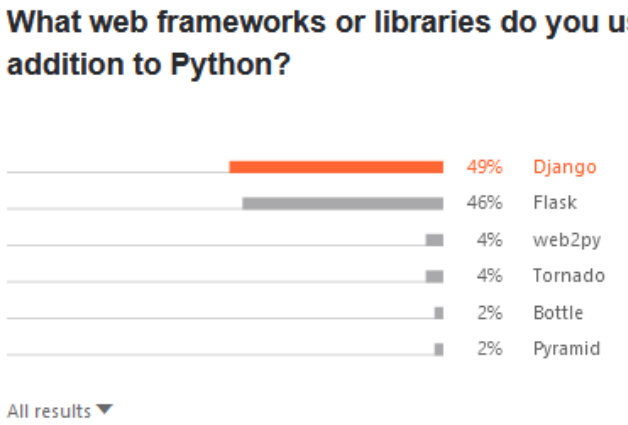
2.2.2. Servera izstrādes ietvars

Tā kā 2.2.1. nodaļā tika veikts secinājums izstrādāt lietotni pēc vienu-lapu pieejas, tad servera pusē ir jāspēj izveidot atbilstošu *HTTP API*, kas ar to sazināsies. Kā jau tika minēts 1. nodaļas aprakstā, pašreizējā *ReSeLa* platforma ir ļoti atkarīga no saziņas ar *OpenStack* mākoņa pakalpojumu, lai realizētu virtuālās topoloģijas, tāpēc viens faktors servera programmēšanas valodu izvēlē ir *OpenStack* saziņas veikšanas atbalsta bibliotēkas, kuras netika izskatītas un pielietotas pašreizējā *ReSeLa* platformā.

OpenStack piedāvā vairākas klienta jeb savienojuma veikšanas *Python* programmēšanas valodas atbalsta bibliotēkas [9] vairākām tā komponentēm, kā, piemēram, *Nova* (*python-novaclient*), *Neutron* (*python-neutronclient*), *Swift* (*python-swiftclient*) un *Heat* (*python-heatclient*) (sīkāk par *OpenStack* pakalpojuma komponentēm skatīt 3. nodaļu). Galvenā komponente, ar kuru *ReSeLa* platformai ir nepieciešams sazināties ir *Heat*, kurai pašlaik atbalsta bibliotēka *python-heatclient* ir pieejama, bet tā vēl ir izstrādes procesā [9]. Apskatot sīkāk *python-heatclient* bibliotēkas dokumentāciju [10], var redzēt, ka tās izmantošanas instrukcija jau ir iekļauta un, kā tika noskaidrots 4.3. nodaļā, tā ir funkcionējoša un pašlaik tiek izmantota jaunajā platformā.

Šī paša autora kursa darbā [1] tika aprakstīta ietvara izmantošanas nozīmība serveru izstrādē, kur galvenie ieguvumi bija atvieglota pirmkoda izstrāde, lasāmība un uzturēšana, jo ietvari iekļauj daudzas populāras, labi testētas atbalsta bibliotēkas, un tie parasti jau nosaka lietotnes struktūru. Tā kā piemeklētās *OpenStack* atbalsta bibliotēkas ir izstrādātas *Python* programmēšanas valodā, tad jaunās platformas servera izstrādei ir nepieciešams izvēlēties ietvaru tieši *Python* programmēšanas valodā. Līdzīgi kā tas tika apskatīts 2.2.1. nodaļā, populārāko *Python* serveru

izstrādes ietvaru var noteikt pēc *JetBrains The State of Developer Ecosystem 2020* aptaujas rezultātiem [11]. Attēlā 2.16 ir redzams, ka *Django* un *Flask* ir paši populārākie *Python* ietvari tieši tīkla lietotņu izstrādē.



2.16. attēls – ***JetBrains* aptaujas rezultāti par *Python* izstrādes ietvariem** [11]

Šī paša autora kursa darbā [1] tika veikta analīze par tīkla lietotņu serveru izstrādes ietvariem vairākās programmēšanas valodās, iekļaujot arī *Django* un *Flask*. Analīzes veikšanai tika noteiktas vairākas prasības, kuras servera izstrādes ietvars varētu nodrošināt jebkuras modernas mācību platformas izstrādē, un katram ietvaram izvērtēta to spēja prasības apmierināt. Katra ietvara nepilnībām tika piemeklētas arī trešo pušu atbalsta bibliotēkas, kas attiecīgās nepilnības varētu aizpildīt.

Ietvariem tika noteiktas šādas galvenās prasības:

- Dokumentācija un popularitāte – ir nepieciešama oficiāla, publiski pieejama lietošanas dokumentācija, kļūdu uzskaites sistēma un liels jautājumu skaits *Stack Overlfow* platformā;
- *REST API* izstrādes iespējas (sīkāk par *REST API* skatīt 2.3. nodaļu);
- Autentifikācijas un autorizācijas atbalsts – atbalsts vairāku veidu autentifikācijas un autorizācijas iespējām:
    - Iebūvēts risinājums ar iespēju definēt pieejas tiesības;
    - Autentifikācija un autorizācija ar *LDAP*;
    - Autentifikācija un autorizācija ar *OAuth2.0* protokolu [12], lai nodrošinātu iespēju izmantot profilus no citām platformām, kā *Google* un *Facebook*.

- Datubāzes atbalsts – atbalsts savienojumu un vaicājumu veikšanai ar populārākajiem datubāžu dzinējiem no *JetBrains The State of Developer Ecosystem 2020* aptaujas rezultātiem [13], kuri iekļauj *MySQL*, *PostgreSQL*, *MongoDB* un *SQLite*. Papildus, ir nepieciešams atbalsts migrāciju skriptu versiju kontrolei;
- *HTTP* pieprasījumu veikšanas atbalsts;
- *WebSocket* protokolu atbalsts – asinhronu tīklu savienojumu veikšanai, lai atbalstītu reālā laika satura izmaiņas attēlošanu lietotājam (sīkāk skatīt 2.4. nodaļā).

Zemāk redzamajās 2.1. un 2.2. tabulās, norādot ietvara funkcionalitātes pieejamību, tiek lietoti apzīmējumu saraksta 2. attēlā redzamie apzīmējumi. Tiek uzskatīts, ka ietvars atbalsta funkcionalitāti, ja tā ir pieejama kopā ar ietvaru vai, izmantojot kādu ietvara izstrādātāju izveidotu atbalsta bibliotēku. Pretējā gadījumā tiek uzskatīts, ka funkcionalitāte netiek atbalstīta, un tiek piemeklēta trešo pušu bibliotēka, kas nepilnību aizpildītu.

Tabulās 2.1 un 2.2 ir redzams, ka *Django* nodrošina daudz lielāku klāstu funkcionalitāšu nekā *Flask*, kas nenodrošina gandrīz nevienu. *Flask* ir *microframework* [14] jeb mikro ietvars, kas paredz nodrošināt tikai pamata funkcionalitāti, bet piedāvā iespējas to viegli papildināt ar plašu klāstu lietotāju izstrādātiem papildinājumiem, lai labāk pielāgotos visu projektu izmēriem un mazāk ierobežotu izstrādātājus. Tā kā *ReSeLa* ir liela apjoma projekts jeb tas paredz izmantot lielu daļu no augstāk minētajām ietvara prasībām, tad secinājums ir izmantot *Django* kā jaunās platformas serveru izstrādes ietvaru, lai mazāk paļautos uz trešo pušu bibliotēkām. Papildus, *Django* piedāvā iekļautu atbalstu administratoru skatiem, un 4. nodaļā ir redzams, ka šis ir izmantotais risinājums jaunajā platformā administratoru skatu ieviešanai.

2.1. tabula – ***Django*** **prasību atbalsta tabula**

Funkcionalitāte	Pieejamība	Piezīmes
Dokumentācija un popularitāte	✓	Atrastās kļūdas tiek uzskaitītas *GitHub*, un *Stack Overflow* jautājumu skaits pārsniedz 255 000.
API izstrādes iespējas	~	*REST API* izstrādes atbalstam ir pieejama *Django REST framework* bibliotēka.
Autentifikācija un autorizācija	~	Iebūvēts risinājums ar iespējām definēt piekļuves tiesības lietotājiem. Nav atbalsts *LDAP* un *OAuth2.0*. Lai nodrošinātu atbalstu *LDAP* ir pieejama bibliotēka *django-auth-ldap*.

		Lai nodrošinātu atbalstu *OAuth2.0* protokolam, ir pieejama bibliotēka *django-oauth-toolkit*.
Datubāzes atbalsts		Atbalsta migrāciju skriptu versiju kontroli. Atbalsta *MySQL*, *PostgreSQL*, *SQLite* datubāžu dzinējus. Lai nodrošinātu atbalstu *MongoDB* ir pieejama bibliotēka *django-mongodb-engine*.
HTTP pieprasījumu veikšana		*Python* pamata instalācijā iekļauj *urllib2* bibliotēku, sākot ar *Python 2.1*. Sākot ar *Python 2.1*, ir pieejama populāra *Requests* bibliotēka.
WebSocket protokola atbalsts		Nepieciešams lietot *Django* izstrādāto *channels* bibliotēku.

2.2. tabula – ***Flask*** **prasību atbalsta tabula**

Funkcionalitāte	Pieejamība	Piezīmes
Dokumentācija un popularitāte		Atrastās kļūdas tiek uzskaitītas *GitHub*, un *Stack Overflow* jautājumu skaits pārsniedz 42 000.
API izstrādes iespējas		*REST API* izstrādes atbalstam ir pieejama *Flask-RESTful* bibliotēka.
Autentifikācija un autorizācija		Vairāku veidu autentifikācijas un autorizācijas atbalsts ir iespējams ar *Flask-Security*, *Flask-OAuthlib* un *Flask-SimpleLDAP* bibliotēkām.
Datubāzes atbalsts		Migrācijas skriptu versiju kontroles atbalstam ir pieejama *Flask-Migrate* bibliotēka. Atbalsts vairāku veidu datubāžu dzinējiem ir pieejams ar *Django-SQLAlchemy* bibliotēku.
HTTP pieprasījumu veikšana		*Python* pamata instalācijā iekļauj *urllib2* bibliotēku, sākot ar *Python 2.1*. Sākot ar *Python 2.1*, ir pieejama populāra *Requests* bibliotēka.
WebSocket protokola atbalsts		Ir pieejama *Flask-SocketIO* bibliotēka.

2.3. Standartizēts *API*

Tā kā 2.2. nodaļā tika izskatīta nepieciešamība veidot platformu pēc vienu-lapu pieejas, un serverim ir nepieciešams nodrošināt *API* datu apmaiņai ar klienta puses lietotni, tad ir svarīgi veikt lēmumu par šī *API* formātu. Viens no risinājumiem ir izstrādāt *API* ar patvaļīgu struktūru, bet vēlamāks risinājums ir sekot noteiktai *API* izstrādes pieejai un standartam.

Galvenie ieguvumi [15] *API* standarta izmantošanai ir vieglāks un ātrāks izstrādes process, jo ir mazāk neskaidrību par to, kā labāk strukturēt adreses un pārsūtamos datus *HTTP* ziņojumos. Papildus, ja pie produkta darbojas vairāki cilvēki, tad šis ir arī labs veids kā novērst domstarpības starp izstrādātājiem un daudzi standarti, kā *REST*, ir pietiekami populāri, lai jebkurš izstrādātājs ar tiem jau būtu pazīstams.

Viena no nozīmīgākajām un populārākajām *API* izstrādes pieejām ir *REST*, kur pētījumi [16] rāda, ka 2017. gadā virs 80% no apskatītajiem *API* izmantoja *REST* izstrādes pieeju. Galvenās *REST* pieejas prasības ir:

- *JSON* [17] strukturētu datu izmantošana;
- *CRUD* (*Create, Read, Update, Delete*) operāciju veikšanai ir jāizmanto atbilstošie *HTTP* pieprasījumu tipi [18] – *POST, GET, PATCH* un *DELETE*;
- *API* adrešu strukturēšana atbilstoši sistēmas resursiem, piemēram, pieprasījums uz adresi "*/course/4/"* ir operāciju veikšana kursam ar identifikatoru 4;
- Atbilstošu *HTTP* atbilžu kodu [19] izmantošanu, kā, piemēram, *200 OK, 201 Created, 400 Bad Request, 500 Internal Server Error*.

Tā kā *REST* pieejas labā prakse ir izmantot *JSON* strukturētus datus, tad ir pieejami arī standarti kā *"JSON API"*, kuri nosaka *JSON* datu strukturēšanas formātu. Galvenās tā prasības [20] ir:

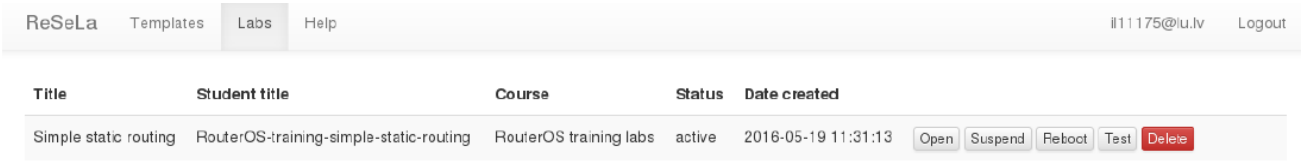
- Īpašs "*Content-Type*" ieraksts *HTTP* ziņojumu galvenē ar vērtību "*application/vnd.api+json*", kas palīdz abām ziņojuma pusēm atpazīt, ka tiek lietots "*JSON API"* standarta formāts;
- *JSON* datu izkārtojums jeb *JSON* struktūras lauki, zem kuriem ir jāatrodas galvenajai ziņojuma informācijai gan veiksmīgu, gan kļūdainu ziņojumu gadījumos;
- Ja *JSON* dati apraksta sistēmas resursu, tad ir jānorāda tā tips, resursa identifikators un tā attiecības ar citiem resursiem;

Vairāk piemēri *REST* un *JSON API* ir pieejami 4.5. nodaļā, kur tiek aprakstīts izveidotās jaunās platformas servera *API*.

2.4. Reālā laika satura atjaunināšana

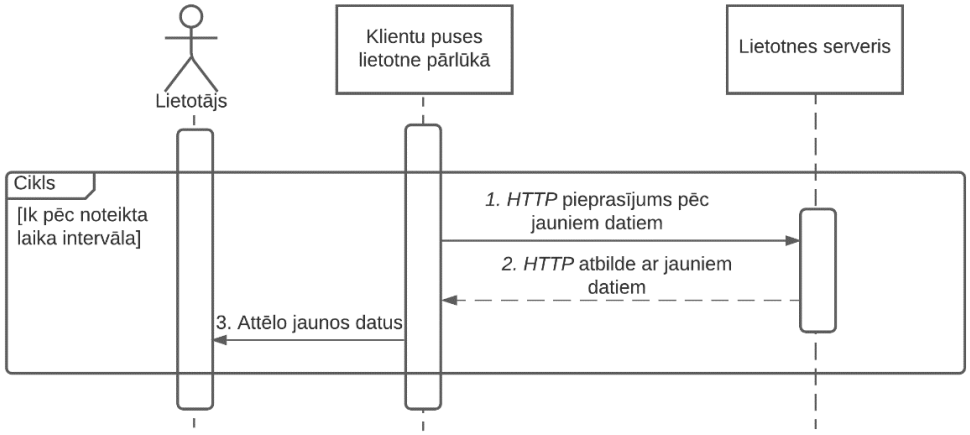
Daudzu mājaslapu gadījumos, lietotājam attēlotā lapa ātri var saturēt novecojušu informāciju, piemēram, ja lietotājs saņem lapas saturu biļešu servisa mājaslapai, kura uzrāda pašreizējo pieejamo biļešu skaitu, tad šī informācija ļoti ātri var vairs nebūt aktuāla, tiklīdz kāds cits lietotājs veic biļetes iegādi.

Pašreizējā *ReSeLa* platforma saskārās ar līdzīgu problēmu, jo *OpenStack* topoloģijas izveidošana prasa nozīmīgu laiku, un lietotājam pēc tās izveides pieprasīšanas ir nepieciešams gaidīt pirms ir iespēja veikt virtuālo iekārtu konfigurēšanu. Pašreizējā *ReSeLa* platformā lietotājiem bija nepieciešams atkārtoti, pašrocīgi atjaunināt 2.17. attēlā redzamo skatu, lai uzzinātu par pašreizējo topoloģijas izveides statusu, kas pasliktina lietotāja pieredzi un ir skaidrs funkcionalitātes trūkums. Precīzāk aprakstot problēmu, ir nepieciešams ieviest jaunas informācijas iegūšanu no servera un veikt satura nomaiņu reālā laikā. Šajā nodaļā tiks apskatīti vairāki šīs problēmas risinājumi un salīdzinātas to efektivitātes.



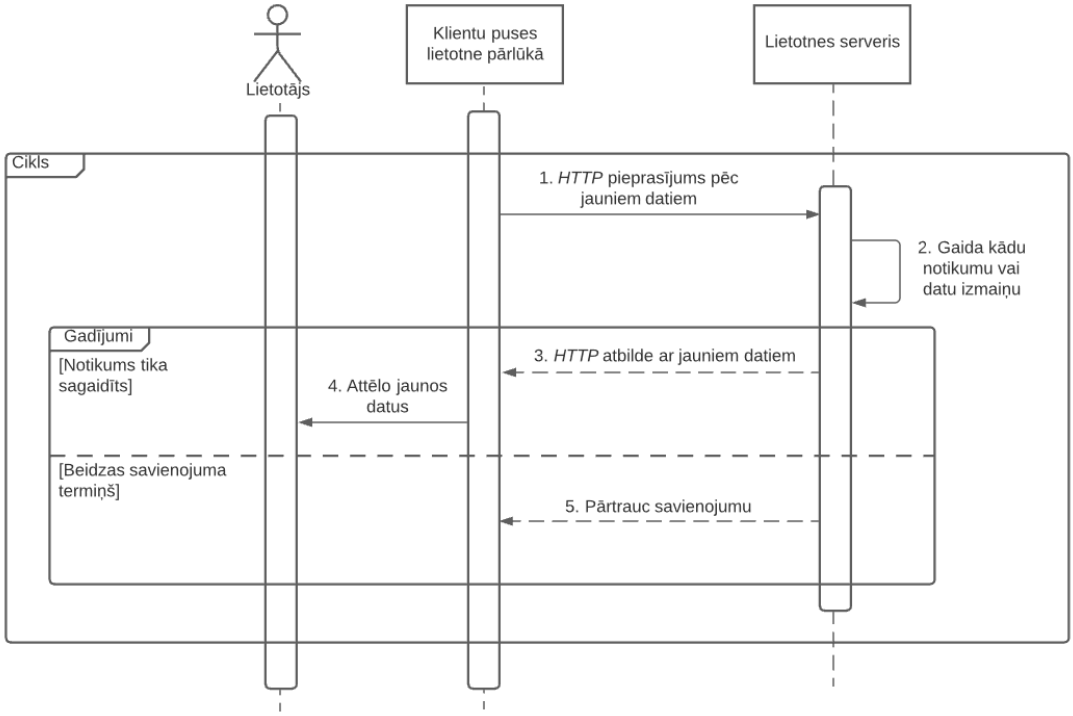
2.17. attēls – **Lietotāja skats ar izveidotajām laboratorijām** [2]

Viena no vienkāršākajām metodēm ir *polling* [21, 22, 23], kura izmanto *HTTP* protokolu, lai nodrošinātu reālā laika informācijas iegūšanu. Viens no *polling* variantiem ir *short-polling* [21, 22, 23] jeb īslaicīgais *polling* (skat. 2.18. attēlu), kur klients pēc noteiktiem laika intervāliem veic atkārtotus *HTTP* pieprasījumus serverim, un katra iegūtā atbilde tiek izmantota kā pašreizējā aktuālā informācija, kuru attēlot lietotājam. *Short-polling* ir ļoti vienkāršs risinājums, bet tam ir ļoti daudz trūkumu, jo tiek veikti daudzi lieki pieprasījumi, kad jauni dati vēl nav pieejami, un serveris tos visus ir spiests apstrādāt. Viens risinājums ir veikt laika intervāla optimizāciju, lai dati tiktu pieprasīti pietiekami bieži, lai informācija būtu aktuāla, bet pietiekami reti, lai neveiktu liekus pieprasījumus.



2.18. attēls – ***Short-polling*** **darbības secību diagramma**

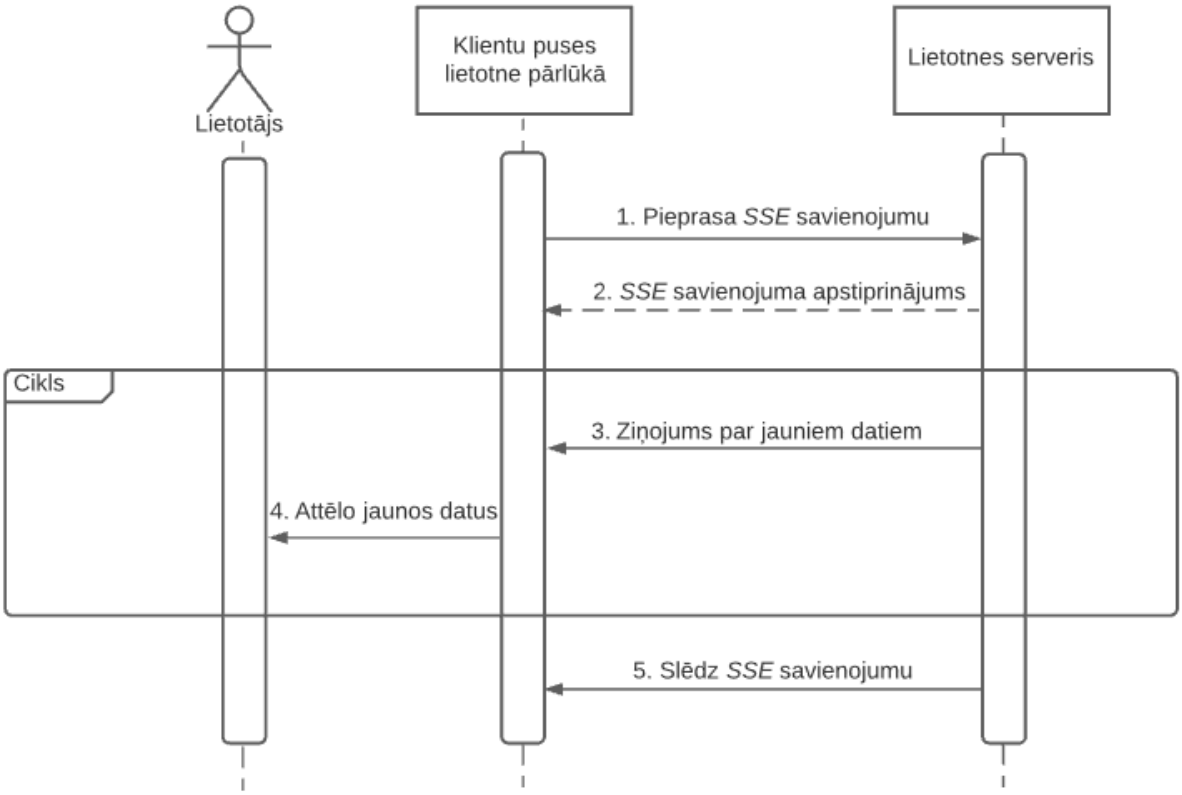
Otrs *polling* variants ir *long-polling* [21, 22, 23, 24] jeb ilglaicīgais *polling* (skat. 2.19. attēlu), kur klients veic vienu *HTTP* pieprasījumu serverim, un savienojums paliek atvērts tik ilgi, līdz serveris ir gatavs atbildēt ar jaunu informāciju vai beidzas savienojuma pastāvēšanas termiņš. Šī pieeja novērš galveno *short-polling* trūkumu, jo netiek veikti lieki pieprasījumi, kad jauni dati vēl nav pieejami. Savukārt, pārlūki parasti pieļauj tikai no 6 līdz 8 vienlaicīgiem *HTTP* savienojumiem katrā mājaslapā [25], kas var traucēt *long-polling* savienojumiem, ja to uz viena lietotāja ir pārāk daudz ar pārāk lieliem termiņiem.



2.19. attēls – ***Long-polling*** **darbības secību diagramma**

Gan *short-polling*, gan *long-polling* ir risinājumi, kuri paļaujas uz klienta iniciatīvu atkārtoti veidot *HTTP* pieprasījumus, kā rezultātā, kopā ar nepieciešamo informāciju katru reizi tiks lieki sūtīti *HTTP Header* jeb galvenes dati, kurus katru reizi ir nepieciešams pārsūtīt caur tīklu un abiem saziņas locekļiem apstrādāt un, kuri kopumā var aizņemt vairāk baitus nekā pati nosūtāmā informācija [21]. Tā kā *long-polling* optimizē nepieciešamo veicamo pieprasījumu skaitu, tad, salīdzinājumā ar *short-polling*, lieko nosūtīto baitu skaits ir mazāks. Abos šajos risinājumos patiess kanāls starp klientu un serveri, kur viena savienojuma ietvaros ir iespēja nosūtīt vairāk par vienu ziņojumu, netiek izveidots, tāpēc ir vērts apskatīt risinājumus, kuri ir šim nolūkam paredzēti.

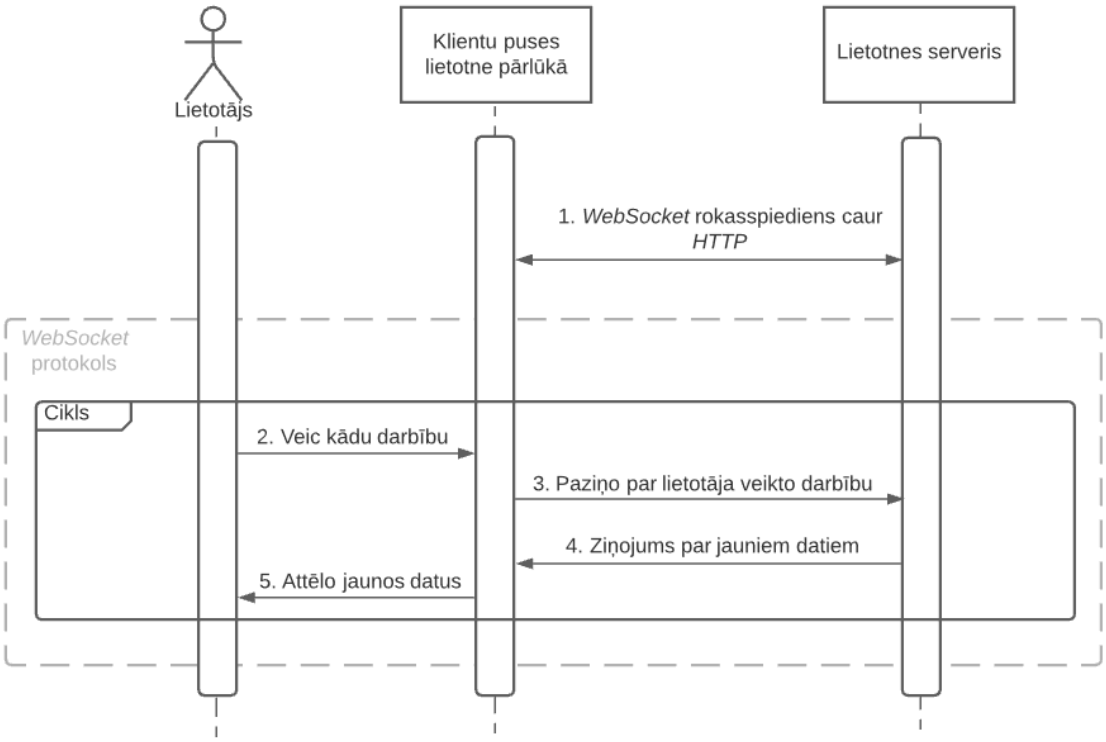
Viens risinājums, lai izveidotu vairāku ziņojumu kanālu, ir *SSE* [21, 23] (*Server-Sent Events* jeb servera sūtīti notikumi, turpmāk *SSE*), kurš arī ir balstīts uz *HTTP* protokolu un ir ticis izstrādāts, lai būtu iespēja izveidot efektīvu vienvirziena kanālu, parasti no servera uz klientu, kurā klients spēj saņemt vairākus ziņojumus. Kā redzams 2.20. attēlā, klientam ir nepieciešams tikai pieprasīt vienu savienojumu ar serveri, pēc kura tas spēj saņemt vairākus ziņojumus un katru reizi attēlot jauno informāciju lietotājam.



2.20. attēls – ***SSE* darbības secību diagramma**

Šī iemesla dēļ, *SSE* ir labs risinājums lietotnēs, kurās vienvirziena kanāls ir pietiekams, un klientam nav nepieciešams sūtīt informāciju atpakaļ serverim, kā iepriekš minētajam biļešu servisa mājaslapas piemēram. Papildus, *SSE* ir labs pārlūka atbalsts [26], kur tikai *Internet Explorer* versijas to neatbalsta, un tā *JavaScript* implementācija jau iekļauj automātisku savienojuma atkārtošanu negaidītas kļūdas gadījumā [23]. Galvenā problēma *SSE* savienojumiem ir, ka izveidotais ziņojumu kanāls ir tikai vienvirziena, ko var pārtaisīt par divvirziena kanālu ar diviem *SSE* savienojumiem, ko var būt sarežģīti izstrādāt un uzturēt [21]. *SSE* ir arī līdzīga problēma kā *long-polling,* jo *SSE* ir balstīts uz *HTTP* protokolu, un tas var būt mūžīgi aktīvs, bet pārlūki pieļauj tikai no 6 līdz 8 aktīviem *HTTP* savienojumiem vienlaikus katrā mājaslapā [25].

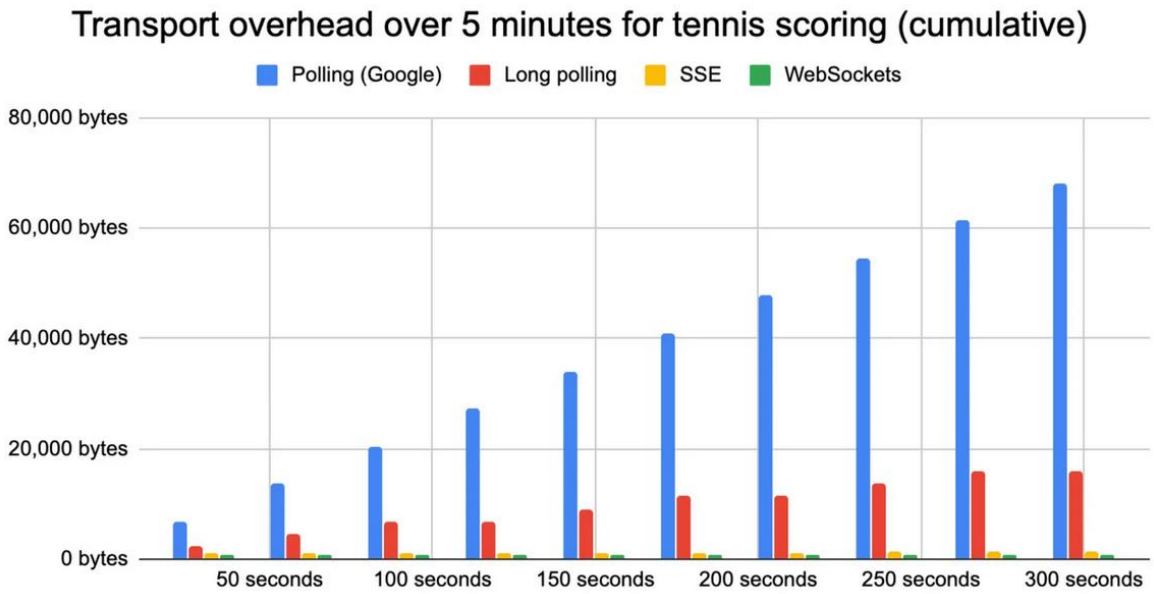
Mūsdienās vairākām tīkla lietotnēm ir nepieciešams divvirziena kanāls starp klientu un serveri, lai pārsūtītu vairākus ziņojumus abos virzienos, piemēram, tiešsaistes spēlēm vai sarakstes lietotnēm, kur ir nepieciešams reālā laikā pārsūtīt lietotāja veiktās darbības un saņemt, un attēlot no servera saņemtās citu dalībnieku veiktās darbības. Šāds divvirzienu kanāls ir iespējams, izmantojot *WebSocket* protokolu [21] *HTTP* protokola vietā. Kā redzams 2.21. attēlā, 1. solī *HTTP* protokols ir nepieciešams tikai, lai ar serveri vienotos par *WebSocket* protokola izmantošanu, pēc kā diagrammas visi pārējie soļi notiek, izmantojot *WebSocket* protokolu, kur soļi 2,3 un 4,5 var savstarpēji notikt vienlaicīgi.



2.21. attēls – ***WebSocket* darbības secību diagramma**

Papildus, *WebSocket* protokolam ir labs pārlūku atbalsts, kur arī *Internet Explorer* versijās tas ir pieejams [27]. Galvenā problēma *WebSocket* protokolam ir tā izstrādes sarežģītība, jo ir nepieciešams pašam izstrādāt funkcionalitātes, kā automātisku savienojuma atjaunošanu, bet šim nolūkam ir pieejamas daudzas atbalsta bibliotēkas, kā *react-use-websocket React* ietvaram.

Tā kā *SSE* un *WebSocket* pieejas nodrošina iespēju izveidot noturīgus vairāku ziņojumu kanālus, to galvenais ieguvums ir, ka ziņu nosūtīšana caur kanālu ir ļoti efektīva, jo, tiklīdz savienojums ir izveidots, nosūtāmiem datiem nav jāsatur lieka papildus informācija, kā *HTTP Header* dati *polling* gadījumā, un pat vienvirziena saziņas gadījumā *WebSocket* ir vēlamāks risinājums nekā *polling* varianti. Lai labāk nodemonstrētu *SSE* un *WebSocket* efektivitāti, salīdzinājumā ar *polling* metodēm, var apskatīt pētījumu [28] par *Google* tenisa turnīra rezultātu satura atjaunošanu reālā laikā. Pētījumi rāda, ka *Google* izmantoja *short-polling* metodi, kur katra *HTTP* pieprasījuma *Header* dati saturēja 1 650 baitus un atbilde 600 baitus. Attēlā 2.22 ir redzams šī scenārija veiktais salīdzinājums ar citām metodēm, kur piecu minūšu intervālā *short-polling* metode kopumā nosūtīja līdz 80 reizes vairāk lieku baitu, nekā *WebSocket* risinājums.



2.22. attēls – **Reālā laika satura atjaunošanas metožu salīdzinājums[28]**

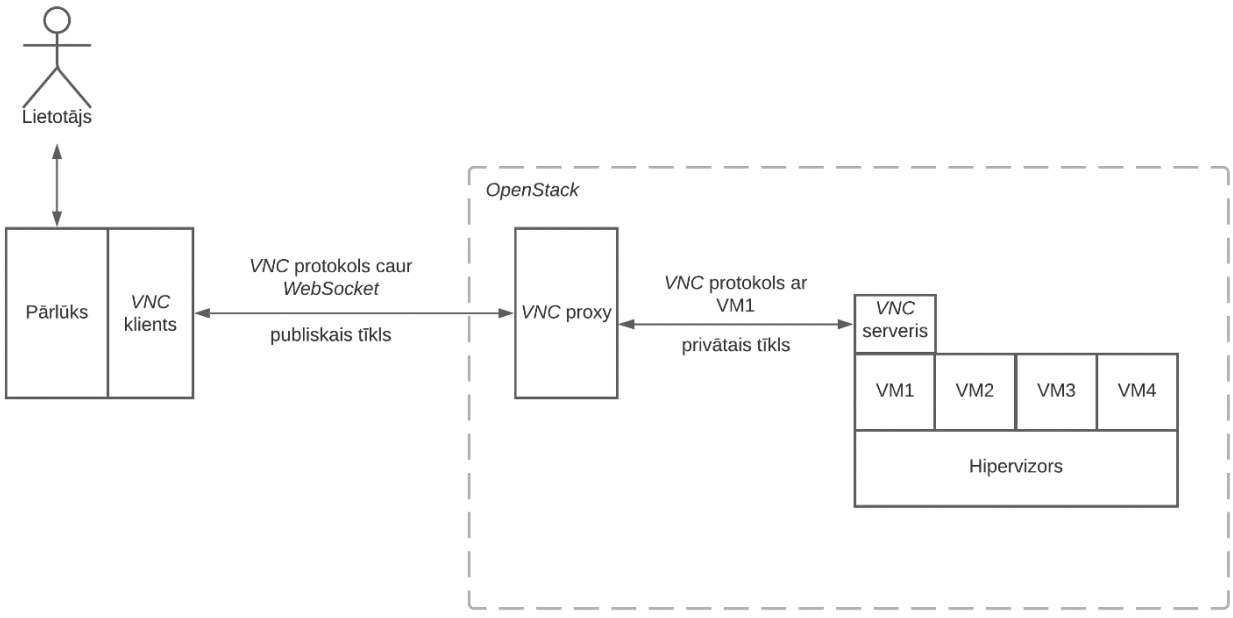
Pēc vairāku metožu un tehnoloģiju izpētes, varam secināt, ka ar *HTTP* ir grūti realizēt noturīgu vairāku ziņojumu saziņas kanālu starp serveri un klientu, kur gan *short-polling*, gan *long-polling* ir ļoti primitīvi un neefektīvi risinājumi, bet, savukārt, *SSE* (*Server-Sent Events*) ir efektīvs risinājums vienvirziena kanāla izveidošanai. Lai izveidotu efektīvu divvirziena kanālu starp klientu

un serveri, ir nepieciešams izmantot *WebSocket* protokolu *HTTP* protokola vietā, kuru ir grūti izstrādāt, bet uzrāda ļoti lielus uzlabojumus, salīdzinājumā ar *HTTP polling* metodēm, pat vienvirziena komunikācijas gadījumā. *Long-polling* metode, savukārt ir efektīva, ja no servera ir nepieciešams saņemt mazu skaitu jaunu datu ziņojumus, bet *short-polling* vienīgais ieguvums ir tā vienkāršība.

2.5. Attālinātas konfigurācijas vadības konsole

Pašreizējā *ReSeLa* platformā ļoti liela nozīme ir attālinātai virtuālo iekārtu konfigurēšanai, un viena no *ReSeLa* unikālākām funkcionalitātēm ir lietotnē iekļauts virtuālo iekārtu komandrindas vadības logs, lai nodrošinātu pēc iespējas lielāku ērtību konfigurāciju veikšanai. Papildus, *ReSeLa* piedāvā visu nepieciešamo informāciju, kā *VPN* adreses, lai lietotāji var veikt savienojumu ar iekārtām pašrocīgi un ar sev vēlamiem risinājumiem. *OpenStack* piedāvā jau gatavus risinājumus [29] virtuālo iekārtu saskarnes attālinātai piekļuvei no publiskā tīkla, izmantojot vienu no *VNC* [30], *SPICE* [31] vai seriālās konsoles [32] savienojumu, kur visām šīm trim pieejām *OpenStack* risinājums ir ļoti līdzīgs. Skatīt 3.2. nodaļu, kur tiek veikta katra risinājuma uzstādīšana un eksperimentēšana *OpenStack* testa vidē.

Attēlā 2.23 ir redzams pieejamais *OpenStack* risinājums *VNC* protokolam, kur diagrammā attēlotais *OpenStack* ir kopa ar pakalpojumiem, kuri nodrošina virtuālu tīklu un darbina virtuālas iekārtas uz uzstādītu hipervizoru.



2.23. attēls – ***OpenStack* risinājums *VNC* savienojumam ar virtuālo iekārtu [29]**

Galvenās risinājuma komponentes ir:

- *VNC* serveris – atrodas uz virtuālās mašīnas un spēj nosūtīt tās saskarni klientiem, izmantojot *VNC* protokolu;
- *VNC* klients – programma lietotāja pārlūkā, kas, izmantojot *VNC* protokolu, spēj attēlot *VNC* servera nosūtītos iekārtas saskarnes datus un nosūtīt pretī lietotāja veiktās darbības;
- *VNC proxy* – atrodas *OpenStack* sistēmā un darbojas kā starpnieks saziņā starp *VNC* serveri, kur tiek izmantots *OpenStack* privātais tīkls, un *VNC* klientu, kur tiek izmantots publiskais tīkls un *WebSocket* protokols, lai *VNC* klients jaunos saskarnes datus varētu atkārtoti saņemt un reālā laikā lietotājam attēlot.

Pašreizējā *ReSeLa* platforma izmantoja *OpenStack VNC* protokola risinājumu, lai nodrošinātu platformā iekļautas iekārtu konfigurācijas iespējas, kur galvenais tā trūkums ir teksta kopēšanas un ielīmēšanas funkcionalitāte tekstuālās saskarnes gadījumos, kura ir nepieciešama, ja studenti veic konfigurēšanu, piemēram, sekojot kādiem materiāliem ar dotām komandām.

SPICE protokola risinājums darbojas ļoti līdzīgi un *OpenStack* piemin [29], ka *SPICE* risinājums tika iekļauts, lai nodrošinātu vairākus *VNC* trūkumus, kā uzticamu kopēšanas un ielīmēšanas funkcionalitāti, kur 3.2. nodaļā tika secināts, ka šī funkcionalitāte tāpat nav iespējama tekstuālās saskarnes gadījumā. Abiem šiem risinājumiem *OpenStack* piedāvā jau gatavu *proxy* pakalpojumu, un klienta programmas ir viegli atrodamas (skat. 3.2. nodaļu).

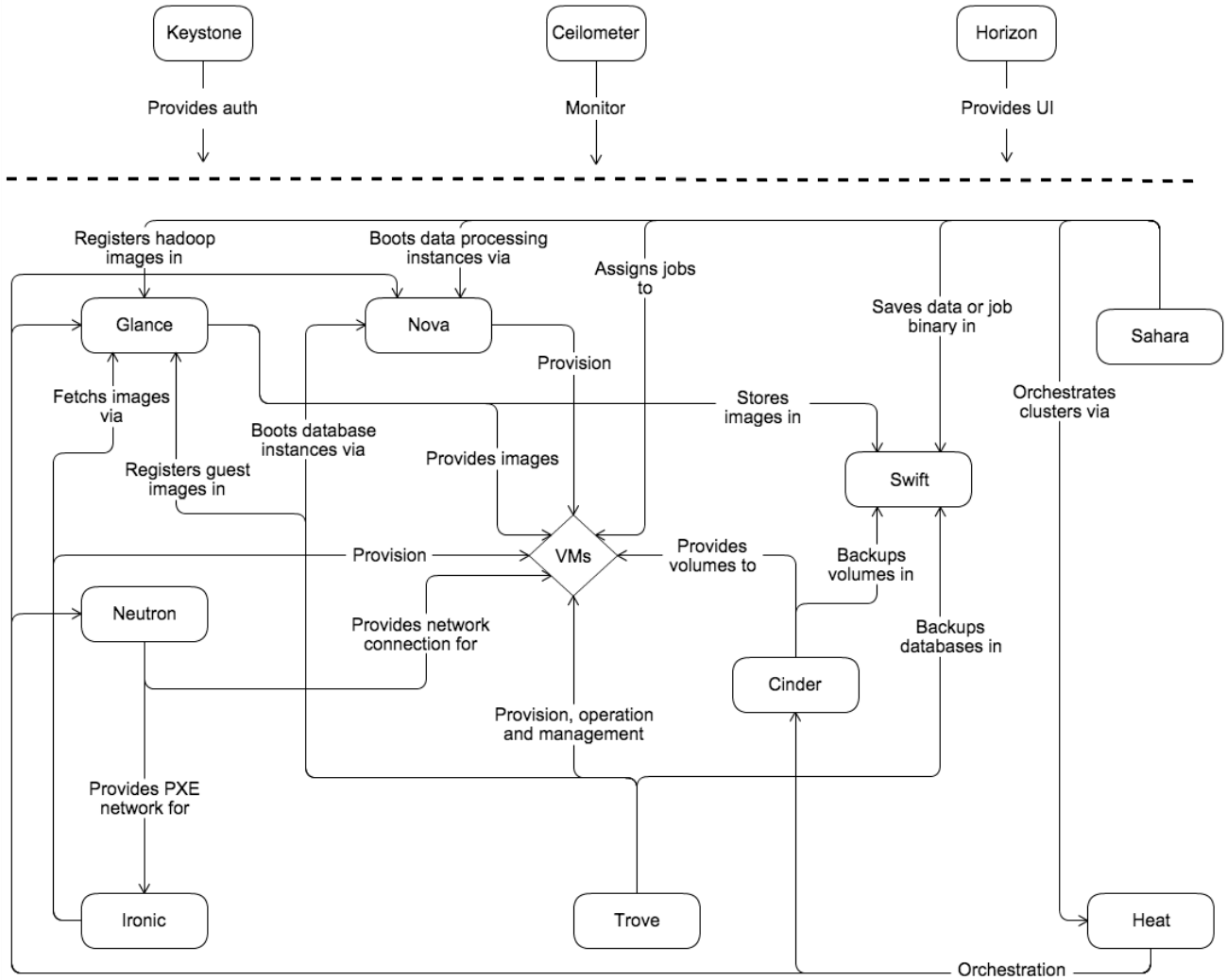
Pēdējais piedāvātais risinājums ir piekļuve virtuālās iekārtas seriālai konsolei, kur kopējā darbība ir līdzīga 2.23. attēlā redzamajai, bet *proxy* pakalpojums sazinās ar hipervizoru, kurš darbojas kā vēl viens starpnieks, lai piekļūtu virtuālās iekārtas seriālajam portam. Seriālais ports dod piekļuvi tikai tekstuālai iekārtas saskarnei, tāpēc, atšķirībā no *VNC* un *SPICE* protokoliem, klienta programma lietotāja pārlūkā caur *WebSocket* saņem lietotājam attēlojamo tekstu nevis informāciju grafikas attēlošanai. Šī iemesla dēļ, klienta programmai ir iespēja lietotājam attēlot tekstu, kuru ir iespējams iezīmēt, kopēt un ielīmēt, tāpēc šis risinājums ir piemērots, ja ir pieejama tikai tekstuālā komandrindas saskarne ar virtuālo iekārtu. Arī šim risinājumam *OpenStack* piedāvā *proxy* pakalpojumu, bet atbilstošā klienta programma netika atrasta.

Lai izstrādātu klienta programmu seriālās konsoles risinājumam, ir nepieciešams lietotājam pārlūkā attēlot mākslīgi uzzīmētu komandrindas logu, kurā tiek attēloti no *proxy* pakalpojuma caur *WebSocket* saņemtā iekārtas seriālā porta izvade un, kurā lietotāja ievadītā informācija caur to pašu kanālu tiek nosūtīta atpakaļ *proxy* pakalpojumam. Lai lietotājam pārlūkā uzzīmētu mākslīgu

komandrindas logu, ir pieejama *Xterm.js JavaScript* bibliotēka, kura ir izmantota daudzos citos projektos [33] un, kurai ir iekļauta iespēja pievienot *WebSocket* kanālu, lai nodrošinātu automātisku komandrindas informācijas apmaiņu ar *WebSocket* serveri kā *OpenStack proxy* pakalpojumu.

3. *DevStack*

Laboratorijas darbu virtuālo topoloģiju izveidošanai tiek izmantots *OpenStack* mākoņa pakalpojums, kas piedāvā infrastruktūru kā pakalpojumu jeb *IaaS* (*Infrastructure as a Service*). *OpenStack* mākonis sastāv no vairākām komponentēm kā *Nova, Keystone, Neutron, Heat*, kur katra nodrošina kādu no virtuālās topoloģijas uzstādīšanas funkcionalitātēm [34] (skat. 3.1. attēlu [35]).



3.1. attēls – ***OpenStack Wallaby* konceptuālā diagramma[35]**

Piemēram, *Nova* jeb skaitļošanas pakalpojums nodrošina virtuālo mašīnu izveidošanu un darbināšanu [36], *Keystone* nodrošina autentifikāciju un autorizāciju [37], *Neutron* nodrošina tīklus virtuālajām mašīnām [38]. Sīkāk par *OpenStack*, tās struktūru un komponentēm skatīt Ievas Lapiņas izstrādāto kvalifikācijas darbu [2], kur tiek pētīts *OpenStack* kā risinājums virtuālu topoloģiju izveidošanai.

ReSeLa galvenā saskarne ir ar *Heat* [39] jeb orķestrēšanas komponenti, kura veic vairāku minēto *OpenStack* komponenšu saskaņošanu, lai realizētu kopēju virtuālu tīkla topoloģiju. Papildus, *Heat* piedāvā tīkla *API* [40], lai platformas, kā *ReSeLa*, varētu ērti pieprasīt topoloģijas izveidošanu, sekojot *HOT* (*Heat Orchestration Template*) formāta [41] topoloģijas aprakstam, kurš var būt kā *YAML* fails vai *JSON* strukturēti dati.

Jaunās platformas izstrādē ir nepieciešama uzstādīta *OpenStack* vide, lai būtu iespēja testēt saziņu ar *Heat* komponentes *API* un topoloģiju izveidošanu. *OpenStack* uzstādīšana ir apjomīgs projekts, tāpēc šajā nodaļā tiek aprakstīts *DevStack* [42], kas ir *OpenStack* pakalpojuma samazināta komplektācija, lai nodrošinātu testa vidi izstrādes un testēšanas nolūkos. Jaunās platformas izstrādē galvenais mērķis *DevStack* uzstādīšanai ir nodrošināt minimālo komponenšu komplektāciju virtuālo topoloģiju izveidošanai un iespējot 2.5. nodaļā aprakstītās attālinātās virtuālo iekārtu konfigurēšanas iespējas.

3.1. *DevStack* uzstādīšana

DevStack uzstādīšanai ir pieejami apraksti [43] vairākām tā uzstādīšanas metodēm, kur izvēlētā metode bija visas tā komponentes uzstādīt uz vienas virtuālās mašīnas ar *Ubuntu Server 20.04.2 LTS* operētājsistēmu, 4 kodolu procesoru, 6GB *RAM* un virs 50GB cietā diska atmiņas. *DevStack* installācija jau iekļauj *Keystone*, *Glance*, *Nova*, *Cinder*, *Neutron* un *Horizon* komponentes, un *Heat* komponente ir pievienojama kā papildinājums pirms instalācijas [44], ar ko pietiek, lai nodrošinātu sistēmu virtuālo topoloģiju realizēšanai. Pirms instalācijas nepieciešamā konfigurācija ir jāveic "*local.conf*" failā, kur šī darba ietvaros izmantotā konfigurācija (skat. 3.2. attēlu) norāda piekļuves paroles, izmantojamās adreses un iespējo *Heat* komponenti un seriālās konsoles opciju attālinātai iekārtu konfigurēšanai.

```
[[local|localrc]]
# Uzstāda paroles visiem pakalpojumiem
ADMIN_PASSWORD=password
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD

# Publiskā adrese DevStack virtuālajai mašīnai
HOST_IP=192.168.0.98

# Pievieno HEAT komponentes papildinājumu
enable_service h-eng h-api h-api-cfn h-api-cw
enable_plugin heat https://opendev.org/openstack/heat

# Pievieno seriālās konsoles papildinājumu
enable_service n-sproxy

# Pievieno papildus Fedora operētājsistēmas iekārtas attēlu
IMAGE_URL_SITE="https://download.fedoraproject.org"
IMAGE_URL_PATH="/pub/fedora/linux/releases/33/Cloud/x86_64/images/"
IMAGE_URL_FILE="Fedora-Cloud-Base-33-1.2.x86_64.qcow2"
IMAGE_URLS+=","$IMAGE_URL_SITE$IMAGE_URL_PATH$IMAGE_URL_FILE

# Uzstāda virtuālajām mašīnām piešķiramās publiskās adreses
FLOATING_RANGE=192.168.0.224/27
# DevStack virtuālās mašīnas publiskā tīkla ports
FLAT_INTERFACE=enp0s3
```

3.2. attēls – **Izmantotā *DevStack* instalācijas konfigurācija**

Lai pārbaudītu *Heat* un citu komponenšu veiksmīgu instalāciju, var pieprasīt vienkāršas topoloģijas izveidi, izmantojot *HOT* formāta [41] topoloģijas apraksta *YAML* failu un *DevStack* pieejamo komandrindas saskarni. Izmantotais *HOT lab_stack_example.yaml* topoloģijas apraksta fails ir pieejams 1. pielikumā (skat. fragmentu 3.3. attēlā), un tas apraksta privātu tīklu ar vienu darbstaciju un maršrutētāju, kas privāto tīklu savieno ar publisko tīklu. Pielikumā iekļautais topoloģijas apraksts iekļauj:

- *workstation_network* – privātā tīkla resursa apraksts;
- *workstation_subnet* – privātā tīkla apakštīkla resursa apraksts;
- *workstation_subnet_port_gateway* – privātā tīkla apakštīkla adrese, kura tiks piešķirta maršrutētājam;
- *workstation_subnet_port_1* – privātā tīkla apakštīkla adrese, kura tiks piešķirta darbstacijai;
- *router* – maršrutētāja resursa apraksts;
- *router_workstation_subnet_port_interface* – maršrutētāja savienojums ar privāto tīklu;
- *workstation_1* – darbstacijas resursa apraksts.

```
49      workstation_1:
50          type: OS::Nova::Server
51          properties:
52              image: cirros-0.5.2-x86_64-disk
53              name: workstation_1
54              flavor: m1.tiny
55              networks:
56                  - port: { get_resource: workstation_subnet_port_1 }
57              user_data: |
58                  /system identity set name=PC_1
59              user_data_format: RAW
60
```

3.3. attēls – ***DevStack* instalācijas testēšanas topoloģijas darbstacijas apraksts**

Attēlā 3.3 ir redzams darbstacijas resursa jeb *workstation_1* apraksta fragments, kurā ir norādīti šādi parametri:

- tips (*type)* – *OS::Nova::Server* jeb tiek pieprasīts servera resurss, kurš definēts *Nova* komponentē;
- iekārtas attēls (*image*) – darbstacija izmantos gatavu *cirros-0.5.2* operētājsistēmas instalāciju;
- virtuālās mašīnas tips (*flavor*) – nosaka virtuālās mašīnas resursus, kur tips *m1.tiny* atbilst virtuālai mašīnai ar 512MB *RAM*, 1GB diska atmiņu un 1 *VCPU*;
- tīkli (*networks*) – norāda, kuram tīklam darbstacija ir pieslēgta, kur šajā gadījumā tā ir pieslēgta privātajam tīklam (*workstation_subnet*) ar adresi 10.0.0.50;
- lietotāja dati (*user_data*) – skripts, kurš tiek palaists uz virtuālās mašīnas tā startēšanas brīdī, šajā gadījumā uzstādot iekārtas nosaukumu uz “PC_1”.

Ar 3.4. attēlā redzamo komandu tiek pieprasīta virtuālās topoloģijas izveide, sekojot 1. pielikumā pievienotajam *lab_stack_example.yaml HOT* formāta topoloģijas apraksta failam, pēc kuras tiek paziņots par izveidotās topoloģijas jeb *stack* parametriem, kā identifikatoru, kurš tiek izmantots visu turpmāko darbību veikšanai ar izveidoto topoloģiju.

```
stack@ralfs:~/templates$ openstack stack create -t lab_stack_example.yaml test_stack
/usr/lib/python3/dist-packages/secretstorage/dhcrypto.py:15: CryptographyDeprecationWa
d, use int.from_bytes instead
  from cryptography.utils import int_from_bytes
/usr/lib/python3/dist-packages/secretstorage/util.py:19: CryptographyDeprecationWarnin
se int.from_bytes instead
  from cryptography.utils import int_from_bytes
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| id                  | 6d790e16-8cee-40a2-ab65-cfa49d891684 |
| stack_name          | test_stack                           |
| description         | ReSeLa test lab                      |
|                     |                                      |
| creation_time       | 2021-05-25T13:45:48Z                 |
| updated_time        | None                                 |
| stack_status        | CREATE_IN_PROGRESS                   |
| stack_status_reason | Stack CREATE started                 |
+---------------------+--------------------------------------+
stack@ralfs:~/templates$
```

3.4. attēls – ***DevStack* topoloģijas izveidošana ar *Heat* komponenti**

Rezultātā, 3.5. attēlā ir redzams, ka *DevStack* grafiskās saskarnes komponente, *Horizon*, uzrāda izveidoto tīkla topoloģiju jeb ir veiksmīgi izveidots privāts tīkls ar darbstaciju *workstation_1* 10.0.0.0/24 apakštīklā, kuru ar publisko tīklu savieno maršrutētājs.

3.5. attēls – ***DevStack* izveidotās topoloģijas grafiskais attēlojums *Horizon* komponentē**

3.2. *DevStack* iekārtu attālinātās piekļuves uzstādīšana

Par virtuālajām iekārtām un to attālināto piekļuvi ir atbildīga *Nova* komponente, un tās galvenais konfigurācijas fails *DevStack* vidē ir "*/etc/nova_cell1.conf*" un "*/etc/nova-cpu.conf*", kas satur konfigurāciju tieši par *nova-compute* pakalpojumu [45], kurš nodrošina hipervizoru un virtuālo mašīnu uzturēšanu. Atbilstošās platformas izstrādē lietotās konfigurācijas skatīt 3.6. un 3.7. attēlos, kur adrese 0.0.0.0 nozīmē klausīšanos jebkurā adresē, un *nova-cpu.conf* failā ir atļauts iespējot tikai vienu no *VNC* vai *SPICE* sistēmām vienlaikus.

```
#/etc/nova_cell1.conf

[vnc]
# ports, kurā klausās proxy pakalpojums
novncproxy_port = 6080
# adrese, kurā klausās proxy pakalpojums
novncproxy_host = 0.0.0.0

[spice]
# ports, kurā klausās proxy pakalpojums
html5proxy_port = 6081
# adrese, kurā klausās proxy pakalpojums
html5proxy_host = 0.0.0.0

[serial_console]
# ports, kurā klausās proxy pakalpojums
serialproxy_port = 6082
# adrese, kurā klausās proxy pakalpojums
serialproxy_host = 0.0.0.0
```

3.6. attēls – ***DevStack nova_cell1.conf* konfigurācija attālinātās piekļuves risinājumu iespējošanai**

```
#/etc/nova-cpu.conf

[vnc]
enabled = False
# adrese kurā virtuālai mašīnai gaidīt pieprasījumu
server_listen = 0.0.0.0
# adrese, kura tiek atgriezta lietotājam
novncproxy_base_url = http://192.168.0.97:6080/vnc_lite.html
# nova-compute pakalpojuma adrese
server_proxyclient_address = 127.0.0.1

[spice]
enabled = True
# iespējo vairākus funkcionalitāšu papildinājumus
agent_enabled = True
# adrese, kura tiek atgriezta lietotājam
html5proxy_base_url = http://192.168.0.97:6081/spice_auto.html
# adresē, kurā virtuālai mašīnai gaidīt pieprasījumu
server_listen = 127.0.0.1
# nova-compute pakalpojuma adrese
server_proxyclient_address = 127.0.0.1

[serial_console]
enabled = True
# adrese, kura tiek atgriezta lietotājam
base_url = ws://192.168.0.97:6082/
# nova-compute pakalpojuma adrese
proxyclient_address = 127.0.0.1
```

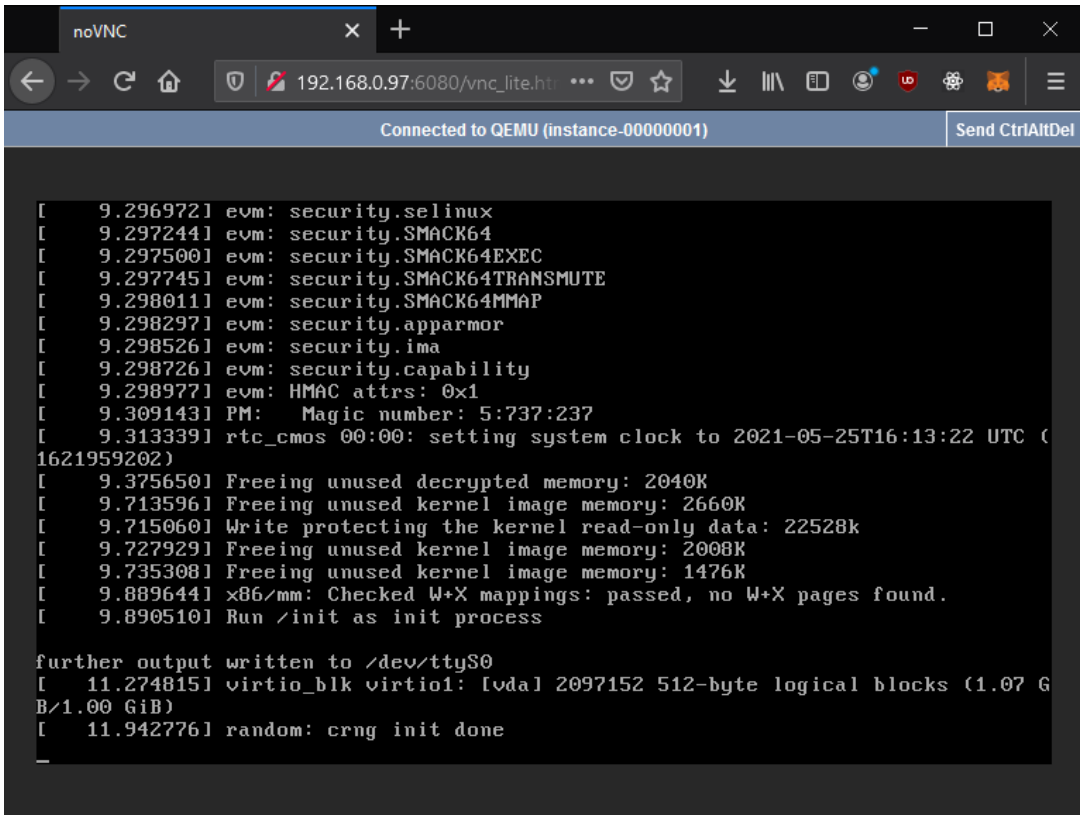
3.7. attēls – ***DevStack nova-cpu.conf* konfigurācija attālinātās piekļuves risinājumu iespējošanai**

Pēc atbilstošās konfigurācijas veikšanas, 3.1. nodaļā izveidotās virtuālās darbstacijas, *workstation_1*, attālinātās piekļuves adreses ir pieejamas ar 3.8. attēlā redzamo komandu.

```
stack@ralfs:~$ openstack console url show --novnc workstation_1
/usr/lib/python3/dist-packages/secretstorage/dhcrypto.py:15: CryptographyDeprecationWarning: int_from_bytes
  from cryptography.utils import int_from_bytes
/usr/lib/python3/dist-packages/secretstorage/util.py:19: CryptographyDeprecationWarning: int_from_bytes is
  from cryptography.utils import int_from_bytes
+----------+-------------------------------------------------------------------------------------------+
| Field    | Value                                                                                     |
+----------+-------------------------------------------------------------------------------------------+
| protocol | vnc                                                                                       |
| type     | novnc                                                                                     |
| url      | http://192.168.0.97:6080/vnc_lite.html?path=%3Ftoken%3De00d1aa1-3a44-4eb1-8269-453c4f11c5cb |
+----------+-------------------------------------------------------------------------------------------+
```

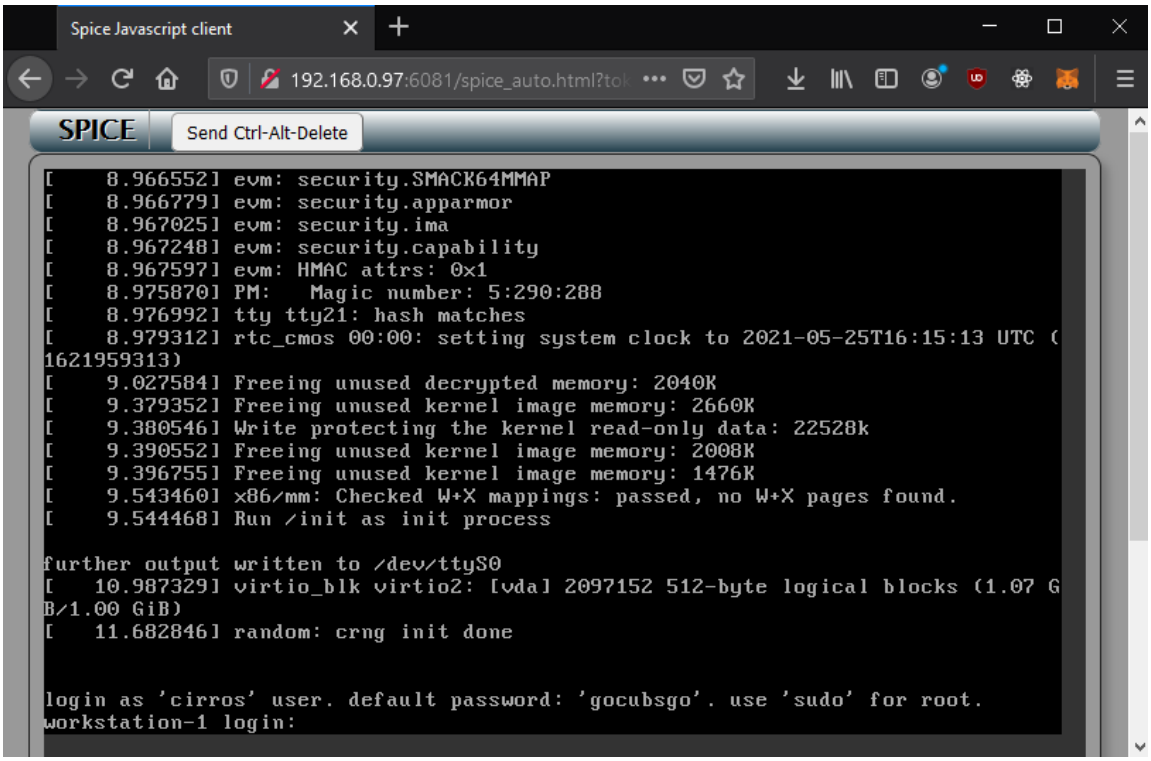
3.8. attēls – ***DevStack VNC* piekļuves adreses iegūšana**

VNC protokola gadījumā, *DevStack* jau iekļauj visu nepieciešamo, lai iegūto adresi varētu atvērt pārlūkā un redzēt *workstation_1* komandrindas logu (skat. 3.9. attēlu).



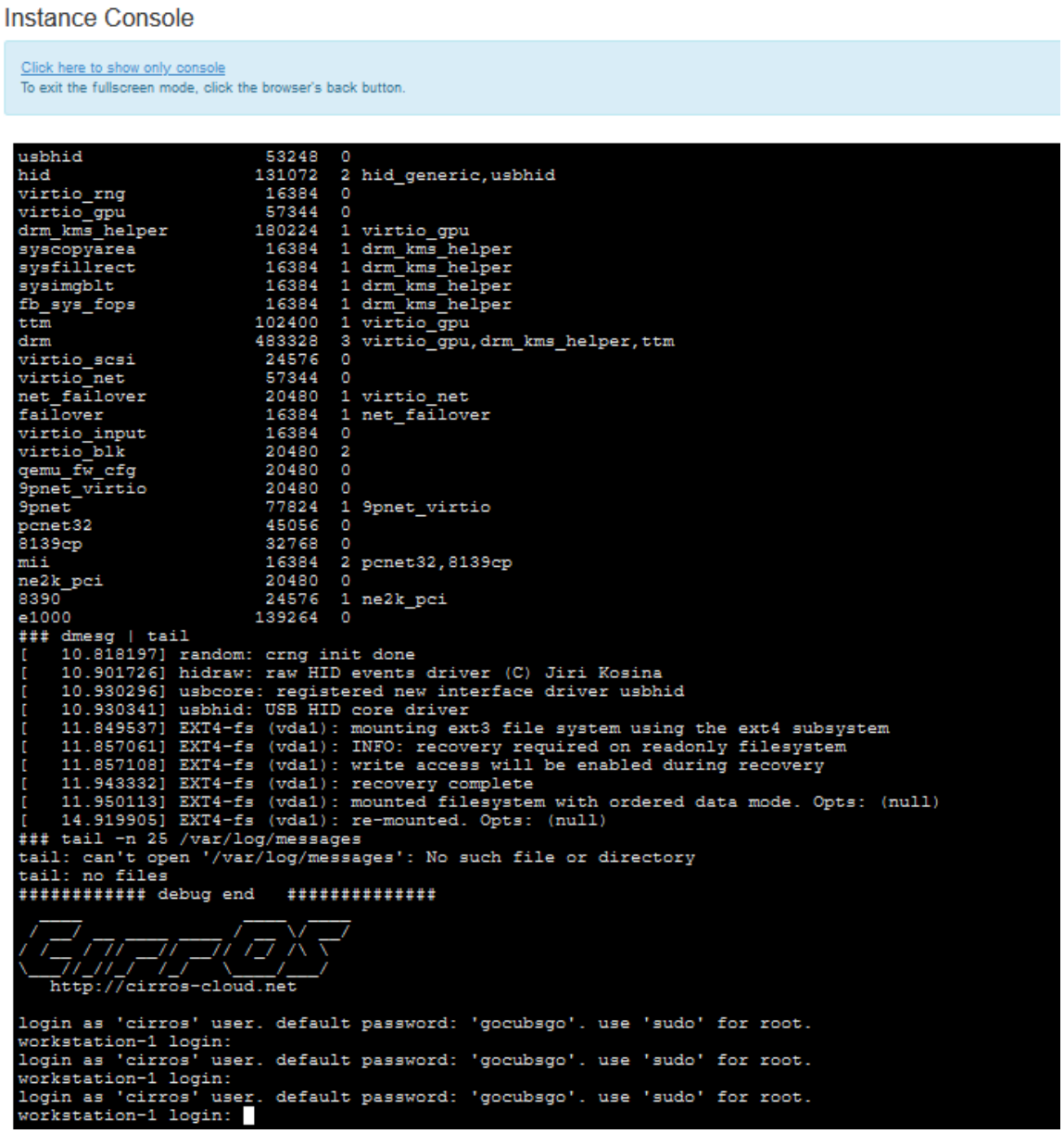
3.9. attēls – ***DevStack VNC* protokola komandrindas logs**

SPICE sistēmas gadījumā, *proxy* pakalpojums *nova-spicehtml5proxy DevStack* vidē ir jāiedarbina pašrocīgi, un klienta programma ir jāiegūst atsevišķi, kur šajā gadījumā tika izmantota *spice-html5 SPICE* protokola klienta programma. Pēc *proxy* iedarbināšanas un klienta programmas iegūšanas, atverot iegūto adresi pārlūkā var redzēt *workstation_1* komandrindas logu (skat. 3.10. attēlu).



3.10. attēls – ***DevStack SPICE* protokola komandrindas logs**

Seriālās konsoles gadījumā, *DevStack* trūkst klienta programma, kuru neizdevās piemeklēt, bet viens veids, kā to apskatīt, ir veikt *VNC* un *SPICE* protokolu risinājumu atslēgšanu *nova-cpu.conf* failā, pēc kura *OpenStack Horizon* administrācijas komponente uzrāda tai iebūvētu seriālās konsoles klienta programmas risinājumu (skat. 3.11. attēlu).



3.11. attēls – ***DevStack* seriālās konsoles risinājuma komandrindas logs**

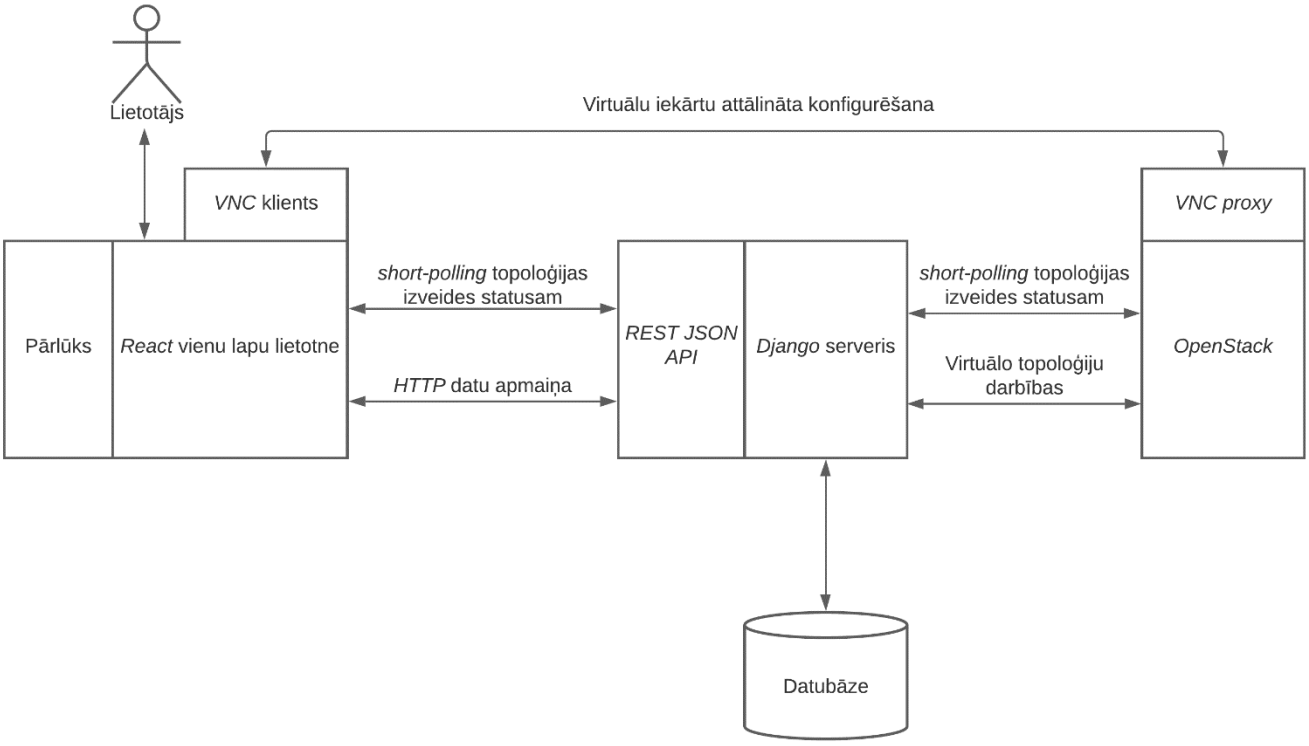
Atšķirībā no iegūtajiem *VNC* un *SPICE* protokolu komandrindas logiem, tikai seriālās konsoles savienojuma gadījumā komandrindas logā ir iespēja tekstu iezīmēt, iekopēt un ielīmēt, kas ir viens no nepieciešamajiem uzlabojumiem pašreizējā *ReSeLa* platformā, tātad secinājums ir lietot šo pieeju jaunajā platformā, kas iekļauj klienta programmas izstrādi, izmantojot 2.5. nodaļā minētās atbalsta bibliotēkas.

4. JAUNĀS PLATFORMAS APRAKSTS

Balstoties uz veiktajiem uzlabojumu un tehnoloģisko risinājumu pētījumiem 2. nodaļā, šī darba ietvaros tika iesākta jaunās platformas izstrāde, kuras pagaidu nosaukums ir *leetsim*. Šajā nodaļā tiek aprakstīts pašreizējās izstrādātās platformas projektējums, dizains, servera *API*, vairākas izstrādē izmantotās tehnoloģijas un metodes, turpmāk nepieciešamie uzlabojumi, kā arī tiek apskatītas iespējas platformas izvietošanai publiskajā tīklā un mērogojamībai.

4.1. Jaunās platformas projektējums

Attēlā 4.1 ir redzama izstrādātās platformas struktūra, kur, atbilstoši 2.2. nodaļā veiktajiem ieteikumiem, platforma ir sadalīta divās daļās – klienta puses lietotne lietotāja pārlūkā, kura ir izstrādāta ar *React JavaScript* ietvaru, un servera puses lietotne, kura ir izstrādāta ar *Python* programmēšanas valodas *Django* ietvaru.

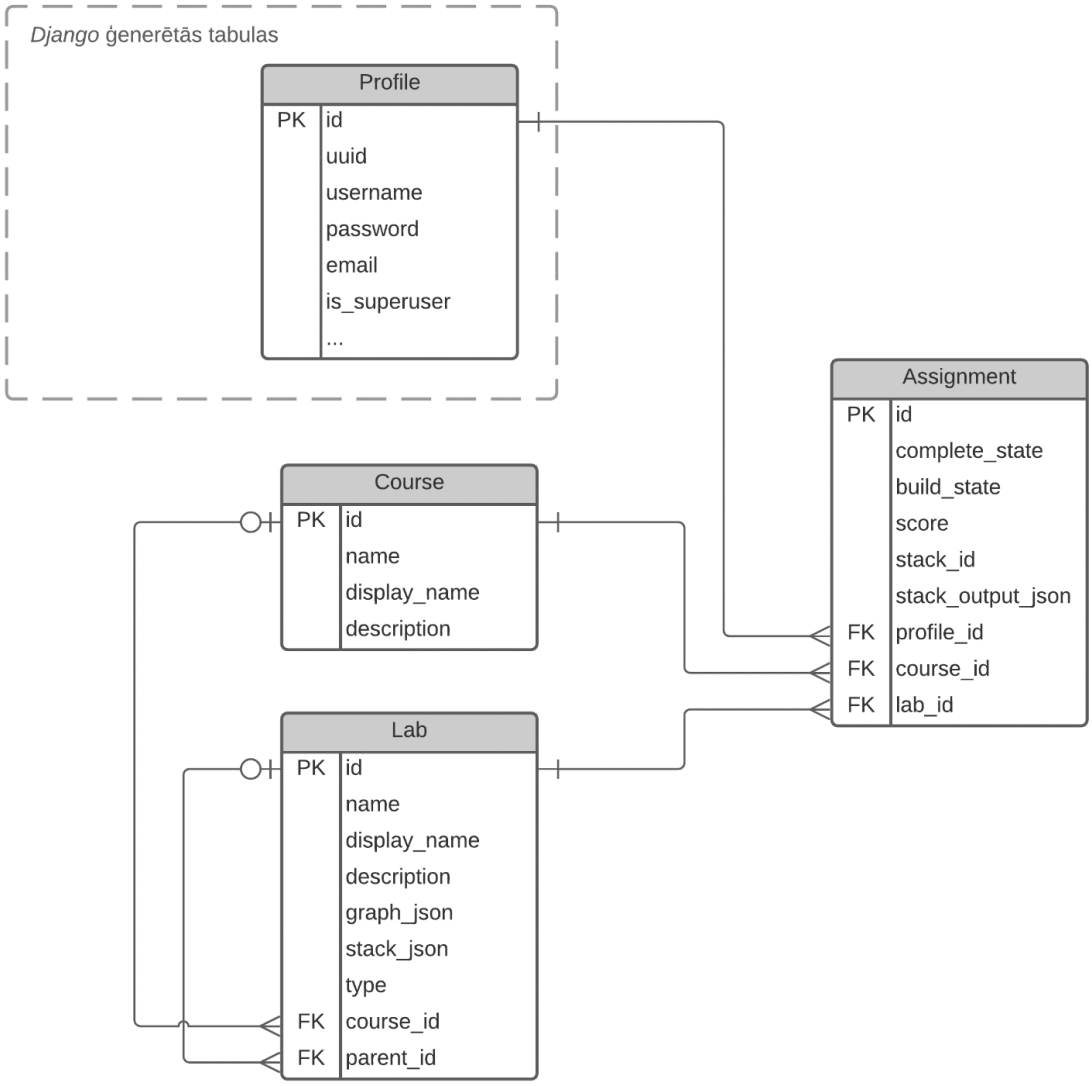


4.1. attēls – **Jaunās platformas struktūra**

Abas šīs lietotnes daļas savā starpā sazinās, izmantojot *API*, kurš izstrādāts pēc *REST* pieejas un dati ir strukturēti, sekojot *JSON API* standartam. Galvenā ārējā saskarne serverim ir ar *OpenStack* pakalpojumu, kurš nodrošina virtuālo topoloģiju uzturēšanu, kuru iekārtas lietotājam ir

iespēja attālināti konfigurēt, izmantojot *OpenStack* piedāvāto *VNC* risinājumu (skat. 2.5. nodaļu). Pēc topoloģijas izveidošanas pieprasīšanas *OpenStack* pakalpojumam, pašlaik tiek pielietota *short-polling* metode, lai atkārtoti noteiktu tās izveides statusu un lietotājam to reālā laikā attēlotu.

Attēlā 4.2 ir redzama servera datubāzes *ER* diagramma, kur daļu no tabulām, kā lietotāju datu tabulu, nodrošina *Django* ietvars, bet pašrocīgi veidotās tabulas iekļauj kursu datu tabulu (*Course*), laboratorijas darbu datu tabulu (*Lab*) un iesākto laboratorijas darbu datu tabulu (*Assignment*), kur *display_name* lauki ir laboratorijas darbu un kursu nosaukumi, kuri ir paredzēti attēlošanai lietotājam, un laboratorijas darbu *type* lauks pieņem vērtību "*topology*" vai "*tutorial*", lai tos klienta puses lietotne var atšķirt un atbilstoši attēlot. Citu lauku nozīme tiek aprakstīta 4.3. nodaļā, kur tiek aprakstīti izmantotie risinājumi.



4.2. attēls – **Jaunās platformas datubāzes *ER* modelis**

4.2. Dizains un funkcionalitātes

4.2.1. Lietotāja skati

Jaunās platformas dizains seko 2.1.2. nodaļā veidotajiem projektējumiem, kuri iekļauj visas dizaina iezīmes un funkcionalitātes, kuras tika sastaptas citās pētītajās mācību platformās. Platforma pašlaik piedāvā tikai svarīgākās ar lietotāju kontiem veicamās darbības, kā pieslēgšanos (skat pieslēgšanās skatu 4.3. attēlā) un atvienošanos.

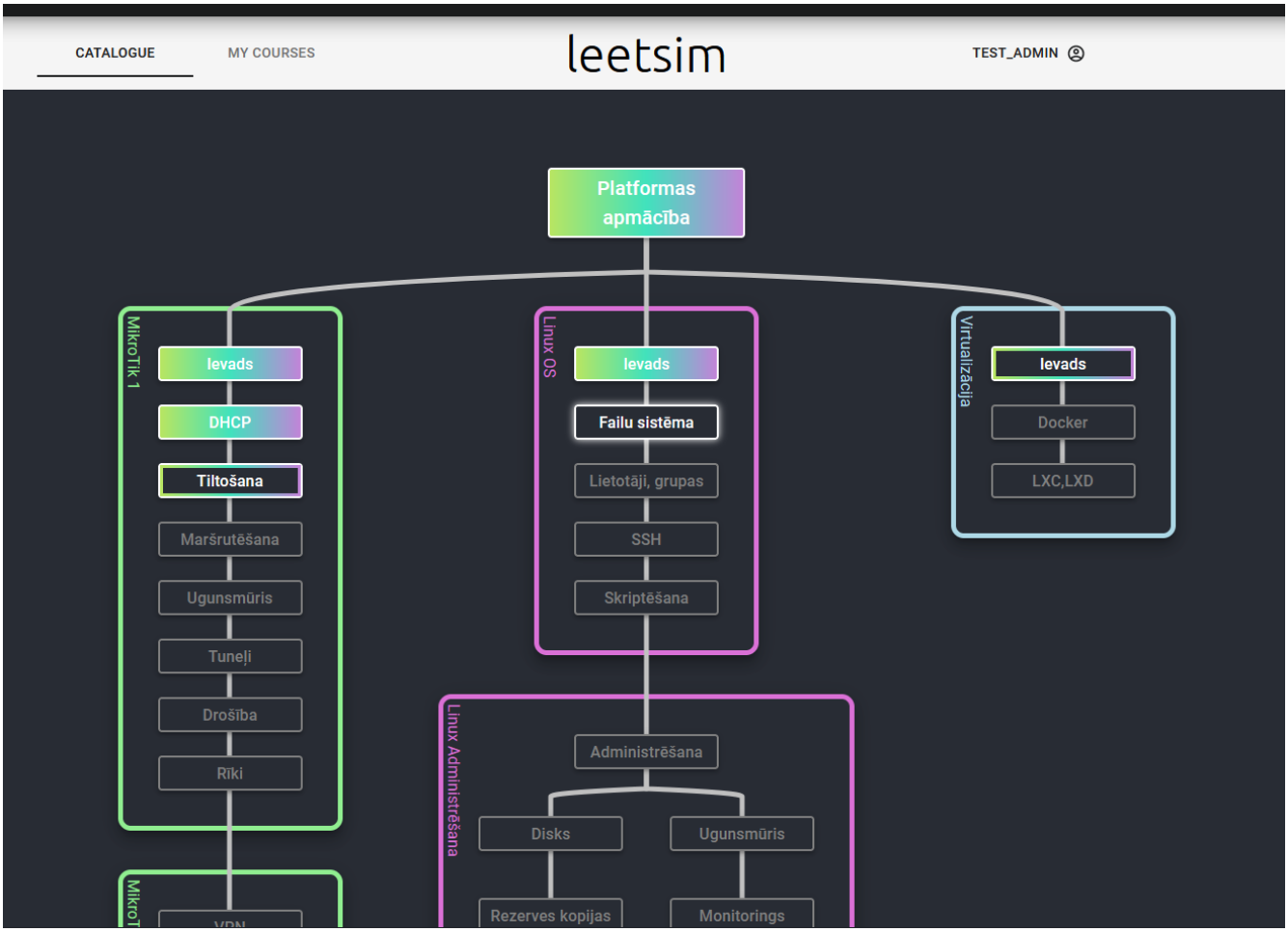
4.3. attēls – **Jaunās platformas pieslēgšanās skats**

Attēlos 4.4 un 4.5 ir redzams "katalogs" jeb visu pieejamo kursu skats, kur visi kursi un to laboratorijas darbi ir attēloti kokveida struktūrā, lai nodrošinātu spēļu elementu, atbilstoši 2.1.2. nodaļā veiktajam projektējumam. Pirmā virsotne kokā ir platformas apmācības laboratorijas darbs, kuram tālāk seko ceļi uz laboratorijas darbiem kursos *MikroTik 1*, *Linux OS* un Virtualizācija, pēc kuriem seko ceļi uz laboratorijas darbiem citos kursos. Katrs laboratorijas darbs ir pieejams tikai tad, kad tā "vecāka" laboratorijas darbs kokā ir pabeigts, kur:

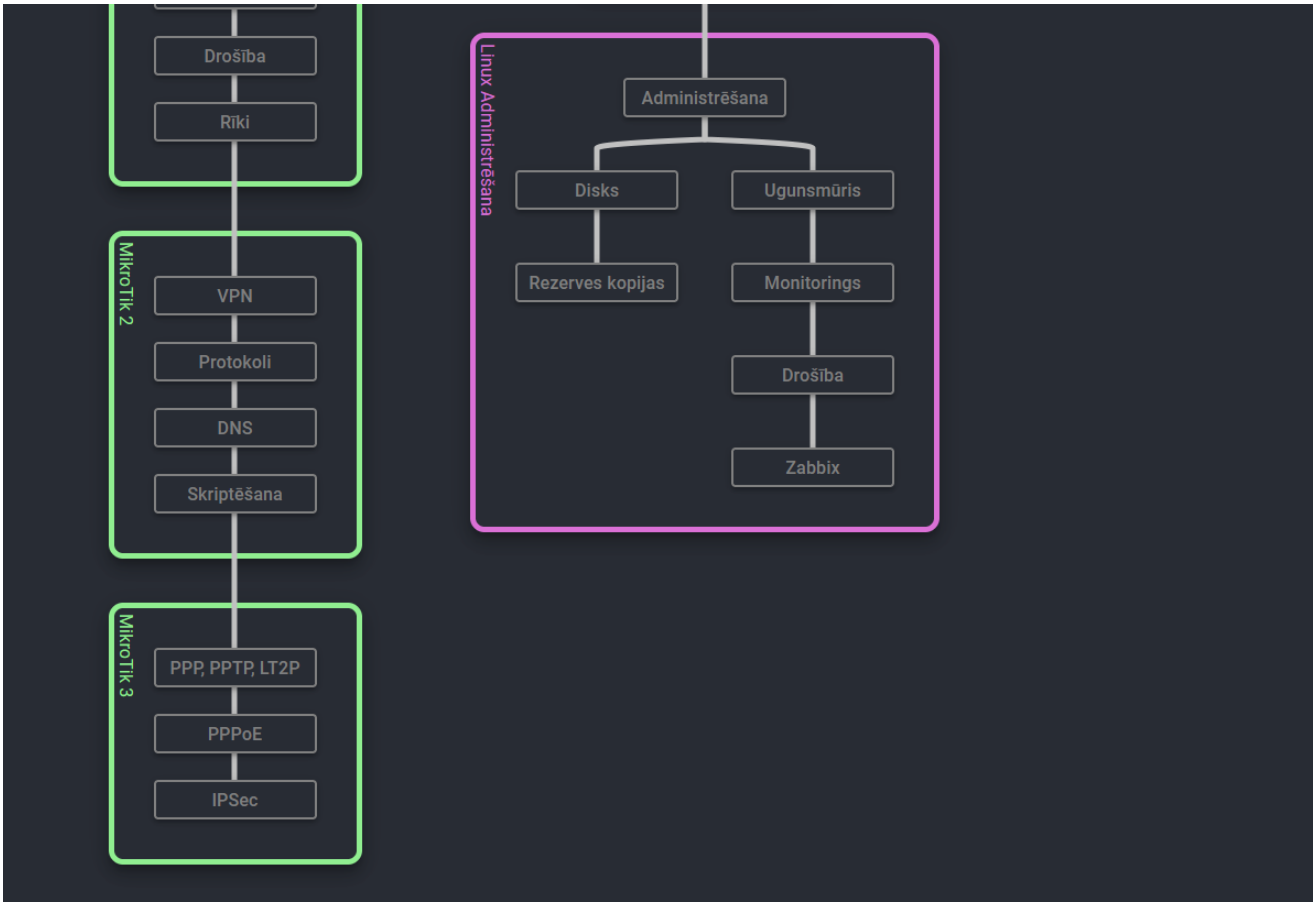
- Kokā redzamie iekrāsotie laboratorijas darbi ir **pabeigti**;
- Ar iekrāsotu rāmi ir **iesākti**;
- Tukši ar baltu tekstu ir **pieejami**;
- Tukši ar pelēku tekstu **nav pieejami**, jo to vecāku laboratorijas darbs nav pabeigts.

Attēlos 4.4 un 4.5 ir atbilstoši redzams, ka šim lietotājam:

- Platformas apmācība, *Linux OS* ievads, *MikroTik 1* ievads un *DCHP* laboratorijas darbi ir **pabeigti**;
- *MikroTik 1* tiltošana un Virtualizācijas ievads ir **iesākti**;
- *Linux OS* failu sistēmas laboratorijas darbs ir **pieejams** un pārējie pašlaik **nav pieejami**.

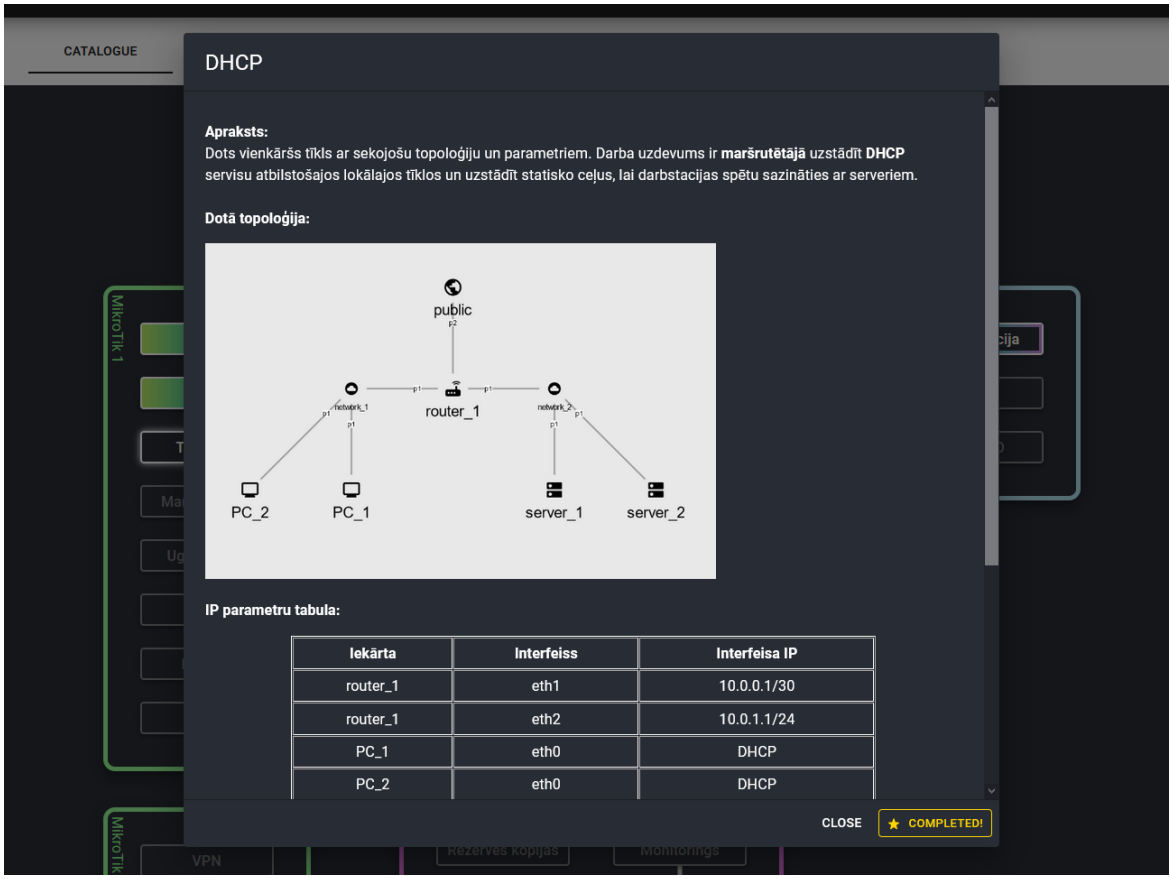


4.4. attēls – **Jaunās platformas kursu koks 1**



4.5. attēls – **Jaunās platformas kursu koks**

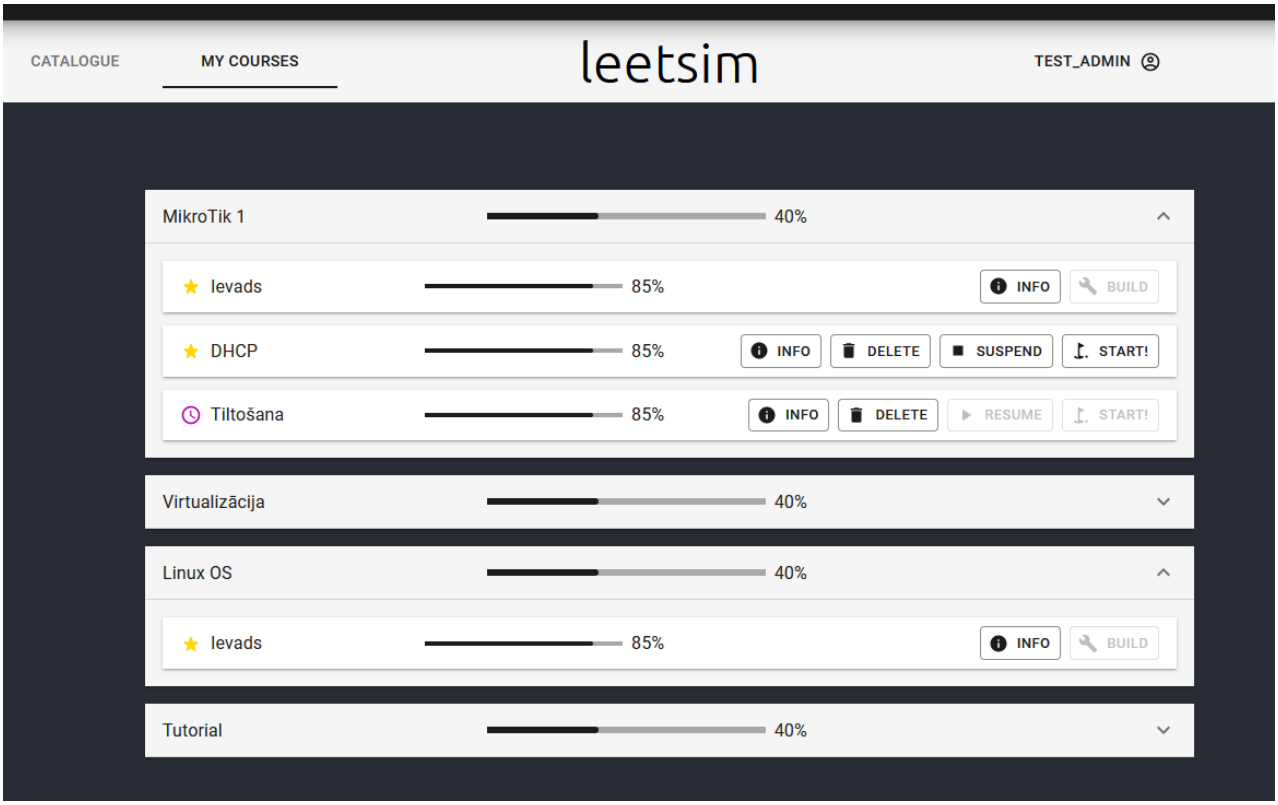
Visiem laboratorijas darbiem kokā ir iespēja apskatīt aprakstu (skat. *DHCP* laboratorijas darba aprakstu 4.6. attēlā), kas iekļauj uzdevuma formulējumu un aprakstu par virtuālo topoloģiju, kura tiks realizēta *OpenStack* pakalpojumā, kad laboratorijas darbs tiks izveidots. Ja laboratorijas darbs ir pieejams, tad šajā skatā to ir iespējams iesākt.



4.6. attēls – **Jaunās platformas laboratorijas darba apraksts**

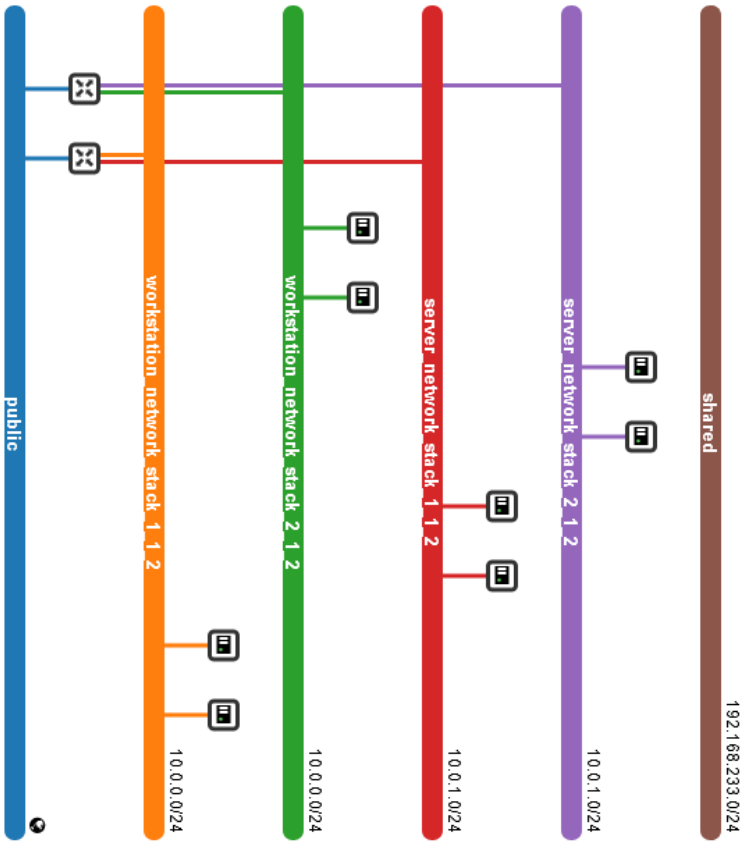
Attēlā 4.7 ir redzams lietotāja iesākto laboratorijas darbu skats, kas satur informāciju par katra kursa iesāktajiem un pabeigtajiem laboratorijas darbiem. Katram kursam un laboratorijas darbam ir attēlots tā progress, kurš pašreizējā platformā vēl netiek aprēķināts, un katram laboratorijas darbam ir iespējams:

- Apskatīt tā aprakstu ar *info* pogu (skat. 4.6. attēlu);
- Izveidot laboratorijas darbu jeb iesākt tā topoloģijas realizēšanu *OpenStack* pakalpojumā ar *build* pogu;
- Ja laboratorijas darbs ir izveidots, tad ir iespēja:
    - Sākt laboratorijas darba izpildi ar *start!* pogu;
    - Apturēt laboratorijas darba realizēto topoloģiju ar *suspend* pogu;
    - Atsākt laboratorijas darba apturēto topoloģiju ar *resume* pogu;
    - Izdzēst laboratorijas darba realizēto topoloģiju ar *delete* pogu.



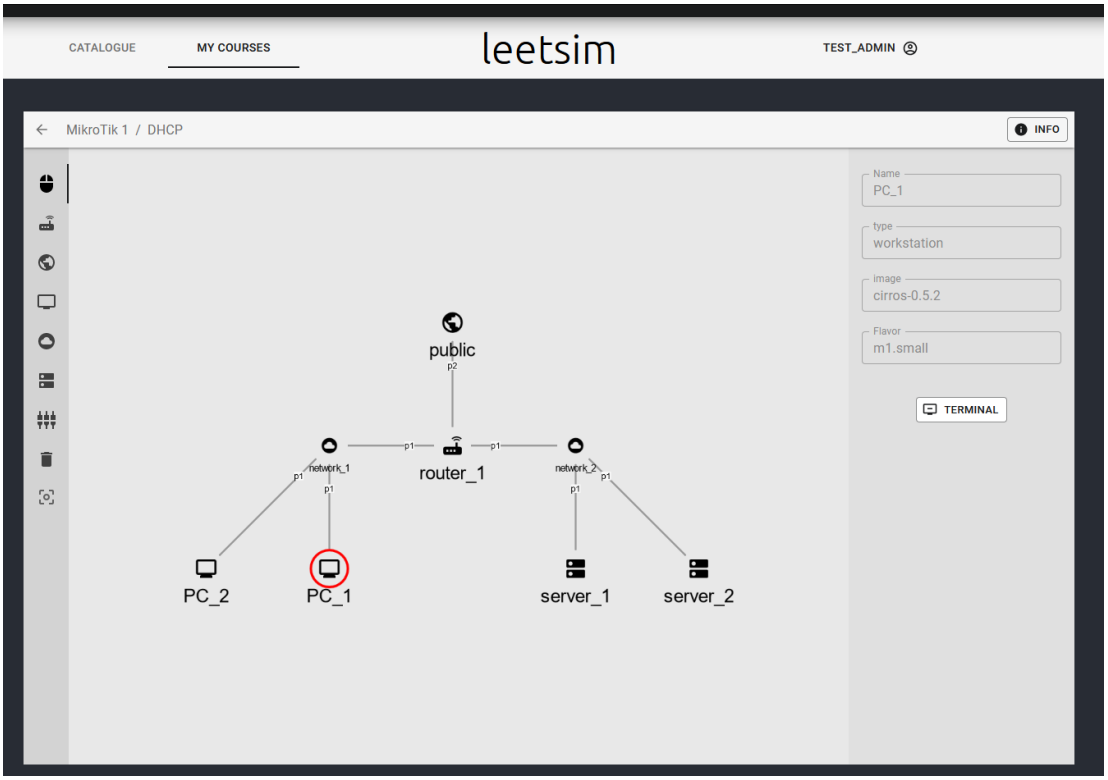
4.7. attēls – **Jaunās platformas lietotāja laboratorijas darbu skats**

Attēlā 4.7 ir redzams, ka *MikroTik 1* kursa *DHCP* laboratorijas darbs ir ticis izveidots, kas nozīmē, ka tā 4.6. attēlā redzamā topoloģija ir tikusi realizēta *OpenStack* pakalpojumā, un šim lietotājam nav iespējas izveidot vai atsākt citus laboratorijas darbus, jo, resursu ietaupīšanas nolūkos, katram lietotājam ir atļauts vienlaikus realizēt tikai vienu topoloģiju. *DHCP* laboratorijas darba izveidošanas rezultātā, tā realizētā topoloģija ir redzama *OpenStack Horizon* komponentes administratora skatā (skat. 4.8. attēlu), kurš uzrāda, ka pašlaik sistēmā kopumā divi lietotāji vienlaikus ir realizējuši *DHCP* laboratorijas darba topoloģiju.

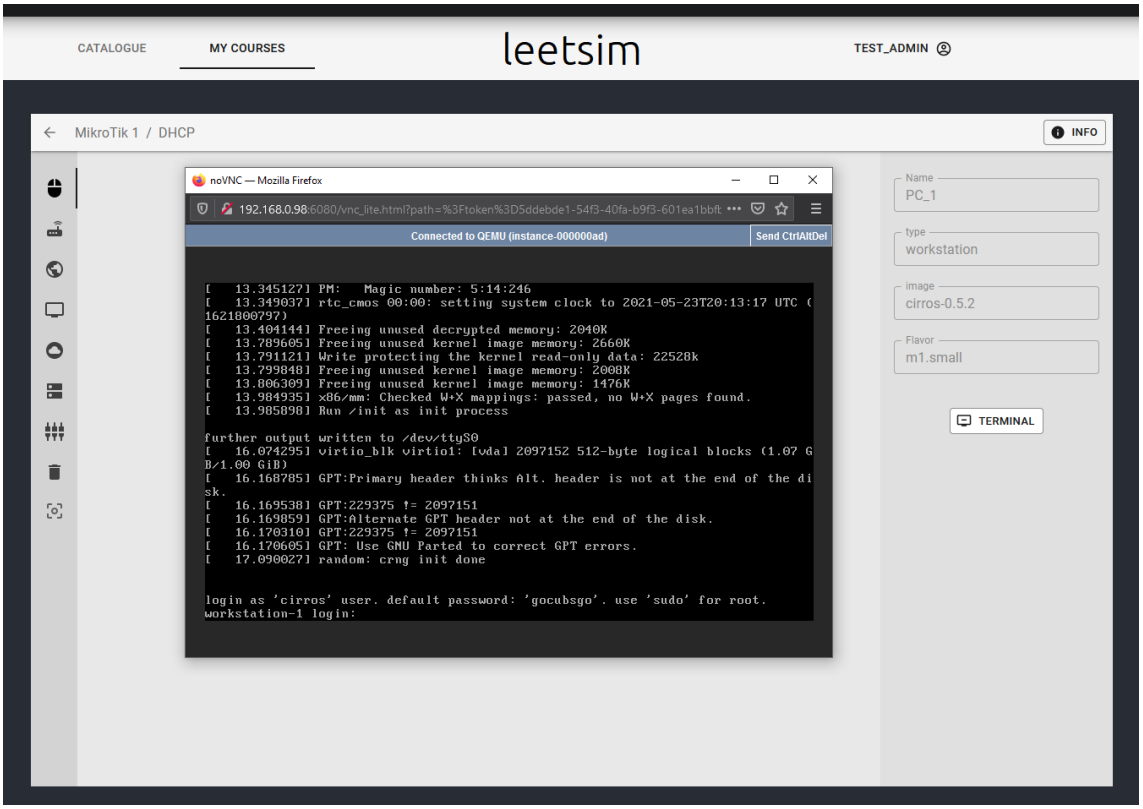


4.8. attēls – ***OpenStack* divas realizētas *DCHP* laboratorijas darba topoloģijas**

Attēlā 4.9 ir redzams izveidota laboratorijas darba skats, kur ir attēlots realizētās topoloģijas grafs ar iespēju topoloģijas iekārtas izvēlēties, apskatīt to informāciju labajā panelī un atvērt izvēlētās iekārtas komandrindas logu ar pogu *terminal* (skat. 4.10. attēlu). Šajā skatā kreisajā pusē ir redzama arī izvēlne, kura nodrošina topoloģijas grafa rediģēšanu, kā, piemēram, iekārtu un iekārtu savienojumu izveidošanu vai dzēšanu. Atbalsts laboratorijas darbiem, kur lietotāji var realizēt pašu veidotas topoloģijas vēl nav iekļauts pašreizējā platformā.



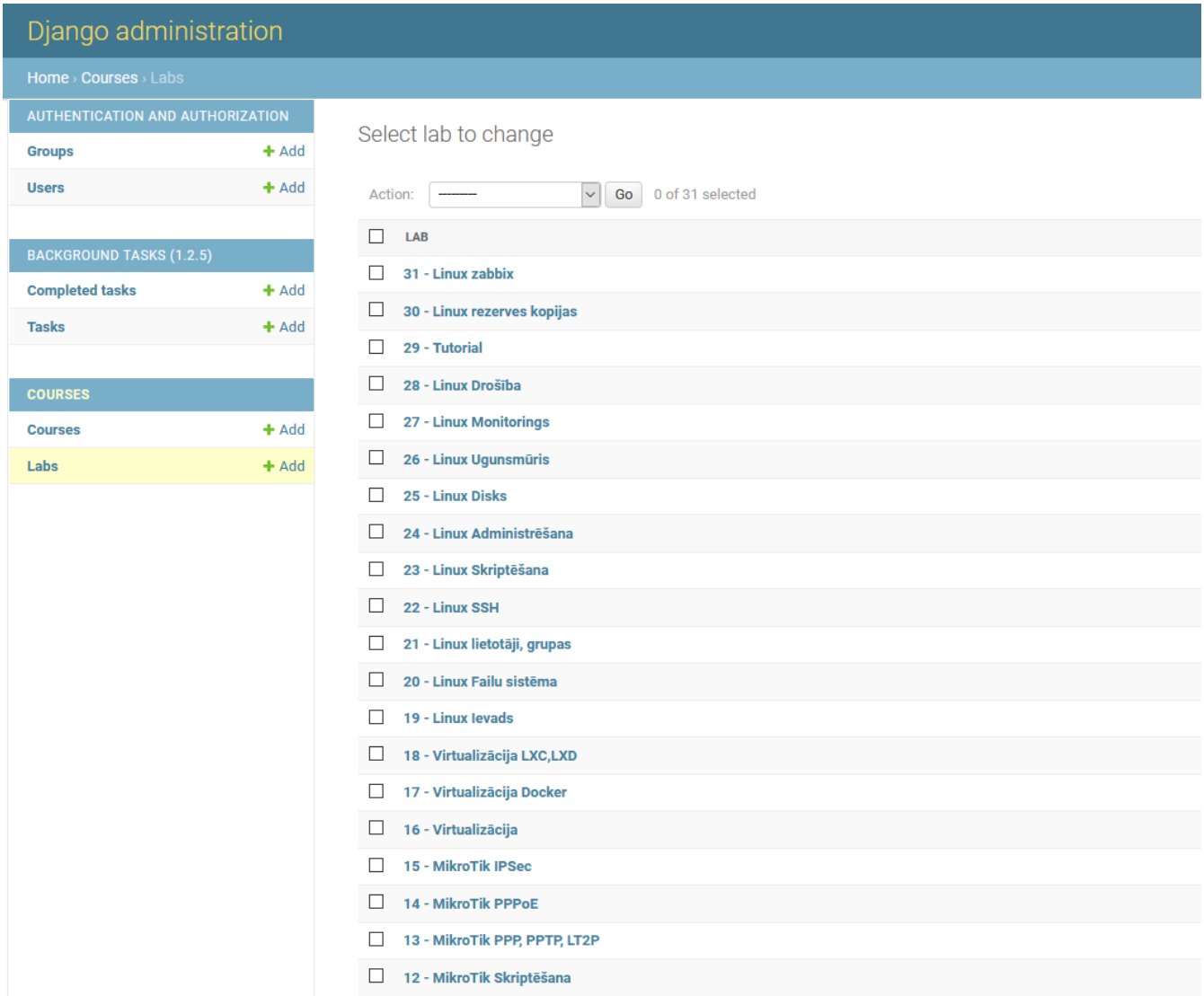
4.9. attēls – **Jaunās platformas izveidotas laboratorijas darba skats**



4.10. attēls – **Jaunās platformas laboratorijas darba iekārtas konfigurēšana**

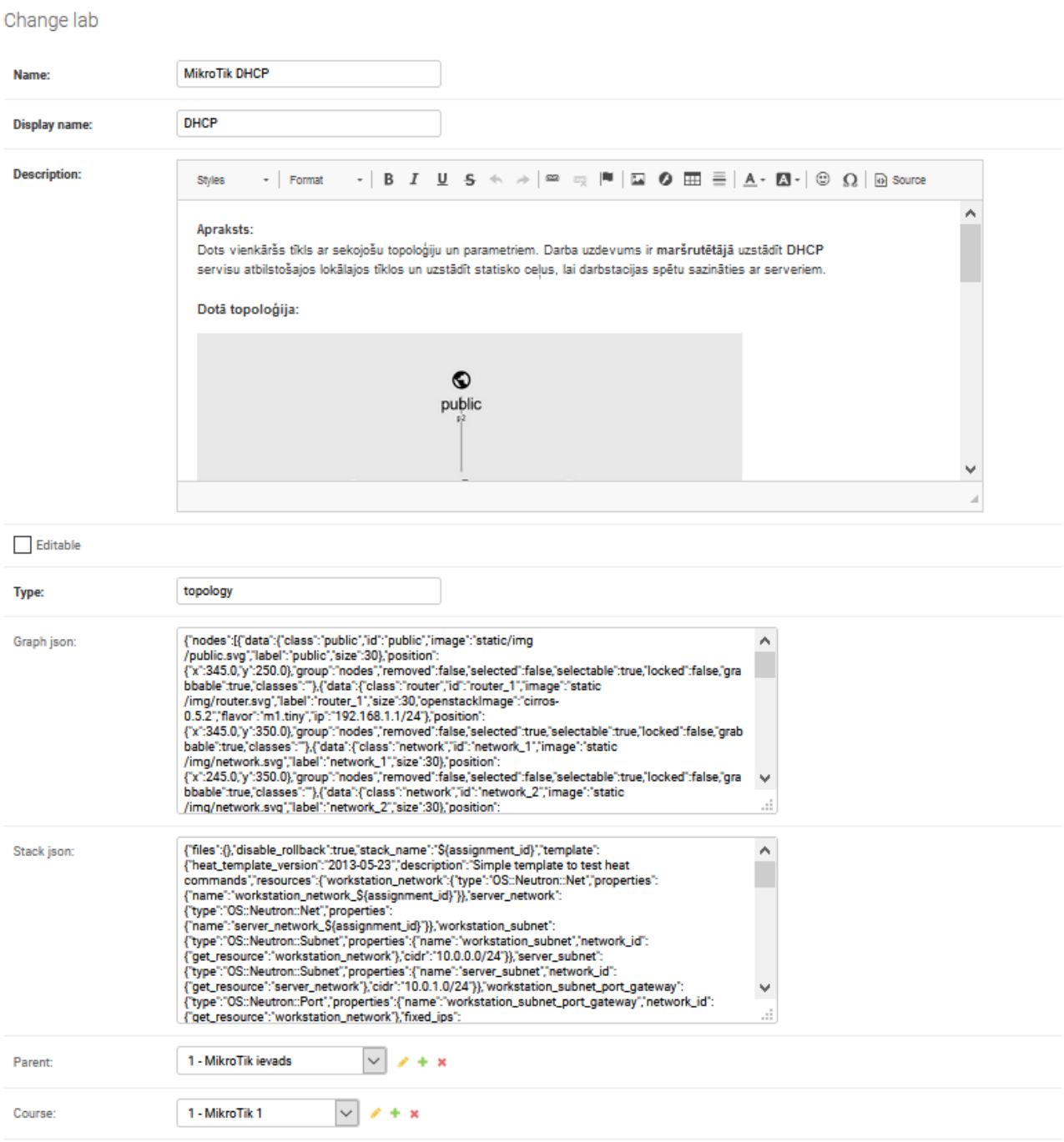
4.2.2. Administratora skati

Atbilstoši 2.2.2. nodaļā veiktajiem ieteikumiem, servera puses lietotne tika izstrādāta, izmantojot *Python* programmēšanas valodas *Django* ietvaru, kuram ir iekļauts atbalsts administratora skatu izveidei (skat. 4.11. attēlu), ar kuru administratoram ir iespējas rediģēt 4.1. nodaļā aprakstītās datubāzes entītijas, kā kursus, laboratorijas darbus un lietotājus.



4.11. attēls – **Jaunās platformas administratora skats**

Attēlā 4.12 ir redzams administratora skats *DHCP* laboratorijas darba rediģēšanai, kur ir iespējas rediģēt visus laboratorijas darba entītijas atribūtus, ieskaitot aprakstu, kurš redzams 4.6. attēlā.

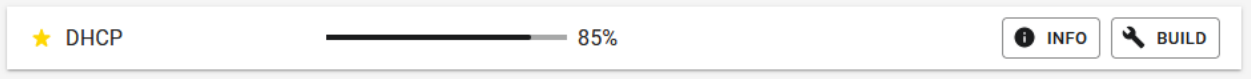


4.12. attēls – **Jaunās platformas administratora skats laboratorijas darba rediģēšanai**

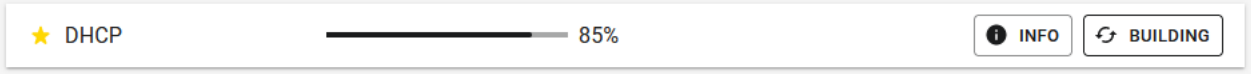
4.3. Izmantotie risinājumi

4.3.1. Reālā laika satura atjaunošana

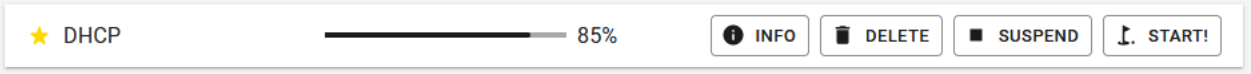
Pašreizējā platformā reālā laika satura atjaunošanas metodes ir nepieciešamas tikai lietotāja izveidoto laboratorijas darbu skatā (skat. 4.7. attēlu), lai lietotājam attēlotu pašreizējo laboratorijas darba topoloģijas izveides, dzēšanas, apturēšanas un atsākšanas statusu (skat. 4.13., 4.14. un 4.15. attēlus).



4.13. attēls – **Laboratorijas darbs lietotāja skatā pirms izveides**



4.14. attēls – **Laboratorijas darbs lietotāja skatā kamēr notiek tā izveide**



4.15. attēls – **Laboratorijas darbs lietotāja skatā pēc veiksmīgas izveides**

Jaunā platforma šim nolūkam izmanto *short-polling* metodi (skat. 2.4. nodaļu) gan klienta puses, gan servera puses lietotnē, kur serveris atkārtoti veic pieprasījumus *OpenStack* pakalpojumam par topoloģijas izveides statusu, un klienta puses lietotne atkārtoti veic pieprasījumus servera puses lietotnei. Klienta puses lietotnē šim nolūkam tiek izmantota *React* ietvaram izstrādāta *react-polling* bibliotēka (skat. pirmkoda pielietojumu 2. pielikumā), bet servera pusē tiek lietota *background-tasks* bibliotēka (skat. pirmkoda pielietojumu 3. pielikumā), kura nodrošina uzdevumu izpildi fonā un veicamos uzdevumus saglabā datubāzē, platformas izslēgšanas gadījumiem.

4.3.2. Laboratorijas darbu koka struktūra

Jaunajā platformā izstrādātajam kursu kokam (skat. 4.4. un 4.5. attēlus) tika apskatītas vairākas *JavaScript* koka struktūras izveides atbalsta bibliotēkas, kur pirmais bija *beautiful-skill-tree* bibliotēka, kuru ir viegli iekļaut platformā, bet tai ir ļoti maz pielāgošanas iespējas, lai izveidotās virsotnes aizstātu ar savām, un tā pieļauj tikai trīs virsotņu stāvokļus – izpildīts, pieejams un nepieejams –, bet jaunajā platformā ir nepieciešams arī ceturtais stāvoklis iesākta kursa apzīmēšanai. Otra apskatītā bibliotēka bija *react-tech-tree*, kura piedāvā lielākas pielāgošanas

iespējas, kā lietot savas virsotnes, bet tās metode elementu novietošanai ir ļoti primitīva, un lielus sarežģītus kokus ar vairākiem apakškokiem ir grūti attēlot. Gala secinājums bija veikt koka struktūras implementāciju pašrocīgi, kura galvenās implementācijas daļas ir koka virsotņu korekta attēlošana un virsotņu savienojumu zīmēšana.

Koka virsotņu attēlošana netiek veikta pilnībā dinamiski, atbilstoši datubāzē glabātajiem kursu, laboratorijas darbu un to vecāku datiem, bet tā paļaujas uz *JSON* formāta koka izkārtojuma apraksta failu (skat. 4. pielikumu). Šajā *JSON* aprakstā "*tutorialLab*" ieraksts norāda uz kokā attēlojamo saknes laboratorijas darbu, un "*trees*" saraksts apraksta kursu kolonnas. Piemēram, "*trees*" saraksta otrais ieraksts apraksta *Linux OS* un *Linux* Administrēšanas kursus (skat. 4.4. un 4.5. attēlus), kur šī ieraksta objekts ar atslēgu 6 apraksta *Linux* Administrēšanas kursu, norādot tā rāmja krāsu un laboratorijas darbu apakškoka izkārtojumu.

Virsotņu savienojumu zīmēšana notiek dinamiski, atbilstoši datubāzes datiem par laboratorijas darbiem un to vecākiem koka struktūrā (*parent* atribūts datubāzē). Savienojumu zīmēšanai pāri visam kokam tiek attēlota *svg HTML* komponente, kurā savienojumi tiek zīmēti ar *path HTML* komponentēm, kuras koordinātes tiek aprēķinātas ar 5. pielikumā redzamo algoritmu, kurš ņem vērā savienojamo virsotņu savstarpējās relatīvās atrašanās vietas. Piemēram, ja abu savienojamo virsotņu horizontālā distance ir vismaz 150 pikseļu vienības, tad abām virsotnēm tiek novilkti kāti, kuri tiek savienoti ar kvadrātisko Bezjē līkni (skat. 4.4. attēla platformas apmācības un *MikroTik 1* ievada laboratorijas darbu virsotņu savienojumu).

4.3.3. Izveidota laboratorijas darba topoloģijas attēlošana

Attēlā 4.9 redzamais laboratorijas darba topoloģijas grafs tiek attēlots, izmantojot *Cytoscape.js* grafu vizualizācijas bibliotēku, kurai ir pieejama *react-cytoscape.js React* ietvara atbalsta bibliotēka un, kura grafu attēlošanai pieņem *JSON* aprakstu par virsotnēm un to savienojumiem. *DHCP* laboratorijas darba grafa *JSON* apraksta fragments ir pieejams 6. pielikumā, kur "*edges*" ieraksti apraksta savienojumus un "*nodes*" ieraksti apraksta virsotnes, kurām, galvenokārt, tiek norādītas pozīcijas, identifikatori un attēls, kurš tiek uzturēts serverī. Šis apraksts tiek glabāts *graph_json* laboratorijas darba entītijas atribūtā, atbilstoši 4.1. nodaļas datubāzes projektējumam, un platformas administratora skatā (skat. 4.2.2. nodaļu) administratoriem to ir ērti izmainīt.

4.3.4. Saziņa ar *OpenStack*

Saziņai ar *OpenStack* tiek izmantotas 2.2.2. nodaļā piemeklētās *OpenStack Python* programmēšanas valodas atbalsta bibliotēkas *python-heatclient,* lai sazinātos ar *Heat* komponentes *API* virtuālu topoloģiju izveidošanai, un *python-novaclient*, lai iegūtu virtuālo iekārtu *VNC* protokola attālinātās piekļuves adreses (skat. *OpenStack* atbalsta bibliotēku pielietojumu 7. pielikumā).

Katra laboratorijas darba topoloģijas *JSON HOT* formāta apraksts tiek glabāts laboratorijas darbu *stack_json* atribūtā, atbilstoši 4.1. nodaļas datubāzes projektējumam, kurš tiek nosūtīts *OpenStack Heat* komponentei, lai doto topoloģiju izveidotu (skat. *DHCP* laboratorijas darba *JSON HOT* topoloģijas aprakstu 8. pielikumā). Papildus, vairākiem resursiem, kā maršrutētājiem un tīkliem, *OpenStack* ir nepieciešams nodrošināt unikālus nosaukumus, kas tiek paveikts ar identifikatoru formā "<profila_id>_<kursa_id>_<lab_id>", kurš *JSON HOT* topoloģijas aprakstos aizvieto visus sastaptos "${assignment_id}" laukus, pirms to nosūtīšanas *Heat* komponentei. Pēc topoloģijas veiksmīgas izveidošanas, *OpenStack* atgrieztais topoloģijas identifikators tiek saglabāts iesākto laboratorijas darbu *stack_id* datubāzes atribūtā, un atgrieztie izveidoto iekārtu identifikatori tiek saglabāti iesākto laboratorijas darbu *stack_output_json* datubāzes atribūtā, kuri tika pieprasīti ar *outputs* ierakstu *JSON HOT* topoloģijas aprakstā. Iekārtu identifikatori pēc tam tiek izmantoti, lai pieprasītu to *VNC* protokola attālinātās piekļuves adreses.

4.3.5. Izmantoto risinājumu trūkumu novērtējums

Viens no galvenajiem trūkumiem pašreizējā platformā ir *VNC* protokola izmantošana iekārtu attālinātai konfigurēšanai, jo tā tika izmantota pašreizējā *ReSeLa* platformā, un tā neiekļāva teksta iezīmēšanas, kopēšanas un ielīmēšanas funkcionalitātes, kas radīs lielu neērtību, lietotājiem veicot laboratorijas darbus, piemēram, sekojot kādiem materiāliem ar dotām komandām. Kā tika secināts 2.5. un 3.2. nodaļās, labākais risinājums ir veidot seriālās konsoles savienojumu ar iekārtām, lai šo funkcionalitāti nodrošinātu.

Reālā laika satura atjaunošanai pašreizējā platforma izmanto *short-polling* metodi, lai lietotājam attēlotu pašreizējo topoloģiju izveides statusu. Kā jau tika izskatīts 2.4. nodaļā, *short-polling* ir visprimitīvākais risinājums ar vislielāko slodzi serverim, un to ir nepieciešams aizvietot ar vienu no *SSE* vai *WebSocket* protokola risinājumiem, kur atbilstošais savienojums tiek izveidots un uzturēts visas lietotāja sesijas laikā. Neskatoties uz to, pašreizējā platformā *short-polling* tiek izmantots reti, jo tas ir nepieciešams tikai laboratorijas darba izveides, dzēšanas,

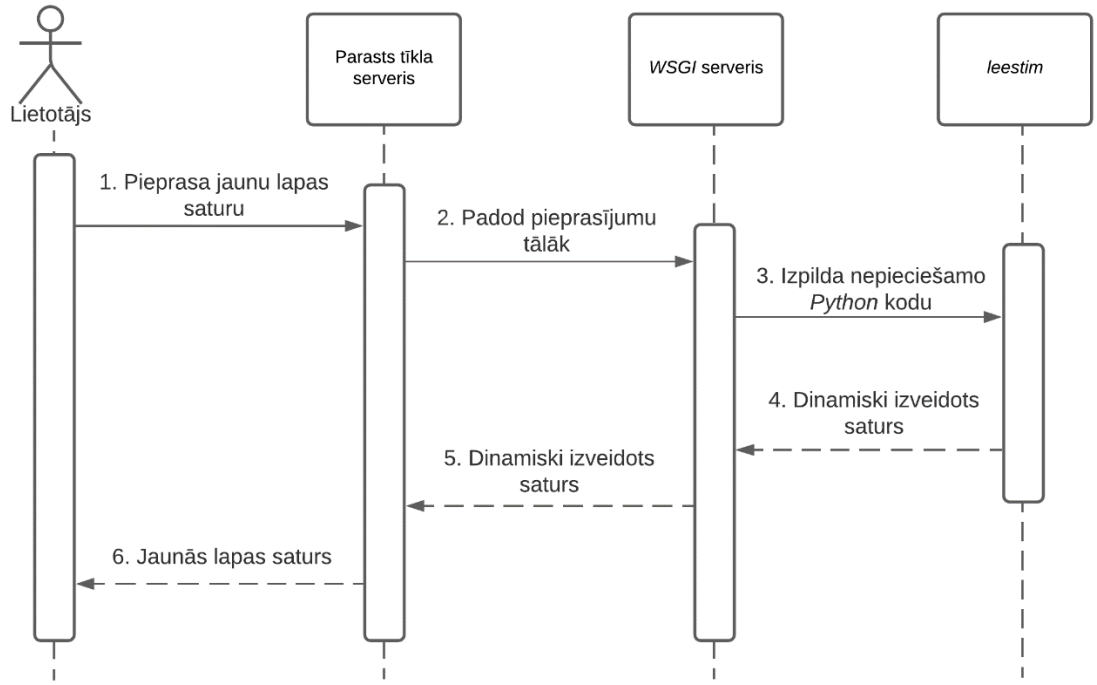
apturēšanas vai atsākšanas brīdī, kur lietotājs drīkst izveidot tikai vienu topoloģiju vienlaikus, un topoloģijas izveides laiks testa vidē netika konstatēts pārāk liels (40 sekundes), izmantojot *DevStack* uz virtuālās mašīnas ar ierobežotiem resursiem.

Laboratorijas darbu koka struktūra, kura tiek attēlota lietotājam, pašlaik netiek dinamiski izveidota, bet ir nepieciešams atbalsta *JSON* fails, kurā jānorāda koka virsotņu atrašanās vietas. Pašlaik, ja administratori veic jauna kursa vai laboratorijas darba pievienošanu, tiem ir nepieciešams apgūt izmantoto *JSON* koka apraksta failu un veikt izmaiņas arī tur, lai tas pareizi attēlotos lietotājiem. Papildus, kokā pašlaik ir iespēja vecākam pievienot vairākus bērnus, bet bērnam nav iespējami vairāki vecāki, tāpēc sarežģītākas koka struktūras nav iespējamas, kas būtu nepieciešams, lai uzlabotu koka struktūras kā spēļu elementa efektivitāti.

4.4. Platformas izvietošana

Lai jauno izstrādāto platformu radītu pieejamu jebkuram lietotājam, to ir nepieciešams izvietot publiskajā tīklā. Mūsdienās *Python* programmēšanas valodā izstrādātu platformu izvietošanai ir nepieciešams specifisks uzstādījums, kur pieņemtais risinājums ir starp lietotāju un izveidoto platformu novietot *WSGI* serveri [46]. *WSGI* (*Web Server Gateway Interface*) ir standarts, kurš nosaka saskarni starp *Python* tīkla lietotni un tīkla serveri, kā *Apache*, lai *Python* programmas varētu izstrādāt bez nepieciešamības to pielāgot katram tīkla serverim atsevišķi [46, 47]. Jaunajā platformā izmantotais *Django* ietvars jau nodrošina platformas atbilstību *WSGI* protokolam [48], un populārākie tīkla serveri [46], kuri nodrošina *WSGI* standartu, ir *Green Unicorn*, *uWSGI* un *mod_wsgi*, kurš ir papildinājums *Apache* tīkla serverim.

Papildus, pieņemtais izvietojuma risinājums iekļauj vēl viena parasta tīkla servera, kā *Apache* vai *Nginx*, novietošanu starp klientu un *WSGI* serveri [46, 49] (skat. 4.16. attēlu), jo *WSGI* serveri bieži nenodrošina pārāk lielu funkcionalitāšu klāstu ārpus *WSGI* protokola, un tiek zaudētas citas uzstādījumu iespējas. Attēlā 4.16. ir redzams, ka lietotājs pieprasa jaunu lapas saturu un, lai to iegūtu no *Python* lietotnes, pieprasījumam ir jāiet cauri parastajam tīkla serverim un *WSGI* tīkla serverim.



4.16. attēls – **Jaunās platformas izvietošanas struktūra[46]**

Tā kā *WSGI* serveri spēj nodrošināt paralēlu pieprasījumu apstrādi [50], un starp lietotāju ir vēl viens parasts tīkla serveris, tad mērogojamības nolūkos ir pieejami visi standarta risinājumi [51], kā divu tīkla serveru izvietošanu ar *load balancer*.

4.5. Servera *API* apraksts

Šajā nodaļā tiek dokumentēts servera puses lietotnes *API*, ar kuru sazinās klienta puses lietotne un, kurš, atbilstoši 2.3. nodaļas ieteikumiem, ir izstrādāts pēc *REST* pieejas un *JSON API* standarta. Visos zemāk aprakstītajos *API* galapunktos, kuri atgriež sarakstu ar resursiem, atbilde tiek sadalīta "lapās", kur katra lapa ir jāpieprasa atsevišķi, lai viens pieprasījums nesaturētu pārāk lielu datu apjomu. Katras lapas atbilde papildus iekļauj 4.1. tabulā redzamos laukus.

4.1. tabula – **Resursu saraksta atbildes lapu dati**

Lauka nosaukums	Lauka formāts	Lauka apraksts
`meta.pagination`	*JSON Object*	Satur datus par pašreizējās atbildes lapu
`meta.pagination.page`	*Integer*	Pašreizējās atbildes lapas numurs
`meta.pagination.pages`	*Integer*	Kopējais atbildes lapu skaits

`meta.pagination.count`	*Integer*	Ierakstu skaits kopā pa visām atbildes lapām
`links.first`	*String*	Saite uz pirmo atbildes lapu
`links.last`	*String*	Saite uz pēdējo atbildes lapu
`links.next`	*String*	Saite uz nākamo atbildes lapu
`links.prev`	*String*	Saite uz iepriekšējo atbildes lapu

Piemērs:

```
{
    "data": [
        ...
        ...
        ...
        ...
        ...
    ],
    "meta": {
        "pagination": {
            "page": 1,
            "pages": 4,
            "count": 20
        }
    },
    "links": {
        "first": "http://localhost:8000/api/assignments/?page%5Bnumber%5D=1",
        "last": "http://localhost:8000/api/assignments/?page%5Bnumber%5D=4",
        "next": "http://localhost:8000/api/assignments/?page%5Bnumber%5D=2",
        "prev": null
    }
}
```

4.5.1. Kursu datu iegūšana

URL	`/api/courses/<course_id>`
***HTTP* metode**	`GET`
Piezīme	Ja `<course_id>` saitē tiek norādīts, tad tiek atgriezti dati tikai par attiecīgo kursu, pretēji tiek atgriezts saraksts ar visiem kursiem.

Atbildes *JSON* dati

Lauka nosaukums	Lauka formāts	Lauka apraksts
`data`	*JSON Object*	Satur datus par kursu
`data.type`	*String*	Vērtība sakrīt ar "*course*"
`data.id`	*String*	Kursa identifikators
`data.attributes`	*JSON Object*	Kursa atribūti

`data.attributes.name`	*String*	Kursa nosaukums
`data.attributes.display_name`	*String*	Kursa nosaukums, kuru ir paredzēts attēlot lietotājam
`data.attributes.description`	*String*	Kursa apraksts
`data.relationships.labs`	*JSON Object*	Kursam piederošie laboratorijas darbi
`data.relationships.labs .meta.count`	*Integer*	Kursam piederošo laboratorijas darbu skaits
`data.relationships.labs .data`	*JSON Array*	Saraksts, ar kursam piederošajiem laboratorijas darbiem
`data.relationships.labs .data[n].type`	*String*	Sakrīt ar vērtību "*lab*"
`data.relationships.labs .data[n].id`	*String*	Kursam piederošā laboratorijas darba identifikators
Atbildes piemērs, ja <course_id> saitē nav norādīts		

```
{
    "data": [
        {
            "type": "course",
            "id": "1",
            "attributes": {
                "name": "MikroTik 1",
                "display_name": "MikroTik 1",
                "description": "This is a description for MikroTik 1"
            },
            "relationships": {
                "labs": {
                    "meta": {
                        "count": 8
                    },
                    "data": [
                        {
                            "type": "lab",
                            "id": "1"
                        },
                        {
                            "type": "lab",
                            "id": "2"
                        },
                        ...
                        ...
                    ]
                }
            }
        },
        ...
        ...
    ],
    "meta": {
        "pagination": {
            "page": 1,
            "pages": 1,
            "count": 7
        }
    },
    "links": {
        "first": "http://localhost:8000/api/courses/?page%5Bnumber%5D=1",
        "last": "http://localhost:8000/api/courses/?page%5Bnumber%5D=1",
        "next": null,
        "prev": null
    }
}
```


4.5.2. Laboratorijas darbi

4.5.2.1. Laboratorijas darbu datu iegūšana

URL	`/api/labs/<lab_id>`
***HTTP* metode**	`GET`
Piezīme	Ja `<lab_id>` saitē tiek norādīts, tad tiek atgriezti dati tikai par attiecīgo laboratorijas darbu, pretēji tiek atgriezts saraksts ar visiem laboratorijas darbiem.
Atbildes *JSON* dati	

Lauka nosaukums	Lauka formāts	Lauka apraksts
`data`	*JSON Object*	Satur datus par laboratorijas darbiem
`data.type`	*String*	Vērtība sakrīt ar "*lab*"
`data.id`	*String*	Laboratorijas darba identifikators
`data.attributes`	*JSON Object*	Laboratorijas darba atribūti
`data.attributes.name`	*String*	Laboratorijas darba nosaukums
`data.attributes.display_name`	*String*	Laboratorijas darba nosaukums, kuru paredzēts attēlot lietotājam
`data.attributes.type`	*String*	Laboratorijas darba tips, satur vērtību "*topology*" vai "*tutorial*". Nākotnē var tikt pievienotas papildus vērtības, lai noteiktu, kuriem laboratorijas darbiem lietotājs drīkst rediģēt topoloģiju.
`data.relationships.parent`	*JSON Object*	Laboratorijas darba "vecāka" laboratorijas darbs, lai noteiktu, kad šo ir atļauts iesākt.
`data.relationships.parent .data.type`	*String*	Sakrīt ar vērtību "*lab*"
`data.relationships.parent data.id`	*String*	Vecāka laboratorijas darba identifikators
`data.relationships.course`	*JSON Object*	Kurss, kuram pieder šis laboratorijas darbs
`data.relationships.course .data.type`	*String*	Sakrīt ar vērtību "*course*"
`data.relationships.course .data.id`	*String*	Piederošā kursa identifikators
Atbildes piemērs, ja <lab_id> saitē netiek norādīts		

```
{
    "data": [
        {
            "type": "lab",
            "id": "1",
            "attributes": {
                "name": "MikroTik ievads",
                "display_name": "Ievads",
                "type": "topology"
            },
            "relationships": {
                "parent": {
                    "data": {
                        "type": "lab",
                        "id": "29"
                    }
                },
                "course": {
                    "data": {
                        "type": "course",
                        "id": "1"
                    }
                }
            }
        },
        ...
        ...
    ],
    "meta": {
        "pagination": {
            "page": 1,
            "pages": 4,
            "count": 31
        }
    },
    "links": {
        "first": "http://localhost:8000/api/labs/?page%5Bnumber%5D=1",
        "last": "http://localhost:8000/api/labs/?page%5Bnumber%5D=4",
        "next": "http://localhost:8000/api/labs/?page%5Bnumber%5D=2",
        "prev": null
    }
}
```


4.5.2.2. Laboratorijas darba apraksta iegūšana

URL	`/api/labs/<lab_id>/description/`	
***HTTP* metode**	`GET`	
Piezīme	Laboratorijas darba apraksti parasti ir apjomīgi, tāpēc tie tiek atdalīti no 4.5.2.1. pieprasījuma, lai aprakstu var iegūt tikai, kad tas ir nepieciešams.	
Atbildes *JSON* dati		
Lauka nosaukums	Lauka formāts	Lauka apraksts
`data.description`	*String*	*HTML* laboratorijas darba apraksts iekodēts *String* datu tipā
Atbildes piemērs		

```
{
    "data": {
        "description": "<p><strong>Apraksts:</strong><br />\r\nDots vienkār&sca
    }
}
```


4.5.2.3. Laboratorijas darba topoloģijas grafa datu iegūšana

URL	`/api/labs/<lab_id>/graph/`	
***HTTP* metode**	`GET`	
Piezīme	Laboratorijas darba topoloģijas grafa dati parasti ir apjomīgi, tāpēc tie tiek atdalīti no 4.5.2.1. pieprasījuma, lai tos var iegūt tikai, kad tas ir nepieciešams.	
Atbildes *JSON* dati		
Lauka nosaukums	Lauka formāts	Lauka apraksts
`data.graph`	*String*	Topoloģijas grafam nepieciešamais *JSON* fails iekodēts *String* datu tipā
Atbildes piemērs		


```
{
    "data": {
        "graph": "{\"nodes\":[{\"data\":{\"class\":\"public\",\
    }
}
```


4.5.3. Lietotāja iesāktie laboratorijas darbi

4.5.3.1. Iesākto laboratorijas darbu datu iegūšana

URL	`/api/assignments/<course_id>/labs/<lab_id>`	
***HTTP* metode**	`GET`	
Piezīme	Ja saitē tiek norādīts `/<course_id>/labs/<lab_id>`, tad tiek atgriezti dati par attiecīgo iesākto laboratorijas darbu attiecīgajā kursā. Ja saitē tiek norādīta tikai līdz `/<course_id>/`, tad tiek atgriezts saraksts ar visiem lietotāja iesāktajiem laboratorijas darbiem attiecīgajā kursā. Ja saite tiek norādīta tikai līdz `/api/assignments/`, tad tiek atgriezts saraksts ar visiem lietotāja iesāktajiem laboratorijas darbiem visos kursos.	
Atbildes *JSON* dati		
Lauka nosaukums	Lauka formāts	Lauka apraksts
`data`	*JSON Object*	Satur datus par iesākto laboratorijas darbu

`data.type`	*String*	Vērtība sakrīt ar "*assignment*"
`data.id`	*String*	Iesāktā laboratorijas darba identifikators
`data.attributes`	*JSON Object*	Iesāktā laboratorijas darba atribūti
`data.attributes.complete_state`	*String*	Iesāktā laboratorijas darba statuss, sakrīt ar vienu no "*started*" vai "*completed*"
`data.attributes.build_state`	*String*	Iesāktā laboratorijas darba topoloģijas izveides statuss, sakrīt ar vienu no: • "*COMPLETED*"; • "*SUSPENDED*"; • "*DELETE_IN_PROGRESS*"; • "*SUSPEND_IN_PROGRESS*"; • "*RESUME_IN_PROGRESS*"; • *null* (ja nav veikta topoloģijas izveide)
`data.attributes.score`	*Integer*	Iesāktā laboratorijas darba pašreizējais vērtējums. **Pašreiz netiek izmantots un vērtība sakrīt ar 0**.
`data.relationships.course`	*JSON Object*	Iesāktā laboratorijas darba piederošais kurss
`data.relationships.course.type`	*String*	Sakrīt ar vērtību "*course*"
`data.relationships.course.id`	*String*	Piederošā kursa identifikators
`data.relationships.lab`	*JSON Object*	Laboratorijas darbs, uz kuru attiecas šie dati
`data.relationships.lab.type`	*String*	Sakrīt ar vērtību "*lab*"
`data.relationships.lab.id`	*String*	Atbilstošā laboratorijas darba identifikators
Atbildes piemērs, ja tiek atgriezts saraksts		

```
{
    "data": [
        {
            "type": "assignment",
            "id": "29",
            "attributes": {
                "complete_state": "completed",
                "build_state": null,
                "score": 0
            },
            "relationships": {
                "course": {
                    "data": {
                        "type": "course",
                        "id": "7"
                    }
                },
                "lab": {
                    "data": {
                        "type": "lab",
                        "id": "29"
                    }
                }
            }
        },
        ...
        ...
    ],
    "meta": {
        "pagination": {
            "page": 1,
            "pages": 4,
            "count": 20
        }
    },
    "links": {
        "first": "http://localhost:8000/api/assignments/?page%5Bnumber%5D=1",
        "last": "http://localhost:8000/api/assignments/?page%5Bnumber%5D=4",
        "next": "http://localhost:8000/api/assignments/?page%5Bnumber%5D=2",
        "prev": null
    }
}
```


4.5.3.2. Jauna laboratorijas darba iesākšana

URL	`/api/assignments/`	
***HTTP* metode**	`POST`	
Pieprasījuma *JSON* dati		
Lauka nosaukums	Lauka formāts	Lauka apraksts
`data`	*JSON Object*	Satur laboratorijas darba iesākšanas datus
`data.type`	*String*	Vērtība sakrīt ar "*assignment*"
`data.relationships.course`	*JSON Object*	Iesāktā laboratorijas darba piederošais kurss
`data.relationships.course.type`	*String*	Sakrīt ar vērtību "*course*"
`data.relationships.course.id`	*String*	Piederošā kursa identifikators

`data.relationships.lab`	*JSON Object*	Laboratorijas darbs, kurš tiek iesākts
`data.relationships.lab.type`	*String*	Sakrīt ar vērtību "*lab*"
`data.relationships.lab.id`	*String*	Atbilstošā laboratorijas darba identifikators
Atbildes *JSON* dati	Sakrīt ar 4.5.3.1. atbildi, kas satur datus par attiecīgo jauno iesākto laboratoriju.	

Pieprasījuma piemērs

```
{
  "data": {
    "type": "assignment",
    "relationships": {
      "course": {
        "data": { "type": "course", "id": "1" }
      },
      "lab": {
        "data": { "type": "lab", "id": "3" }
      }
    }
  }
}
```

Atbildes piemērs

```
{
    "data": {
        "type": "assignment",
        "id": "3",
        "attributes": {
            "complete_state": "started",
            "build_state": null,
            "score": 0
        },
        "relationships": {
            "course": {
                "data": {
                    "type": "course",
                    "id": "1"
                }
            },
            "lab": {
                "data": {
                    "type": "lab",
                    "id": "3"
                }
            }
        }
    }
}
```

4.5.3.3. Iesākta laboratorijas darba *OpenStack* topoloģijas darbības pieprasīšana

URL	`/api/assignments/<course_id>/labs/<lab_id>`	
***HTTP* metode**	`PATCH`	
Pieprasījuma *JSON* dati		
Lauka nosaukums	Lauka formāts	Lauka apraksts
`data`	*JSON Object*	Satur iesākta laboratorijas darba darbības datus
`data.type`	*String*	Vērtība sakrīt ar "*assignment*"
`data.id`	*String*	Iesāktā laboratorijas darba identifikators
`data.attribues.build_state`	*String*	Atbilstoši veicamajai darbībai ar topoloģiju, vērtība sakrīt ar vienu no: • "*BUILD_IN_PROGRESS*", • "*DELETE_IN_PROGRESS*", • "*SUSPEND_IN_PROGRESS*", • "*RESUME_IN_PROGRESS*"
Atbildes *JSON* dati	Sakrīt ar 4.5.3.1. atbildi, kas satur datus par attiecīgo iesākto laboratoriju.	
Pieprasījuma piemērs		

```
{
    "data": {
        "id": "3",
        "type": "assignment",
        "attributes": {
            "build_state": "BUILD_IN_PROGRESS"
        }
    }
}
```

Atbildes piemērs

```
{
    "data": {
        "type": "assignment",
        "id": "29",
        "attributes": {
            "complete_state": "started",
            "build_state": "BUILD_IN_PROGRESS",
            "score": 0
        },
        "relationships": {
            "course": {
                "data": {
                    "type": "course",
                    "id": "1"
                }
            },
            "lab": {
                "data": {
                    "type": "lab",
                    "id": "29"
                }
            }
        }
    }
}
```


4.5.3.4. Izveidota laboratorijas darba virtuālās iekārtas piekļuves adreses iegūšana

URL	`/api/assignments/<course_id>/labs/<lab_id>/instances/?instance-id=<iekārtas_id>`
***HTTP* metode**	`GET`

Atbildes *JSON* dati

Lauka nosaukums	Lauka formāts	Lauka apraksts
`data.url`	*String*	Saite, kurā lietotājam atveras *VNC* komandrindas logs.

Pieprasījuma piemērs

```
/api/assignments/<course_id>/labs/<lab_id>/instances
/?instance-id=workstation_1
```

Atbildes piemērs


```
{
    "data": {
        "url": "http://192.168.0.98:6080/vnc_lite.html?path=%3Ftoken%3D40a
    }
}
```

REZULTĀTI

Dokumenta rezultātā, tika veikts apraksts par pašreizējo *ReSeLa* platformu, un izpētītas vairākas iespējas tās veicamajiem uzlabojumiem atjaunotajā platformā. Viens no veicamajiem uzlabojumiem iekļauj dizainu un funkcionalitātes, kur tika apskatīta šī paša autora kursa darbā [1] un Ievas Lapiņas bakalaura darbā [3] veiktā analīze par citu populāru mācību platformu iezīmēm un apkopoti galvenie rezultāti un projektējumi, lai virzītu jaunās platformas izstrādi.

Līdz ar platformas pilnīgu pārveidi, tika pētītas iespējas izmantot modernākas un aktuālākas izstrādes pieejas un programmēšanas valodas, kā nolūkam tika apskatītas *OpenStack* atbalsta bibliotēkas un to izmantotās programmēšanas valodas, kā arī tika sekots *JetBrains* veiktai *The State of Developer Ecosystem 2020* aptaujai par populārākajām izstrādātāju izmantotajām tehnoloģijām.

Pēc veiktajiem secinājumiem par nepieciešamību platformai izstrādāt *API*, tika apskatītas pieejamās un populārākās *API* izstrādes pieejas, lai būtu mazāk neskaidrību un domstarpību par tā struktūru un datu formātu.

Pašreizējā *ReSeLa* platforma nepiedāvā reālā laikā atjaunotas informācijas pasniegšanu lietotājam, kā, piemēram, pašreizējo laboratorijas darbu topoloģiju izveides statusu, tāpēc tika veikts pētījums par šādas funkcionalitātes realizēšanas tehnoloģijām un pieejām, kas iekļāva *short-polling*, *long-polling*, *SSE* (*Server-Sent Events*) un *WebSocket* protokolu, kur katram ir savi ieguvumi un trūkumi.

Viena no galvenajām pašreizējās *ReSeLa* platformas piedāvātajām funkcionalitātēm ir mājaslapā iekļauts virtuālo iekārtu attālinātās konfigurēšanas komandrindas logs, izmantojot *VNC* protokolu. Lai novērstu *VNC* protokola trūkumus, dokumentā tika apskatīti visi *OpenStack* piedāvātie risinājumi attālinātai iekārtu piekļuvei – *VNC*, *SPICE* un seriālais savienojums –, iekļaujot aprakstu par katra darbību, ieguvumiem un trūkumiem, un piemērotību jaunai platformai.

Jaunās platformas izstrādei tika veikta *DevStack* uzstādīšana, kas ir samazināta *OpenStack* pakalpojuma komplektācija, lai izstrādātu jaunās platformas laboratorijas darbu virtuālo topoloģiju realizēšanu atbilstoši *JSON HOT* formāta topoloģijas aprakstiem.

Noslēgumā, tika izstrādāta jaunās platformas sākuma versija, kura iekļāva funkcionalitātes un dizaina iezīmes, kā laboratorijas darbu koka struktūras spēļu elementu, laboratorijas darbu topoloģiju realizēšanu ar *OpenStack*, topoloģiju attēlošanu grafu veidā, attālinātu virtuālo iekārtu piekļuvi un reālā laika pašreizējo topoloģiju statusu attēlošanu. Dokumentā tika dots jaunās platformas struktūras un datubāzes projektējums, aprakstīts dizains un iekļautās funkcionalitātes, aprakstīti visi sarežģītākie izmantotie platformas risinājumi un to pašreizējie trūkumi, aprakstītas iespējas platformas izvietošanai publiskajā tīklā un mērogojamībai, kā arī tika dokumentēts servera *API*.

SECINĀJUMI

Pēc pašreizējās *ReSeLa* platformas apraksta un citu populāru mācību platformu dizaina iezīmju un funkcionalitāšu apkopojuma, tika secināts, ka pašreizējā platformā trūkst vairākas iezīmes, kā pilnvērtīgākas sadarbības iespējas starp pasniedzēju un studentiem ar virtuālo klašu palīdzību, skaidru vizuālu elementu izmantošana laboratorijas darbu un kursu progresa attēlošanai, plašāks pieejamo materiālu klāsts, ieskaitot lasāmvielu, video materiālus un testus, kā arī ir ieteicams iekļaut spēļu elementus lietotāju motivācijas veicināšanai. Spēļu elementu iekļaušanai tika secināts izmantot nopelnāmas medaļas par noteiktu aktivitāšu paveikšanu, un laboratorijas darbu attēlošanu koka struktūrā, lai nodrošinātu spēles gaitas sajūtu, atbilstoši izstrādātajai kursu un laboratorijas darbu programmai.

Populārāko tīkla lietotņu programmēšanas valodu pētījumu rezultātā, tika secināts, ka populārākā pieeja ir lietotni sadalīt divās daļās – klientu puse un serveru puse –, kur klientu puse tiek izstrādāta ar vienu-lapu pieeju, izmantojot vienu no *JavaScript* programmēšanas valodas *React* vai *Vue.js* ietvariem. Vienu-lapu pieeja izrādījās ļoti piemērota jaunajai platformai, un vairāki no tās ieguvumiem ir slodzes samazināšana serverim, kā arī tiek nodrošināts atsevišķas datorprogrammas izskats, jo netiek veikta atkārtota lapas ielāde, kas ir svarīgi spēles vides sajūtas radīšanai. Savukārt, vienu-lapu lietotnes pieejai ir vairāki trūkumi, kā izstrādes sarežģītība un drošības riski. Klienta puses lietotnes izstrādei tika secināts izmantot *React* ietvaru tās popularitātes dēļ. Atbilstošā servera izstrādei tika secināts izmantot *Python* programmēšanas valodu, jo tajā ir pieejamas *OpenStack* atbalsta bibliotēkas, un tās populārākie ietvari ir *Django* un *Flask*. Jaunās platformas izstrādei tika izvēlēts lietot *Django* ietvaru, jo tas nodrošināja daudz lielāku funkcionalitāšu klāstu nekā *Flask*, kurš paļaujas uz lielu trešo pušu bibliotēku izmantošanu lielas lietotnes izstrādes gadījumā.

Līdz ar secinājumu lietotni sadalīt divās daļās, tika veikti pētījumi par servera *API* izstrādes pieejām, kur populārākā ir *REST API*, kas nosaka adrešu strukturēšanu atbilstoši servera resursiem un atbilstošu *HTTP* metožu un atbilžu kodu izmantošanu. Papildus, tika secināts izmantot *JSON API* standartu, lai novērstu neskaidrības un domstarpības par *JSON* datu formātu.

Apskatot pieejas reālā laika satura atjaunošanai, tika atrasti četri risinājumi, izmantojot *short-polling*, *long-polling*, *SSE* (*Server-Sent Events*) vai *WebSocket* protokolu. Secinājumi rāda, ka, ar *HTTP* protokolu ir grūti nodrošināt noturīgu vairāku ziņojumu kanālu starp klientu un serveri, un efektīvākie risinājumi ir *SSE* vienvirziena kanālam un *WebSocket* protokols vienvirziena vai divvirziena kanāliem. *Long-polling* metode ir efektīva, ja ir nepieciešams saņemt mazu skaitu jaunu datu ziņojumus no servera, bet *short-polling* vienīgais ieguvums ir tā vienkāršība.

Pēc izpētes par *OpenStack* piedāvātajiem *VNC*, *SPICE* un seriālās konsoles savienojuma attālinātās piekļuves risinājumiem, tika secināts, ka visu šo risinājumu darbība ir ļoti līdzīga – *proxy* pakalpojums, kurš atrodas *OpenStack* sistēmā tiek izmantots kā starpnieks, lai virtuālās iekārtas saskarnes datus caur *WebSocket* kanālu nosūtītu klientu programmai lietotāja pārlūkā, kas attiecīgos datus spēj nolasīt, lietotājam attēlot un pretī nosūtīt lietotāja ievadīto informāciju. Pēc *DevStack* testa vides uzstādīšanas un šo risinājumu eksperimentu veikšanas, tika konstatēts, ka tikai seriālās konsoles savienojuma risinājums nodrošināja galveno pašreizējās platformas trūkumu, lai attālinātās piekļuves tekstuālajā saskarnē tekstu būtu iespējams iezīmēt, kopēt un ielīmēt. Lai seriālās konsoles savienojumu ieviestu, netika atrasta izmantojama klienta programma, bet tā izstrādei tika piemeklēta populāra atbalsta bibliotēka *Xterm.js*.

Nobeigumā, izstrādātā jaunā platforma iekļāva vairākus dokumentā minētos nepieciešamos uzlabojumus, bet tika konstatēti vairāki trūkumi, kurus nepieciešams risināt turpmāk, kā koku struktūras dinamisku attēlošanu, balstoties uz datubāzes datiem, *short-polling* metodes aizvietošanu un seriālās konsoles savienojuma risinājuma izmantošanu *VNC* protokola vietā. Neskatoties uz to, jaunajā platformā jau ir vērojami būtiski uzlabojumi, salīdzinot ar pašreizējo *ReSeLa* platformu, un ir izveidota visa funkcionalitāte, lai lietotāji varētu sākt apgūt datortīklu iekārtas un to konfigurēšanu. Nākamie soļi ir veikt minētos uzlabojumus un ieviest platformu produkcijā ar mazu lietotāju skaitu, piemēram, Latvijas Universitātes Datorikas fakultātes datortīklu kursu studentiem.

IZMANTOTĀ LITERATŪRA UN AVOTI

[1] R. E. Braunfelds, «Tehnoloģiju un metožu izpēte mācību platformu veidošanā,» kursa darbs, Datorikas fakultāte, Latvijas Universitāte, Rīga, 2021.

[2] I. Lapiņa, «MikroTik virtuālās laboratorijas ReSeLa vidē,» kvalifikācijas darbs, Datorikas fakultāte, Latvijas Universitāte, Rīga, 2016.

[3] I. Lapiņa, «Spēliskošanas elementi it mācību sistēmā,» bakalaura darbs, Datorikas fakultāte, Latvijas Universitāte, Rīga, 2018.

[4] JetBrains, «The State of Developer Ecosystem in 2020 Infographic,» 2020. [Tiešsaiste]. Pieejams: https://www.jetbrains.com/lp/devecosystem-2020/. [Piekļūts 29.05.2021].

[5] JetBrains, «JavaScript Programming - The State of Developer Ecosystem in 2020 Infographic,» 2020. [Tiešsaiste]. Pieejams: https://www.jetbrains.com/lp/devecosystem-2020/javascript/. [Piekļūts 29.05.2021].

[6] K. Lawson, «What Is a Single Page Application and Why Do People Like Them so Much?,» [Tiešsaiste]. Pieejams: https://www.bloomreach.com/en/blog/2018/07/what-is-a-single-page-application.html. [Piekļūts 29.05.2021].

[7] W. Ezell, «What is a Single Page Application? (And Should You Use One?),» 11.01.2018. [Tiešsaiste]. Pieejams: https://dotcms.com/blog/post/what-is-a-single-page-application-and-should-you-use-one-. [Piekļūts 29.05.2021].

[8] T. Rascoa, «React Tutorial: An Overview and Walkthrough,» 20.08.2018. [Tiešsaiste]. Pieejams: https://www.taniarascia.com/getting-started-with-react/. [Piekļūts 29.05.2021].

[9] OpenStack, «OpenStackClients,» [Tiešsaiste]. Pieejams: https://wiki.openstack.org/wiki/OpenStackClients. [Piekļūts 29.05.2021].

[10] OpenStack, "OpenStack Docs: Python bindings to the OpenStack Heat API," [Tiešsaiste]. Pieejams: https://docs.openstack.org/python-heatclient/latest/. [Piekļūts 29.05.2021].

[11] JetBrains, «Python Programming - The State of Developer Ecosystem in 2020 Infographic,» 2020. [Tiešsaiste]. Pieejams: https://www.jetbrains.com/lp/devecosystem-2020/python/. [Piekļūts 29.05.2021].

[12] J. Richer, «End User Authentication with OAuth 2.0,» [Tiešsaiste]. Pieejams: https://oauth.net/articles/authentication/. [Piekļūts 29.05.2021].

[13] JetBrains, «Databases - The State of Developer Ecosystem in 2020 Infographic,» 2020. [Tiešsaiste]. Pieejams: https://www.jetbrains.com/lp/devecosystem-2020/databases/. [Piekļūts 29.05.2021].

[14] Pallets, «Foreword - Flask Documentation,» [Tiešsaiste]. Pieejams: https://flask.palletsprojects.com/en/2.0.x/foreword/#what-does-micro-mean. [Piekļūts 29.05.2021].

[15] J. Au-Yeung un R. Donovan, «Best practices for REST API design,» 02.03.2020. [Tiešsaiste]. Pieejams: https://stackoverflow.blog/2020/03/02/best-practices-for-rest-api-design/. [Piekļūts 29.05.2021].

[16] W. Santos, «Which API Types and Architectural Styles are Most Used?,» 26.11.2017. [Tiešsaiste]. Pieejams: https://www.programmableweb.com/news/which-api-types-and-architectural-styles-are-most-used/research/2017/11/26. [Piekļūts 29.05.2021].

[17] «Introducting JSON,» [Tiešsaiste]. Pieejams: https://www.json.org/json-en.html. [Piekļūts 29.05.2021].

[18] Mozilla, «HTTP request methods,» 4.12.2020. [Tiešsaiste]. Pieejams: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods. [Piekļūts 29.05.2021].

[19] Mozilla, "HTTP response status codes," 19.05.2021. [Tiešsaiste]. Pieejams: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status. [Piekļūts 29.05.2021].

[20] «JSON:API - Latest Specification (v1.0),» [Tiešsaiste]. Pieejams: https://jsonapi.org/format/. [Piekļūts 29.05.2021].

[21] P. Lubbers un F. Greco, «HTML5 WebSocket: A Quantum Leap in Scalability for the Web,» [Tiešsaiste]. Pieejams: https://websocket.org/quantum.html. [Piekļūts 29.05.2021].

[22] Ably, «Long Polling - Concepts and Considerations,» [Tiešsaiste]. Pieejams: https://ably.com/topic/long-polling. [Piekļūts 29.05.2021].

[23] E. Bidelman, «Stream Updates with Server-Sent Events,» 30.11.2010. [Tiešsaiste]. Pieejams: https://www.html5rocks.com/en/tutorials/eventsource/basics/. [Piekļūts 29.05.2021].

[24] «Long-polling doesn’t totally suck,» 4.05.2013. [Tiešsaiste]. Pieejams: https://blog.fanout.io/2013/03/04/long-polling-doesnt-totally-suck/. [Piekļūts 29.05.2021].

[25] Internet Engineering Task Force (IETF), «Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP,» 2011. [Tiešsaiste]. Pieejams: https://datatracker.ietf.org/doc/html/rfc6202#section-5.1. [Piekļūts 29.05.2021].

[26] Mozilla, «Using server-sent events,» 04.04.2021. [Tiešsaiste]. Pieejams: https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events/Using_server-sent_events. [Piekļūts 29.05.2021].

[27] Mozilla, «The WebSocket API (WebSockets),» 25.05.2021. [Tiešsaiste]. Pieejams: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API. [Piekļūts 29.05.2021].

[28] M. O'Riordan, «Google - polling like it's the 90s,» 20.11.2019. [Tiešsaiste]. Pieejams: https://ably.com/blog/google-polling-like-its-the-90s. [Piekļūts 29.05.2021].

[29] OpenStack, «OpenStack Docs: Configure remote console access,» [Tiešsaiste]. Pieejams: https://docs.openstack.org/nova/latest/admin/remote-console-access.html. [Piekļūts 29.05.2021].

[30] The Olivetti & Oracle Research Lab, «VNC - How it works,» [Tiešsaiste]. Pieejams: http://web.mit.edu/cdsdev/src/howitworks.html. [Piekļūts 29.05.2021].

[31] Red Hat, Inc., «Spice for Newbies,» [Tiešsaiste]. Pieejams: https://www.spice-space.org/spice-for-newbies.html. [Piekļūts 29.05.2021].

[32] G. Turner un M. Komarinski, «Remote Serial Console HOWTO,» 31.05.2003. [Tiešsaiste]. Pieejams: https://tldp.org/HOWTO/Remote-Serial-Console-HOWTO/. [Piekļūts 29.05.2021].

[33] «Xterm.js,» [Tiešsaiste]. Pieejams: https://xtermjs.org/. [Piekļūts 29.05.2021].

[34] OpenStack, «Open Source Cloud Computing Platform Software,» [Tiešsaiste]. Pieejams: https://www.openstack.org/software/project-navigator/openstack-components#openstack-services. [Piekļūts 29.05.2021].

[35] OpenStack, «OpenStack Docs: Conceptual architecture,» [Tiešsaiste]. Pieejams: https://docs.openstack.org/install-guide/get-started-conceptual-architecture.html. [Piekļūts 29.05.2021].

[36] OpenStack, «OpenStack Docs: OpenStack Compute (nova),» [Tiešsaiste]. Pieejams: https://docs.openstack.org/nova/latest/. [Piekļūts 29.05.2021].

[37] OpenStack, «OpenStack Docs: Keystone, the OpenStack Identity Service,» [Tiešsaiste]. Pieejams: https://docs.openstack.org/keystone/latest/. [Piekļūts 29.05.2021].

[38] OpenStack, «OpenStack Docs: Welcome to Neutron's documentation!,» [Tiešsaiste]. Pieejams: https://docs.openstack.org/neutron/latest/. [Piekļūts 29.05.2021].

[39] OpenStack, «OpenStack Docs: Welcome to the Heat documentation!,» [Tiešsaiste]. Pieejams: https://docs.openstack.org/heat/latest/. [Piekļūts 29.05.2021].

[40] OpenStack, «OpenStack Docs: Orchestration Service API v1,» [Tiešsaiste]. Pieejams: https://docs.openstack.org/api-ref/orchestration/v1/index.html#general-api-information. [Piekļūts 29.05.2021].

[41] OpenStack, «OpenStack Docs: Template Guide,» [Tiešsaiste]. Pieejams: https://docs.openstack.org/heat/latest/template_guide/index.html. [Piekļūts 29.05.2021].

[42] OpenStack, «OpenStack Docs: DevStack,» [Tiešsaiste]. Pieejams: https://docs.openstack.org/devstack/latest/. [Piekļūts 29.05.2021].

[43] OpenStack, «OpenStack Docs: All-In-One Single Machine,» [Tiešsaiste]. Pieejams: https://docs.openstack.org/devstack/latest/guides/single-machine.html. [Piekļūts 29.05.2021].

[44] OpenStack, «OpenStack Docs: Heat and DevStack,» [Tiešsaiste]. Pieejams: https://docs.openstack.org/heat/latest/getting_started/on_devstack.html. [Piekļūts 29.05.2021].

[45] OpenStack, «OpenStack Docs: Compute service overview,» [Tiešsaiste]. Pieejams: https://docs.openstack.org/nova/latest/install/get-started-compute.html. [Piekļūts 29.05.2021].

[46] M. Makai, «WSGI Servers - Full Stack Python,» [Tiešsaiste]. Pieejams: https://www.fullstackpython.com/wsgi-servers.html. [Piekļūts 30.05.2021].

[47] T. Brown, «An Introduction to the Python Web Server Gateway Interface (WSGI),» [Tiešsaiste]. Pieejams: http://ivory.idyll.org/articles/wsgi-intro/what-is-wsgi.html. [Piekļūts 30.05.2021].

[48] Django Software Foundation, «How to deploy with WSGI,» [Tiešsaiste]. Pieejams: https://docs.djangoproject.com/en/3.2/howto/deployment/wsgi/. [Piekļūts 30.05.2021].

[49] D. Orehovsky, «What is WSGI and Why Do You Need Gunicorn and Nginx in Django,» 22 05 2021. [Tiešsaiste]. Pieejams: https://apirobot.me/posts/what-is-wsgi-and-why-do-you-need-gunicorn-and-nginx-in-django. [Piekļūts 30.05.2021].

[50] Gunicorn, «Design,» [Tiešsaiste]. Pieejams: https://docs.gunicorn.org/en/latest/design.html. [Piekļūts 30.05.2021].

[51] B. Hartley, «Scaling Your Web App 101: Lessons in Architecture Under Load,» 27.03.2018. [Tiešsaiste]. Pieejams: https://blog.hartleybrody.com/scale-load/. [Piekļūts 30.05.2021].

PIELIKUMI

1. pielikums – *DevStack* instalācijas testēšanas *lab_stack_example.yaml* topoloģijas apraksta fails

```yaml
heat_template_version: 2018-08-31

description: >
  ReSeLa test lab

resources:

    workstation_network:
        type: OS::Neutron::Net
        properties:
            name: workstation_network

    workstation_subnet:
        type: OS::Neutron::Subnet
        properties:
            name: workstation_subnet
            network_id: { get_resource: workstation_network }
            cidr: 10.0.0.0/24

    workstation_subnet_port_gateway:
        type: OS::Neutron::Port
        properties:
            name: workstation_subnet_port_gateway
            network_id: { get_resource: workstation_network }
            fixed_ips:
                - ip_address:  10.0.0.1

    workstation_subnet_port_1:
        type: OS::Neutron::Port
        properties:
            name: workstation_subnet_port_1
            network_id: { get_resource: workstation_network }
            fixed_ips:
                - ip_address:  10.0.0.50

    router:
        type: OS::Neutron::Router
        properties:
            name: router
            external_gateway_info:
                network: public

    router_workstation_subnet_port_interface:
        type: OS::Neutron::RouterInterface
        properties:
            router: { get_resource: router }
            port: { get_resource: workstation_subnet_port_gateway }
```

```
48
49      workstation_1:
50          type: OS::Nova::Server
51          properties:
52              image: cirros-0.5.2-x86_64-disk
53              name: workstation_1
54              flavor: m1.tiny
55              networks:
56                  - port: { get_resource: workstation_subnet_port_1 }
57              user_data: |
58                  /system identity set name=PC_1
59              user_data_format: RAW
60
```

2. pielikums – *react-polling* bibliotēkas pielietojums

```
93      const onPollSuccess = responseJSON => {
94          const state = responseJSON['data']['attributes']['build_state']
95          if (state === 'COMPLETE') {
96              console.log('lab is ready for: ' + courseId + ', ' + labId)
97              courseStore.markCourseLabBuildStateAsReady(labId)
98              return false
99          } else if (state === 'FAILED') {
100             console.log('lab is failed for: ' + courseId + ', ' + labId)
101             courseStore.markCourseLabBuildStateAsFailed(labId)
102             return false
103         }
104
105         console.log('lab is still building for: ' + courseId + ', ' + labId)
106         return true
107     }
108
109     const onPollError = error => {
110         console.log('error while polling for lab ' + courseId + ', ' + labId + ': ' + error)
111         courseStore.markCourseLabAsFailed(courseId, labId)
112     }
113
114     const actionButton =
115         labBuildState === 'IN_PROGRESS' ?
116             <ReactPolling url={statePollUrl} interval={5000} retryCount={1} onSuccess={onPollSuccess} onFailure={onPollError}
117                 render={({ startPolling, stopPolling, isPolling }) => {
118                     return (
119                         <BuildingButton />
120                     )
121                 }}
122             />
123         :
124         labBuildState === 'COMPLETE' ?
125                 <React.Fragment>
126                         <DeleteButton onClick={onDeleteClick} customStyles={classes.deleteButton} />
127                         <SuspendButton onClick={onSuspendClick} customStyles={classes.suspendButton} />
128                         <AvailableButton onClick={onAvailableClick} customStyles={classes.availableButton} />
129                 </React.Fragment>
130         :
131         <BuildButton onClick={onBuildClick} disabled={courseStore.builtLab ? true : false} />
```

3. pielikums – *background-task* bibliotēkas pielietojums

```python
@background(schedule=3)
def ping_for_lab_state(profile_id, course_id, lab_id, stack_id, state_on_complete):
    heat_client = get_heat_client()
    stack = heat_client.stacks.get(stack_id, resolve_outputs=True)
    stack_status = stack.status
    print('stack_status: ' + stack_status)

    if stack_status == 'IN_PROGRESS':
        ping_for_lab_state(profile_id, course_id, lab_id, stack_id, state_on_complete, verbose_name=stack_id)
    if stack_status != 'IN_PROGRESS':
        print('stack:')
        print(stack)

        assignment = Assignment.objects.get(profile=profile_id, course=course_id, lab=lab_id)

        try:
            outputs = getattr(stack, 'outputs')
            saved_outputs = {}
            for output in outputs:
                output_name = output['output_key']
                output_value = output['output_value']
                saved_outputs[output_name] = output_value
            assignment.stack_output_json = json.dumps(saved_outputs)
        except Exception:
            print('No outputs')

        assignment.build_state = state_on_complete
        if state_on_complete is None:
            assignment.stack_output_json = None
            assignment.stack_id = None
            assignment.stack_output_json = None

        assignment.save()
```

4. pielikums – kursu koka izvietojuma *JSON* apraksts

```
export const treesLayout = {
    "tutorialLab": "29",
    "trees": [
        {
            "1": {
                "styles": {
                    "borderColor": 'lightgreen'
                },
                "labs": [
                    {"labId": "1"},
                    {"labId": "2"},
                    {"labId": "3"},
                    {"labId": "4"},
                    {"labId": "5"},
                    {"labId": "6"},
                    {"labId": "7"},
                    {"labId": "8"}
                ]
            },
```

```
        "2": {
            "styles": {
                "borderColor": 'lightgreen'
            },
            "labs": [
                {"labId": "9"},
                {"labId": "10"},
                {"labId": "11"},
                {"labId": "12"}
            ]
        },
        "3": {
            "styles": {
                "borderColor": 'lightgreen'
            },
            "labs": [
                {"labId": "13"},
                {"labId": "14"},
                {"labId": "15"}
            ]
        }
    },
    {
        "5": {
            "styles": {
                "borderColor": 'orchid'
            },
            "labs": [
                {"labId": "19"},
                {"labId": "20"},
                {"labId": "21"},
                {"labId": "22"},
                {"labId": "23"}
            ]
        },
        "6": {
            "styles": {
                "borderColor": 'orchid'
            },
            "labs": [
                {"labId": "24"},
                {
                    "columns": [
                        [
                            {"labId": "25"},
                            {"labId": "30"}
                        ],
                        [
                            {"labId": "26"},
                            {"labId": "27"},
                            {"labId": "28"},
                            {"labId": "31"}
                        ]
                    ]
                }
            ]
        }
```

```
        },
        {
            "4": {
                "styles": {
                    "borderColor": 'lightblue'
                },
                "labs": [
                    {"labId": "16"},
                    {"labId": "17"},
                    {"labId": "18"}
                ]
            }
        }
    ]
}
```

5. pielikums – kursu koka savienojumu koordinātes aprēķināšanas algoritms

```
const EPSILON = 4;

const linkPathMaker = (p0, p1, linkProps) => {
        const { x: x0, y: y0, width: w0, height: h0 } = p0;
        const { x: x1, y: y1, width: w1, height: h1 } = p1;

        const [sx, sy] = [Math.floor(x0 + w0 / 2), Math.floor(y0 + h0 / 2)];
        const [tx, ty] = [Math.floor(x1 + w1 / 2), Math.floor(y1 + h1 / 2)];

        if (!(Math.abs(sx - tx) < 150)) {
                return  `M ${sx} ${sy + h0 / 2} L ${sx} ${sy + h0 / 2 + 35}` +
                                `M ${sx} ${sy + h0 / 2 + 35} Q ${tx} ${sy + (ty - sy) / 2} ${tx} ${ty - 35 - h1 / 2}` +
                                `M ${tx} ${ty - 35 - h1 / 2} L ${tx} ${ty}`
        }

        // same row
        if (Math.abs(sy - ty) < EPSILON) {
                const yOffset = -(tx - sx) / 3;
                return `M ${sx} ${sy} Q ${(sx + tx) / 2} ${sy + yOffset} ${tx} ${ty}`;
        }
        // same column
        if (Math.abs(sx - tx) < EPSILON) {
                return `M ${sx} ${sy} L ${tx} ${ty}`;
        }

        //return `M ${sx} ${sy} Q ${tx} ${sy + (ty - sy) / 2} ${tx} ${ty}`;
        return  `M ${sx} ${sy + h0 / 2} L ${sx} ${sy + h0 / 2 + 20}` +
                        `M ${sx} ${sy + h0 / 2 + 20} Q ${tx} ${sy + (ty - sy) / 2} ${tx} ${ty - 20 - h1 / 2}` +
                        `M ${tx} ${ty - 20 - h1 / 2} L ${tx} ${ty}`
};
```

6. pielikums – *DHCP* laboratorijas darba topoloģijas grafa *JSON* apraksta fragments

```
{
  "nodes": [
    {
      "data": {
        "class": "public",
        "id": "public",
        "image": "static/img/public.svg",
        "label": "public",
        "size": 30
      },
      "position": {
        "x": 345.0,
        "y": 250.0
      },
      "group": "nodes",
      "removed": false,
      "selected": false,
      "selectable": true,
      "locked": false,
      "grabbable": true,
      "classes": ""
    },
    ...
    ...
  ],
  "edges": [
    {
      "data": {
        "source": "public",
        "target": "router_1",
        "id": "public_edge",
        "port": "2"
      },
      "position": {},
      "group": "edges",
      "removed": false,
      "selected": false,
      "selectable": true,
      "locked": false,
      "grabbable": true,
      "classes": ""
    },
    ...
    ...
  ]
}
```

7. pielikums – *OpenStack Python* atbalsta bibliotēku pielietojums pirmkodā

```python
from keystoneauth1 import loading, session
from heatclient import client as heat_client
from novaclient import client as nova_client


def get_openstack_session():
    openstack_auth_loader = loading.get_plugin_loader('password')
    openstack_auth = openstack_auth_loader.load_from_options(
        auth_url='http://192.168.0.98/identity',
        username='demo',
        password='password',
        project_name='resela',
        user_domain_id='default',
        project_domain_id='default'
    )
    return session.Session(auth=openstack_auth)


def get_heat_client():
    openstack_session = get_openstack_session()
    return heat_client.Client('1', session=openstack_session)


def get_nova_client():
    openstack_session = get_openstack_session()
    return nova_client.Client('2.1', session=openstack_session)
```

8. pielikums – Jaunās platformas *DHCP* laboratorijas darba *JSON HOT* topoloģijas apraksts

```
{
  "files": {},
  "disable_rollback": true,
  "stack_name": "${assignment_id}",
  "template": {
    "heat_template_version": "2013-05-23",
    "description": "Simple template to test heat commands",
    "resources": {
      "workstation_network": {
        "type": "OS::Neutron::Net",
        "properties": {
          "name": "workstation_network_${assignment_id}"
        }
      },
      "server_network": {
        "type": "OS::Neutron::Net",
        "properties": {
          "name": "server_network_${assignment_id}"
        }
      },
      "workstation_subnet": {
        "type": "OS::Neutron::Subnet",
        "properties": {
          "name": "workstation_subnet",
          "network_id": { "get_resource": "workstation_network"},
          "cidr": "10.0.0.0/24"
        }
      },
      "server_subnet": {
        "type": "OS::Neutron::Subnet",
        "properties": {
          "name": "server_subnet",
          "network_id": {"get_resource": "server_network"},
          "cidr": "10.0.1.0/24"
        }
      },
      "workstation_subnet_port_gateway": {
        "type": "OS::Neutron::Port",
        "properties": {
          "name": "workstation_subnet_port_gateway",
          "network_id": {"get_resource": "workstation_network"},
          "fixed_ips": [{"ip_address": "10.0.0.1"}]
        }
      },
      "workstation_subnet_port_1": {
        "type": "OS::Neutron::Port",
        "properties": {
          "name": "workstation_subnet_port_1",
          "network_id": {"get_resource": "workstation_network"},
          "fixed_ips": [{"ip_address": "10.0.0.50"}]
        }
      },
      "workstation_subnet_port_2": {
        "type": "OS::Neutron::Port",
        "properties": {
          "name": "workstation_subnet_port_2",
          "network_id": {"get_resource": "workstation_network"},
```

```
        "fixed_ips": [{"ip_address": "10.0.0.51"}]
      }
    },
    "server_subnet_port_gateway": {
      "type": "OS::Neutron::Port",
      "properties": {
        "name": "server_subnet_port_gateway",
        "network_id": {"get_resource": "server_network"},
        "fixed_ips": [{"ip_address": "10.0.1.1"}]
      }
    },
    "server_subnet_port_1": {
      "type": "OS::Neutron::Port",
      "properties": {
        "name": "server_subnet_port_1",
        "network_id": {"get_resource": "server_network"},
        "fixed_ips": [{"ip_address": "10.0.1.40"}]
      }
    },
    "server_subnet_port_2": {
      "type": "OS::Neutron::Port",
      "properties": {
        "name": "server_subnet_port_2",
        "network_id": {"get_resource": "server_network"},
        "fixed_ips": [{"ip_address": "10.0.1.41"}]
      }
    },
    "router": {
      "type": "OS::Neutron::Router",
      "properties": {
        "name": "router_${assignment_id}",
        "external_gateway_info": {"network": "public"}
      }
    },
    "router_workstation_subnet_interface": {
      "type": "OS::Neutron::RouterInterface",
      "properties": {
        "router": {"get_resource": "router"},
        "port": {"get_resource": "workstation_subnet_port_gateway"}
      }
    },
    "router_server_subnet_interface": {
      "type": "OS::Neutron::RouterInterface",
      "properties": {
        "router": {"get_resource": "router"},
        "port": {"get_resource": "server_subnet_port_gateway"}
      }
    },
    "workstation_1": {
      "type": "OS::Nova::Server",
      "properties": {
        "name": "workstation_1",
        "image": "cirros-0.5.2-x86_64-disk",
        "flavor": "m1.tiny",
        "networks": [{"port": {"get_resource": "workstation_subnet_port_1"}}],
        "user_data": "/system identity set name=PC_1",
        "user_data_format": "RAW"
```

```
        }
      },
      "workstation_2": {
        "type": "OS::Nova::Server",
        "properties": {
          "name": "workstation_2",
          "image": "cirros-0.5.2-x86_64-disk",
          "flavor": "m1.tiny",
          "networks": [{"port": {"get_resource": "workstation_subnet_port_2"}}],
          "user_data": "/system identity set name=PC_2",
          "user_data_format": "RAW"
        }
      },
      "server_1": {
        "type": "OS::Nova::Server",
        "properties": {
          "name": "server_1",
          "image": "cirros-0.5.2-x86_64-disk",
          "flavor": "m1.tiny",
          "networks": [{"port": {"get_resource": "server_subnet_port_1"}}],
          "user_data": "/system identity set name=server_1",
          "user_data_format": "RAW"
        }
      },
      "server_2": {
        "type": "OS::Nova::Server",
        "properties": {
          "name": "server_1",
          "image": "cirros-0.5.2-x86_64-disk",
          "flavor": "m1.tiny",
          "networks": [{"port": {"get_resource": "server_subnet_port_2"}}],
          "user_data": "/system identity set name=server_2",
          "user_data_format": "RAW"
        }
      }
    },
    "outputs": {
      "workstation_1_uuid": {
        "value": {"get_resource": "workstation_1"}
      },
      "workstation_2_uuid": {
        "value": {"get_resource": "workstation_2"}
      },
      "server_1_uuid": {
        "value": {"get_resource": "server_1"}
      },
      "server_2_uuid": {
        "value": {"get_resource": "server_2"}
      }
    }
  },
  "timeout_mins": 60
}
```

Bakalaura darbs „Datortīklu mācību platformas pārveide” izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti.

Autors: Ralfs Eduards Braunfelds 31.05.2021.

Rekomendēju darbu aizstāvēšanai

Vadītājs: docents, Dr.sc.comp. Leo Trukšāns 31.05.2021.

Recenzents: profesors, Dr.sc.comp. Leo Seļāvo

Darbs iesniegts Datorikas fakultātē 31.05.2021.

Dekāna pilnvarotā persona: vecākā metodiķe Ārija Sproģe

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

__.06.2021. prot. Nr. ___.

Komisijas sekretārs: docents, Dr.sc.comp. Aivars Niedrītis